Bypass Control Flow Guard Comprehensively

Zhang Yunhai
Agenda

• Overview
• CFG Internals
• Attack Surface
• Universal Bypass
• Fix for the Issue
Who am I

- From Beijing, China
- Researcher of NSFOCUS Security Team
- Focus on exploit detection and prevention
Overview

• Control Flow Guard
  – Mitigation that prevent redirecting control flow to unexpected location
  – First introduced in Windows 8.1 Preview
  – Disabled in Windows 8.1 RTM for compatibility
  – Enabled in Windows 10 Technical Preview and Windows 8.1 Update
Agenda

- Overview
- CFG Internals
- Attack Surface
- Universal Bypass
- Fix for the Issue
Compile Stage

- Append 5 Load Configuration Table entries

<table>
<thead>
<tr>
<th>Load Configuration Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>62f043fc</td>
</tr>
<tr>
<td>00000000</td>
</tr>
<tr>
<td>62b2105c</td>
</tr>
<tr>
<td>00001d54</td>
</tr>
<tr>
<td>00003500</td>
</tr>
</tbody>
</table>
Compile Stage

Guard CF Function Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00009330</td>
<td><code>__security_check_cookie</code></td>
</tr>
<tr>
<td>00009360</td>
<td><code>__EH_epilog3</code></td>
</tr>
<tr>
<td>00009450</td>
<td><code>memcmp</code></td>
</tr>
<tr>
<td>000094e0</td>
<td><code>DllMainCRTStartup</code></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Compile Stage

- Inject a check to ensure that target address is valid

```assembly
mov    eax, dword ptr [ebx]
mov    ecx, ebx
call   dword ptr [eax+7Ch]
```
Compile Stage

- Inject a check to ensure that target address is valid

```
mov     eax, dword ptr [ebx]
mov     esi, dword ptr [eax+7Ch]
mov     ecx, esi
        call    dword ptr [jscript9!_guard_check_icall_fptr (651743fc)]
mov     ecx, ebx
        call    esi
```
Load Stage

- **CFG Bitmap**
  - Track all valid target address
  - Mapped into process memory address space
Load Stage

• Update the CFG Bitmap
Load Stage

- Update the check function pointer

```
0:017> dds_jscript9!__guard_check_icall_fptr l1
62f043fc 77acd970 ntdll!LdrpValidateUserCallTarget
```
Runtime

- ntdll!LdrpValidateUserCallITarget
Runtime

- ntdll!RtlpHandleInvalidUserCallTarget

```
return
```
Agenda

• Overview
• CFG Internals
• Attack Surface
• Universal Bypass
• Fix for the Issue
Attack Surface

- Non-CFG Module
- JIT Generated Code
- Indirect Jump
- Return Address
- Valid API Function
Non-CFG Module

• Contain unprotected indirect call

```
0:006> u slc!SLConsumeWindowsRight+0xe3
slc!SLConsumeWindowsRight+0xe3:
73127463 8b06    mov    eax, dword ptr [esi]
73127465 56    push    esi
73127466 ff5008    call    dword ptr [eax+8]
```
Non-CFG Module

- All bits in the CFG Bitmap are set

```
0:006> lmm slc
start   end       module name
73110000 73138000   slc        (deferred)
0:006> dd poi(ntdll!LdrSystemDllInitBlock+0x60) + 731100 * 4
025e4400 ffffffff ffffffff ffffffff ffffffff
025e4410 ffffffff ffffffff ffffffff ffffffff
025e4420 ffffffff ffffffff ffffffff ffffffff
025e4430 ffffffff ffffffff ffffffff ffffffff
025e4440 ffffffff ffffffff ffffffff ffffffff
025e4450 ffffffff ffffffff ffffffff ffffffff
025e4460 ffffffff ffffffff ffffffff ffffffff
025e4470 ffffffff ffffffff ffffffff ffffffff
```
Non-CFG Module

- Will exhaust eventually
  - Vendors trend to compile new modules with CFG enable
JIT Generated Code

- Just like a non-CFG module
  - Contain unprotected indirect call
  - All bits in the CFG Bitmap are set
JIT Generated Code

• Both are no longer the case in Edge
  – JIT code is instrumented
  – JIT code pages don't have all bits set
Indirect Jump

• Redirect control flow like indirect call

```
jscript9!NativeCodeGenerator::CheckCodeGenThunk:
62b3c5e2 55       push   ebp
62b3c5e3 8bec     mov    ebp, esp
62b3c5e5 ff742408 push   dword ptr [esp+8]
62b3c5e9 e812ffffff call  jscript9!NativeCodeGenerator::CheckCodeGen
62b3c5ee 5d       pop    ebp
62b3c5ef ffe0     jmp    eax
```
Indirect Jump

• Protect using the same mechanism as indirect call

```
chakra!NativeCodeGenerator::CheckCodeGenThunk:
621b4020 55        push   ebp
621b4021 8bec      mov     ebp, esp
621b4023 ff742408  push    dword ptr [esp+8]
621b4027 e8c4b4f6ff call    chakra!NativeCodeGenerator::CheckCodeGen
621b402c 50        push    eax
621b402d 8bc8      mov     ecx, eax
621b402f ff1504154f62 call    dword ptr [chakra!__guard_check_icall_fptr]
621b4035 58        pop     eax
621b4036 5d        pop     ebp
621b4037 ffe0      jmp     eax
```
Return Address

• Overwrite return address
  – Locate the stack
  – Search the stack for an appropriate frame
  – Replace the stack frame with crafted one
Valid API Function

- ntdll!NtContinue
- KERNELBASE!SetThreadContext
- msvcr!longjmp
- KERNEL32!WinExec
- SHELL32!ShellExecuteExA
- KERNEL32!LoadLibraryA
Agenda

- Overview
- CFG Internals
- Attack Surface
- Universal Bypass
- Fix for the Issue
Objective

- ☐ Bypass CFG Comprehensively
Guard CF Check Function

• Called through Guard CF Check Function Pointer

```
62c31e1b 8bce          mov    ecx,esi
62c31e18 8b707c        mov    esi,dword ptr [eax+7Ch]
62c31e1d ff15fc43f062  call   dword ptr [jscript9!__guard_check_icall_fptr]
```
Guard CF Check Function

- Behavior when target address is valid

```assembly
mov edx, dword ptr [ntdll!LdrSystemDllInitBlock+0x60 (7753e170)]
mov eax, ecx
shr eax, 8
mov edx, dword ptr [edx+eax*4]
mov eax, ecx
shr eax, 3
test cl, 0Fh
jne ntdll!LdrpValidateUserCallTargetBitMapRet+0x1 (774bd98e)
bte edx, eax
jae ntdll!LdrpValidateUserCallTargetBitMapRet+0xa (774bd997)
ret```

Objective

- ☐ Overwrite Guard CF Check Function Pointer
- ☐ Bypass CFG Comprehensively
Overwrite Guard CF Check Function Pointer
Objective

- Make Read-only Memory Writeable
- Overwrite Guard CF Check Function Pointer
- Bypass CFG Comprehensively
## CustomHeap::Heap

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x000</td>
<td>HeapPageAllocator</td>
<td>PageAllocator</td>
</tr>
<tr>
<td>+0x060</td>
<td>HeapArenaAllocator</td>
<td>Ptr32 ArenaAllocator</td>
</tr>
<tr>
<td>+0x064</td>
<td>PartialPageBuckets</td>
<td>[7] DListBase<a href="">CustomHeap::Page</a></td>
</tr>
<tr>
<td>+0x09c</td>
<td>FullPageBuckets</td>
<td>[7] DListBase<a href="">CustomHeap::Page</a></td>
</tr>
<tr>
<td>+0x0d4</td>
<td>LargeObjects</td>
<td>DListBase<a href="">CustomHeap::Page</a></td>
</tr>
<tr>
<td>+0x0dc</td>
<td>DecommittedBuckets</td>
<td>DListBase<a href="">CustomHeap::Page</a></td>
</tr>
<tr>
<td>+0x0e4</td>
<td>DecommittedLargeObjects</td>
<td>DListBase<a href="">CustomHeap::Page</a></td>
</tr>
<tr>
<td>+0x0ec</td>
<td>CriticalSection</td>
<td>LPCRITICAL_SECTION</td>
</tr>
</tbody>
</table>
Destructor Behavior

CustomHeap::Heap::~Heap

CustomHeap::Heap::FreeAll

CustomHeap::Heap::FreeBucket

CustomHeap::Heap::EnsurePageReadWrite<1,4>

VirtualProtect(address, 0x1000, 0x4, &flOldProtect)
Locate CustomHeap::Heap

- CustomHeap::Heap is a member of InterpreterThunkEmitter at offset 0xC

```
0:018> dd 0441d380 l4
0441d380 00000000 0c6cdd28 00000000 679521d8
0:018> dds 0441d380 + c l1
0441d38c 679521d8 jscript9!HeapPageAllocator::`vftable`
```
Locate CustomHeap::Heap

- InterpreterThunkEmitter is pointed by a member of Js::ScriptContext at offset 0x4b0

```
0:018> dd 0c6c0b80 + 4b0 14
0c6ce030 0441d380 0c66e860 00000000 00000000
```
Locate CustomHeap::Heap

- Js::ScriptContext is pointed by a member of ScriptEngine at offset 0x4

```
0:018> dds 0441d138 l1
0441d138 6794a5f4 jscript9!ScriptEngine::`vftable'
0:018> dd 0441d138 l4
0441d138 6794a5f4 0c6cdb80 00000009 043591a8
```
Locate CustomHeap::Heap

- ScriptEngine is pointed by a member of ScriptSite at offset 0x4

```
0:018> dd 0c629d18 14
0c629d18  00000003 0441d138 0441d138 00000000
```
Decommit Issue

• All Buckets are empty when destructor is called
  – All CustomHeap::Page are decommitted in Js::ScriptContext::Close
  – Decommitted CustomHeap::Page is removed from Bucket
  – Js::ScriptContext::Close is called before CustomHeap::Heap::~Heap
Decommit Issue

• Resolution
  – Insert a fake CustomHeap::Page into Bucket
  – Prevent a CustomHeap::Page from being decommitted
Objective

- ☑ Make Read-only Memory Writeable
- ☑ Overwrite Guard CF Check Function Pointer
- ☑ Bypass CFG Comprehensively
Agenda

• Overview
• CFG Internals
• Attack Surface
• Universal Bypass
• Fix for the Issue
Timeline

- Jan 22 2015: Report to MSRC
- Jan 30 2015: Confirmed
- Mar 10 2015: Patch Released
HeapPageAllocator::ProtectPages

- Wrapper of VirtualProtect
- Check before call VirtualProtect
  - lpAddress is 0x1000 aligned
  - lpAddress >= Segment address
  - lpAddress + dwSize <= Segment address + Segment size
  - dwSize <= RegionSize
  - Protect == Expected Protect
EnsurePageReadWrite<1,4>

- Call HeapPageAllocator::ProtectPages instead of VirtualProtect
- Expected Protect is PAGE_EXECUTE
Black Hat Sound Bytes

• No Silver Bullet
• Read-only ≠ Secure
• Control the Data Control the Execute