WHAT'S UNDER THE HOOD? ROOT CAUSE AND PATCH ANALYSES OF ELEVATION OF PRIVILEGE VULNERABILITIES IN THE WINDOWS OPERATING SYSTEM

Background

The Windows operating system is the most widely used in the world. However, its popularity and widespread use makes it a prime target for **malicious threat actors** seeking to **exploit vulnerabilities**. Studies have found that the Windows operating system is highly vulnerable to CWE-269: **improper privilege management**.

Aims of Project

1. To **replicate** each vulnerability on a virtual machine.
2. To understand the **root cause** of each vulnerability.
3. To identify the **patch** released by various vendors.
4. To determine whether each patch is **sufficient** in preventing future replications of the vulnerability.

Hypothesis

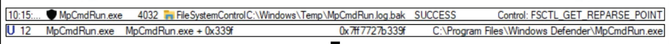
Current security patches of elevation of privilege vulnerabilities are typically designed to address a **specific method** of replication and exploitation. An attacker can use a **slightly altered method** of replication and exploitation based on a **similar concept**, which can **bypass the patch**.

CVE-2020-1170 (Windows Defender MpCmdRun.exe)

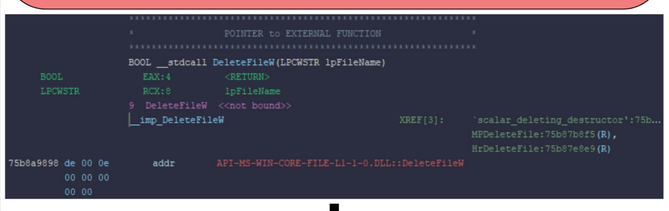
CVSS Score: 7.8

Root cause: MpCmdRun.exe **did not check** if MpCmdRun.log.bak had a **reparse point**.

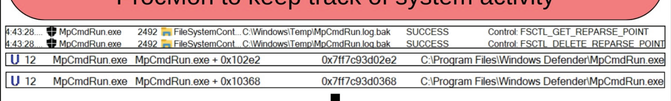
In a vulnerable virtual machine:
1. Create the folder C:\ZZ_SANDBOX\target
2. Create the junction C:\Windows\Temp\MpCmdRun.log.bak and set it as a mountpoint to C:\ZZ_SANDBOX
3. Fill MpCmdRun.log with 16MB of data
4. Open ProcMon to keep track of system activity
5. Perform a signature update



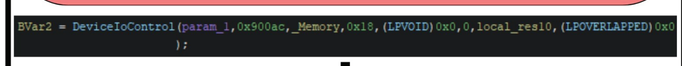
Perform static analysis in Ghidra, find the path to file deletion



Replicate vulnerability in patched virtual machine, open ProcMon to keep track of system activity

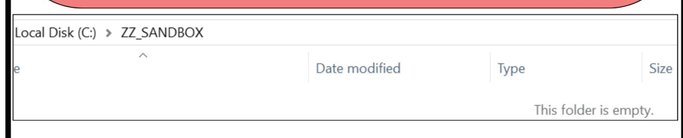


Find differences between vulnerable and patched MpCmdRun.exe (addition of FSCTL_DELETE_REPARSE_POINT)



In a patched virtual machine:
1. Create the folder C:\ZZ_SANDBOX\target
2. Create the junction C:\Windows\Temp\MpCmdRun.log.bak\junction and set it as a mountpoint to C:\ZZ_SANDBOX
3. Fill MpCmdRun.log with 16MB of data
4. Open ProcMon to keep track of system activity
5. Perform a signature update

On File Explorer, C:\ZZ_SANDBOX\target is deleted.
On ProcMon, reparse point is not deleted.
By using a different method in the patched VM, the vulnerability can still be replicated.



Patch analysis: As a similar method was used to replicate the vulnerability on the patched VM, the patch was **insufficient**.

CVE-2021-21551 (Dell dbutil_2_3.sys Driver)

CVSS Score: 8.8

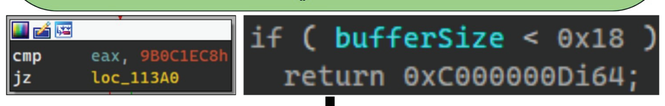
Root cause: **Write-what-where** condition - unprivileged users can use the driver to **write to kernel memory**.

Arbitrary Write Primitive

Perform static analysis in IDA: Find the file name for CreateFileA() to obtain a handle to the driver

```
memmove(SourceString, L"\\Device\\DBUtil_2_3", 0x26ui64);
```

Find the IOCTL code to access the function containing memmove(), and size of buffer. Use these as parameters for DeviceIoControl() to interact with the driver



1. Write C code to hit the breakpoint
2. Attach the VM to WinDbg and create a breakpoint
3. Run the code in the VM

```
0: kd> bp dbutil_2_3+5294
0: kd> bl
     0 e Disable Clear  fffff802`5e0a5294     0001 (0001) dbutil_2_3+0x5294
```

By stepping through functions, value of memmove() destination becomes 0. Edit code to use KUSER_SHARED_DATA + 0x800 as destination since it is a fixed kernel address that is writeable

```
1: kd> r rcx
rcx=0000000000000000
```

By displaying the bytes of destination and source, our buffer has been written to the kernel

```
2: kd> dqs rdx L1
ffffe08f`0a5ee298  43434343`43434343
2: kd> dqs 0xFFFFF78000000800 L1
fffff780`00000800  43434343`43434343
```

Arbitrary Read Primitive

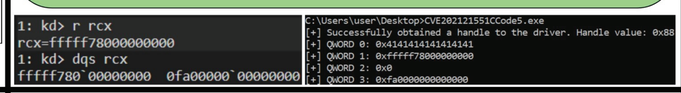
Find another IOCTL code to access the function containing memmove(). Use this as a parameter for DeviceIoControl() to interact with the driver

```
.text:00000000000111E5                  cmp     eax, 9B0C1EC4h
```

1. Write C code to hit the breakpoint
2. Attach the VM to WinDbg and create a breakpoint
3. Run the code in the VM

```
0: kd> bp dbutil_2_3+11e5
0: kd> bl
     0 e Disable Clear  fffff800`1bf811e5     0001 (0001) dbutil_2_3+0x11e5
```

By stepping through functions, value stored at KUSER_SHARED_DATA can be read by user



Patch analysis: The access control list was elevated. However, as system administrators can still replicate the vulnerability, the patch was **insufficient**.

CVE-2023-21768 (Windows afd.sys Driver)

CVSS Score: 7.8

Root cause: **Write-what-where** condition - unprivileged users can use the driver to **write to kernel memory**.

Perform static analysis in Ghidra: Find the difference between the vulnerable and patched version

Vulnerable

```
if (checkforuser == '\0') {
  **(uint **)(address + 0x18) = writtenvalue;
}
else {
                    /* probeforwrite */
  **(uint **)(address + 0x18) = writtenvalue;
}
```

Patched

```
if (checkforuser == '\0') {
  **(uint **)(address + 0x18) = writtenvalue;
}
else {
  ProbeForWrite(*(undefined8 *)(address + 0x18),4
  **(uint **)(address + 0x18) = writtenvalue;
}
```

Find prerequisites for buffer such as buffer size and configured IoCompletion Object in Ghidra

```
if ((bufferSize != 0x30) || (param_7 != 0)) {
v10 = ObReferenceObjectByHandle(v11, 2164, IoCompletionObjectType, a4, &v26, 0i64);
```

Find IOCTL code for AfdNotifySock() in IDA

```
.rdata:00000001C004F5F8                 dq offset AfdNotifySock
```

Write a POC code, attach the VM to WinDbg and run the POC code

```
0: kd> bp afd!AfdNotifySock
0: kd> bl
     0 e Disable Clear  fffff800`72c6fc30     0001 (0001) afd!AfdNotifySock

0: kd> g
Breakpoint 0 hit
afd!AfdNotifySock:
fffff800`72c6fc30 488bc4          mov     rax,rsp
```

The POC code attempts to write to the address supplied and results in a blue screen

```
3: kd> p
afd!AfdNotifyRemoveIoCompletion+0x251:
fffff800`72c6fb7d 8b442444        mov     eax,dword ptr [rsp+44h]
3: kd> p
afd!AfdNotifyRemoveIoCompletion+0x255:
fffff800`72c6fb81 8901            mov     dword ptr [rcx],eax
3: kd> p
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x00000050
                       (0xFFFFF78000000800,0x0000000000000003,0xFFFFF80072C6FB81,0x0000000000000002)

Driver at fault:
***       afd.sys - Address FFFFF80072C6FB81 base at FFFFF80072C00000, DateStamp 0c5c6994
.
.
A fatal system error has occurred.
Debugger entered on first try; Bugcheck callbacks have not been invoked.

A fatal system error has occurred.
```

KUSER_SHARED_DATA is read-only in Windows 11. When the kernel attempts to write to the address supplied, it results in a blue screen of death

```
3: kd> !pte fffff78000000000
                                           VA fffff78000000000
PXE at FFFF8341A0D06F78    PPE at FFFF8341A0DEF000    PDE at FFFF8341BDE00000    PTE at FFFF837BC0000000
contains 0000000011F02063  contains 0000000011F03063  contains 0000000011F04063  contains 8A0000011FF35021
pfn 11fe02     ---DA--KWEV  pfn 11fe03     ---DA--KWEV  pfn 11fe04     ---DA--KWEV  pfn 11ff35     ----A--KR-V
```

Before crashing, KUSER_SHARED_DATA's address was in RCX register and "1" was written to RAX register

```
rax=0000000000000001 rbx=0000000000000000 rcx=fffff78000000800
rdx=0000000000000000 rsi=0000000000000000 rdi=0000000000000000
rip=fffff80072c6fb81 rsp=ffff8006a0be2170 rbp=ffff8006a0be2ca0
 r8=ffffac052814fed0  r9=ffffac052985ce98 r10=ffffac0522aa5d50
r11=ffffac0528224930 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
```

Patch analysis: ProbeForWrite() function call was **sufficient** as it ensures that the user mode buffer resides in user mode portion of address space and not kernel mode.

Software developers can leverage this information to **enhance their patch monitoring practices**. They could consider incorporating a **dedicated monitoring period** to promptly **detect and respond** to any exploitation following the implementation of patches. Additionally, this knowledge can be used to **create robust patches** that offer **broader protection** against a range of potential vulnerabilities. The improvements in software security and privilege management practices can lead to **safer systems** for end users.

Members:
Low Li Ying Amy, Raffles Girls' School
Pak Tze Bin Bryan, Raffles Institution
Mentor:
Yap Ni, DSO National Laboratories

YDSP Young Defence Scientists Programme
DSTA Defence Science & Technology Agency
DSO NATIONAL LABORATORIES