Reversing Windows8:
Interesting Features of Kernel Security

MJ0011
th_decoder@126.com

Agenda

Goal:

Revising Windows 8 Release Preview

Find new security features to defend or mitigate kernel vulnerability attack

Target:

ntoskrnl

Tools: IDA Pro/Hex-rays/windbg

Agenda

- Disable Null Page Memory Allocation
- Disable Win32k System Call
- Security Failure Interrupt
- Nonexecutable NonPaged Pool
- Apply Intel® Secure Key Technology
- Apply Intel® SMEP Technology

Disallow Null Page Allocation

- Null-page memory：for 16bit VM:ntvdm
- Allocate null-page memory by using ZwAllocateVirtualmemory to Trigger uninitialized object pointer reference vulnerability or to achieve other vulnerability attack
  - Example： CVE-2010-4398
    N-Protect TKRgAc2k.sys kernel 0day(POC2010)
- Now the system disallow low address (0x0~0x10000) allocation in Windows8
- EPROCESS->Flags.VdmAllowed

Disallow Null Page Allocation

- 16bit virtual machine is disabled by default in windows8, only administrators can enable it

Disallow Null Page Allocation

- Windows8 checks all the locations to which null page can be allocated.
  - MiCreatePebOrTeb : create peb or teb
  - MiMapViewOfImageSection->MiIsVaRangeAvailable: Mapping image section
  - MiMapViewOfDataSection/MiMapViewOfPhysicalSection Mapping data/physical section
  - MmMapLockedPagesSpecifyCache/MmMapLockedPages-> MiMapLockedPagesInUserSpace
  - Mapping in user address space
  - NtAllocateVirtualMemory:Allocate process memory

Disallow win32k system call

Disallow win32k system call

- EPROCESS->Flags2.DisallowWin32kSystemCalls
- KiFastCallEntry(2)->PsConvertToGuiThread

```
; ---------------------------------------------------------------------------

loc_7BC7F6:                             ; CODE XREF: PsConvertToGuiThread()+23↑j
                mov     eax, [ebp+CurrentThread]
                cmp     [eax+KTHREAD.ServiceTable], offset _KeServiceDescriptorTable
                jz      short @ThreadServiceTableIsSSDT

                mov     eax, STATUS_ALREADY_WIN32
                jmp     locret_7BC8E3

; ---------------------------------------------------------------------------

@ThreadServiceTableIsSSDT:              ; CODE XREF: PsConvertToGuiThread()+39↑j
                mov     eax, [ebp+CurrentThread]
                mov     eax, [eax+KTHREAD.Process]
                mov     [ebp+CurrentProcess], eax
                mov     eax, [ebp+CurrentProcess]
                mov     [ebp+CurrentProcess_], eax
                mov     eax, [ebp+CurrentProcess_]
                mov     eax, [eax+EPROCESS.Flags2]
                and     eax, 80000000h  ; DisallowWin32kSystemCalls : Pos 31, 1 Bit
                jz      short @AllowConvertToGuiThread

                mov     eax, STATUS_ACCESS_DENIED
                jmp     locret_7BC8E3

; ---------------------------------------------------------------------------

@AllowConvertToGuiThread:               ; CODE XREF: PsConvertToGuiThread()+65↑j
```

Disallow win32k system call

- Why disallow win32k system call
- Win32k.sys: a high incidence of windows kernel vulnerability, can be called without process privilege control
  - MS11-087 Trojan.win32.Duqu : win32k.sys font parse vulnerability
- Current application sandbox defense method
  - Job UI restriction (ineffective)
- Disallowing win32k system call can easily defend any win32k related 0day without using 3rd party kernel driver
- Also can defense user/gdi sandbox attack trick which does not use 0day

Disallow win32k system call

- PsConvertToGuiThread : Used by GUI thread to make its initial win32k system call
- After applying DisallowWin32kSystemCalls flag, any system call for user/gdi will fail.

- 3 methods to get this flag :
  - 1.IEFO Registry Configuration :
    - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\MitigationOptions (0x10000000)
    - NtCreateUserProcess->PspAllocateProcess-> PspApplyMitigationOptions
  - 2.Documented API:SetProcessMitigationPolicy
    - NtSetInformationProcess->ProcessMitigationPolicy
  - 3.Inherit from parent process

Security Failure Interruption

- New security failure interruption in Windows8: INT 0x29
- Will trigger BSOD when used during security failure of windows kernel or other drivers.
- Most commonly used in double-linked list operation. Such interruption is added to all the double-linked list in Windows OS Loader / kernel and kernel drivers
- So called "Safe Linking & Safe Unlinking"
  - Safe Linking::IoRegisterFsRegistrationChangeMountAware
  - Safe Unlinking:IoUnregisterFileSystem
- To defense attack trick such as using tampered list entry structure to manipulate a Write-What-Where condition

Security Failure Interrupt

Safe unlinking and int 0x29 interrupt: IoUnregisterFileSystem

```
PAGE:00786288                   mov     edi, [ebp+DeviceObject]
PAGE:0078628B                   lea     eax, [edi+DEVICE_OBJECT.Queue.ListEntry.Flink]
PAGE:0078628E                   xor     ebx, ebx
PAGE:00786290                   cmp     [eax+LIST_ENTRY.Flink], ebx
PAGE:00786292                   jz      short loc_7862A7
PAGE:00786292
PAGE:00786294                   mov     edx, [eax+LIST_ENTRY.Flink]
PAGE:00786296                   mov     ecx, [eax+LIST_ENTRY.Blink]
PAGE:00786299                   cmp     [edx+LIST_ENTRY.Blink], eax
PAGE:0078629C                   jnz     short @SecurityFailure
PAGE:0078629C
PAGE:0078629E                   cmp     [ecx+LIST_ENTRY.Flink], eax
PAGE:007862A0                   jnz     short @SecurityFailure
PAGE:007862A0
PAGE:007862A2                   mov     [ecx], edx
PAGE:007862A4                   mov     [edx+4], ecx
PAGE:007862A4
PAGE:007862A7
PAGE:007862A7 loc_7862A7:                               ; CODE XREF: IoUnregisterFileSystem(x)+2B↑j
PAGE:007862A7                   mov     esi, _IopFsNotifyChangeQueueHead
PAGE:007862AD                   jmp     short loc_7862BD
PAGE:007862AD
PAGE:007862AF ;---------------------------------------------------------------------------
PAGE:007862AF
PAGE:007862AF @SecurityFailure:                         ; CODE XREF: IoUnregisterFileSystem(x)+35↑j
PAGE:007862AF                                           ; IoUnregisterFileSystem(x)+39↑j
PAGE:007862AF                   push    3
PAGE:007862B1                   pop     ecx
PAGE:007862B2                   int     29h             ; KiRaiseSecurityFailure
PAGE:007862B2
```

Security Failure Interruption

- KiRaiseSecurityCheckFailure :
  - Int 0x29 Interrupts handler routine
  - It simply calls KiFastFailDispatch->KiBugCheck to show BSOD
- Bug check code: 0x139 : Currently not documented
  - Parameter:ecx :The Error ID
- Known Security Fast-Fail Error ID:
  - 0x2: Kernel driver security cookie exception
  - 0x3: Safe unlinking / Safe linking exception
  - 0x6: Kernel driver security cookie initialize exception
  - 0x9: RtlQueryRegistryValuesEx using untrust key(CVE-2010-4398)

Nonexecutable Nonpaged Pool

- Before Windows8 , kernel and kernel drivers can only use ExAllocatePoolXXX API to allocate executable nonpaged memory
- Executable nonpaged pool can be used to create kernel vulnerability ROP attack
- In Windows8 , There are some new pool types:
  - NonPagedPoolNx
  - NonPagedPoolNxCacheAligned
  - NonPagedPoolSessionNx
- Kernel pool memory which is allocated from NonPagedPoolNx type is nonexecutable now, code executable in this type of pool will cause a system crash
- Windows8 kernel and kernel drivers now use NonPagedPoolNx instead of NonPagedPool type

Nonexecutable Nonpaged Pool

- Kernel uses nonexecutable nonpagedpool
- IoAllocateDriverObjectExtension

```
__stdcall IoAllocateDriverObjectExtension(x, x, x, x) proc near

var_1           = byte ptr -1
DriverObject    = dword ptr  8
ClientIdentificationAddress= dword ptr  0Ch
DriverObjectExtensionSize= dword ptr  10h
DriverObjectExtension= dword ptr  14h

; FUNCTION CHUNK AT .text:004D4D37 SIZE 00000006 BYTES
; FUNCTION CHUNK AT .text:00566040 SIZE 00000015 BYTES
; FUNCTION CHUNK AT .text:0056605A SIZE 00000012 BYTES

                mov     edi, edi
                push    ebp
                mov     ebp, esp
                push    ecx
                mov     eax, [ebp+DriverObjectExtensionSize]
                push    edi
                mov     edi, [ebp+DriverObjectExtension]
                and     dword ptr [edi], 0
                mov     [ebp+var_1], 0
                cmp     eax, 0FFFFFFF7h
                ja      loc_4D4D29

                push    ebx
                push    esi
                push    'virD'          ; Tag
                lea     ebx, [eax+8]
                push    ebx             ; NumberOfBytes
                push    NonPagedPoolNx  ; PoolType
                call    ExAllocatePoolWithTag(x,x,x)
```

Apply Intel® Secure Key Technology

- Intel® Secure Key Technology , code name: Bull Mountain Technology
- Introduced in April 2012, Intel 3rd generation Core processor: Ivy Bridge
  - Offers hardware approach to high-quality,high-performance entropy and random number generator
- New Intel 64 Architecture instruction: RDRAND
- Windows8 kernel uses this instruction to generate random number to produce security cookie and ASLR address
- Related Function：ExGenRandom

Apply Intel® Secure Key Technology

- Past kernel random number attacks: security cookie prediction & ASLR brute force
- Before Windows8 , Windows kernel use system clock to generate security cookie and ASLR address
- Base on module loading time, security cookie can be easily predicted with a success rate of more than 46 percent(j00ru).
- J00ru. Windows Kernel-mode GS Cookies subverted.
- H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the effectiveness of address-space randomization.
- Windows 8 kernel use security cookie generated by Intel secure key technology and apply it to all loaded kernel drivers

Apply Intel® Secure Key Technology

- When loading the kernel driver, Windows 8 calls MiProcessLoadConfigForDriver to generate security cookie, locates old security cookie in PE and replaces it.

```
PAGE:006F4053 ; __stdcall MiProcessLoadConfigForDriver(x)
PAGE:006F4053 _MiProcessLoadConfigForDriver@4 proc near
PAGE:006F4053                                         ; CODE XREF: MmLoadSystemImage(x,x,x,x,x,x)+265↑p
PAGE:006F4053                                         ; MiReloadBootLoadedDrivers(x)+36D↓p
PAGE:006F4053                 xor     eax, eax
PAGE:006F4055                 call    _ExGenRandom@4  ; ExGenRandom(x)
PAGE:006F405A                 push    eax
PAGE:006F405B                 push    dword ptr [esi+20h]
PAGE:006F405E                 mov     eax, [esi+18h]
PAGE:006F4061                 call    _LdrInitSecurityCookie@16 ; LdrInitSecurityCookie(x,x,x,x)
PAGE:006F4066                 retn
PAGE:006F4066 _MiProcessLoadConfigForDriver@4 endp
PAGE:006F4066
PAGE:006F4066 ; ---------------------------------------------------------------------------
```

- New Windows8 kernel drivers will check if their security cookies are already replaced.

```
INIT:00319643 ___security_init_cookie proc near       ; CODE XREF: GsDriverEntry(x,x)+5↑p
INIT:00319643                 xor     eax, eax
INIT:00319645                 push    eax
INIT:00319646                 push    eax
INIT:00319647                 push    eax
INIT:00319648                 push    eax
INIT:00319649                 call    _ForceSEHExceptionHandler@16 ; ForceSEHExceptionHandler(x,x,x,x)
INIT:0031964E                 mov     eax, ___security_cookie
INIT:00319653                 test    eax, eax
INIT:00319655                 jz      short loc_319666
INIT:00319657                 cmp     eax, 0BB40E64Eh
INIT:0031965C                 jz      short loc_319666
INIT:0031965E                 not     eax
INIT:00319660                 mov     ___security_cookie_complement, eax
INIT:00319665                 retn
INIT:00319666 ; ---------------------------------------------------------------------------
INIT:00319666
INIT:00319666 loc_319666:                             ; CODE XREF: ___security_init_cookie+12↑j
INIT:00319666                                         ; ___security_init_cookie+19↑j
INIT:00319666                 push    6
INIT:00319668                 pop     ecx
INIT:00319669                 int     29h             ; Win8: RtlFailFast(ecx)
```

Apply Intel® Secure Key Technology

- The way of Windows7 kernel generates security cookie:
  HalQueryRealTimeClock(from CMOS) ^ rdtsc

- The way of Windows8 kernel generates security cookie:
  ExGenRandom-> ExpSecurityCookieRandomData ^ rdtsc

- Windows8 runtime kernel does not directly use RDRAND instruction

- ExGenRandom uses random entropy source generated from OS Loader calling RDRAND instruction in system booting process
  - Winload! OslpGatherRdrandEntropy

- In fact , OS Loader use 5 methods to get high quality random number entropy sources

- External entropy(from registry)\TPM entropy\clock entropy\ACPI entropy\RDRAND entropy

Apply Intel® Secure Key Technology

- IDA Pro 6.3 supports RDRAND instruction decoding.
- Winload initializing SecureKey in system booting process

```
.text:0040B8B5
.text:0040B8B5 loc_40B8B5:                             ; CODE XREF: OslpGatherRdrandEntropy(x,x)+66↓j
.text:0040B8B5                                         ; OslpGatherRdrandEntropy(x,x)+71↓j
.text:0040B8B5                 rdrand  edx
.text:0040B8B8                 jnb     short loc_40B8B5
.text:0040B8BA                 mov     [ecx+eax*4], edx
.text:0040B8BD                 inc     eax
.text:0040B8BE                 cmp     eax, 1800h
.text:0040B8C3                 jb      short loc_40B8B5
.text:0040B8C5                 lea     esi, [edi+24h]
.text:0040B8C8                 push    6000h
.text:0040B8CD                 mov     edx, ecx
.text:0040B8CF                 call    @SymCryptSha512@12 ; SymCryptSha512(x,x,x)
.text:0040B8D4                 mov     esi, [ebp+arg_4]
.text:0040B8D7                 mov     edx, 6000h
.text:0040B8DC                 mov     ecx, esi
.text:0040B8DE                 call    @SymCryptWipeAsm@8 ; SymCryptWipeAsm(x,x)
.text:0040B8E3                 push    esi
.text:0040B8E4                 call    _BlMmFreeHeap@4 ; BlMmFreeHeap(x)
.text:0040B8E9                 xor     esi, esi
.text:0040B8EB
.text:0040B8EB loc_40B8EB:                             ; CODE XREF: OslpGatherRdrandEntropy(x,x)+21↑j
.text:0040B8EB                                         ; OslpGatherRdrandEntropy(x,x)+3E↑j
.text:0040B8EB                 mov     [edi+10h], ebx
.text:0040B8EE                 mov     [edi+14h], esi
.text:0040B8F1                 pop     edi
.text:0040B8F2                 pop     esi
.text:0040B8F3                 pop     ebx
.text:0040B8F4                 pop     ebp
.text:0040B8F5                 retn    8
.text:0040B8F5 _OslpGatherRdrandEntropy@8 endp
.text:0040B8F5
```

Apply Intel® Secure Key Technology

- ExGenRandom is also used in these kernel functions :
  - Kernel pool quota cookie
  - Kernel pool address allocation randomization
  - PEB/TEB address randomization
  - Kernel module address randomization
  - Thread stack and heap address randomization

- And user functions:
  - Shared User Data->Cookie(ring3 Ldr* encode and decode)
  - User address space memory allocation randomization
  - User data section and image section allocation randomization

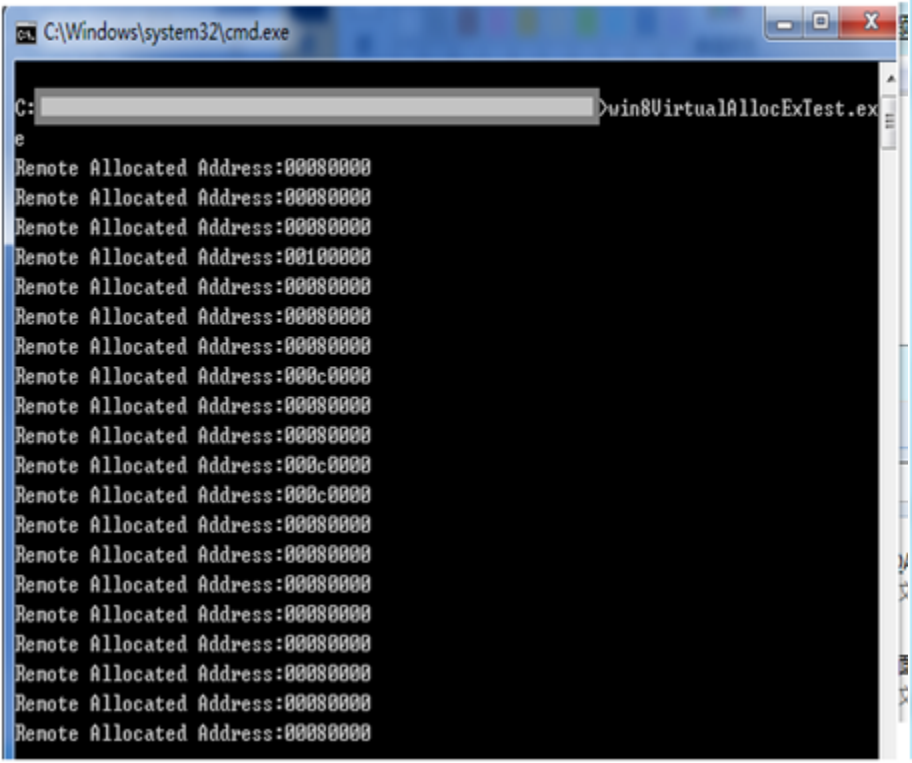
Apply Intel® Secure Key Technology

- Guillaume. Bypassing ASLR and DEP on Adobe Reader X
- The sandbox inside Adobe Reader X and Google Chrome browser uses VirtualAllocEx function to allocate memory and copy System Call Stub jump shell code into it.
- In Win7 and previous OS, memory allocated by VirtualAllocEx function is not randomized. There is more than 85 percent chance the shell code base address will hit a fixed address in every booting.
- The attacker uses System Call Stub jump code in fixed address to allocate executable memory and bypass DEP+ASLR
- Windoows8：System uses MmInitializeProcessAddressSpace to call ExGenRandom and generate random number during process startup
- When process uses NtAllocateVirtualMemory to allocate memory ,system uses MiSelectAddress to select a randomized address with generated random number

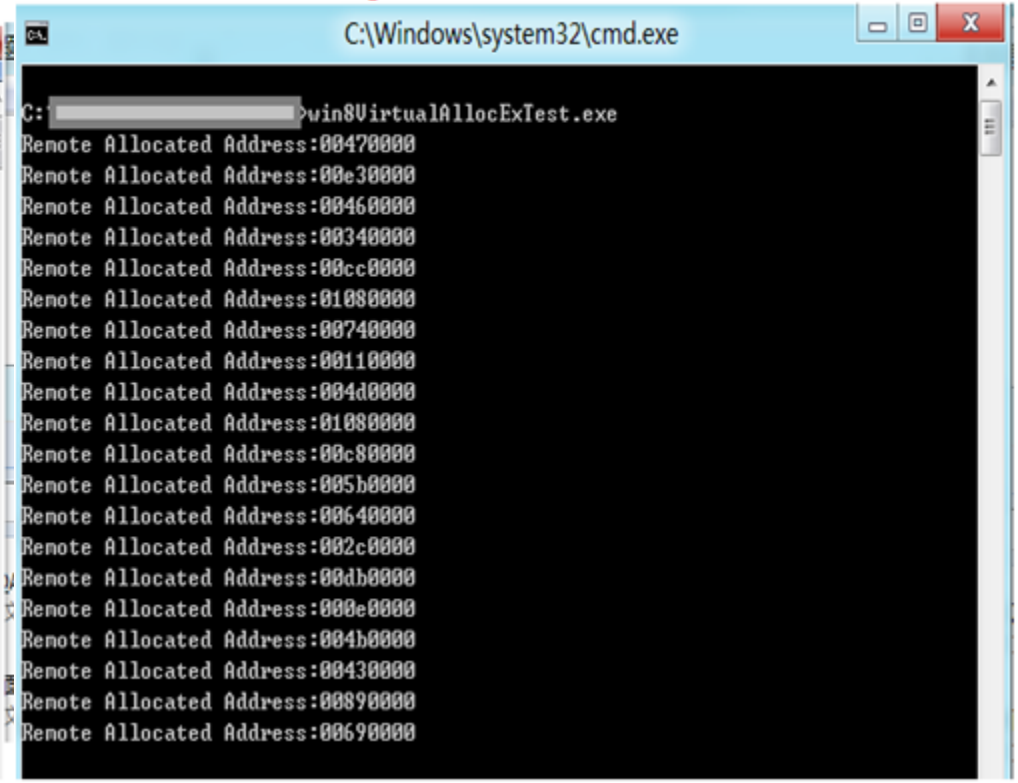
Apply Intel® Secure Key Technology

- A comparison test between Windows7 and Windows8 in remote user memory allocation address
- Start calc.exe process 20 times and allocate remote buffer in it

Windows7: almost hit at 0x80000



Windows8: very random



Apply Intel® SMEP Technology

- SMEP : Supervisor-Mode Execution Prevention
- Also introduced in April 2012 of Intel 3rd generation Core processor: Ivy Bridge

- New hardware protection mechanism provided by Intel CPU, allows pages to be protected from supervisor mode instruction fetches.

- Background：Most kernel vulnerability attacks use tricks to make kernel code jumping to preset shell code which is placed in user address space

- Classic trick：
- Replace HalDispatchTable-> HalQuerySystemInformation

- Why place shell code in ring3 address space? Payload and address randomization.

Apply Intel® SMEP Technology

- When SMEP is enabled:
  - Supervisor-mode(CPL<3)instruction will check the U/S flag of paging-structure entry during instruction fetching . The CPU will raise a exception when PTE owner is user.

- Set SMEP bit(bit 20) of cr4 register to 1 will enable SMEP

- Windows 8 kernel enables SMEP by default:
- Phase1Initialization-> Phase1InitializationDiscard ->KiInitMachineDependent

```
INIT:0092BB50
INIT:0092BB50                         @EnableSMEP:                            ; CODE XREF: KiInitMachineDependent()+23E↑j
INIT:0092BB50 0F 20 E0                                mov     eax, cr4
INIT:0092BB53 0D 00 00 10 00                          or      eax, 100000h
INIT:0092BB58 0F 22 E0                                mov     cr4, eax
INIT:0092BB5B E9 FB 4C FD FF                          jmp     loc_90085B
INIT:0092BB5B
INIT:0092BB5B                         ; END OF FUNCTION CHUNK FOR _KiInitMachineDependent@0
INIT:0092BB60
```

Apply Intel® SMEP Technology

- MI_CHECK_KERNEL_NOEXECUTE_FAULT

- Windows8 uses this function to process two kinds of nonexecutable exceptions in Page Fault Trap handler: KiTrap0E

```
1  int __userpurge MI_CHECK_KERNEL_NOEXECUTE_FAULT<eax>(int FaultStatus<eax>, ULONG_PTR ReasonId<edx>, ULONG_PTR VirtualA
2  {
3    if ( FaultStatus & MmPaeErrMask )
4    {
5      if ( KeFeatureBits & KF_SMEP && MmPte & 4 ) // SMEP Enabled and PTE Owner = User
6      {
7        ReasonId |= REASON_ID_SMEP;
8      }
9      else
10     {
11       FaultStatus = HIDWORD(MmPte) & HIDWORD(MmPaeMask);
12       if ( !(MmPte & MmPaeMask) )
13         return FaultStatus;
14     }
15     KeBugCheckEx(ATTEMPTED_EXECUTE_OF_NOEXECUTE_MEMORY, VirtualAddress, MmPte, TrapInformation, ReasonId);// BSOD
16   }
17   return FaultStatus;
18 }
```

Apply Intel® SMEP Technology

- An way to bypass SMEP: put shell code into kernel object memory, and get kernel object address with NtQuerySystemInformation->SystemHandleInformation(Ex)

```
typedef struct _SYSTEM_HANDLE_TABLE_ENTRY_INFO {
    USHORT UniqueProcessId;
    USHORT CreatorBackTraceIndex;
    UCHAR ObjectTypeIndex;
    UCHAR HandleAttributes;
    USHORT HandleValue;
    PVOID Object;
    ULONG GrantedAccess;
} SYSTEM_HANDLE_TABLE_ENTRY_INFO, *PSYSTEM_HANDLE_TABLE_ENTRY_INFO;
```

- Available target object：FileObject ?

```
typedef struct _FILE_OBJECT {
    CSHORT Type;
    CSHORT Size;
    PDEVICE_OBJECT DeviceObject;
    PVPB Vpb;
    PVOID FsContext;
    PVOID FsContext2;
    PSECTION_OBJECT_POINTERS SectionObjectPointer
    PVOID PrivateCacheMap;
    NTSTATUS FinalStatus;
    struct _FILE_OBJECT *RelatedFileObject;
    BOOLEAN LockOperation;
    BOOLEAN DeletePending;
    BOOLEAN ReadAccess;
    BOOLEAN WriteAccess;
    BOOLEAN DeleteAccess;
    BOOLEAN SharedRead;
    BOOLEAN SharedWrite;
    BOOLEAN SharedDelete;
    ULONG Flags;
    UNICODE_STRING FileName;
    LARGE_INTEGER CurrentByteOffset;
    ULONG Waiters;
    ULONG Busy;
    PVOID LastLock;
    KEVENT Lock;
    KEVENT Event;
    PIO_COMPLETION_CONTEXT CompletionContext;
} ?end _FILE_OBJECT ? FILE_OBJECT;
typedef struct _FILE_OBJECT *PFILE_OBJECT; // ntndis
```

Apply Intel® SMEP Technology

- Impossible in Windows8: SMEP + NonPagedPoolNx
- All kernel objects memory are nonexecutable

- The pool type of kernel object is assigned by ObCreateObjectType call in system booting process

- Windows8 has assigned pool type of FileObject as NonPagedPoolNx

IDA View-A | Pseudocode-D | Pseudocode-C | Pseudocode-B | Stack

```
1  char __cdecl IoCreateObjectTypes()
2  {
3    char result; // al@8
4    _OBJECT_TYPE_INITIALIZER v1; // [sp+10h] [bp-60h]@1
5    LSA_UNICODE_STRING DestinationString; // [sp+68h] [bp-8h]@1
6
7    memset(&v1, 0, 0x58u);
8    v1.Length = 88;
9    v1.InvalidAttributes = 256;
10   v1.GenericMapping.GenericRead = IopFileMapping[0];
11   v1.GenericMapping.GenericWrite = IopFileMapping[1];
12   v1.GenericMapping.GenericExecute = IopFileMapping[2];
13   v1.GenericMapping.GenericAll = IopFileMapping[3];
14   v1.ObjectTypeFlags |= 4u;
15   v1.PoolType = NonPagedPoolNx;
16   v1.ValidAccessMask = 2032127;
```

Apply Intel® SMEP Technology

– The defense situation of known SMEP attack trick in Windows8

Attack Trick	Windows 8 Defense Method
SystemHandleInformation(Ex)	Kernel object memory NX
SystemLockInformation	Safe Linking/Unlinking
SystemModuleInformation	No protection in data area Write protection in code area
SystemExtendProcessInformation	No protection
GDT/IDT	No protection
0xFFDF0000 (User Shared Data)	MiProtectKernelRegions set Nx
0xFFC00000~0xFFFFFFFF(KPCR)	KPCR randomization
Win32k Shared Section	USER/Kernel object memory Nx

Reference

- Intel. Intel® Digital Random Number Generator Software Implementation Guide
- Intel. Intel® 64 and IA-32 Architectures Developer's Manual: Vol. 3A
- J00ru . Exploiting the otherwise non-exploitable:Windows Kernel-mode GS Cookies subverted
- H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh.On the Effectiveness of Address-Space Randomization
- Guillaume. Bypassing ASLR and DEP on Adobe Reader X

Q&A

- Thanks for:
  - CHROOT Security Group
  - 360Safe MDT/HIPS Team