exploits review
0Eh выпуск

крис касперски ака мыщъх, a.k.a. nezumi, a.k.a elraton, a.k.a. souriz, no-email

еще не началась массовая миграция леммингов на Вислу, а дыры в ней уже обнаруживаются косяками. наш сегодняшний обзор вновь посвящен этой операционной системе, которая, вопреки всем заверениям корпорации Microsoft намного более дырява, чем предшествующая ей Хрюша.

Рисунок 1 вот такая она эта загадочная и неповторимая Виста, которую возможно создать лишь однажды

удаленный обход Windows Firewall

brief: начиная с XP, в состав Windows входит некая пародия на брандмауэр а-ля Windows Firewall, блокирующая по умолчанию все локальные порты и без существенных изменений перекочевавшая в Висту. А там, в Висте, сетевой стек полностью переписан, заточена поддержка IPv6 с кучей тоннельных протоколов типа Teredo, инкапсулирующих IPv6 в IPv4/UDP, и высаживающих брандмауэры сторонних производителей на полную измену, поскольку, чтобы определить целевой IP-адрес и порт назначения необходимо раздербанить инкапсулировать пакет, декодируя его с учетом формата Teredo. О том, что Teredo представляет собой мощное орудие для "пробивания" уже существующих брандмауэров, всем специалистам и без того было известно (см. например, мыщъхиную статью "глубины и вершины сетевого стека висты", опубликованную в "хакере-спец"), но вот тот факт, что Teredo, разработанный Microsoft, обходит Windows Firewall, разработанный ей же, вызывал большой переполох — игнорируя настройки строенного брандмауэры хакеры получили возможность сканировать порты и устанавливать соединения с любыми сетевыми службами, используя штатные средства Висты (достаточно послать жертве письмо со

ссылкой на IPv6-адрес и Windows автоматически активирует протокол Teredo, который по умолчанию дезактивируется через 60 минут неактивности). Дыра обнаружена двумя сотрудниками корпорации Symantec – Jim'ом Hoagland'ом с Ollie Whitehouse и подробно описана на http://www.securityfocus.com/bid/24779, а так же в бюллетене безопасности от самой Microsoft, зарегистрированным под номером MS07-038 (http://www.microsoft.com/technet/security/Bulletin/MS07-038.mspx), впрочем, как обычно из последнего ничего понять невозможно, т.к. все технические подробности там жестоко покоцаны;

targets: уязвимость затрагивает Microsoft Windows Vista beta 1/2, Home Basic/Home Premium/Business/Enterprise/Ultimate, включая в том числе и x64-битные редакции. На Windows 2000/XP и Server 2003 эта угроза _не_ распространяется;

exploit: не требуется, для атаки достаточно воспользоваться любой утилитой, умеющей передавать и принимать IPv4 пакеты (например, netcat);

solution: поскольку, IPv6 все еще не получил большого распространения, то для предотвращения вторжения, протокол Teredo рекомендуется отключить, например, путем блокирования TCP-порта 5357 на внешнем брандмауэре. Менее радикальная мера сводится к установки заплатки безопасности KB935807, которую можно скачать с сервера Windows Update;

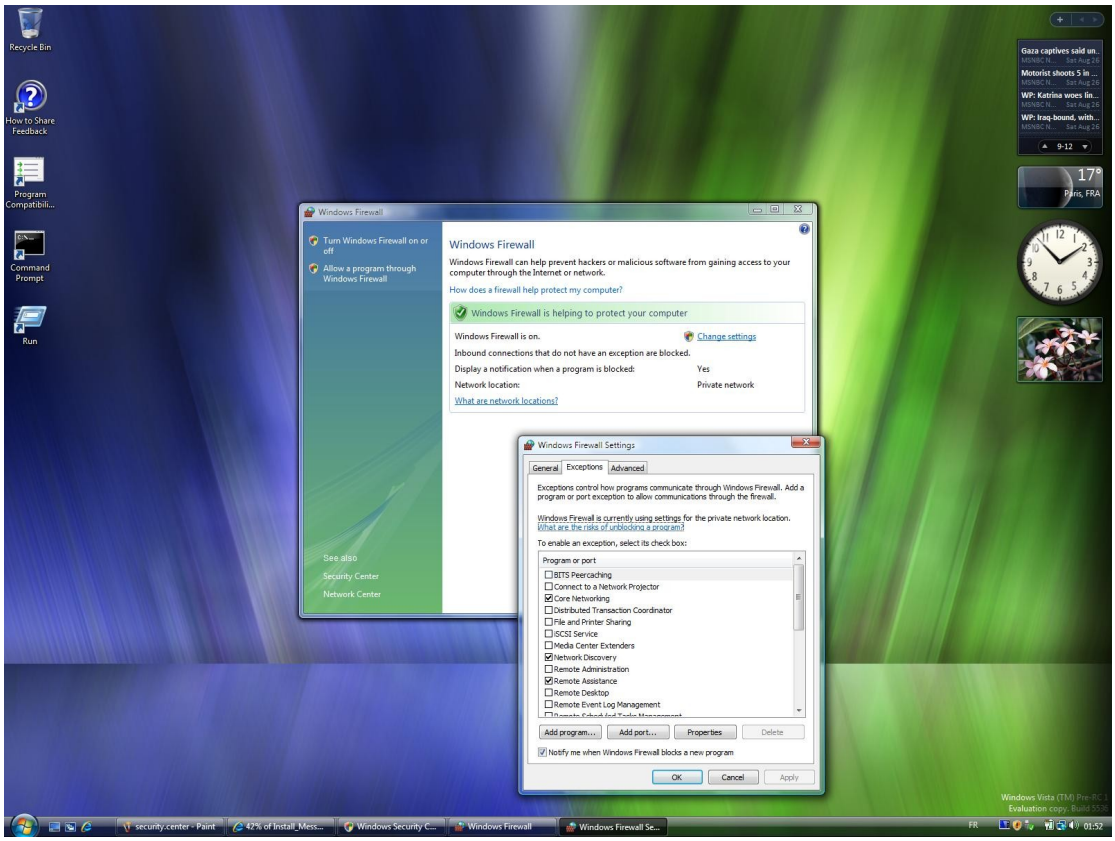

Рисунок 2 внешний вид Windows Firewall

удаленный обход защиты кучи от переполнения

brief: массовые переполнения кучи начались со статьи "Once upon a free()...", опубликованной в августовском выпуске электронного журнала "PHRACK" за 2001 год. Microsoft напрягалась и встроила в XP SP2 защиту от переполнения кучи, значительно усиленную в Висте, пробить которую удалось лишь совместными усилиями целой плеяды выдающихся хакеров: Brett'a Moore, Oded'a Horovitz'a, Matt'a Connover'a и нашего соотечественника Александра Сотирова. Сложность атаки объясняется тем, что

Виста не только проверяет целостность служебных данных кучи (так же называемых метаданными — metadata), но еще и шифрует их для надежности (кстати говоря, аналогичным образом постает и библиотека glib для Linux и xBSD). Впрочем, ключ шифровки и контрольная сумма лежат рядом с "подшефными" метаданными, где они могут быть перезаписаны хакером. Достаточно "всего лишь" разобраться с форматом служебных данных и "зеленый" билет у нас в кармане. На сингапурской конференции BlackHat-2007, Nicolas Waisman (nicolas@immunityinc.com) из исследовательской компании Immunityinc прочла доклад "Understanding and bypassing Windows Heap Protection" ("Понимание и обход защиты от переполнения кучи в Windows"), текст которого можно найти на http://www.immunityinc.com/downloads/Heap_Singapore_Jun_2007.pdf, а так же продемонстрировала хакерский инструментарий, созданный в лаборатории Immunityinc специально для атаки на кучу, распространяющийся на коммерческой основе: http://www.immunityinc.com/products-canvas.shtml;

targets: уязвимость затрагивает Microsoft Windows Vista beta 1/2, Home Basic/Home Premium/Business/Enterprise/Ultimate, включая в том числе и x64-битные редакции, а так же Windows 2000/XP/Server 2003;

exploit: для реализации атаки можно использовать библиотеку Immlib, созданную компанией Immunityinc Inc для своего же отладчика Immunityinc Debugger. Исходный текст демонстрационного скрипа, пробивающего защиту кучи от переполнения, приведен ниже:

```
fast = immlib.STDCALLFastLogHook( imm )
fast.logFunction( rtlallocate, 3)
fast.logRegister( "EAX" )
fast.logFunction( rtlfree, 3 )
fast.Hook()
```

Листинг 1 скрипт для Immunityinc Debugger, обходящий защиту кучи в Windows Виста от переполнения

solution: Microsoft делает вид, что ничего не происходит и никак не комментирует ситуации, поэтому для защиты кучи приходится прибегать к продуктам сторонних производителей, например, к коммерческому программному комплексу BufferShield, созданному на базе открытого проекта PaX (www.ngsec.com/ngproducts/stackdefender/). Ознакомительная 30-ти дневная версия BufferShield'a лежит на http://www.sys-manage.com/sites/D_BuffShld.html. Это действительно хорошее средство для защиты от атак, направленных на переполняющиеся буфера.

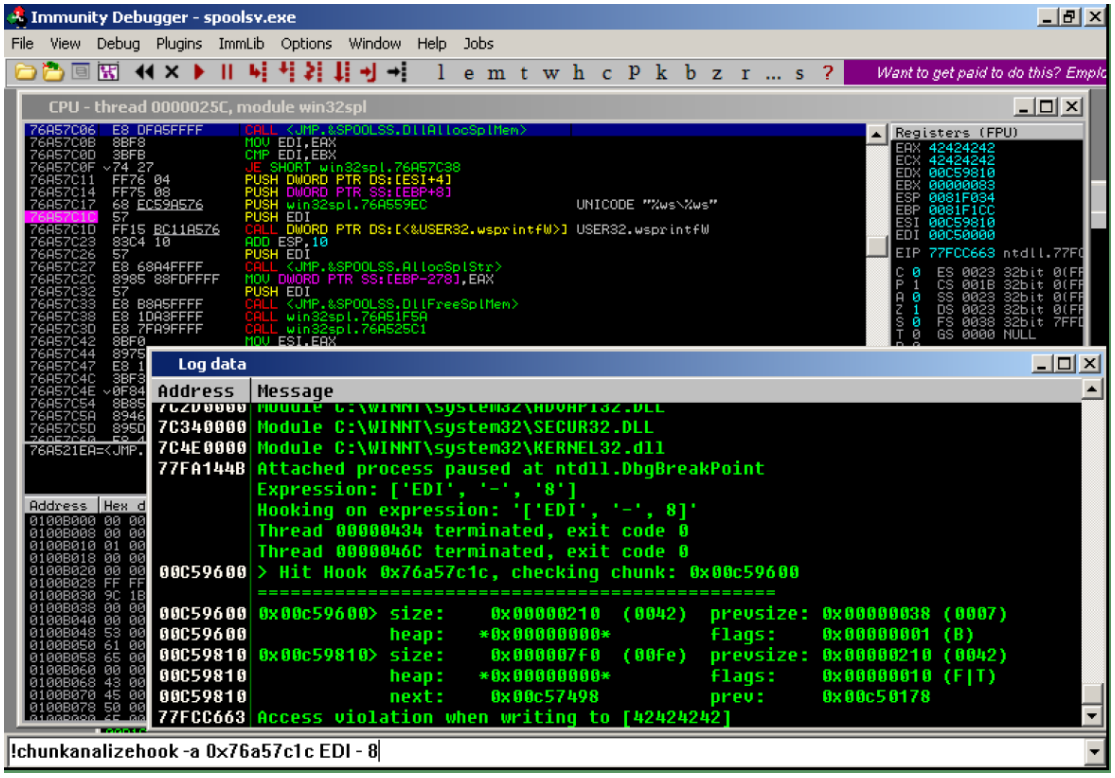

Рисунок 3 внешний вид отладчика коммерческого Immunityinc Debugger, подозрительно похожего на некоммерческий OllyDbg

локальная атака на ACL'ы

brief: в июне 2007 года хакер по имени Robbie Sohlman обратил внимание на "неаккуратную" установку прав в Висте, регламентирующих доступ к реестру и файловой системе, в результате чего непривилегированные пользователи могут беспрепятственно читать закрома защищенных хранилищ, в которых помимо прочих данных находится пароль администратора, захват которого приводит к известным последствиям. Хотя уязвимость носит локальный характер, ей могут пользоваться сетевые черви для повышения своих привилегий и установке root-kit'ов, скрывающих их от глаз антивирусных служб. Microsoft подсветилась и выпустила заплатку, доступную через службу Windows Update. Технические подробности содержаться в бюллетене по безопасности под номером MS07-032 (http://www.microsoft.com/technet/security/Bulletin/MS07-032.mspx) из которого ровным счетом понять ничего не возможно. Тоже самое относится и к прочим ресурсам по безопасности, например, к Security Focus: http://www.securityfocus.com/bid/24411, ограничившимся общими словами, но ничего не сказавшего по существу вопроса, так что мыщъх'у пришлось разбираться с проблемой самостоятельно. Анализ обновляемых файлов быстро выявил драйвер Fs_rec.sys, отвечающий за распознавание и автоматическое монтирование файловых систем, а так же ряд модулей и системных библиотек: imagehlp.dll (Библиотека для работы с PE-файлами), wmi.dll (Инструментарий Управления Windows – Windows Management Instrumentation) и upgclean.exe (Компонент Системы Обновления). По-видимому, ошибки, относящиеся непосредственно к данной уязвимости, кроются в Fs_rec.sys, а остальные измененные файлы просто фиксят мелкие посторонние баги. Беглое дизассемблирование показало, что он так же не при делах и реальную ответственность несет установщик Висты, устанавливающий слишком демократичные права на потенциально опасные файлы и ветви реестра, доступные штатными средствами операционной системы. В процессе наложения заплатки помимо обновления непричастных к уязвимости файлов, меняется политика прав доступа, что легко обнаружить сравнением образов файловой системы до и после "латания" дыры;

targets: уязвимость затрагивает Microsoft Windows Vista beta 1/2, Home Basic/Home Premium/Business/Enterprise/Ultimate, включая в том числе и x64-битные редакции. На Windows 2000/XP и Server 2003 эта угроза _не_ распространяется;

exploit: не требуется — атака реализуется штатными средствами доступа к файловой системе и реестру;

solution: установить заплатку, которую можно получить на узле Windows Update.

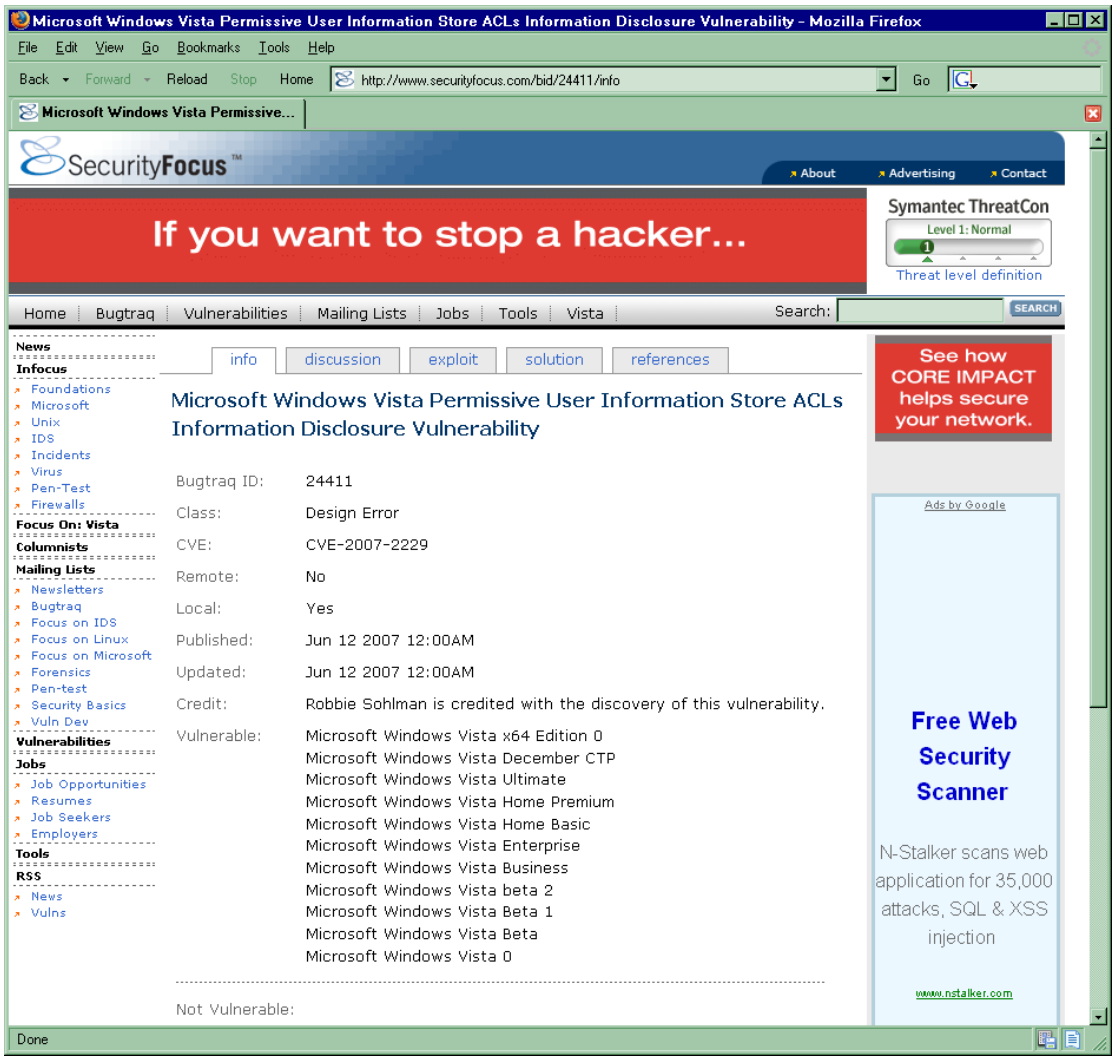

Рисунок 4 скудная информация о дыре в ACL'ах на сайте Security Focus

full disclose
каскад дыр в Microsoft .NET 2

Под влиянием агрессивной политики маркетингового отдела Microsoft платформа .NET продолжает свое неуклонное наступление на рынок, потеснив классический Си++ и некоторые другие языки (Visual Basic, Java, etc). NET-библиотеки по умолчанию входят во все Windows-системы, начиная с XP (владельцам NT и W2K приходится их скачивать отдельно), а потому представляют собой весьма соблазнительную мишень для атаки, благо там атаковать есть что. Сегодня мы рассмотрим как концептуальные уязвимости .NET'а, так и отдельные ошибки реализации, устраняемые путем установки соответствующей заплатки. Поскольку, .NET представляет собой кросс платформенную системно-независимую среду, абстрагирующуюся от конкретной архитектуры, то даже крошечная уязвимость автоматически распространяется на миллионы машин, работающих по всему миру — от серверов и рабочих станций до домашних компьютеров.

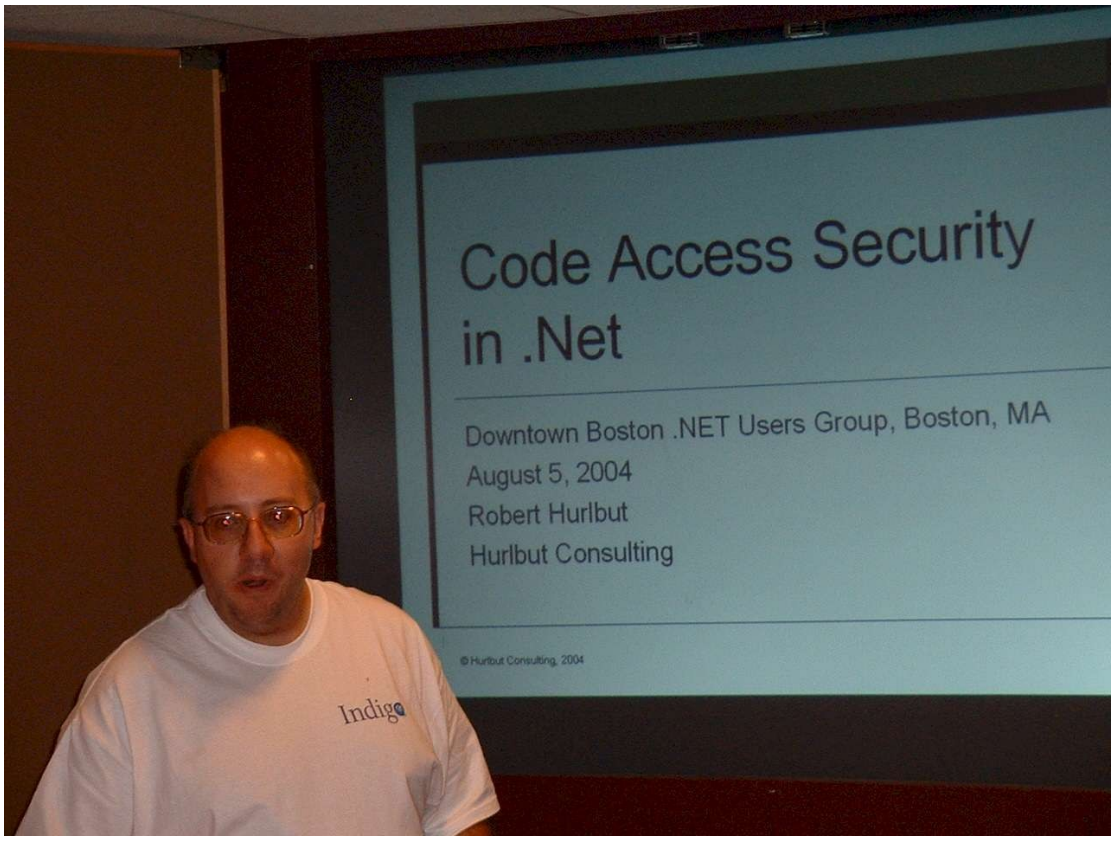

Рисунок 5 конференция по безопасности платформы .NET в Бостоне – одна из многих…

инъекция нулевых байт в текстовые строки

Язык .NET Common Language Runtime (сокращенно .NET CLR) поддерживает строки смешенного типа в MFC-стиле, трактующие символ нуля как обычный символ. В то же самое время, native-функции языка Си воспринимают нулевой байт как завершитель строки и потому при передаче CLR-строк native-Си функциям возникает противоречие, подробно описанное в предыдущем обзоре exploit'ов.

Но, если в нормальных языках ответственность за некорректное преобразование типа целиком лежит на совести программиста, вызывающего функции внешних библиотек без дополнительных проверок, то в случае с .NET мы имеем грабли даже при использовании встроенных методов, разработчики которых так и не смогли определиться какой строковой тип они должны поддерживать.

Возьмем, например, метод String.Compare, сравнивающего ASCIIZ-строки, и сопоставим его оператору "=", сравнивающего CLR-строки — налицо различие в поведении, наглядно демонстрируемое следующим ASP-приложением:

```
Sub Page_Load()
        dim allowed, sFirstItem, sSecondItem as string
        sFirstItem = Request("first")
        sSecondItem = Request("second")
        response.Write ("String.Compare - First item = " & sFirstItem & "<br>")
        response.Write ("String.Compare - Second item = " & sSecondItem & "<br>")
        if String.Compare(sFirstItem, sSecondItem) =0 then
                response.Write("String.Compare:Matched! Strs're the same"&"<br>")
        else
                response.Write("String.Compare:FAILED!Strs're not th esame"&"<br>")
        End If

        if sFirstItem=sSecondItem then
                response.Write ("Direct eval - Matched! Strings are the same" & "<br>")
        else
                response.Write("Direct eval - FAILED! Strings are not the same"&"<br>")
        End If
```

```
End Sub
```

Листинг 2 ASP-приложение, демонстрирующее разницу в поведении метода String.Compare и оператора "=" сравнивающих текстовые строки

Если передать приложению две полностью идентичных строки "test" (string.compare.demo.asp?first=test&second=test), оно выдаст вполне ожидаемый и совершенно законный результат:

```
String.Compare - First item = test
String.Compare - Second item = test
String.Compare - Matched! Strings are the same
Direct Eval - Matched! Strings are the same
```

Листинг 3 результат сравнения строки "test" со строкой "test" методом String.Compare и оператором "="

Но вот при передаче строк, несущих на своем борту символ нуля (например, "string.compare.demo.asp?first=test%00&second=test") ситуация изменится весьма радикальным образом. Оператор "=" покажет различие, а метод String.Compare – нет:

```
String.Compare - First item = test?
String.Compare - Second item = test
String.Compare - Matched! Strings are the same
Direct Eval - Failed! Strings are not the same
```

Листинг 4 результат сравнения строки "test%00" со строкой "test" методом String.Compare и оператором "="

Помимо String.Compare подобной болезнью страдают Server.MapPath, System.Net.Mail.SmtpMail.Send, Server.Execute, Server.Transfer и многие другие методы. Проблема затрагивает Microsoft .NET Framework 1.0, Framework 1.0 SP1/SP2/SP3, Framework 1.1, Framework 1.1 SP1 и Framework 2.0. Для ее устранения Microsoft выпустила пару заплаток KB928365 (для Framework 2.0) и KB928366 (для .NET Framework 1.1), однако их установка приводит к изменению поведения встроенных методов и в ряде случаев приводит к развалу ранее написанных приложений, которые так же приходится обновлять.

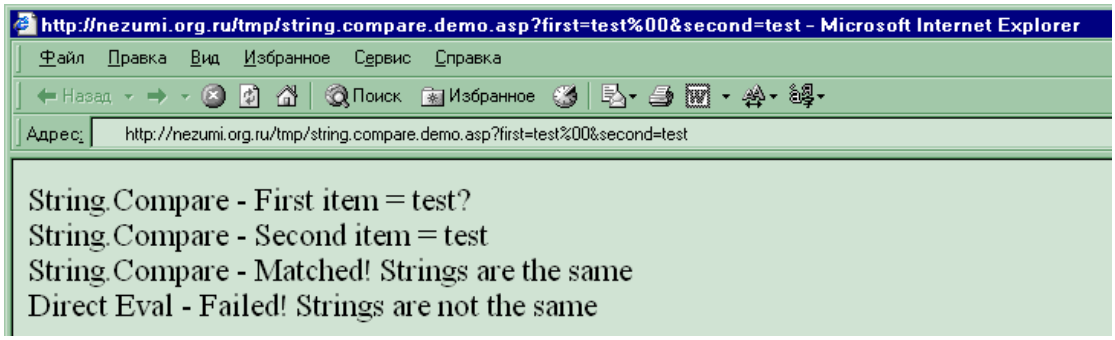

Рисунок 6 демонстрация уязвимости в .NET'е

множественные переполнения буфера в JIT-компиляторе

Платформа .NET (как и Java) представляет собой интерпретатор байт-кода, заметно уступающий в быстродействии "чистым" Си-компилятором и потому для сокращения разрыва в производительности используется техника компиляции байт-кода в память, за что отвечает JIT-компилятор. Сгенерированный им код работает в обход виртуальной машины, игнорируя многочисленные проверки, что отнюдь не идет на пользу безопасности. Ни одной фирме так и не удалось реализовать JIT-компилятор для языка Java, полностью свободный от ошибок, так что платформа .NET в этом смысле не исключение. Механизмы, обеспечивающие защиту буферов от переполнения, главным образом ориентированы на виртуальную машину, обрабатывающую байт-код и отсутствуют в машинной коде, сгенерированным JIT-компилятором (подробнее см. статью на Википедии: http://en.wikipedia.org/wiki/Just-in-time_compilation)

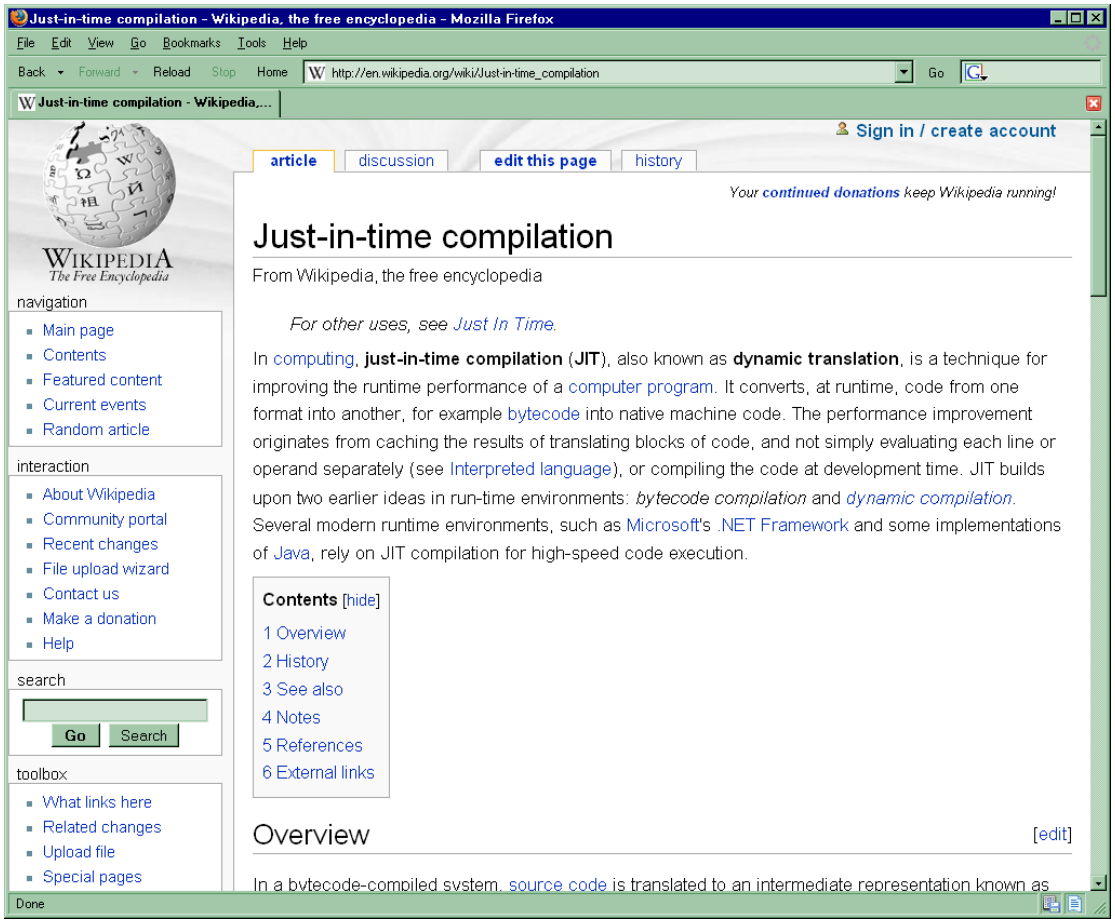

Рисунок 7 JIT-компиляторы на Википедии

Более того, JIT-компилятор так же является весьма соблазнительным объектом для атаки, поскольку содержит множество ошибок переполнения, обнаруженных исследователем по имени Jeroen Frijters из компании Sumatra. Оказалось, что JIT-компилятор всецело полагается на обрабатываемый им байт-код, наивно полагая, что он никем не изменен после трансляции. В Java по крайней мере имеется верификатор, тщательно проверяющий байт-код на соответствие спецификациям…

Для реализации DoS атаки достаточно подать на вход JIT-компилятора слегка подпорченный файл с байт-кодом. Так же возможен и удаленный захват управления во всяком случае теоретически. Практически — для этого потребуется обойти многочисленные защиты, предотвращающие выполнение машинного кода в неисполняемых областях памяти. В живой природе подобные exploit'ы отсутствуют… А жаль! Ведь дыре подвержены практически все операционные системы линейки NT – от W2K до Висты и хотя Microsoft уже выпустила соответствующие пакеты обновлений http://www.microsoft.com/downloads/details.aspx?FamilyId=BA3CEB78-8E1B-4C38-ADFD-E8BC95AE548D (для Windows 2000) и http://www.microsoft.com/downloads/details.aspx?FamilyId=CBC9F3CF-C3C3 -45C4-82E3-E11398BC2CD2 (для Вситы) далеко не все пользователи позаботились об их установке и количество потенциальных жертв просто огромно.

Дополнительные технически подробности можно найти в бюллетене безопасности MS07-040 от Microsoft: http://www.microsoft.com/technet/security/Bulletin/MS07-040.mspx, а так же на Security Focus'e: http://www.securityfocus.com/bid/24811

>>> врезка переполнение буфера в загрузчике PE-файлов

.NET трансляторы позволяют компилировать приложения в псевдо-исполняемые файлы формата PE. "Псевдо" — потому что байт-код, насильно засунутый в exe-файл, остается интерпретируемым байт-кодом и без .NET-платформы работать ни за что не будет, хоть бейся головой об стену, хоть убейся об газель, хоть грызи ее зубами (хинт: везде в маршрутках висит табличка с надписью "место для удара головой", о которое и предполагается убиться).

Собственно говоря, PE-формат представляет собой весьма продвинутый контейнер для различных типов данных, и чтобы не плодить сущности, Microsoft (в целях унификации) решила воспользоваться уже готовыми наработками, доверив это дело пионерам, допустившим в парсере PE-формата тучу ошибок, среди которых есть и фатальные — приводящие к переполнению с возможностью передачи управления на машинный код, что влечет за собой захват управления системой. И хотя угрозе присвоен статус "критической" (и она воздействует на все системы линейки NT, включая Висту), спешить с установкой заплаток нет никакой нужны — если у хакера есть возможность передать жертве исполняемый файл, который она запустит, то захват управления будет обеспечен и без переполнения. Исключение составляет, пожалуй, лишь CLR-код, передаваемый браузеру на выполнение. Тогда, чтобы подкинуть трояна жертве (или установить backdoor) достаточно заманить ее на специальным образом сконструированную Web-страничку.

Подробнее об этой дыре можно прочитать на Security Focus'e: http://www.securityfocus.com/bid/24778.

многочисленные ошибки фильтрации и SQL/HTML инъекции

Техника SQL/HTML-инъекций отработана уже давно и вполне успешно. Допустим, при входе на некоторый сайт пользователь вводит свое имя и пароль, которые web-скрипт сравнивает с эталонными данными, извлеченными из базы, взаимодействие с которой может быть организовано, например, так:

```
$result = mysql_db_query("database", "select * from userTable
        where login = '$userLogin' and password = '$userPassword' ");
```

Листинг 5 типичный пример запроса к SQL-базе

Здесь: $userlogin — переменная с именем пользователя, а $userPassword — пароль, подставляемые в строку SQL-запроса. Если в теле имени или пароля присутствует парная закрывающая кавычка, то его остаток интерпретируется как последовательность SQL-команд. Посмотрим, что произойдет, если вместо пароля ввести строку "fuck' or '1'= '1". А произойдет при этом следующее. Базе данных будет послан запрос "select * from userTable where login = 'KPNC' and password = 'fuck' or '1' = '1'", в результате чего кавычка, стоящая после fuck'a, закроет пользовательский пароль, а остаток строки превратится в логическое выражение, обрабатываемое базой данных. Поскольку очень трудно представить себе ситуации при которой один не был бы равен одному, то выражение всегда окажется истинным — вне зависимости от введенного пароля, в результате чего хакер сможет зайти на сайт под именем другого пользователя со всеми вытекающими отсюда последствиями.

Рисунок 8 ролик, демонстрирующий технику SQL-впрыска на YouTube: www.youtube.com/watch?v=MJNJjh4jORY

Для предотвращения SQL/HTML-инъекций необходимо осуществлять фильтрацию всего пользовательского ввода на предмет наличия недопустимых символов, но разработчики web-приложений люди ленивые и фильтрацию реализуют кое-как, спустя рукава, поэтому средства автоматической фильтрации, встроенные в платформу .NET, оказались большим подарком и завоевали всеобщую любовь и популярность. Между тем, качество автоматического фильтра оставляет желать лучшего и он содержит множество ошибок, затрагивающих в том числе и Microsoft .NET Framework 2.0, входящий в состав "неуязвимой" Висты.

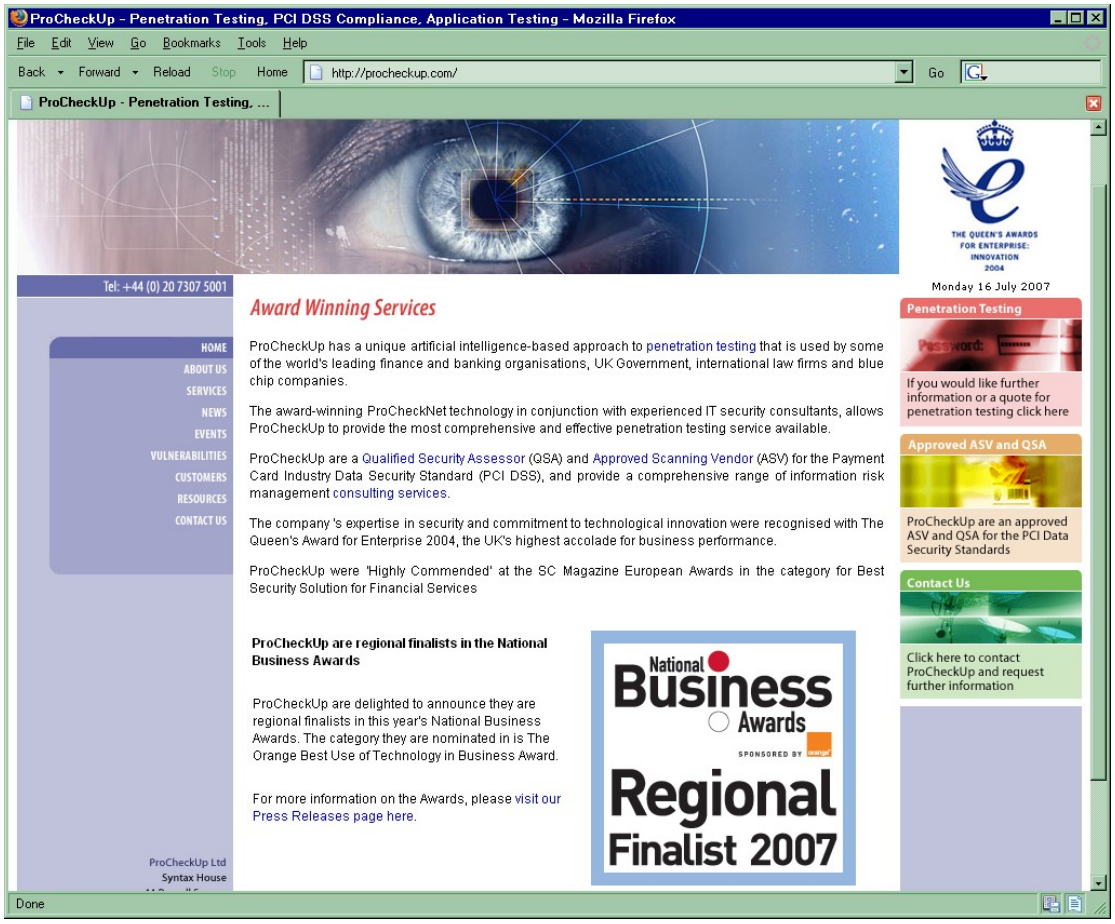

Рисунок 9 официальный сайт комании ProCheckUp

В частности, компания ProCheckUp (http://procheckup.com) обнаружила весьма "солидный" баг в системе управления контентом (.NET Content Management System или, сокращенно, .NET CMS), допускающим подстановку .NET-скриптов в поле 'lang' стандартного приложения 'logon.aspx', выполняемом в контексте уязвимого WEB-сайта, что позволяет реализовать атаку типа Cross Site Scripting путем отправки следущего GET-запроса:

```
GET /logon.aspx?lang=<SCRIPT>alert('Can%20Cross%20Site%20Attack')</SCRIPT> HTTP/1.1
Host: example.host.co.uk
Cookie: ASINFO=...; ASP.NET_SessionId=...; CNBOOK=...; ASPSESSIONIDSCDAQTST=...
Referer: http://example.host.co.uk:80/environ.pl
User-Agent: Mozilla/4.0 (compatible;MSIE 5.5;Windows NT 5.0;T312461;.NET CLR 1.0.3705)
Connection: close
```

Листинг 6 GET-запрос к logon.aspx, подставляющий .NET-скрипт в поле lang