I Got 99 Problem But a Kernel Pointer Ain’t One

There’s an info leak party at Ring 0

Alex Ionescu, Chief Architect
Recon 2013

@aionescu
alex@crowdstrike.com

Bio

- Reverse engineered Windows kernel since 1999
  - Lead kernel developer for ReactOS Project
- Co-author of Windows Internals 5th and 6th Edition
- Founded Winsider Seminars & Solutions Inc., to provide services and Windows Internals training for enterprise/government
- Interned at Apple for a few years (Core Platform Team)
- Now Chief Architect at CrowdStrike

Introduction

Outline

- Introduction
- Motivation and Previous Work
- Old School API Leaks
- System Design Leaks
- Tracing/Debugging API Leaks
- System Memory Leaks
- SuperFetch Leaks
- Conclusion

Motivation

- Making Spender (grsecurity) troll really hard
  - “Kernel ASLR has never been broken by anyone I know”
  - Got a really well thought out article in response

Motivation (seriously)

- Windows has been making a decent job of improving their ASLR in Windows 8
  - And newer protections are yet to come
- Guessing of user-mode addresses now requires bypassing:
  - High Entropy ASLR
  - Top-down and Bottom-up Anonymous Memory Randomization
  - Heap Allocation Order Randomization
  - …etc…
- But Kernel ASLR remains a big problem
  - As part of a local exploit, too much information is present/given away on the system to the attacker
- Disparate papers/presentations exist on this issue

Previous Work

- Too many to list them all
- Matthew Jurczyk, Tavis Ormandy, Tarjei Mandt & the other usual suspects

Old School API Leaks

You Want… Module Base Addresses?

- NtQuerySystemInformation
  - Class: SystemModuleInformation
  - *NEW* Class: SystemModuleInformationEx
- Return type is RTL_PROCESS_MODULES with
  - RTL_PROCESS_MODULE_INFORMATION
  - RTL_PROCESS_MODULE_INFORMATION_EX
- EX Adds Checksum, TimeStamp, and Original Base
- Before Windows 8, could also be used to query user-mode libraries

You Want… All Kernel Object Addresses?

- NtQuerySystemInformation
  - Class: SystemObjectInformation
- Return type is SYSTEM_OBJECT_INFORMATION
- Contains
  - PVOID of the Kernel Object Address
  - PEPROCESS of the Kernel Object Creator
- Requires the object type/system to enable “Maintain Type List”

You Want… Named Kernel Object Addresses?

- NtQuerySystemInformation
  - Class: SystemHandleInformation
  - *NEW* Class: SystemHandleInformationEx

- Return type
  - SYSTEM_HANDLE_INFORMATION(_EX)

- Contains
  - PVOID of the Kernel Object Address
  - HANDLE value in the process

- Only returns 16-bit handles and PIDs – must use Ex version

You Want… Kernel Lock Addresses?

- NtQuerySystemInformation
  - Class: SystemLockInformation

- Return type
  - RTL_PROCESS_LOCKS with
  - RTL_PROCESS_LOCK_INFORMATION

- Contains
  - PVOID of the Kernel Resource
  - PVOID of Kernel Thread Owner

You Want… Kernel Stack Addresses?

- NtQuerySystemInformation
  - Class: SystemExtendedProcessInformation

- Return type
  - SYSTEM_EXTENDED_THREAD_INFORMATION

- Contains
  - PVOID of the Kernel Stack Base and Kernel Stack Limit
  - PVOID of the TEB

You Want… Kernel Pool Addresses?

- NtQuerySystemInformation
  - Class: SystemBigPoolInformation
- Return type
  - SYSTEM_BIGPOOL_INFORMATION with
  - SYSTEM_BIGPOOL_ENTRY
- Contains
  - PVOID of the Kernel Pool Address (if > 4KB) (“Big”)
    - And Tag

System Design Leaks

Selectors and Descriptors

- GDT and IDT are required pieces of any x86-based processor design
  - GDT highly deprecated in x64
- Address of the GDT and IDT is stored in GDTR and IDTR
  - CPU instruction exists to retrieve this (SGDT/SIDT)
  - It's not privileged!
- Additionally, entries in the GDT can be dumped on 32-bit Windows
  - 32-bit Windows has support for LDT, and implements API for querying it
  - But if no LDT is present, GDT is dumped instead
- Use NtQueryInformationThread (ThreadDescriptorTableEntry)
- Reveals three TSS addresses, and KPCR address
- Does not work on 64-bit because no LDT is supported

ARM Software Thread ID Registers

- Modern ARM processors implement TLS registers that can be used by operating system developers
  - Similar to fs/gs on x86/x64
- Three are currently defined in the Cortex-A9 architecture
  - TPIDRURW (User Read Write)
  - TPIDRURO (User Read Only)
  - TPIDRPRW (Privileged Read Write)
- Windows 8 on ARM (Windows RT) uses these registers, as seen in the public header files
  - RURW -> TEB
  - RPRW -> KPCR
  - RURO -> KTHREAD!

ACPI Table Data

- \Device\PhysicalMemory was accessible up until Windows Server 2003 SP1 in order to dump contents of RAM as desired
- Functionality was removed, but replaced with new API for
  - ACPI, SMBIOS, and 0xC0000->0xE0000 memory access

- NtQuerySystemInformation
  - Class: SystemFirmwareTableInformation

- Use SYSTEM_FIRMWARE_TABLE_INFORMATION

- Tables can store physical (RAM) addresses to devices and EFI

Trap Handler Leaks

- Worked with a lot of these while writing ReactOS…
- As an optimization, the kernel does not always build an SEH frame during certain operations
  - Such as a system call
- Instead, the page fault handler recognizes if the exception came from one such optimized location
  - And does correct exception handling back to user-mode
- However, this is based on reading the EIP!
  - Playing guessing games with the EIP can reveal kernel addresses based on the exception generated
- “j00ru” also discovered that some of these checks make crazy assumptions about other registers -> can cause crashes

Memory-Based Leaks

Win32k Shared Memory Regions

- Two “heaps” are implemented by the window management system
  - Session Heap (contains the object handle table)
  - Desktop Heap (contains the objects themselves)
- To get session heap: user32!gSharedInfo
  - aheList -> Session Heap Start (handle table)
  - ulSharedDelta → Difference between user and kernel
- To get desktop heap: TEB->Win32ClientInfo
  - pvDesktopBase → Desktop Heap Start
  - ulClientDelta → Difference between user and kernel

Win32k Objects

- Win32k Window Manager Handle Entries contain
  - PVOID of the Win32k Object (many/most are mapped in user-space)
  - PVOID of the NT Kernel Object owner (PETHREAD and/or PEPROCESS)
- Other structures are tagDESKTOPINFO, tagSHAREDINFO, tagCLIENTTHREADINFO, tagDISPLAYINFO, tagSERVERINFO
- These leak addresses of pointers inside kernel mode memory as well as things like mouse cursor position, last keys states…
- The objects themselves contain many pointers to NT objects/addreses

HAL Heap

- When the HAL initializes extremely early in the boot process, it does not have access to any memory management functionality

- The boot loader, HAL, and kernel’s memory manager all collaborate to define a region of memory reserved for the HAL

- 0xFFD00000->0xFFFFFFFF is for the HAL (even on x64)
  - !halpte shows current mappings on x86
  - hal!HalpHeapStart shows start of the heap

- Used to store ACPI tables, as well as all the HAL Objects on Windows 8

Tracing/Debugging API Leaks

Trace-Based ETW/WMI Leaks

- The kernel has extensive tracing performed through either legacy Windows Management Instrumentation (WMI) or Event Tracing for Windows (ETW)
- The relevant (documented) APIs are
  - StartTrace
  - ProcessTrace
- Many of these come from "MSNT_SystemTrace"
  - See http://msdn.microsoft.com/en-us/library/windows/desktop/aa364083(v=vs.85).aspx
- System Profiling Privilege is required

You Want… Kernel Process Pointers?

- ETW “Crimson” Provider
  - Or Legacy WMI
  - PERF_PROC

- Return type
  - WMI_PROCESS_INFORMATION

- Contains
  - PVOID of the Kernel Object Address (“UniqueProcessKey”)
  - PVOID of the Process Page Directory (“DirectoryTableBase”)

You Want… Kernel Thread Pointers?

- ETW “Crimson” Provider
  - Or Legacy WMI
  - PERF_THREAD
- Return type
  - WMI_EXTENDED_THREAD_INFORMATION
- Contains
  - PVOID of the Kernel Stack Base and Stack Limit
  - PVOID of the Kernel Start Address

You Want… Kernel Spinlock Addresses?

- ETW “Crimson” Provider
  - PERF_SPINLOCK
- Return type
  - WMI_SPINLOCK
- Contains
  - PVOID of the Kernel Spinlock Address
  - PVOID of the Kernel Caller Address
    - And if Address is DPC or ISR

You Want… Kernel Resource Addresses?

- ETW “Crimson” Provider
  - PERF_RESOURCE
- Return type
  - WMI_RESOURCE
- Contains
  - PVOID of the Kernel Resource Address

You Want… Kernel IRP and File Object Addresses?

- ETW “Crimson” Provider
  - PERF_FILENAME
  - EVENT_TRACE_FLAG_DISK_IO
- Return type
  - WMI_DISKIO_READWRITE
  - PERFINFO_FILE_INFORMATION/FILE_READ_WRITE
- Contains
  - PVOID of the IRP
  - PVOID of the FILE_OBJECT

You Want… Kernel Page Fault Addresses?

- ETW “Crimson” Provider
  - PERF_ALL_FAULTS
- Return type
  - WMI_PAGE_FAULT
- Contains
  - PVOID of the Fault Address
  - PVOID of the Program Counter

And there's more…

- DPC/ISR Tracing reveals the kernel pointer of every interrupt and DPC handler
- Image Load Tracing reveals kernel base address of every kernel module
- Pool Tracing reveals kernel address of every pool allocation
  - Even non-big ones
- *New* Windows 8 Object/handle-based Notifications
  - Leak the Kernel Object Pointer (and handle)

Triage Dumps

- NtSystemDebugControl was a goldmine API in Windows XP
  - Allowed complete Ring 0 control from Ring 3
- In Server 2003 SP1, almost all commands were disabled
  - A driver, kldbgdrv.sys is used by WinDBG instead
  - Calls KdSystemDebugControl, which checks if /DEBUG is active
- In Vista, a new command was added, and allowed even without being in /DEBUG mode
  - SYSDBG_COMMAND::SysDbgGetTriageDump
- Debug Privilege is required

What's in a Triage Dump?

- A typical crash dump header
  - KPCR, KPRCB, KUSER_SHARED_DATA, DPC Queues, Timer Table, etc…
- Information on the process that was selected for the dump
  - PEPROCESS structure and relevant fields
- Information on all the threads part of the process selected
  - PETHREAD structure and relevant fields
  - APC queue, pending IRPs, and wait queues
  - Kernel Stack Trace and Context
- And then Win32k "callback" gets called…
  - Dumps all tagTHREADINFO + tagPROCESSINFO
  - Dumps all global variables!

SuperFetch API Leaks

What’s SuperFetch?

- System component that tracks usage patterns and activities across one or multiple users on the machine
  - Application Launch
  - System Power Transitions
  - User Session Transitions
- Also tracks usage
  - All File I/O
  - All Page Faults
- Builds predictive database of application launches (Markov chain) and informs memory manager of priorities that each page should be given in memory and in the cache
  - Based on usage patterns over periods of up to 6 months

SuperFetch API

- SuperFetch lives in user-mode!
  - sysmain.dll service inside one of the hosts
- How does it track all page faults and File I/O
  - Partially through IOCTLs to FileInfo driver
  - Partially through undocumented API
- NtQuerySystemInformation
  - Class: SystemSuperfetchInformation
- Implements a variety of subclasses…

SuperFetch Information Subclasses

- SUPERFETCH_INFORMATION must be the buffer passed in
- SUPERFETCH_INFORMATION_CLASS determines the operation
  - Query all “sources”
  - Dump memory lists
  - Dump PFN database and page usages
  - ~12 total queries in Win7, ~20 in Win8
- Need version number (45 on Windows 7)
  - Need “magic password” (‘Chuk’)
- Need System Profile privilege

SuperFetch Information Leaks

- Querying for all sources will dump all PEPROCESS pointers
- Querying for the trace (if you don’t race with the actual SuperFetch service, or if you disable it) will dump file object pointers, virtual addresses, and program counters
- But the real deal is querying the PFN database!
  - PFN Database contains information on every physical page on the system and its usage
- A few years ago, I wrote a tool to dump this…
  - Now there’s RAMMap

Conclusion

Key Takeaways

- Unlike certain platforms such as iOS/OS X where kernel information disclosures seem to be taken rather seriously (even the GDT/IDT is aliased to prevent leaking the kernel base address!), Windows has a rather liberal policy toward kernel pointers
- Not quite as bad as Linux, however. Microsoft does care.
- Why don’t they “fix” these?
  - Most of the times, the answer is app compatibility
  - Other times, it’s developer support/requests
- However, requiring admin rights across the board for such system-level APIs may hit the right balance
- That’s not enough for DRM/Surface environments, however

Further Reading

- The NDK (Native Development Kit) is a header kit that I maintain which has the closest possible undocumented structure definitions
  - Even “j00ru” used old/incorrect/unknown structures in his papers ☹
  - NDK was built with information from PDBs, ASSERTs (before NT_ASSERT), private PDB (yep… the Windows 8 ones are still on the symbol server….) and .h leaks over the years, etc…
  - *NO* source code leak/etc material used.
- J00ru’s blog and most recent talks at CONFidence 2013 and Syscan 2013

QA

- Greetz/shouts to: j00ru, msuiche, lilhoser

CROWDSTRIKE