server 2008 – прерванный полет

крис касперски ака мыщъх, no-email

Server 2008 – последний 32-битный сервер в линейке NT. дальше будут только 64-битные версии. Microsoft сосредоточилась на x86-64 ядре, существенно переработав защитные механизмы, в то время как x86- и IA64-ядра не претерпели радикальных изменений в плане безопасности, а потому, главным образом мы будем говорить о x86-64 редакции, анализируя ее достоинства и недостатки с точки зрения хакеров, уже разломавших ядро в дизассемблере и расщепивших его на отдельные машинные команды в отладчике, благодаря чему удалось создать первые жизнеспособный exploit'ы под Server 2008 RC0.

введение
/ полоса 1, колонка 1 */*

Sever 2008 базируется на коде Вислы, доставшийся ей в наследство от Server 2003, причем, наибольшая разница наблюдается при переходе с Server 2003 на Вислу, дальнейшие изменения которой носят скорее косметическо-идеологический, чем технический характер. Именно Висла получила переписанный с нуля сетевой стек, именно в ней была впервые реализована рандомизация адресного пространства, продвинутая защита кучи и стека от переполнения, контроль целостности ядерных компонентов, обязательная цифровая подпись драйверов, а так же ряд других технологий, обещающих положить конец хакерству раз и навсегда (в который раз!). Действительно, атаковать x86-64 версии Server 2008 _намного_ сложнее, чем Server 2003, но безопасность не допускает компромиссов. Как нельзя быть "слегка беременной", так нельзя полагаться на "вполне" надежный сервер.

Что касается x86- и IA64-ядер, то они в целях обратной совместимости остаются незащищенными и допускают не только загрузку драйверов без цифровой подписи, но и позволяют модифицировать ядро, чем с успехом пользуются root-kit'ы для сокрытия своего присутствия в системе. Учитывая незначительную разницу в стоимости x86-64 и x86 процессоров, использовать 32-битные версии Server 2008 категорически не рекомендуется. Какой смысл вкладывать деньги в апгрейд системы, экономя на железе и получая тормозной и небезопасный сервер?! При всей своей симпатии к x86-архитектуре, автор вынужден признать, что ее дни сочтены, Microsoft во всю продвигает x86-64, а против воли Microsoft, как известно, не попрешь.

Но прежде, чем погрузиться в технические дебри, необходимо сделать одно маленькое, но очень существенное замечание. Традиционно x86-64 архитектура ассоциируется с корпорацией AMD, которую она выложила на алтарь в то время как Intel носилось со своей IA64, воплощенной в процессорах Itanium, предназначенных для high-end серверов. Формально, Microsoft поддерживает IA64, но… в плане защиты IA64-ядра ничуть не лучше x86, не говоря уже о том, что позиция IA64 на массовом рынке выглядит неуверенно и чтобы не терять потребителей, Intel скопировала архитектуру AMD x86-64 и реализовала ее в Pentium и Xenon процессорах, промаркированных загадочной аббревиатурой I32e/EM64T, которую многие аналитики с перепугу приняли за IA64. Но мы-то с вами знаем, что Intel I32e/EM64T это _практически_ тоже самое, что AMD x86-64, во всяком случае Server 2008 на Intel I32e/EM64T процессорах работает ничуть не хуже, чем на AMD x86-64! Вот такие, значит, маркетинговые премудрости.

.NET Framework и /GS [PRO]
/ полоса 1, колонка 2 */*

Ошибки типа переполнения буферов (ведущие к возможности удаленных атак) являются фундаментальной проблемой Си, неразрешимой в рамках самого языка. В несколько меньшей мере они свойственны Си++, но только при условии соблюдения всех заповедей безопасного программирования, а поскольку уровень программистов, работающих над созданием Windows, оставляет желать лучшего (не секрет, что большинство сотрудников Microsoft – китайцы и индусы), то ошибки переполнения неизбежны.

Для решения этой проблемы Microsoft разработала технологию .NET Framework фактически представляющую собой аналог виртуальной Java-машины, только: а) "заточенный"

под более узкий круг задач; б) идущий на компромисс между производительностью и безопасностью; в) укомплектованный большим количеством библиотек и заготовок, существенно ускоряющий разработку пользовательских интерфейсов и приложений, работающих с базами данных.

Компилятор поддерживает два режима: управляемого p-кода, выполняемого на виртуальной машине, и неуправляемого "нативного" (native) машинного кода, выполняемого на "живом" процессоре. В неуправляемом коде по умолчанию отсутствует контроль границ массивов и для предотвращения ошибок переполнения предусмотрен специальный ключ /GS, выполняющий принудительную проверку целостности стека при выходе из функции, в результате чего, максимум что может получить хакер – это банальный отказ в обслуживании, но вот захватить управление ему уже не удастся (во всяком случае теоретически).

В состав Sever 2008 входит .NET Framework 3.0, отныне (по заверениям Microsoft) являющаяся неотъемлемой частью системы, прочно интегрированной в ее нутро, что позволяет программистам писать сетевые приложения на C#, забыв об утечках памяти и ошибках переполнения как о страшном сне, в результате чего конечные пользователи получат более стабильную и безопасную систему, на которой можно реально работать, а не латать ее по несколько раз на день.

Естественно, сам по себе .NET Framework не делает систему ни лучше, ни хуже. Это делают использующие его приложения, как входящие в штатный комплект поставки, так и написанные сторонними разработчиками (конечно, при условии, что они используют .NET, а не какой-нибудь другую среду, например, DELPHI).

.NET Framework и /GS [Contra]
/ полоса 1, колонка 2 */*

Заявления об интеграции .NET Framework в Server 2008 несостоятельны. Ядро написано на смеси ассемблера, Си, Си++ и никто переписывать его не собирается. Базовые системные компоненты, написанные в "дотнетовскую" эпоху, так же не претерпели существенных изменений. Чтобы "отодрать" глубоко интегрированный .NET Framework от системы достаточно выбрать в меню инсталлятора режим Core, при котором мы получаем вполне работоспособный сервер, но…. только без .NET Framework'a. Оно и понятно — написать устойчивый и производительный код на .NET'е практически невозможно.

Ладно, а как обстоят дела с неуправляемым кодом? Увы, в угоду производительности, базовые системные компоненты откомпилированы _без_ использования ключа /GS, то есть ошибки переполнения никуда не делись. Правда, IE откомпилирован с ключом /GS, но держать IE на сервере может только начинающий администратор, остальные уже давно перешли на Горящего Лиса/Оперу/Рыся, но даже в этом случае, все необходимые файлы скачиваются на рабочую станцию и передаются на сервер через локальную сеть.

Но все-таки сосредоточимся на .NET Framework, а точнее на написанных на нем приложениях. В отличии от Java, язык C# допускает кучу вольностей, таящих в себе потенциальные опасности. К счастью, модель выделения памяти, используемая C#, тяготеет к размещению всех буферов в куче, а не в стеке (как на Си) и потому вместо стековых переполнений мы получаем переполнение кучи, разрушая служебные данные, используемые функцией освобождения памяти, которая "осведомлена" об этом типе атак и предотвращает захват управления. А вот сборщик мусора (garbage collector), встроенный в .NET Framework и занимающийся освобождением памяти, работает по небезопасной схеме и хотя Microsoft осведомлена о проблеме, в Server 2009 RC0 она до сих пор не исправлена и хакеры легко захватывают управление.

Кроме того, .NET Framework открывает новый, специфичный для него, класс атак, связанный со смешанным использованием MFC и ASCIIZ-сток. Функции, встроенные в .NET Framework, допускают присутствие нулевых символов в строке и потому при попытке определения расширения файла "my_file.exe\x00.txt" мы получим ".txt", а вот при передаче этой строке любой API-функции операционной системы (например, функции запуска ассоциированного с ним приложения) произойдет следующее: система отбрасывает все символы, стоящие за нулевым байтом, в результате чего имя файла оказывается усечено до "my_file.exe" и его открытие приводит к захвату управления системой. Анализ приложений, написанных на .NET Framework показал, что практически все они содержат ошибки подобного типа и разработчики спешно выпускают заплатки, выполняющие проверку всех строк, переданных серверу на наличие нулевых символов.

Другая проблема – р-код несет избыточную информацию о структуре классов, существенно упрощая процедуру декомпиляции, что в свою очередь ускоряет поиск дыр к огромной радости хакеров. В качестве контрмеры разработчики прибегают к запутываю (обфускации) кода, но во-первых, обфускация снижает производительность, а, во-вторых, существуют достаточно эффективные деобфускаторы, уже взятые на вооружение всеми хакерами и прочими злоумышленниками, ломающими программы.

Короче говоря, устраняя одни проблемы, Microsoft создает другие и в целом ситуация изменяется от плохой в строну еще более худшей.

в ядре [PRO]
/ полоса 2, колонка 1 */*

Зловредные программы, работающие на уровне ядра, обладают наивысшими привилегиями и могут эффективно противостоять защитным механизмам (типа антивирусов и брандмауэров), скрывая факт своего присутствия в системе. Чтобы проникнуть на уровень ядра, хакеру достаточно вызывать API-функцию **NtLoadDriver/ZwLoadDriver**, воспользовавшись дырой типа "ошибка переполнения буфера" или просто послав жертве программу от запуска которой она не сможет отказаться.

Частично эта проблема была решена еще во времена Windows 2000 путем введения цифровой подписи драйверов и при попытке инсталляции неподписанного драйвера система выводила угрожающее предупреждение (впрочем, функции **NtLoadDriver/ZwLoadDriver**) загружали драйвер и без него. В x86-64 версиях Вислы и Server 2008 цифровая подпись стала обязательной и неподписанный драйвер загрузить штатными средствами уже не удастся. Причем, в Server 2008 заложен механизм отзыва сертификатов, выданных нечестным на руку компаниям или просто украденных хакером.

Но даже если зловредному драйверу каким-то образом удастся проникнуть в ядро, ни атаковать антивирус/брандмауэр, ни перехватить системные функции он все равно не сможет, поскольку за целостью ядра и прилегающих к нему компонентами бдительно следит страж **PatchGuard**. Кстати говоря, перехватом системных функций занимаются не только хакеры, но и вполне легальные разработчики. Именно так устроены антивирусы, проверяющие файлы на лету, брандмауэры и многие другие защитные комплексы. Некорректно выполненный перехват (а корректный в живой природе пока что не замечен) приводит к критическим ошибкам и голубым экранам смерти. А пользователи во всем винят Microsoft, которая, устав отдуваться за чужие грехи, просто запретила лапать ядро грязными руками.

Для себя же любимой она оставила лазейку известную под кодовым именем **hot-patch** — в начале каждой функции размещается незначащая машинная команда, специально предназначенная для внедрения перехватчика, благодаря которому наложение заплаток на ядро уже не требует перезагрузки системы, а потому администратор может латать сервер сразу же после выхода очередного пакета обновления, не откладывая это дело до плановой перезагрузки. Раньше об этом приходилось только мечтать, но… это в теории все хорошо и прекрасно. А как обстоят дела на практике?

в ядре [CONTRA]
/ полоса 2, колонка 2 */*

Цифровая подпись — огромная головная боль для разработчиков, особенно выпускающих некоммерческое программное обеспечение и вынужденных платить за сертификат только потому, что так хочется Microsoft. Как следствие, на рынке появляются легальным образом подписанные драйвера, снабженные своим собственным загрузчиками, позволяющими грузить все драйвера без разбора, что очень практично и удобно.

Лаборатория Linchpin Labs (занимающаяся разработкой системных утилит для Windows) выпустила драйвер **Atsiv**, позволяющий загружать неподписанные драйвера, и не просто загружать, а еще и скрывать их присутствие в системе, что достигается путем исключения драйвера из списка PsLoadedModuleslist (точнее _не_ включения драйвера в список PsLoadedModuleslist, поскольку **Atsiv** использует собственный загрузчик), то есть **Atsiv** фактически представляет собой самый настоящий rootkit, образующий огромную дыру в безопасности. Сам-то он подписан, а вот загружаемые им драйвера — нет. Microsoft отреагировала вполне адекватно, "забанив" **Atsiv** в своем антивирусе Defender, встроенным в Server 2008, однако, сам факт появления **Atsiv'a** говорит о полной бесполезности цифровой подписи (к тому же, проверку цифровой подписи легко отключить, путем модификации файлов

NTOSKRNL.EXE и WINLOAD.EXE, правда, для этого потребуются права администратора и перезагрузка системы)

Кстати, об антивирусах и брандмауэрах. Они же ведь не от хорошей жизни занимаются перехватом системных функций. По другому надежную защиту никак не реализуешь. Самое интересное, что хакеру уже давно нашли пути обхода PatchGuard'a, однако, все они недостаточно надежны и зачастую рушат систему. То есть, зловредные программы могут хачить ядро, а честные программисты — нет. Фактически, Microsoft просто выталкивает неугодных ей игроков с рынка, усиливая свою монополию и подсаживая пользователей на свои (не самые лучшие) продукты. Замечательно, правда?

А если еще вспомнить технологию hot-patch, то вирусы вообще торжествуют, подменяя системные функции на свои собственные процедуры без перезагрузки системы. Но это еще что! Механизм Windows Update, обнаружив постороннюю "заплатку", просто отказывается устанавливать свою собственную, в результате чего автоматическое обновление перестает работать — администраторы просто рвут и мечут в экстазе от восторга.

 # картинки
/* полоса 2, колонка 3 */

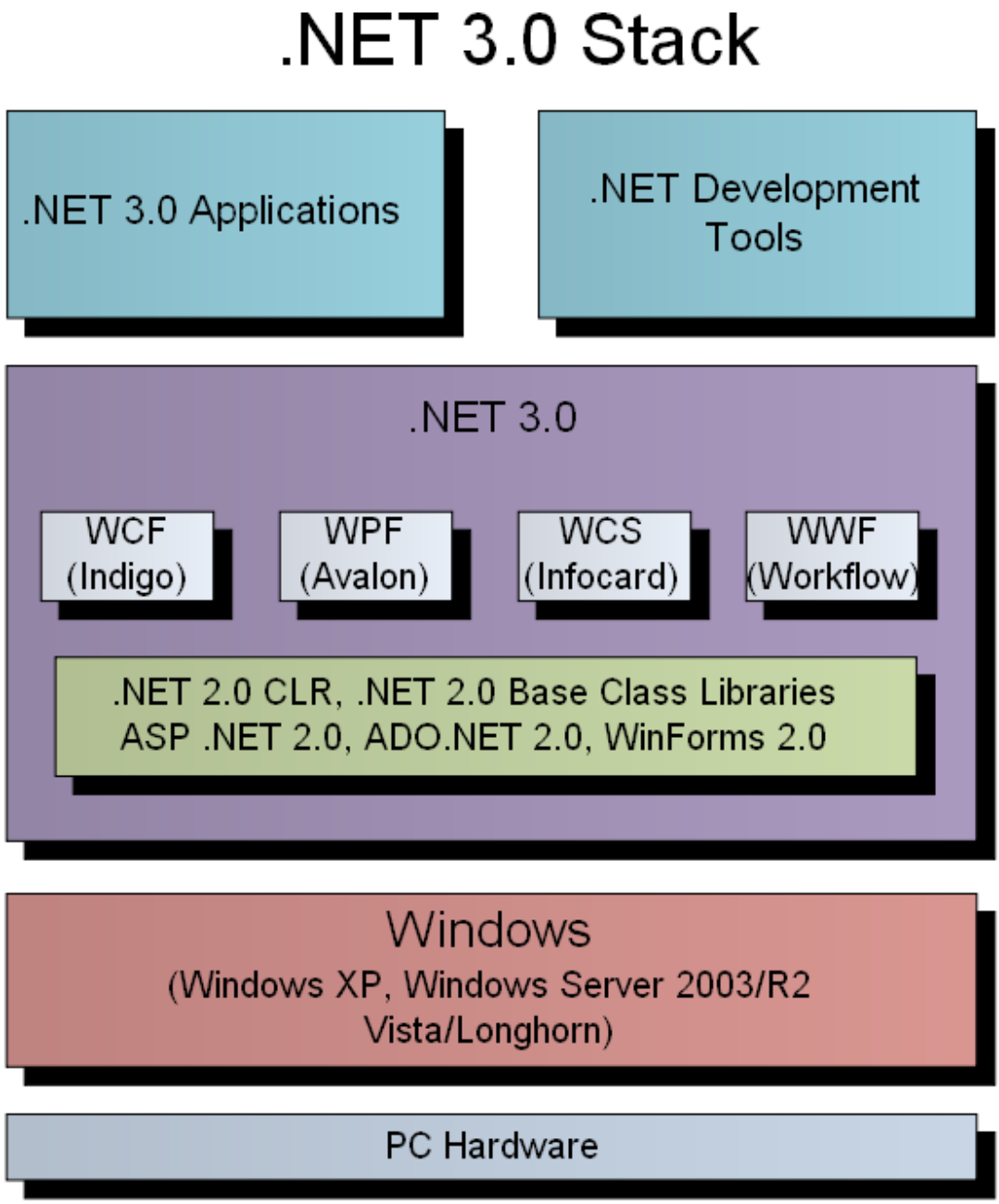

Рисунок 1 архитектура .NET Framework 3.0

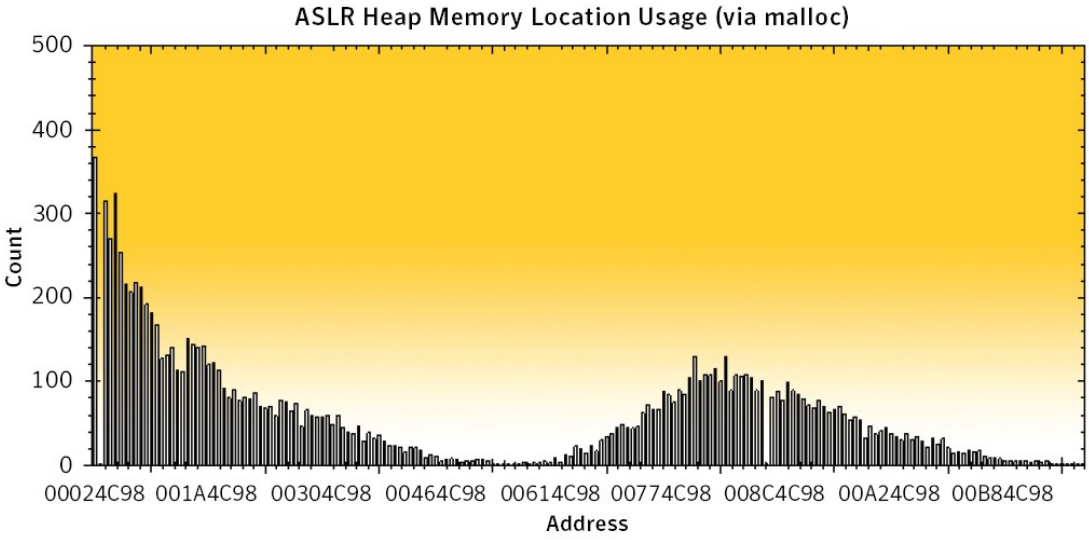

Рисунок 2 оценка качества работы рандомизатора Server 2008, как видно, адреса совсем не случайны и вполне хорошо предсказуемы

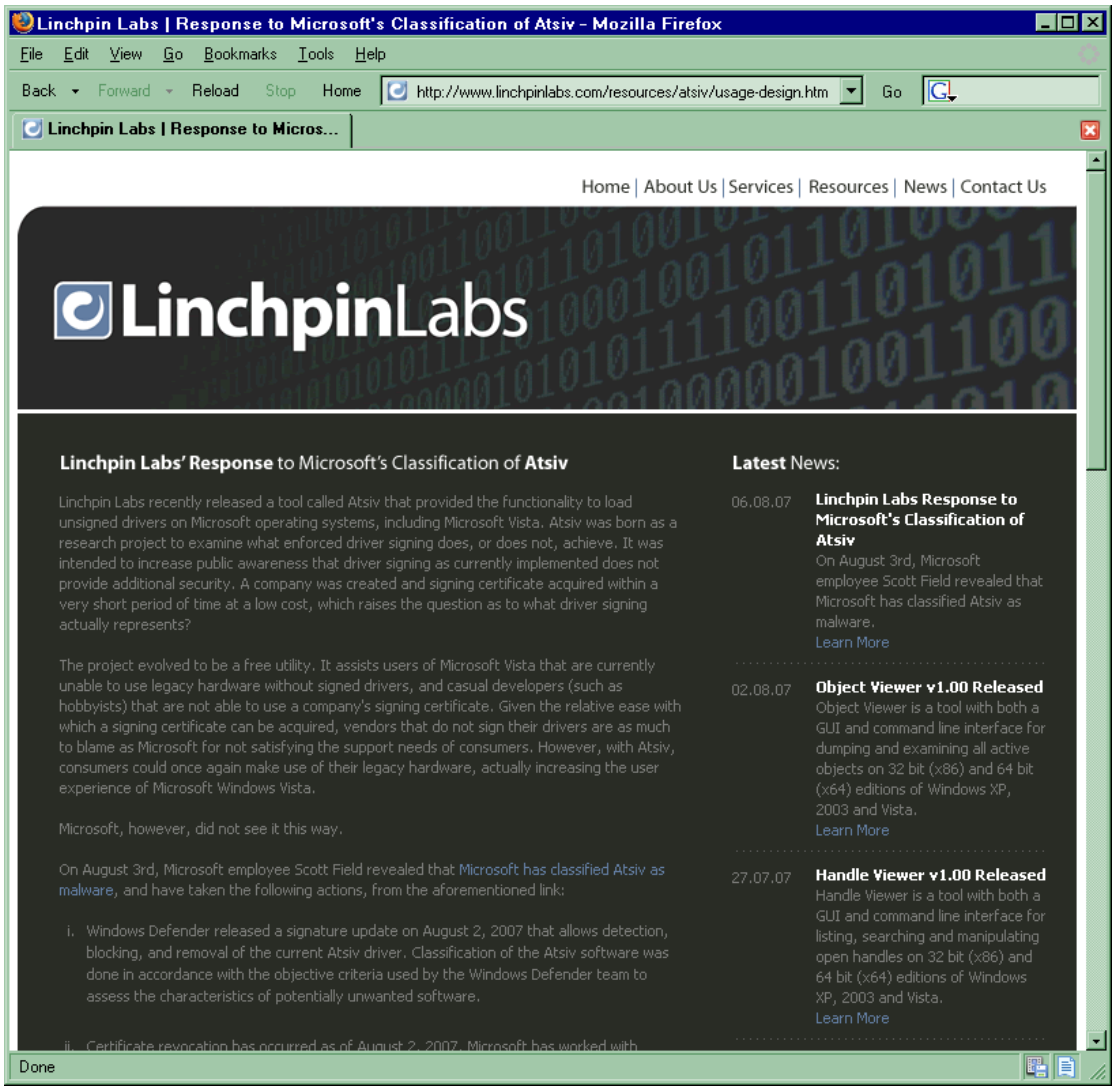

Рисунок 3 отсюда можно скачать драйвер для Server 2008, позволяющий загружать неподписанные драйвера

user-land [PRO]
/ полоса 3, колонка 1 */*

Наряду с усилением защиты ядра, Microsoft так же усилила безопасность прикладных приложений, являющихся основным объектом атаки, особенно в свете инициативы смены драйверной модели, перетаскивающий значительную часть кода с уровня ядра на прикладной уровень. Сетевые службы так же работают на прикладном уровне и нуждаются в защите.

Рассмотрим три ключевых технологии, выводящих Server 2008 на принципиально новый уровень. Это рандомизация адресного пространства, защита кучи от переполнения и блокировка сырой записи на неразмонтированный том.

Рандомизация адресного пространства (Address space layout randomization или, сокращенно, ASLR) пришла в Windows из мира UNIX. В Linux, начиная с ядра, 2.6.20 она включена по умолчанию, что существенно осложняет атаки, основанные на переполнении. Ведь чтобы передавать управление на shell-код (расположенный в стеке или куче) необходимо предварительно вызвать API-функции операционной системы, присваивающие данному региону памяти атрибут "исполняемый", а чтобы это сделать, необходимо _заранее_ знать адреса этих функций. Если же они выбираются произвольным образом при загрузке операционной системы, то хакер вместо захвата управления получает крэш сервера, что, конечно, очень плохо, но все же не смертельно.

Защита кучи от переполнения представляет дополнительный уровень обороны, предотвращающий передачу управления на shell-код/API-функции даже если хакер каким-то мистическим образом угадает точный адрес их расположения в памяти. Однако, переполнение кучи (даже с учетом защиты) приводит к аварийному завершению приложения, а потому от DoS атак никоим образом не спасает.

Наконец, сырая (она же посекторная) запись на системный том или том, содержащий файл подкачки, позволяла хакерам обходить любые защиты, в том числе и защиту от цифровой подписи в ранних версиях Вислы, что с успехом продемонстрировала Жанна Рутковская в своей знаменитой презентации на BlackHat'e. Теперь же попытка вызова API-функции WriteFile будет возражать ошибку до тех пор, пока дисковый том не будет размонтирован, а размонтировать системный том нельзя, поскольку для этого пришлось бы закрыть все файлы, включая файл реестра, который всегда открыт и загружен в память.

В общем, Microsoft проделала большую работу, возбудив интерес со стороны хакеров. И что же мы получили в итоге?

user-land [PRO]
/ полоса 3, колонка 2 */*

А ничего! То есть все эти защитные технологии оказались полной профанаций. Начнем с широко разрекламированной рандомизации, загружающей системные библиотеки по псевдослучайным адресам, выбираемых из 256 возможных вариантов, откуда с неизбежностью следует, что успешная атака потребует в среднем 256/2 = 128 попыток, то есть меньше, чем ничего (правда, неуспешная атака привет к краху атакуемого приложения, поднимая администратора с постели на предмет его перезагрузки), тем не менее, если у нас имеется один миллион серверов, работающих под управлением Server 2008, то по меньшей мере 6.000 из них будут поражены при первой же итерации. Вот такая, с позволения сказать, х… хорошая защита.

Защита кучи от переполнения, действительно, представляет для хакеров серьезную проблему над решением которой бились лучшие умы и… в конце концов нашли решение, продемонстрированное на последней конференции BlackHat, правда, оно относилось к Висле, а не к Server 2008, но, поскольку, последний использует тот же самый аллактор, он может быть атакован без всякого изменения сценария. Возможно, в финальной версии Server 2008 защита будет усилена, но… навряд ли она сможет продолжаться сколь ни будь значительное время.

Что же касается "сырой" записи, то помимо WriteFile существует множество других методов низкоуровневой работы с диском (как документированных, так и нет). Прежде всего это интерфейс **SPTI** (SCSI Pass-Through Interface), присутствующий во всех NT-подобных системах и позволяющий посылать дисковым устройствам SCSI-команды, преобразуемые операционной системой в "нативные" команды данного устройства, в роли которого может выступать хоть "флешка", воткнутая в USB, хоть IDE-винт. Эксперимент показывает, что низкоуровневая запись на системный том через SPTI до сих пор не заблокирована (хоть и требует прав администратора). Существует так же большое количество недокументированных IOCTL-кодов. Например, следующие команды предназначены для "прямой" работы с IDE-

дисками: SCSIOP_ATA_PASSTHROUGH/IOCTL_IDE_PASS_THROUGH и они так же _не_ заблокированы (во всяком случае пока… ну а там хакеры что-нибудь придумают). Так же хочется вспомнить **ASPI**-драйвер от компании Adaptec, через который работают многие программы пишущие или копирующие CD/DVD. Это очень глючный драйвер и при определенных обстоятельствах он дает прямой доступ не только к оптическим приводам, но и к жестким дискам, причем, _без_ прав администратора (впрочем, и без всяких гарантий, что он вообще заработает).

заключение
/ полоса 3, колонка 3 */*

Попробуем подвести итог, старясь быть максимально объективными и… позитивными. Server 2008, действительно, отличается от своего предшественника Server 2003 и его защитные механизмы претерпели существенные изменения, но ведь и технологии атаки уже не те, что были пять лет тому назад. Хакеры не спят и непрерывно совершенствуют свой арсенал. К тому же, если раньше атаками занимались в основном студенты и школьники, демонстрирующие миру свою крутость, то теперь хакерство из хобби превратились в хорошо отлаженную бизнес-машину, проворачивающую огромные суммы денег.

Компании, занимающиеся консалтиногом по безопасности, растут как грибы после дождя, а главным условием их роста является постоянный приток свежих дыр, над поиском которых бьются лучшие умы планеты, выкладывая результаты своих исследований в открытый доступ, откуда их стягивают программисты среднего уровня, занимающиеся рассылкой спама "чужими руками", кражей электронных кошельков, номеров кредитных карт и т. д. Наконец, студентов и школьников тоже не стоит списывать со счетов, поскольку, всякие сообщения о крутости защиты действуют на них как красная тряпка на быка и они не успокоятся пока не взломают очередной "подарок" от Microsoft.

Server 2008 еще не успел выйти, а exploit'ы для него уже появились. Забавно, правда? Чтобы там ни утверждала Microsoft, и какие бы цифры, диаграммы и графики она ни приводила, Server 2008 будут атаковать, поскольку все защитные технологии, реализованные в нем, _уже_ взломаны, а на разработку новых у Microsoft просто физически не хватает времени. Поэтому, вопрос о целесообразности перехода на Server 2008 остается открытым. Давайте не будем забывать, что поддержка Server 2003 еще не прекращена (и навряд ли будет прекращена в обозримом будущем) и заплатки для него выходят с ничуть не худшей регулярностью, чем для Server 2008, а своевременно залатанный сервер атаковать практически невозможно (практически, потому что помимо известных дыр существуют так же и неизвестные, точнее известные, то только тем, кому удалось их обнаружить).

Следовательно, в практическом плане, Server 2008 и Server 2003 стоят на одной ступени и усиленная защищенность Server 2008 – это еще не повод для обновления системы.