Uncovering Kernel Exploits: Exploring Vulnerabilities in AMD's Windows Kernel Drivers

TeamT5 Engine Team－Zeze

TEAMT5
杜 浦 數 位 安 全
Persistent Cyber Threat Hunters

Zeze

- HITCON Staff
- TeamT5 Security Researcher
- Member of BambooFox and ⚔️TSJ⚔️
- 50+ Windows Kernel CVEs
- HITCON 2022, VXCON 2022, and CYBERSEC 2023 Speaker

Outline

TEAMT5
杜 浦 數 位 安 全

01 **Introduction** to Windows kernel exploit and its impact.

02 **Background** knowledge to the targets.

03 **Vulnerabilities** I found and how they were exploited.

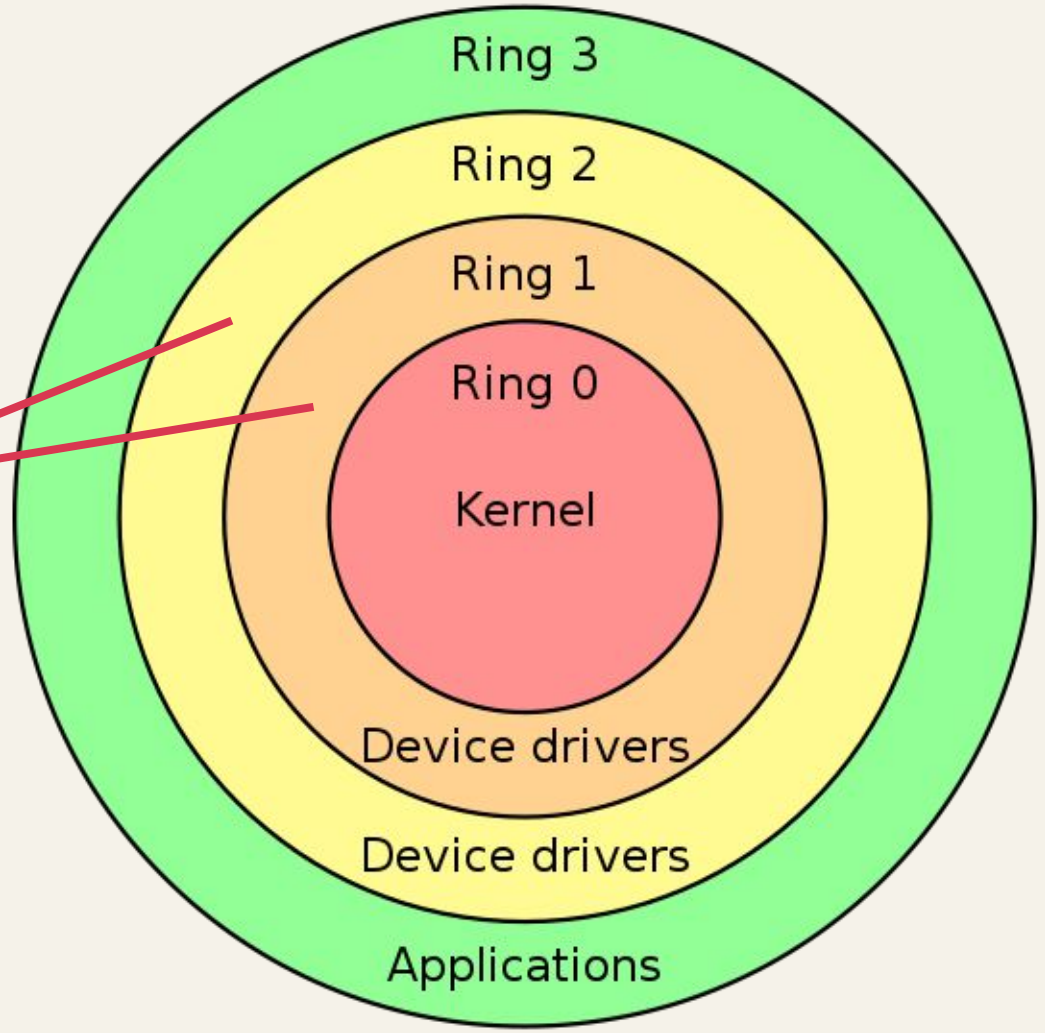
04 **Report** to AMD PSIRT and their response.

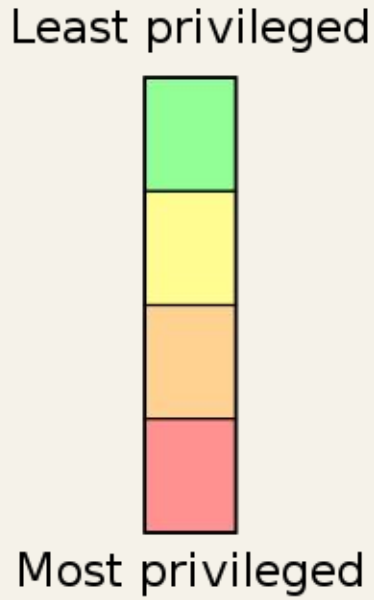
05 **Conclusion**

Introduction

Introduction to Windows kernel exploit and its impact.

Ring0 & Ring3

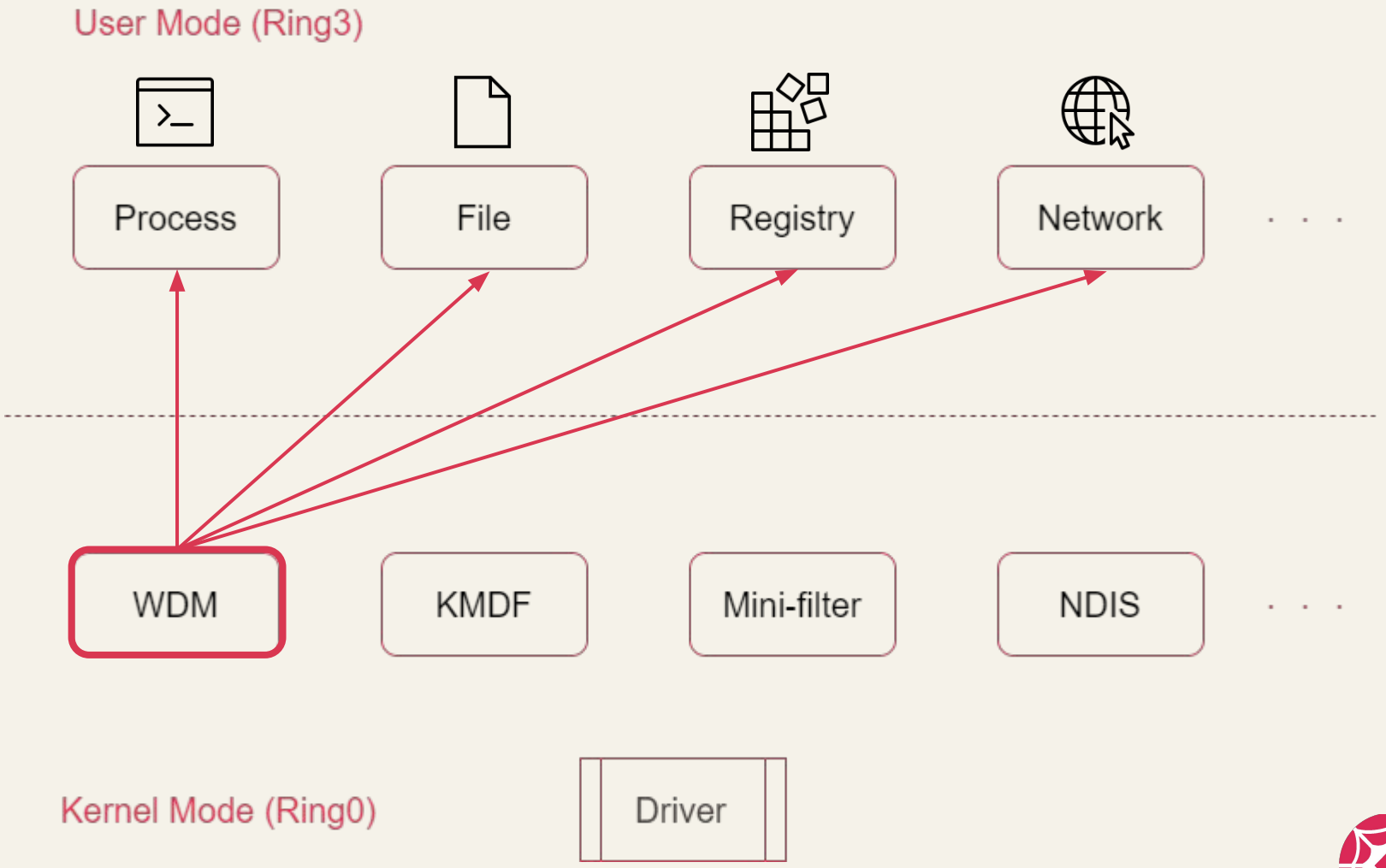




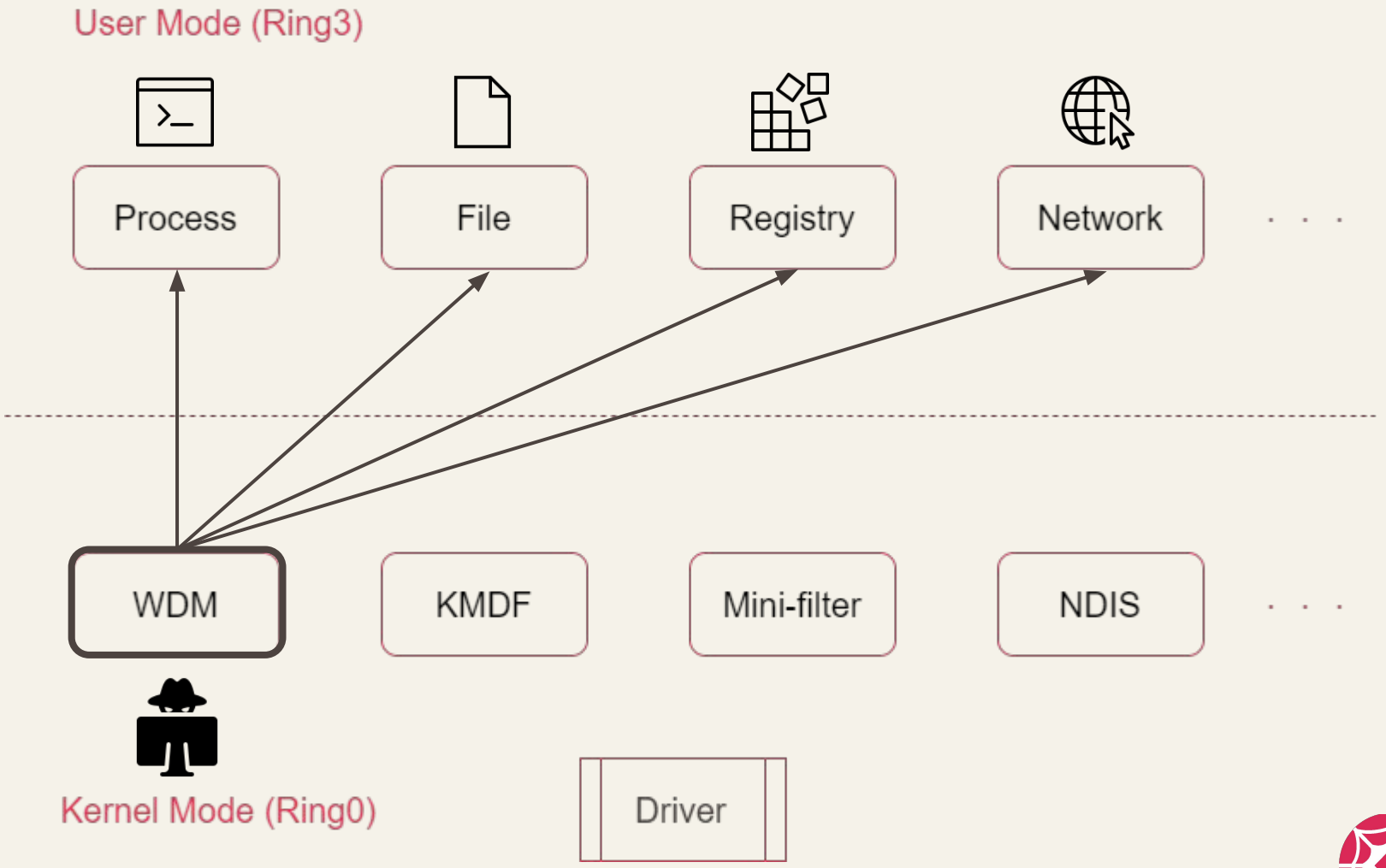
Windows reserves

Ring1 and Ring2.

Driver Types



Impact of Kernel Exploit



BYOVD

Simply explain BYOVD attack in 50 words

A BYOVD (**Bring Your Own Vulnerable Driver**) attack **targets weak kernel drivers**. Attackers exploit driver vulnerabilities to **gain unauthorized access**, **bypass security**, and **execute malicious code** with elevated privileges, compromising system integrity and user data. Regular updates and monitoring can help prevent and detect such attacks.

- ChatGPT

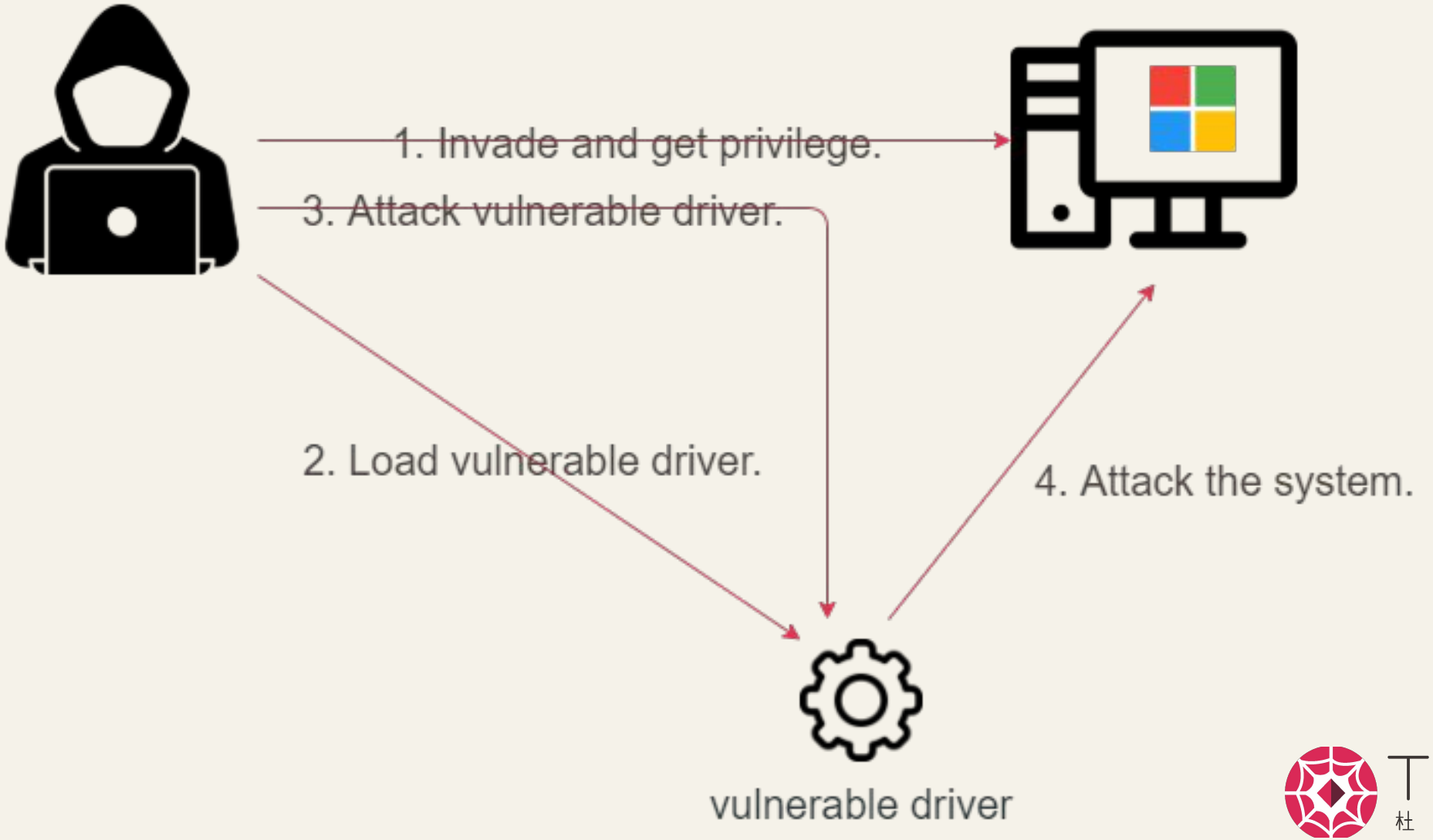
Driver Signing

Driver Signing

Windows device installation uses **digital signatures** to verify the integrity of driver packages and to verify the identity of the vendor (software publisher) who provides the driver packages. In addition, the kernel-mode code signing policy for **64-bit versions** of **Windows Vista and later versions** of Windows specifies that a kernel-mode driver must be signed for the driver to load.

- MSDN

BYOVD Diagram



AMD µProf

a software profiling analysis tool offering valuable event information specific to AMD "Zen" processors and AMD INSTINCT MI Series accelerators.

AMD Ryzen Master

AMD Ryzen™ Master is a utility provided by AMD that allows users to overclock and fine-tune their AMD Ryzen processors.

Contributions

TEAMT5
杜 浦 數 位 安 全

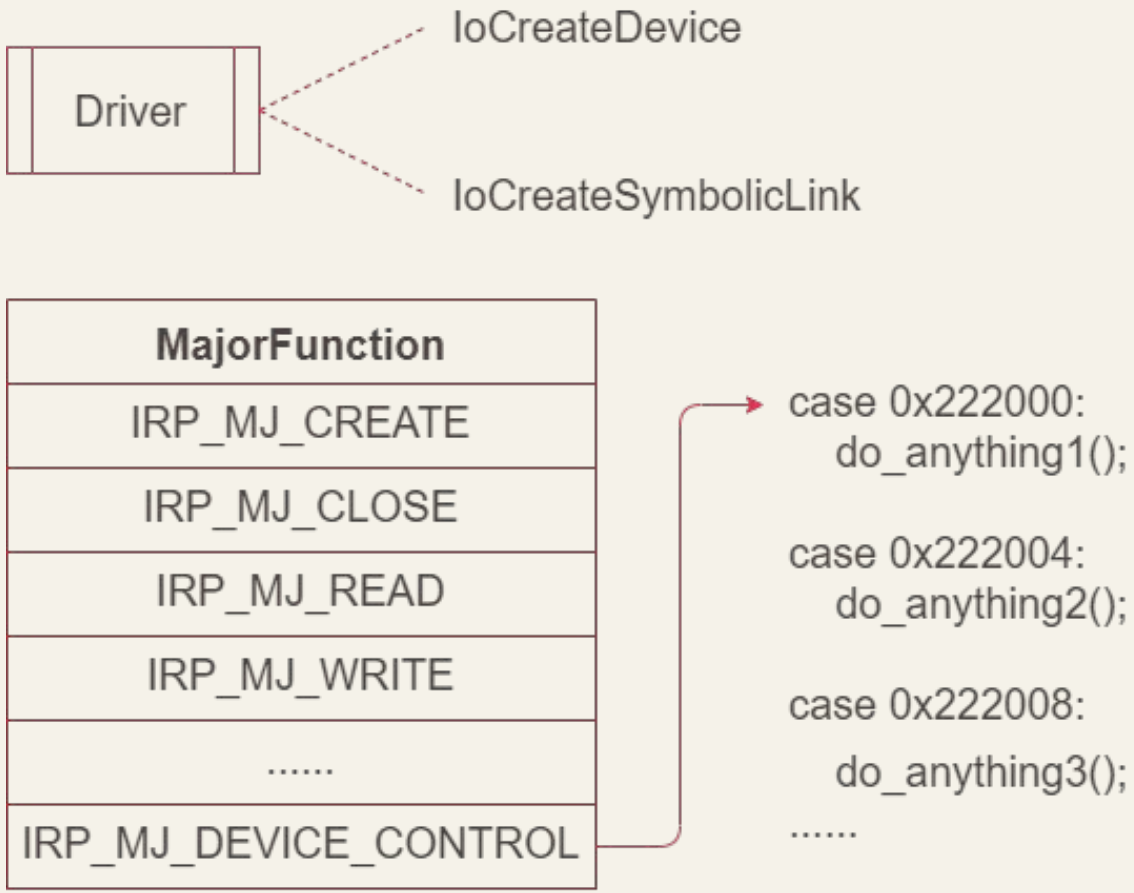
CVE	Product	Driver	Impact
CVE-2023-20560	AMD Ryzen Master	AMDRyzenMasterDriver.sys	DoS
CVE-2023-20564	AMD Ryzen Master	AMDRyzenMasterDriver.sys	EoP
CVE-2023-20556	AMD μProf	AMDPowerProfiler.sys	DoS
CVE-2023-20561	AMD μProf	AMDCpuProfiler.sys	DoS
CVE-2023-20562	AMD μProf	AMDCpuProfiler.sys	EoP

Background

Background knowledge to the targets.

WDM Driver

1. Create a device.
2. Create a symbolic link for the device.
3. Define dispatch routines for each IRP.
4. Implement IOCTL handler.



IRP

I/O request packets

Most of the requests that are sent to device drivers are packaged in **I/O request packets** (IRPs). An operating system component or a driver sends an IRP to a driver by calling IoCallDriver, which has two parameters: a pointer to a **DEVICE_OBJECT** and a pointer to an **IRP**. The DEVICE_OBJECT has a pointer to an associated DRIVER_OBJECT.

- MSDN

IRP

User Mode (Ring3)

Process

IRP (I/O Request Package)

Win32 API	IRP Function Code
CreateFile	IRP_MJ_CREATE
CloseFile	IRP_MJ_CLOSE
ReadFile	IRP_MJ_READ
WriteFile	IRP_MJ_WRITE
DeviceIoControl	IRP_MJ_DEVICE_CONTROL

Driver

Kernel Mode (Ring0)

IOCTL

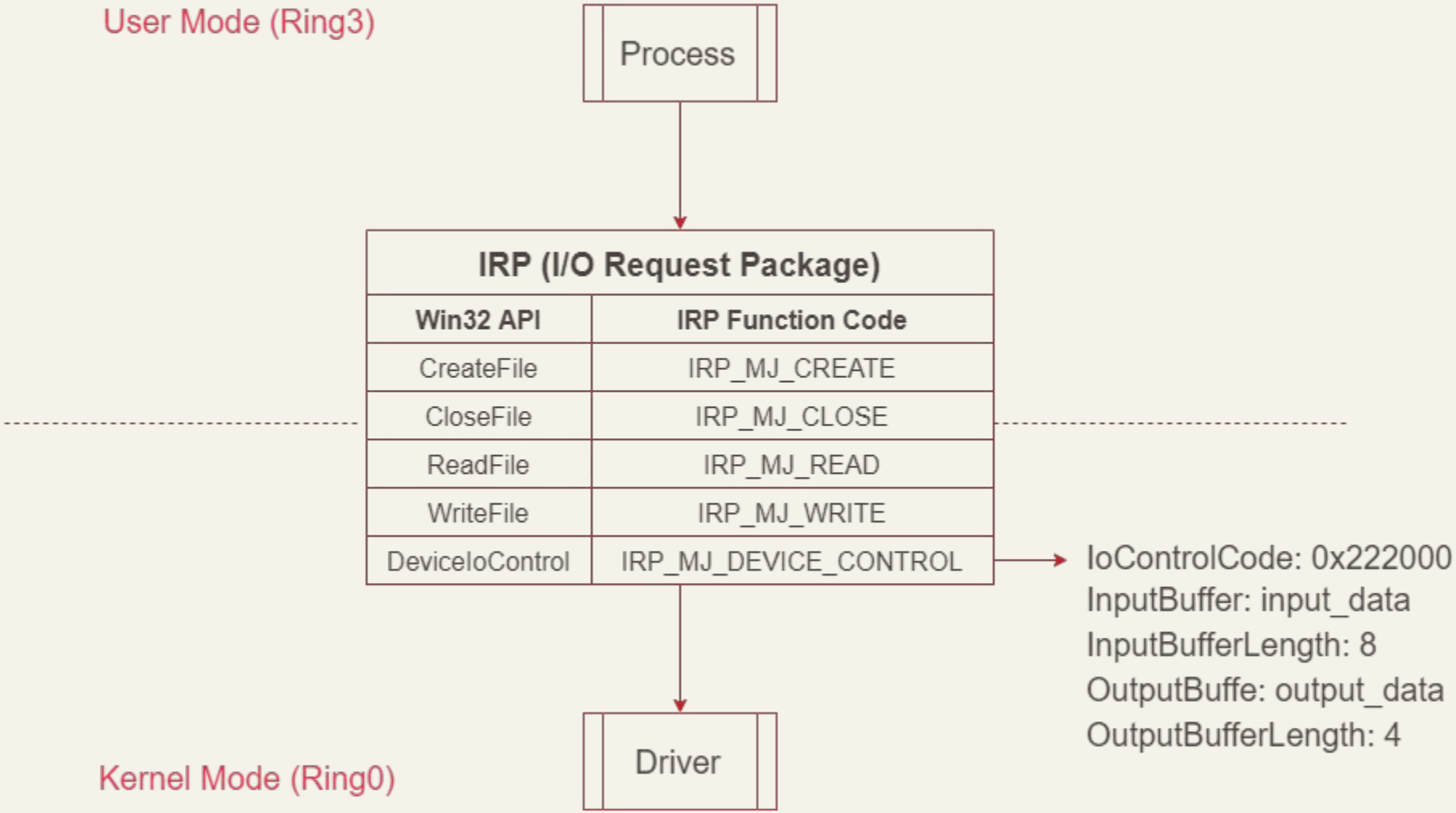
TEAMT5
杜 浦 數 位 安 全

Device Input and Output Control (IOCTL)

The DeviceIoControl function provides a **device input and output control** (IOCTL) interface through which an application can communicate directly with a device driver. The **DeviceIoControl** function is a general-purpose interface that can send control codes to a variety of devices. **Each control code represents an operation for the driver to perform.**

- MSDN

IOCTL


User Mode (Ring3)
Process
IRP (I/O Request Package)
Win32 API
IRP Function Code
CreateFile
IRP_MJ_CREATE
CloseFile
IRP_MJ_CLOSE
ReadFile
IRP_MJ_READ
WriteFile
IRP_MJ_WRITE
DeviceIoControl
IRP_MJ_DEVICE_CONTROL
IoControlCode: 0x222000
InputBuffer: input_data
InputBufferLength: 8
OutputBuffe: output_data
OutputBufferLength: 4
Driver
Kernel Mode (Ring0)

Kernel Fuzzer

TEAMT5
杜 浦 數 位 安 全

koutto/ioctlbf

Aims to fuzz WDM drivers by providing the symbolic link name and IoControlCode.

k0keoyo/kDriver-Fuzzer

Extension of ioctlbf that enhances its functionality by supporting features such as logging and fuzzing by filling null values into the input buffer.

What do we need for fuzzing?

TEAMT5 杜浦數位安全

DeviceIoControl

```
BOOL DeviceIoControl(
  [in]                HANDLE        hDevice,
  [in]                DWORD         dwIoControlCode,
  [in, optional]      LPVOID        lpInBuffer,
  [in]                DWORD         nInBufferSize,
  [out, optional]     LPVOID        lpOutBuffer,
  [in]                DWORD         nOutBufferSize,
  [out, optional]     LPDWORD       lpBytesReturned,
  [in, out, optional] LPOVERLAPPED  lpOverlapped
);
```

Symbolic Link Name

a range of control codes

not important

What do we need for fuzzing?

TEAMT5 杜浦數位安全

DeviceIoControl

```
BOOL DeviceIoControl(
  [in]                HANDLE        hDevice,
  [in]                DWORD         dwIoControlCode,
  [in, optional]      LPVOID        lpInBuffer,
  [in]                DWORD         nInBufferSize,
  [out, optional]     LPVOID        lpOutBuffer,
  [in]                DWORD         nOutBufferSize,
  [out, optional]     LPDWORD       lpBytesReturned,
  [in, out, optional] LPOVERLAPPED lpOverlapped
);
```

Buffers to be sent or received which may not be properly handled.

What do we need for fuzzing?

TEAMT5 杜浦數位安全

fuzzer.exe -d <SYMBOLIC_LINK_NAME> -i <IOCTL_CODE>

Vulnerabilities

Vulnerabilities I found and how they were exploited.

CVE-2023-1643	A vulnerability has been found in IObit Malware Fighter 9.4.0.776 and classified as problematic. Affected by this vulnerability is the function 0x8001E000/0x8001E004/0x8001E018/0x8001E01C/0x8001E024/0x8001E040 in the library ImfHpRegFilter.sys of the component IOCTL Handler. The manipulation leads to denial of service. Attacking locally is a requirement. The exploit has been disclosed to the public and may be used. The associated identifier of this vulnerability is VDB-224023.
CVE-2023-1642	A vulnerability, which was classified as problematic, was found in IObit Malware Fighter 9.4.0.776. Affected is the function 0x222034/0x222038/0x22203C/0x222040 in the library ObCallbackProcess.sys of the component IOCTL Handler. The manipulation leads to denial of service. Local access is required to approach this attack. The exploit has been disclosed to the public and may be used. VDB-224022 is the identifier assigned to this vulnerability.
CVE-2023-1641	A vulnerability, which was classified as problematic, has been found in IObit Malware Fighter 9.4.0.776. This issue affects the function 0x222018 in the library ObCallbackProcess.sys of the component IOCTL Handler. The manipulation leads to denial of service. An attack has to be approached locally. The exploit has been disclosed to the public and may be used. The identifier VDB-224021 was assigned to this vulnerability.
CVE-2023-1640	A vulnerability classified as problematic was found in IObit Malware Fighter 9.4.0.776. This vulnerability affects the function 0x222010 in the library ObCallbackProcess.sys of the component IOCTL Handler. The manipulation leads to denial of service. The attack needs to be approached locally. The exploit has been disclosed to the public and may be used. The identifier of this vulnerability is VDB-224020.
CVE-2023-1639	A vulnerability classified as problematic has been found in IObit Malware Fighter 9.4.0.776. This affects the function 0x8001E04C in the library ImfRegistryFilter.sys of the component IOCTL Handler. The manipulation leads to denial of service. It is possible to launch the attack on the local host. The exploit has been disclosed to the public and may be used. The associated identifier of this vulnerability is VDB-224019.
CVE-2023-1638	A vulnerability was found in IObit Malware Fighter 9.4.0.776. It has been rated as problematic. Affected by this issue is the function 0x8001E024/0x8001E040 in the library ImfRegistryFilter.sys of the component IOCTL Handler. The manipulation leads to denial of service. Attacking locally is a requirement. The exploit has been disclosed to the public and may be used. VDB-224018 is the identifier assigned to this vulnerability.
CVE-2023-1631	A vulnerability, which was classified as problematic, was found in JiangMin Antivirus 16.2.2022.418. This affects the function 0x222010 in the library kvcore.sys of the component IOCTL Handler. The manipulation leads to null pointer dereference. Attacking locally is a requirement. The exploit has been disclosed to the public and may be used. The identifier VDB-224013 was assigned to this vulnerability.
CVE-2023-1630	A vulnerability, which was classified as problematic, has been found in JiangMin Antivirus 16.2.2022.418. Affected by this issue is the function 0x222000 in the library kvcore.sys of the component IOCTL Handler. The manipulation leads to denial of service. Local access is required to approach this attack. The exploit has been disclosed to the public and may be used. The identifier of this vulnerability is VDB-224012.
CVE-2023-1629	A vulnerability classified as critical was found in JiangMin Antivirus 16.2.2022.418. Affected by this vulnerability is the function 0x222010 in the library kvcore.sys of the component IOCTL Handler. The manipulation leads to memory corruption. An attack has to be approached locally. The exploit has been disclosed to the public and may be used. The associated identifier of this vulnerability is VDB-224011.
CVE-2023-1628	A vulnerability classified as problematic has been found in Jianming Antivirus 16.2.2022.418. Affected is an unknown function in the library kvcore.sys of the component IoControlCode Handler. The[illegible] identifier ass[illegible] ay be used. VDB-224010 is the
CVE-2023-1627	A vulnerabili[illegible] IoControlCo[illegible] identifier VD[illegible] kvcore.sys of the component [illegible]he public and may be used. The
CVE-2023-1626	A vulnerabili[illegible] e.sys of the component IoControlCode Handler. The manipulation leads to memory corruption. Attacking locally is a requirement. The exploit has been disclosed to the public and may be used. The identifier of this vulnerability is VDB-224008.
CVE-2023-1513	A flaw was found in KVM. When calling the KVM_GET_DEBUGREGS ioctl, on 32-bit systems, there might be some uninitialized portions of the kvm_debugregs structure that could be copied to userspace, causing an information leak.
CVE-2023-1493	A vulnerability was found in Max Secure Anti Virus Plus 19.0.2.1. It has been rated as problematic. This issue affects the function 0x220019 in the library MaxProctetor64.sys of the component IoControlCode Handler. The manipulation leads to denial of service. It is possible to launch the attack on the local host. The exploit has been disclosed to the public and may be used. The associated identifier of this vulnerability is VDB-223379.
CVE-2023-1492	A vulnerability was found in Max Secure Anti Virus Plus 19.0.2.1. It has been declared as problematic. This vulnerability affects the function 0x220019 in the library MaxProc64.sys of the component IoControlCode Handler. The manipulation of the argument SystemBuffer leads to denial of service. Attacking locally is a requirement. The exploit has been disclosed to the public and may be used. VDB-223378 is the identifier assigned to this vulnerability.
CVE-2023-1491	A vulnerability was found in Max Secure Anti Virus Plus 19.0.2.1. It has been classified as critical. This affects the function 0x220020 in the library MaxCryptMon.sys of the component

Keywords: ioctl & IoControlCode

Previous CVEs

CVE-2022-23831

Insufficient validation of the **IOCTL input buffer** in AMD μProf may allow an attacker to send an arbitrary buffer leading to a potential Windows kernel crash resulting in **denial of service**.

- MITRE CVE

Previous CVEs

CVE-2022-27674

Insufficient validation in the **IOCTL input/output buffer** in AMD µProf may allow an attacker to **bypass bounds checks** potentially leading to a Windows kernel crash resulting in **denial of service**.

- MITRE CVE

Testing Environment

- VirtualKD-Redux
- WinDbg
- Visual Studio 2017
- Dbgview
- KmdManager
- kDriver-Fuzzer

- Windows 10 1909
- Kernel Debug
- Test Signing
- Analysis Tools

CVE-2023-20560

TEAMT5
杜 浦 數 位 安 全

CVE	Product	Driver	Impact
CVE-2023-20560	**AMD Ryzen Master**	**AMDRyzenMasterDriver.sys**	**DoS**
CVE-2023-20564	AMD Ryzen Master	AMDRyzenMasterDriver.sys	EoP
CVE-2023-20556	AMD μProf	AMDPowerProfiler.sys	DoS
CVE-2023-20561	AMD μProf	AMDCpuProfiler.sys	DoS
CVE-2023-20562	AMD μProf	AMDCpuProfiler.sys	EoP

AMDRyzenMasterDriver.sys

TEAMT5 杜浦數位安全

DriverEntry

```
13  RtlInitUnicodeString(&DestinationString, SourceString);
14  RtlInitUnicodeString(&SymbolicLinkName, aDosdevicesAmdr);// \DosDevices\AMDRyzenMasterDriverV20
15  RtlInitUnicodeString(&v5, L"D:P(A;;GW;;;BA)(A;;GR;;;BA)");
16  v2 = sub_14000808C(
17          (__int64)DriverObject,
18          0,
19          (__int64)&DestinationString,
20          0x8111u,
21          256,
22          0,
23          (__int64)&v5,
24          0i64,
25          (__int64)&DeviceObject);
26  if ( v2 >= 0 )
27  {
28    v2 = IoCreateSymbolicLink(&SymbolicLinkName, &DestinationString);
29    if ( v2 < 0 )
30      DbgPrint("!!!RMDriver::DriverEntry(): IoCreateSymbolicLink() failed\n");
31    DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)IOCTL_Handler;
32    DriverObject->MajorFunction[2] = DriverObject->MajorFunction[14];
33    DriverObject->MajorFunction[0] = DriverObject->MajorFunction[2];
34    DriverObject->DriverUnload = (PDRIVER_UNLOAD)sub_140001F10;
```

IOCTL Handler

Symbolic Link Name

Fuzz AMDRyzenMasterDriver.sys

fuzzer.exe -d AMDRyzenMasterDriverV20 -i 0x81112F00

- IoControlCode: 0x81113000

CVE-2023-20560 - DoS

```
183             case 0x81113000:
184               *(_DWORD *)pSystemBuffer = 20;
185               goto LABEL_92;
```

```
*(_DWORD *)pSystemBuffer = 20;
```

CVE-2023-20564

TEAMT5
杜 浦 數 位 安 全

CVE	Product	Driver	Impact
CVE-2023-20560	AMD Ryzen Master	AMDRyzenMasterDriver.sys	DoS
CVE-2023-20564	**AMD Ryzen Master**	**AMDRyzenMasterDriver.sys**	**EoP**
CVE-2023-20556	AMD μProf	AMDPowerProfiler.sys	DoS
CVE-2023-20561	AMD μProf	AMDCpuProfiler.sys	DoS
CVE-2023-20562	AMD μProf	AMDCpuProfiler.sys	EoP

Read Physical Memory

```
258             case 0x81112F08:
259               if ( InputBufferLength >= 0xCui64
260                 && OutputBufferLength >= (unsigned __int64)*((unsigned int *)pSystemBuffer + 2) + 12 )
261               {
262                 _mm_lfence();
263                 if ( ReadPhysicalMemory(
264                        *(PHYSICAL_ADDRESS *)pSystemBuffer,
265                        *((_DWORD *)pSystemBuffer + 2),
266                        pSystemBuffer + 12) )
267                 {
268                   goto LABEL_92;
269                 }
270               }
271               break;
```

```
ReadPhysicalMemory(
  *(PHYSICAL_ADDRESS *)pSystemBuffer,
  *((_DWORD *)pSystemBuffer + 2),
  (pSystemBuffer + 12)
)
```

Read Physical Memory

```
11  BaseAddress = MmMapIoSpace(SystemBuffer_0, SystemBuffer_8, MmNonCached);
12  if ( BaseAddress )
13  {
14    switch ( SystemBuffer_8 )
15    {
16      case 1u:
17        *pSy
18        brea
19      case 2u:
20        *(_WORD *)pSystemBuffer_12 = *(_WORD *)BaseAddress;
21        break;
22      case 4u:
23        *(_DWORD *)pSystemBuffer_12 = *(_DWORD *)BaseAddress;
24        break;
25      case 8u:
26        *(_QWORD *)pSystemBuffer_12 = *(_QWORD *)BaseAddress;
27        break;
28      default:
29        for ( i = 0; i < SystemBuffer_8; ++i )
30          pSystemBuffer_12[i] = BaseAddress[i];
```

```
MmMapIoSpace(SystemBuffer_0, SystemBuffer_8, MmNonCached);
```

```
*(_QWORD *)pSystemBuffer_12 = *(_QWORD *)BaseAddress;
```

TEAMT5 杜浦數位安全

Write Physical Memory

```
272                 case 0x81112F0C:
273                   if ( InputBufferLength >= (unsigned __int64)*((unsigned int *)pSystemBuffer + 2) + 12 )
274                   {
275                     _mm_lfence();
276                     if ( WritePhysicalMemory(
277                            *(PHYSICAL_ADDRESS *)pSystemBuffer,
278                            *((_DWORD *)pSystemBuffer + 2),
279                            (__int64)(pSystemBuffer + 12)) )
280                     {
281 LABEL_92:
282                       v19 = 0;
283                       pIrp->IoStatus.Informati
284                     }
285                   }
286                   break;
```

```
WritePhysicalMemory(
  *(PHYSICAL_ADDRESS *)pSystemBuffer,
  *((_DWORD *)pSystemBuffer + 2),
  (__int64)(pSystemBuffer + 12)
)
```

Write Physical Memory

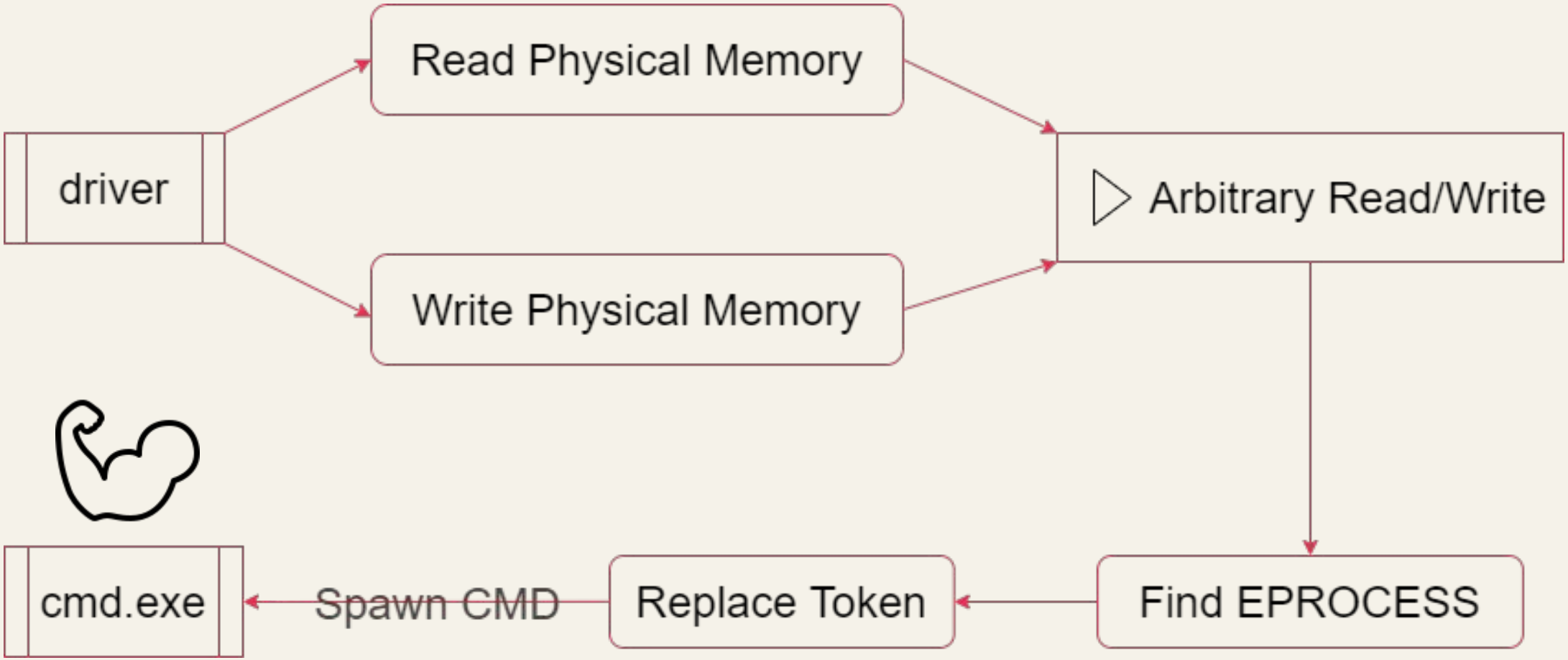
TEAMT5 杜浦數位安全

```
1  char __fastcall WritePhysicalMemory(
2          PHYSICAL_ADDRESS SystemBuffer_0,
3          unsigned int SystemBuffer_8,
4          __int64 pSystemBuffer_12)
5  {
6    char v4; // [rsp+20h] [rbp-8h]
7    unsigned int i; // [rsp+24h] [rbp-24h]
8    _BYTE *BaseAddress; // [rsp+28h] [rbp-20h]
9
10   v4 = 0;
11   BaseAddress = MmMapIoSpace(SystemBuffer_0, SystemBuffer_8, MmNonCached);
12   if ( BaseAddress )
13   {
14     for ( i = 0; i < SystemBuffer_8; ++i )
15       BaseAddress[i] = *(_BYTE *)(pSystemBuffer_12 + i);
16     MmUnmapIoSpace(BaseAddress, SystemBuffer_8);
17     v4 = 1;
18   }
19   return v4;
20 }
```

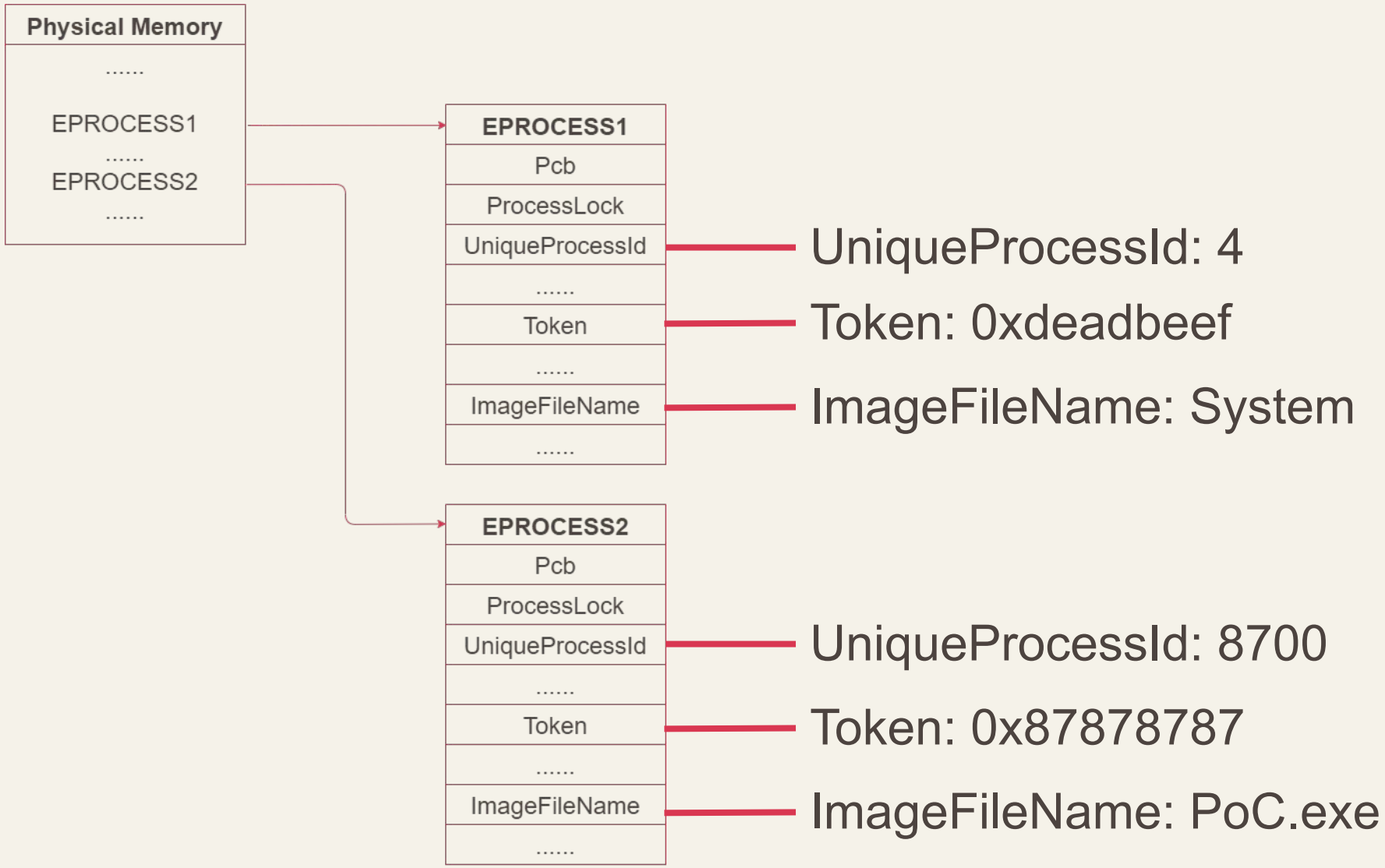
```
MmMapIoSpace(SystemBuffer_0, SystemBuffer_8, MmNonCached);
```

```
for ( i = 0; i < SystemBuffer_8; ++i )
  BaseAddress[i] = *(_BYTE *)(pSystemBuffer_12 + i);
```

Deeper In AMDRyzenMasterDriver.sys

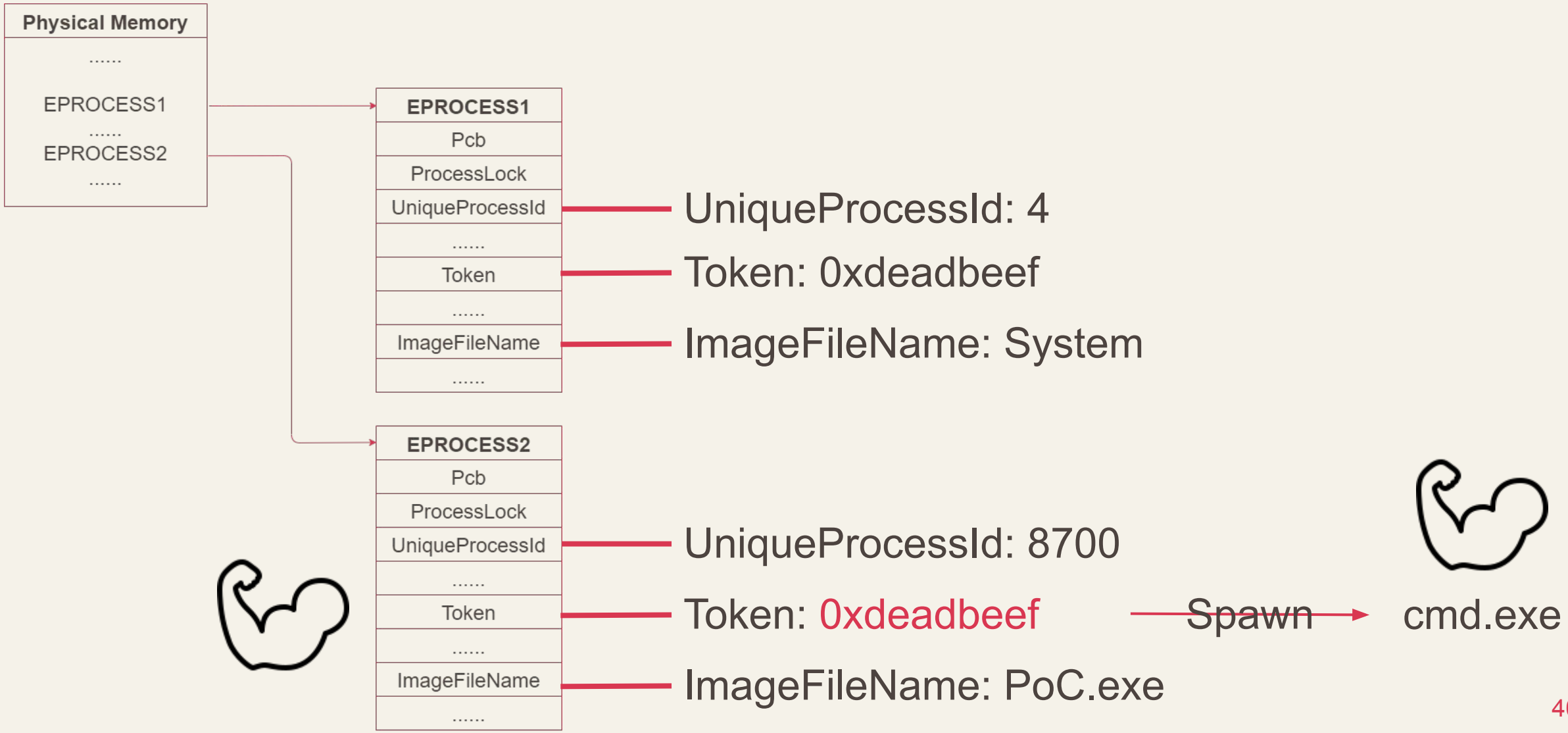


Replace Token



CVE-2023-20564 - EoP

TEAMT5 杜浦數位安全



CVE-2023-20556

TEAMT5
杜 浦 數 位 安 全

CVE	Product	Driver	Impact
CVE-2023-20560	AMD Ryzen Master	AMDRyzenMasterDriver.sys	DoS
CVE-2023-20564	AMD Ryzen Master	AMDRyzenMasterDriver.sys	EoP
CVE-2023-20556	**AMD µProf**	**AMDPowerProfiler.sys**	**DoS**
CVE-2023-20561	AMD µProf	AMDCpuProfiler.sys	DoS
CVE-2023-20562	AMD µProf	AMDCpuProfiler.sys	EoP

AMDPowerProfiler.sys

TEAMT5 杜浦數位安全

DriverEntry

IOCTL Handler

```
54    DriverObject->MajorFunction[0] = (PDRIVER_DISPATCH)&sub_140007990;
55    DriverObject->MajorFunction[2] = (PDRIVER_DISPATCH)&sub_140007990;
56    DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)IOCTL_Handler;
57    DriverObject->MajorFunction[18] = (PDRIVER_DISPATCH)sub_140007C90;
58    DriverObject->DriverUnload = (PDRIVER_UNLOAD)sub_140007040;
59    RtlInitUnicodeString(&SymbolicLinkName, L"\\??\\AMDPowerProfiler0");
60    if ( IoCreateSymbolicLink(&SymbolicLinkName, &DestinationString) )
61    {
62      IoDeleteDevice(DriverObject->DeviceObject);
63      return 322122473i64;
64    }
```

Symbolic Link Name

Fuzz AMDPowerProfiler.sys

TEAMT5 杜浦數位安全

fuzzer.exe -d AMDPowerProfiler0 -i 0x222000

- IoControlCode: 0x22201C
- InputBufferLength: 0x18
- OutputBufferLength: 0x18

IOCTL Handler

TEAMT5 杜浦數位安全

```
177     case 0x22201Cu:
178       if ( IoStack_v2->Parameters.Create.Options != 24 || IoStack_v2->Parameters.Read.Length != 24 )// 0x22201C
179         goto LABEL_130;
180       SystemBuffer = (char *)Pirp_a2->AssociatedIrp.MasterIrp;
181       if ( !SystemBuffer )
182         goto LABEL_150;
183       SystemBuffer_0 = *(unsigned int *)SystemBuffer;
184       v21 = 0;
185       if ( !(_DWORD)SystemBuffer_0 )
186       {
187         item_v22 = *( FILE_OBJECT **)(112 * SystemBuffer_0 + *(( QWORD *)DeviceExtension + 1) + 64);
```

```
if ( !MmIsAddressValid(*((PVOID *)SystemBuffer + 1)) || *((_DWORD *)SystemBuffer + 1) )
  goto LABEL_150;
memcpy(*((_OWORD **)SystemBuffer + 1), *(char **)(items + 40), 0x1000ui64);
```

```
195       if ( !v21 )
196         goto LABEL_151;
197       items = *((_QWORD *)DeviceExtension + 1) + 112 * SystemBuffer_0;
198       if ( !MmIsAddressValid(*((PVOID *)SystemBuffer + 1)) || *((_DWORD *)SystemBuffer + 1) )
199         goto LABEL_150;
200       memcpy(*((_OWORD **)SystemBuffer + 1), *(char **)(items + 40), 0x1000ui64);
201       *((_QWORD *)SystemBuffer + 2) = 1i64;
202       Pirp_a2->IoStatus.Information = 24i64;
203       goto LABEL_153;
```

Why can’t we fuzz the vuln?

TEAMT5 杜浦數位安全

The program needs a valid address.

```
if ( !MmIsAddressValid(*((PVOID *)SystemBuffer + 1)) || *((_DWORD *)SystemBuffer + 1) )
  goto LABEL_150;
memcpy(*((_OWORD **)SystemBuffer + 1), *(char **)(items + 40), 0x1000ui64);
```

CVE-2023-20556 - DoS

TEAMT5 杜浦數位安全

```
if ( !MmIsAddressValid(*((PVOID *)SystemBuffer + 1)) || *((_DWORD *)SystemBuffer + 1) )
  goto LABEL_150;
memcpy(*((_OWORD **)SystemBuffer + 1), *(char **)(items + 40), 0x1000ui64);
```

The structure “items” is not initialized.

CVE-2023-20561 & CVE-2023-20562

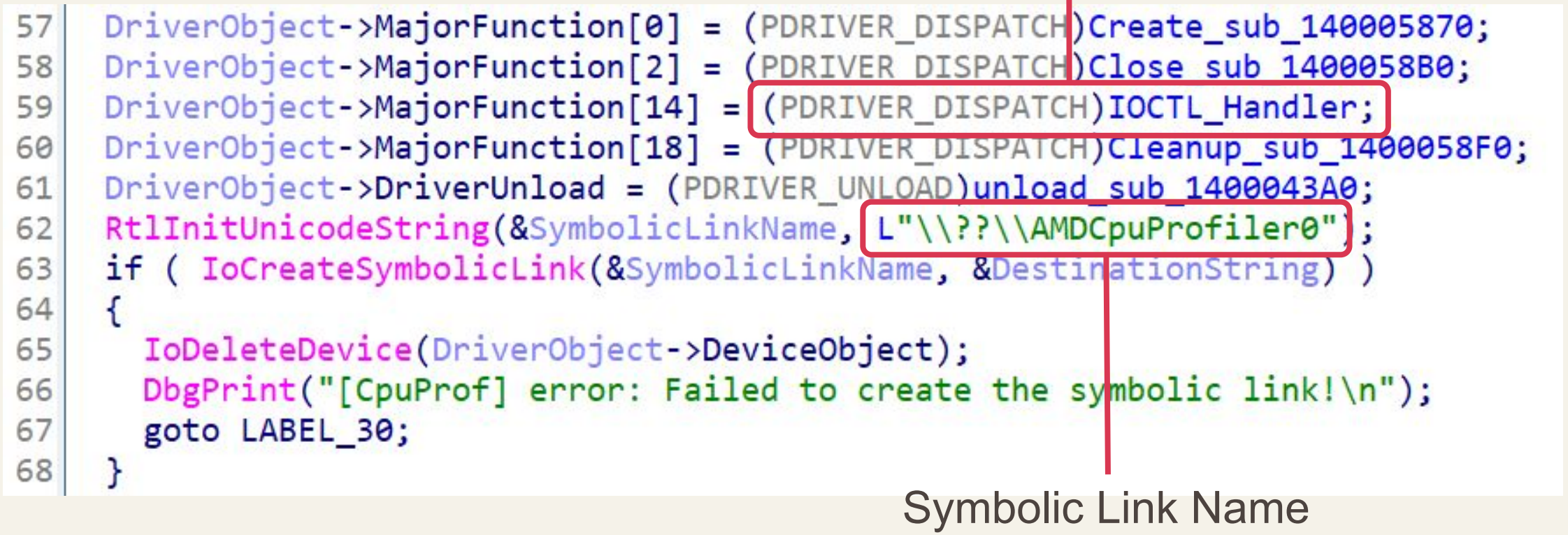
TEAMT5 杜浦數位安全

CVE	Product	Driver	Impact
CVE-2023-20560	AMD Ryzen Master	AMDRyzenMasterDriver.sys	DoS
CVE-2023-20564	AMD Ryzen Master	AMDRyzenMasterDriver.sys	EoP
CVE-2023-20556	AMD µProf	AMDPowerProfiler.sys	DoS
CVE-2023-20561	**AMD µProf**	**AMDCpuProfiler.sys**	**DoS**
CVE-2023-20562	**AMD µProf**	**AMDCpuProfiler.sys**	**EoP**

AMDCpuProfiler.sys

TEAMT5 杜浦數位安全

DriverEntry



Fuzz AMDCpuProfiler.sys

TEAMT5 杜浦數位安全

fuzzer.exe -d AMDCpuProfiler0 -i 0x222000

- IoControlCode: 0x222058
- InputBufferLength: 0x28
- OutputBufferLength: 0x28

IOCTL Handler

```
469   case 0x222058u:
470     DbgPrint("[CpuProf] Processing %s (Function: 0x%03X)...\n", "IOCTL", (IoControlCode >> 2) & 0xFFF);// 0x222058: IOCTL_GET_OUTPUT_FILE
471     IoStackLocation = pIrp->Tail.Overlay.CurrentStackLocation;
472     pIrp->IoStatus.Information = 0i64;
473     if ( IoStackLocation->Parameters.Create.Options != 0x28 )
474       goto LABEL_24;
475     OutputLength = IoStackLocation->Parameters.Read.Length;
476     v11 = OutputLength < 0x28;
477     if ( OutputLength < 0x28 )
478       goto LABEL_23;
479     SystemBuffer = (char *)pIrp->AssociatedIrp.MasterIrp;
480     status = OutputLength < 0x28 ? 0xC0000023 : 0;
481     SystemBuffer_0_v58 = *(unsigned int *)SystemBuffer;
482     if ( (unsigned int)SystemBuffer_0_v58 < 8
483       && (items = (__int64)&DeviceExtension_v5[0x948 * SystemBuffer_0_v58 + 32]) != 0
484       && *(_QWORD *)(items + 136) )
485     {
486       SystemBuffer_32 = *((_DWORD *)SystemBuffer + 8);
487       SystemBuffer_24 = (_IRP *)*((_QWORD *)SystemBuffer + 3);
488       if ( !*(_QWORD *)items
489         || !*(_QWORD *)(items + 104)
490         || !(unsigned int)IOCTL_GET_OUTPUT_FILE(
491                             (void **)items,
492                             *((_QWORD *)SystemBuffer + 1),
493                             *((_DWORD *)SystemBuffer + 4))
494         || !(unsigned int)IOCTL_GET_OUTPUT_FILE((void **)(items + 104), (__int64)SystemBuffer_24, SystemBuffer_32) )
495       {
496         status = 0xC00000E8;
497         if ( (*(_DWORD *)(items + 220) & 1) == 0 )
498           status = 0xC000000F;
499       }
500       *((_DWORD *)SystemBuffer + 9) = 0;
501       status_v3 = status;
502       pIrp->IoStatus.Information = 40i64;
503     }
504     else
505     {
506       DbgPrint(
507         "[CpuProf] error: IOCTL_GET_OUTPUT_FILE request with invliad client ID (%u).\n",
508         *(unsigned int *)SystemBuffer);
509       status_v3 = 0xC0000022;
510     }
511     goto LABEL_228;
```

```
IOCTL_GET_OUTPUT_FILE(
  (void **)items,
  *((_QWORD *)SystemBuffer + 1),
  *((_DWORD *)SystemBuffer + 4)
);
```

IOCTL_GET_OUTPUT_FILE

TEAMT5 杜浦數位安全

```
__int64 __fastcall IOCTL_GET_OUTPUT_FILE(void **items, _OWORD *SystemBuffer_8, int SystemBuffer_16)
{
  void *v3; // rcx
  unsigned int len; // ecx
  unsigned int v7; // ebx
  __int64 result; // rax
  struct _IO_STATUS_BLOCK IoStatusBlock; // [rsp+30h] [rbp-228h] BYREF
  unsigned int FileInformation; // [rsp+40h] [rbp-218h] BYREF
  __int16 filepath[262]; // [rsp+44h] [rbp-214h] BYREF

  v3 = *items;
  if ( !v3 )
    return 0i64;
  FileInformation = 0;
  filepath[0] = 0;
  if ( ZwQueryInformationFile(v3, &IoStatusBlock, &FileInformation, 0x20Eu, FileNameInformation) )
    return 0i64;
  if ( IoStatusBlock.Status
    return 0i64;
  len = FileInformation;
  if ( FileInformation <= 2 )
    return 0i64;
  if ( FileInformation >= 2 * SystemBuffer_16 )
    len = 2 * SystemBuffer_16;
  v7 = len;
  memcpy(SystemBuffer_8, filepath, len);
  result = v7 >> 1;
  *((_WORD *)SystemBuffer_8 + result) = 0;
  return result;
}
```

memcpy(SystemBuffer_8, filepath, len);

CVE-2023-20561 - DoS

TEAMT5
杜 浦 數 位 安 全

```
memcpy(SystemBuffer_8, filepath, len);
```

Attackers can control an arbitrary address to write.

Deeper In AMDCpuProfiler.sys

TEAMT5 杜浦數位安全

```
memcpy(SystemBuffer_8, filepath, len);
```

How about the file path?

Control File Path

```
410     case 0x222044u:
411       status_v3 = IOCTL_SET_OUTPUT_FILE((__int64)DeviceObject->DeviceExtension, pIrp, IoStack);
412       goto LABEL_228;
```


```
IOCTL_SET_OUTPUT_FILE((__int64)DeviceObject->DeviceExtension, pIrp, IoStack);
```

IOCTL_SET_OUTPUT_FILE

```
156           PerformanceCounter_v20 = KeQueryPerformanceCounter(&PerformanceFrequency).QuadPart;
157           *(_QWORD *)(items + 224) = PerformanceCounter_v20;
158           PerformanceFrequency_v37 = PerformanceFrequency.QuadPart;
159           sub_140007FC0(items);
160           if ( CreateFile((PHANDLE)items, SystemBuffer_8, v25)
161             && HasError(items, PerformanceCounter_v20, PerformanceFrequency_v37)
162             && *(_QWORD *)items )
163           {
164             ZwClose(*(HANDLE *)items);
165             *(_QWORD *)items = 0i64;
166           }
167           if ( *(_QWORD *)items )
168           {
169             sub_14000A0B0(items + 104);
170             if ( CreateFile((PHANDLE)(items + 104), SystemBuffer, v19) )
171             {
172               PoolWithTag = ExAllocatePoolWithTag(PagedPool, 0x200000ui64, 'PupC');
173               *(_QWORD *)(items + 112) = PoolWithTag;
174               if ( PoolWithTag )
175               {
176                 _InterlockedExchangeAdd((volatile signed __int32 *)(items + 124), 1u);
177               }
178               else
179               {
180                 v22 = *(void **)(items + 104);
181                 if ( v22 )
182                 {
183                   ZwClose(v22);
184                   *(_QWORD *)(items + 104) = 0i64;
185                 }
186               }
187             }
```

```
CreateFile((PHANDLE)items, SystemBuffer_8, v25)
```

Create File

TEAMT5 杜浦數位安全

```
1  bool __fastcall CreateFile(PHANDLE FileHandle, char *SystemBuffer_8, int a3)
2  {
3    void *v4; // rcx
4    unsigned int length; // ebx
5    __int16 UnicodeFilepath[4]; // [rsp+60h] [rbp-2C8h] BYREF
6    char **v10; // [rsp+68h] [rbp-2C0h]
7    struct _IO_STATUS_BLOCK IoStatusBlock; // [rsp+70h] [rbp-2B8h] BYREF
8    struct _OBJECT_ATTRIBUTES ObjectAttributes; // [rsp+80h] [rbp-2A8h] BYREF
9    char *v13; // [rsp+B0h] [rbp-278h] BYREF
10   __int128 Dst[38]; // [rsp+B8h] [rbp-270h] BYREF
11
12   v4 = *FileHandle;
13   if ( v4 )
14   {
15     ZwClose(v4);
16     *FileHandle = 0i64;
17   }
18   if ( SystemBuffer_8 )
19   {
20     length = 2 * a3;
21     v13 = (char *)L"\\??\\C:\\";
22     memcpy(Dst, SystemBuffer_8, length + 2i64);
23     UnicodeFilepath[1] = 0x26C;
24     v10 = &v13;
25     ObjectAttributes.ObjectName = (PUNICODE_STRING)UnicodeFilepath;
26     UnicodeFilepath[0] = length + 8;
27     ObjectAttributes.Length = 48;
28     ObjectAttributes.RootDirectory = 0i64;
29     ObjectAttributes.Attributes = 576;
30     *(_QWORD *)&ObjectAttributes.SecurityDescriptor = 0i64;
31     if ( ZwCreateFile(FileHandle, 0x40000000u, &ObjectAttributes, &IoStatusBlock, 0i64, 0x80u, 0, 0, 0x20u, 0i64, 0)
32       || IoStatusBlock.Status )
33     {
34       *FileHandle = 0i64;
35     }
```

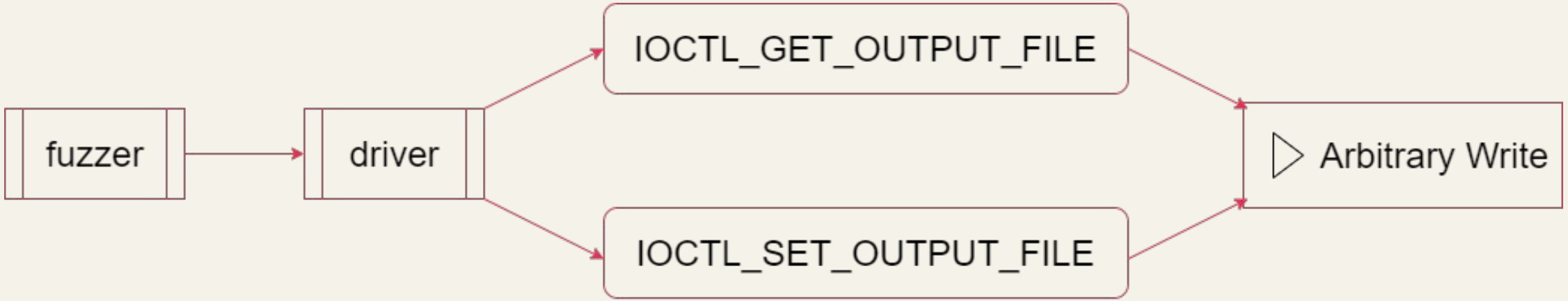
```
memcpy(Dst, SystemBuffer_8, length + 2i64);
```

```
ObjectAttributes.ObjectName = (PUNICODE_STRING)UnicodeFilepath;
```

```
ZwCreateFile(FileHandle, 0x40000000u, &ObjectAttributes, &IoStatusBlock, 0i64, 0x80u, 0, 0, 0x20u, 0i64, 0)
```

Arbitrary Write

TEAMT5 杜浦數位安全

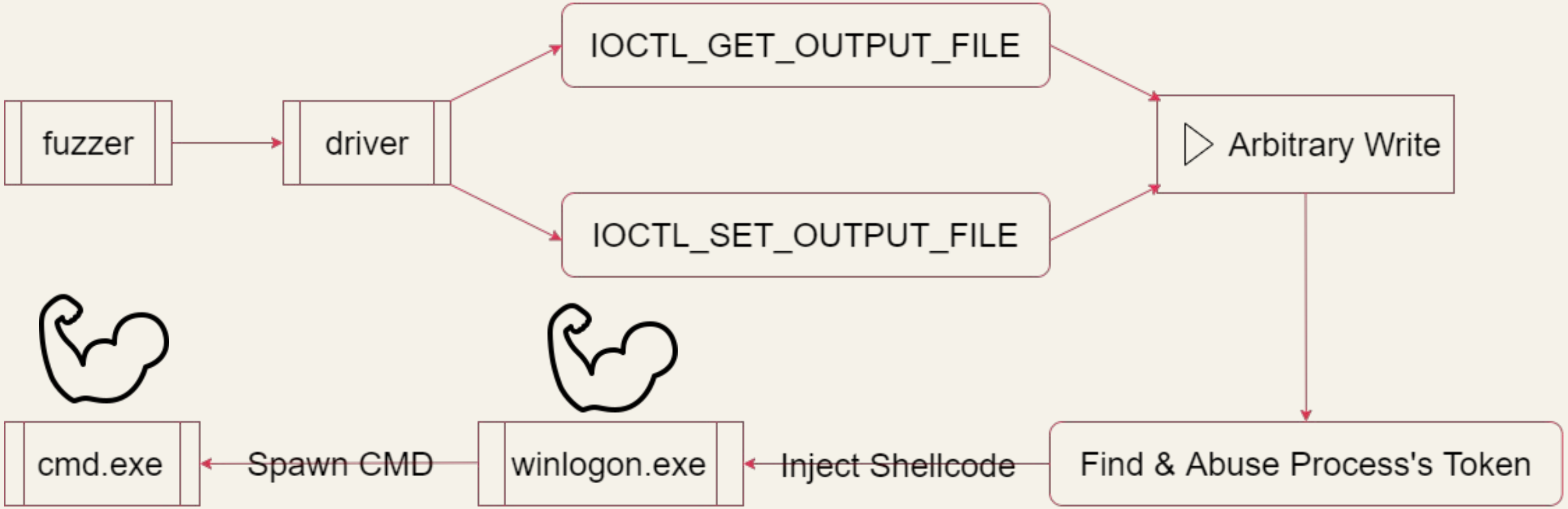


Restrictions for arbitrary write

1. wide char
2. valid file path

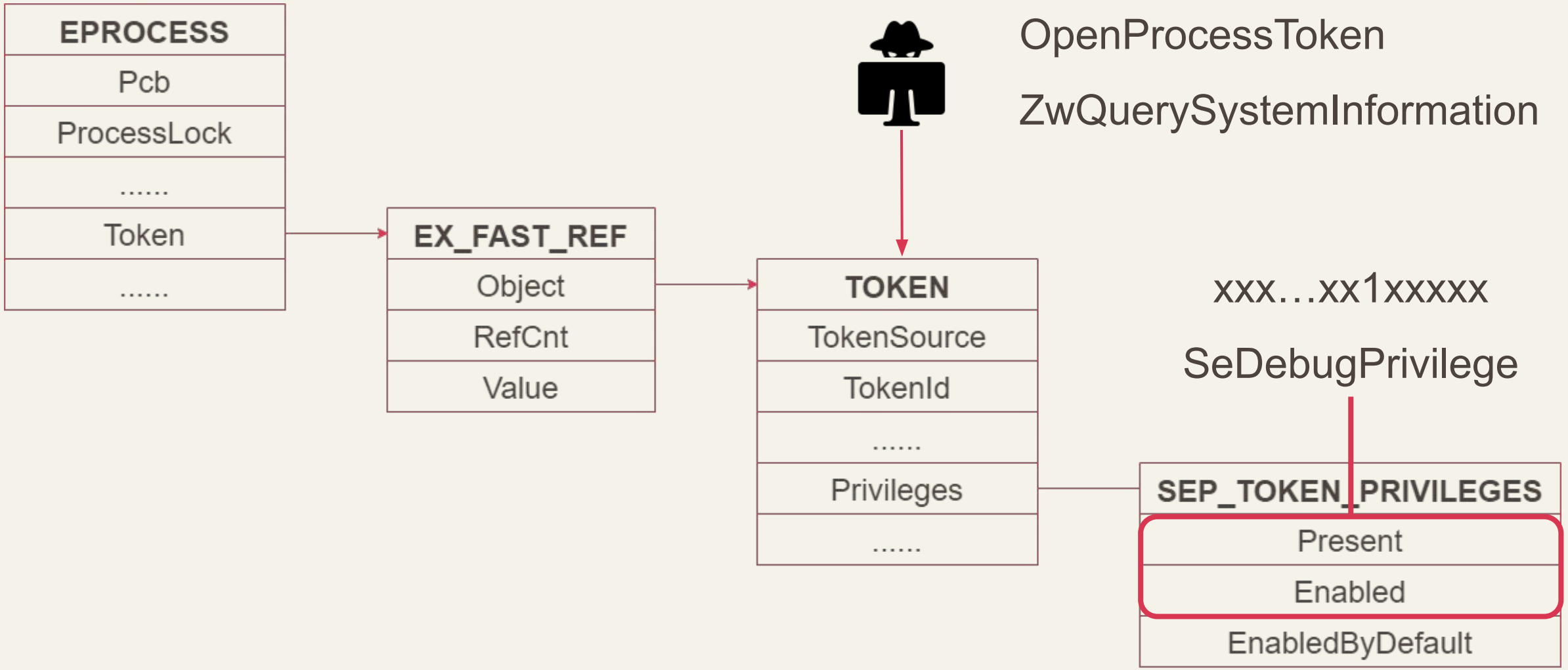
Exploit

TEAMT5 杜浦數位安全

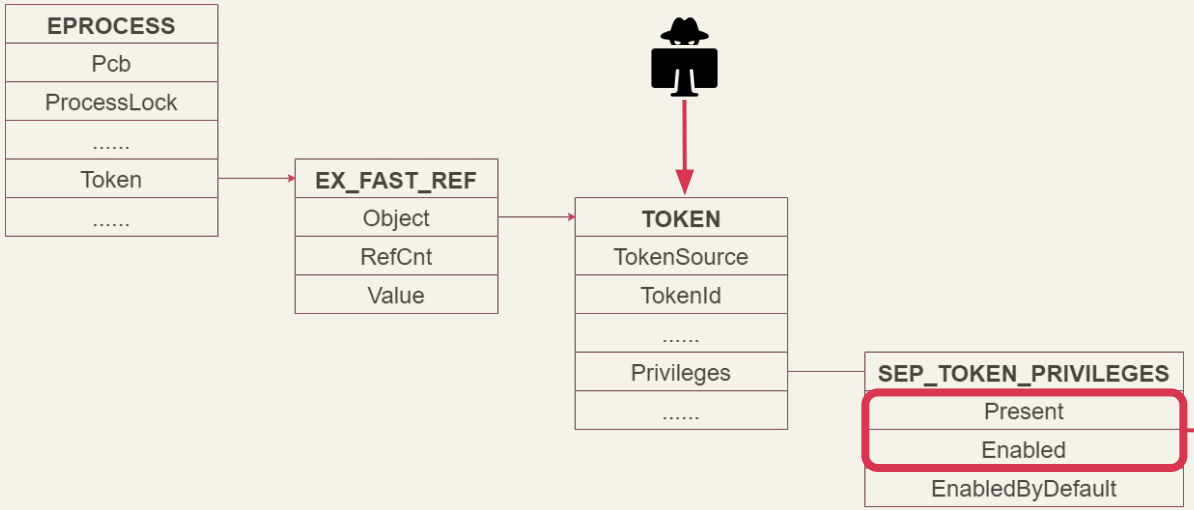


Find & Abuse Process's Token

TEAMT5 杜浦數位安全



CVE-2023-20562 - EoP



cmd.exe

Spawn

Inject Shellcode → winlogon.exe

1. OpenProcess
2. VirtualAllocEx
3. WriteProcessMemory
4. CreateRemoteThread

DEMO

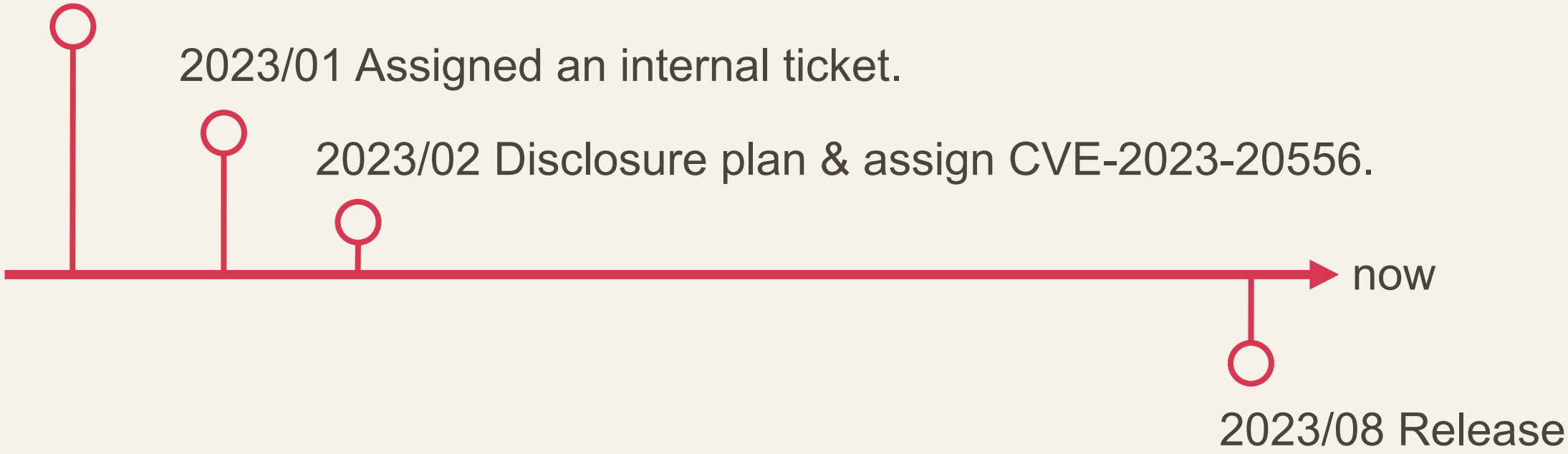
Report

Report to AMD PSIRT and their response.

Report Timeline - AMD uProf

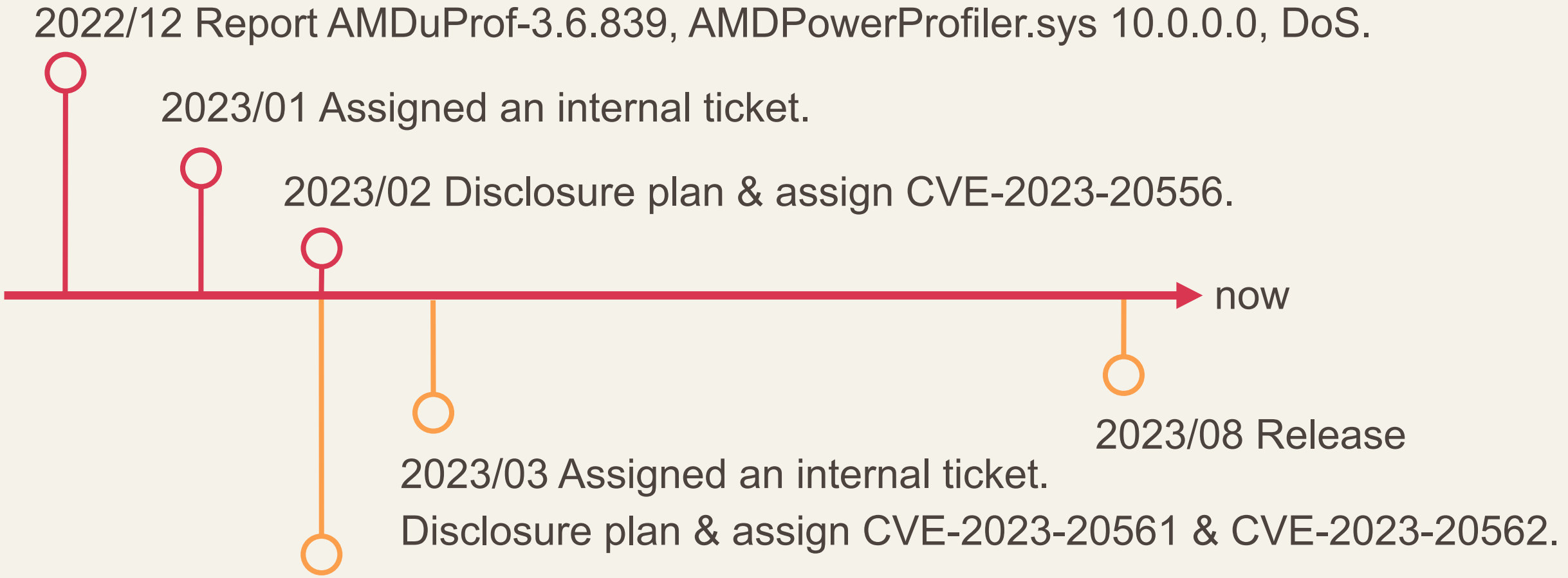
TEAMT5 杜浦數位安全

2022/12 Report AMDuProf-3.6.839, AMDPowerProfiler.sys 10.0.0.0, DoS.



Report Timeline - AMD uProf

TEAMT5 杜浦數位安全



Report Timeline - AMD Ryzen Master

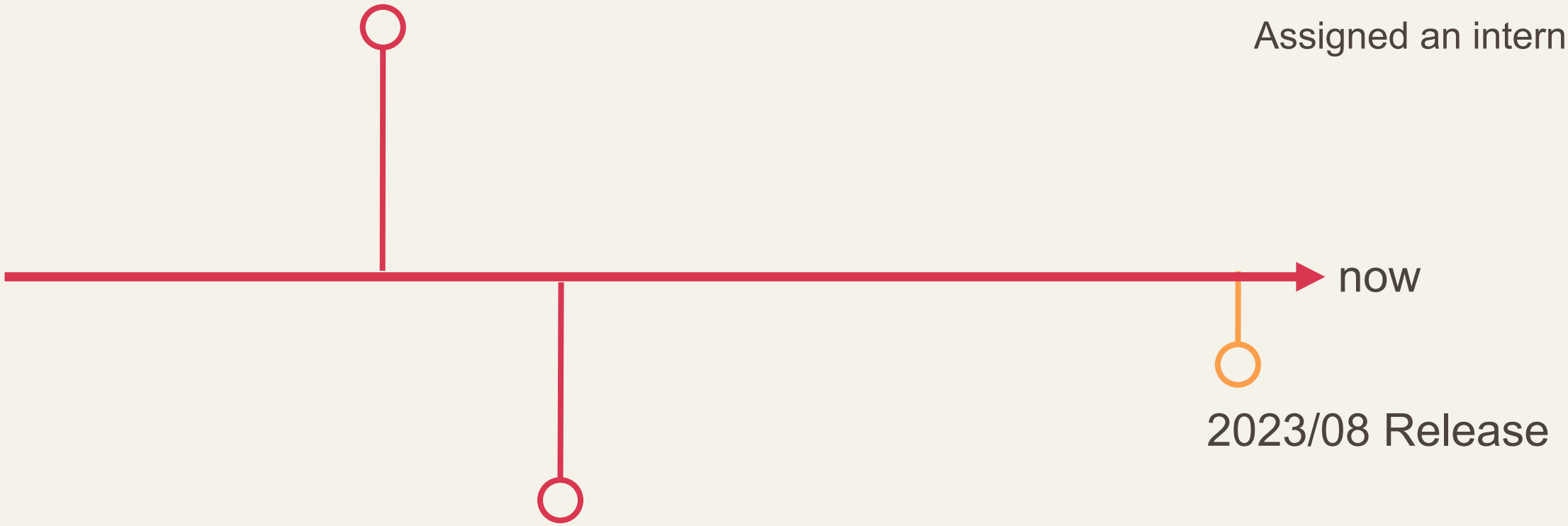
2023/02 Report AMD Ryzen Master 2.10.1.2287, AMDRyzenMasterDriver.sys 2.0.0.0, DoS & EoP.

Assigned an internal ticket.

Report Timeline - AMD Ryzen Master

2023/02 Report AMD Ryzen Master 2.10.1.2287, AMDRyzenMasterDriver.sys 2.0.0.0, DoS & EoP.

Assigned an internal ticket.



Acknowlegement

TEAMT5
杜 浦 數 位 安 全

Kenny

Angelboy

Conclusion

Conclusion

- WDM drivers constitute a significant portion of Windows kernel drivers.
- Through fuzzing and manual reverse-engineering, a total of 5 vulnerabilities were discovered and reported to AMD μProf and AMD Ryzen Master, consisting of 3 DoS and 2 EoP.
- AMD PSIRT demonstrated a commendable response to these security issues and actively addressed them.

THANKS FOR LISTENING !

Zeze

zeze@teamt5.org