TONGDAO

同道 KCon 2024

模仿游戏消亡史

汇报人：张云海

时间：2024.08.24

目录
CONTENT

KCon

PART ONE

01

背景

Background

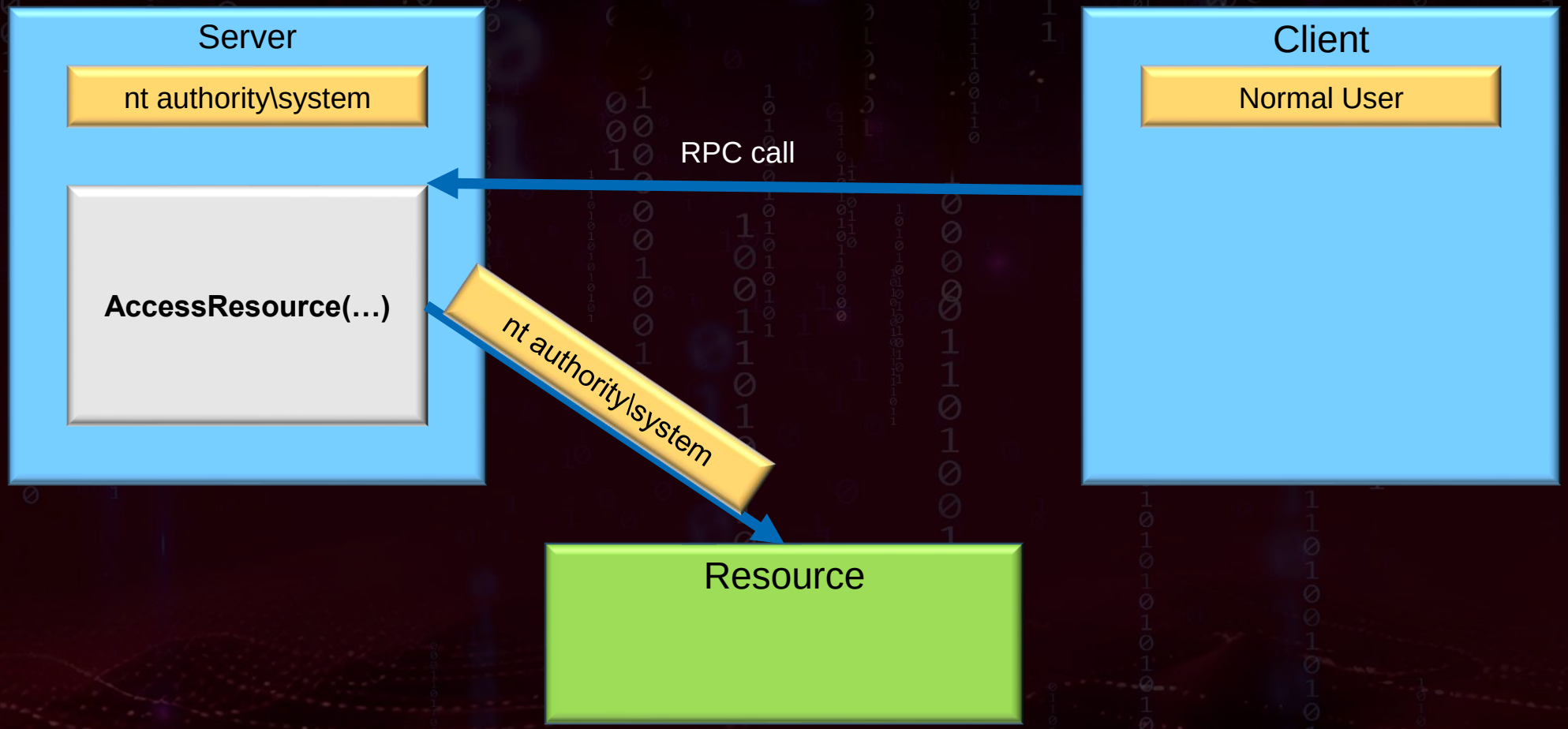
Impersonation 是什么

KCon

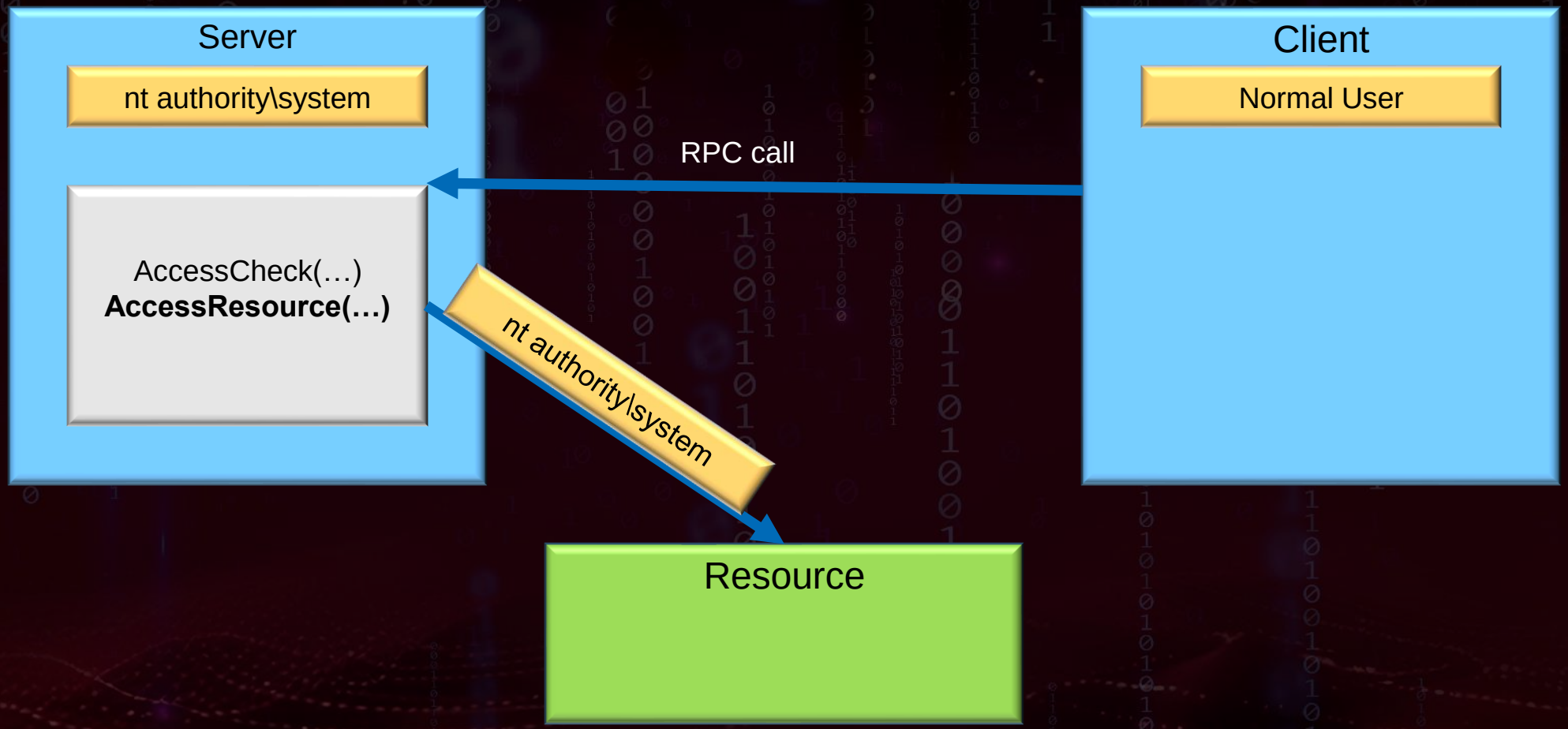
impersonation

A mechanism that allows a server process to run by using the security credentials of the client or some other user using the credentials. When the server is impersonating the client, any operations performed by the server are performed by using the client's (impersonating user's) credentials. Impersonation does not allow the server to access remote resources on behalf of the client. This requires delegation.

为何引入 Impersonation



为何引入 Impersonation



Impersonation 如何工作

```
nt!_EPROCESS
   …
   …
   +0x4b8 Token : _EX_FAST_REF
   …
   …
```

```
nt!_ETHREAD
   …
   …
   +0x608 AdjustedClientToken : Ptr64 Void
   …
   …
```

RpcImpersonateClient function (rpcdce.h)

Article • 02/23/2024

Feedback

A server thread that is processing client remote procedure calls can call the **RpcImpersonateClient** function to impersonate the active client.

Syntax

C++ Copy

```
RPC_STATUS RpcImpersonateClient(
  RPC_BINDING_HANDLE BindingHandle
);
```

RpcRevertToSelf function (rpcdce.h)

Article • 02/23/2024

Feedback

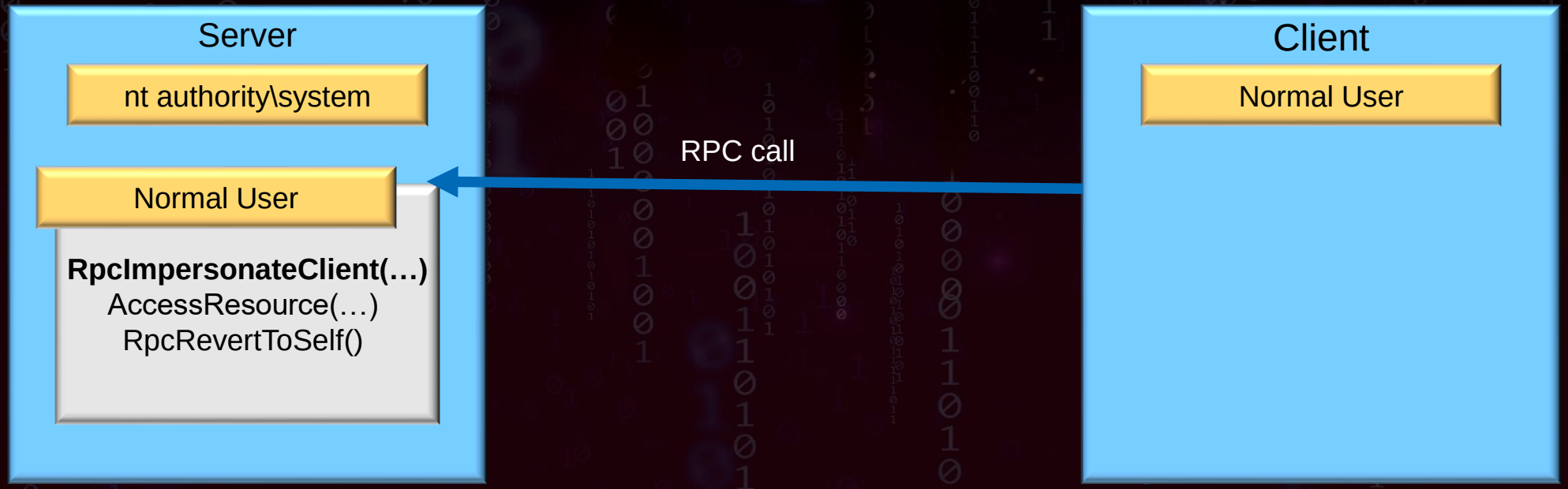
After calling RpcImpersonateClient and completing any tasks that require client impersonation, the server calls **RpcRevertToSelf** to end impersonation and to reestablish its own security identity.

Syntax

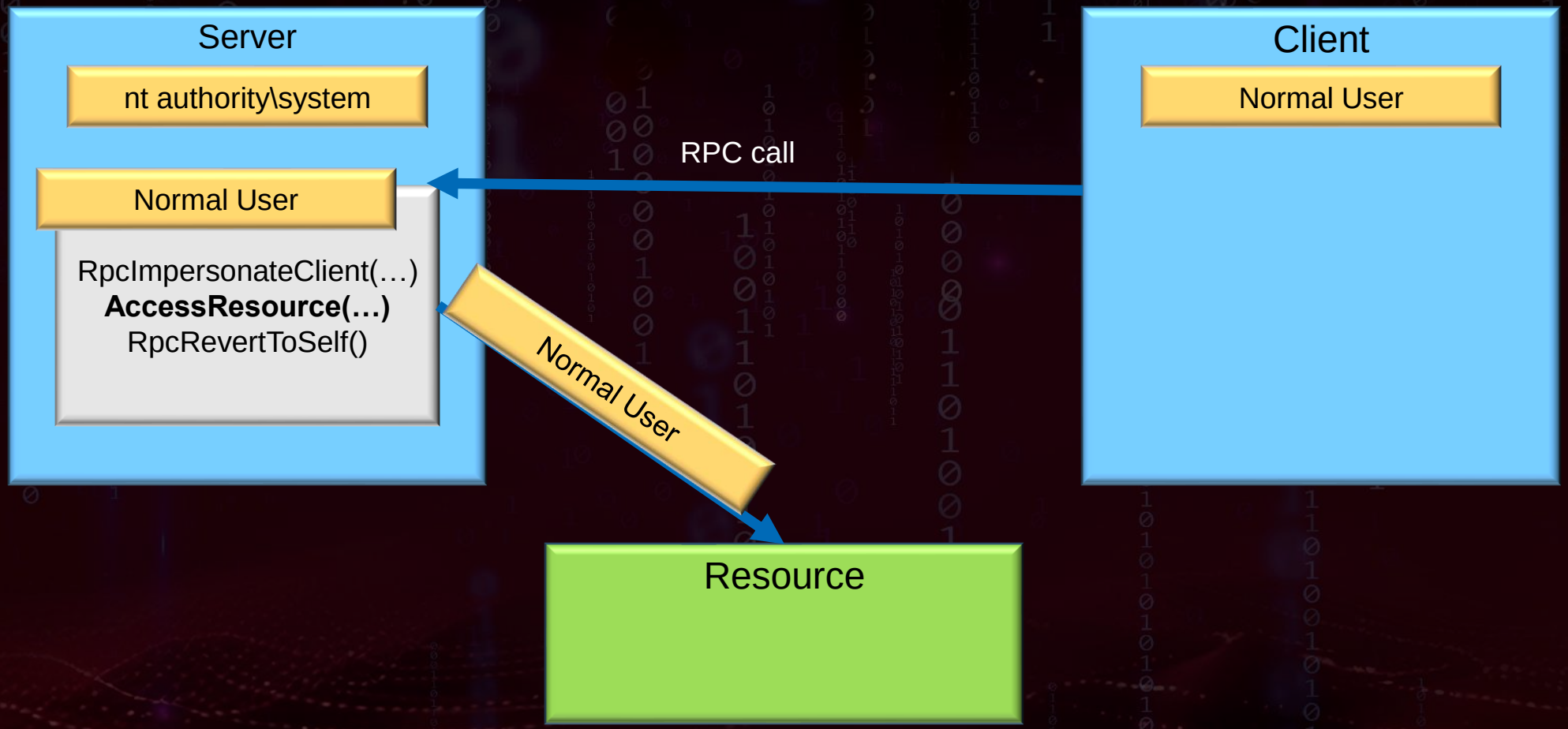
C++ Copy

```
RPC_STATUS RpcRevertToSelf();
```

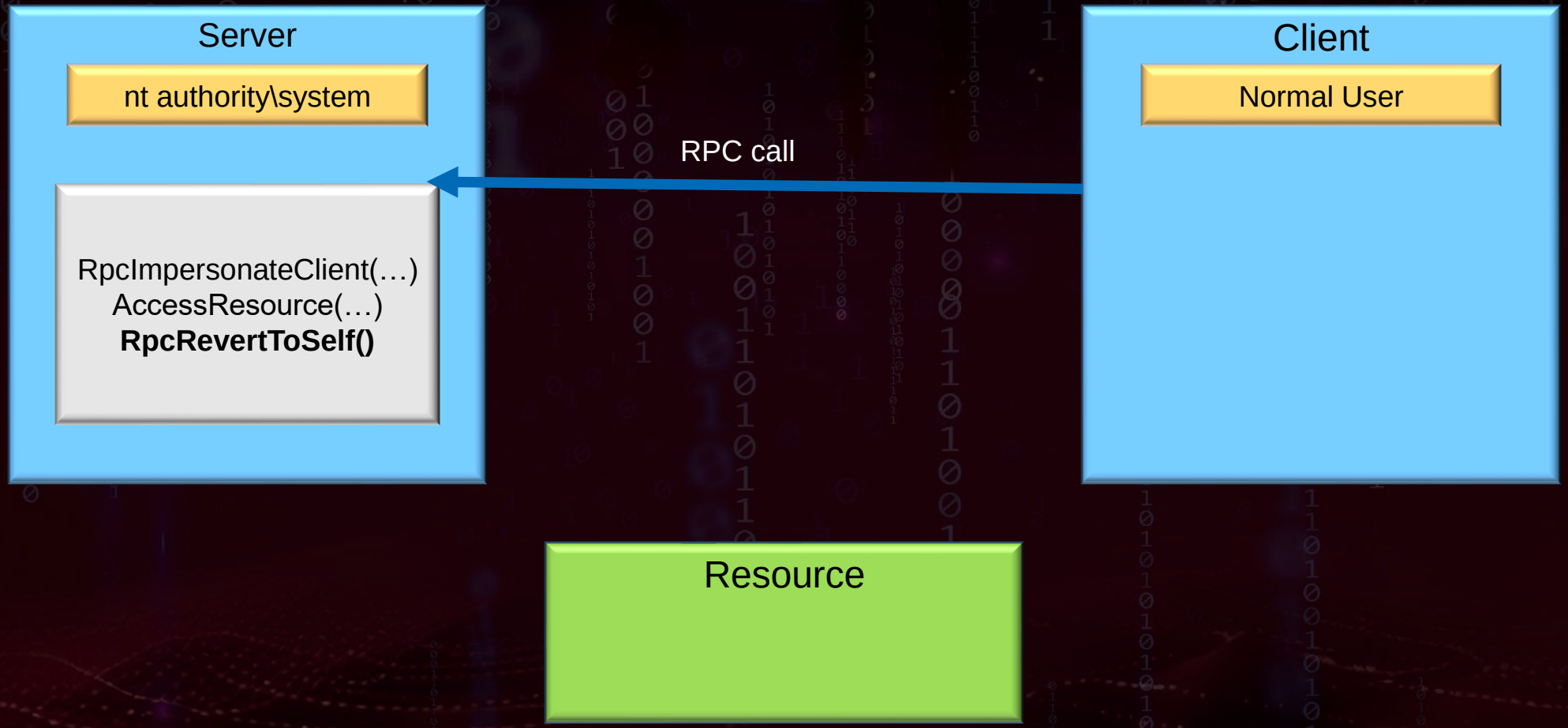
Impersonation 如何工作



Impersonation 如何工作



Impersonation 如何工作



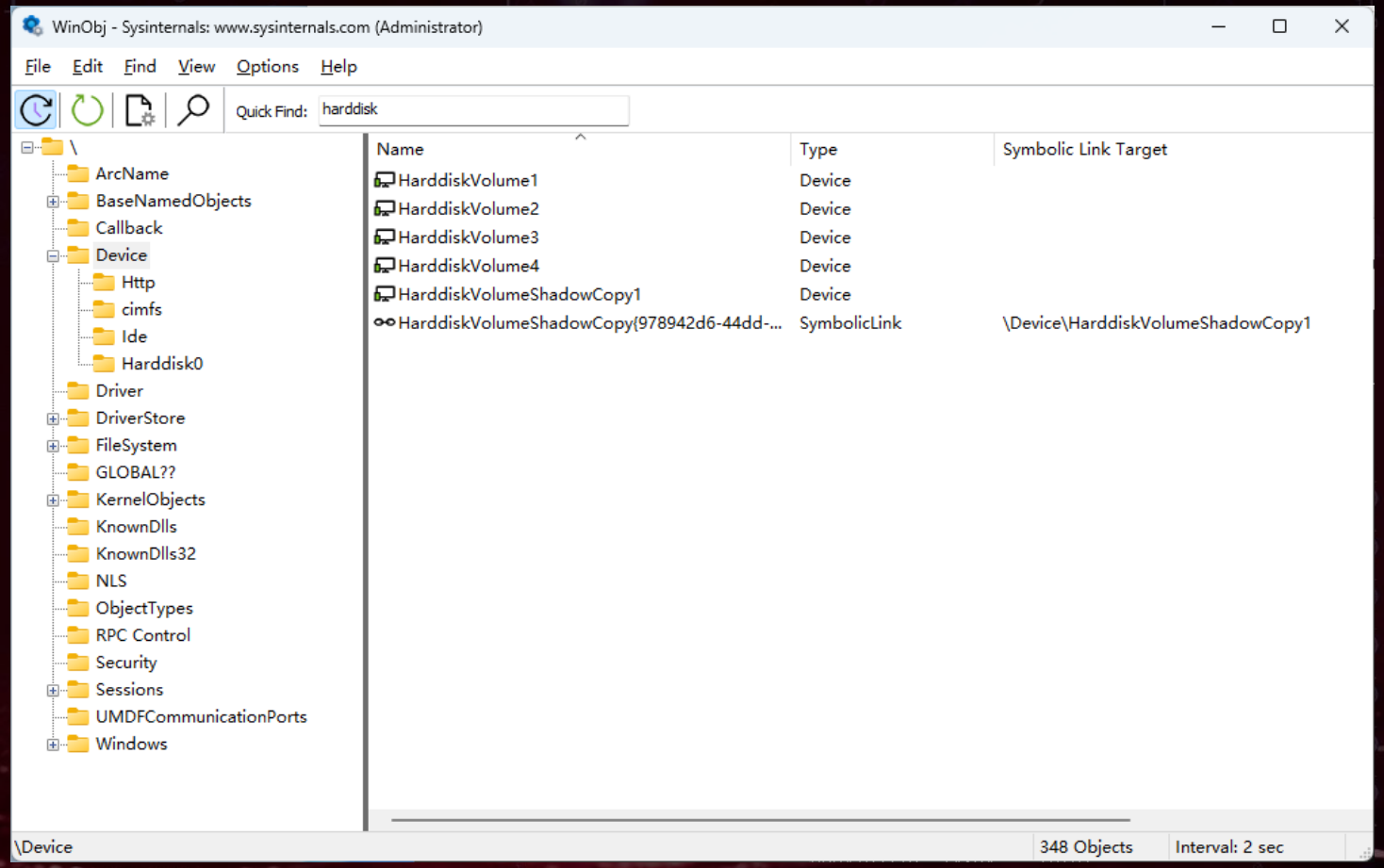
DeviceMap 是什么

我可不是保存图标
的手办哦！(>▿<)

- DOS 操作系统使用盘符来标识磁盘驱动器
- 初代 IBM PC 支持最多两个软盘驱动器
- 硬盘驱动器的盘符从 C: 开始

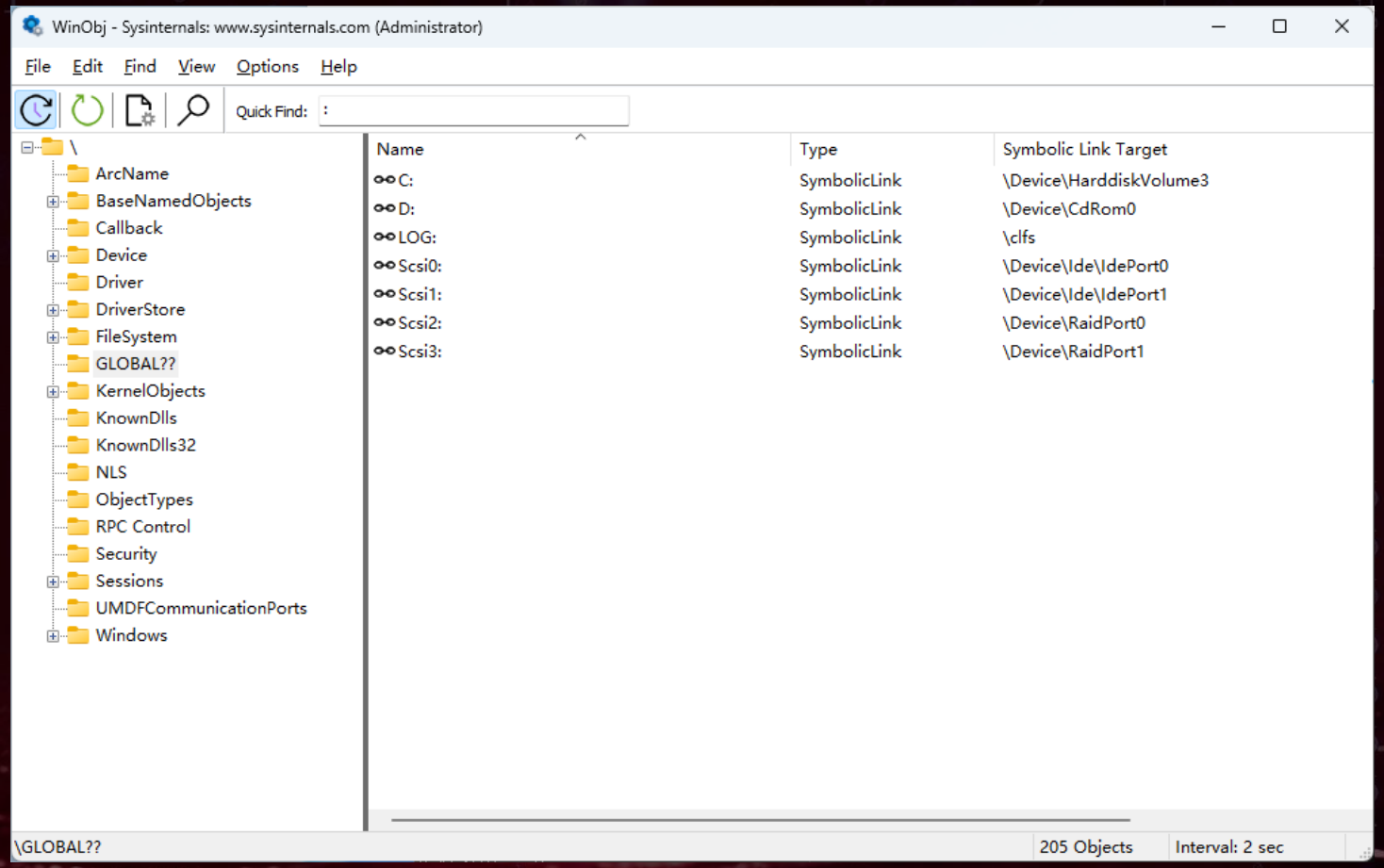
DeviceMap 是什么

- Windows 操作系统用对象管理器来管理磁盘驱动器



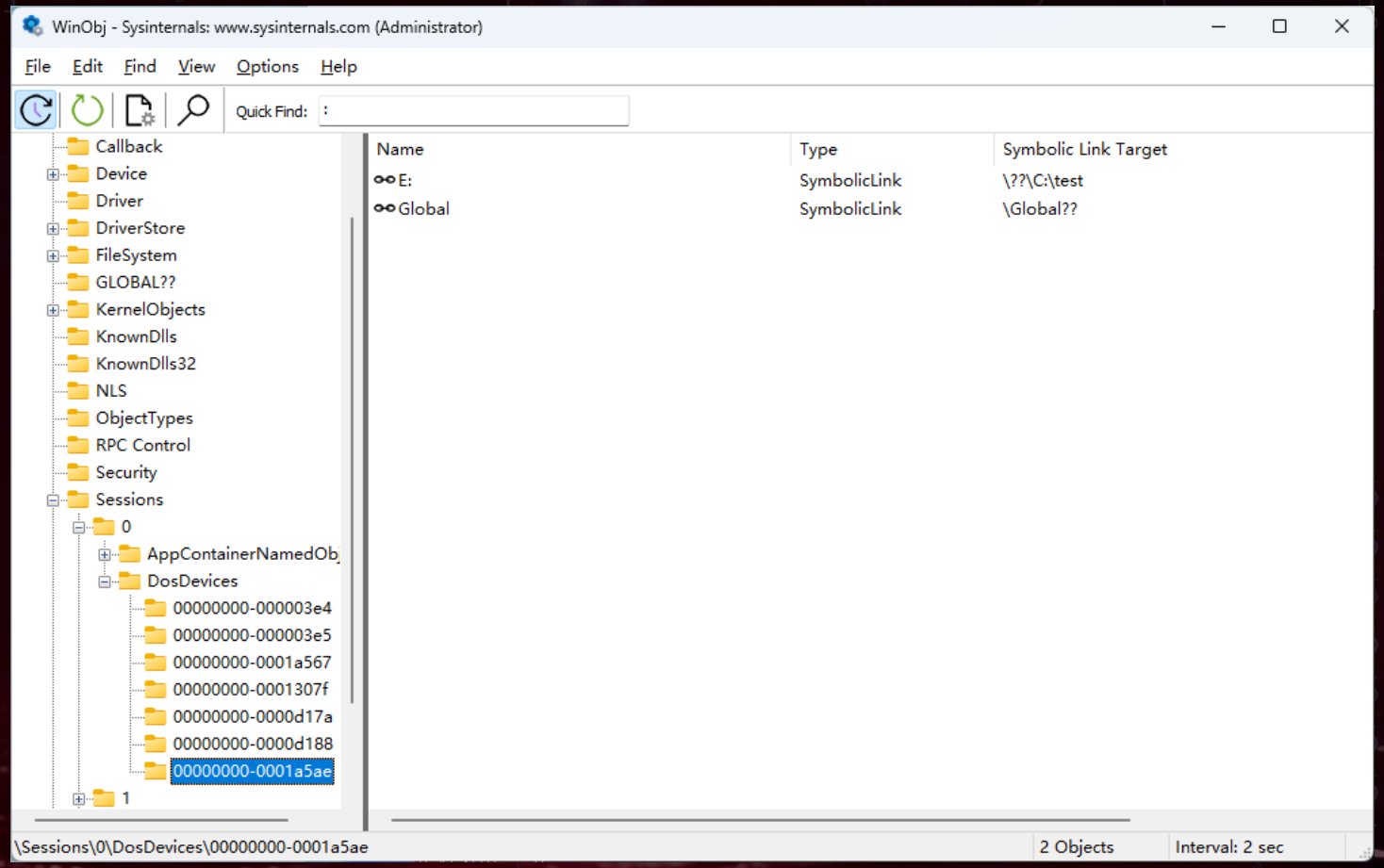
DeviceMap 是什么

- 对 DOS 设备进行映射来保持兼容



DeviceMap 是什么

- 每个登录会话都有自己的本地 DosDevices 上下文



DeviceMap 如何工作

- 将 Win32NtName 转换成 NtPathName

DeviceMap 如何工作

- 优先使用会话本地 DosDevices 上下文

DeviceMap 如何工作

- 未找到则使用全局 DosDevices 上下文

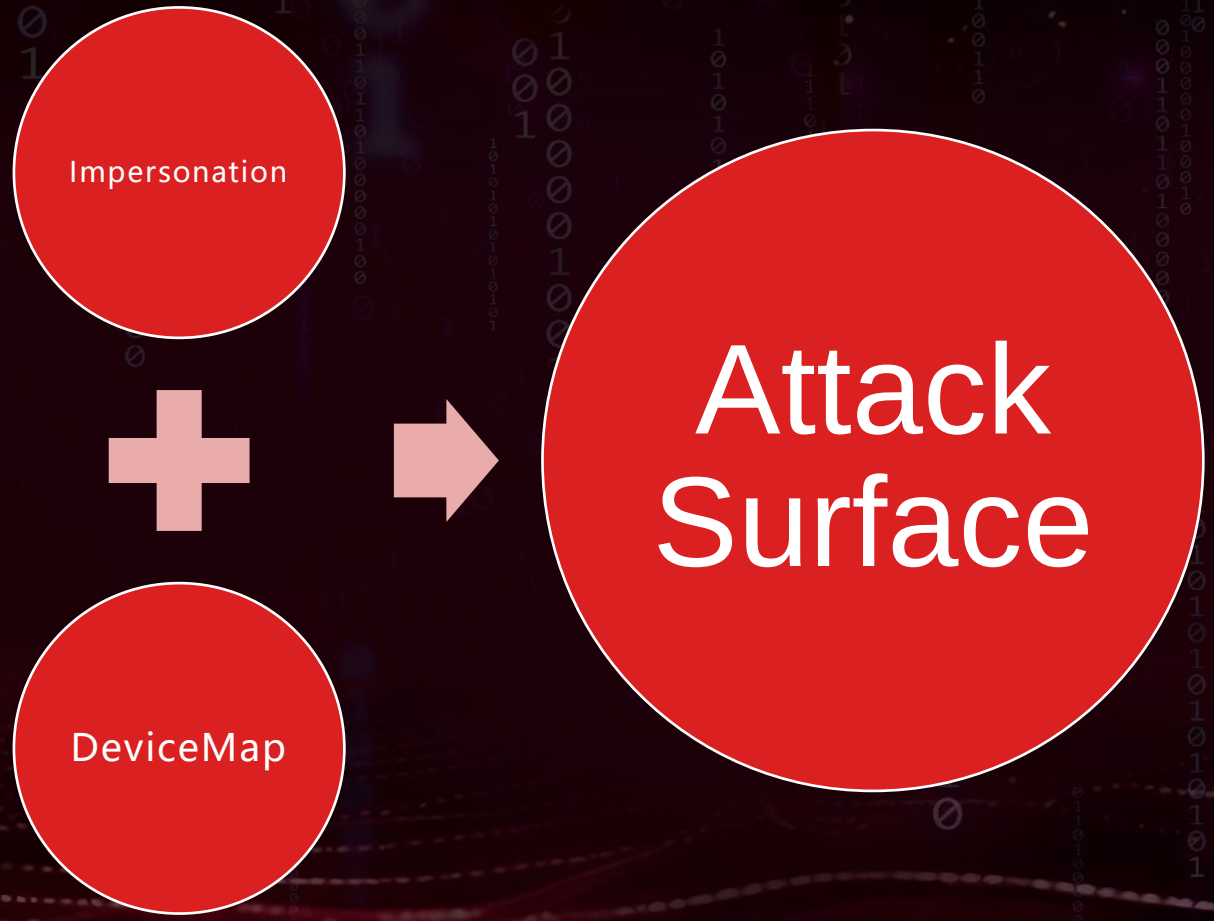
PART ONE

02

攻击面的诞生

The birth of an attack surface

Impersonation 遇上 DeviceMap



Impersonation 遇上 DeviceMap

KCon

Impersonation 使用客户端的 Token

Token 中包含 AuthenticationId

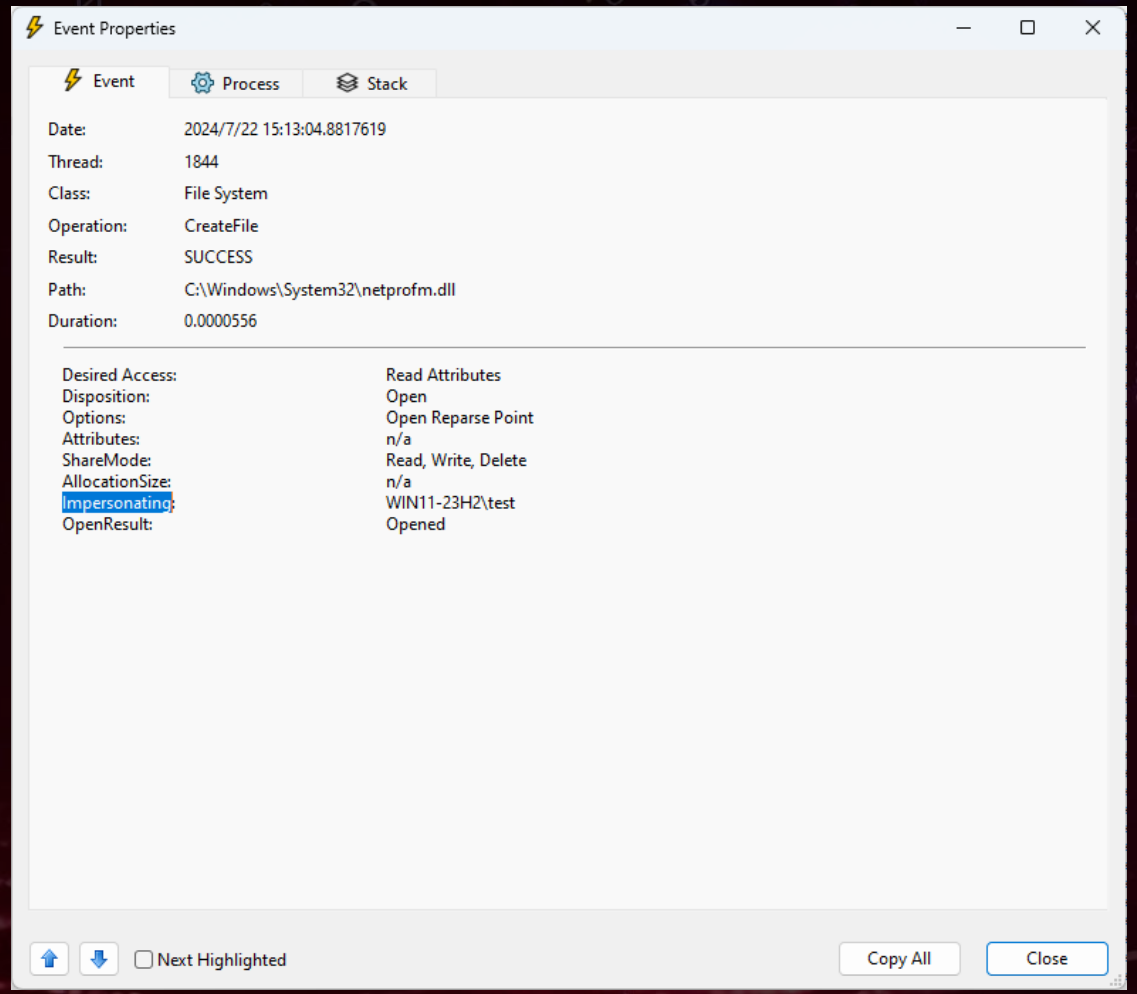
AuthenticationId 决定会话本地 DosDevices 上下文

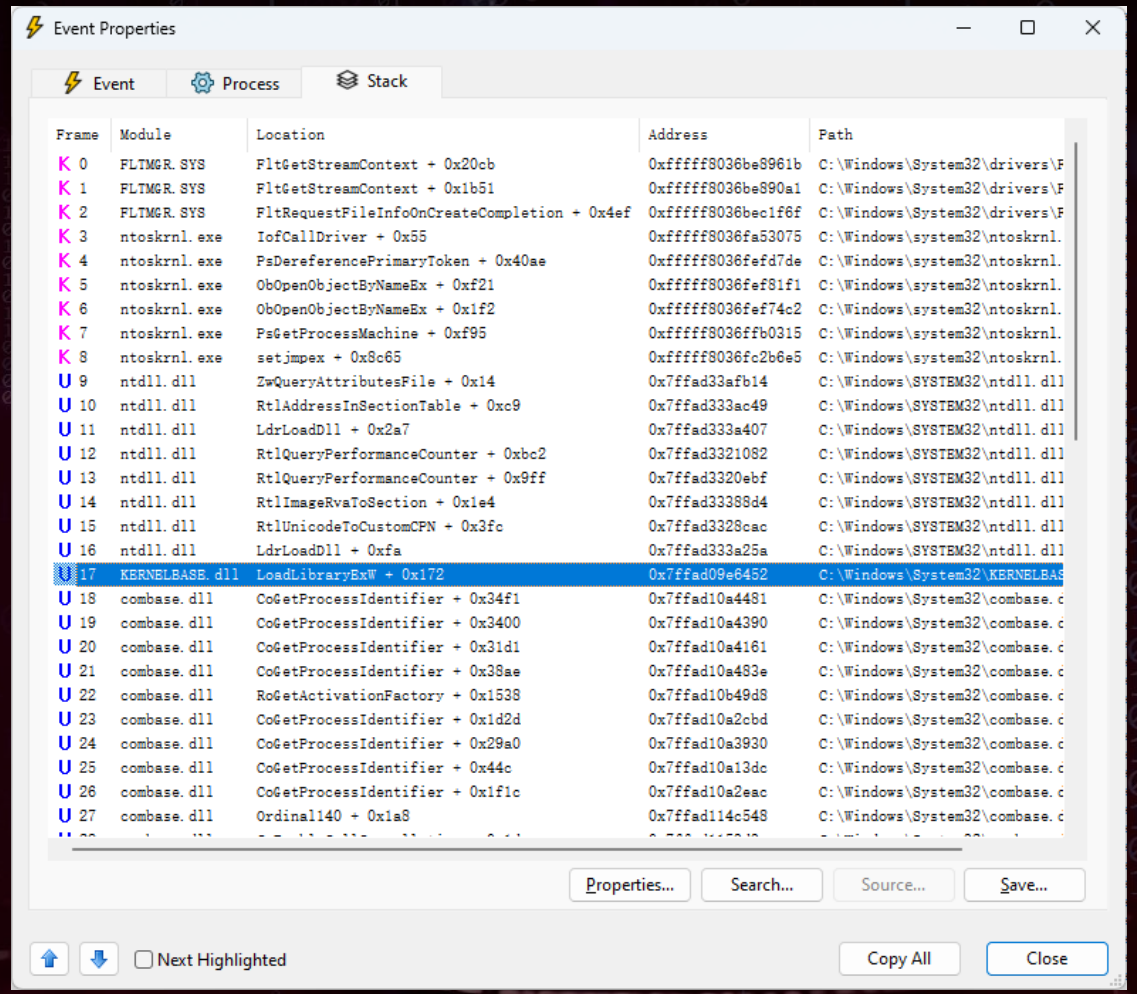
会话本地 DosDevices 上下文决定盘符的映射

用户可以修改自己的会话本地 DosDevices 上下文中的映射

Impersonation 遇上 DeviceMap

攻击面1：劫持DLL


Event Properties
Event
Process
Stack
Date: 2024/7/22 15:13:04.8817619
Thread: 1844
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Windows\System32\netprofm.dll
Duration: 0.0000556
Desired Access: Read Attributes
Disposition: Open
Options: Open Reparse Point
Attributes: n/a
ShareMode: Read, Write, Delete
AllocationSize: n/a
Impersonating: WIN11-23H2\test
OpenResult: Opened
Next Highlighted
Copy All
Close


Event Properties
Event
Process
Stack
Frame Module Location Address Path
K 0 FLTMGR.SYS FltGetStreamContext + 0x20cb 0xfffff8036be8961b C:\Windows\System32\drivers\F
K 1 FLTMGR.SYS FltGetStreamContext + 0x1b51 0xfffff8036be890a1 C:\Windows\System32\drivers\F
K 2 FLTMGR.SYS FltRequestFileInfoOnCreateCompletion + 0x4ef 0xfffff8036bec1f6f C:\Windows\System32\drivers\F
K 3 ntoskrnl.exe IofCallDriver + 0x55 0xfffff8036fa53075 C:\Windows\system32\ntoskrnl.
K 4 ntoskrnl.exe PsDereferencePrimaryToken + 0x40ae 0xfffff8036fefd7de C:\Windows\system32\ntoskrnl.
K 5 ntoskrnl.exe ObOpenObjectByNameEx + 0xf21 0xfffff8036fef81f1 C:\Windows\system32\ntoskrnl.
K 6 ntoskrnl.exe ObOpenObjectByNameEx + 0x1f2 0xfffff8036fef74c2 C:\Windows\system32\ntoskrnl.
K 7 ntoskrnl.exe PsGetProcessMachine + 0xf95 0xfffff8036ffb0315 C:\Windows\system32\ntoskrnl.
K 8 ntoskrnl.exe setjmpex + 0x8c65 0xfffff8036fc2b6e5 C:\Windows\system32\ntoskrnl.
U 9 ntdll.dll ZwQueryAttributesFile + 0x14 0x7ffad33afb14 C:\Windows\SYSTEM32\ntdll.dll
U 10 ntdll.dll RtlAddressInSectionTable + 0xc9 0x7ffad333ac49 C:\Windows\SYSTEM32\ntdll.dll
U 11 ntdll.dll LdrLoadDll + 0x2a7 0x7ffad333a407 C:\Windows\SYSTEM32\ntdll.dll
U 12 ntdll.dll RtlQueryPerformanceCounter + 0xbc2 0x7ffad3321082 C:\Windows\SYSTEM32\ntdll.dll
U 13 ntdll.dll RtlQueryPerformanceCounter + 0x9ff 0x7ffad3320ebf C:\Windows\SYSTEM32\ntdll.dll
U 14 ntdll.dll RtlImageRvaToSection + 0x1e4 0x7ffad33388d4 C:\Windows\SYSTEM32\ntdll.dll
U 15 ntdll.dll RtlUnicodeToCustomCPN + 0x3fc 0x7ffad3328cac C:\Windows\SYSTEM32\ntdll.dll
U 16 ntdll.dll LdrLoadDll + 0xfa 0x7ffad333a25a C:\Windows\SYSTEM32\ntdll.dll
U 17 KERNELBASE.dll LoadLibraryExW + 0x172 0x7ffad09e6452 C:\Windows\System32\KERNELBAS
U 18 combase.dll CoGetProcessIdentifier + 0x34f1 0x7ffad10a4481 C:\Windows\System32\combase.c
U 19 combase.dll CoGetProcessIdentifier + 0x3400 0x7ffad10a4390 C:\Windows\System32\combase.c
U 20 combase.dll CoGetProcessIdentifier + 0x31d1 0x7ffad10a4161 C:\Windows\System32\combase.c
U 21 combase.dll CoGetProcessIdentifier + 0x38ae 0x7ffad10a483e C:\Windows\System32\combase.c
U 22 combase.dll RoGetActivationFactory + 0x1538 0x7ffad10b49d8 C:\Windows\System32\combase.c
U 23 combase.dll CoGetProcessIdentifier + 0x1d2d 0x7ffad10a2cbd C:\Windows\System32\combase.c
U 24 combase.dll CoGetProcessIdentifier + 0x29a0 0x7ffad10a3930 C:\Windows\System32\combase.c
U 25 combase.dll CoGetProcessIdentifier + 0x44c 0x7ffad10a13dc C:\Windows\System32\combase.c
U 26 combase.dll CoGetProcessIdentifier + 0x1f1c 0x7ffad10a2eac C:\Windows\System32\combase.c
U 27 combase.dll Ordinal140 + 0x1a8 0x7ffad114c548 C:\Windows\System32\combase.c
Properties...
Search...
Source...
Save...
Next Highlighted
Copy All
Close

攻击面1：劫持DLL

KCon

☆ Starred by 4 users

Owner: forshaw@google.com

CC: proje...@google.com

Status: Fixed *(Closed)*

Components: ----

Modified: Jul 29, 2015

Finder-forshaw
CVE-2015-1644
Deadline-90
Product-Windows-Kernel
Reported-2015-Jan-27
CCProjectZeroMembers
Severity-High
MSRC-21430
Vendor-Microsoft

Issue 240: Windows: DosDevices Impersonation Elevation of Privilege

Reported by forshaw@google.com on Wed, Jan 28, 2015, 9:57 AM GMT+8 Project Member

Code

1 of 11
Back to list

Windows: DosDevices Impersonation Elevation of Privilege
Platform: Windows 8.1 Update, Windows 7
Class: Elevation of Privilege

Summary:
When an application impersonates another user all file accesses are performed using the current DOS device map under that token. This allows a user to force a system service to load DLLs or start processes at higher privileges leading to EoP.

Description:

Each login session has a DosDevices mapping under \Sessions\0\DosDevices\X-Y where X-Y is the login session ID. This object directory is writeable by the user. When a \??\ path is looked up the kernel first checks the per-login session mapping for a symlink to the drive mapping, if not found it will fallback to looking up in \GLOBAL??. This mapping is also done when impersonating another user, which is typical of system services when performing actions on behalf of another user.

The vulnerability occurs because a user can place symlinks for the system drives in the per-login session device map and the kernel will follow them during impersonation. If for example a system service when impersonating calls LoadLibrary for a system DLL it's possible for the file open to be redirected to an arbitrary location. So for example if the service tries to load c:\windows\system32\some.dll a user can create a dos device mapping for c: to somewhere else and get a DLL loaded into a system service.

I've fully tested this on Windows 8.1 update 32 bit, but basic testing on Windows 7 x64 indicates the vulnerability is also on that platform. It isn't a bug in the implementation of the services, but a kernel issue.

Proof of Concept:

I've provided a PoC which causes the uses the spooler service to load an arbitrary DLL. As the spooler service runs as local system this is a complete EoP. I've only chosen the spooler service because it was a convenient one to do so and I knew it does a lot of work while impersonating the user. The PoC is only designed for 32 bit Windows 8.1 update. It might work on x64 version, but it doesn't by default on Windows 7 possibility due to differences in the printer driver I'm relying on for execution.

1) Extract the PoC to a location on a local harddisk which is writable by a normal user
2) Execute the Poc_DosDeviceSymlink_EoP.exe file

攻击面1：劫持DLL

- 微软仅对DLL加载进行了修复

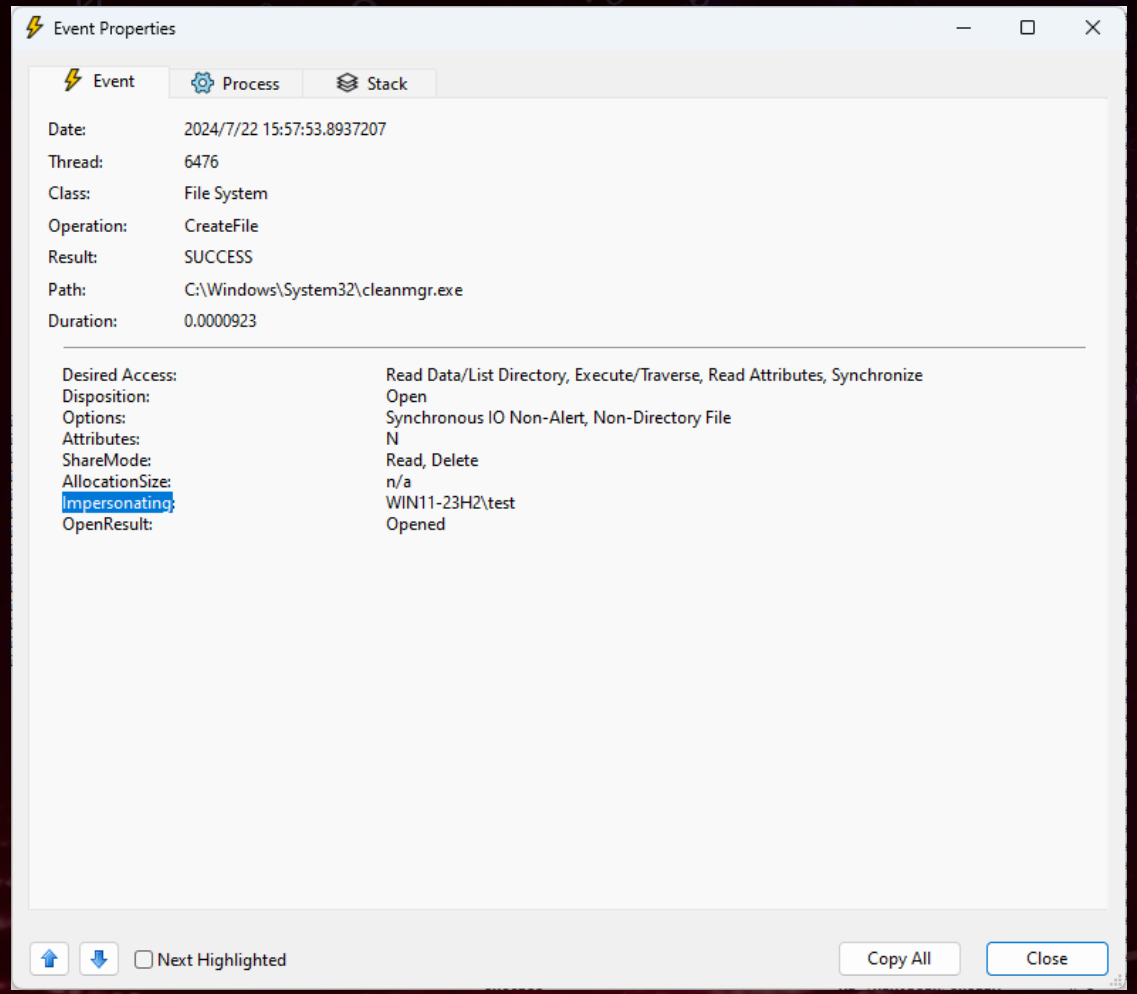
ntdll!LdrpMapDllNtFileName

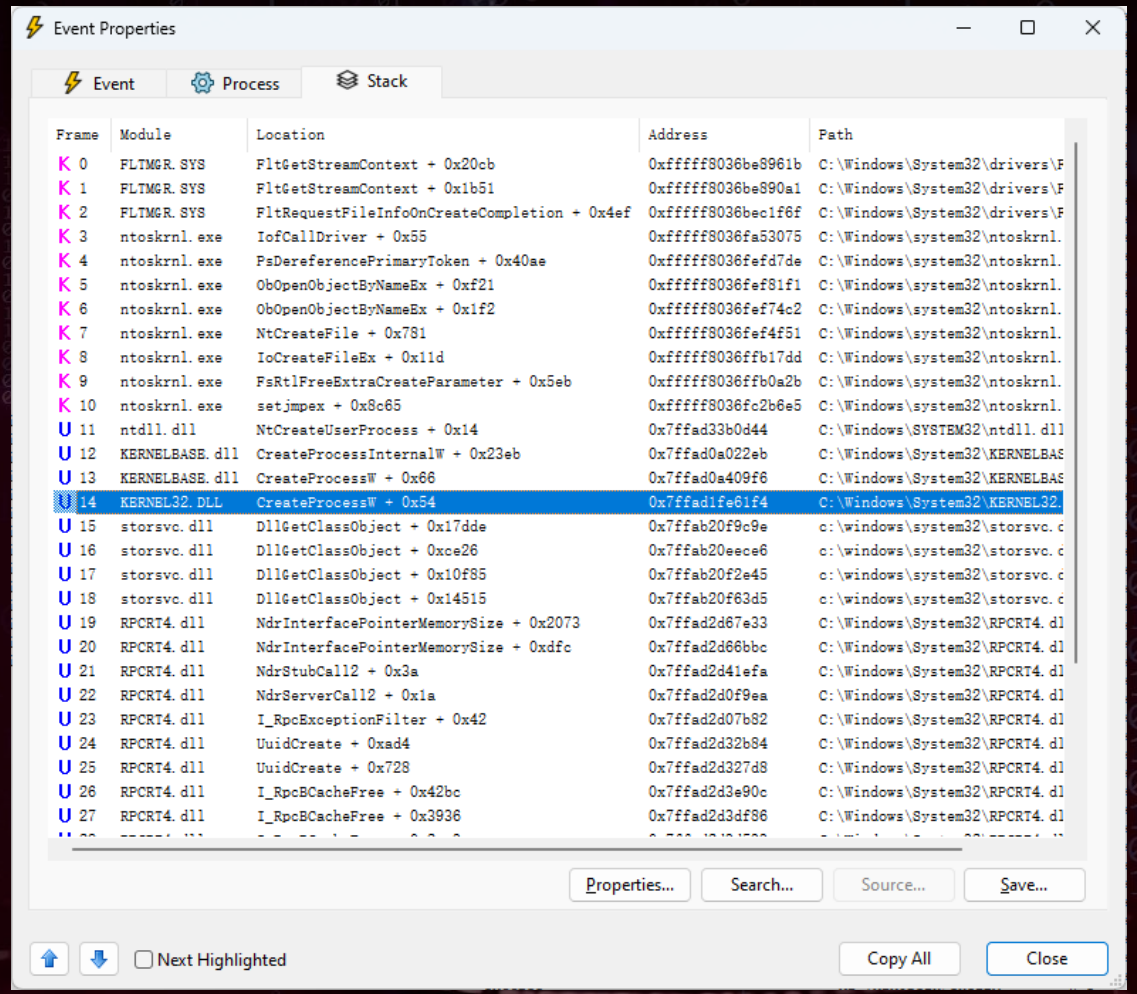
```
ObjectAttributes.Length = 0x30;
if ( !LdrpUseImpersonatedDeviceMap )
  Attributes = 0x840;
ObjectAttributes.RootDirectory = 0i64;
ObjectAttributes.Attributes = Attributes;
ObjectAttributes.ObjectName = lpFileName;
ObjectAttributes.SecurityDescriptor = 0i64;
ObjectAttributes.SecurityQualityOfService = 0i64;
```

```
NtOpenFile(&FileHandle, 0x100021u, &ObjectAttributes, &IoStatusBlock, 5u, 0x60u);
```

OBJ_IGNORE_IMPERSONATED_DEVICEMAP	A device map is a mapping between DOS device names and devices in the system, and is used when resolving DOS names. Separate device maps exists for each user in the system, and users can manage their own device maps. Typically during impersonation, the impersonated user's device map would be used. However, when this flag is set, the process user's device map is used instead.

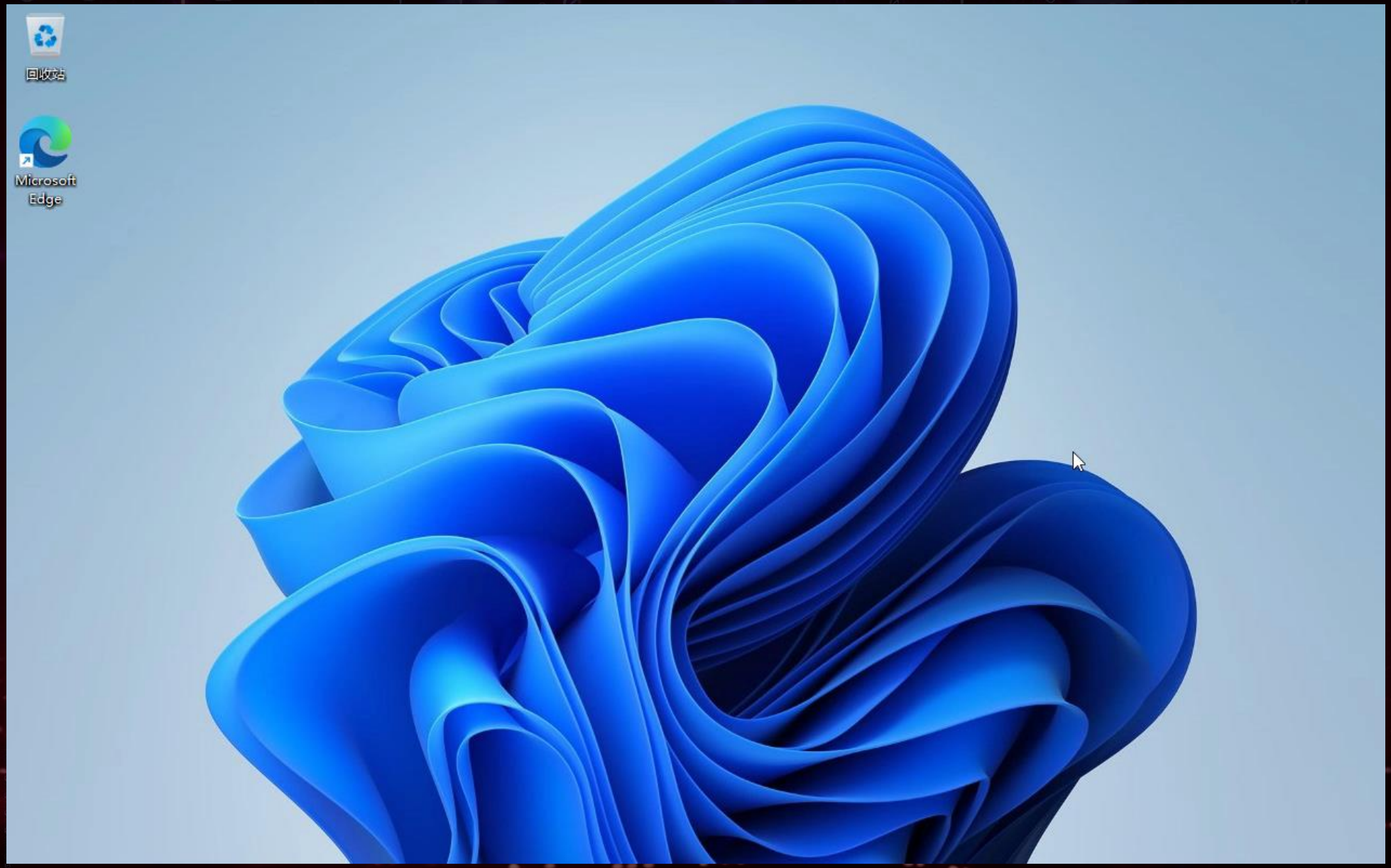
攻击面2：劫持EXE


Event Properties
Event
Process
Stack
Date: 2024/7/22 15:57:53.8937207
Thread: 6476
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Windows\System32\cleanmgr.exe
Duration: 0.0000923
Desired Access: Read Data/List Directory, Execute/Traverse, Read Attributes, Synchronize
Disposition: Open
Options: Synchronous IO Non-Alert, Non-Directory File
Attributes: N
ShareMode: Read, Delete
AllocationSize: n/a
Impersonating: WIN11-23H2\test
OpenResult: Opened
Next Highlighted
Copy All
Close


Event Properties
Event
Process
Stack
Frame Module Location Address Path
K 0 FLTMGR.SYS FltGetStreamContext + 0x20cb 0xfffff8036be8961b C:\Windows\System32\drivers\F
K 1 FLTMGR.SYS FltGetStreamContext + 0x1b51 0xfffff8036be890a1 C:\Windows\System32\drivers\F
K 2 FLTMGR.SYS FltRequestFileInfoOnCreateCompletion + 0x4ef 0xfffff8036bec1f6f C:\Windows\System32\drivers\F
K 3 ntoskrnl.exe IofCallDriver + 0x55 0xfffff8036fa53075 C:\Windows\system32\ntoskrnl.
K 4 ntoskrnl.exe PsDereferencePrimaryToken + 0x40ae 0xfffff8036fefd7de C:\Windows\system32\ntoskrnl.
K 5 ntoskrnl.exe ObOpenObjectByNameEx + 0xf21 0xfffff8036fef81f1 C:\Windows\system32\ntoskrnl.
K 6 ntoskrnl.exe ObOpenObjectByNameEx + 0x1f2 0xfffff8036fef74c2 C:\Windows\system32\ntoskrnl.
K 7 ntoskrnl.exe NtCreateFile + 0x781 0xfffff8036fef4f51 C:\Windows\system32\ntoskrnl.
K 8 ntoskrnl.exe IoCreateFileEx + 0x11d 0xfffff8036ffb17dd C:\Windows\system32\ntoskrnl.
K 9 ntoskrnl.exe FsRtlFreeExtraCreateParameter + 0x5eb 0xfffff8036ffb0a2b C:\Windows\system32\ntoskrnl.
K 10 ntoskrnl.exe setjmpex + 0x8c65 0xfffff8036fc2b6e5 C:\Windows\system32\ntoskrnl.
U 11 ntdll.dll NtCreateUserProcess + 0x14 0x7ffad33b0d44 C:\Windows\SYSTEM32\ntdll.dll
U 12 KERNELBASE.dll CreateProcessInternalW + 0x23eb 0x7ffad0a022eb C:\Windows\System32\KERNELBAS
U 13 KERNELBASE.dll CreateProcessW + 0x66 0x7ffad0a409f6 C:\Windows\System32\KERNELBAS
U 14 KERNEL32.DLL CreateProcessW + 0x54 0x7ffad1fe61f4 C:\Windows\System32\KERNEL32.
U 15 storsvc.dll DllGetClassObject + 0x17dde 0x7ffab20f9c9e c:\windows\system32\storsvc.c
U 16 storsvc.dll DllGetClassObject + 0xce26 0x7ffab20eece6 c:\windows\system32\storsvc.c
U 17 storsvc.dll DllGetClassObject + 0x10f85 0x7ffab20f2e45 c:\windows\system32\storsvc.c
U 18 storsvc.dll DllGetClassObject + 0x14515 0x7ffab20f63d5 c:\windows\system32\storsvc.c
U 19 RPCRT4.dll NdrInterfacePointerMemorySize + 0x2073 0x7ffad2d67e33 C:\Windows\System32\RPCRT4.dl
U 20 RPCRT4.dll NdrInterfacePointerMemorySize + 0xdfc 0x7ffad2d66bbc C:\Windows\System32\RPCRT4.dl
U 21 RPCRT4.dll NdrStubCall2 + 0x3a 0x7ffad2d41efa C:\Windows\System32\RPCRT4.dl
U 22 RPCRT4.dll NdrServerCall2 + 0x1a 0x7ffad2d0f9ea C:\Windows\System32\RPCRT4.dl
U 23 RPCRT4.dll I_RpcExceptionFilter + 0x42 0x7ffad2d07b82 C:\Windows\System32\RPCRT4.dl
U 24 RPCRT4.dll UuidCreate + 0xad4 0x7ffad2d32b84 C:\Windows\System32\RPCRT4.dl
U 25 RPCRT4.dll UuidCreate + 0x728 0x7ffad2d327d8 C:\Windows\System32\RPCRT4.dl
U 26 RPCRT4.dll I_RpcBCacheFree + 0x42bc 0x7ffad2d3e90c C:\Windows\System32\RPCRT4.dl
U 27 RPCRT4.dll I_RpcBCacheFree + 0x3936 0x7ffad2d3df86 C:\Windows\System32\RPCRT4.dl
Properties...
Search...
Source...
Save...
Next Highlighted
Copy All
Close

攻击面2：劫持EXE

KCon



攻击面2：劫持EXE

```
NTSTATUS __fastcall PspReferenceTokenForNewProcess(
        _EPROCESS *ParentProcess,
        HANDLE TokenHandle,
        KPROCESSOR_MODE AccessMode,
        _TOKEN **ppToken)
{
  NTSTATUS Status; // eax
  _TOKEN *Token; // rbx MAPDST

  if ( TokenHandle )
  {
    Token = 0i64;
    Status = ObReferenceObjectByHandle(TokenHandle, 1u, SeTokenObjectType, AccessMode, &Token, 0i64);
    if ( Status < 0 )
      return Status;
  }
  else if ( ParentProcess )
  {
    Token = PsReferencePrimaryTokenWithTag(ParentProcess, 'tlfD');
  }
  else
  {
    Token = PspBootAccessToken;
    ObfReferenceObject(PspBootAccessToken);
  }
  *ppToken = Token;
  return 0;
}
```

攻击面2：劫持EXE

Windows 错误报告服务特权漏洞提升

CVE-2023-36874
Security Vulnerability

发行版： 2023年7月11日

Assigning CNA: Microsoft

CVE-2023-36874

影响: 特权提升　最高严重性: 重要

Weakness: CWE-59: Improper Link Resolution Before File Access ('Link Following')

CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

攻击面3：劫持SXS

- 创建 Activation Context 时未设置 OBJ_IGNORE_IMPERSONATED_DEVICEMAP

kernel32!BasepCreateActCtx

```
ObjectAttributes.RootDirectory = 0i64;
ObjectAttributes.Length = 0x30;
ObjectAttributes.ObjectName = &ManifestPath;
ObjectAttributes.Attributes = 0x40;
ObjectAttributes.SecurityDescriptor = 0i64;
ObjectAttributes.SecurityQualityOfService = 0i64;
Status = NtOpenFile(FileHandle, 0x1200A9u, &ObjectAttributes, &IoStatusBlock, 5u, 0x60u);
```

攻击面3：劫持SXS

Windows 打印后台处理程序特权提升漏洞

CVE-2022-29104
Security Vulnerability

发行版： 2022年5月10日

最后更新：2022年6月3日

Assigning CNA: Microsoft

CVE-2022-29104

影响: 特权提升　最高严重性: 重要

CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

加载 DLL 之前调用 RpcRevertToSelf

攻击面3：劫持SXS

Windows 客户端服务器运行时子系统(CSRSS)权限提升漏洞

CVE-2022-22047
Security Vulnerability

发行版： 2022年7月12日

Assigning CNA: Microsoft

CVE-2022-22047

影响: 特权提升　最高严重性: 重要

CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

禁用 Manifest 中的 loadFrom 属性

攻击面3：劫持SXS

Windows 客户端服务器运行时子系统(CSRSS)权限提升漏洞

CVE-2022-37989
Security Vulnerability

发行版： 2022年10月11日

Assigning CNA: Microsoft

CVE-2022-37989

影响: 特权提升　最高严重性: 重要

CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

修复 Manifest 中 name 属性的目录穿越

攻击面3：劫持SXS

Windows 打印后台处理程序特权提升漏洞

CVE-2022-41073
Security Vulnerability

发行版： 2022年11月8日

Assigning CNA: Microsoft

CVE-2022-41073

影响: 特权提升　最高严重性: 重要

CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

加载 DLL 之前调用 RpcRevertToSelf

攻击面3：劫持SXS

Windows 内核特权提升漏洞

CVE-2023-35359
Security Vulnerability

发行版： 2023年8月8日

最后更新：2023年8月24日

Assigning CNA: Microsoft

CVE-2023-35359

影响: 特权提升　最高严重性: 重要

Weakness: CWE-23: Relative Path Traversal

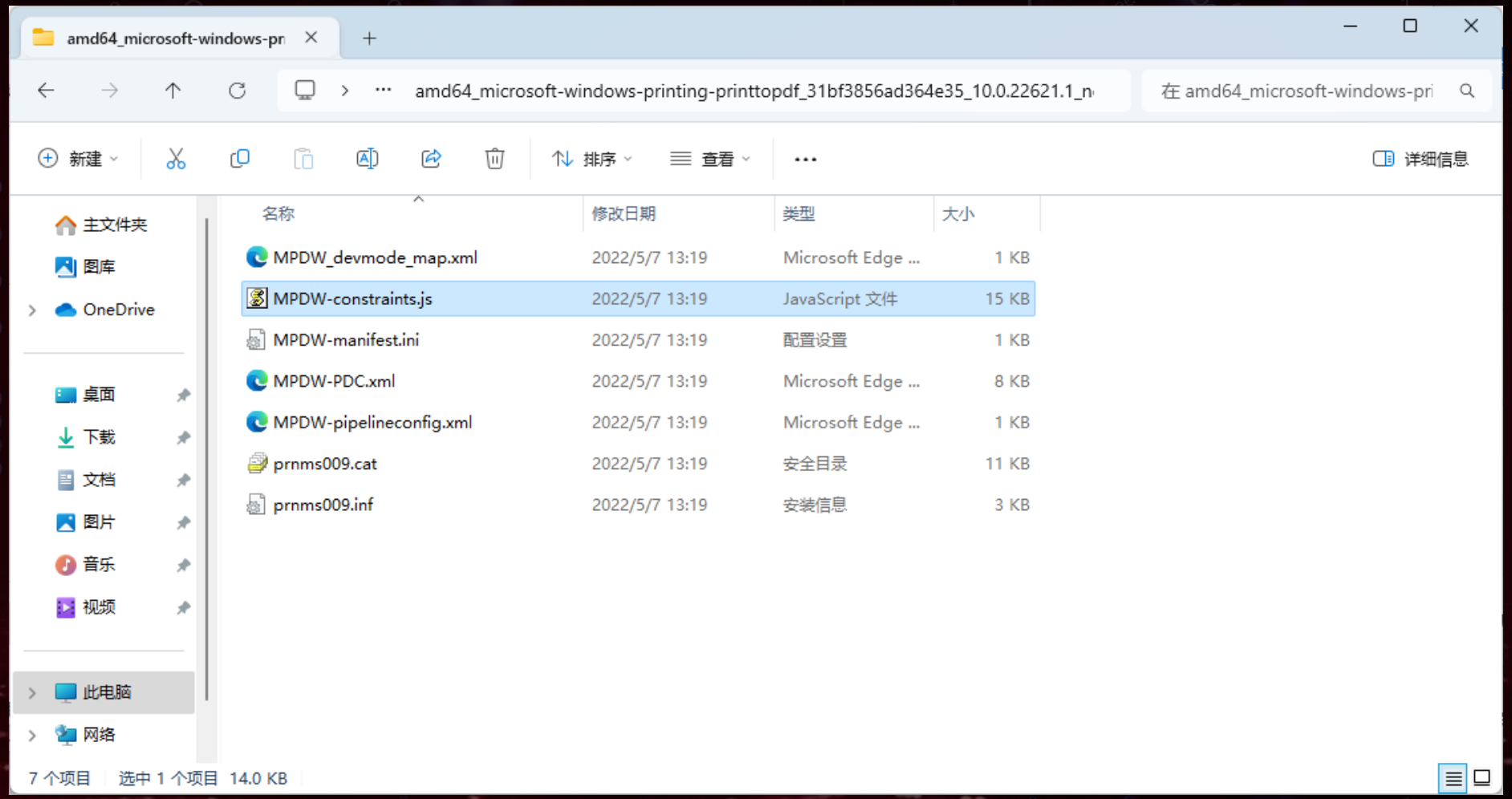
CVSS Source: Microsoft

CVSS:3.1 7.8 / 6.8

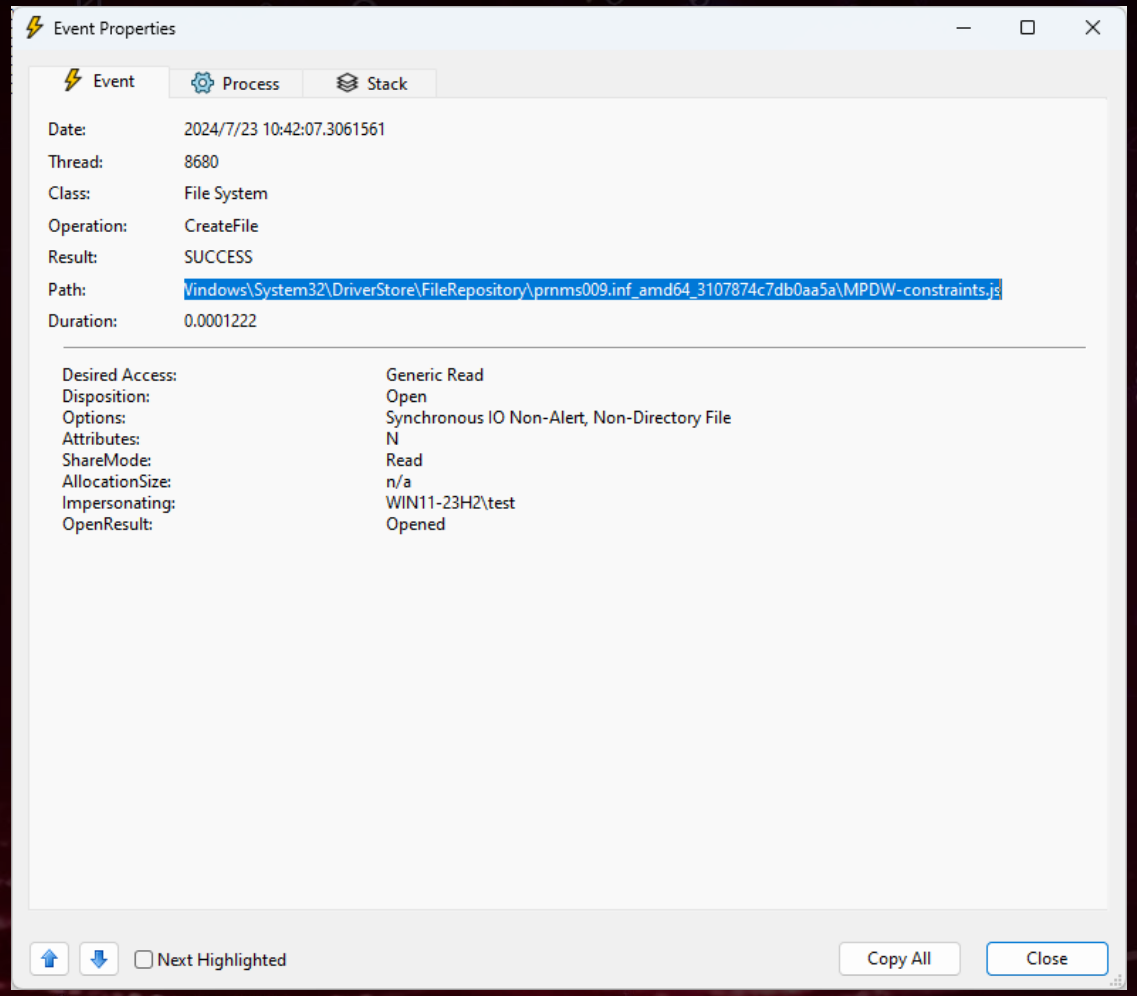
关闭攻击面

攻击面4：劫持脚本

- Windows V4 打印机驱动中包含 JavaScript 文件



攻击面4：劫持脚本


Event Properties
Event
Process
Stack
Date: 2024/7/23 10:42:07.3061561
Thread: 8680
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: Windows\System32\DriverStore\FileRepository\prnms009.inf_amd64_3107874c7db0aa5a\MPDW-constraints.js
Duration: 0.0001222
Desired Access: Generic Read
Disposition: Open
Options: Synchronous IO Non-Alert, Non-Directory File
Attributes: N
ShareMode: Read
AllocationSize: n/a
Impersonating: WIN11-23H2\test
OpenResult: Opened
Next Highlighted
Copy All
Close

攻击面5：劫持TypeLib

攻击面5：劫持TypeLib

- LoadTypeLib 的特殊语法

The function **LoadTypeLib** loads a type library (usually created with MkTypLib) that is stored in the specified file. If *szFile* specifies only a file name without any path, **LoadTypeLib** searches for the file and proceeds as follows:

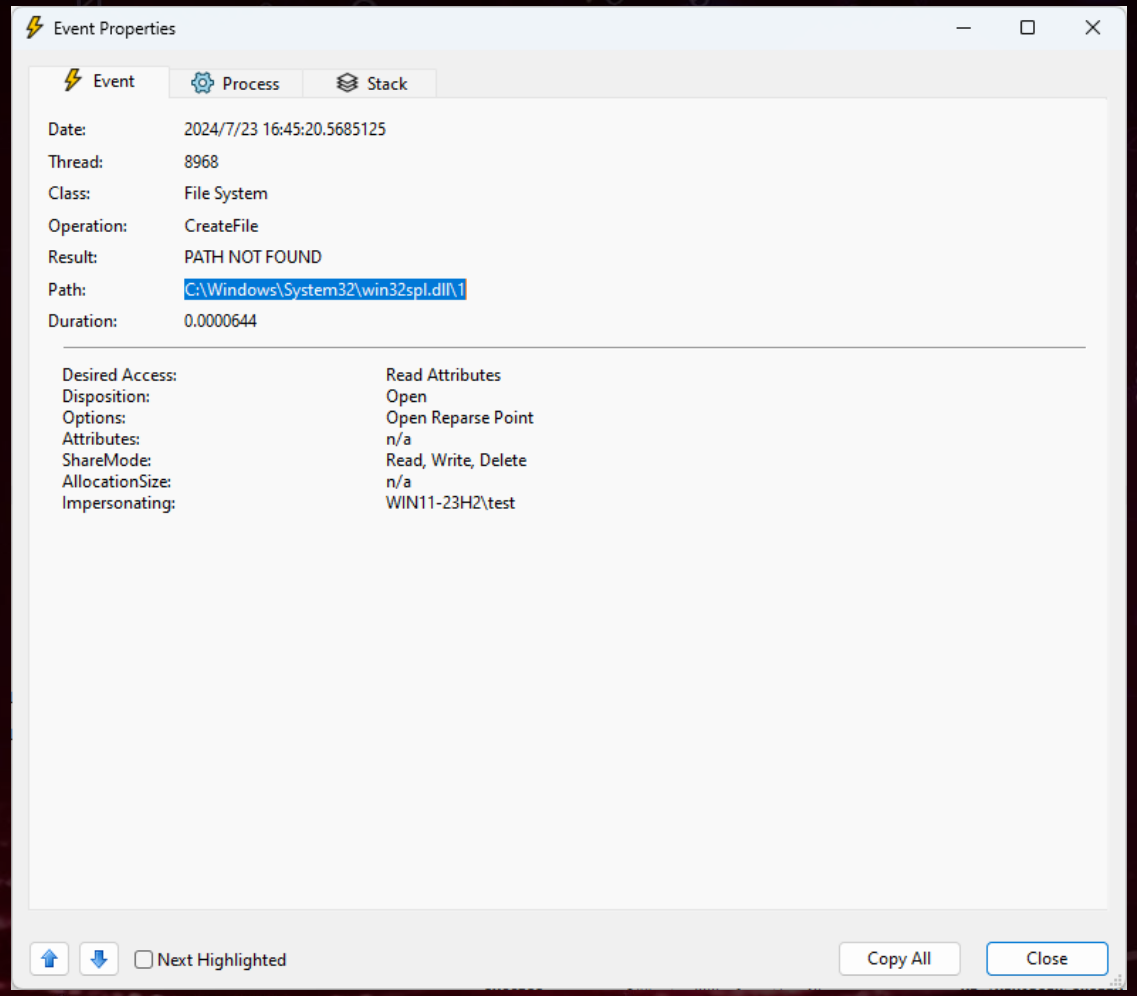
- If the file is a stand-alone type library implemented by Typelib.dll, the library is loaded directly.
- If the file is a DLL or an executable file, it is loaded. By default, the type library is extracted from the first resource of type ITypeLib. To load a different type of library resource, append an integer index to *szFile*. For example:

```
C++                                                    Copy

ITypeLib *ptlib;
LoadTypeLib("C:\\MONTANA\\EXE\\MFA.EXE\\3", &ptlib)
```

  This statement loads the type library resource 3 from the file Mfa.exe file.
- If the file is none of the above, the file name is parsed into a moniker (an object that represents a file-based link source), and then bound to the moniker. This approach allows **LoadTypeLib** to be used on foreign type libraries, including in-memory type libraries. Foreign type libraries cannot reside in a DLL or an executable file. For more information on monikers, see the COM Programmer's Reference.

攻击面5：劫持TypeLib



PART ONE

03

攻击面的消亡

The death of the attack surface

问题修复

- 微软在2023年8月对此问题进行了修复

```
BOOLEAN __fastcall ObpUseSystemDeviceMap(UNICODE_STRING *ObjectName)
{
  WCHAR *SystemRoot; // rax
  WCHAR SystemRootDriver; // di
  WCHAR ObjectNameDriver; // si
  WCHAR *Buffer; // rcx
  BOOLEAN bRet; // al

  bRet = 0;
  if ( (*(&KeGetCurrentThread()[1].SwapListEntry + 1) & 8) != 0 && ObjectName->Length >= 0xEu )
  {
    SystemRoot = RtlGetNtSystemRoot();
    SystemRootDriver = RtlUpcaseUnicodeChar(*SystemRoot);
    ObjectNameDriver = RtlUpcaseUnicodeChar(ObjectName->Buffer[4]);
    if ( Feature_3566457150__private_IsEnabled() )
    {
      Buffer = ObjectName->Buffer;
      if ( Buffer[5] == ':' && Buffer[6] == '\\' && SystemRootDriver == ObjectNameDriver )
        return 1;
    }
  }
  return bRet;
}
```

问题修复

- 微软在2023年8月对此问题进行了修复

nt!ObpLookupObjectName

```
if ( ObjectType == IoFileObjectType && ObpUseSystemDeviceMap(ObjectName) )
  Attribute |= 0x800u;
```

KCon

PART ONE

04

总结与展望

Summary

总结

- 系统特性的组合可能带来意想不到的副作用
- 基础特性的副作用可能导致巨大的攻击面
- 修复漏洞时不能仅仅修复漏洞本身的 Root Cause
- 还需要全面分析是否涉及新的攻击面
- 及时修复攻击面的 Root Cause 以关闭攻击面

TONGDAO

KCon 2024

THANKS

汇报人：张云海　　时间：2024.08.24