Discover Flash Player Zero-day Attacks In The Wild From Big Data

Peter Pi
@heisecode

Agenda

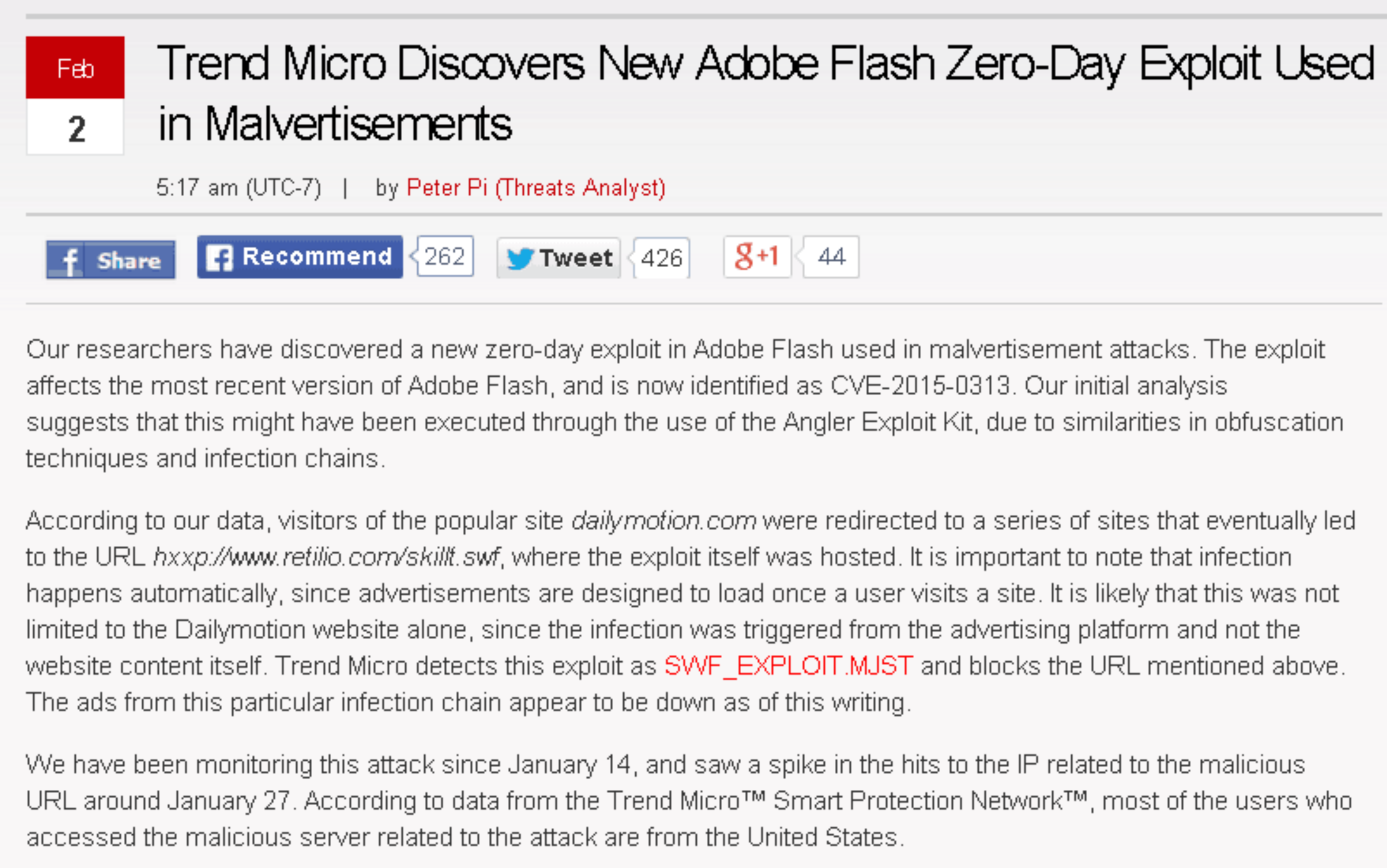
- Who am I
- Background
- Discover flash 0-day attacks from big set samples
- Vector Length mitigation

About me

- Security researcher
- APT product developer
- Interested in discovering vulnerabilities and writing exploit.
- Focus on Flash and Android recently.

WAR3 & Ping Pong Hobbyist

Found CVE-2015-0313 flash 0-day attack



Feb 2

Trend Micro Discovers New Adobe Flash Zero-Day Exploit Used in Malvertisements

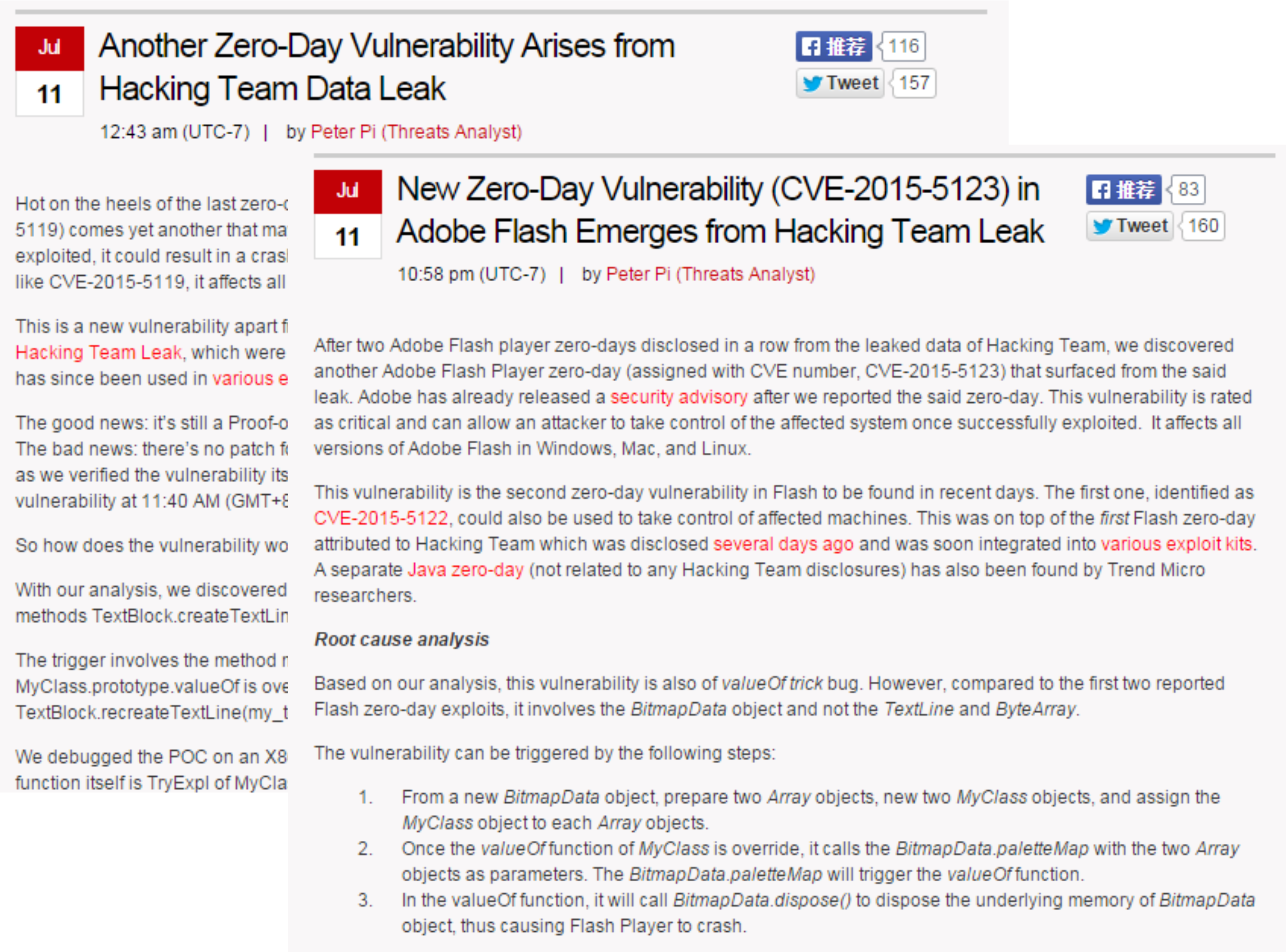
5:17 am (UTC-7) | by Peter Pi (Threats Analyst)

Our researchers have discovered a new zero-day exploit in Adobe Flash used in malvertisement attacks. The exploit affects the most recent version of Adobe Flash, and is now identified as CVE-2015-0313. Our initial analysis suggests that this might have been executed through the use of the Angler Exploit Kit, due to similarities in obfuscation techniques and infection chains.

According to our data, visitors of the popular site *dailymotion.com* were redirected to a series of sites that eventually led to the URL *hxxp://www.retilio.com/skillt.swf*, where the exploit itself was hosted. It is important to note that infection happens automatically, since advertisements are designed to load once a user visits a site. It is likely that this was not limited to the Dailymotion website alone, since the infection was triggered from the advertising platform and not the website content itself. Trend Micro detects this exploit as SWF_EXPLOIT.MJST and blocks the URL mentioned above. The ads from this particular infection chain appear to be down as of this writing.

We have been monitoring this attack since January 14, and saw a spike in the hits to the IP related to the malicious URL around January 27. According to data from the Trend Micro™ Smart Protection Network™, most of the users who accessed the malicious server related to the attack are from the United States.

CVE-2015-5122 & 5123 from hacked team



Jul 11

Another Zero-Day Vulnerability Arises from Hacking Team Data Leak

12:43 am (UTC-7) | by Peter Pi (Threats Analyst)

Hot on the heels of the last zero-(
5119) comes yet another that ma
exploited, it could result in a crasl
like CVE-2015-5119, it affects all

This is a new vulnerability apart fi
Hacking Team Leak, which were
has since been used in various e

The good news: it's still a Proof-o
The bad news: there's no patch fo
as we verified the vulnerability its
vulnerability at 11:40 AM (GMT+8

So how does the vulnerability wo

With our analysis, we discovered
methods TextBlock.createTextLin

The trigger involves the method r
MyClass.prototype.valueOf is ove
TextBlock.recreateTextLine(my_t

We debugged the POC on an X8
function itself is TryExpl of MyCla

Jul 11

New Zero-Day Vulnerability (CVE-2015-5123) in Adobe Flash Emerges from Hacking Team Leak

10:58 pm (UTC-7) | by Peter Pi (Threats Analyst)

After two Adobe Flash player zero-days disclosed in a row from the leaked data of Hacking Team, we discovered another Adobe Flash Player zero-day (assigned with CVE number, CVE-2015-5123) that surfaced from the said leak. Adobe has already released a security advisory after we reported the said zero-day. This vulnerability is rated as critical and can allow an attacker to take control of the affected system once successfully exploited. It affects all versions of Adobe Flash in Windows, Mac, and Linux.

This vulnerability is the second zero-day vulnerability in Flash to be found in recent days. The first one, identified as CVE-2015-5122, could also be used to take control of affected machines. This was on top of the *first* Flash zero-day attributed to Hacking Team which was disclosed several days ago and was soon integrated into various exploit kits. A separate Java zero-day (not related to any Hacking Team disclosures) has also been found by Trend Micro researchers.

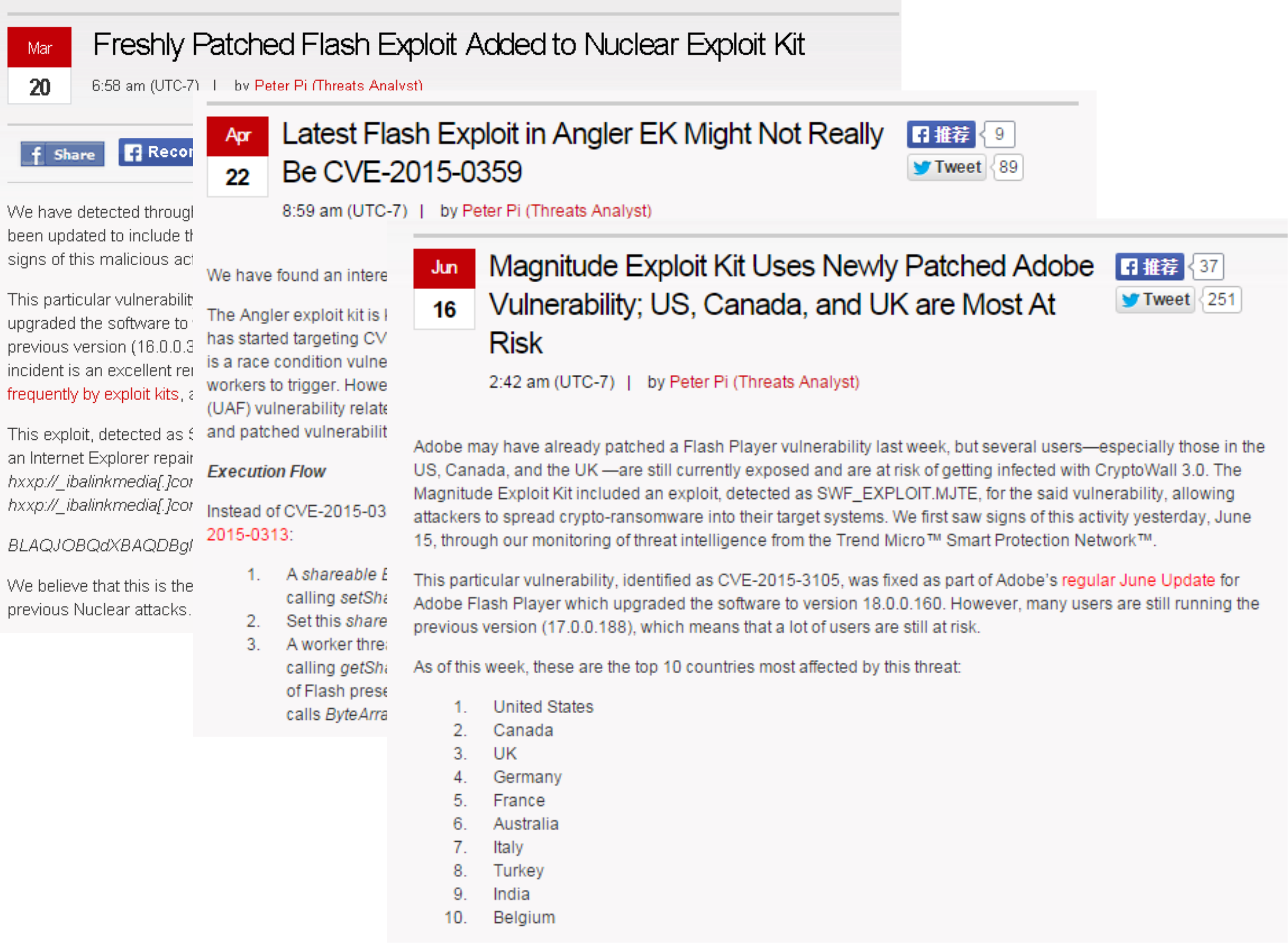
Root cause analysis

Based on our analysis, this vulnerability is also of *valueOf trick* bug. However, compared to the first two reported Flash zero-day exploits, it involves the *BitmapData* object and not the *TextLine* and *ByteArray*.

The vulnerability can be triggered by the following steps:

1. From a new *BitmapData* object, prepare two *Array* objects, new two *MyClass* objects, and assign the *MyClass* object to each *Array* objects.
2. Once the *valueOf* function of *MyClass* is override, it calls the *BitmapData.paletteMap* with the two *Array* objects as parameters. The *BitmapData.paletteMap* will trigger the *valueOf* function.
3. In the valueOf function, it will call *BitmapData.dispose()* to dispose the underlying memory of *BitmapData* object, thus causing Flash Player to crash.

Found some newly patched flash attack



Mar 20

Freshly Patched Flash Exploit Added to Nuclear Exploit Kit

6:58 am (UTC-7) | by Peter Pi (Threats Analyst)

We have detected througl been updated to include tl signs of this malicious act

This particular vulnerabilit upgraded the software to previous version (16.0.0.3 incident is an excellent rei frequently by exploit kits, a

This exploit, detected as S an Internet Explorer repair *hxxp://_ibalinkmedia[.]cor* *hxxp://_ibalinkmedia[.]cor*

BLAQJOBQdXBAQDBgl

We believe that this is the previous Nuclear attacks.

Apr 22

Latest Flash Exploit in Angler EK Might Not Really Be CVE-2015-0359

8:59 am (UTC-7) | by Peter Pi (Threats Analyst)

We have found an intere

The Angler exploit kit is I has started targeting CV is a race condition vulne workers to trigger. Howe (UAF) vulnerability relate and patched vulnerabilit

Execution Flow

Instead of CVE-2015-03 2015-0313:

1. A *shareable E* calling *setSha*
2. Set this *share*
3. A worker threa calling *getSha* of Flash prese calls *ByteArra*

Jun 16

Magnitude Exploit Kit Uses Newly Patched Adobe Vulnerability; US, Canada, and UK are Most At Risk

2:42 am (UTC-7) | by Peter Pi (Threats Analyst)

Adobe may have already patched a Flash Player vulnerability last week, but several users—especially those in the US, Canada, and the UK —are still currently exposed and are at risk of getting infected with CryptoWall 3.0. The Magnitude Exploit Kit included an exploit, detected as SWF_EXPLOIT.MJTE, for the said vulnerability, allowing attackers to spread crypto-ransomware into their target systems. We first saw signs of this activity yesterday, June 15, through our monitoring of threat intelligence from the Trend Micro™ Smart Protection Network™.

This particular vulnerability, identified as CVE-2015-3105, was fixed as part of Adobe's regular June Update for Adobe Flash Player which upgraded the software to version 18.0.0.160. However, many users are still running the previous version (17.0.0.188), which means that a lot of users are still at risk.

As of this week, these are the top 10 countries most affected by this threat:

1. United States
2. Canada
3. UK
4. Germany
5. France
6. Australia
7. Italy
8. Turkey
9. India
10. Belgium

My Blog address

- http://blog.trendmicro.com/trendlabs-security-intelligence/author/peterpi/
- Will publish some Android bugs I found.

Agenda

- Who am I
- Background
- Discover flash 0-day attacks from big set samples
- Vector Length mitigation

Flash Year

- Because of browsers’ UAF mitigations and JAVA pop-up window, Flash Player became the weakest out of popular targets in PC.

Flash Year

- Finally, we can see that Zero-day attacks' targets are mostly Flash Player in 2015

> CVE-2015-0310

> CVE-2015-0311

> CVE-2015-0313

> CVE-2015-3043

> CVE-2015-3113

> CVE-2015-5119

> CVE-2015-5122

Flash Year

- Newly patched N-day attacks in Exploit Kits this year almost are based on Flash Player vulnerabilities.

  > CVE-2014-8439

  > CVE-2014-9163 & CVE-2014-9162

  > CVE-2015-0336

  > CVE-2015-0359

  > CVE-2015-3090

  ….

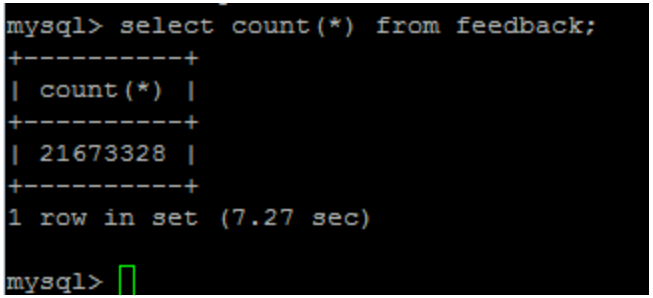
  ….

Flash Year

- In this situation, I wanted to disclose Flash 0-day attacks when tried to guess future perspective in late 2014.
- Disclose newly patched n-day attacks also has value to users.

Background

- Got tens of millions of suspicious SWFs in our Hadoop server, and thousands newly added every day.
- I think this is a good resource to find 0-day attacks
- So, this topic title's big data is a trick, and not related to data mining or machine learning ☺



```
mysql> select count(*) from feedback;
+----------+
| count(*) |
+----------+
| 21673328 |
+----------+
1 row in set (7.27 sec)

mysql>
```

Agenda

- Who am I
- Background
- Discover flash 0-day attacks from big set samples
- Vector Length mitigation

Problem I face

- Big set samples to handle.
- I need a automation process.
- It can achieve very low False Alert rate, fast processing speed.
- Final manual check only needs handle little Flash samples.

Need a tool

- I need a tool to help me identify a SWF file can exploit target version of Flash Player.

  > This tool must have very low False Alert.

  > This tool must have logger for improving automation.

  > This tool must can record exploit event when detect.

  > This tool must can stop the exploit.

FlashExploitDetector(FED)

- FED is an IE BHO written by C++
- Dynamic hook Flash OCX when Flash Player loaded to IE tab process.
- Hook IE event to get current URL name.
- Write log to file when detect, it will save the time and the SWF/URL name.
- Infinite loop when detect exploit, waiting for automation process to kill IE and continue next SWF file.

Automation Process

- Simple Python code.
- Register FED BHO using regsvr32.exe
- Every time load a HTML contains SWF in IE
- FED will hook Flash Player OCX to detect exploit
- Kill IE processes to load next SWF file in new IE
- When finished all SWF files, parse log file and get the detected SWF files.

Key Point

- How to achieve extremely low False Alert rate? There are match points in the flow of exploit.

1. Match vulnerability triggers? This means one vulnerability one rule, no use here, discard

2. Match Vector Heap Spray? This is good, but FA is still high for this special problem, for example old samples will trigger vector heap spray also. And 0-day may no need heap spray(CVE-2015-5119)

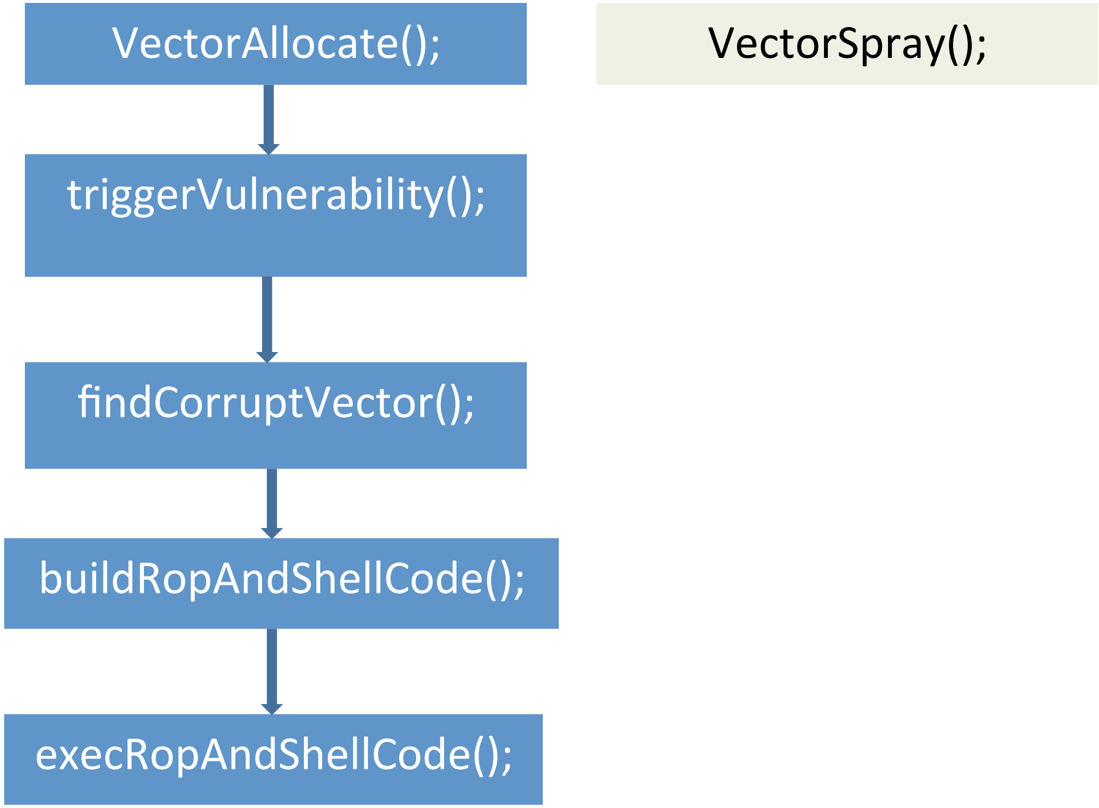
3. Match ROP and Shellcode execution stage? It is like EMET. But EMET is hard to automation, can't record the file name, 0-day may bypass EMET. And implement your EMET with a logger is big effort.

Key Point

- In 2014 and 2015, Flash Exploits are all use corrupt Vector to achieve arbitrary read and write memory.
- By achieved arbitrary read and write, exploits can bypass DEP, ALSR, CFG and even EMET.
- The corrupt Vector need huge length for reading and writing big memory address space of the process.
- May be I can match this generic point.

Key Point

- Simplified Exploit Flow



VectorAllocate();

VectorSpray();

triggerVulnerability();

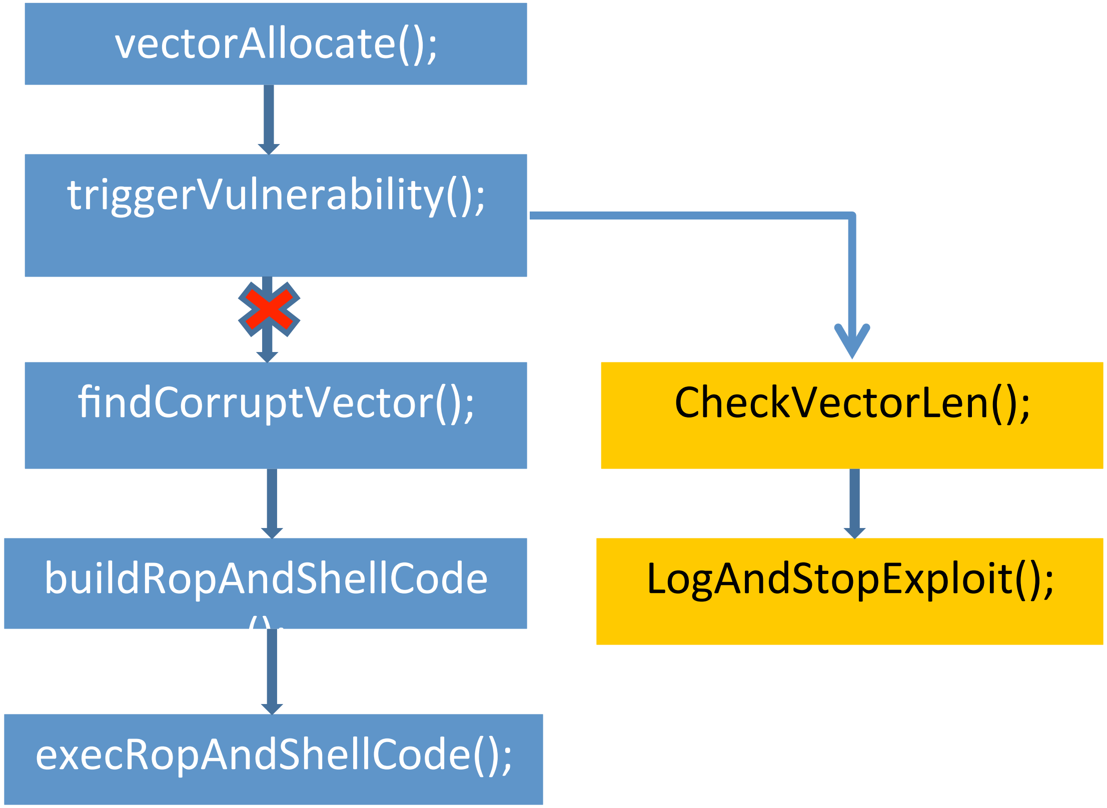
findCorruptVector();

buildRopAndShellCode();

execRopAndShellCode();

Key Point

- Ideally



How to implement?

- Because before AS3 methods been called, it will be JITed, So I hook the JIT flow of AVM2
- When hit the hook point, I can check the AS3 Vector status change between previous hit and this hit.
- So, this is likely check whether previous AS3 method has corrupt an AS3 Vector

How to implement it?

- Background knowledge

> AVM2 will JIT AS3 methods for performance.

> AVM2's verifier will check security when doing JIT

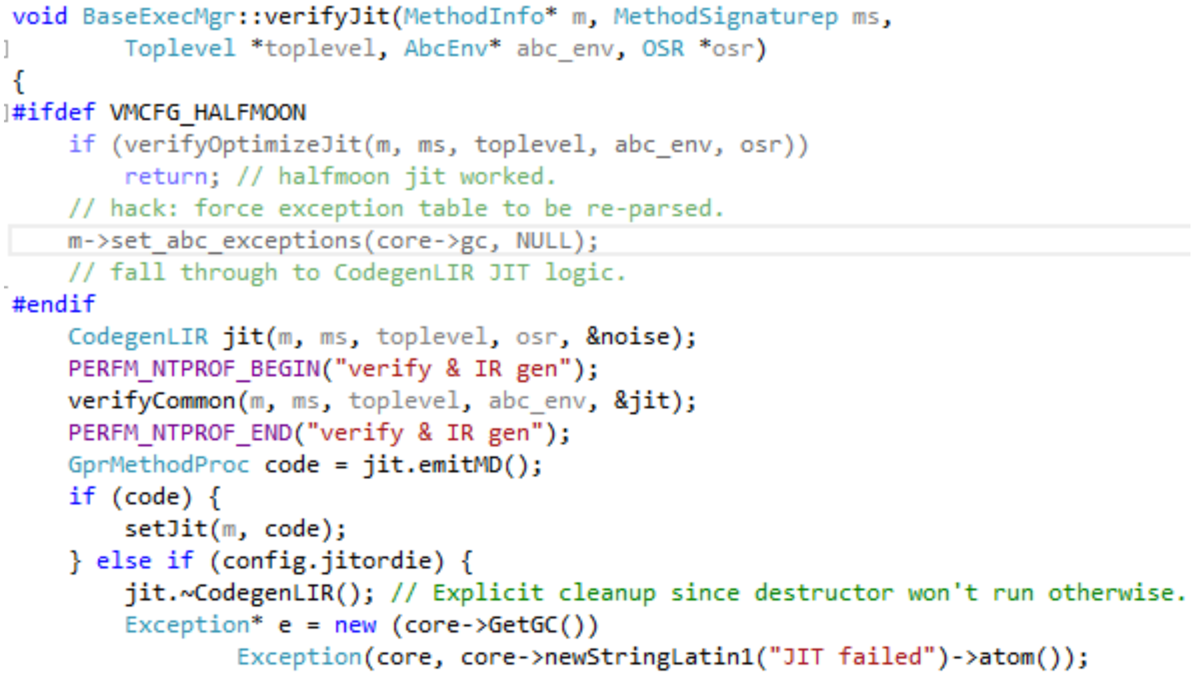
> After JIT, the emitted machine code address will be saved in a struct named MethodInfo.

> MethodInfo also saves a method id, uses method id we can get AS3 method name.

How to implement it?

- Key function

> In AVM2(https://github.com/adobe-flash/avmplus), BaseExecMgr::verifyJit is the function to verify and emit code.



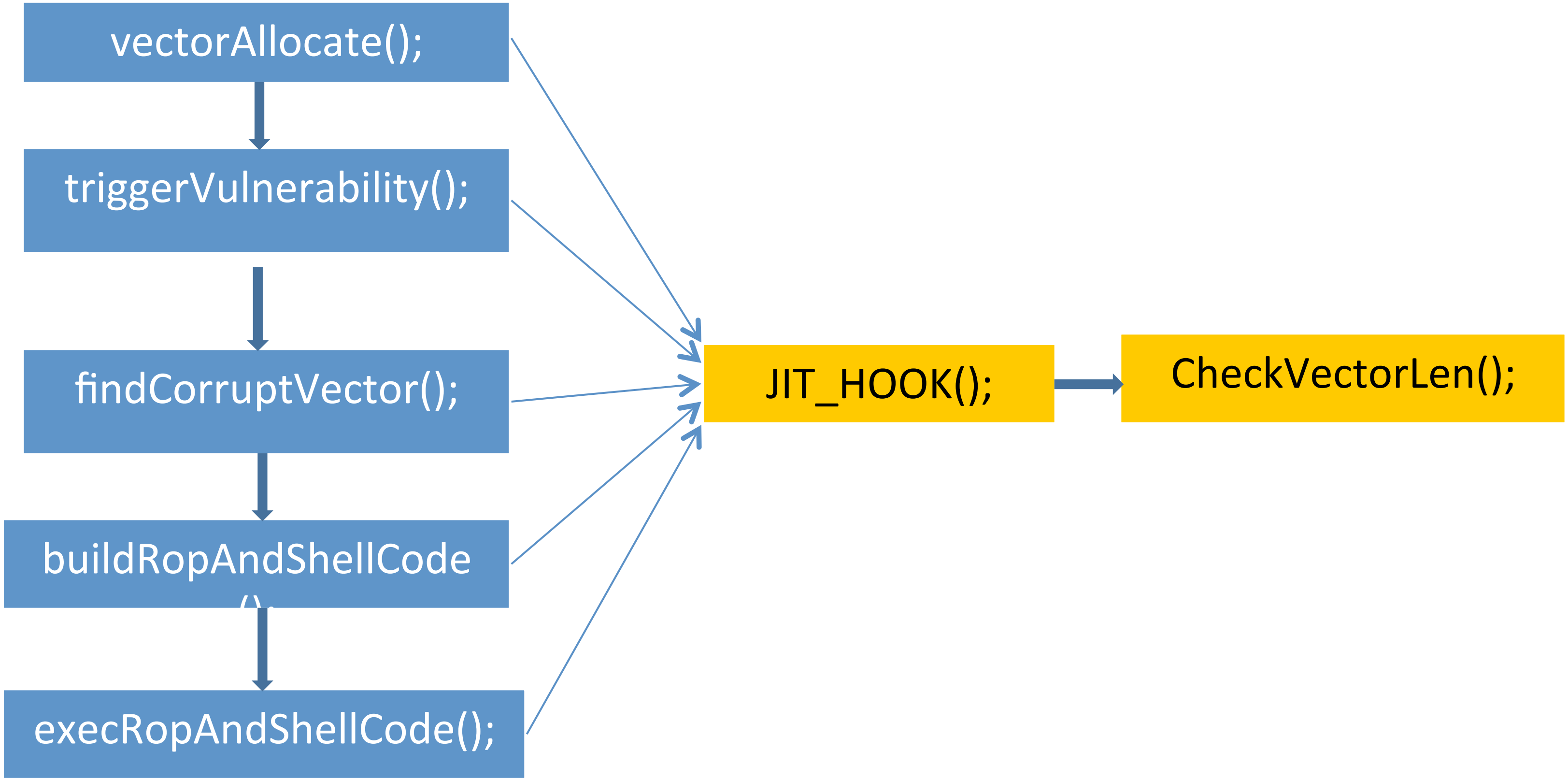
```
void BaseExecMgr::verifyJit(MethodInfo* m, MethodSignaturep ms,
        Toplevel *toplevel, AbcEnv* abc_env, OSR *osr)
{
#ifdef VMCFG_HALFMOON
    if (verifyOptimizeJit(m, ms, toplevel, abc_env, osr))
        return; // halfmoon jit worked.
    // hack: force exception table to be re-parsed.
    m->set_abc_exceptions(core->gc, NULL);
    // fall through to CodegenLIR JIT logic.
#endif
    CodegenLIR jit(m, ms, toplevel, osr, &noise);
    PERFM_NTPROF_BEGIN("verify & IR gen");
    verifyCommon(m, ms, toplevel, abc_env, &jit);
    PERFM_NTPROF_END("verify & IR gen");
    GprMethodProc code = jit.emitMD();
    if (code) {
        setJit(m, code);
    } else if (config.jitordie) {
        jit.~CodegenLIR(); // Explicit cleanup since destructor won't run otherwise.
        Exception* e = new (core->GetGC())
                Exception(core, core->newStringLatin1("JIT failed")->atom());
```

How to implement it?

- After hooked the JIT flow, we have chance to check the vector status in our JIT_HOOK function
- This means we can check vector has been corrupted or not after previous AS3 methods has been executed.

How to implement?

- So, Practically



vectorAllocate();

triggerVulnerability();

findCorruptVector();

buildRopAndShellCode
();

execRopAndShellCode();

JIT_HOOK();

CheckVectorLen();

How to check vector length?

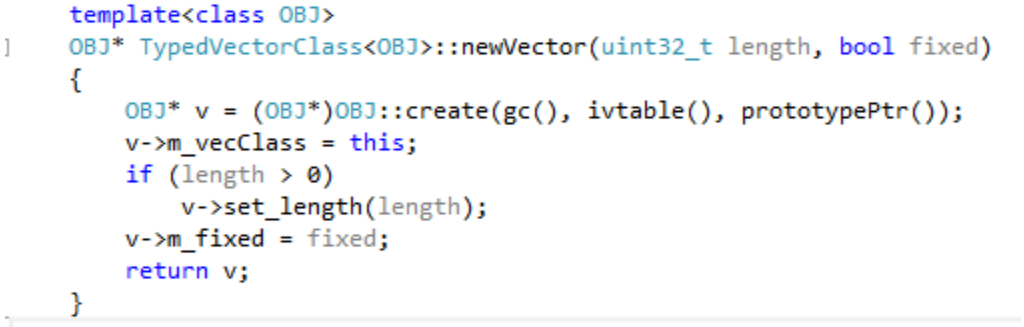
- Hook Vector Creating

1. Flash Player has 4 types AS3 Vector object.

2. Vector.<int>, Vector.<uint>, Vector.<Number> and Vector.<Object>.

3. I hook Vector.<int> and Vector.<uint> object create function.

4. In AVMplus source code, we can see the create function is a template function. Means that there are 4 instances in flash binary.



```
template<class OBJ>
OBJ* TypedVectorClass<OBJ>::newVector(uint32_t length, bool fixed)
{
    OBJ* v = (OBJ*)OBJ::create(gc(), ivtable(), prototypePtr());
    v->m_vecClass = this;
    if (length > 0)
        v->set_length(length);
    v->m_fixed = fixed;
    return v;
}
```

How to check vector length?

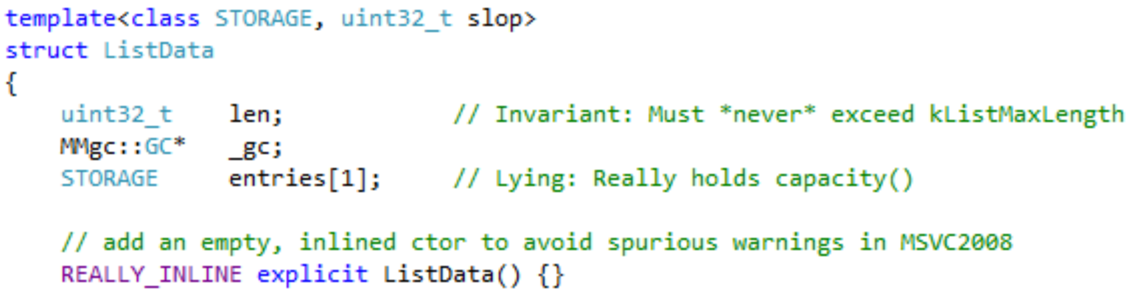
- Check Vector length

  > When there is a vector object created, I will save the vector object address.

  > vector_obj_addr + 0x18 is the data list which save vector data.

  > First 4 bytes of data list is the vector length.

  > So, poi(poi(vector_obj_addr + 0x18)) is vector length

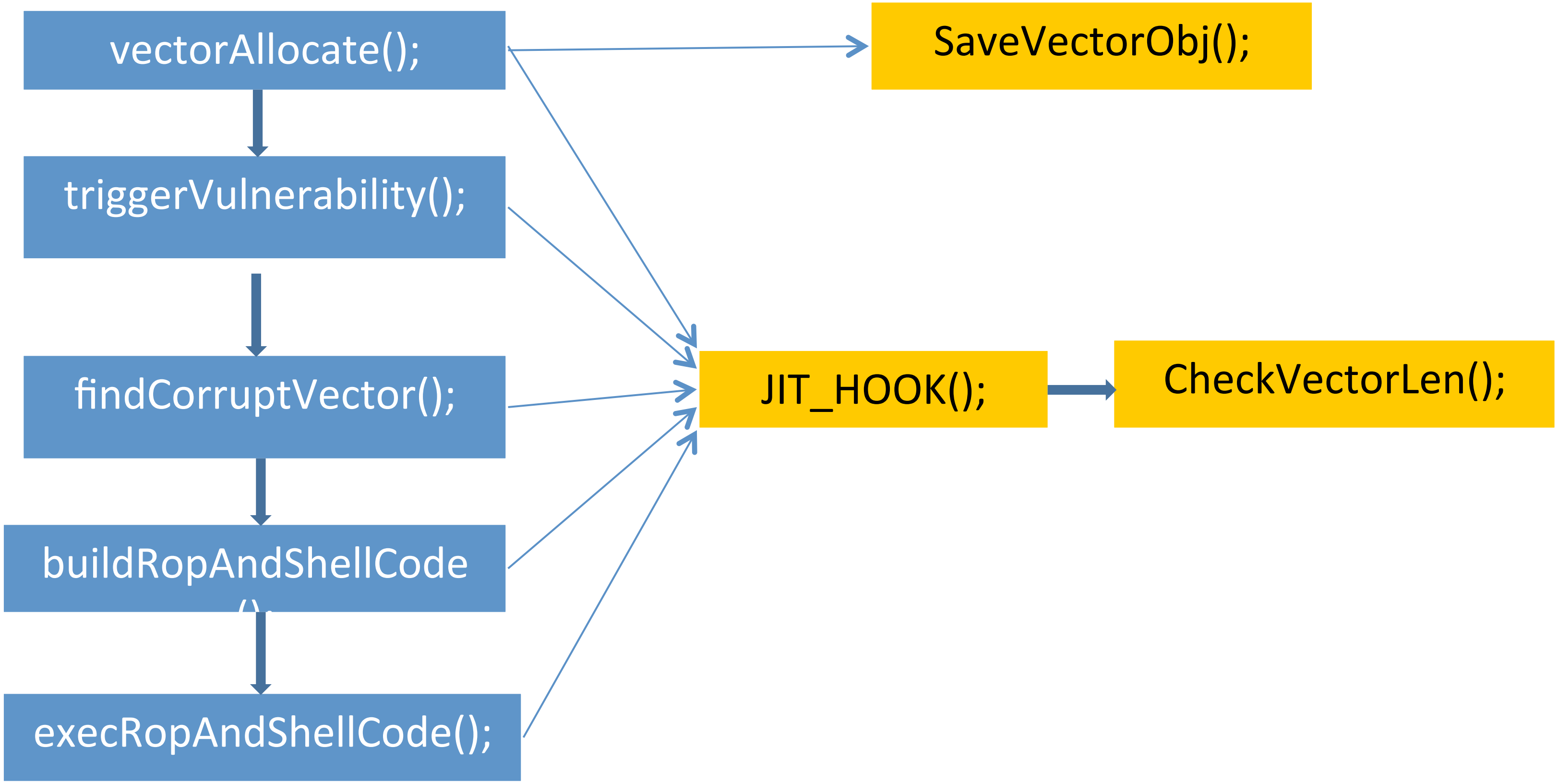


```
template<class STORAGE, uint32_t slop>
struct ListData
{
    uint32_t    len;            // Invariant: Must *never* exceed kListMaxLength
    MMgc::GC*   _gc;
    STORAGE     entries[1];     // Lying: Really holds capacity()

    // add an empty, inlined ctor to avoid spurious warnings in MSVC2008
    REALLY_INLINE explicit ListData() {}
```

How to implement?

- So, Practically



vectorAllocate();

triggerVulnerability();

findCorruptVector();

buildRopAndShellCode
();

execRopAndShellCode();

SaveVectorObj();

JIT_HOOK();

CheckVectorLen();

Hook Version

- Hook Version

  > Some sample check Flash Player version, if version is too high or too low, it will terminate execution.

  > So I change Flash Player version string in memory

  > For example, change WIN 18,0,0,160 to WIN 16,0,0,160

  > Just search WIN x,0,0,x in OCX image memory

How to Hook Flash OCX load?

- Need to hook Flash OCX when it being loaded first time.
- Like Windbg’s module load event
- Flash OCX in IE is a COM component.
- Hook COM component create in IE, check CLSID of Flash OCX

How to Hook Flash OCX load?

- Hook CoGetClassObject function in urlmon.dll
- IAT hook
- In Hook_CoGetClassObject function, use IsEqualCLSID(rclsid, CLSID_Flash) to identify Flash component is being loaded.
- Find Flash OCX module base address and module size, search binary sequence to hook JIT, hook vector create, hook version

OK, Just Run it

- DEMO
- CVE-2015-5119

Manual Check

- FED finally gives me little samples for manual checking.
- I need to debug this samples to confirm it is an 0-day or for getting root cause of the 0-day.

Debugging Hard Point

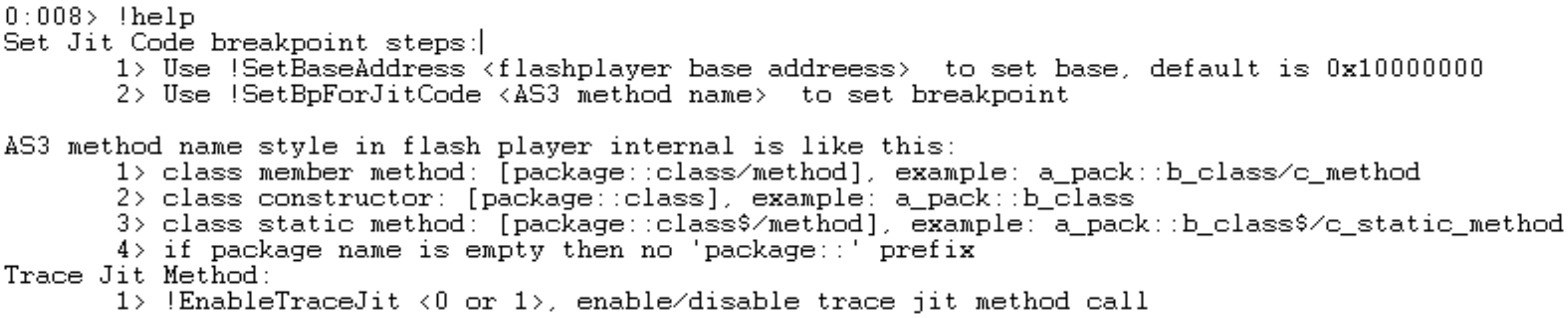
- No symbol of Flash Player.
- All AS3 methods are JITed. Address is dynamic.
- Flash player has script execution time out.

DbgFlashVul

- So I wrote a tool to help debug.
- A windbg extension named DbgFlashVul written in C++.
- It can trace AS3 method.
- It can set break point based on AS3 method name.

DbgFlashVul

- !help

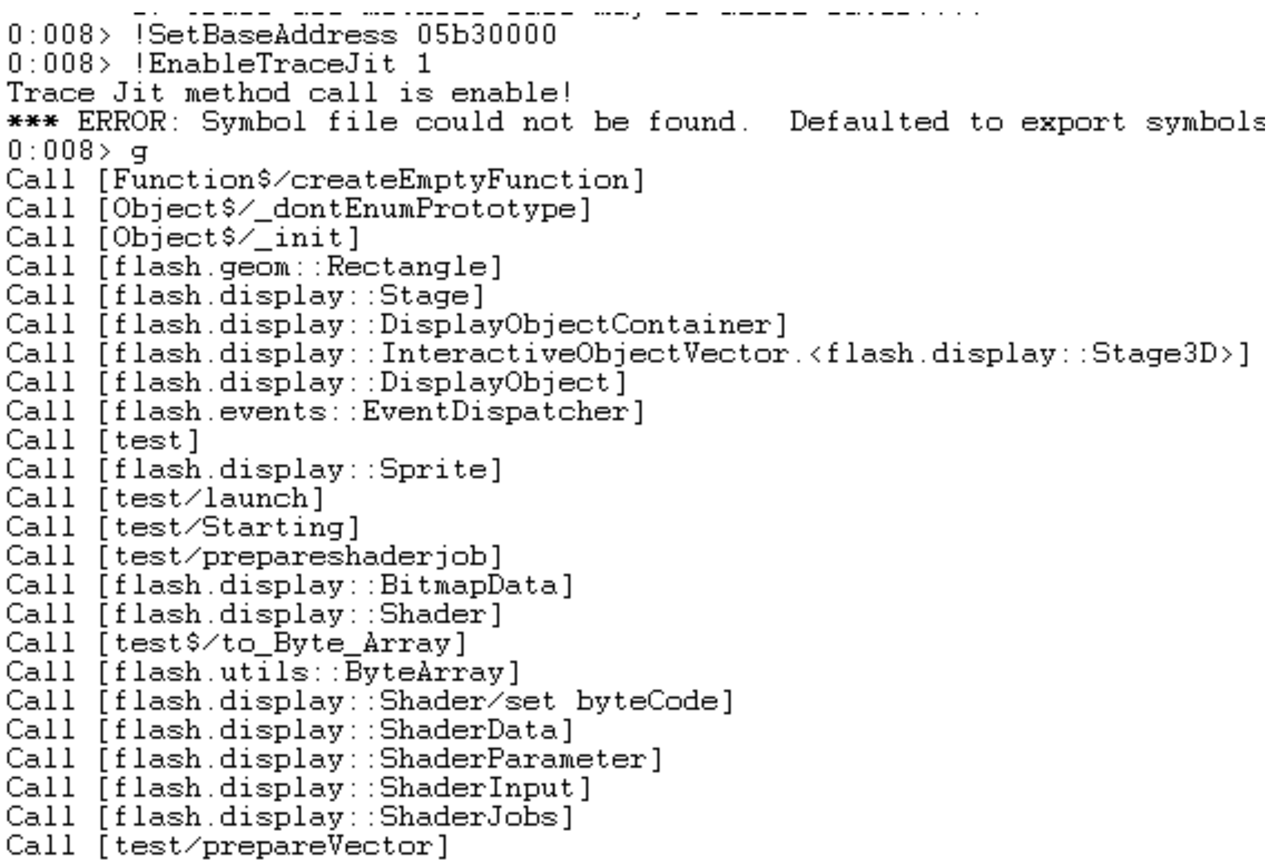


```
0:008> !help
Set Jit Code breakpoint steps:
        1> Use !SetBaseAddress <flashplayer base addreess>  to set base, default is 0x10000000
        2> Use !SetBpForJitCode <AS3 method name>  to set breakpoint

AS3 method name style in flash player internal is like this:
        1> class member method: [package::class/method], example: a_pack::b_class/c_method
        2> class constructor: [package::class], example: a_pack::b_class
        3> class static method: [package::class$/method], example: a_pack::b_class$/c_static_method
        4> if package name is empty then no 'package::' prefix
Trace Jit Method:
        1> !EnableTraceJit <0 or 1>, enable/disable trace jit method call
```

DbgFlashVul

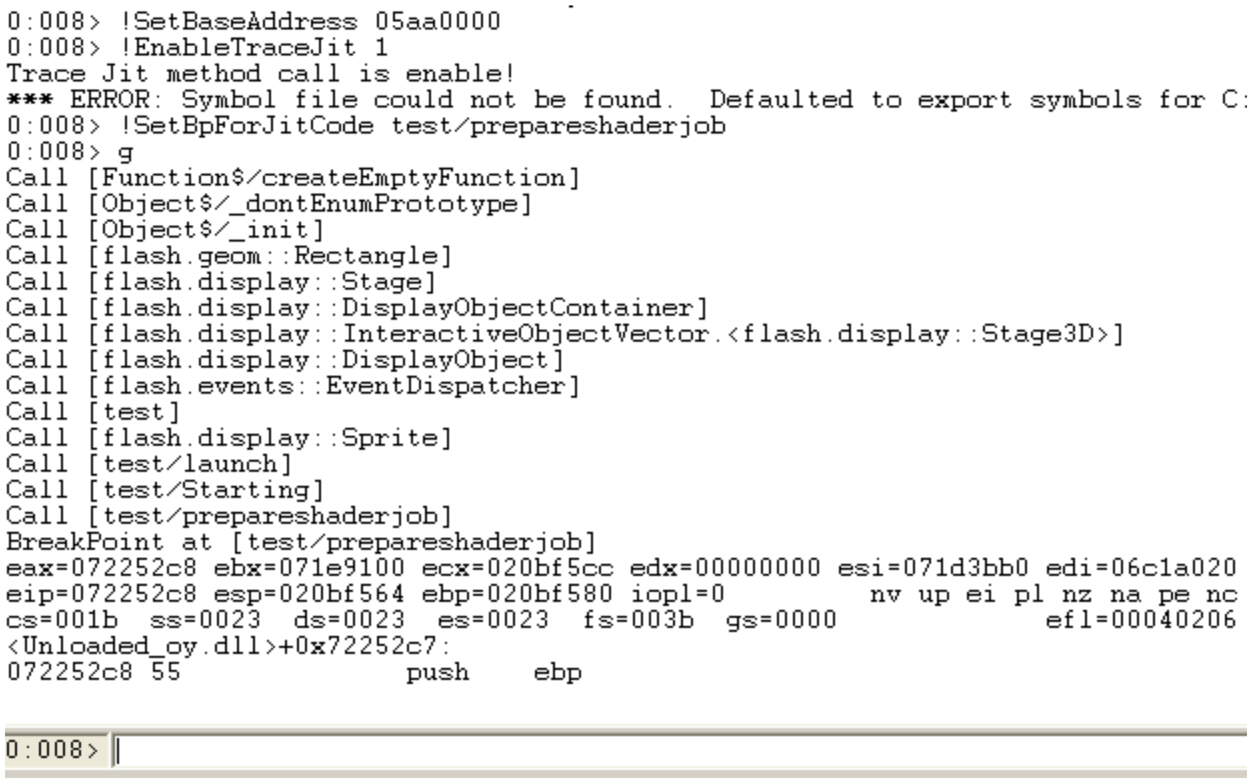
- ! EnableTraceJit 1



```
0:008> !SetBaseAddress 05b30000
0:008> !EnableTraceJit 1
Trace Jit method call is enable!
*** ERROR: Symbol file could not be found.  Defaulted to export symbols
0:008> g
Call [Function$/createEmptyFunction]
Call [Object$/_dontEnumPrototype]
Call [Object$/_init]
Call [flash.geom::Rectangle]
Call [flash.display::Stage]
Call [flash.display::DisplayObjectContainer]
Call [flash.display::InteractiveObjectVector.<flash.display::Stage3D>]
Call [flash.display::DisplayObject]
Call [flash.events::EventDispatcher]
Call [test]
Call [flash.display::Sprite]
Call [test/launch]
Call [test/Starting]
Call [test/prepareshaderjob]
Call [flash.display::BitmapData]
Call [flash.display::Shader]
Call [test$/to_Byte_Array]
Call [flash.utils::ByteArray]
Call [flash.display::Shader/set byteCode]
Call [flash.display::ShaderData]
Call [flash.display::ShaderParameter]
Call [flash.display::ShaderInput]
Call [flash.display::ShaderJobs]
Call [test/prepareVector]
```

DbgFlashVul

- !SetBpForJitCode



```
0:008> !SetBaseAddress 05aa0000
0:008> !EnableTraceJit 1
Trace Jit method call is enable!
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:
0:008> !SetBpForJitCode test/prepareshaderjob
0:008> g
Call [Function$/createEmptyFunction]
Call [Object$/_dontEnumPrototype]
Call [Object$/_init]
Call [flash.geom::Rectangle]
Call [flash.display::Stage]
Call [flash.display::DisplayObjectContainer]
Call [flash.display::InteractiveObjectVector.<flash.display::Stage3D>]
Call [flash.display::DisplayObject]
Call [flash.events::EventDispatcher]
Call [test]
Call [flash.display::Sprite]
Call [test/launch]
Call [test/Starting]
Call [test/prepareshaderjob]
BreakPoint at [test/prepareshaderjob]
eax=072252c8 ebx=071e9100 ecx=020bf5cc edx=00000000 esi=071d3bb0 edi=06c1a020
eip=072252c8 esp=020bf564 ebp=020bf580 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00040206
<Unloaded_oy.dll>+0x72252c7:
072252c8 55              push    ebp

0:008> |
```

A real example : CVE-2015-3090

- Used by most exploit kits.
- Vulnerability can be simplified like this:

```
private var myShaderjob:ShaderJob = null;
this.myShaderjob = new ShaderJob(this.myShader);
....
this.myShaderjob.width = 0;
this.myShaderjob.start();
this.myShaderjob.width = 606;
```

- When changing ShaderJob width asynchronously, it will cause memory overwrite.

A real example : CVE-2015-3090

- The exploit flow can be simplified like this:

```
prepareshaderjob();
prepareVector();    // vector spray
attacking();        // trigger vulnerability to overwrite vector length
if ( !findCorruptVector() ) {
    return (false);
};
buildRopAndShellcode()
exec();
```

A real example : CVE-2015-3090

- For example, we want to get the ROP gagdets and shellcode used by this exploit.
- Uses DbgFlashVul can easily do this.

A real example : CVE-2015-3090

- Almost every flash exploit using corrupt vector will have two AS3 functions, like read_memory and write_memory.
- The two function use corrupt vector to read and write arbitrary memory.
- So, we can use DbgFlashVul to break the execution on write_memory. Exploit uses this function to construct ROP chain and shellcode.

A real example : CVE-2015-3090

- Steps:

> Set break point at write_memory

> After break, get the address of “corruptVector[index] = value”, the assembly is like “mov dword ptr [edx+eax*4+8], ecx”

> Set break point on the address.

> When break, every ecx is a part of ROP chain and shellcode

A real example : CVE-2015-3090

```
0:008> !SetBaseAddress   038f0000
0:008> !SetBpForJitCode   test/write_memory
0:008> g
BreakPoint at [test/write_memory]
eax=05072424 ebx=05039100 ecx=020bf4b0 edx=00000002 esi=05023b08 edi=05023b08
eip=05072424 esp=020bf464 ebp=020bf480 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00040206
<Unloaded_oy.dll>+0x5072423:
05072424 55             push    ebp
0:008> p
……
0:008> p
eax=00089352 ebx=05039100 ecx=03bcbeb6 edx=0510e2c0 esi=05023b08 edi=05023b08
eip=05072553 esp=020bf428 ebp=020bf460 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00040283
<Unloaded_oy.dll>+0x5072552:
05072553 894c8208       mov     dword ptr [edx+eax*4+8],ecx ds:0023:05333010=00000000
0:008> bu 05072553
0:008> g
Breakpoint 4 hit
eax=0008937b ebx=05039100 ecx=03b66ea0 edx=0510e2c0 esi=05023a78 edi=04a6a020
eip=05072553 esp=020bf548 ebp=020bf580 iopl=0         nv up ei ng nz na pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00040287
<Unloaded_oy.dll>+0x5072552:
05072553 894c8208       mov     dword ptr [edx+eax*4+8],ecx ds:0023:053330b4=00000000
0:008> u ecx
Flash32_17_0_0_134!DllUnregisterServer+0x92fe4:
03b66ea0 94             xchg    eax,esp        // stack pivot
03b66ea1 c3             ret
```

How to implement it?

- Get MethodInfo:: getMethodName address by binary searching
- Hook BaseExecMgr::verifyJit like FED
- In Hook function:

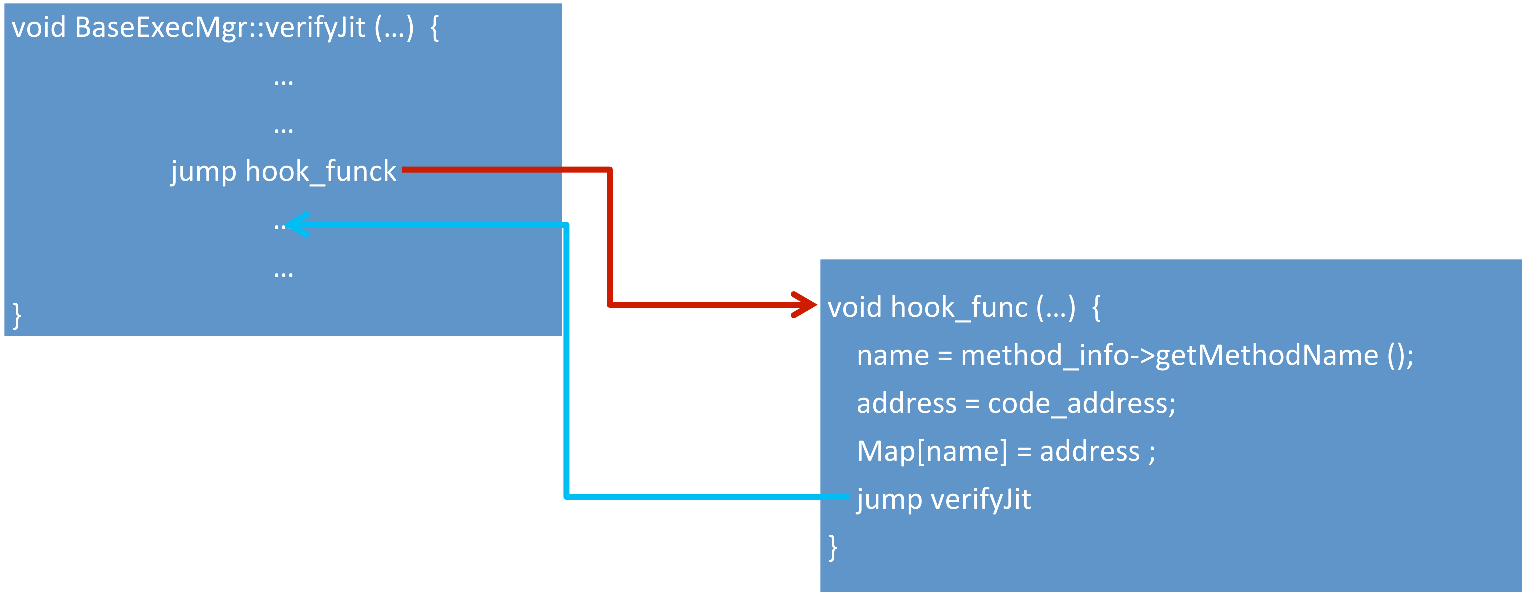
> Get emitted code address and MethodInfo object

> Call MethodInfo:: getMethodName with MethodInfo object(ecx)

> Get AS3 method name from eax

> Save AS3 method name and code address

How to implement it?



```
void BaseExecMgr::verifyJit (…)  {
                                        …
                                        …
                        jump hook_funck
                                        …
                                        …
}
```

```
void hook_func (…)  {
    name = method_info->getMethodName ();
    address = code_address;
    Map[name] = address ;
    jump verifyJit
}
```

DbgFlashVul can do other things

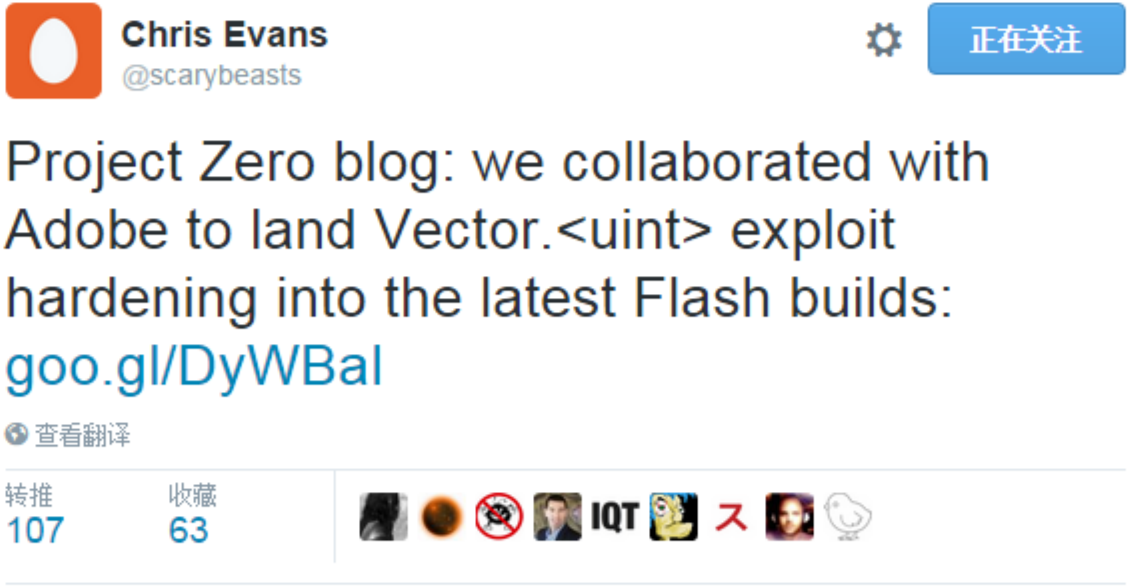
- Help to write flash player exploit
- Help to verify template SWF is correct or not when do fuzzing
- Help to dump embedded SWF by setting break point at LoadBytes

……

Agenda

- Who am I
- Background
- Discover flash 0-day attacks from big set samples
- Vector Length mitigation

Vector exploit mitigation



Chris Evans
@scarybeasts

Project Zero blog: we collaborated with Adobe to land Vector.<uint> exploit hardening into the latest Flash builds: goo.gl/DyWBal

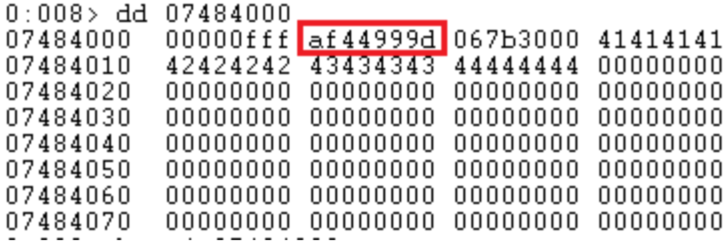
转推	收藏
107	63

Vector exploit mitigation

- Vector length check

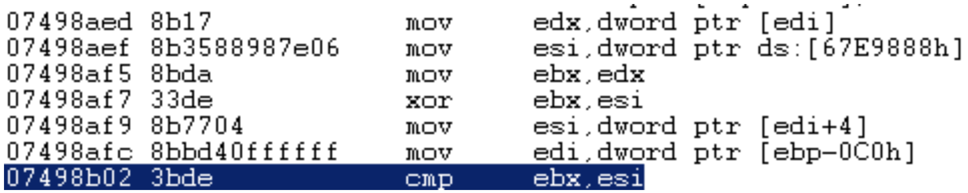
> add a length XOR cookie in vector buffer object

| length | cookie | gc relate | data |



```
0:008> dd 07484000
07484000  00000fff af44999d 067b3000 41414141
07484010  42424242 43434343 44444444 00000000
07484020  00000000 00000000 00000000 00000000
07484030  00000000 00000000 00000000 00000000
07484040  00000000 00000000 00000000 00000000
07484050  00000000 00000000 00000000 00000000
07484060  00000000 00000000 00000000 00000000
07484070  00000000 00000000 00000000 00000000
```

> compare when using length, (length ^ seed) == cookie



```
07498aed 8b17            mov     edx,dword ptr [edi]
07498aef 8b3588987e06    mov     esi,dword ptr ds:[67E9888h]
07498af5 8bda            mov     ebx,edx
07498af7 33de            xor     ebx,esi
07498af9 8b7704          mov     esi,dword ptr [edi+4]
07498afc 8bbd40ffffff    mov     edi,dword ptr [ebp-0C0h]
07498b02 3bde            cmp     ebx,esi
```

Vector exploit mitigation

- Vector length check **bypass**

> need a strong info leak bug to read both length and cookie to calculate the seed

> seed = (length ^ cookie)

Vector exploit mitigation

- Vector buffer object isolated

> allocate vector object in system heap not in flash gc heap

> makes vector buffer memory hard to occupy the freed memory, mitigate the exploit of UAF bugs

> makes heap buffer overflow bugs hard to overwrite vector buffer object.

Vector exploit mitigation

- Vector buffer object isolated bypass

  > need to heap spray many vector objects to some address

  > need a overwrite bug to overwrite a heap sprayed address

Conclusion

- The mitigation makes vector length based exploit hard.
- This mitigation doesn't decrease the number of vulnerabilities of Flash Player.
- The mitigation can bypass but need more good bugs
- Some one may find replacement for vector

Reference

- “Smashing The Heap With Vector,” Haifei Li
- “Inside AVM,” Haifei Li
- Google Project zero, http://googleprojectzero.blogspot.tw/2015/07/significant-flash-exploit-mitigations_16.html

Special Thanks To

- @LambdaTea

  > Implemented FED together with me

Thank you!