Reversing Windows8: Interesting Features of Kernel Security

MJ0011
th_decoder@126.com
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
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 Windows 8

EPROCESS->Flags.VdmAllowed
16-bit virtual machine is disabled by default in Windows 8, only administrators can enable it.
Disallow Null Page Allocation

- Windows 8 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

- EPROCESS->Flags2.DisallowWin32kSystemCalls

- KiFastCallEntry(2)->PsConvertToGuiThread

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

    mov    eax, STATUS_ALREADY_WIN32
    jmp    locret_7BC8E3

@ThreadServiceTableIsSSDT:  ; CODE XREF: PsConvertToGuiThread()+39↑
    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 @DisallowConvertToGuiThread

    mov    eax, STATUS_ACCESS_DENIED
    jmp    locret_7BC8E3

@DisallowConvertToGuiThread:  ; CODE XREF: PsConvertToGuiThread()+65↑
```
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
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. IEOF 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
Safe unlinking and int 0x29 interrupt: IoUnregisterFileSystem

```
mov    edi, [ebp+DeviceObject]
lea    eax, [edi+DEVICE_OBJECT.Queue.ListEntry.Flink]
xor    ebx, ebx
cmp    [eax+LIST_ENTRY.Flink], ebx
jz     short loc_7862A7
mov    edx, [eax+LIST_ENTRY.Flink]
mov    ecx, [eax+LIST_ENTRY.Blink]
cmp    [edx+LIST_ENTRY.Blink], eax
jnz    short @SecurityFailure
cmp    [ecx+LIST_ENTRY.Flink], eax
jnz    short @SecurityFailure
mov    [ecx], edx
mov    [edx+4], ecx
```

```
loc_7862A7: ; CODE XREF: IoUnregisterFileSystem(x)+28j
mov    esi, _IopFsNotifyChangeQueueHead
jmp    short loc_7862BD
```

```
@SecurityFailure: ; CODE XREF: IoUnregisterFileSystem(x)+35j
push   3
pop    ecx
int    29h ; KiRaiseSecurityFailure
```
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

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

var_1 = byte ptr -1
DriverObject = dword ptr 8
ClientIdIdentificationAddress = 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, 0FFFFFFFh
    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

- Based 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.


- 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.

```
; stdCall MiProcessLoadConfigForDriver(x)
MiProcessLoadConfigForDriver@4 proc near
    xor eax, eax
    call _ExGenRandom@4 ; ExGenRandom(x)
    push eax
    push dword ptr [esi+20h]
    mov eax, [esi+18h]
    call _LdrInitSecurityCookie@16 ; LdrInitSecurityCookie(x,x,x,x)
    ret
```

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

```
security_init_cookie proc near
    xor eax, eax
    push eax
    push eax
    push eax
    push eax
    push eax
    call _ForceSEHExceptionHandler@16 ; ForceSEHExceptionHandler(x,x,x,x)
    mov eax, _security_cookie
    test eax, eax
    jz short loc_319666
    cmp eax, 0BB40E64Eh
    jz short loc_319666
    not eax
    mov __security_cookie_complement, eax
    ret
security_init_cookie endp
```

```
loc_319666:
```

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

```
.loc_40b8b5:
  rdrand    edx
  jnb short loc_40b8b5
  mov [ecx+eax*4], edx
  inc eax
  cmp eax, 1000h
  jb short loc_40b8b5
  lea esi, [edi+24h]
  push 6000h
  mov edx, ecx
  call @Syncryptsha512@12 ; Syncryptsha512(x,x,x)
  mov esi, [ebp+arg_4]
  mov edx, 6000h
  mov ecx, esi
  mov ecx, esi
  call @Syncryptwipem@8 ; Syncryptwipem(x,x)
  push esi
  call _B1MmFreeHeap@4 ; B1MmFreeHeap(x)
  xor esi, esi

.loc_40b8eb:
  mov [edi+10h], edx
  mov [edi+14h], esi
  pop edi
  pop esi
  pop ebx
  pop ebp
  ret 8
`
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
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

Windows8: 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

```assembly
; EnableSMEP:       ; CODE XREF:   KiInitMachineDependent()\+23E1j
    mov    eax, cr4
    or     eax, 1000000h
    mov    cr4, eax
    jmp    loc_908858
```

```assembly
; END OF FUNCTION CHUNK FOR _KiInitMachineDependent@Q
```
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

```c
int __userpurge MI_CHECK_KERNEL_NOEXECUTE_FAULT<eax>(int FaultStatus<eax>, ULONG_PTR ReasonId<edx>, ULONG_PTR VirtualAd)
{
    if ( FaultStatus & MmPaeErrMask )
    {
        if ( KeFeatureBits & KF_SMEP && MmPte & 4 ) // SMEP Enabled and PTE Owner - User
            ReasonId |= REASON_ID_SMEP;
        else
            FaultStatus = HIWORD(MmPte) & HIWORD(MmPaeMask);
        if ( !(MmPte & MmPaeMask) )
            return FaultStatus;
    }
    KeBugCheckEx(ATTENDED_EXECUTE_OF_NOEXECUTE_MEMORY, VirtualAddress, MmPte, TrapInformation, ReasonId);// BSOD
    return FaultStatus;
}
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)

```c
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?

```c
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 Look;
    KEVENT Event;
    PIO_COMPLETION_CONTEXT CompletionContext;
} ?end_FILE_OBJECT ? FILE_OBJECT;

typedef struct _FILE_OBJECT *PFIELD_OBJECT; // ntdis
```
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
Apply Intel® SMEP Technology

- The defense situation of known SMEP attack trick in Windows8

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