The Power of Data-Oriented Attacks: Bypassing Memory Mitigation Using Data-Only Exploitation Technique Part I

Bing Sun, Chong Xu, Stanley Zhu

McAfee by Intel

About Speaker

- **Bing Sun**
  - Bing Sun is a senior information security researcher, and leads the IPS security research team of McAfee. He has extensive experiences in operating system kernel layer and information security R&D, with especially deep diving in advanced vulnerability exploitation and detection, Rootkits detection, firmware security, and virtualization technology.
- **Chong Xu**
  - Chong Xu received his PhD degree from Duke University with networking and security focus. He is currently a director leading McAfee Labs IPS team, which leads the McAfee Labs vulnerability research, malware and APT detection, botnet detection, and feeds security content and advanced detection features to McAfee's network IPS, host IPS, and firewall products, as well as global threat intelligence.
- **Stanley Zhu**
  - Stanley Zhu is a security researcher at McAfee. One of designers of Bytehero Heuristic Detection Engine, he provided the BDV engine with Virustotal and OPSWAT platform. He has many years of research experience in information security techniques. He is interested in advanced vulnerability exploitation and detection as well as virus, rootkit and reverse engineering. He has spoken at security conferences such as CanSecWest2014, AVAR2012, XCon2010, XCon2015, and XCon2016.

Abstract

- As Control Flow Integrity (CFI) enforcement solutions are widely adopted by major applications, traditional memory vulnerability exploitation techniques aiming to hijack the control flow have become increasingly difficult. For example, Microsoft's Control Flow Guard (CFG) is an effective CFI solution against traditional memory exploits. However, due to the CFG implementation limitations, we have seen new exploitation techniques such as using the out-of-context function call to bypass CFG. We believe eventually these limitations could all be overcome or improved, and ultimately we expect a fine-grained CFG solution to completely defeat control-flow hijacking. Consequently, attackers have begun to seek alternatives to exploit memory vulnerabilities without diverting the control flow. As a result of this trend, the data-oriented attacks have emerged. As its name suggests, a data-oriented attack focuses on altering or forging the critical data of an application, rather than attempting to alter its control flow. The data-oriented attack may allow the attacker to do some powerful things, such as loading certain unwanted or disabled modules or changing the attributes of certain memory pages. Sometimes this can be achieved by changing only a few bits of data. Today, most successful memory exploits can gain some level of memory read/write primitives during exploitation of memory corruption vulnerability, which makes data-oriented attacks possible. In this talk, we will present some interesting examples that show the power of data-oriented attacks. We then discuss ways to prevent such attacks. We conclude by live demonstrations of CFG/DEP bypass on Windows 10’s Edge using data-only exploitation technique.

Agenda

- The Principle of CFG
- Known CFG Bypass Methods
  - Corrupt function’s return address on stack
  - Transit via unguarded trampoline code
  - Call function out-of-context
- Call Function Out-of-Context
  - Test an address’ readability by calling kernel32!GlobalLock
  - Bypass CFG by calling ntdll!RtlProtectHeap
- Data-only Attack
  - Bypass CFG by abusing WARP Shader JIT
  - Bypass CFG/DEP by loading disabled Silverlight ActiveX Plugin
- Suggestions for Preventing Data-only Attack
- Conclusion
- Acknowledgement

The Principle of CFG

- About CFG (Control Flow Guard)
  - A compiler-aided exploitation mitigation mechanism that prevents exploit from hijacking the control flow.
  - Compiler inserts CFG check before each indirect control transfer instruction (call/jmp), and at runtime the CFG check will validate the call target address against a pre-configured CFG bitmap to determine whether the call target is valid or not. The process will be terminated upon an unexpected call target being identified.
  - The Relative Virtual Address (RVA) of all valid call targets determined at the time of compilation are kept in a Guard CF Function table in PE file. During the PE loading process, the loader will read CF info from guard CF function table and update the CFG bitmap.
  - The read-only CFG bitmap is maintained by the OS, and part of the bitmap is shared by all processes. An even bit in CFG bitmap corresponds to one 16-bytes aligned address, while an odd bit corresponds to 15 non 16-bytes aligned addresses.
  - When the PE file is loaded, __guard_check_icall_fptr will be resolved to point to ntdll!LdrpValidateUserCallTarget. (on x64, __guard_dispatch_icall_fptr -> ntdll!LdrpDispatchUserCallTarget)

The Principle of CFG (Continued)

```
0:034> u RPCRT4!Ndr64pCommonStringUnmarshall+0x137
RPCRT4!Ndr64pCommonStringUnmarshall+0x137:
00007fff`67c34167 ff154b330e00     call    qword ptr [RPCRT4!_guard_dispatch_icall_fptr (00007fff`67d174b8)]
00007fff`67c3416d 488bd0           mov     rdx,rax
00007fff`67c34170 4885c0           test    rax,rax
00007fff`67c34173 0f845e3c0700     je      RPCRT4!Ndr64pCommonStringUnmarshall+0x73da7 (00007fff`67ca7dd7)
00007fff`67c34179 488d0c07         lea     rcx,[rdi+rax]
00007fff`67c3417d c7014c4d454d     mov     dword ptr [rcx],4D454D4Ch
00007fff`67c34183 897904           mov     dword ptr [rcx+4],edi
00007fff`67c34186 488b8338010000   mov     rax,qword ptr [rbx+138h]
0:034> dq 7fff67d174b8 L1
00007fff`67d174b8  00007fff`67ea5850
0:034> u 7fff67ea5850 Lf
ntdll!LdrpDispatchUserCallTarget:
00007fff`67ea5850 4c8b1db1ea0c00   mov     r11,qword ptr [ntdll!LdrSystemDllInitBlock+0x68 (00007fff`67f74308)]
00007fff`67ea5857 4c8bd0           mov     r10,rax
00007fff`67ea585a 49c1ea09         shr     r10,9
00007fff`67ea585e 4f8b1cd3         mov     r11,qword ptr [r11+r10*8]
00007fff`67ea5862 4c8bd0           mov     r10,rax
00007fff`67ea5865 49c1ea03         shr     r10,3
00007fff`67ea5869 a80f             test    al,0Fh
00007fff`67ea586b 7509             jne     ntdll!LdrpDispatchUserCallTarget+0x26 (00007fff`67ea5876)
00007fff`67ea586d 4d0fa3d3         bt      r11,r10
00007fff`67ea5871 7310             jae     ntdll!LdrpDispatchUserCallTarget+0x33 (00007fff`67ea5883)
00007fff`67ea5873 48ffe0           jmp     rax
00007fff`67ea5876 4983ca01         or      r10,1
00007fff`67ea587a 4d0fa3d3         bt      r11,r10
00007fff`67ea587e 7303             jae     ntdll!LdrpDispatchUserCallTarget+0x33 (00007fff`67ea5883)
00007fff`67ea5880 48ffe0           jmp     rax
```

16-byte aligned

Non 16-byte aligned, set bit 0 of offset

Compiler inserts a call target check before each indirect function call/jmp

CFG bitmap base

High 55-bit of call target address is used as an index into the bitmap to get a 64-bit bitmap entry

Bit 3 ~ 8 of target address is used as an offset

Test the bit “offset” of that 64-bit bitmap entry. Target address is valid if bit is set, otherwise trigger INT 29h

Known CFG Bypass Methods

- Corrupt function’s return address on stack
  - “Bypassing Control Flow Guard in Windows 10”
  - Mitigation: RFG (Return Flow Guard), Intel’s CET

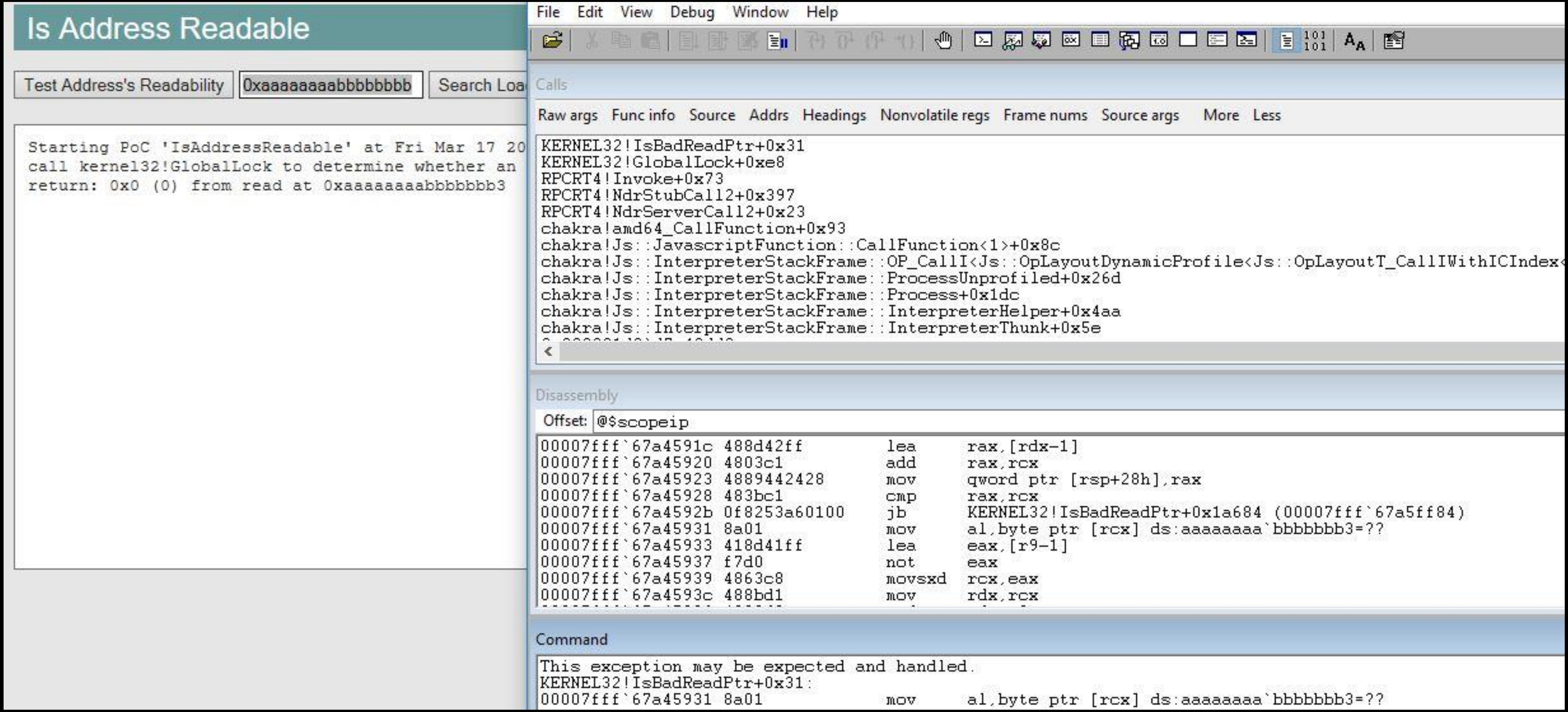
- Transit via unguarded trampoline code (mostly involving dynamic code, such as JIT)
  - “Use Chakra engine again to bypass CFG”
  - “Chakra JIT CFG Bypass”
  - Mitigation: JIT security improvement (JIT code checksum, remote JIT etc)

- Call function out-of-context
  - “Bypass Control Flow Guard Comprehensively”
  - “Mitigation bounty — 4 techniques to bypass mitigations”
  - Mitigation: Fine-grained CFI (improvement on CFG after WIP build 14986)

Call Function Out-of-context

- The basic idea of calling function out-of-context
  - Issue a function call to certain unexpected target via memory indirect call instruction; however from the program's logic perspective such a call is not supposed to happen from that call site. This is essentially one type of execution control hijacking.
- How to make an out-of-context call
  - Overwrite an existing function pointer (such as in an object's vftable) with the target function of out-of-context call.
  - The target function needs to be able to pass the CFG check because almost all memory indirect calls have CFG check prior to them.
  - The call to the overwritten function can be reliably and repeatedly triggered from the context of scripting language.
  - The number and order of arguments to the target function should be (at least partially) controllable to the scripting language.
  - It's preferable to be able to get the target function's return value in its original form.
- Example
  - "From read-write anywhere to controllable calls" This is a very good example of calling function out-of-context, controlling all arguments and getting the return value back.

Test Address' Readability By Calling kernel32!GlobalLock



Call kernel32!GlobalLock Out-of-context

```
RPCRT4!Invoke+0x70
RPCRT4!NdrStubCall2+0x397
RPCRT4!NdrServerCall2+0x23
chakra!amd64_CallFunction+0x93
chakra!Js::JavascriptFunction::CallFunction<1>+0x8c
chakra!Js::InterpreterStackFrame::OP_CallI<Js::OpLayoutDynamicProfile<Js::OpLayoutT_CallIWithICIndex<Js::LayoutSizePolicy<0> > > >+0x129
chakra!Js::InterpreterStackFrame::ProcessUnprofiled+0x26d
chakra!Js::InterpreterStackFrame::Process+0x1dc
chakra!Js::InterpreterStackFrame::InterpreterHelper+0x4aa
chakra!Js::InterpreterStackFrame::InterpreterThunk+0x5e
0x000002ba`d74f0dd2
chakra!amd64_CallFunction+0x93
chakra!Js::JavascriptFunction::CallFunction<1>+0x8c
chakra!Js::InterpreterStackFrame::OP_CallI<Js::OpLayoutDynamicProfile<Js::OpLayoutT_CallIWithICIndex<Js::LayoutSizePolicy<0> > > >+0x129
```

out-of-context call via RPC

Disassembly

Offset: @$scopeip

```
00007fff`67c9eb64 f30f7e4c2408     movq    xmm1,mmword ptr [rsp+8]
00007fff`67c9eb6a 4c8b442410       mov     r8,qword ptr [rsp+10h]
00007fff`67c9eb6f f30f7e542410     movq    xmm2,mmword ptr [rsp+10h]
00007fff`67c9eb75 4c8b4c2418       mov     r9,qword ptr [rsp+18h]
00007fff`67c9eb7a f30f7e5c2418     movq    xmm3,mmword ptr [rsp+18h]
00007fff`67c9eb80 41ffd2           call    r10 {KERNEL32!GlobalLock (00007fff`67a45800)}
00007fff`67c9eb83 488b7528         mov     rsi,qword ptr [rbp+28h]
00007fff`67c9eb87 488b7d30         mov     rdi,qword ptr [rbp+30h]
00007fff`67c9eb8b 488be5           mov     rsp,rbp
00007fff`67c9eb8e 488b6d20         mov     rbp,qword ptr [rbp+20h]
```

Command

```
0:017> r
rax=00000fffecf48b00 rbx=000000000000000c rcx=aaaaaaaabbbbbbb3
rdx=0000000000000000 rsi=000000f259df8d70 rdi=00007fff67a45800
rip=00007fff67c9eb80 rsp=000000f259df8820 rbp=000000f259df8880
 r8=0000000000000000  r9=0000000000000000 r10=00007fff67a45800
r11=00007fff67c3a644 r12=000002baeaf55560 r13=000002baeaf53048
r14=000000f259df8f98 r15=0000000000000000
iopl=0         nv up ei pl nz na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
RPCRT4!Invoke+0x70:
00007fff`67c9eb80 41ffd2          call    r10 {KERNEL32!GlobalLock (00007fff`67a45800)}
```

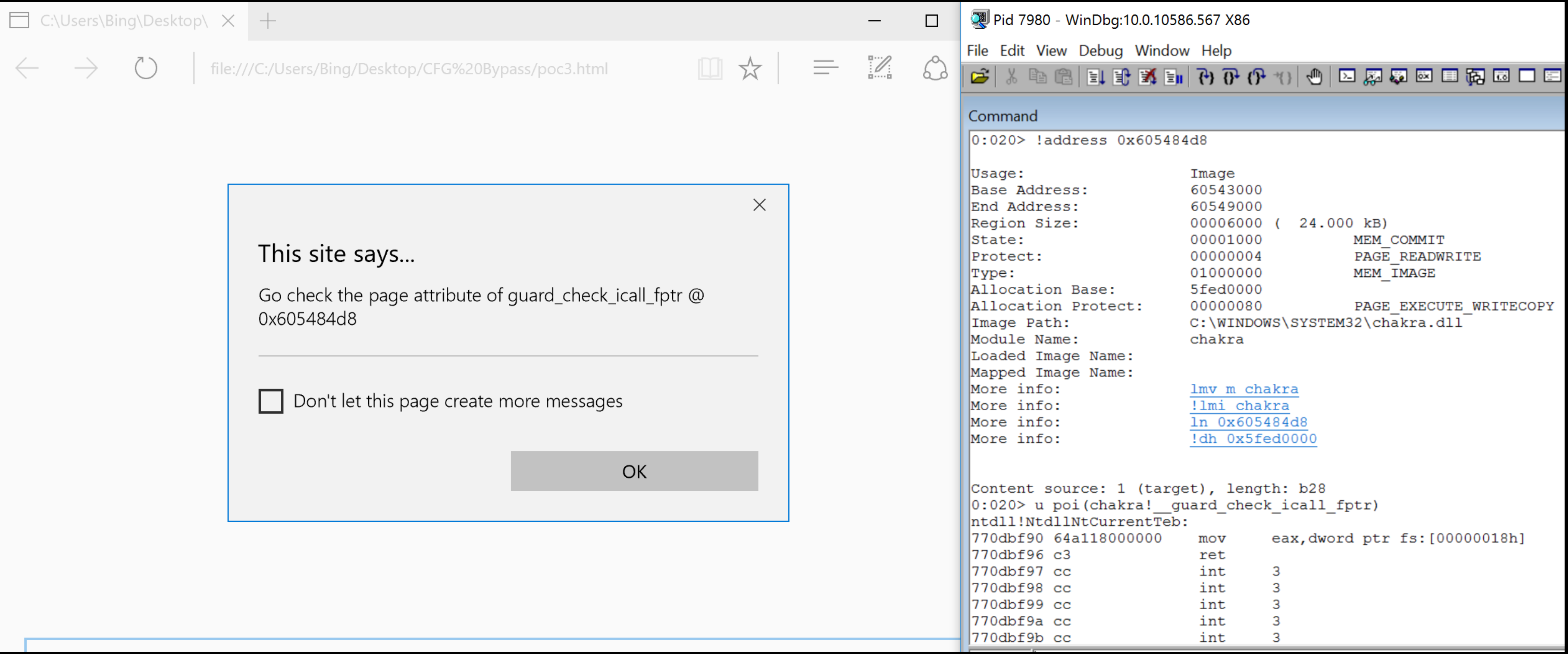
address to test is passed in as the 1st argument

kernel32!GlobalLock

kernel32!GLobalLock is valid for CFG (in WIP build 14986 and before), while kernel32!IsBadReadPtr has been marked as sensitive API. Internally GLobalLock calls IsBadReadPtr that is protected by SEH, so it can be used to reliably test an address' readability.

```
.text:0000000180015800 ; LPVOID __stdcall GlobalLock(HGLOBAL hMem)
...
.text:000000018001580A        mov    rbx, rcx
.text:000000018001580D        mov    r8d, 8
.text:0000000180015813        test   r8b, cl
.text:0000000180015816        jz     loc_1800158C2        // the bit3 of target address must be cleared
...
..text:00000001800158DB       mov    edx, 1     ; ucb
.text:00000001800158E0        mov    rcx, rbx   ; lp
.text:00000001800158E3        call   IsBadReadPtr         // call IsBadReadPtr
.text:00000001800158E8        test   eax, eax
.text:00000001800158EA        jz     short loc_1800158B4
.text:00000001800158EC        jmp    loc_18002FF71
.text:00000001800158EC GlobalLock    endp
```

Bypass CFG By Calling ntdll!RtlProtectHeap



Call ntdll!RtlProtectHeap Out-of-context

```
chakra!Js::JavascriptOperators::OP_DeleteElementI+0x102
chakra!Js::InterpreterStackFrame::ProcessProfiled+0x11c3
chakra!Js::InterpreterStackFrame::Process+0xa6
chakra!Js::InterpreterStackFrame::OP_TryCatch+0x49
chakra!Js::InterpreterStackFrame::ProcessProfiled+0xffd
chakra!Js::InterpreterStackFrame::InterpreterHelper+0x4a1
chakra!Js::InterpreterStackFrame::InterpreterThunk+0x38
0x8ae0fe2
chakra!Js::JavascriptFunction::CallFunction<1>+0x91
```

out-of-context call via
chakra!Js::JavascriptNativeIntArray::DeleteItem

Disassembly

Offset: @$scopeip

```
5ce14fc2 8bc8            mov     ecx,eax
5ce14fc4 ff7514          push    dword ptr [ebp+14h]
5ce14fc7 ff75fc          push    dword ptr [ebp-4]
5ce14fca ff15dc54395d    call    dword ptr [chakra!__guard_check_icall_fptr (5d3954dc)]
5ce14fd0 8bcb            mov     ecx,ebx
5ce14fd2 ff55f4          call    dword ptr [ebp-0Ch]  ss:0023:049fce44={ntdll!RtlProtectHeap (77db5840)}
5ce14fd5 3bf4            cmp     esi,esp
5ce14fd7 7407            je      chakra!Js::JavascriptOperators::OP_DeleteElementI+0x110 (5ce14fe0)
5ce14fd9 b904000000      mov     ecx,4
5ce14fde cd29            int     29h
5ce14fe0 eb8a            jmp     chakra!Js::JavascriptOperators::OP_DeleteElementI+0x9c (5ce14f6c)
```

Command

```
0:009> r
eax=0efb6b08 ebx=12000180 ecx=12000180 edx=00000100 esi=049fce38 edi=04419c10
eip=5ce14fd2 esp=049fce30 ebp=049fce50 iopl=0         nv up ei pl zr na pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200247
chakra!Js::JavascriptOperators::OP_DeleteElementI+0x102:
5ce14fd2 ff55f4          call    dword ptr [ebp-0Ch]  ss:0023:049fce44={ntdll!RtlProtectHeap (77db5840)}
0:009> ? poi(esp)
Evaluate expression: 302055424 = 12010000
0:009> ? poi(esp+4)
Evaluate expression: 0 = 00000000
```

1st arg: fake heap with the address to unprotect at _HEAP_SEGMENT.BaseAddress

2nd arg: 0 (unprotect), 1 (protect)

ntdll!RtlProtectHeap

As its name suggests, native API ntdll!RtlProtectHeap can be invoked to protect (read-only, 2nd arg = 1) or unprotect (read/write, 2nd arg = 0) a heap. For example, ntdll.dll relies on this function to protect its critical data on LdrpMrdataHeap. It calls ntdll!RtlpProtectHeap internally to do the actual job of protection change.

```
.text:6A275840 ; int __stdcall RtlProtectHeap(PVOID BaseAddress, char)
.text:6A275840          mov    edi, edi
...
.text:6A275899          cmp    dword ptr [esi+8], 0DDEEDDEEh
.text:6A2758A0          mov    edx, edi      ; NewProtect
.text:6A2758A2          mov    ecx, esi      ; BaseAddress
.text:6A2758A4          jz     short loc_6A2758D4
.text:6A2758A6          call   _RtlpProtectHeap@8 ; RtlpProtectHeap(x,x)
...
.text:6A2758BF          pop    esi
.text:6A2758C0          pop    ebp
.text:6A2758C1          retn   8
```

ntdll!RtlpProtectHeap

ntdll!RtlpProtectHeap makes the memory attribute change by calling system service ntdll!ZwProtectVirtualMemory.

```
.text:6A2758DB ; __stdcall RtlpProtectHeap(x, x)
...
.text:6A27592C        mov    eax, [ebp+var_24]
.text:6A27592F        mov    [ebp+ProtectSize], eax
.text:6A275932        lea    eax, [ebp+OldProtect]
.text:6A275935        push   eax             ; OldProtect
.text:6A275936        push   [ebp+NewProtect] ; NewProtect
.text:6A275939        lea    eax, [ebp+ProtectSize]
.text:6A27593C        mov    [ebp+BaseAddress], esi
.text:6A27593F        push   eax             ; ProtectSize
.text:6A275940        lea    eax, [ebp+BaseAddress]
.text:6A275943        push   eax             ; BaseAddress
.text:6A275944        push   0FFFFFFFFh      ; ProcessHandle
.text:6A275946        call   _ZwProtectVirtualMemory@20 ; ZwProtectVirtualMemory(x,x,x,x,x)
...
```

Call ntdll!RtlProtectHeap Out-of-context

```
ntdll!RtlpProtectHeap+0x70
ntdll!RtlProtectHeap+0x6b
chakra!Js::JavascriptOperators::OP_DeleteElementI+0x105
chakra!Js::InterpreterStackFrame::ProcessProfiled+0x11c3
chakra!Js::InterpreterStackFrame::Process+0xa6
chakra!Js::InterpreterStackFrame::OP_TryCatch+0x49
```

Disassembly

Offset: @$scopeip

```
77db593c 8975f4          mov     dword ptr [ebp-0Ch],es
77db593f 50              push    eax
77db5940 8d45f4          lea     eax,[ebp-0Ch]
77db5943 50              push    eax
77db5944 6aff            push    0FFFFFFFFh
77db5946 e875b50100      call    ntdll!NtProtectVirtualMemory (77dd0ec0)
77db594b 8bf8            mov     edi,eax
77db594d 85ff            test    edi,edi
77db594f 0f881f9f0400    js      ntdll!RtlpProtectHeap+0x49f99 (77dff874)
77db5955 0375dc          add     esi,dword ptr [ebp-24h]
77db5958 3b7318          cmp     esi,dword ptr [ebx+18h]
77db595b 72aa            jb      ntdll!RtlpProtectHeap+0x2c (77db5907)
```

Internally ntdll!NtProtectVirtualMemory is called to perform the actual memory attribute change

Command

```
0:009> !address 5d395000

Usage:                  Image
Base Address:           5d395000
End Address:            5d396000
Region Size:            00001000 (   4.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000004          PAGE_READWRITE
Type:                   01000000          MEM_IMAGE
Allocation Base:        5cd50000
Allocation Protect:     00000080          PAGE_EXECUTE_WRITECOPY
Image Path:             C:\WINDOWS\SYSTEM32\chakra.dll
Module Name:            chakra
Loaded Image Name:      C:\WINDOWS\SYSTEM32\chakra.dll
Mapped Image Name:
More info:              lmv m chakra
More info:              !lmi chakra
More info:              ln 0x5d395000
More info:              !dh 0x5cd50000


Content source: 1 (target), length: 1000
0:009> d 12010000 + 8c L10
1201008c  00 50 39 5d 00 00 00 00-00 00 00 00 00 90 39 5d  .P9].........9]
```

The address to unprotect is preset at _HEAP_SEGMENT.BaseAddress of the fake heap

CFG Improvement in WIP 15048

```
ntdll!LdrpDispatchUserCallTargetES:
00007fff`ed876410 4c8b1d594f0d00  mov     r11,qword ptr [ntdll!LdrSystemDllInitBlock+0xb0 (00007fff`ed94b370)]
00007fff`ed876417 4c8bd0          mov     r10,rax
00007fff`ed87641a 49c1ea09        shr     r10,9
00007fff`ed87641e 4f8b1cd3        mov     r11,qword ptr [r11+r10*8]
00007fff`ed876422 4c8bd0          mov     r10,rax
00007fff`ed876425 49c1ea03        shr     r10,3
00007fff`ed876429 a80f            test    al,0Fh
00007fff`ed87642b 7509            jne     ntdll!LdrpDispatchUserCallTargetES+0x26 (00007fff`ed876436)
00007fff`ed87642d 4d0fa3d3        bt      r11,r10
00007fff`ed876431 7316            jae     ntdll!LdrpDispatchUserCallTargetES+0x39 (00007fff`ed876449) [br=1]
00007fff`ed876433 48ffe0          jmp     rax
```

Command

```
0:023> r rax=kernel32!GlobalLock
0:023> t
ntdll!LdrpDispatchUserCallTargetES+0x7:
00007fff`ed876417 4c8bd0          mov     r10,rax
0:023>
ntdll!LdrpDispatchUserCallTargetES+0xa:
00007fff`ed87641a 49c1ea09        shr     r10,9
0:023>
ntdll!LdrpDispatchUserCallTargetES+0xe:
00007fff`ed87641e 4f8b1cd3        mov     r11,qword ptr [r11+r10*8] ds:00007ff5`ff86a990=0000000000200000
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x12:
00007fff`ed876422 4c8bd0          mov     r10,rax
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x15:
00007fff`ed876425 49c1ea03        shr     r10,3
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x19:
00007fff`ed876429 a80f            test    al,0Fh
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x1b:
00007fff`ed87642b 7509            jne     ntdll!LdrpDispatchUserCallTargetES+0x26 (00007fff`ed876436) [br=0]
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x1d:
00007fff`ed87642d 4d0fa3d3        bt      r11,r10
0:023>
ntdll!LdrpDispatchUserCallTargetES+0x21:
00007fff`ed876431 7316            jae     ntdll!LdrpDispatchUserCallTargetES+0x39 (00007fff`ed876449) [br=1]
0:023> u 7fff`ed876449
ntdll!LdrpDispatchUserCallTargetES+0x39:
00007fff`ed876449 41ba01000000    mov     r10d,1
00007fff`ed87644f e90cfeffff      jmp     ntdll!LdrpHandleInvalidUserCallTarget (00007fff`ed876260)
```

kernel32!GlobalLock failed to pass CFG check

CFG Improvement in WIP 15048

```
ntdll!LdrpDispatchUserCallTargetES:
00007fff`ed876410 4c8b1d594f0d00  mov     r11,qword ptr [ntdll!LdrSystemDllInitBlock+0xb0 (00007fff`ed94b370)]
00007fff`ed876417 4c8bd0          mov     r10,rax
00007fff`ed87641a 49c1ea09        shr     r10,9
00007fff`ed87641e 4f8b1cd3        mov     r11,qword ptr [r11+r10*8]
00007fff`ed876422 4c8bd0          mov     r10,rax
00007fff`ed876425 49c1ea03        shr     r10,3
00007fff`ed876429 a80f            test    al,0Fh
00007fff`ed87642b 7509            jne     ntdll!LdrpDispatchUserCallTargetES+0x26 (00007fff`ed876436)
00007fff`ed87642d 4d0fa3d3        bt      r11,r10
00007fff`ed876431 7316            jae     ntdll!LdrpDispatchUserCallTargetES+0x39 (00007fff`ed876449) [br=1]
00007fff`ed876433 48ffe0          jmp     rax
00007fff`ed876436 4d0fa3d3        bt      r11,r10
```

Command

```
0:057> r rax=ntdll!RtlProtectHeap
0:057> t
ntdll!LdrpDispatchUserCallTargetES+0x7:
00007fff`ed876417 4c8bd0          mov     r10,rax
0:057>
ntdll!LdrpDispatchUserCallTargetES+0xa:
00007fff`ed87641a 49c1ea09        shr     r10,9
0:057>
ntdll!LdrpDispatchUserCallTargetES+0xe:
00007fff`ed87641e 4f8b1cd3        mov     r11,qword ptr [r11+r10*8] ds:00007ff5`ff880738=0000000000000000
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x12:
00007fff`ed876422 4c8bd0          mov     r10,rax
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x15:
00007fff`ed876425 49c1ea03        shr     r10,3
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x19:
00007fff`ed876429 a80f            test    al,0Fh
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x1b:
00007fff`ed87642b 7509            jne     ntdll!LdrpDispatchUserCallTargetES+0x26 (00007fff`ed876436) [br=0]
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x1d:
00007fff`ed87642d 4d0fa3d3        bt      r11,r10
0:057>
ntdll!LdrpDispatchUserCallTargetES+0x21:
00007fff`ed876431 7316            jae     ntdll!LdrpDispatchUserCallTargetES+0x39 (00007fff`ed876449) [br=1]
0:057> u 7fff`ed876449
ntdll!LdrpDispatchUserCallTargetES+0x39:
00007fff`ed876449 41ba01000000    mov     r10d,1
00007fff`ed87644f e90cfeffff      jmp     ntdll!LdrpHandleInvalidUserCallTarget (00007fff`ed876260)
```

ntdll!RtlProtectHeap failed to pass CFG check

Significant improvement was made on CFG after WIP build 14986, many previous valid call targets now are no longer valid (see previous two slides)! Obviously, MS has been working hard to make its CFG implementation more fine-grained to defeat calling function out-of-context. As a consequence, we'll have to find new way to exploit memory vulnerability, and this is really where data-only attack come into play!

"Microsoft Mitigation Bypass Bounty"

Mitigation	In scope	Out of scope
Control Flow Guard(CFG)	Techniques that make it possible to gain control of the instruction pointer through an indirect call in a process that has enabled CFG.	• Hijacking control flow viare turn address corruption • Bypasses related to limitations of coarse-grained CFI (e.g. calling functions out of context) • Leveraging non-CFG images • Bypasses that rely on modifying or corrupting read-only memory

Data-only Attack

- The principle of data-only attack
  - Change the program's default behavior by manipulating the data it depends on without divert the program's execution control flow. Change in program's data may lead to the change of program's original execution path, therefore achieve some powerful things, such as bypassing certain restriction or protection. Please note that the change in program's code path caused by the change of data is still within the program's normal logic, so it won't have problem when CFI is enforced.

- Example of successful data-oriented attack
  - Vital Point Strike (JavaScript god mode)
  - EMET bypass (Replacing EnableProtectionPtr in CONFIG_STRUCT)

- What kind of data will be targeted
  - Any data that can be leveraged to alter the program's default behavior will be of interest to the data-oriented attack, such as certain global flag in data section, or field in an object etc.
  - Unprotected .data section, unprotected heap/private data, stack.

- How to perform the data-only attack
  - Most data-oriented attacks require the ability of arbitrary address read/write.
  - Being able to accurately locating the targeted data at runtime is the key to success.
  - The existence of certain data is transient or time-sensitive, so timing may play an important role in these cases.

CFG Bypass: Abuse WARP Shader JIT

- We discovered a new CFG bypass method and reported it to Microsoft in early 2016, then Microsoft fixed the issue in its Jun'16 Patch Tuesday release.
  - "Microsoft's June Patch Kills Potential CFG Bypass"
  - "JIT Spraying Never Dies - Bypass CFG By Leveraging WARP Shader JIT Spraying" Full slides of our presentation on XCon 2016

- In later research, we found the June fix can be easily bypassed using data-only attack (by simply flipping a global variable). We reported this to Microsoft again, and finally the issue got completely fixed in Oct'16 ☺.

- Let's try reproducing the whole story by doing a comparison on different versions involved.

The 1st Version

CFG was not even added in version 10.0.10586.0 of d3d10warp.dll

```
.text:1009D62C                 push    ebp
.text:1009D62D                 mov     ebp, esp
.text:1009D62F                 push    ecx
.text:1009D630                 push    ebx
.text:1009D631                 mov     bl, [ebp+arg_0]
.text:1009D634                 push    esi
.text:1009D635                 push    edi
.text:1009D636                 push    2
.text:1009D638                 pop     eax
.text:1009D639                 push    20h
.text:1009D63B                 mov     edi, ecx
.text:1009D63D                 test    bl, bl
.text:1009D63F                 pop     ecx
.text:1009D640                 cmovnz  eax, ecx
.text:1009D643                 mov     esi, edx
.text:1009D645                 lea     ecx, [ebp+flOldProtect]
.text:1009D648                 push    ecx             ; lpflOldProtect
.text:1009D649                 push    eax             ; flNewProtect
.text:1009D64A                 push    esi             ; dwSize
.text:1009D64B                 push    edi             ; lpAddress
.text:1009D64C                 call    ds:__imp__VirtualProtect@16 ; VirtualProtect(x,x,x,x)
.text:1009D652                 test    eax, eax
.text:1009D654                 jz      short loc_1009D678
.text:1009D656                 test    bl, bl
.text:1009D658                 jz      short loc_1009D669
.text:1009D65A                 push    esi             ; dwSize
.text:1009D65B                 push    edi             ; lpBaseAddress
.text:1009D65C                 call    ds:__imp__GetCurrentProcess@0 ; GetCurrentProcess()
.text:1009D662                 push    eax             ; hProcess
.text:1009D663                 call    ds:__imp__FlushInstructionCache@12 ; FlushInstructionCache(x,x,x)
.text:1009D669
```

The 2nd Version

A writable global variable is used to keep the current status of CFG in version 10.0.10586.420 of d3d10warp.dll

```
.text:1009D644 loc_1009D644:                          ; CODE XREF: WarpPlatform::ProtectCodePages(void *,uint,bool)+13↑j
.text:1009D644                 cmp     ?gIsCFGEnabled@@3_NA, 0 ; bool gIsCFGEnabled
.text:1009D64B                 mov     ecx, 40000020h
.text:1009D650                 push    20h
.text:1009D652                 pop     eax
.text:1009D653                 cmovnz  eax, ecx
.text:1009D656
.text:1009D656 loc_1009D656:                          ; CODE XREF: WarpPlatform::ProtectCodePages(void *,uint,bool)+18↑j
.text:1009D656                 push    eax             ; flProtect
.text:1009D657                 push    1000h           ; flAllocationType
.text:1009D65C                 push    edi             ; dwSize
.text:1009D65D                 push    esi             ; lpAddress
.text:1009D65E                 call    ds:__imp__VirtualAlloc@16 ; VirtualAlloc(x,x,x,x)
.text:1009D664                 test    eax, eax
.text:1009D666                 jnz     short loc_1009D66C
.text:1009D668                 xor     al, al
.text:1009D66A                 jmp     short loc_1009D6BB
```

```
.text:1009D68A                 cmp     ?gIsCFGEnabled@@3_NA, 0 ; bool gIsCFGEnabled
.text:1009D691                 jz      short loc_1009D6B3
.text:1009D693                 and     [ebp+var_8], 0
.text:1009D697                 lea     eax, [ebp+var_8]
.text:1009D69A                 push    eax
.text:1009D69B                 push    ebx
.text:1009D69C                 push    ?gPageSize@@3IA ; uint gPageSize
.text:1009D6A2                 mov     [ebp+var_4], ebx
.text:1009D6A5                 push    esi
.text:1009D6A6                 call    ds:__imp__GetCurrentProcess@0 ; GetCurrentProcess()
.text:1009D6AC                 push    eax
.text:1009D6AD                 call    ds:__imp__SetProcessValidCallTargets@20 ; SetProcessValidCallTargets(x,x,x,x,x)
.text:1009D6B3
```

The 3rd Version

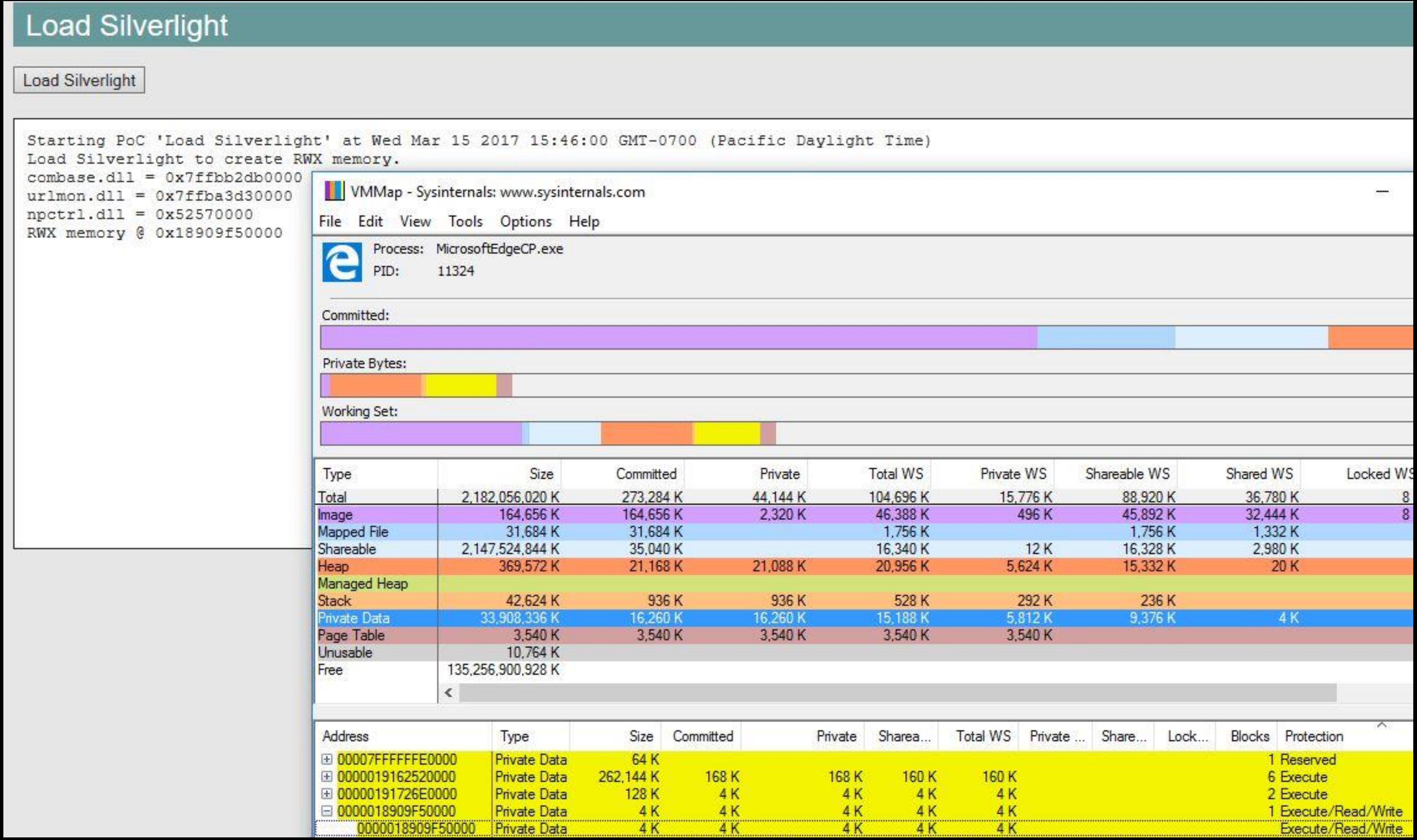
In version 10.0.10586.672 of d3d10warp.dll, instead of using global variable, a new function was introduced, and it'll be called each time when the status of CFG needs to be verified

```
.text:1009D63A                  call    IsCFGEnabled
.text:1009D63F                  mov     bl, byte ptr [ebp+arg_0]
.text:1009D642                  mov     [ebp+var_1], al
.text:1009D645                  test    bl, bl
.text:1009D647                  jnz     short loc_10
.text:1009D649                  push    2
.text:1009D64B                  pop     esi
.text:1009D64C                  jmp     short loc_10
.text:1009D64E ; -----------------------------------
.text:1009D64E
.text:1009D64E loc_1009D64E:
.text:1009D64E                  push    20h
.text:1009D650                  test    al, al
.text:1009D652                  mov     ecx, 4000002
.text:1009D657                  pop     esi
.text:1009D658                  cmovnz  esi, ecx
```

```
.text:101B7394 IsCFGEnabled    proc near               ; CODE XREF: WarpPlatform::ProtectCodePages(void *,uint,bool)+10↑p
.text:101B7394
.text:101B7394 var_4           = dword ptr -4
.text:101B7394
.text:101B7394                 xchg    ax, ax
.text:101B7396                 push    ebp
.text:101B7397                 mov     ebp, esp
.text:101B7399                 push    ecx
.text:101B739A                 push    4
.text:101B739C                 lea     eax, [ebp+var_4]
.text:101B739F                 push    eax
.text:101B73A0                 push    7
.text:101B73A2                 call    ds:__imp__GetCurrentProcess@0 ; GetCurrentProcess()
.text:101B73A8                 push    eax
.text:101B73A9                 call    ds:__imp__GetProcessMitigationPolicy@16 ; GetProcessMitigationPolicy(x,x,x,x)
.text:101B73AF                 test    eax, eax
.text:101B73B1                 jz      short loc_101B73BD
.text:101B73B3                 mov     eax, [ebp+var_4]
.text:101B73B6                 and     eax, 1
.text:101B73B9                 mov     esp, ebp
.text:101B73BB                 pop     ebp
.text:101B73BC                 retn
.text:101B73BD ; ---------------------------------------------------------------------------
.text:101B73BD
.text:101B73BD loc_101B73BD:                           ; CODE XREF: IsCFGEnabled+1D↑j
.text:101B73BD                 xor     al, al
.text:101B73BF                 mov     esp, ebp
.text:101B73C1                 pop     ebp
.text:101B73C2                 retn
.text:101B73C2 IsCFGEnabled    endp
```

That’s it? Is data-only attack always as simple as flipping a bit? Of course NOT! Let’s look at another more complicated example and see how pure data manipulation can be used to complete an impossible task!

CFG/DEP Bypass: Load Disabled ActiveX Extension



Load Silverlight

Load Silverlight

```
Starting PoC 'Load Silverlight' at Wed Mar 15 2017 15:46:00 GMT-0700 (Pacific Daylight Time)
Load Silverlight to create RWX memory.
combase.dll = 0x7ffbb2db0000
urlmon.dll = 0x7ffba3d30000
npctrl.dll = 0x52570000
RWX memory @ 0x18909f50000
```

VMMap - Sysinternals: www.sysinternals.com

File Edit View Tools Options Help

Process: MicrosoftEdgeCP.exe
PID: 11324

Committed:

Private Bytes:

Working Set:

Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Shared WS	Locked WS
Total	2,182,056,020 K	273,284 K	44,144 K	104,696 K	15,776 K	88,920 K	36,780 K	8
Image	164,656 K	164,656 K	2,320 K	46,388 K	496 K	45,892 K	32,444 K	8
Mapped File	31,684 K	31,684 K		1,756 K		1,756 K	1,332 K	
Shareable	2,147,524,844 K	35,040 K		16,340 K	12 K	16,328 K	2,980 K	
Heap	369,572 K	21,168 K	21,088 K	20,956 K	5,624 K	15,332 K	20 K	
Managed Heap								
Stack	42,624 K	936 K	936 K	528 K	292 K	236 K		
Private Data	33,908,336 K	16,260 K	16,260 K	15,188 K	5,812 K	9,376 K	4 K	
Page Table	3,540 K	3,540 K	3,540 K	3,540 K	3,540 K			
Unusable	10,764 K							
Free	135,256,900,928 K							

Address	Type	Size	Committed	Private	Sharea...	Total WS	Private ...	Share...	Lock...	Blocks	Protection
⊞ 00007FFFFFFE0000	Private Data	64 K								1	Reserved
⊞ 0000019162520000	Private Data	262,144 K	168 K	168 K	160 K	160 K				6	Execute
⊞ 00000191726E0000	Private Data	128 K	4 K	4 K	4 K	4 K				2	Execute
⊟ 0000018909F50000	Private Data	4 K	4 K	4 K	4 K	4 K				1	Execute/Read/Write
0000018909F50000	Private Data	4 K	4 K	4 K	4 K	4 K					Execute/Read/Write

Edge Stops the Support for ActiveX (Including Silverlight)

- ActiveX (including Silverlight browser plugin) was no longer supported in Edge .

Microsoft Edge and Silverlight

Support for ActiveX has been discontinued in Microsoft Edge, and that includes removing support for Silverlight. The reasons for this have been discussed in previous blogs and include the emergence of viable and secure media solutions based on HTML5 extensions. Microsoft continues to support Silverlight, and Silverlight out-of-browser apps can continue to use it. Silverlight will also continue to be supported in Internet Explorer 11, so sites continue to have Silverlight options in Windows 10. At the same time, we encourage companies that are using Silverlight for media to begin the transition to DASH/MSE/CENC/EME based designs and to follow a single, DRM-interoperable encoding work flow enabled by CENC. This represents the most broadly interoperable solution across browsers, platforms, content and devices going forward.

- Besides ActiveX, BHO, VML and VBScript were also removed in Edge, and such removal significantly reduced the attack surface of Edge.

Force Silverlight to Be Loaded in Edge

- In order to force Silverlight Plugin to be loaded in Edge, two places need to be modified to bypass the restrictions.
  - The corresponding feature entry in feature cache (urlmon!g_pFeatureCache)
    - Two security checks are implemented when an ActiveX object is instantiated: one is edgehtml!COleSite::CreateObjectSecurityChecks, the other is urlmon!IsActiveXExtensionAllowed. Both security checks call a same function urlmon!CoInternetExtensionAllowedForUri to verify if the ActiveX object is safe to load.
    - urlmon!CoInternetExtensionAllowedForUri checks the 14^{th} entry in feature cache to determine whether internet extension is allowed or not, and the feature cache is managed by urlmon.dll.
  - Search path flag for DLL loading (combase!GetLoadLibraryAlteredSearchPathFlag)
    - For Silverlight plugin module (npctrl.dll) to be successfully loaded by Edge, the dwFlags argument of LoadLibraryExW must be set to LOAD_WITH_ALTERED_SEARCH_PATH.
    - Upon COM module loading, a global variable in combase.dll will be used as the 3^{rd} argument in the call to LoadLibraryExW.

The First Security Check: edgehtml!COleSite::CreateObjectSecurityChecks

```
0:019> kv
 # Child-SP          RetAddr           : Args to Child                                                           : Call Site
00 000000e3`764f6d70 00007ff9`c671064e : 00000000`00000055 000001ff`82378220 00000000`00000001 00007ff9`b5af07c0 : urlmon!CoInternetIsFeatureEnabled+0x4d
01 000000e3`764f6db0 00007ff9`b576dd02 : 00000000`00000055 000000e3`764f6ec1 000000e3`764f7030 00000000`00000001 : urlmon!CoInternetExtensionAllowedForUri+0x5e
02 000000e3`764f6e00 00007ff9`b576eb64 : 00000000`00000055 000001ff`82350c00 00000000`00000000 00000000`00000055 : edgehtml!COleSite::CreateObjectSecurityChecks+0x8a
03 000000e3`764f6f10 00007ff9`b588613c : 00000207`962a9340 00000207`96296580 000000e3`764f7090 00000207`962c0000 : edgehtml!COleSite::CreateObject+0x154
04 000000e3`764f6f90 00007ff9`b576f30b : 00000207`962c0000 00000207`962c0000 00000000`000002eb 000000e3`764f9158 : edgehtml!CObjectElement::FinalCreateObject+0x32c
05 000000e3`764f90e0 00007ff9`b5a06d2b : 00000000`00000000 000001ff`82303cf0 00000000`ffffffff 000001ff`82303cf0 : edgehtml!CObjectElement::CreateObject+0x1b3
06 000000e3`764f91d0 00007ff9`b5bf2d58 : 00000000`000003b8 00000207`82502250 00000000`00000000 00000000`ffffffff : edgehtml!CHtmObject10ParseCtx::Execute+0x2b
07 000000e3`764f9200 00007ff9`b5b92814 : 00000000`ffffffff 00000000`00000000 000001ff`8232a9e0 00000000`00000000 : edgehtml!CHtmParseBase::Execute+0x228
08 000000e3`764f9290 00007ff9`b5b92250 : 000001ff`82356000 000001ff`82356000 000001ff`82356000 000001ff`8232a9e0 : edgehtml!CHtmPost::Broadcast+0x54
09 000000e3`764f92d0 00007ff9`b5ba22a7 : 000001ff`8232a9e0 000001ff`82356000 00000000`00000800 000001ff`8232a9e0 : edgehtml!CHtmPost::Exec+0x180
0a 000000e3`764f94b0 00007ff9`b5b8b20c : 000001ff`82350c00 000000e3`764f9589 000000e3`764f9670 000000e3`764f9670 : edgehtml!CHtmPost::PerformSyncParse+0x14b
0b 000000e3`764f9500 00007ff9`b5b8a5c4 : 00000000`00004003 00000000`00000008 00000000`00000008 00007ff9`00000001 : edgehtml!CDoc::ParseHtmlStream+0x258
…
```

The Second Security Check: urlmon!IsActiveXExtensionAllowed

```
0:019> kv
 # Child-SP          RetAddr           : Args to Child                                                           : Call Site
00 000000e3`764f5a70 00007ff9`c671064e : 00000000`00000000 00007ff9`c672614a 00000000`00000001 00007ff9`b6a11878 : urlmon!CoInternetIsFeatureEnabled+0x4d
01 000000e3`764f5ab0 00007ff9`c671033a : 00000000`00000000 00007ff9`b6a11878 000000e3`764f5c68 000000e3`764f5c68 : urlmon!CoInternetExtensionAllowedForUri+0x5e
02 000000e3`764f5b00 00007ff9`c6710051 : 00000000`00000000 00000207`857c10a0 000001ff`80347d70 00007ff9`c6710306 : urlmon!IsActiveXExtensionAllowed+0x26
03 000000e3`764f5b30 00007ff9`c670f059 : 00000207`83113e50 000000e3`764f5c80 00000207`857c10a0 00000000`00000000 : urlmon!CoGetClassObjectForObjectBinding+0x75
04 000000e3`764f5b80 00007ff9`b5885967 : 000001ff`80347d70 00000000`00000004 00000207`9628f818 00000207`9628f818 : urlmon!CoGetClassObjectFromURLInternal+0x2d9
05 000000e3`764f6d30 00007ff9`b576ef69 : 00000207`00000000 00000207`00000000 000000e3`00000000 000001ff`00000000 : edgehtml!CCodeLoad::BindToObject+0x3eb
06 000000e3`764f6ea0 00007ff9`b576ebcb : 00000000`00000055 000001ff`82350c00 00000000`00000000 00000000`00000055 : edgehtml!CCodeLoad::Init+0x2b9
07 000000e3`764f6f10 00007ff9`b588613c : 00000207`00000001 00000207`96296580 000000e3`764f7090 00000207`962c0000 : edgehtml!COleSite::CreateObject+0x1bb
08 000000e3`764f6f90 00007ff9`b576f30b : 00000207`962c0000 00000207`962c0000 00000000`000002eb 000000e3`764f9158 : edgehtml!CObjectElement::FinalCreateObject+0x32c
09 000000e3`764f90e0 00007ff9`b5a06d2b : 00000000`00000000 000001ff`82303cf0 00000000`ffffffff 000001ff`82303cf0 : edgehtml!CObjectElement::CreateObject+0x1b3
0a 000000e3`764f91d0 00007ff9`b5bf2d58 : 00000000`000003b8 00000207`82502250 00000000`00000000 00000000`ffffffff : edgehtml!CHtmObject10ParseCtx::Execute+0x2b
0b 000000e3`764f9200 00007ff9`b5b92814 : 00000000`ffffffff 00000000`00000000 000001ff`8232a9e0 00000000`00000000 : edgehtml!CHtmParseBase::Execute+0x228
…
```

urlmon!CoInternetExtensionAllowedForUri

Both security checks call urlmon!CoInternetExtensionAllowedForUri to determine whether the ActiveX object is allowed to be loaded. This function calls urlmon!CoInternetIsFeatureEnabled to check the status of 14th feature in feature cache.

```
.text:00000001800105F0 ; int __fastcall CoInternetExtensionAllowedForUri(_GUID *rclsid, unsigned int dwExtensionType, IUri *pUri, int fAllowWebBrowserInImmersiveMode)
.text:00000001800105F0          xchg    ax, ax
.text:00000001800105F2          nop     dword ptr [rax+00000000h]
.text:00000001800105F9          mov     r11, rsp
…
.text:0000000180010643          lea     edx, [rdi+2]    ; dwFlags
.text:0000000180010646          lea     ecx, [rdi+0Dh]  ; FeatureEntry        // ecx = 0x0d, test the 14th feature entry
.text:0000000180010649          call    CoInternetIsFeatureEnabled
.text:000000018001064E          lea     ebx, [rdi+1]
.text:0000000180010651          cmp     eax, ebx
.text:0000000180010653          jz      loc_180010703…
```

urlmon!CoInternetIsFeatureEnabled

urlmon!CoInternetIsFeatureEnabled tests the specified feature entry and returns whether the corresponding feature is enabled or not.

```
.text:00000001800260A0 ; HRESULT __stdcall CoInternetIsFeatureEnabled(INTERNETFEATURELIST FeatureEntry, DWORD dwFlags)
...
.text:00000001800260CA        mov    rdi, cs:?g_pFeatureCache@@3PEAVCFeatureCache@@EA ; // pointer to feature cache
...
.text:00000001800260DA        mov    ecx, esi
.text:00000001800260DC        mov    ebx, 1
.text:00000001800260E1        shl    rbx, cl // compute the entry to test
...
.text:00000001800260E9        test   [rdi+8], rbx // test the corresponding bit, in this case rbx=0x2000
.text:00000001800260ED        mov    ebx, 0
..text:00000001800260F2       setz   bl
.text:00000001800260F5        test   ebx, ebx
.text:00000001800260F7        js     loc_18002619D   ; jumptable 000000018002616D default case
```

urlmon!g_pFeatureCache

- The feature cache is not protected for write, and the default feature entry for internet extension is 1, which means ActiveX object is NOT allowed. Changing it to 0 allows the Silverlight to be loaded.

```
0:030> d urlmon!g_pFeatureCache
00007fff`584fbf30  e0 d5 af b4 c5 01 00 00-00 00 00 00 00 00 00 00  ................
00007fff`584fbf40  c0 be a2 b4 c5 01 00 00-00 00 00 00 00 00 00 00  ................
00007fff`584fbf50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`584fbf60  00 00 00 00 00 00 00 00-ff ff ff ff ff ff ff ff  ................
00007fff`584fbf70  ff ff ff ff 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`584fbf80  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`584fbf90  3e 00 00 00 00 00 00 00-ff ff ff ff ff ff ff ff  >...............
00007fff`584fbfa0  ff ff ff ff 00 00 00 00-00 00 00 00 00 00 00 00  ................
0:030> d 1c5b4afd5e0
000001c5`b4afd5e0  16 51 20 00 00 00 00 00-7e bf ff 1f 00 00 00 00  .Q .....~.......
000001c5`b4afd5f0  ff ff ff ff ff ff ff ff-ff ff ff ff 00 00 00 00  ................
000001c5`b4afd600  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
000001c5`b4afd610  00 00 00 00 00 00 00 00-00 00 00 00 00 00 08 00  ................
000001c5`b4afd620  90 97 e8 48 ff 7f 00 00-d8 a2 f2 48 ff 7f 00 00  ...H.......H....
000001c5`b4afd630  00 00 00 00 00 00 00 00-ed 9c bb a6 92 e2 3c 47  ..............<G
000001c5`b4afd640  91 dd 49 f2 a0 4a 4a bd-00 00 00 00 c0 01 00 00  ..I..JJ.........
000001c5`b4afd650  ed 9c bb a6 92 e2 3c 47-91 dd 49 f2 a0 4a 4a bd  ......<G..I..JJ.
0:030> !address 1c5b4afd5e0

Usage:                  Heap
Base Address:           000001c5`b4af9000
End Address:            000001c5`b4b00000
Region Size:            00000000`00007000 (  28.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000004          PAGE_READWRITE
Type:                   00020000          MEM_PRIVATE
Allocation Base:        000001c5`b4a00000
Allocation Protect:     00000004          PAGE_READWRITE
More info:              heap owning the address: !heap 0x1c5b49b0000
More info:              segment heap segment


Content source: 1 (target), length: a20
0:030> ? poi(poi(urlmon!g_pFeatureCache)+8) & (1 << 0x0d)
Evaluate expression: 8192 = 00000000`00002000
```

Search Path Flag for DLL Loading: combase!CClassCache::CDllPathEntry::LoadDll

```
0:019> kv
 # Child-SP          RetAddr           : Args to Child                                                           : Call Site
00 000000e3`764f39d0 00007ff9`d1d237b4 : 00300045`00380032 00007ff9`d4805136 00390039`002d0039 0042002d`00450046 : combase!GetLoadLibraryAlteredSearchPathFlag+0x13
01 000000e3`764f3a10 00007ff9`d1d235c4 : 000000e3`764f3b60 000000e3`764f3ae0 000000e3`764f3b40 00000000`00000000 : combase!CClassCache::CDllPathEntry::LoadDll+0x48
02 000000e3`764f3a70 00007ff9`d1d74e59 : 000000e3`764f4290 000000e3`764f3e60 000000e3`764f3e60 00000000`00000000 : combase!CClassCache::CDllPathEntry::Create+0x58
03 000000e3`764f3b10 00007ff9`d1d3adb2 : 00000207`831dcf60 00000207`831dcf60 00000000`00000000 000000e3`764f4290 : combase!CClassCache::CClassEntry::CreateDllClassEntry+0xf5
04 000000e3`764f3de0 00007ff9`d1d7772d : 00000000`00000000 00007ff9`00000000 000000e3`00000030 000001ff`80390840 : combase!CClassCache::GetClassObjectActivator+0x7f2
05 000000e3`764f41b0 00007ff9`d1d77258 : 000001ff`80390840 000000e3`764f4c28 00000000`00000000 00000000`00000000 : combase!CClassCache::GetClassObject+0x4d
06 (Inline Function) --------`-------- : --------`-------- --------`-------- --------`-------- --------`-------- : combase!CCGetClassObject+0x52 (Inline Function @ 00007ff9`d1d77258)
07 000000e3`764f4220 00007ff9`d1d7c841 : 00000207`00000001 000000e3`764f5100 000000e3`764f49f0 000000e3`764f4c28 :combase!CServerContextActivator::CreateInstance+0x178
08 000000e3`764f4320 00007ff9`d1d779a9 : 00000000`00000000 00007ff9`d18e502d 000000e3`764f4c28 00000000`00000000 :
combase!ActivationPropertiesIn::DelegateCreateInstance+0xe1
09 000000e3`764f43b0 00007ff9`d1d77eab : 00000000`00000000 00000000`00000000 000000e3`764f49f0 00000000`00000000 : combase!CApartmentActivator::CreateInstance+0xc9
0a 000000e3`764f4460 00007ff9`d1d77f69 : 00007ff9`d1f92380 00000000`00000000 000000e3`764f54d0 000000e3`764f4580 : combase!CProcessActivator::CCICallback+0x7b
0b 000000e3`764f44a0 00007ff9`d1d78037 : 00007ff9`d1f92370 000000e3`764f54d0 00000000`00000000 00000000`00000000 : combase!CProcessActivator::AttemptActivation+0x49
0c 000000e3`764f44f0 00007ff9`d1d782f4 : 00007ff9`d1f92370 00000000`d1f264f0 00000000`00000000 00007ff9`00000001 : combase!CProcessActivator::ActivateByContext+0xb7
0d 000000e3`764f4580 00007ff9`d1d7c88d : 000000e3`764f4c28 000000e3`764f54d0 000000e3`764f49f0 00000000`00000000 : combase!CProcessActivator::CreateInstance+0x94
```

Search Path Flag for DLL Loading (Continued)

```
0e 000000e3`764f45d0 00007ff9`d1d80132 : 000000e3`764f4c28 000000e3`00000000 000000e3`764f4690 000000e3`764f4c28 :
combase!ActivationPropertiesIn::DelegateCreateInstance+0x12d

0f 000000e3`764f4660 00007ff9`d1d7c84b : 000000e3`764f4c28 00000000`00000000 ffffff1c`89b0b610 00000000`00000000 : combase!CClientContextActivator::CreateInstance+0x152

10 000000e3`764f4910 00007ff9`d1d538fe : 00000000`00000000 000000e3`764f5430 00000000`00000000 00000000`00000000 :
combase!ActivationPropertiesIn::DelegateCreateInstance+0xeb

11 000000e3`764f49a0 00007ff9`d1d52b80 : 00000000`0000002c 00000000`00000008 00000000`00000002 00000000`000000f6 : combase!ICoCreateInstanceEx+0xc0e

12 000000e3`764f56f0 00007ff9`d1d5299c : 000001ff`803dcb90 00007ff9`b5c385fa 00007ff9`0000007a 000000e3`00000000 : combase!CComActivator::DoCreateInstance+0x140

13 (Inline Function) --------`-------- : --------`-------- --------`-------- --------`-------- --------`-------- : combase!CoCreateInstanceEx+0x97 (Inline Function @ 00007ff9`d1d5299c)

14 000000e3`764f5810 00007ff9`b5fbe300 : 000001ff`82350c00 000001ff`803a6eb0 000000e3`764f5910 000000e3`00020001 : combase!CoCreateInstance+0xcc

15 000000e3`764f58b0 00007ff9`b5de107c : 00000000`00000000 00000000`00000000 000001ff`82350c00 000001ff`82350c00 : edgehtml!CMarkup::_BuildExtensionValidatorsList+0xb4

16 000000e3`764f5920 00007ff9`b58928f5 : 000001ff`82350c00 00000000`00000000 00004ff9`00000002 00000000`00000000 : edgehtml!CMarkup::ValidateExtension+0x54e744

17 000000e3`764f59b0 00007ff9`c6710265 : 000000e3`764f5c68 00007ff9`b59d7bad 00007ff9`c682fd38 00000000`00000000 : edgehtml!CCodeLoad::OnProgress+0x135

18 000000e3`764f5ac0 00007ff9`c67101d5 : 000001ff`803a6eb0 00000000`00000000 000000e3`764f5c68 00000000`00000001 : urlmon!CExtensionValidationProxy::_SendCLSIDToBSC+0x59
19 000000e3`764f5b00 00007ff9`c6710084 : 00000207`9628f818 00000000`00000000 00000207`9628f818 00007ff9`00000000 :
urlmon!CExtensionValidationProxy::_SendCLSIDToBSCInBindCtx+0x75

1a 000000e3`764f5b30 00007ff9`c670f059 : 00000207`83113e50 000000e3`764f5c80 00000207`857c10a0 00000000`00000000 : urlmon!CoGetClassObjectForObjectBinding+0xa8

1b 000000e3`764f5b80 00007ff9`b5885967 : 000001ff`80347d70 00000000`00000004 00000207`9628f818 00000207`9628f818 : urlmon!CoGetClassObjectFromURLInternal+0x2d9

1c 000000e3`764f6d30 00007ff9`b576ef69 : 00000207`00000000 00000207`00000000 000000e3`00000000 000001ff`00000000 : edgehtml!CCodeLoad::BindToObject+0x3eb

1d 000000e3`764f6ea0 00007ff9`b576ebcb : 00000000`00000055 000001ff`82350c00 00000000`00000000 00000000`00000055 : edgehtml!CCodeLoad::Init+0x2b9

1e 000000e3`764f6f10 00007ff9`b588613c : 00000207`00000001 00000207`96296580 000000e3`764f7090 00000207`962c0000 : edgehtml!COleSite::CreateObject+0x1bb

…
```

combase!CClassCache::CDllPathEntry::LoadDll

Before calling LoadlibraryExW, combase!CClassCache::CDllPathEntry::LoadDll calls GetLoadLibraryAlteredSearchPathFlag to obtain the search path flag for DLL loading.

```
.text:000000018004376C ; HRESULT __fastcall CClassCache::CDllPathEntry::LoadDll(DLL_INSTANTIATION_PROPERTIES *dip, HRESULT (__fastcall **pfnGetClassObject)(_GUID *, _GUID *, void **), HRESULT (__fastcall **pfnGetActivationFactory)(HSTRING__ *, IActivationFactory **), HRESULT (__fastcall **pfnDllCanUnload)(), HINSTANCE__ **hDll)

.text:000000018004376C        xchg    ax, ax

.text:000000018004376E        nop     dword ptr [rax+00000000h]

.text:0000000180043775        mov     [rsp+arg_0], rbx

...

.text:00000001800437AF        call    ?GetLoadLibraryAlteredSearchPathFlag@@YAKXZ ; GetLoadLibraryAlteredSearchPathFlag(void)

.text:00000001800437B4        mov     rdi, [rsp+58h+hDll]

.text:00000001800437BC        mov     r8d, eax      ; dwFlags            // the flag comes from the above call to GetLoadLibraryAlteredSearchPathFlag

.text:00000001800437BF        mov     r9, rdi       ; phMod

.text:00000001800437C2        mov     rdx, rbx      ; pwszFileName

.text:00000001800437C5        call    ?LoadLibraryWithLogging@@YAJW4LoadOrFreeWhy@@PEBGKPEAPEAUHINSTANCE__@@@Z ; LoadLibraryWithLogging(LoadOrFreeWhy,ushort const *,ulong,HINSTANCE__ * *)        // call to kernelbase!LoadLibraryExW

.text:00000001800437CA        mov     rcx, [rdi]
```

combase!GetLoadLibraryAlteredSearchPathFlag

combase!GetLoadLibraryAlteredSearchPathFlag gets the search path flag for DLL loading from a global variable of combase.dll.

```
.text:0000000180043944 ; unsigned int __fastcall GetLoadLibraryAlteredSearchPathFlag()
.text:0000000180043944
.text:0000000180043944                 xchg    ax, ax
.text:0000000180043946                 nop     dword ptr [rax+00000000h]
.text:000000018004394D                 sub     rsp, 38h
.text:0000000180043951                 mov     eax, cs:g_LoadLibraryAlteredSearchPathFlag   // the flag is kept in a global variable of combase.dll
.text:0000000180043957                 cmp     eax, 0FFFFFFFFh
.text:000000018004395A                 jz      short loc_180043965
.text:000000018004395C
.text:000000018004395C loc_18004395C:                          ; CODE XREF: GetLoadLibraryAlteredSearchPathFlag(void)+80j
.text:000000018004395C                                         ; GetLoadLibraryAlteredSearchPathFlag(void)+F9CDEj
.text:000000018004395C                 add     rsp, 38h
.text:0000000180043960                 jmp     __guard_ss_common_verify_stub
```

combase!g_LoadLibraryAlteredSearchPathFlag

- combase!g_LoadLibraryAlteredSearchPathFlag sits in the data section of combase.dll, so it’s open for write. By default, this variable is set to 0, and changes it to 8 (LOAD_WITH_ALTERED_SEARCH_PATH) will let Silverlight plugin (npctrl.dll) to be loaded successfully.

```
0:017> d combase!g_LoadLibraryAlteredSearchPathFlag
00007fff`677b2690  00 00 00 00 00 00 00 00-5c 03 00 00 00 00 00 00  ........\.......
00007fff`677b26a0  ff ff ff ff 80 a9 03 00-f4 01 00 00 00 00 00 00  ................
00007fff`677b26b0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`677b26c0  00 00 00 00 00 00 00 00-ff ff ff ff ff ff ff ff  ................
00007fff`677b26d0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`677b26e0  00 00 60 00 00 00 00 00-03 00 00 00 00 00 00 00  ..`.............
00007fff`677b26f0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00007fff`677b2700  00 00 60 00 00 00 00 00-d8 02 00 00 00 00 00 00  ..`.............
0:017> !address combase!g_LoadLibraryAlteredSearchPathFlag


Mapping file section regions...
Mapping module regions...
Mapping PEB regions...
Mapping TEB and stack regions...
Mapping heap regions...
Mapping page heap regions...
Mapping other regions...
Mapping stack trace database regions...
Mapping activation context regions...

Usage:                  Image
Base Address:           00007fff`677b0000
End Address:            00007fff`677b6000
Region Size:            00000000`00006000 (  24.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000004          PAGE_READWRITE
Type:                   01000000          MEM_IMAGE
Allocation Base:        00007fff`67500000
Allocation Protect:     00000080          PAGE_EXECUTE_WRITECOPY
Image Path:             C:\Windows\System32\combase.dll
Module Name:            combase
Loaded Image Name:      C:\Windows\System32\combase.dll
Mapped Image Name:
More info:              lmv m combase
More info:              !lmi combase
More info:              ln 0x7fff677b2690
More info:              !dh 0x7fff67500000


Content source: 1 (target), length: 970
```

Silverlight Browser Plugin Loaded

```
0:019> g
ModLoad: 00000000`75360000 00000000`754fa000   c:\Program Files\Microsoft Silverlight\5.1.50901.0\npctrl.dll
ntdll!NtMapViewOfSection+0x14:
00007ff9`d48960c4 c3              ret
0:019> kv
 # Child-SP          RetAddr           : Args to Child                                                           : Call Site
00 000000e3`764f3688 00007ff9`d48049ff : 00000000`00000000 00000000`00000000 00000000`00000038 00000000`0000007c : ntdll!NtMapViewOfSection+0x14
01 000000e3`764f3690 00007ff9`d480476e : 00000000`00000000 00000000`00000000 00000000`00000000 00007ff9`d48149be : ntdll!LdrpMinimalMapModule+0xd3
02 000000e3`764f3720 00007ff9`d4802f33 : 00000000`00000000 000000e3`764f37f9 00000000`00000000 01d20413`4a069200 : ntdll!LdrpMapDllWithSectionHandle+0x1e
03 000000e3`764f3790 00007ff9`d4800189 : 00000000`00000000 00000000`00000c60 00000000`00000c74 00000000`00000000 : ntdll!LdrpMapDllNtFileName+0x2eb
04 000000e3`764f3860 00007ff9`d48003b6 : 00000000`c0000135 00000207`85795340 000000e3`764f3b01 000000e3`764f3d00 : ntdll!LdrpMapDllFullPath+0xcd
05 000000e3`764f39e0 00007ff9`d4807466 : 00000207`85795340 000000e3`764f3b01 000000e3`764f3b10 000000e3`764f3b10 : ntdll!LdrpProcessWork+0x8e
06 000000e3`764f3a40 00007ff9`d48072f7 : 000000e3`764f3b10 000000e3`764f3ca0 00000000`00000000 00000000`00000001 : ntdll!LdrpLoadDllInternal+0x132
07 000000e3`764f3ac0 00007ff9`d48064dc : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000009 : ntdll!LdrpLoadDll+0x10b
08 000000e3`764f3c60 00007ff9`d18e2a94 : 000000e3`764f3fa0 00000000`00000008 00000000`00000008 00007ff9`c76b6630 : ntdll!LdrLoadDll+0x8c
09 000000e3`764f3d60 00007ff9`c76b6d54 : 000000e3`00000000 000000e3`764f3fa0 00000000`00000008 00000000`00000000 : KERNELBASE!LoadLibraryExW+0x184
0a 000000e3`764f3dd0 00007ff9`d1d23871 : 00000000`00000000 000000e3`764f3fa0 000000e3`764f3f80 00000000`007d0000 : EShims!NS_EdgeClLoadLibrary::APIHook_LoadLibraryExW+0x64
0b 000000e3`764f3e10 00007ff9`d1d237ca : 000000e3`764f3fa0 000000e3`764f3ef8 000000e3`764f3f80 000000e3`764f3f68 : combase!LoadLibraryWithLogging+0x35
0c 000000e3`764f3e50 00007ff9`d1d235c4 : 000000e3`764f3fa0 000000e3`764f3f20 000000e3`764f3f80 00000000`00000000 : combase!CClassCache::CDllPathEntry::LoadDll+0x5e
...
```

RWX Memory Created By Silverlight Plugin

```
npctrl!DllGetClassObject+0x3ec2
npctrl!DllGetClassObject+0x3dc3
npctrl!DllGetClassObject+0x3c6c
npctrl!DllGetClassObject+0x3c2b
npctrl!DllGetClassObject+0x4981
npctrl+0x32ce3
npctrl+0x32c96
edgehtml!BinaryControlHolder::CallDoVerb+0x90
edgehtml!COleSite::InPlaceActivate+0xe6
edgehtml!COleSite::TransitionTo+0x157
edgehtml!COleSite::TransitionToBaselineState+0x5d
edgehtml!COleSite::Notify_AfterInsertNode_Safe+0xfe
edgehtml!CObjectElement::Notify_AfterInsertNode_Safe+0x16
```

Disassembly

Offset: @$scopeip

```
00000000`6ef18d93 ff156785fcff      call    qword ptr [npctrl+0x1300 (00000000`6eee1300)]
00000000`6ef18d99 4885c0            test    rax,rax
00000000`6ef18d9c 755e              jne     npctrl!DllGetClassObject+0x3f0c (00000000`6ef18dfc)
00000000`6ef18d9e ba00100000        mov     edx,1000h
00000000`6ef18da3 448d4840          lea     r9d,[rax+40h]
00000000`6ef18da7 33c9              xor     ecx,ecx
00000000`6ef18da9 448bc2            mov     r8d,edx
00000000`6ef18dac ff15ce83fcff      call    qword ptr [npctrl+0x1180 (00000000`6eee1180)]
00000000`6ef18db2 488bd8            mov     rbx,rax
00000000`6ef18db5 4885c0            test    rax,rax
00000000`6ef18db8 0f8411520300      je      npctrl!DllCanUnloadNow+0x1b20f (00000000`6ef4dfcf)
00000000`6ef18dbe 8b00              mov     eax,dword ptr [rax]
00000000`6ef18dc0 488b0d01341300    mov     rcx,qword ptr [npctrl!InstallOfflineApp+0x34158 (00000000`6f04c1c8)]
00000000`6ef18dc7 ff153385fcff      call    qword ptr [npctrl+0x1300 (00000000`6eee1300)]
00000000`6ef18dcd 488bf8            mov     rdi,rax
00000000`6ef18dd0 4885c0            test    rax,rax
```

PAGE_EXECUTE_READWRITE

kernelbase!VirtualAlloc

Command

```
0:018> !address rax


Mapping file section regions...
Mapping module regions...
Mapping PEB regions...
Mapping TEB and stack regions...
Mapping heap regions...
Mapping page heap regions...
Mapping other regions...
Mapping stack trace database regions...
Mapping activation context regions...

Usage:                  <unknown>
Base Address:           0000017a`44900000
End Address:            0000017a`44901000
Region Size:            00000000`00001000 (    4.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000040          PAGE_EXECUTE_READWRITE
Type:                   00020000          MEM_PRIVATE
Allocation Base:        0000017a`44900000
Allocation Protect:     00000040          PAGE_EXECUTE_READWRITE
```

Locate RWX Memory

```
npctrl!DllGetClassObject+0x3f00
npctrl!DllGetClassObject+0x3dc3
npctrl!DllGetClassObject+0x3c6c
npctrl!DllGetClassObject+0x3c2b
npctrl!DllGetClassObject+0x4981
npctrl+0x32ce3
npctrl+0x32c96
edgehtml!BinaryControlHolder::CallDoVerb+0x90
edgehtml!COleSite::InPlaceActivate+0xe6
edgehtml!COleSite::TransitionTo+0x157
edgehtml!COleSite::TransitionToBaselineState+0x5d
edgehtml!COleSite::Notify_AfterInsertNode_Safe+0xfe
edgehtml!CObjectElement::Notify_AfterInsertNode_Safe+0x16
```

Disassembly

Offset: @$scopeip

```
00000000`6ef18dc7 ff153385fcff    call    qword ptr [npctrl+0x1300 (00000000`6eee1300)]
00000000`6ef18dcd 488bf8          mov     rdi,rax
00000000`6ef18dd0 4885c0          test    rax,rax
00000000`6ef18dd3 0f851c520300    jne     npctrl!DllCanUnloadNow+0x1b235 (00000000`6ef4dff5)
00000000`6ef18dd9 488dbbe00f0000  lea     rdi,[rbx+0FE0h]
00000000`6ef18de0 488b0de1331300  mov     rcx,qword ptr [npctrl!InstallOfflineApp+0x34158 (00000000`6f04c1c8)]
00000000`6ef18de7 488bd3          mov     rdx,rbx
00000000`6ef18dea ff150885fcff    call    qword ptr [npctrl+0x12f8 (00000000`6eee12f8)]
00000000`6ef18df0 4883c320        add     rbx,20h
00000000`6ef18df4 483bdf          cmp     rbx,rdi
00000000`6ef18df7 72e7            jb      npctrl!DllGetClassObject+0x3ef0 (00000000`6ef18de0)
00000000`6ef18df9 488bc3          mov     rax,rbx
00000000`6ef18dfc 488b5c2438      mov     rbx,qword ptr [rsp+38h]
00000000`6ef18e01 4883c420        add     rsp,20h
00000000`6ef18e05 5f              pop     rdi
00000000`6ef18e06 c3              ret
```

ntdll!RtlInterlockedPushEntrySlist

Command

```
0:018> d npctrl + 16c1c8
00000000`6f04c1c8  50 b3 5e 1b 7a 01 00 00-00 00 00 00 00 00 00 00  P.^.z...........
00000000`6f04c1d8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c1e8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c1f8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c208  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c218  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c228  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`6f04c238  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0:018> d 17a1b5eb350
0000017a`1b5eb350  01 00 01 00 00 00 00 00-00 00 90 44 7a 01 00 00  ..........Dz....
0000017a`1b5eb360  69 00 66 00 72 00 00 00-00 00 00 00 00 00 08 00  i.f.r...........
0000017a`1b5eb370  e4 0d 00 00 6c 0e 00 00-00 00 00 00 00 00 08 00  ....l...........
0000017a`1b5eb380  00 00 00 00 14 00 00 00-28 00 00 00 3c 00 00 00  ........(...<...
0000017a`1b5eb390  30 00 70 00 78 00 00 00-00 00 00 00 00 00 08 00  0.p.x...........
0000017a`1b5eb3a0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000017a`1b5eb3b0  69 00 66 00 72 00 61 00-6d 00 65 00 00 00 02 00  i.f.r.a.m.e.....
0000017a`1b5eb3c0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

PSLIST_HEADER ListHead

The Fix to CFG Bypass By Loading Silverlight

- Arbitrary Code Guard (ACG) prevents this CFG/DEP bypass in a generic way, however this mitigation doesn’t specifically resolve the issue of bypassing ActiveX restriction in Edge.
  - npctrl.dll can’t be loaded when ACG is on, because its PE file’s import table (.idata section) is somehow marked as RX at the compilation time.
  - Even when npctrl.dll is loaded, it’s unable to create RWX memory.

- In Windows 10 build 14986 (WIP slow ring), we are still able to make the exploit work via thread opt out (THREAD_DYNAMIC_CODE_ALLOW = 1).

Suggestions for Preventing Data-only Attack

- Never leave critical data unprotected!
  - If possible, apply write protection on the critical data page. Remove the write protection only when it needs to be updated, and be sure to lock the data page immediately after the updating finishes.
  - If for some reason the scheme above is not feasible (such as granularity, performance etc), the critical data can probably be stored in a dynamically allocated memory (ASLR enabled), and its address needs to be encrypted and stored separately.
  - For the encryption scheme mentioned above, the secret key of encryption has to be in a protected area, such as in kernel space, to prevent the access from user-mode. Moreover, the strength of encoding or encryption algorithm needs to enhanced to prevent brute-force attack.

- Always try verifying the integrity of critical data before using it. With such extra logic being introduced, many data-only attacks can be detected in their early stage.

Conclusion

- With the emergence of fine-grained CFI solution, the approach of calling function out-of-context will gradually lose its effectiveness.

- Today, application programs and operating systems have a lot of unprotected data, which can be leveraged to conduct powerful attack without the need of altering the program's execution flow.

- Even if people have already been aware of the danger of data-only attacks, it's still very difficult to prevent.
  - In some cases, it's almost impossible to distinguish an attack from a legitimate access. Therefore, data-only attacks can't always be resolved from the program's logical perspective.
  - Due to performance consideration, OS/Application can't move all its critical data into kernel space. In most cases, such user-space data will be protected by either "read-only" memory attribute (such as PE module's import/export table section, .mrdata section of ntdll.dll) or simple encoding (RtlEncodePointer).
  - The battle of contending for the protected memory will continue.

- Data-only attack may have some variations, and it can be combined with some other exploitation techniques, such as race condition. We have discovered a couple of such bugs, and we would like to share the details after they are fixed by the vendor.

Q & A

- This concludes the part I of our data-only attack talk, in part II we'll be presenting more examples of data-only attack in detail. Stay tuned!
- You are welcomed to send questions to

  - Bing Sun @ bing.sun@intel.com
  - Chong Xu @ chong.c.xu@intel.com
  - Stanley Zhu @ stanley.zhu@intel.com

- Thank MSRC for getting the issues fixed in a timely manner.
- Special thanks to Haifei Li and the McAfee IPS Vulnerability Research team.

References

- From read-write anywhere to controllable calls
- Mitigation bounty — 4 techniques to bypass mitigations
- Bypassing Control Flow Guard in Windows 10
- Use Chakra engine again to bypass CFG
- Chakra JIT CFG Bypass
- Bypass Control Flow Guard Comprehensively
- JIT Spraying Never Dies - Bypass CFG By Leveraging WARP Shader JIT Spraying
- Look Mom, I don’t use Shellcode
- Write Once, Pwn Anywhere