Exploiting Errors in Windows Error Reporting in 2022

XueFeng Li of Sangfor
Dr. Zhiniang Peng of Sangfor

C:\> whoarewe

- **Xuefeng Li**

  Xuefeng Li (@lxf02942370) is a security researcher at Sangfor. He have forced on Windows vulnerability hunting and exploitation for almost. ranked #10, #22, #23 on the MSRC Most Valuable Security Researcher list in 2020, 2021 and 2022.

- **Zhiniang Peng**

  Dr. Zhiniang Peng (@edwardzpeng) is the Principal Security Researcher at Sangfor. His current research areas include applied cryptography, software security and threat hunting. He has more than 10 years of experience in both offensive and defensive security and published many research in both academia and industry.

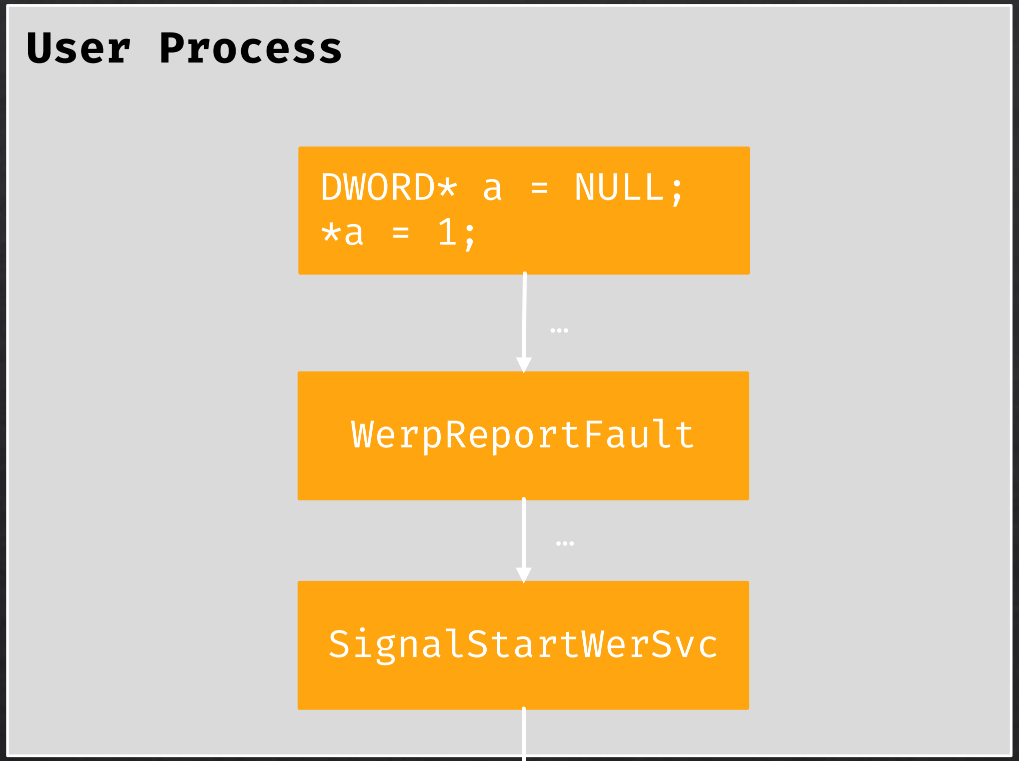
Agenda

- Basic of Windows Error Reporting
- Vulnerabilities history of Windows Error Reporting
- Incorrect Handle Duplicate lead to EOP - CVE-2022-35795
- Conclusion

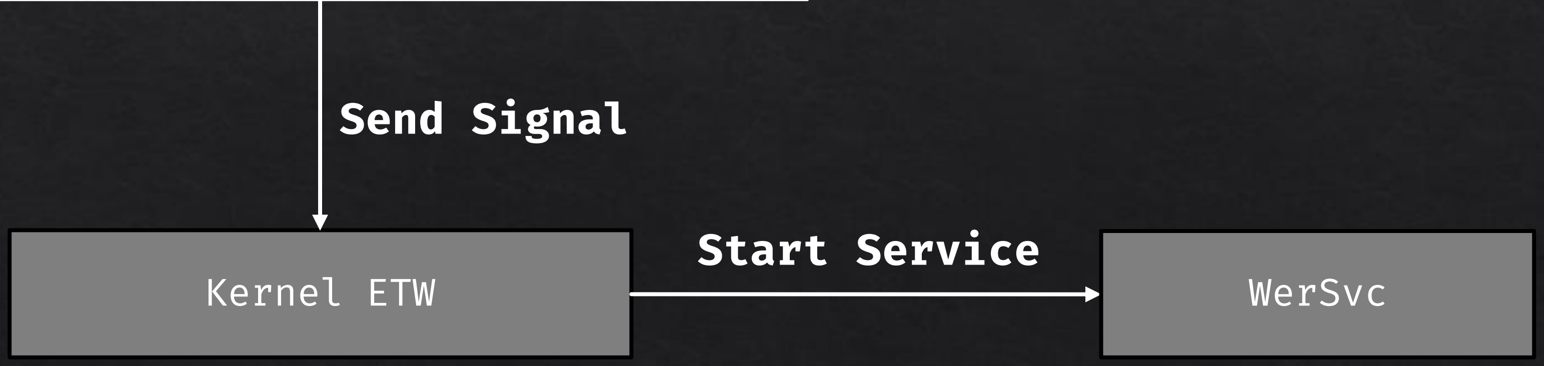
Part1

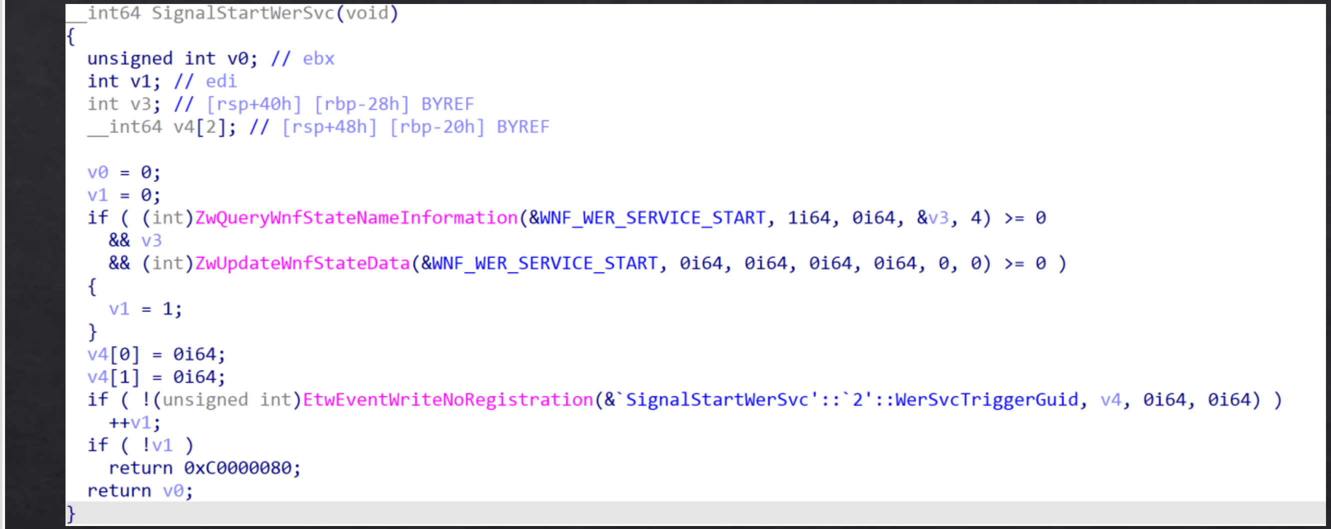
Basic of Windows Error Reporting

Start WerSvc By ETW



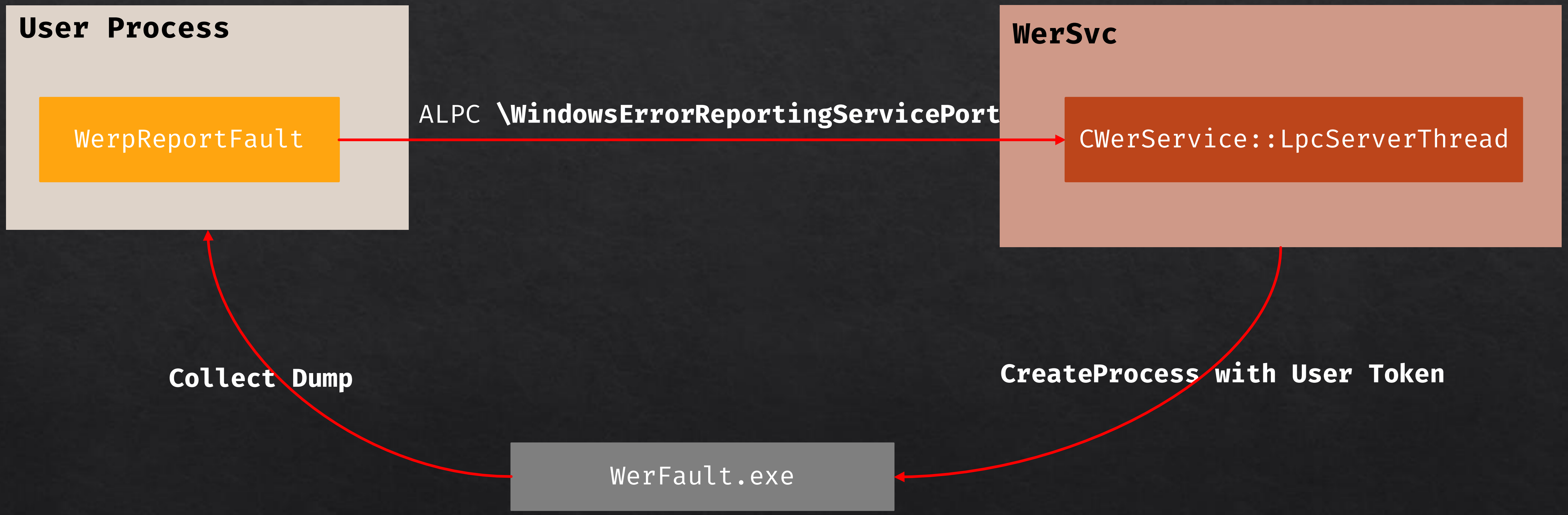
```
DWORD* a = NULL;
*a = 1;
```





```
__int64 SignalStartWerSvc(void)
{
  unsigned int v0; // ebx
  int v1; // edi
  int v3; // [rsp+40h] [rbp-28h] BYREF
  __int64 v4[2]; // [rsp+48h] [rbp-20h] BYREF

  v0 = 0;
  v1 = 0;
  if ( (int)ZwQueryWnfStateNameInformation(&WNF_WER_SERVICE_START, 1i64, 0i64, &v3, 4) >= 0
    && v3
    && (int)ZwUpdateWnfStateData(&WNF_WER_SERVICE_START, 0i64, 0i64, 0i64, 0i64, 0, 0) >= 0 )
  {
    v1 = 1;
  }
  v4[0] = 0i64;
  v4[1] = 0i64;
  if ( !(unsigned int)EtwEventWriteNoRegistration(&`SignalStartWerSvc'::`2'::WerSvcTriggerGuid, v4, 0i64, 0i64) )
    ++v1;
  if ( !v1 )
    return 0xC0000080;
  return v0;
}
```

Collect User Process Memory Dump



User Process

WerpReportFault

ALPC **\WindowsErrorReportingServicePort**

WerSvc

CWerService::LpcServerThread

CreateProcess with User Token

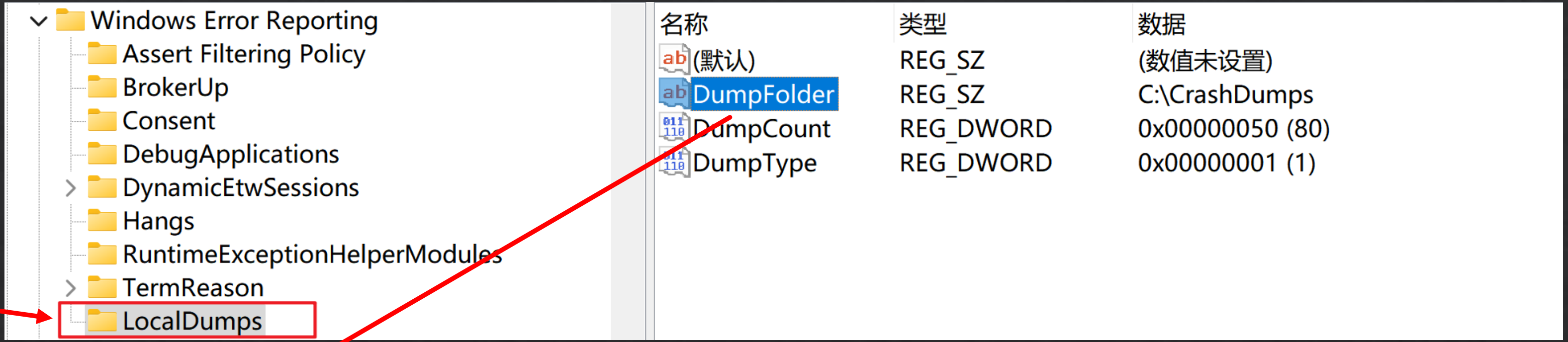
WerFault.exe

Collect Dump

Collecting User-Mode Dumps

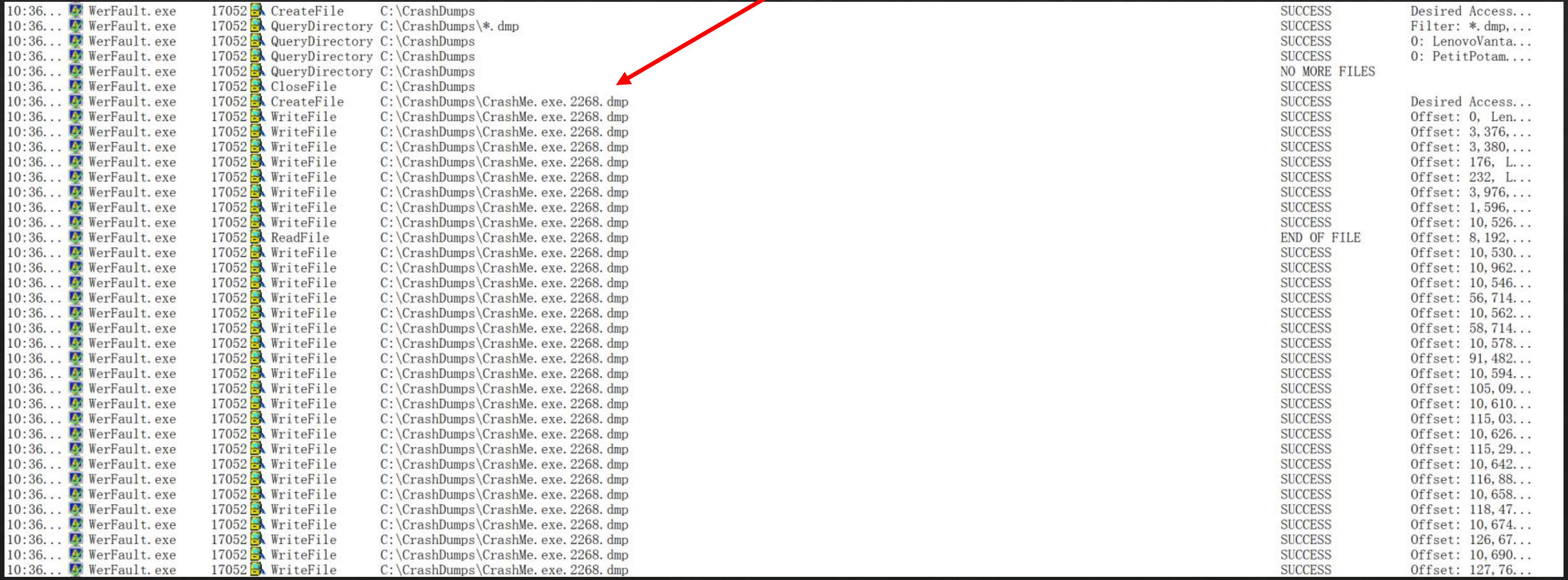
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps

WerFault.exe



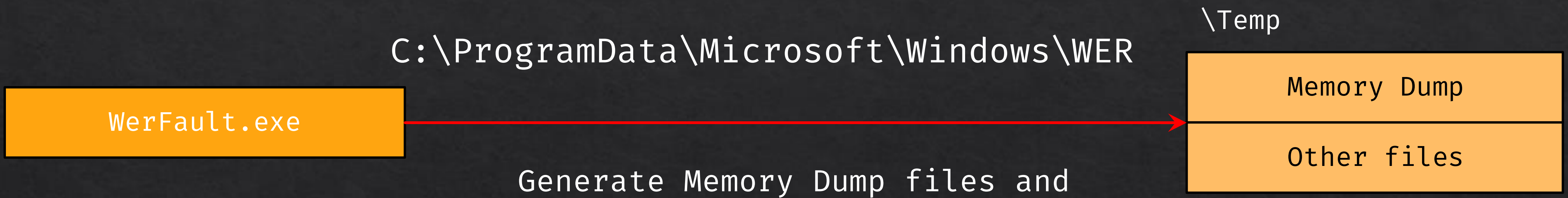
- Windows Error Reporting
  - Assert Filtering Policy
  - BrokerUp
  - Consent
  - DebugApplications
  - DynamicEtwSessions
  - Hangs
  - RuntimeExceptionHelperModules
  - TermReason
  - LocalDumps

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
DumpFolder	REG_SZ	C:\CrashDumps
DumpCount	REG_DWORD	0x00000050 (80)
DumpType	REG_DWORD	0x00000001 (1)



```
10:36... WerFault. exe   17052 CreateFile      C:\CrashDumps                               SUCCESS       Desired Access...
10:36... WerFault. exe   17052 QueryDirectory  C:\CrashDumps\*. dmp                        SUCCESS       Filter: *. dmp,...
10:36... WerFault. exe   17052 QueryDirectory  C:\CrashDumps                               SUCCESS       0: LenovoVanta...
10:36... WerFault. exe   17052 QueryDirectory  C:\CrashDumps                               SUCCESS       0: PetitPotam....
10:36... WerFault. exe   17052 QueryDirectory  C:\CrashDumps                               NO MORE FILES
10:36... WerFault. exe   17052 CloseFile       C:\CrashDumps                               SUCCESS
10:36... WerFault. exe   17052 CreateFile      C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Desired Access...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 0, Len...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 3, 376,...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 3, 380,...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 176, L...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 232, L...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 3, 976,...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 1, 596,...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 526...
10:36... WerFault. exe   17052 ReadFile        C:\CrashDumps\CrashMe. exe. 2268. dmp       END OF FILE   Offset: 8, 192,...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 530...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 962...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 546...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 56, 714...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 562...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 58, 714...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 578...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 91, 482...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 594...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 105, 09...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 610...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 115, 03...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 626...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 115, 29...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 642...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 116, 88...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 658...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 118, 47...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 674...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 126, 67...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 10, 690...
10:36... WerFault. exe   17052 WriteFile       C:\CrashDumps\CrashMe. exe. 2268. dmp       SUCCESS       Offset: 127, 76...
```

Collect and upload crash information for all the process



\Temp

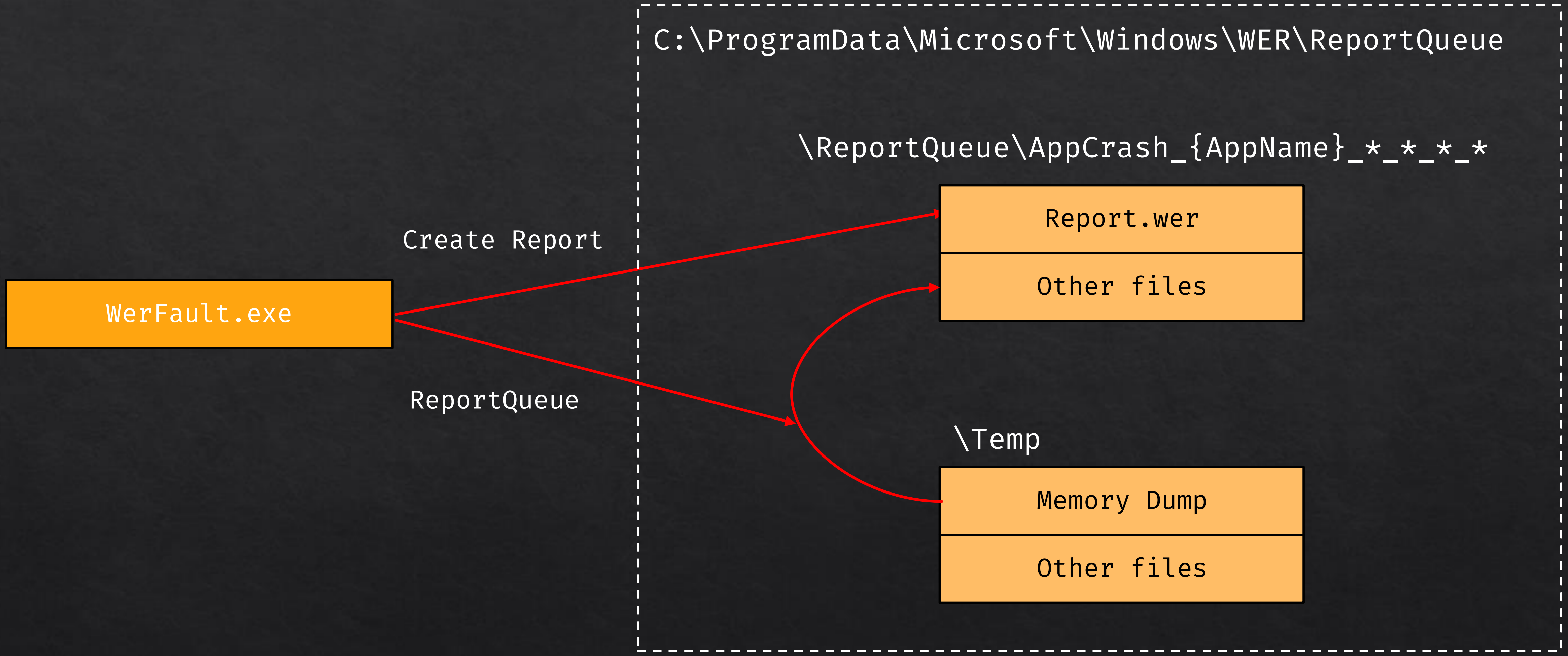
C:\ProgramData\Microsoft\Windows\WER

WerFault.exe

Memory Dump

Other files

Generate Memory Dump files and other files containing crash information.



C:\ProgramData\Microsoft\Windows\WER\ReportQueue

\ReportQueue\AppCrash_{AppName}_*_*_*_*

Report.wer

Other files

Create Report

WerFault.exe

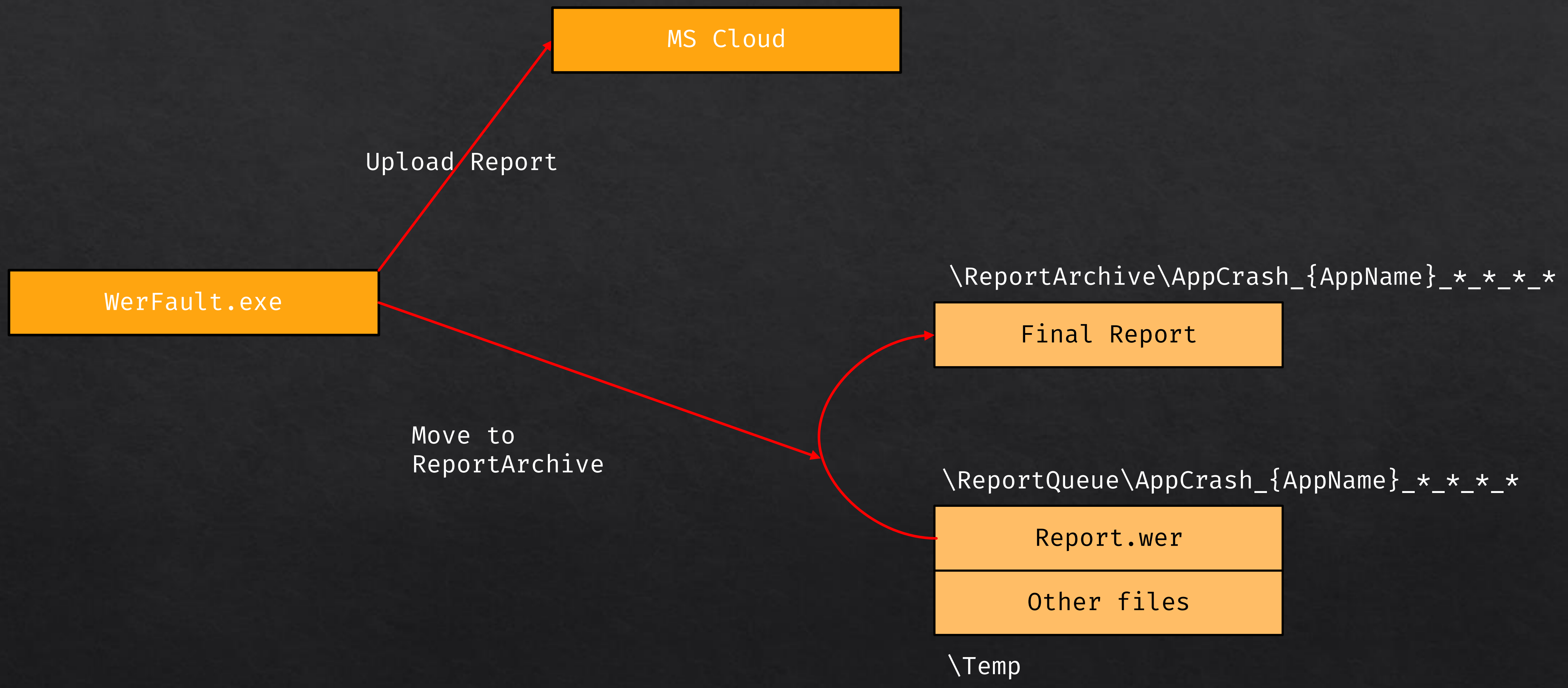
ReportQueue

\Temp

Memory Dump

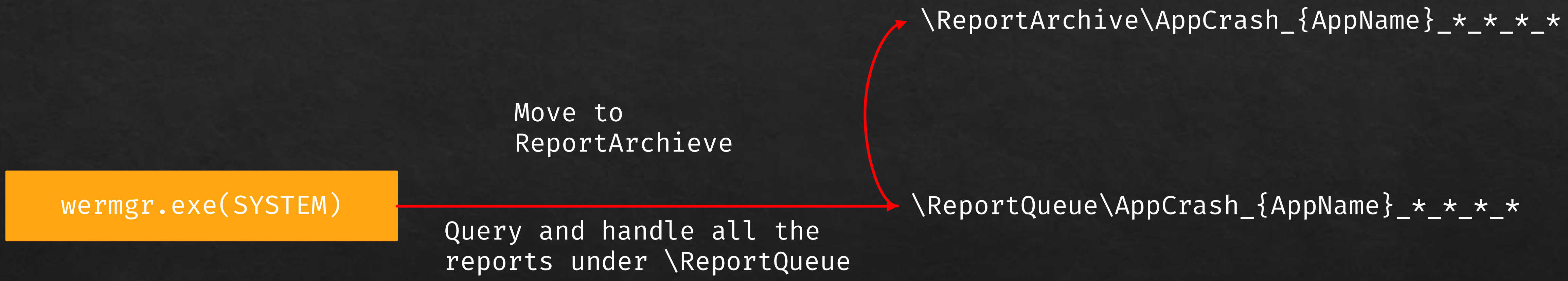
Other files

With Internet Connection



No Internet Connection?

- WerFault.exe remains the reports in \ReportQueue
- WER trigger the schedule task **QueueReporting(wermgr.exe -upload)** to handle the reports



\ReportArchive\AppCrash_{AppName}_*_*_*_*

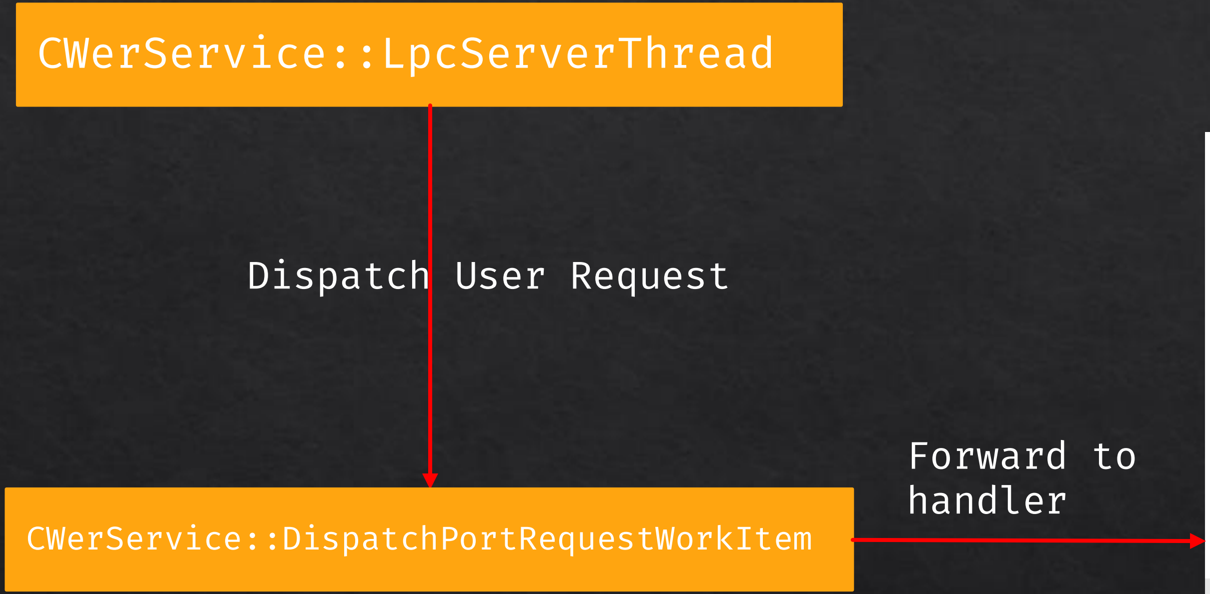
Move to ReportArchieve

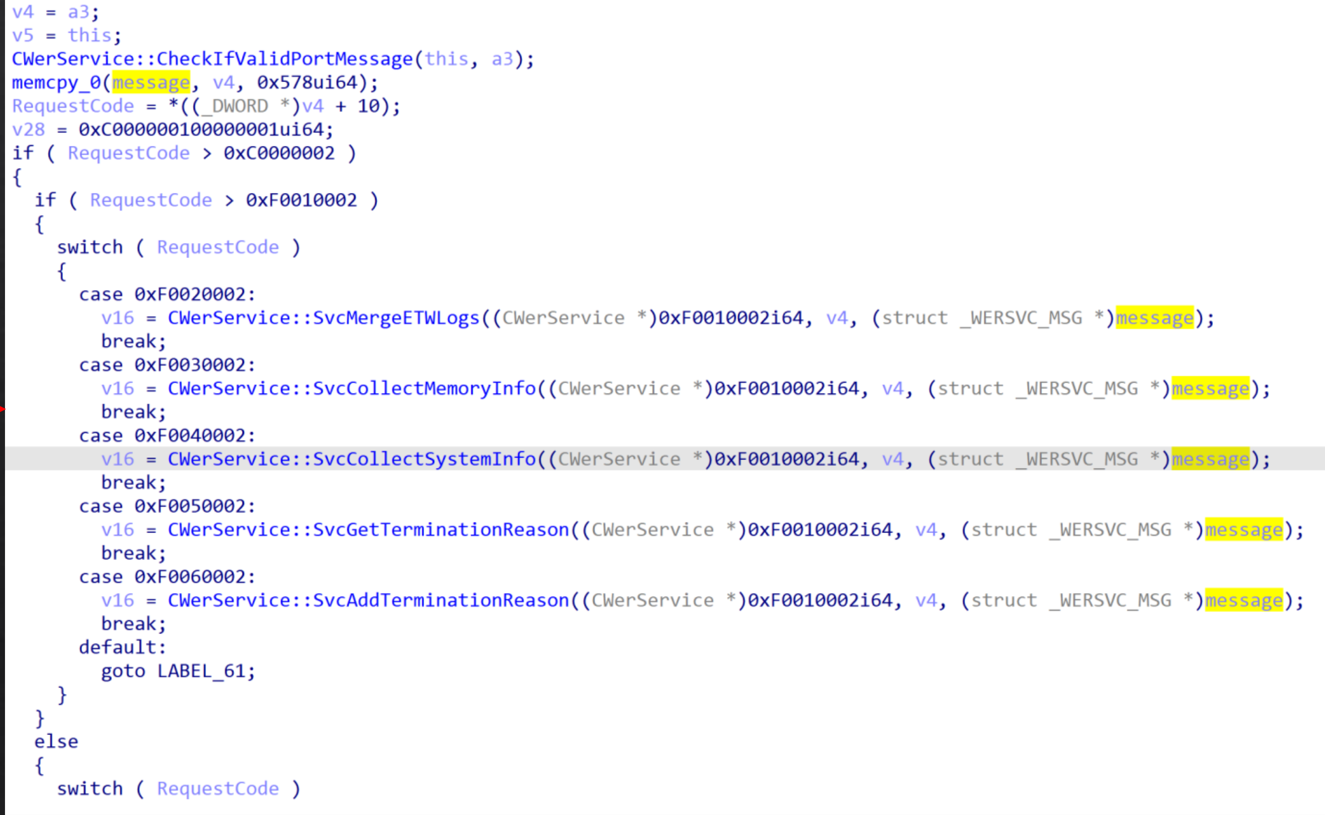
wermgr.exe(SYSTEM)

\ReportQueue\AppCrash_{AppName}_*_*_*_*

Query and handle all the reports under \ReportQueue

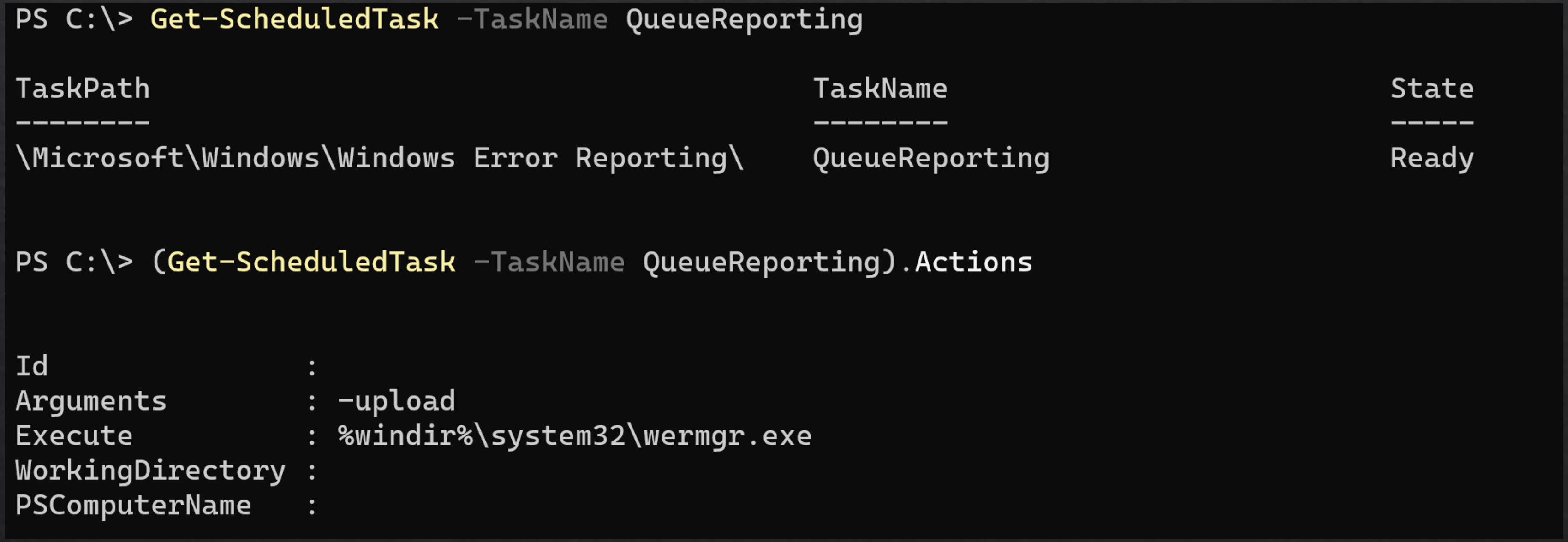
Attack Surface for WER


CWerService::LpcServerThread
Dispatch User Request
CWerService::DispatchPortRequestWorkItem
Forward to handler

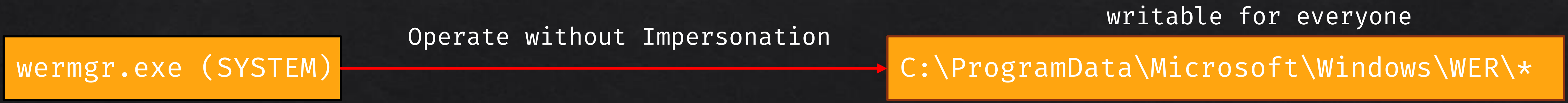


```
v4 = a3;
v5 = this;
CWerService::CheckIfValidPortMessage(this, a3);
memcpy_0(message, v4, 0x578ui64);
RequestCode = *((_DWORD *)v4 + 10);
v28 = 0xC000000100000001ui64;
if ( RequestCode > 0xC0000002 )
{
  if ( RequestCode > 0xF0010002 )
  {
    switch ( RequestCode )
    {
      case 0xF0020002:
        v16 = CWerService::SvcMergeETWLogs((CWerService *)0xF0010002i64, v4, (struct _WERSVC_MSG *)message);
        break;
      case 0xF0030002:
        v16 = CWerService::SvcCollectMemoryInfo((CWerService *)0xF0010002i64, v4, (struct _WERSVC_MSG *)message);
        break;
      case 0xF0040002:
        v16 = CWerService::SvcCollectSystemInfo((CWerService *)0xF0010002i64, v4, (struct _WERSVC_MSG *)message);
        break;
      case 0xF0050002:
        v16 = CWerService::SvcGetTerminationReason((CWerService *)0xF0010002i64, v4, (struct _WERSVC_MSG *)message);
        break;
      case 0xF0060002:
        v16 = CWerService::SvcAddTerminationReason((CWerService *)0xF0010002i64, v4, (struct _WERSVC_MSG *)message);
        break;
      default:
        goto LABEL_61;
    }
  }
  else
  {
    switch ( RequestCode )
```

Attack Surface for WER



```
PS C:\> Get-ScheduledTask -TaskName QueueReporting

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\Microsoft\Windows\Windows Error Reporting\    QueueReporting                    Ready


PS C:\> (Get-ScheduledTask -TaskName QueueReporting).Actions


Id                 :
Arguments          : -upload
Execute            : %windir%\system32\wermgr.exe
WorkingDirectory   :
PSComputerName     :
```



wermgr.exe (SYSTEM)

Operate without Impersonation

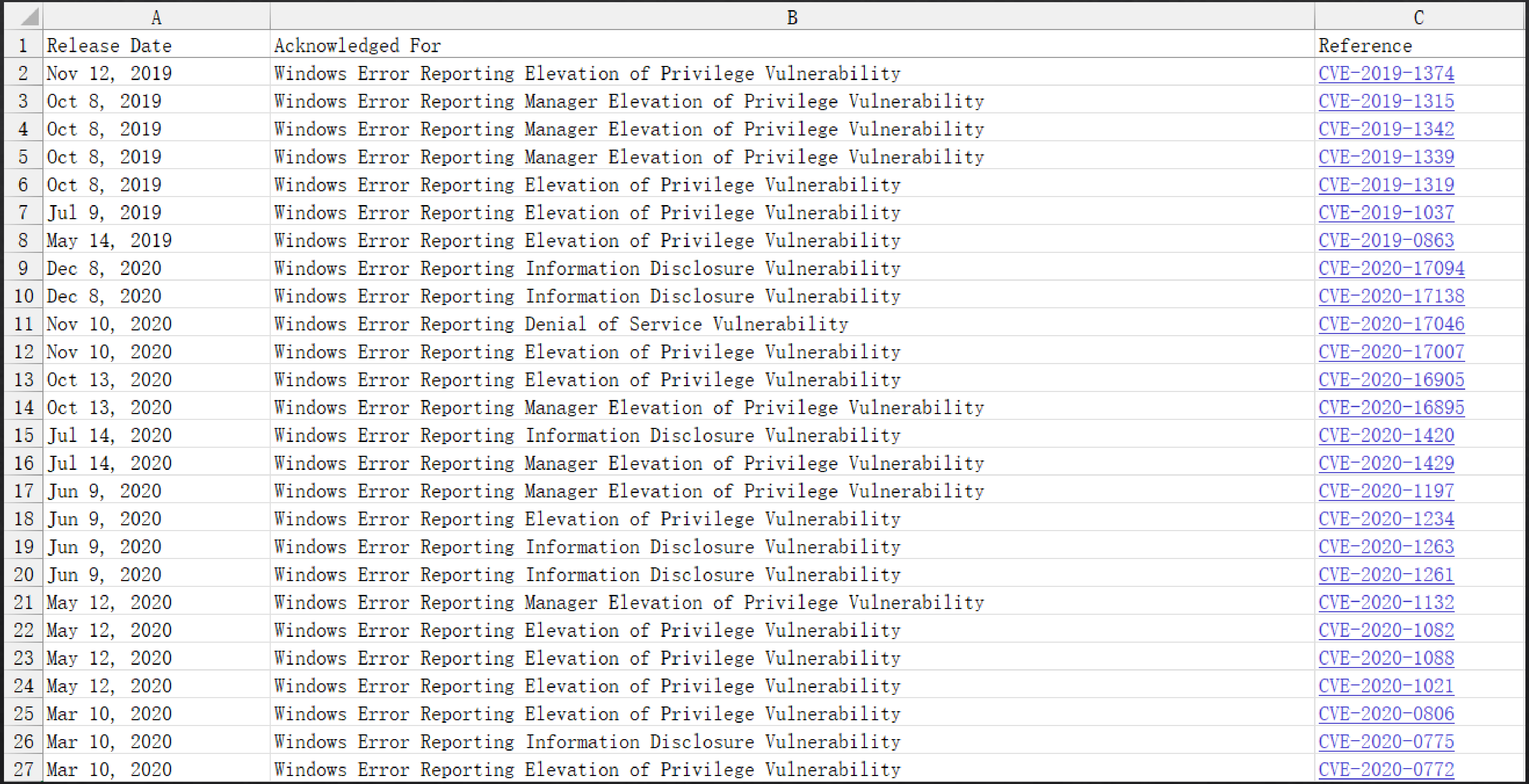
writable for everyone

C:\ProgramData\Microsoft\Windows\WER\*

Part 2

Vulnerabilities history of Windows Error Reporting

More than 25 vulnerabilities in Windows Error Reporting



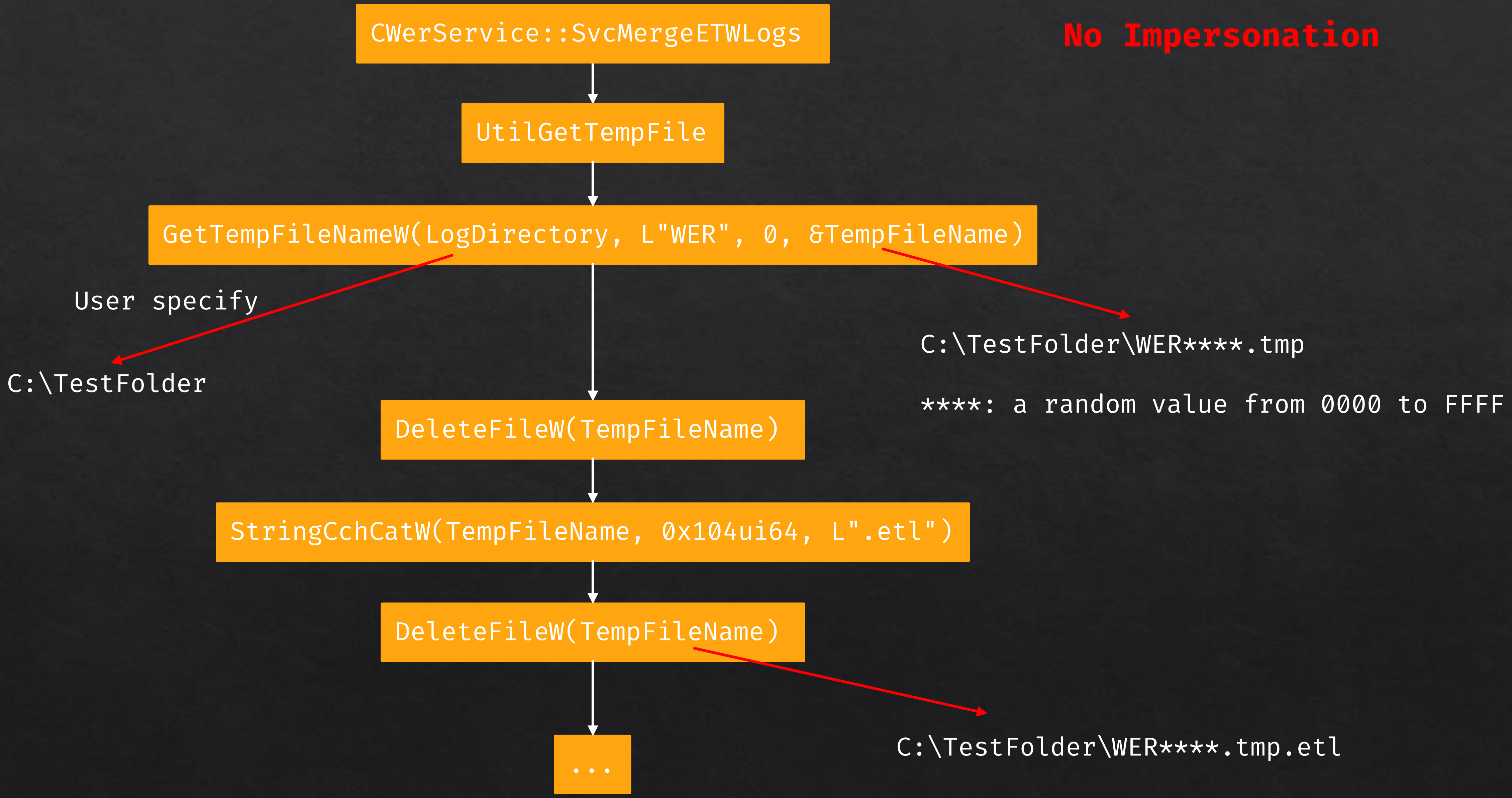
	A	B	C
1	Release Date	Acknowledged For	Reference
2	Nov 12, 2019	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2019-1374
3	Oct 8, 2019	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2019-1315
4	Oct 8, 2019	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2019-1342
5	Oct 8, 2019	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2019-1339
6	Oct 8, 2019	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2019-1319
7	Jul 9, 2019	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2019-1037
8	May 14, 2019	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2019-0863
9	Dec 8, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-17094
10	Dec 8, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-17138
11	Nov 10, 2020	Windows Error Reporting Denial of Service Vulnerability	CVE-2020-17046
12	Nov 10, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-17007
13	Oct 13, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-16905
14	Oct 13, 2020	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2020-16895
15	Jul 14, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-1420
16	Jul 14, 2020	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2020-1429
17	Jun 9, 2020	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2020-1197
18	Jun 9, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-1234
19	Jun 9, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-1263
20	Jun 9, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-1261
21	May 12, 2020	Windows Error Reporting Manager Elevation of Privilege Vulnerability	CVE-2020-1132
22	May 12, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-1082
23	May 12, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-1088
24	May 12, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-1021
25	Mar 10, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-0806
26	Mar 10, 2020	Windows Error Reporting Information Disclosure Vulnerability	CVE-2020-0775
27	Mar 10, 2020	Windows Error Reporting Elevation of Privilege Vulnerability	CVE-2020-0772

Almost all of them are related to path redirection attacks.

Why WER is the good target for logic bug hunting?

- WER ALPC port allows everyone to communicate with it.
- Multiple message handler
- Running as SYSTEM IL and the main working directory C:\ProgramData\Microsoft\Windows\WER is writable for unprivileged user
- File system access without impersonation

Vulnerability Sample



No Impersonation

CWerService::SvcMergeETWLogs

UtilGetTempFile

GetTempFileNameW(LogDirectory, L"WER", 0, &TempFileName)

User specify

C:\TestFolder

C:\TestFolder\WER****.tmp

****: a random value from 0000 to FFFF

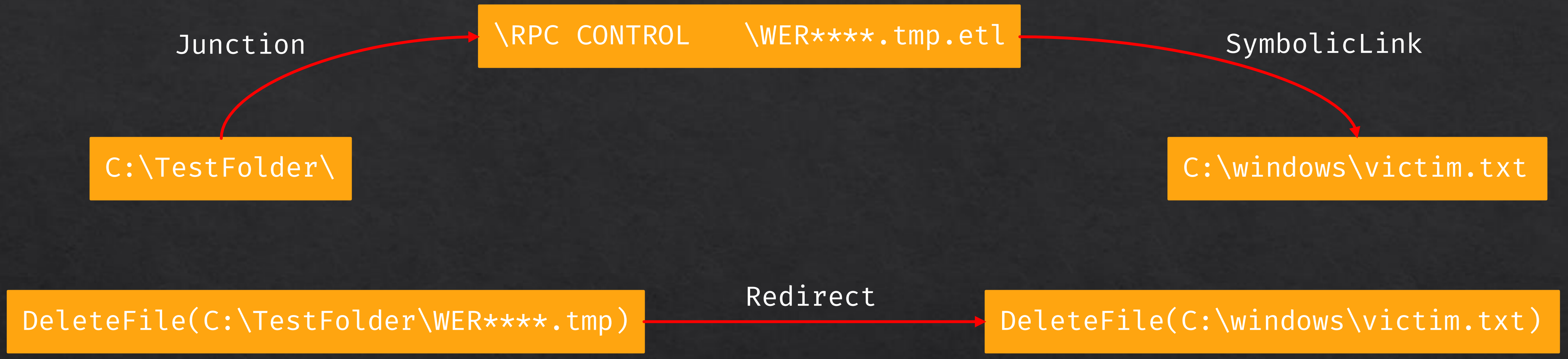
DeleteFileW(TempFileName)

StringCchCatW(TempFileName, 0x104ui64, L".etl")

DeleteFileW(TempFileName)

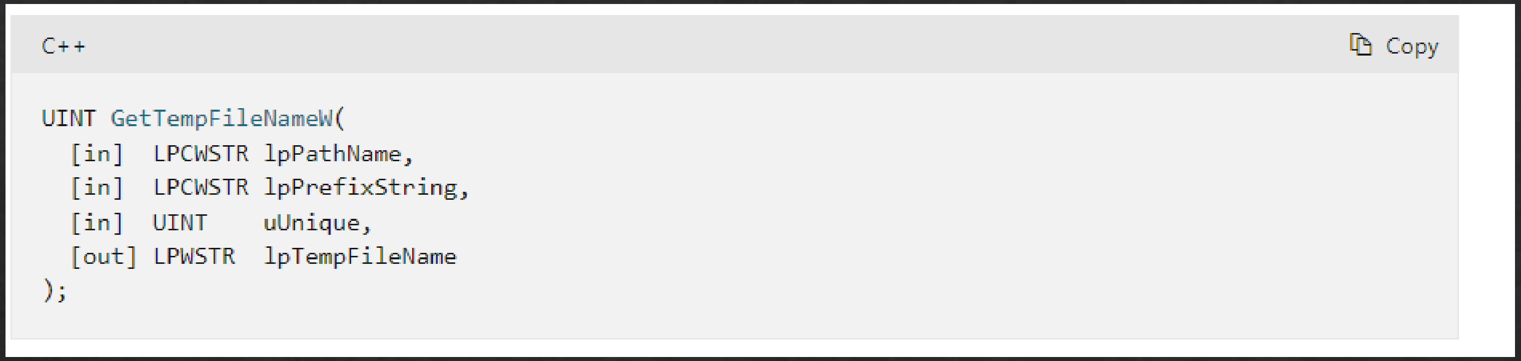
C:\TestFolder\WER****.tmp.etl

...


Junction
\RPC CONTROL   \WER****.tmp.etl
SymbolicLink
C:\TestFolder\
C:\windows\victim.txt
DeleteFile(C:\TestFolder\WER****.tmp)
Redirect
DeleteFile(C:\windows\victim.txt)

Trouble:

- We must know the filename to perform path redirection attack

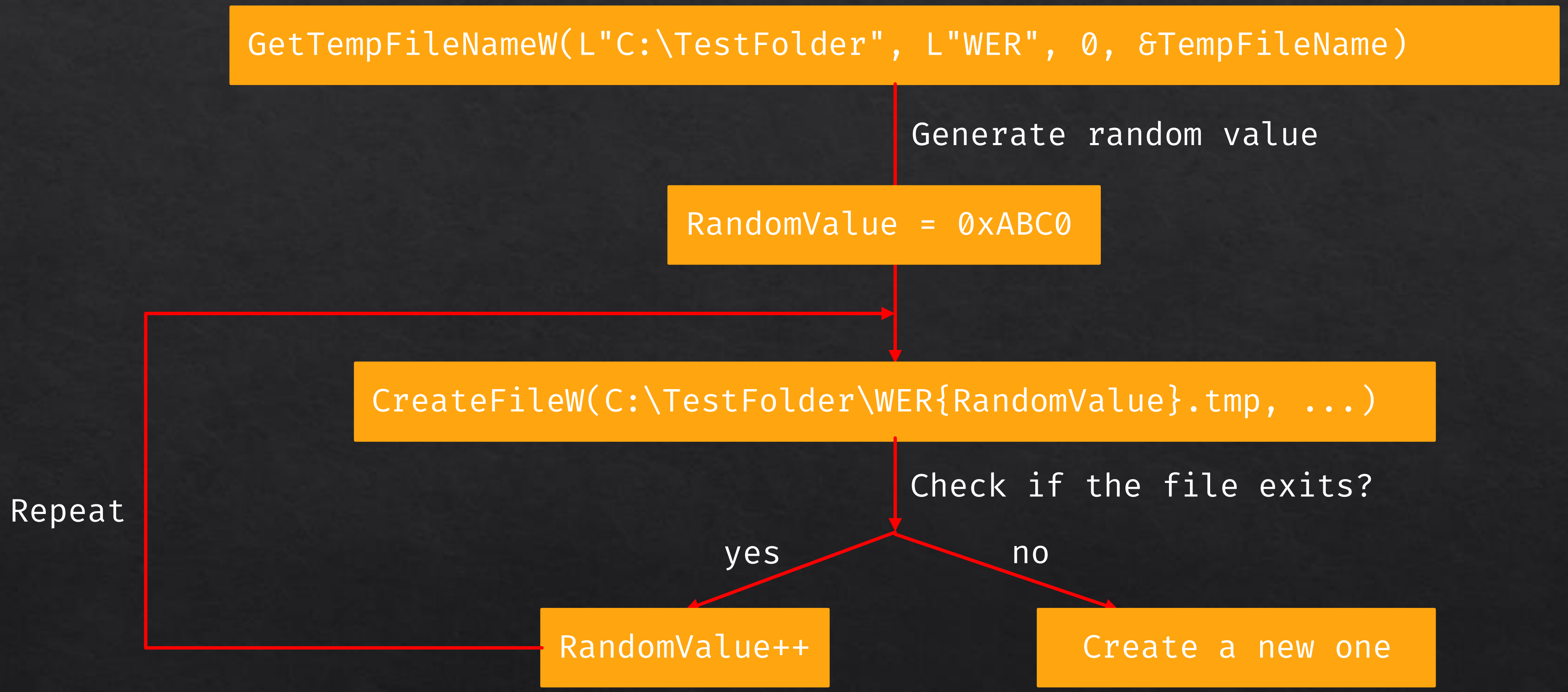


```cpp
UINT GetTempFileNameW(
  [in]  LPCWSTR lpPathName,
  [in]  LPCWSTR lpPrefixString,
  [in]  UINT    uUnique,
  [out] LPWSTR  lpTempFileName
);
```

```cpp
GetTempFileNameW(L"C:\TestFolder", L"WER", 0, &TempFileName)
```

Must be a unique file name under "C:\TestFolder"

How to make sure the filename is unique?



GetTempFileNameW(L"C:\TestFolder", L"WER", 0, &TempFileName)

Generate random value

RandomValue = 0xABC0

CreateFileW(C:\TestFolder\WER{RandomValue}.tmp, ...)

Check if the file exits?

yes

no

RandomValue++

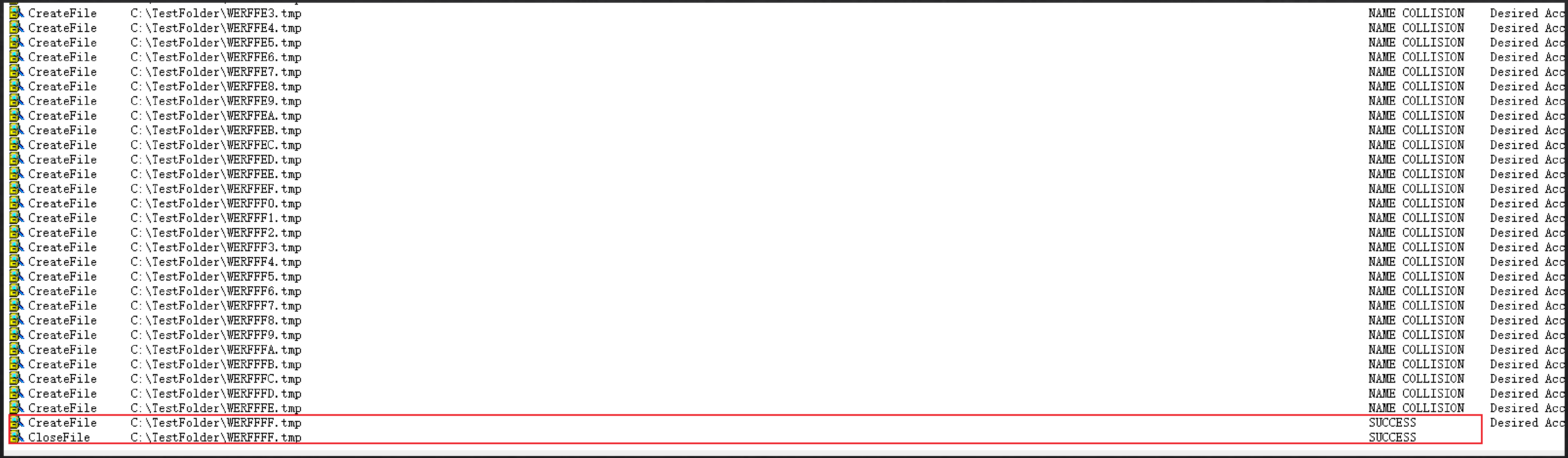
Create a new one

Repeat

Make **GetTempFileName** returns a unique filename

```
GetTempFileNameW(L"C:\TestFolder", L"WER", 0, &TempFileName)
```

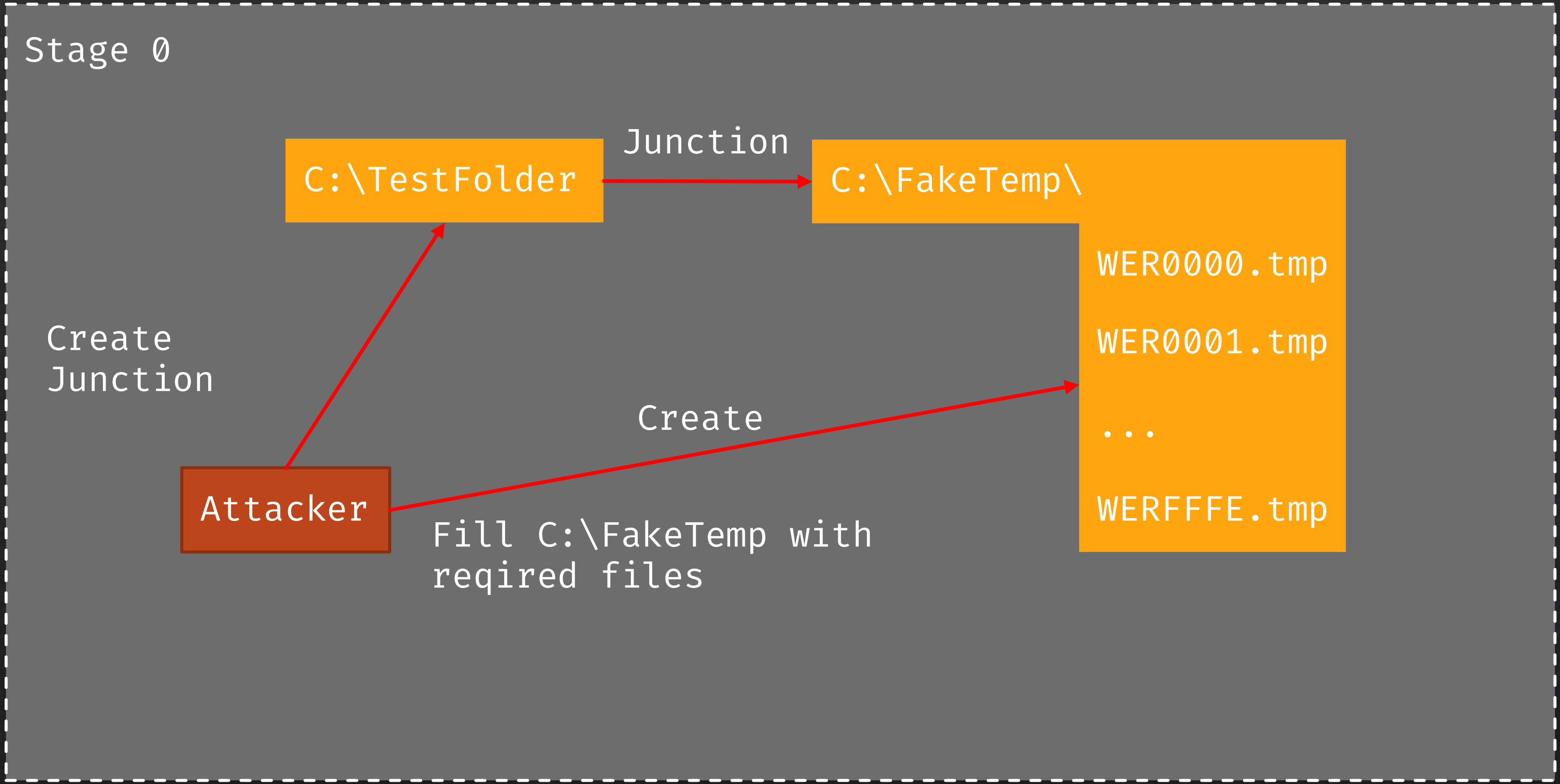
Fill C:\TestFolder with files **WER0000.tmp -> WERFFFE.tmp**



```
CreateFile    C:\TestFolder\WERFFE3.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE4.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE5.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE6.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE7.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE8.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFE9.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFEA.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFEB.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFEC.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFED.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFEE.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFEF.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF0.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF1.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF2.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF3.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF4.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF5.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF6.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF7.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF8.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFF9.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFA.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFB.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFC.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFD.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFE.tmp    NAME COLLISION    Desired Acc
CreateFile    C:\TestFolder\WERFFFF.tmp    SUCCESS           Desired Acc
CloseFile     C:\TestFolder\WERFFFF.tmp    SUCCESS
```

We’ll always get **WERFFFF.tmp**

Exploit



Stage 0

C:\TestFolder

Junction

C:\FakeTemp\

WER0000.tmp

WER0001.tmp

...

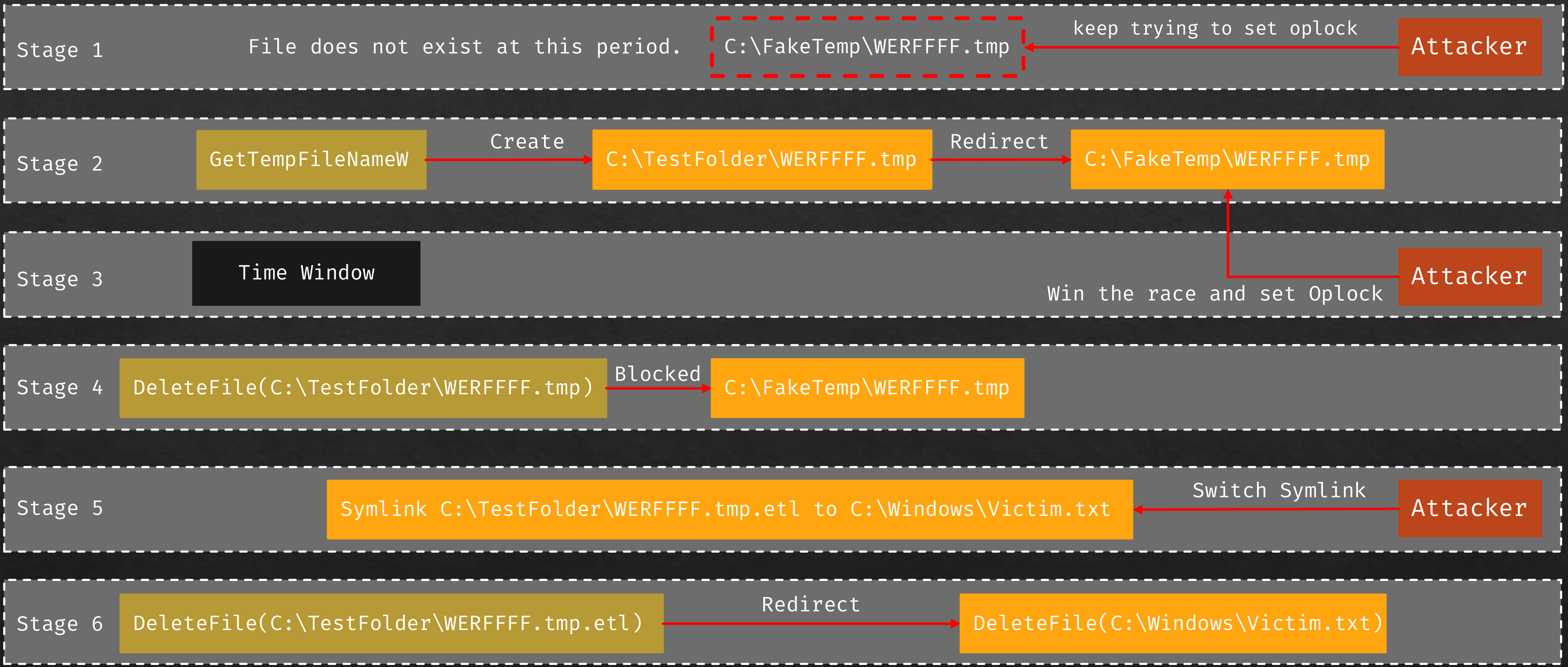
WERFFFE.tmp

Create
Junction

Attacker

Create

Fill C:\FakeTemp with
reqired files



Stage 1 File does not exist at this period. C:\FakeTemp\WERFFFF.tmp keep trying to set oplock Attacker

Stage 2 GetTempFileNameW Create C:\TestFolder\WERFFFF.tmp Redirect C:\FakeTemp\WERFFFF.tmp

Stage 3 Time Window Win the race and set Oplock Attacker

Stage 4 DeleteFile(C:\TestFolder\WERFFFF.tmp) Blocked C:\FakeTemp\WERFFFF.tmp

Stage 5 Symlink C:\TestFolder\WERFFFF.tmp.etl to C:\Windows\Victim.txt Switch Symlink Attacker

Stage 6 DeleteFile(C:\TestFolder\WERFFFF.tmp.etl) Redirect DeleteFile(C:\Windows\Victim.txt)

Even if we used the oplock, we still need to win the race ☹. We can trigger the bug repeatedly to win the race.

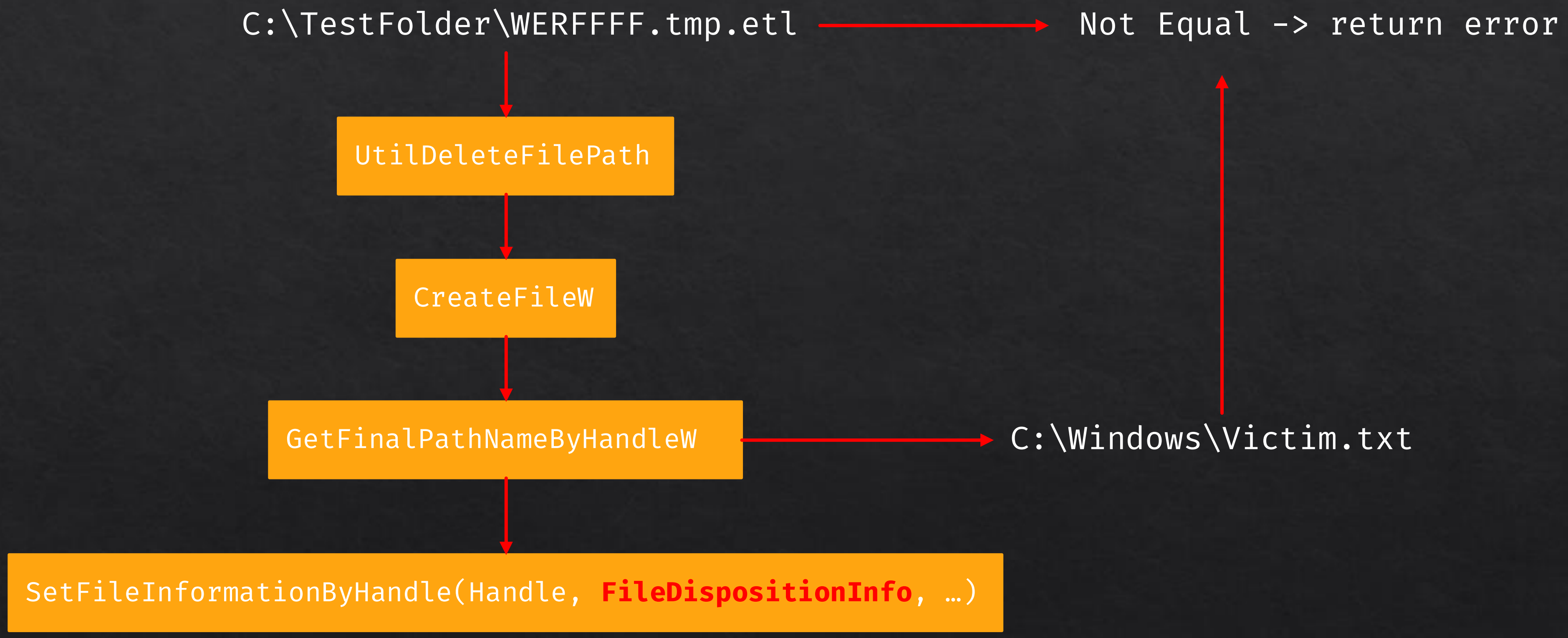
From Arbitrary Deletes to SYSTEM:

Abusing Windows Installer services to get EOP

By researcher: Abdelhamid Naceri

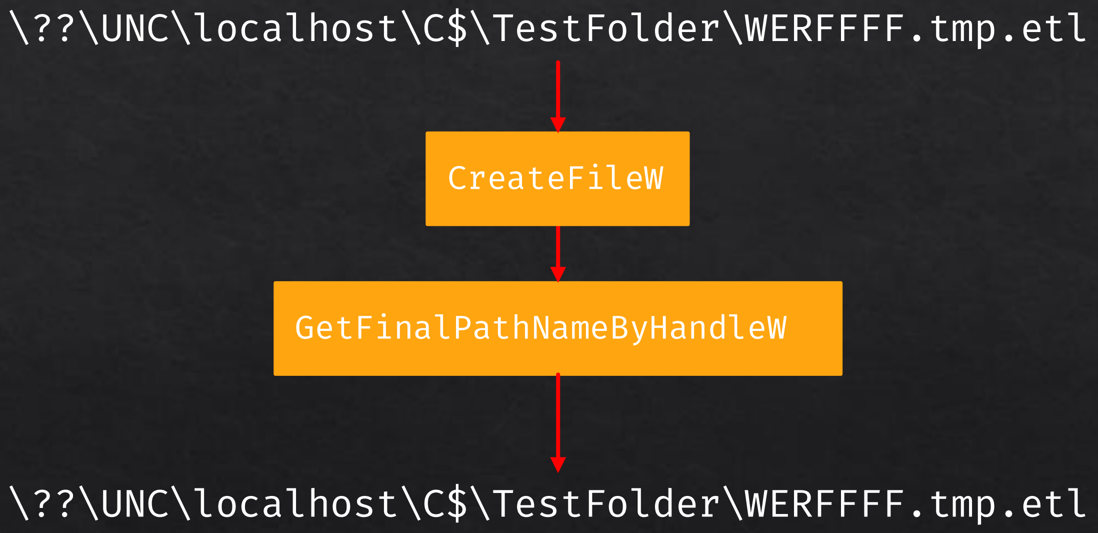
First Patch

- Replace **DeleteFile** with **UtilDeleteFilePath**


C:\TestFolder\WERFFFF.tmp.etl
Not Equal -> return error
UtilDeleteFilePath
CreateFileW
GetFinalPathNameByHandleW
C:\Windows\Victim.txt
SetFileInformationByHandle(Handle, FileDispositionInfo, …)

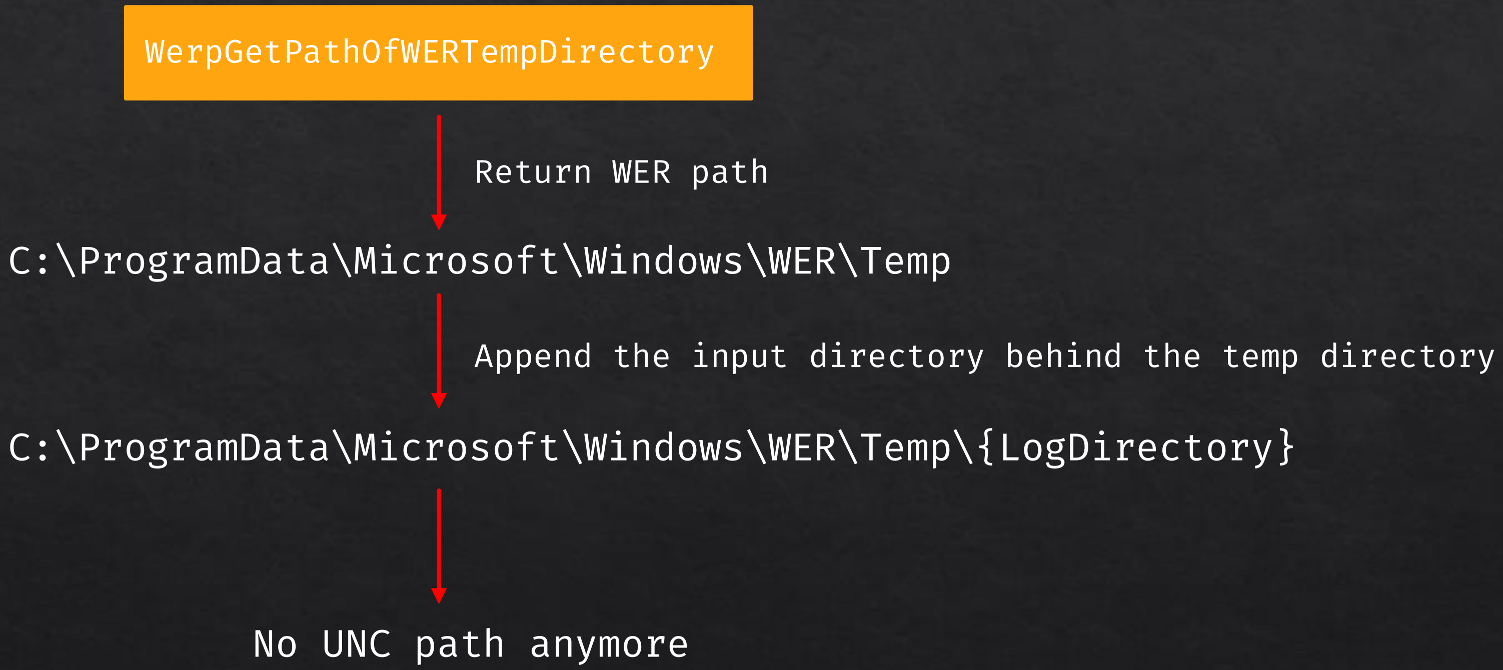
Bypass with UNC Path

- Convert the LogDirectory into UNC format: **C:\TestFolder -> \??\UNC\localhost\C$\TestFolder**
- Symlink C:\TestFolder\WERFFFF.tmp.etl -> C:\Windows\Victim.txt


\??\UNC\localhost\C$\TestFolder\WERFFFF.tmp.etl
CreateFileW
GetFinalPathNameByHandleW
\??\UNC\localhost\C$\TestFolder\WERFFFF.tmp.etl

Final Patch

Add function WerpGetPathOfWERTempDirectory and restrict the LogDirectory



WerpGetPathOfWERTempDirectory

Return WER path

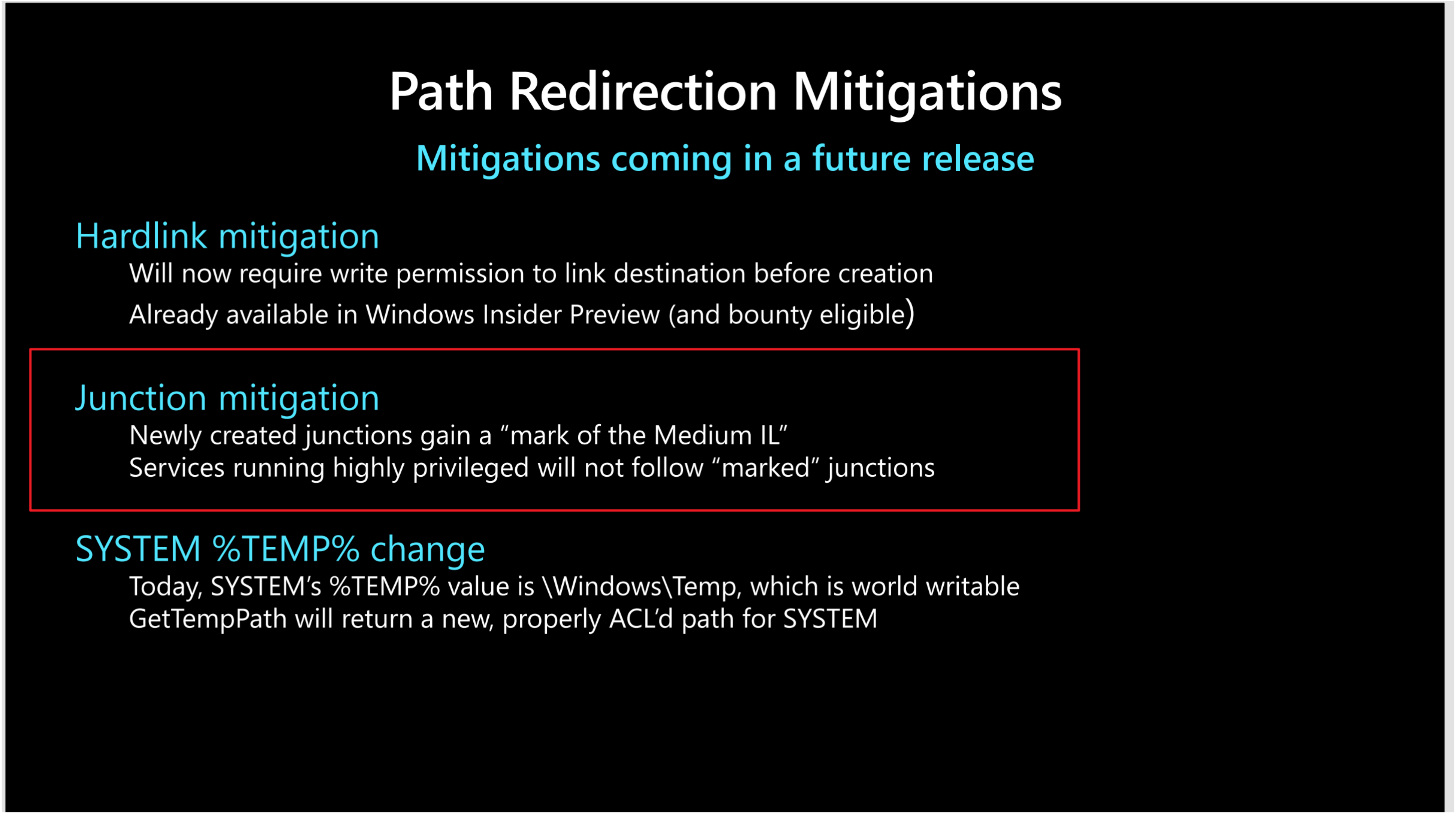
C:\ProgramData\Microsoft\Windows\WER\Temp

Append the input directory behind the temp directory

C:\ProgramData\Microsoft\Windows\WER\Temp\{LogDirectory}

No UNC path anymore

General Mitigation



Path Redirection Mitigations

Mitigations coming in a future release

Hardlink mitigation

Will now require write permission to link destination before creation

Already available in Windows Insider Preview (and bounty eligible)

Junction mitigation

Newly created junctions gain a “mark of the Medium IL”

Services running highly privileged will not follow “marked” junctions

SYSTEM %TEMP% change

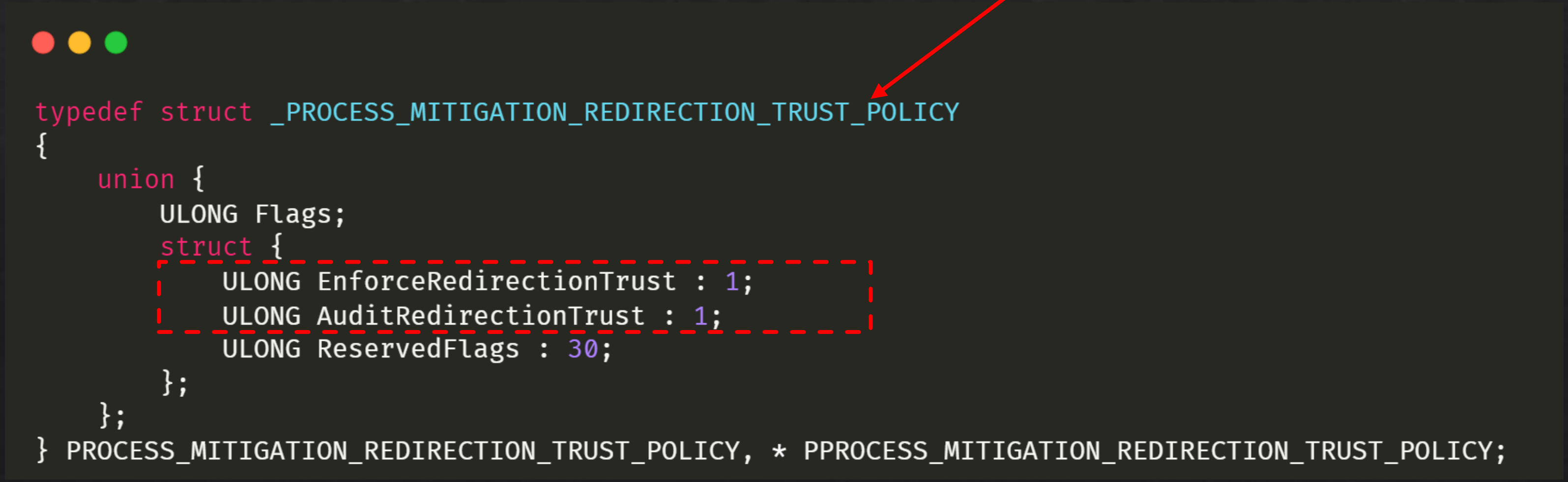
Today, SYSTEM’s %TEMP% value is \Windows\Temp, which is world writable

GetTempPath will return a new, properly ACL’d path for SYSTEM

Mitigation is not enabled to all the processes except the vulnerable process.

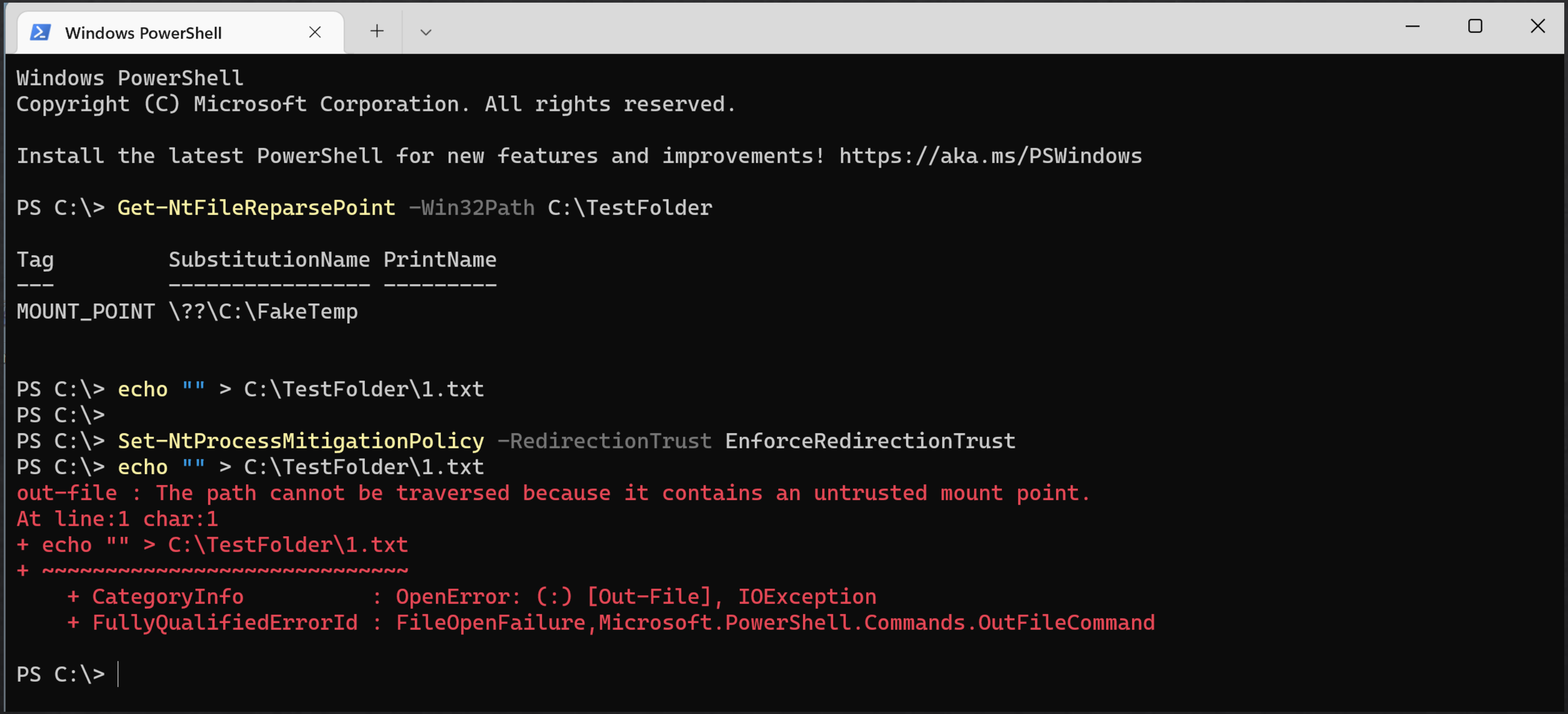
Ways to enable mitigation:

- New mitigation policy ProcessRedirectionTrustPolicy
- SetProcessMitigationPolicy(ProcessRedirectionTrustPolicy, &policy, 4)
- Enable EnforceRedirectionTrust



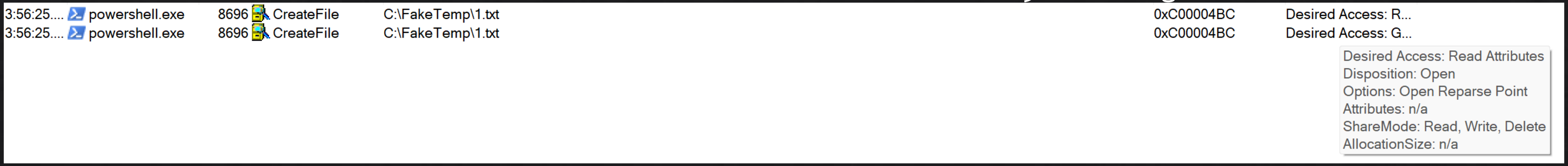
```
typedef struct _PROCESS_MITIGATION_REDIRECTION_TRUST_POLICY
{
    union {
        ULONG Flags;
        struct {
            ULONG EnforceRedirectionTrust : 1;
            ULONG AuditRedirectionTrust : 1;
            ULONG ReservedFlags : 30;
        };
    };
} PROCESS_MITIGATION_REDIRECTION_TRUST_POLICY, * PPROCESS_MITIGATION_REDIRECTION_TRUST_POLICY;
```

Open junction will get error when EnforceRedirectionTrust is enabled



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\> Get-NtFileReparsePoint -Win32Path C:\TestFolder

Tag         SubstitutionName PrintName
---         ---------------- ---------
MOUNT_POINT \??\C:\FakeTemp


PS C:\> echo "" > C:\TestFolder\1.txt
PS C:\>
PS C:\> Set-NtProcessMitigationPolicy -RedirectionTrust EnforceRedirectionTrust
PS C:\> echo "" > C:\TestFolder\1.txt
out-file : The path cannot be traversed because it contains an untrusted mount point.
At line:1 char:1
+ echo "" > C:\TestFolder\1.txt
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (:) [Out-File], IOException
    + FullyQualifiedErrorId : FileOpenFailure,Microsoft.PowerShell.Commands.OutFileCommand

PS C:\>
```

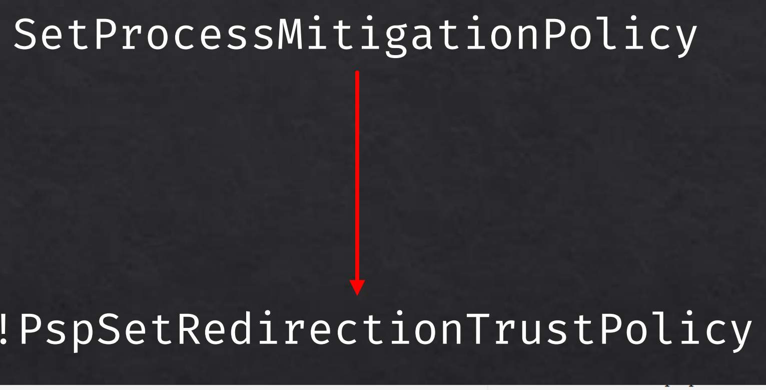
```
Install-Module -Name
NtObjectManager
```



```
3:56:25.... powershell.exe  8696 CreateFile  C:\FakeTemp\1.txt   0xC00004BC  Desired Access: R...
3:56:25.... powershell.exe  8696 CreateFile  C:\FakeTemp\1.txt   0xC00004BC  Desired Access: G...
```

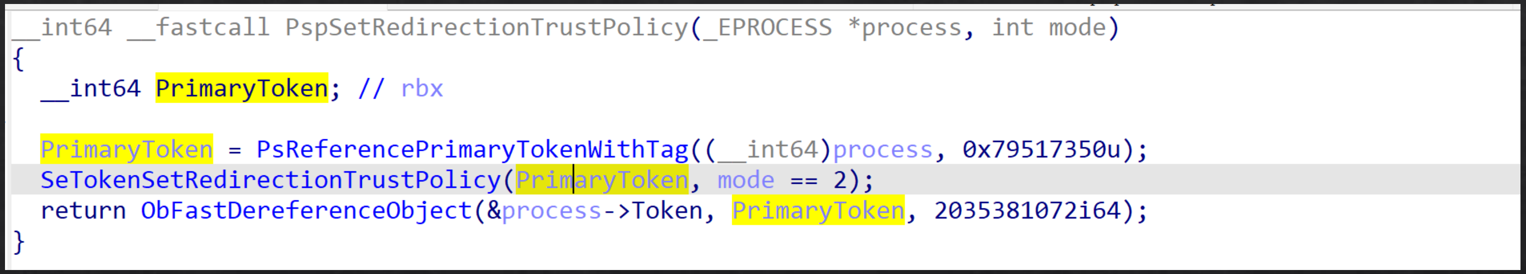
Desired Access: Read Attributes
Disposition: Open
Options: Open Reparse Point
Attributes: n/a
ShareMode: Read, Write, Delete
AllocationSize: n/a

How mitigation works?

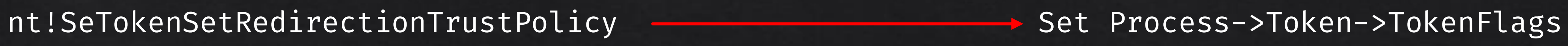


SetProcessMitigationPolicy

nt!PspSetRedirectionTrustPolicy



```
__int64 __fastcall PspSetRedirectionTrustPolicy(_EPROCESS *process, int mode)
{
  __int64 PrimaryToken; // rbx

  PrimaryToken = PsReferencePrimaryTokenWithTag((__int64)process, 0x79517350u);
  SeTokenSetRedirectionTrustPolicy(PrimaryToken, mode == 2);
  return ObFastDereferenceObject(&process->Token, PrimaryToken, 2035381072i64);
}
```

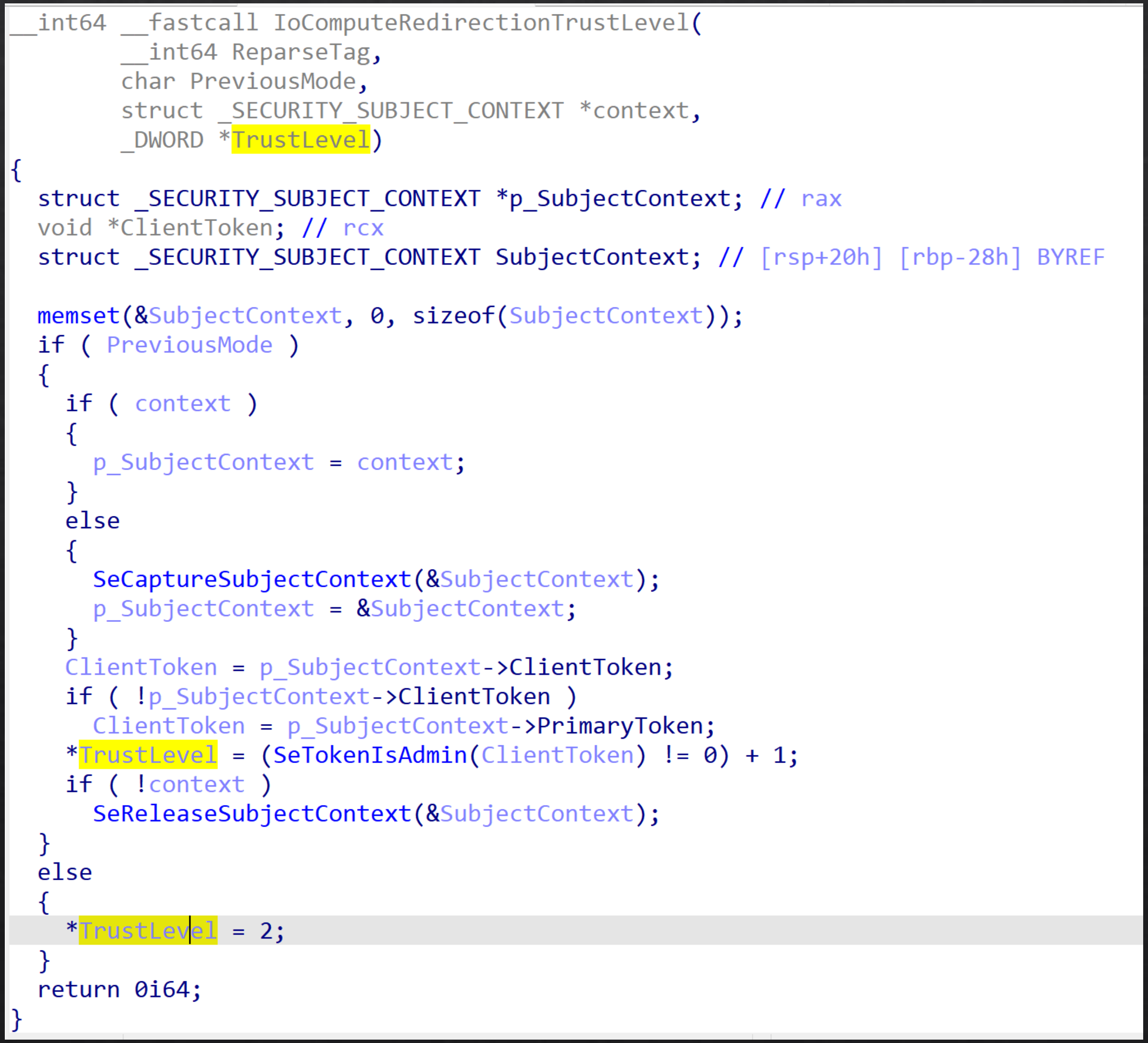


nt!SeTokenSetRedirectionTrustPolicy

Set Process->Token->TokenFlags

How mitigation works?

Create Junction -> … -> ntfs.sys!NtfsSetReparsePointInternal -> nt!IoComputeRedirectionTrustLevel

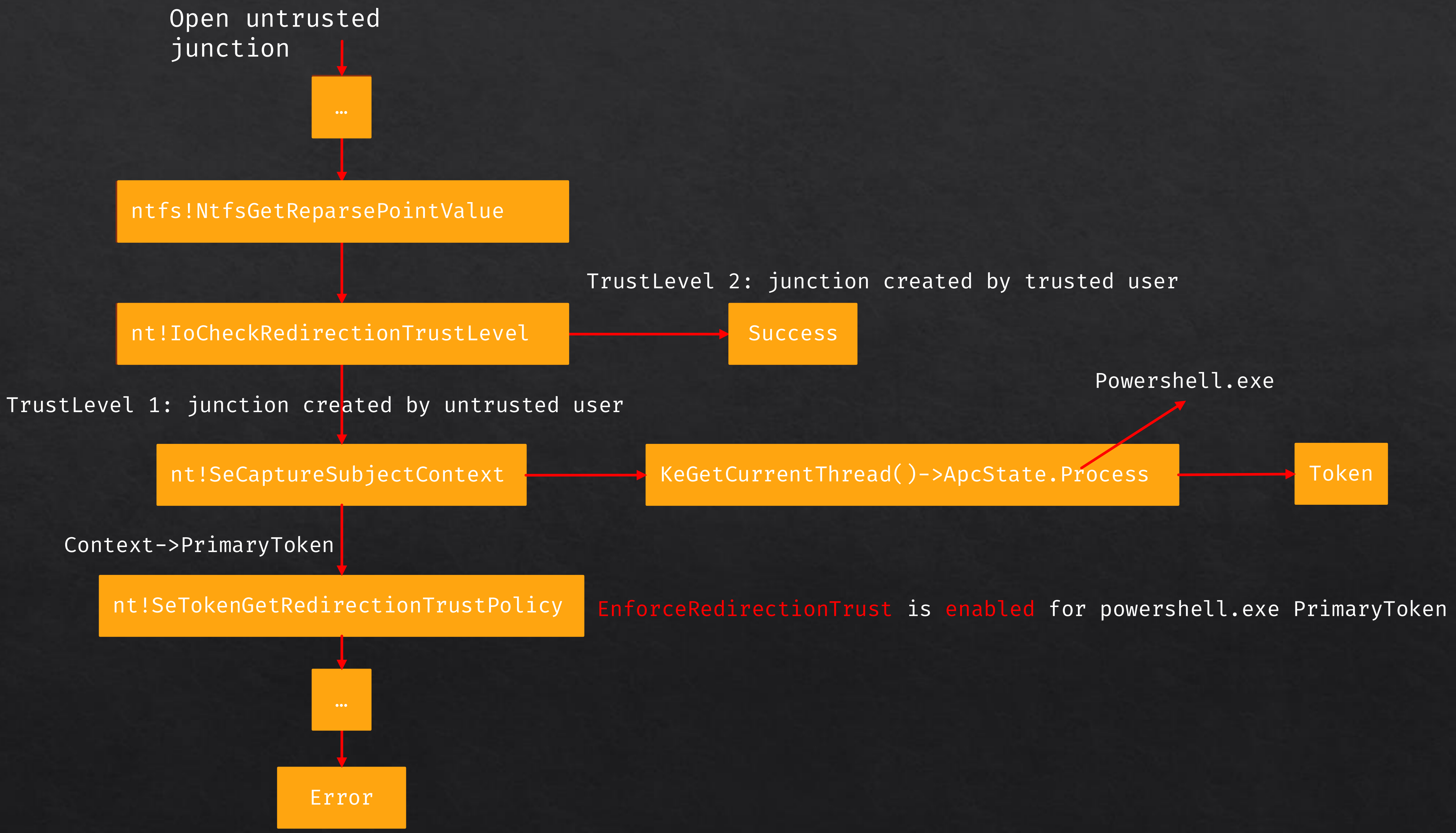


```
__int64 __fastcall IoComputeRedirectionTrustLevel(
        __int64 ReparseTag,
        char PreviousMode,
        struct _SECURITY_SUBJECT_CONTEXT *context,
        _DWORD *TrustLevel)
{
  struct _SECURITY_SUBJECT_CONTEXT *p_SubjectContext; // rax
  void *ClientToken; // rcx
  struct _SECURITY_SUBJECT_CONTEXT SubjectContext; // [rsp+20h] [rbp-28h] BYREF

  memset(&SubjectContext, 0, sizeof(SubjectContext));
  if ( PreviousMode )
  {
    if ( context )
    {
      p_SubjectContext = context;
    }
    else
    {
      SeCaptureSubjectContext(&SubjectContext);
      p_SubjectContext = &SubjectContext;
    }
    ClientToken = p_SubjectContext->ClientToken;
    if ( !p_SubjectContext->ClientToken )
      ClientToken = p_SubjectContext->PrimaryToken;
    *TrustLevel = (SeTokenIsAdmin(ClientToken) != 0) + 1;
    if ( !context )
      SeReleaseSubjectContext(&SubjectContext);
  }
  else
  {
    *TrustLevel = 2;
  }
  return 0i64;
}
```

TrustLevel 1 : Created by Untrusted User (Medium User)

TrustLevel 2 : Created by Trusted User (Privileged User)

How mitigation works?



Open untrusted junction

…

ntfs!NtfsGetReparsePointValue

nt!IoCheckRedirectionTrustLevel

TrustLevel 2: junction created by trusted user

Success

TrustLevel 1: junction created by untrusted user

nt!SeCaptureSubjectContext

KeGetCurrentThread()->ApcState.Process

Powershell.exe

Token

Context->PrimaryToken

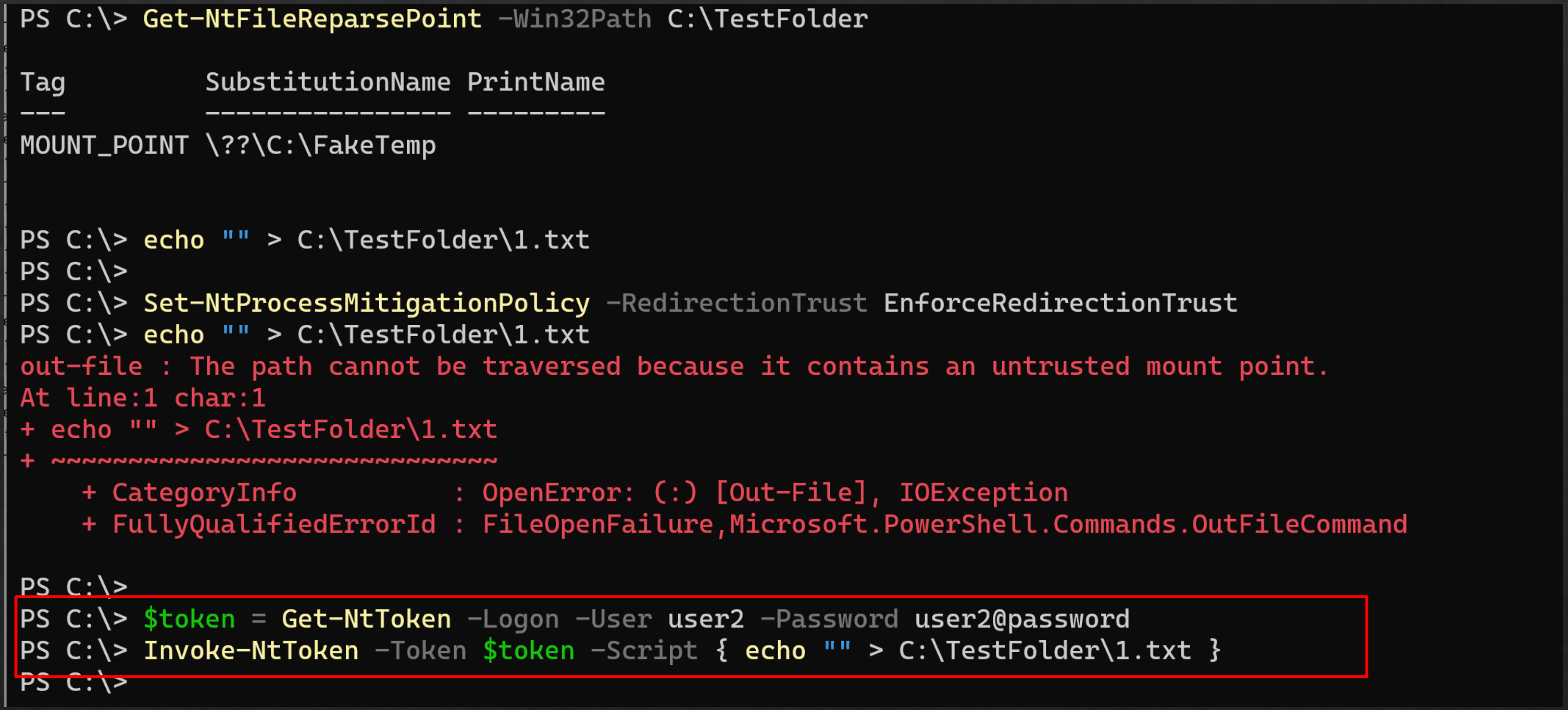
nt!SeTokenGetRedirectionTrustPolicy

EnforceRedirectionTrust is enabled for powershell.exe PrimaryToken

…

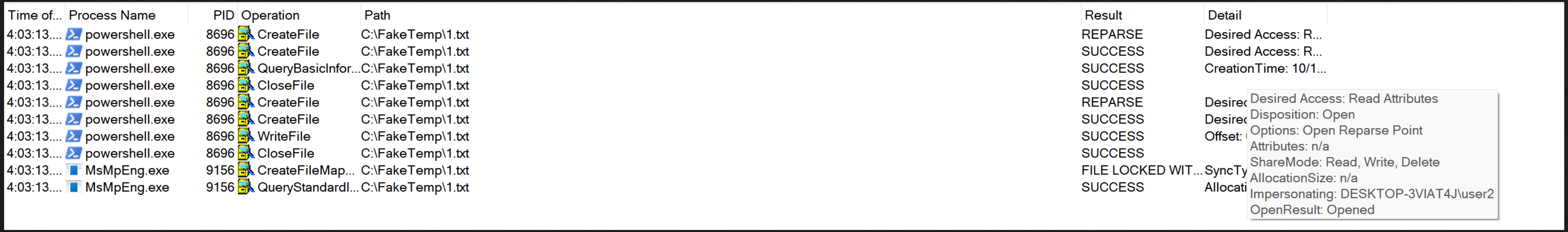
Error

Mitigation bypass when impersonating another user



```
PS C:\> Get-NtFileReparsePoint -Win32Path C:\TestFolder

Tag         SubstitutionName PrintName
---         ---------------- ---------
MOUNT_POINT \??\C:\FakeTemp


PS C:\> echo "" > C:\TestFolder\1.txt
PS C:\>
PS C:\> Set-NtProcessMitigationPolicy -RedirectionTrust EnforceRedirectionTrust
PS C:\> echo "" > C:\TestFolder\1.txt
out-file : The path cannot be traversed because it contains an untrusted mount point.
At line:1 char:1
+ echo "" > C:\TestFolder\1.txt
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (:) [Out-File], IOException
    + FullyQualifiedErrorId : FileOpenFailure,Microsoft.PowerShell.Commands.OutFileCommand

PS C:\>
PS C:\> $token = Get-NtToken -Logon -User user2 -Password user2@password
PS C:\> Invoke-NtToken -Token $token -Script { echo "" > C:\TestFolder\1.txt }
PS C:\>
```

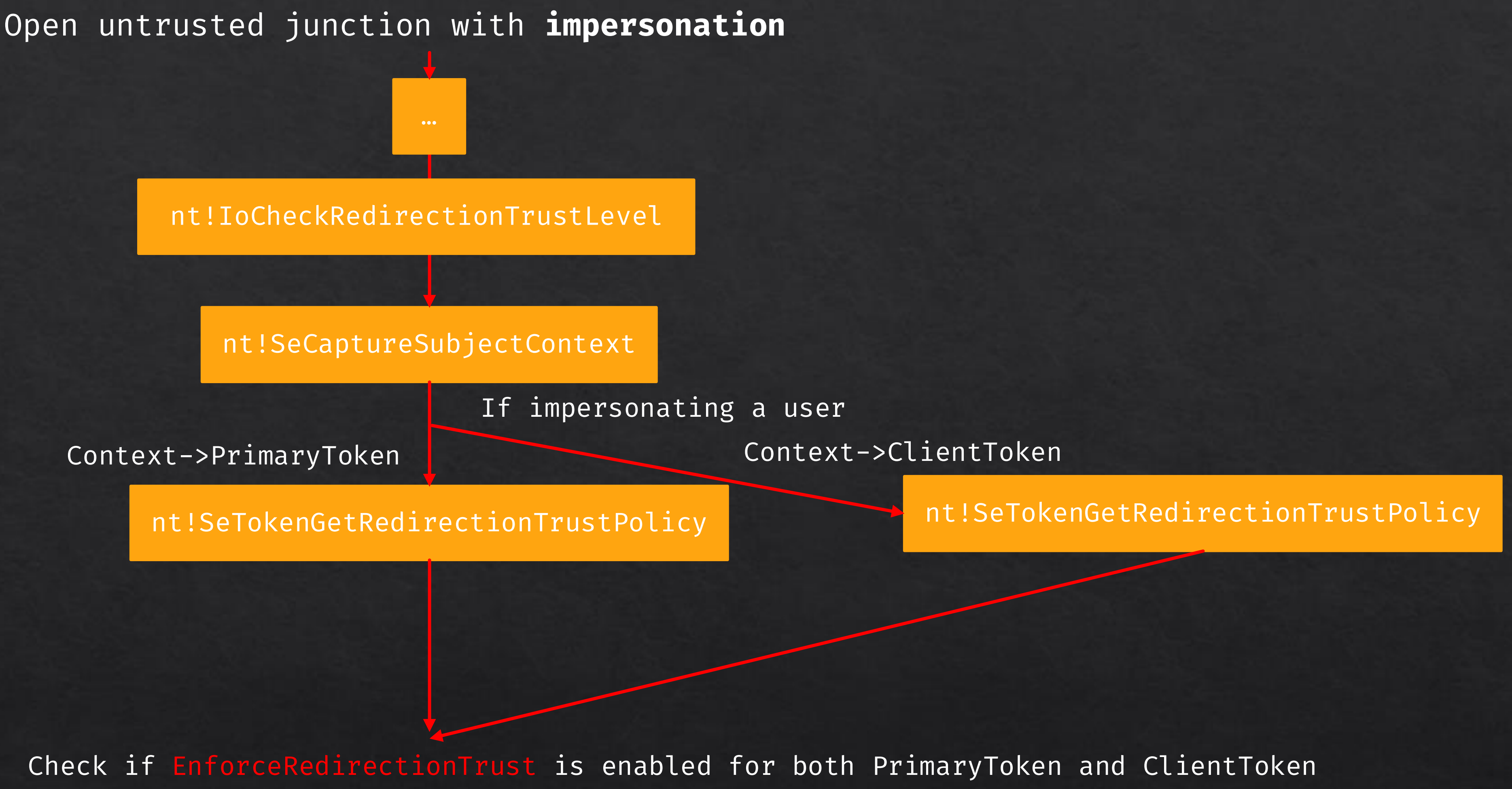
Install-Module -Name NtObjectManager



Time of...	Process Name	PID	Operation	Path	Result	Detail
4:03:13....	powershell.exe	8696	CreateFile	C:\FakeTemp\1.txt	REPARSE	Desired Access: R...
4:03:13....	powershell.exe	8696	CreateFile	C:\FakeTemp\1.txt	SUCCESS	Desired Access: R...
4:03:13....	powershell.exe	8696	QueryBasicInfor...	C:\FakeTemp\1.txt	SUCCESS	CreationTime: 10/1...
4:03:13....	powershell.exe	8696	CloseFile	C:\FakeTemp\1.txt	SUCCESS	
4:03:13....	powershell.exe	8696	CreateFile	C:\FakeTemp\1.txt	REPARSE	Desirec
4:03:13....	powershell.exe	8696	CreateFile	C:\FakeTemp\1.txt	SUCCESS	Desirec
4:03:13....	powershell.exe	8696	WriteFile	C:\FakeTemp\1.txt	SUCCESS	Offset:
4:03:13....	powershell.exe	8696	CloseFile	C:\FakeTemp\1.txt	SUCCESS	
4:03:13....	MsMpEng.exe	9156	CreateFileMap...	C:\FakeTemp\1.txt	FILE LOCKED WIT...	SyncTy
4:03:13....	MsMpEng.exe	9156	QueryStandardI...	C:\FakeTemp\1.txt	SUCCESS	Allocati

Desired Access: Read Attributes
Disposition: Open
Options: Open Reparse Point
Attributes: n/a
ShareMode: Read, Write, Delete
AllocationSize: n/a
Impersonating: DESKTOP-3VIAT4J\user2
OpenResult: Opened

Why mitigation fails?



Open untrusted junction with **impersonation**

…

nt!IoCheckRedirectionTrustLevel

nt!SeCaptureSubjectContext

If impersonating a user

Context->PrimaryToken

nt!SeTokenGetRedirectionTrustPolicy

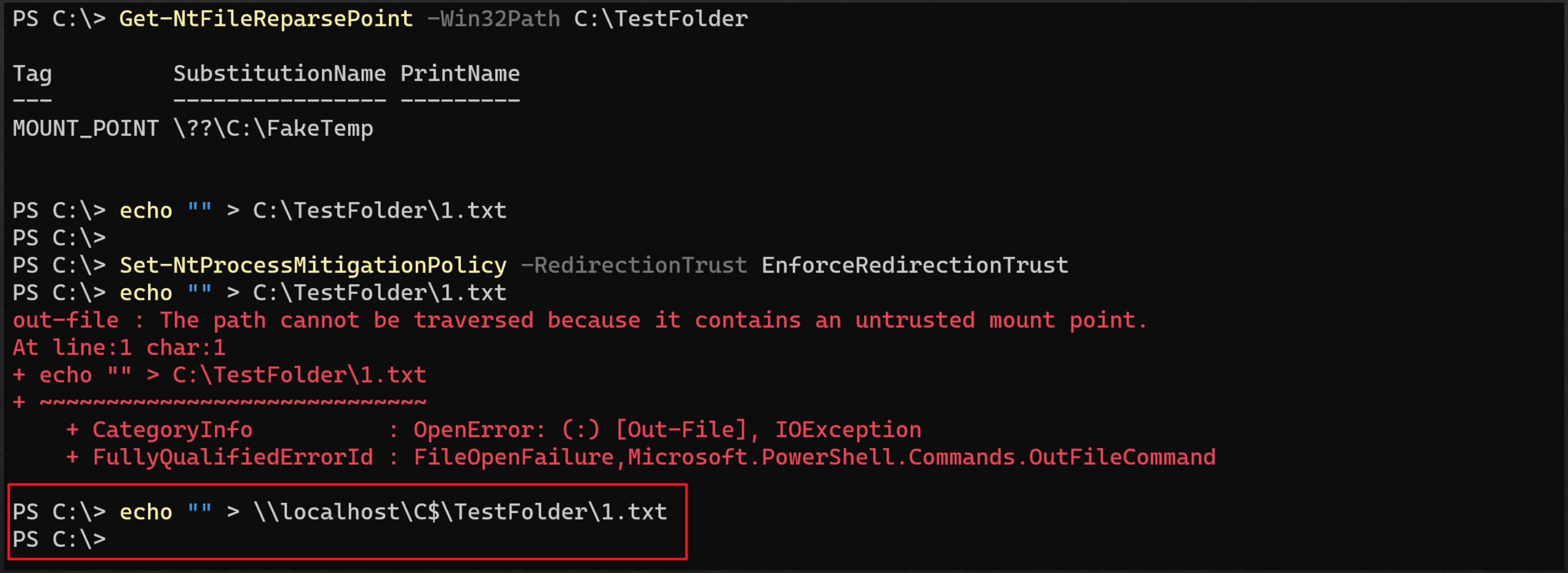
Context->ClientToken

nt!SeTokenGetRedirectionTrustPolicy

Check if EnforceRedirectionTrust is enabled for both PrimaryToken and ClientToken

- EnforceRedirectionTrust is enabled for PrimaryToken (Process Token)
- EnforceRedirectionTrust is disabled for ClientToken (Thread Impersonation Token)

Mitigation bypass when using UNC Path

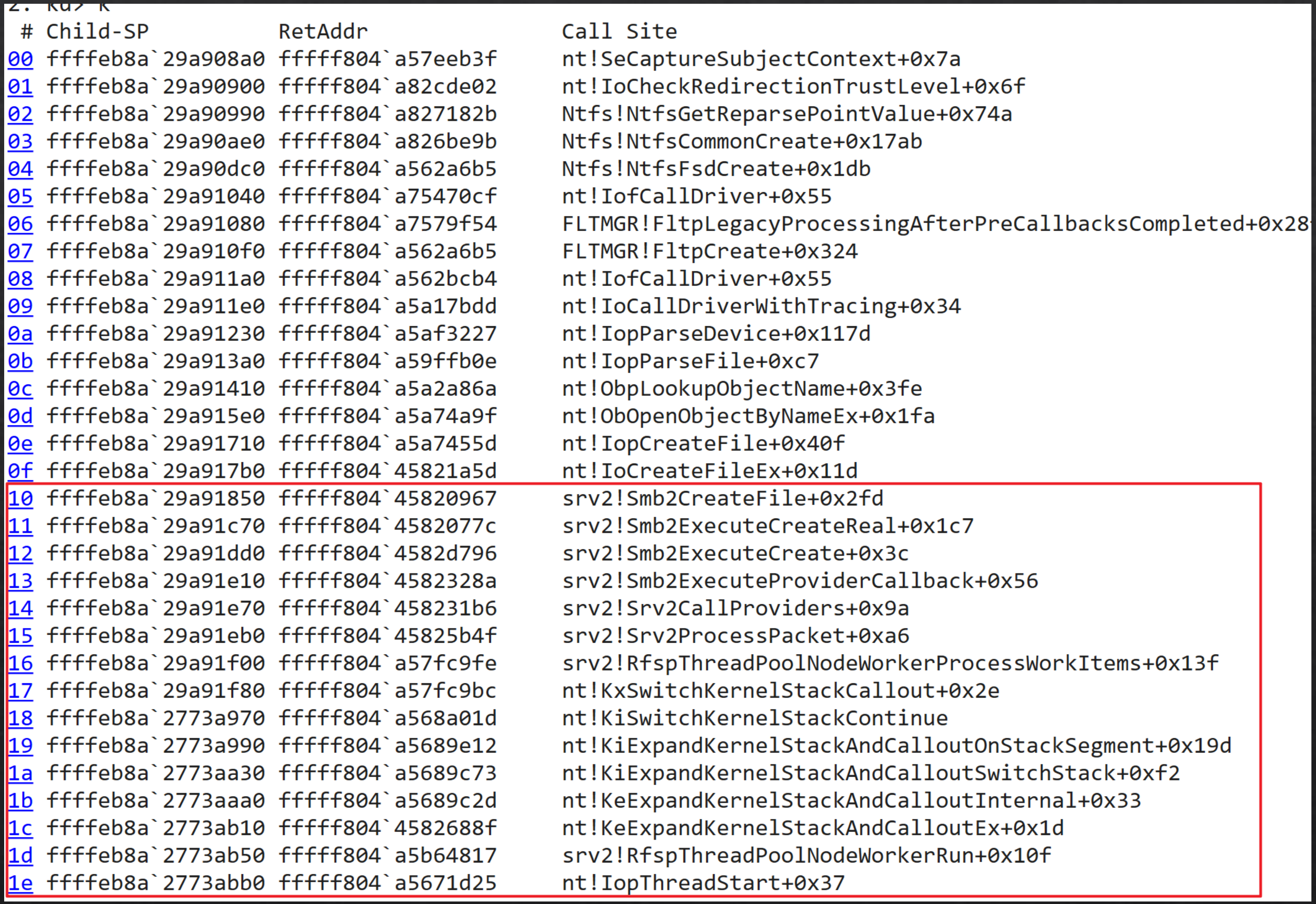


```
PS C:\> Get-NtFileReparsePoint -Win32Path C:\TestFolder

Tag         SubstitutionName PrintName
---         ---------------- ---------
MOUNT_POINT \??\C:\FakeTemp


PS C:\> echo "" > C:\TestFolder\1.txt
PS C:\>
PS C:\> Set-NtProcessMitigationPolicy -RedirectionTrust EnforceRedirectionTrust
PS C:\> echo "" > C:\TestFolder\1.txt
out-file : The path cannot be traversed because it contains an untrusted mount point.
At line:1 char:1
+ echo "" > C:\TestFolder\1.txt
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (:) [Out-File], IOException
    + FullyQualifiedErrorId : FileOpenFailure,Microsoft.PowerShell.Commands.OutFileCommand

PS C:\> echo "" > \\localhost\C$\TestFolder\1.txt
PS C:\>
```

```
Install-Module -Name NtObjectManager
```

Why mitigation fails?

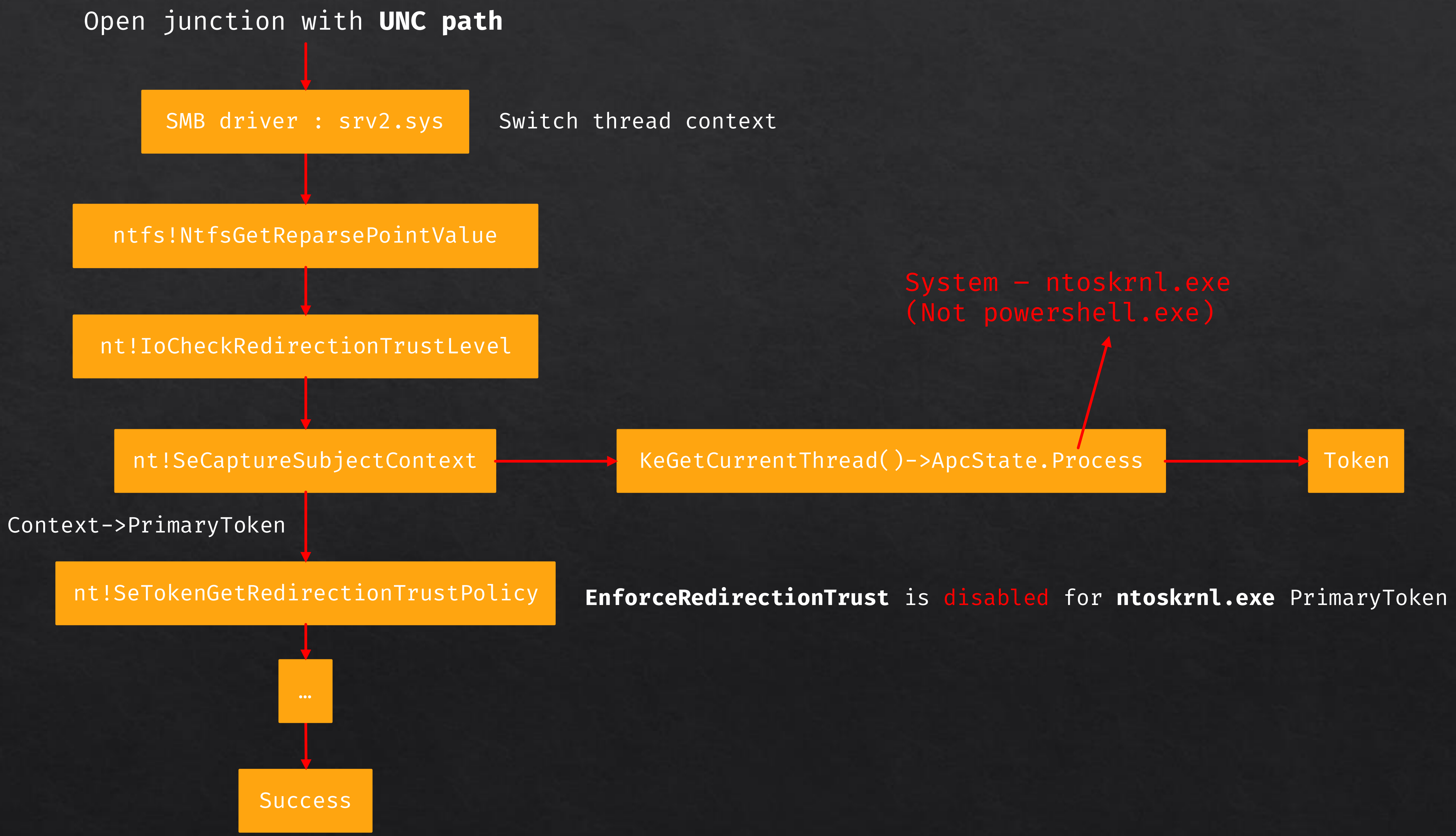
Call stack when open junction with **UNC path**



```
2: kd> k
 # Child-SP          RetAddr               Call Site
00 ffffeb8a`29a908a0 fffff804`a57eeb3f     nt!SeCaptureSubjectContext+0x7a
01 ffffeb8a`29a90900 fffff804`a82cde02     nt!IoCheckRedirectionTrustLevel+0x6f
02 ffffeb8a`29a90990 fffff804`a827182b     Ntfs!NtfsGetReparsePointValue+0x74a
03 ffffeb8a`29a90ae0 fffff804`a826be9b     Ntfs!NtfsCommonCreate+0x17ab
04 ffffeb8a`29a90dc0 fffff804`a562a6b5     Ntfs!NtfsFsdCreate+0x1db
05 ffffeb8a`29a91040 fffff804`a75470cf     nt!IofCallDriver+0x55
06 ffffeb8a`29a91080 fffff804`a7579f54     FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x28
07 ffffeb8a`29a910f0 fffff804`a562a6b5     FLTMGR!FltpCreate+0x324
08 ffffeb8a`29a911a0 fffff804`a562bcb4     nt!IofCallDriver+0x55
09 ffffeb8a`29a911e0 fffff804`a5a17bdd     nt!IoCallDriverWithTracing+0x34
0a ffffeb8a`29a91230 fffff804`a5af3227     nt!IopParseDevice+0x117d
0b ffffeb8a`29a913a0 fffff804`a59ffb0e     nt!IopParseFile+0xc7
0c ffffeb8a`29a91410 fffff804`a5a2a86a     nt!ObpLookupObjectName+0x3fe
0d ffffeb8a`29a915e0 fffff804`a5a74a9f     nt!ObOpenObjectByNameEx+0x1fa
0e ffffeb8a`29a91710 fffff804`a5a7455d     nt!IopCreateFile+0x40f
0f ffffeb8a`29a917b0 fffff804`45821a5d     nt!IoCreateFileEx+0x11d
10 ffffeb8a`29a91850 fffff804`45820967     srv2!Smb2CreateFile+0x2fd
11 ffffeb8a`29a91c70 fffff804`4582077c     srv2!Smb2ExecuteCreateReal+0x1c7
12 ffffeb8a`29a91dd0 fffff804`4582d796     srv2!Smb2ExecuteCreate+0x3c
13 ffffeb8a`29a91e10 fffff804`4582328a     srv2!Smb2ExecuteProviderCallback+0x56
14 ffffeb8a`29a91e70 fffff804`458231b6     srv2!Srv2CallProviders+0x9a
15 ffffeb8a`29a91eb0 fffff804`45825b4f     srv2!Srv2ProcessPacket+0xa6
16 ffffeb8a`29a91f00 fffff804`a57fc9fe     srv2!RfspThreadPoolNodeWorkerProcessWorkItems+0x13f
17 ffffeb8a`29a91f80 fffff804`a57fc9bc     nt!KxSwitchKernelStackCallout+0x2e
18 ffffeb8a`2773a970 fffff804`a568a01d     nt!KiSwitchKernelStackContinue
19 ffffeb8a`2773a990 fffff804`a5689e12     nt!KiExpandKernelStackAndCalloutOnStackSegment+0x19d
1a ffffeb8a`2773aa30 fffff804`a5689c73     nt!KiExpandKernelStackAndCalloutSwitchStack+0xf2
1b ffffeb8a`2773aaa0 fffff804`a5689c2d     nt!KeExpandKernelStackAndCalloutInternal+0x33
1c ffffeb8a`2773ab10 fffff804`4582688f     nt!KeExpandKernelStackAndCalloutEx+0x1d
1d ffffeb8a`2773ab50 fffff804`a5b64817     srv2!RfspThreadPoolNodeWorkerRun+0x10f
1e ffffeb8a`2773abb0 fffff804`a5671d25     nt!IopThreadStart+0x37
```

Switch thread context

Why mitigation fails?



Open junction with **UNC path**

SMB driver : srv2.sys

Switch thread context

ntfs!NtfsGetReparsePointValue

nt!IoCheckRedirectionTrustLevel

nt!SeCaptureSubjectContext

KeGetCurrentThread()->ApcState.Process

Token

System – ntoskrnl.exe
(Not powershell.exe)

Context->PrimaryToken

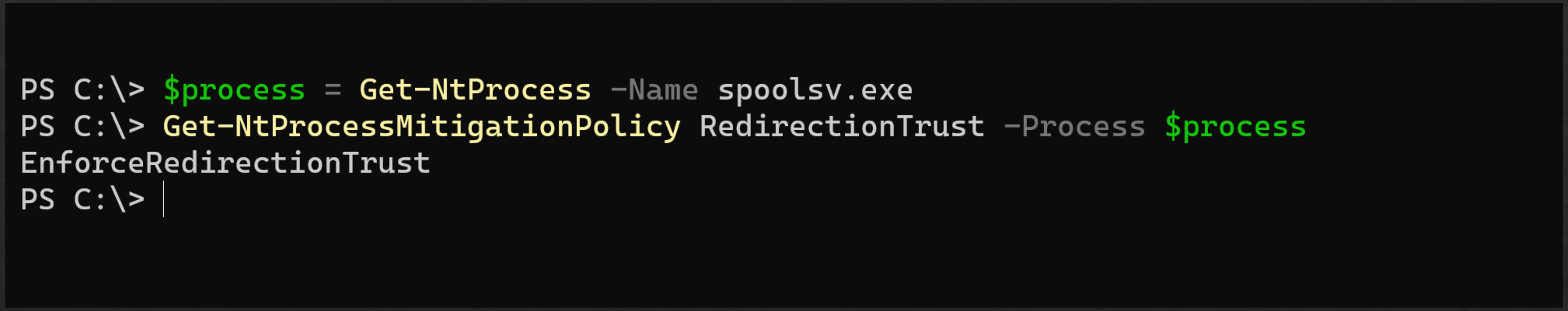
nt!SeTokenGetRedirectionTrustPolicy

EnforceRedirectionTrust is disabled for **ntoskrnl.exe** PrimaryToken

…

Success

Is the Mitigation Enabled for the process?



```
PS C:\> $process = Get-NtProcess -Name spoolsv.exe
PS C:\> Get-NtProcessMitigationPolicy RedirectionTrust -Process $process
EnforceRedirectionTrust
PS C:\>
```

```
Install-Module -Name NtObjectManager
```

Path Redirection Attacks with Symlink in the Future

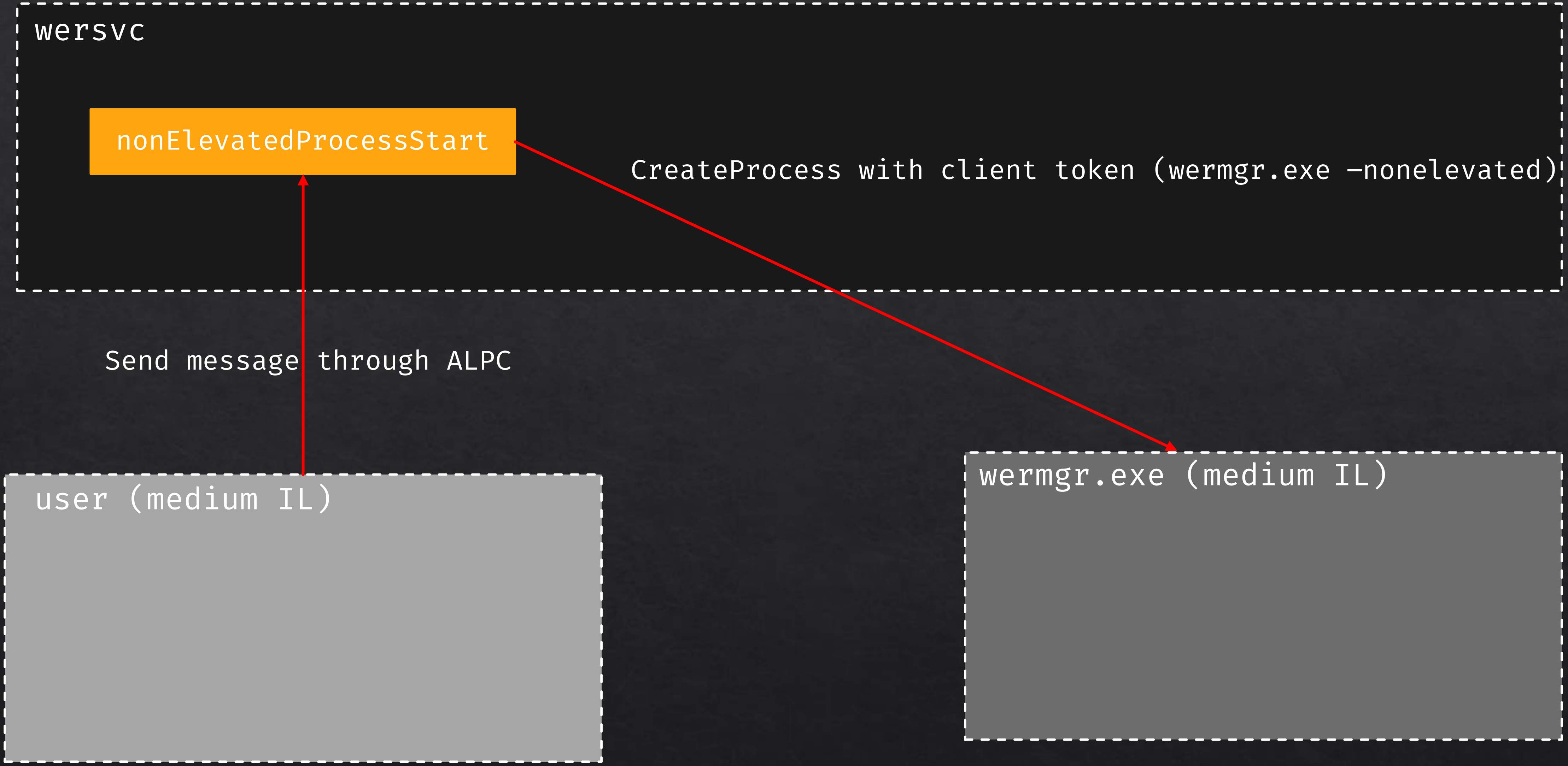
- More and more services enable mitigations
- Although the mitigation has weakness, but it can still block most of the attacks
- Path Redirection Attacks with Symlink are dying

Any different logical bugs in WER?

Part 3

Incorrect Handle Duplicate Lead to EOP

- CVE-2022-35795



wersvc

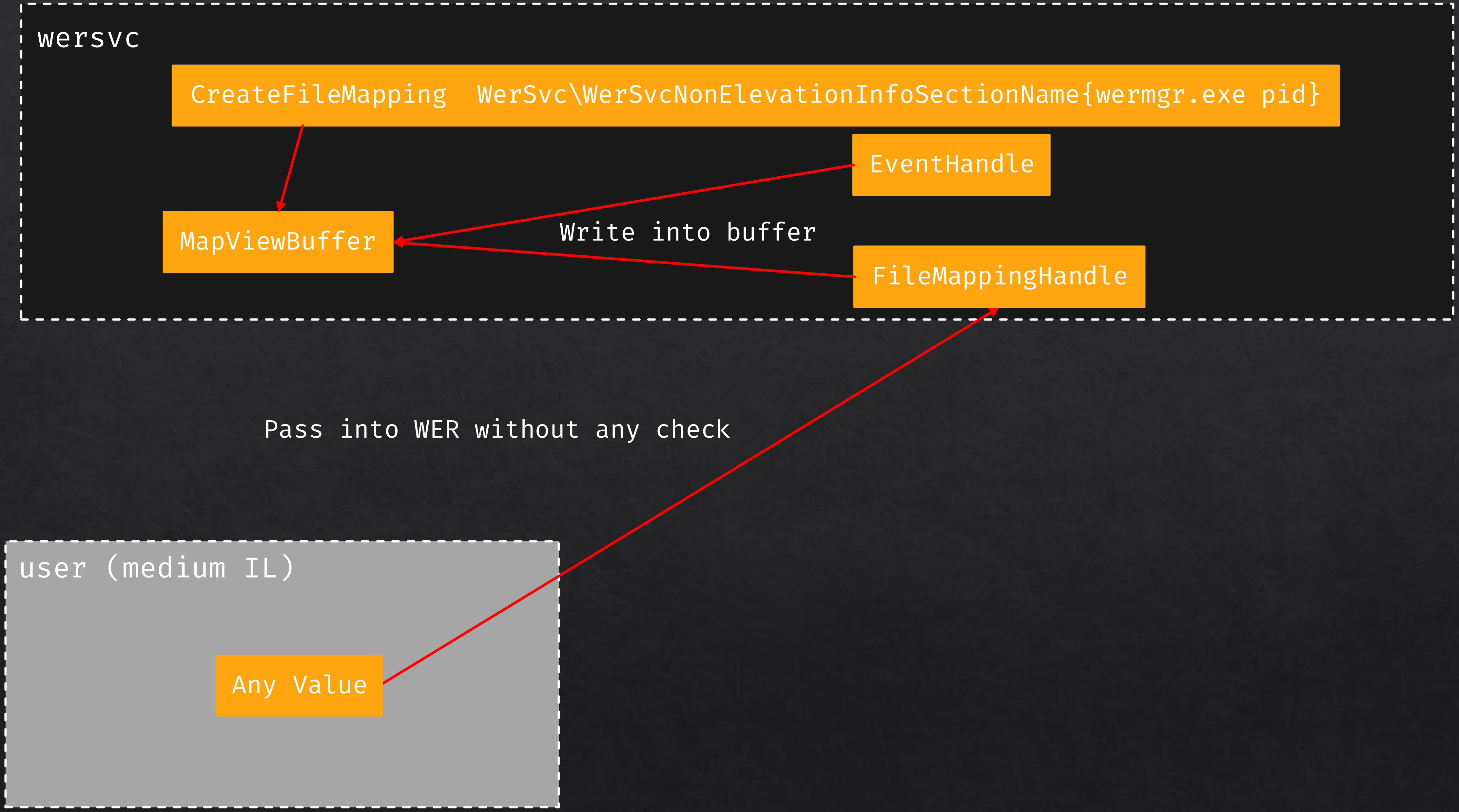
nonElevatedProcessStart

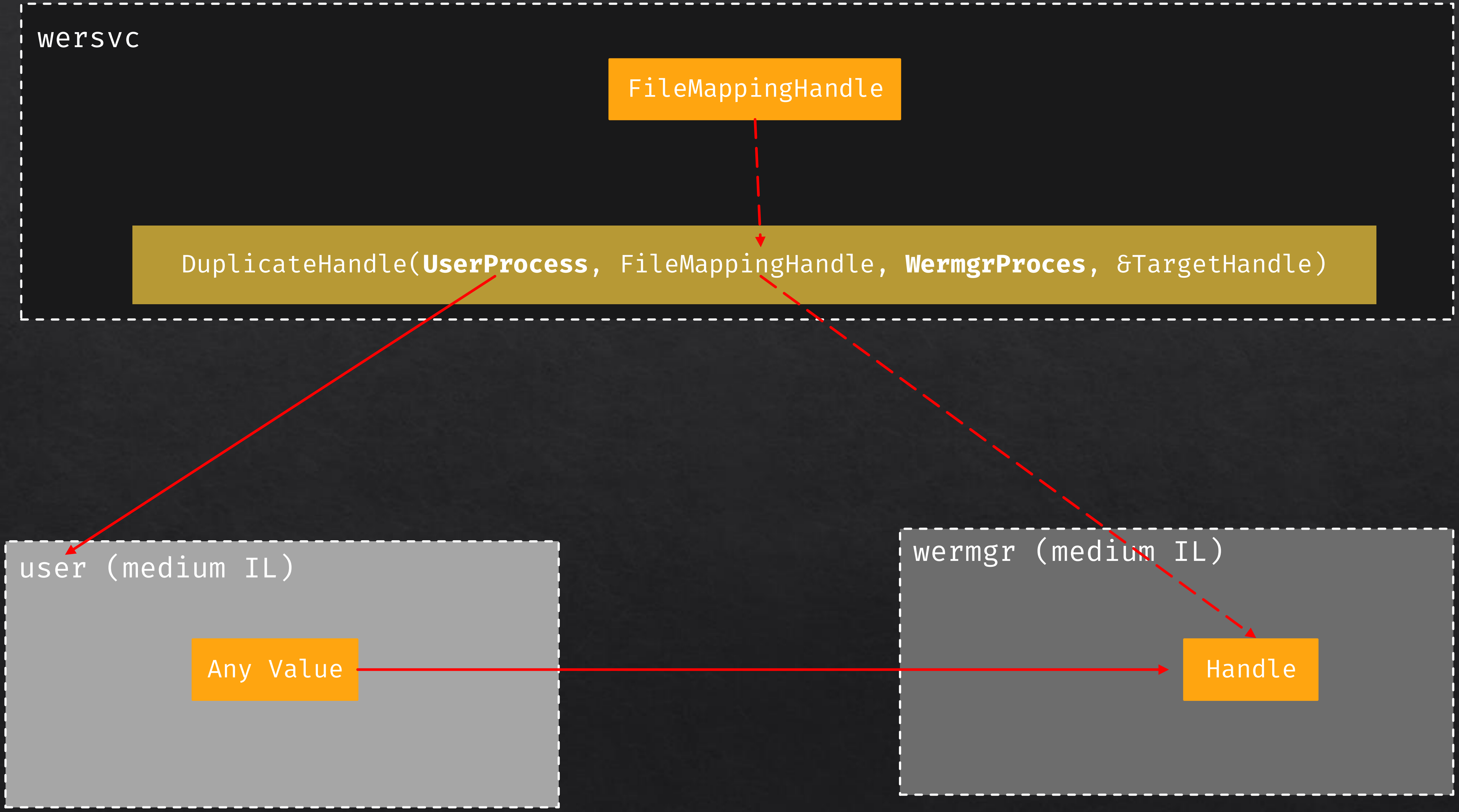
CreateProcess with client token (wermgr.exe -nonelevated)

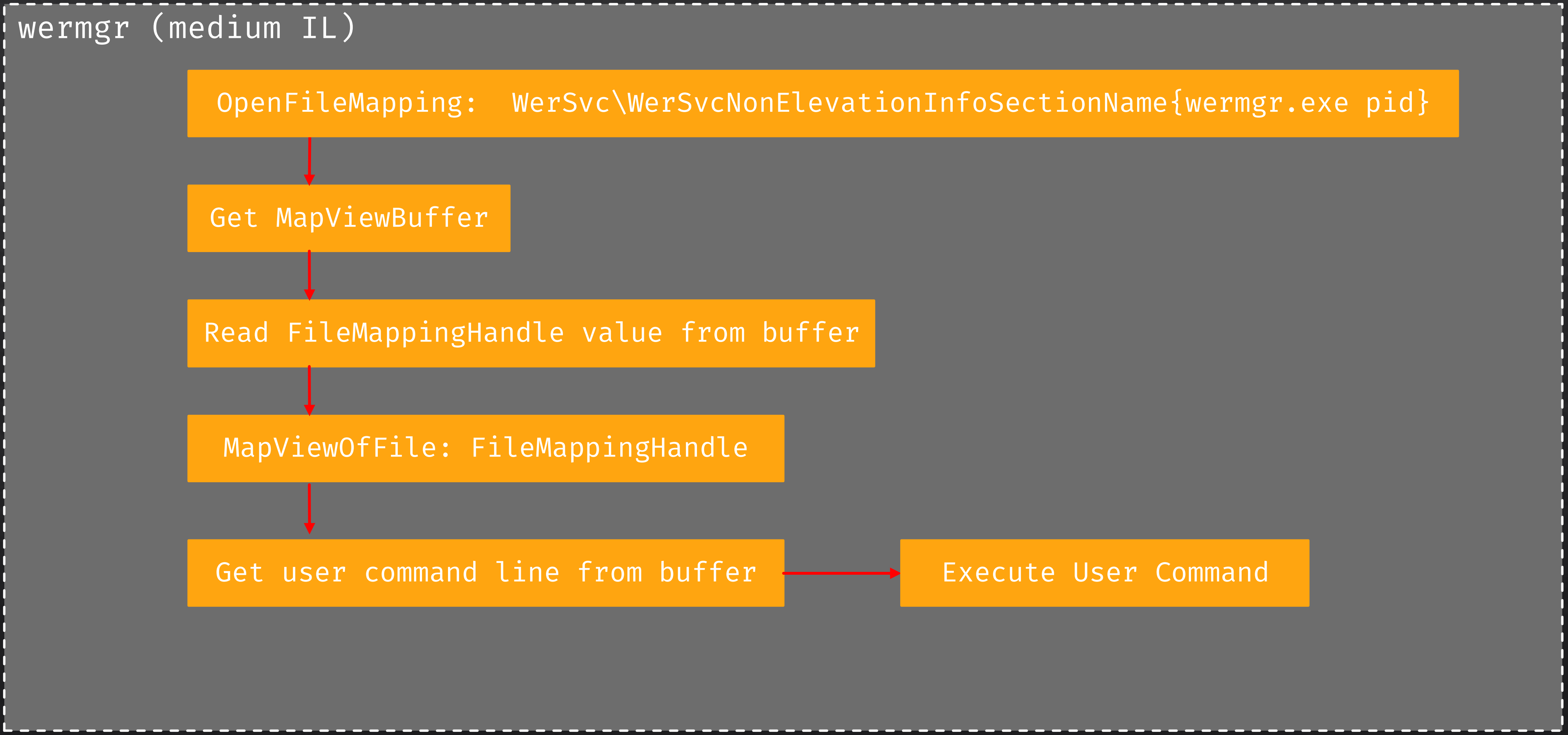
Send message through ALPC

user (medium IL)

wermgr.exe (medium IL)


wersvc
CreateFileMapping WerSvc\WerSvcNonElevationInfoSectionName{wermgr.exe pid}
EventHandle
MapViewBuffer
Write into buffer
FileMappingHandle
Pass into WER without any check
user (medium IL)
Any Value


wersvc
FileMappingHandle
DuplicateHandle(**UserProcess**, FileMappingHandle, **WermgrProces**, &TargetHandle)
user (medium IL)
Any Value
wermgr (medium IL)
Handle



wermgr (medium IL)

OpenFileMapping: WerSvc\WerSvcNonElevationInfoSectionName{wermgr.exe pid}

Get MapViewBuffer

Read FileMappingHandle value from buffer

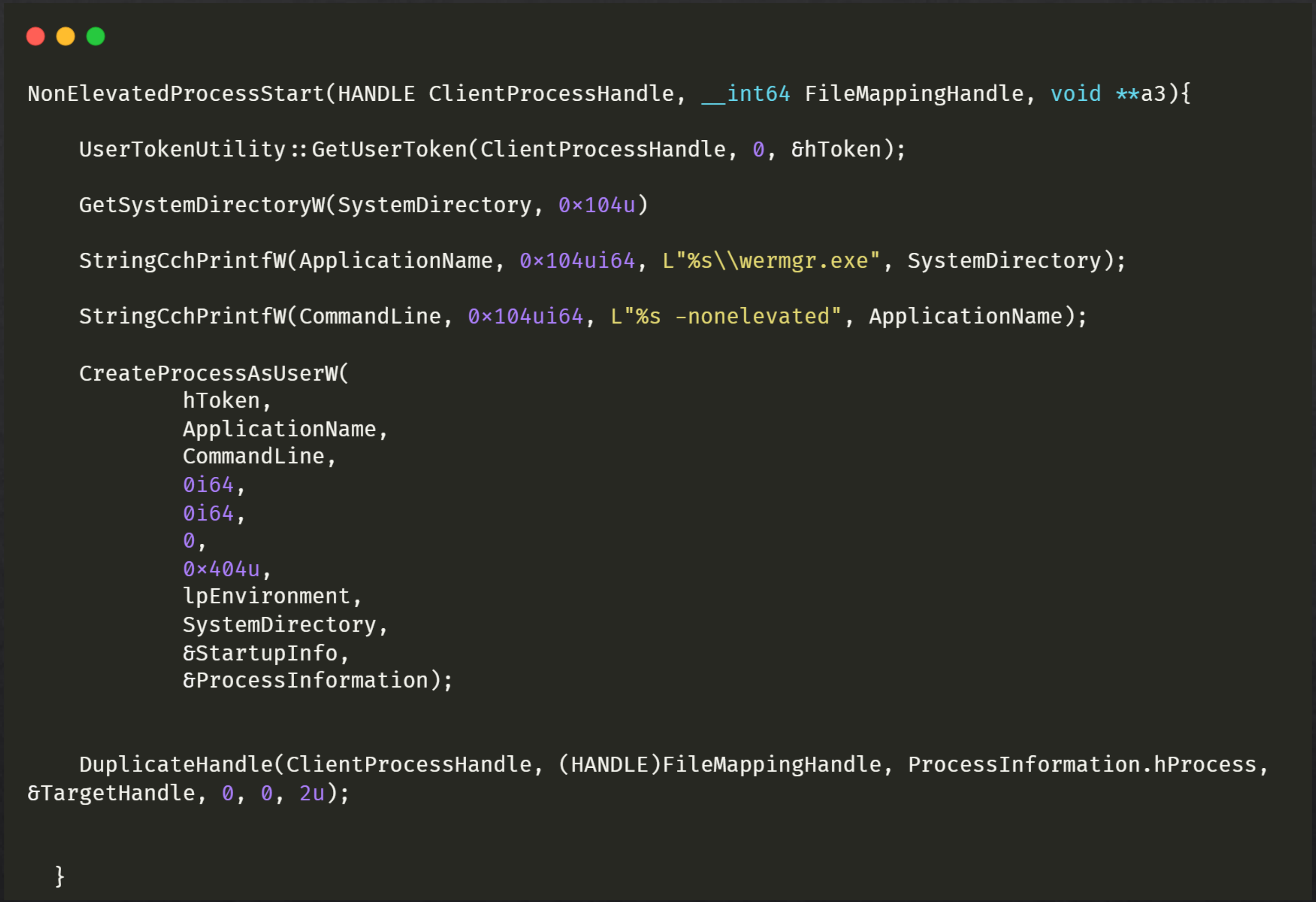
MapViewOfFile: FileMappingHandle

Get user command line from buffer

Execute User Command

Medium to Medium : No Security Boundary

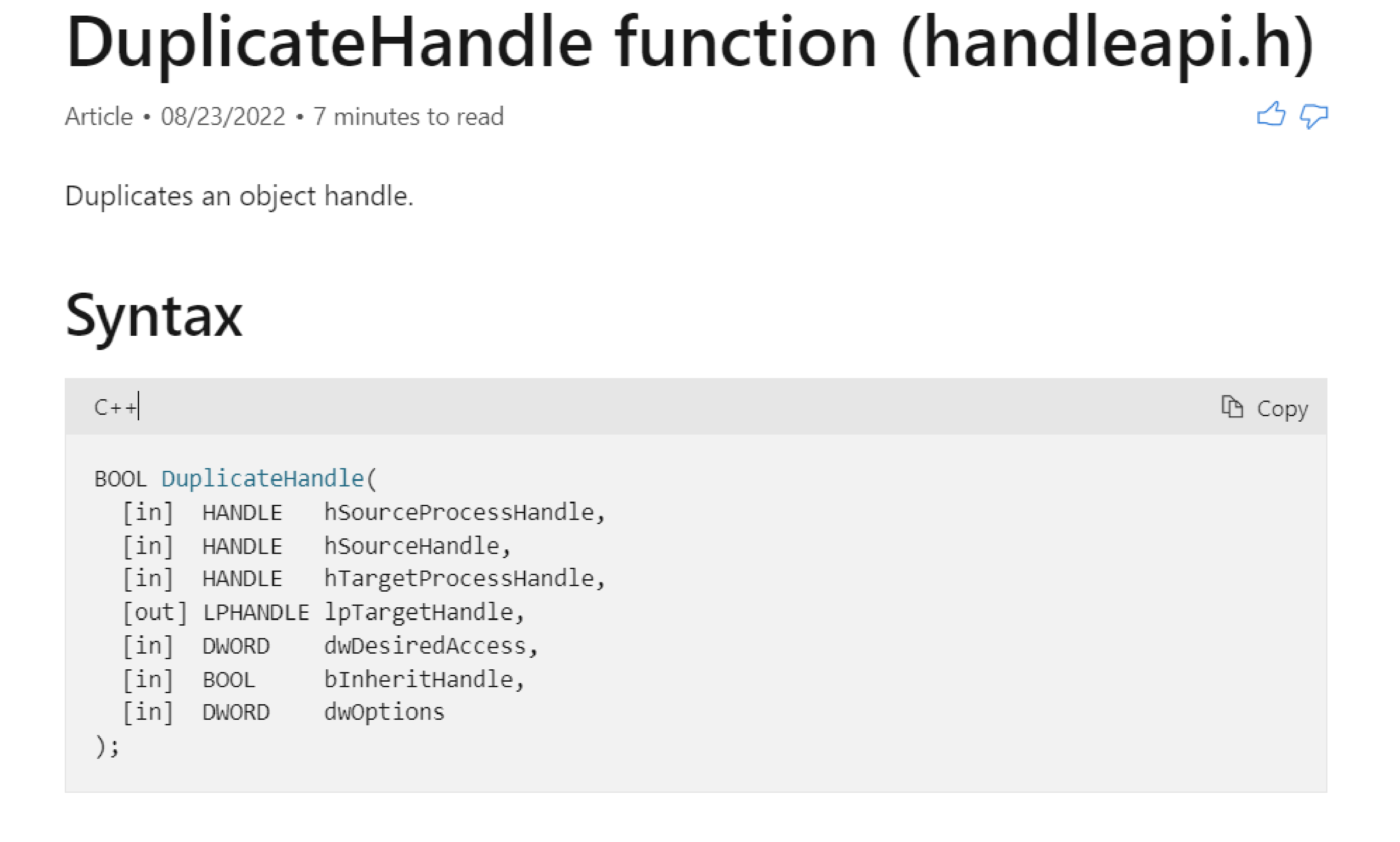
Vulnerable Code Snippet



```
NonElevatedProcessStart(HANDLE ClientProcessHandle, __int64 FileMappingHandle, void **a3){

    UserTokenUtility::GetUserToken(ClientProcessHandle, 0, &hToken);

    GetSystemDirectoryW(SystemDirectory, 0x104u)

    StringCchPrintfW(ApplicationName, 0x104ui64, L"%s\\wermgr.exe", SystemDirectory);

    StringCchPrintfW(CommandLine, 0x104ui64, L"%s -nonelevated", ApplicationName);

    CreateProcessAsUserW(
            hToken,
            ApplicationName,
            CommandLine,
            0i64,
            0i64,
            0,
            0x404u,
            lpEnvironment,
            SystemDirectory,
            &StartupInfo,
            &ProcessInformation);


    DuplicateHandle(ClientProcessHandle, (HANDLE)FileMappingHandle, ProcessInformation.hProcess,
&TargetHandle, 0, 0, 2u);


  }
```

- Create Non-Elevated process **wermgr.exe** with client token.
- Duplicate **FileMappingHandle** from client process to **wermgr.exe** process.
- No Check for **FileMappingHandle**

Weakness of DuplicateHandle



DuplicateHandle function (handleapi.h)

Article • 08/23/2022 • 7 minutes to read

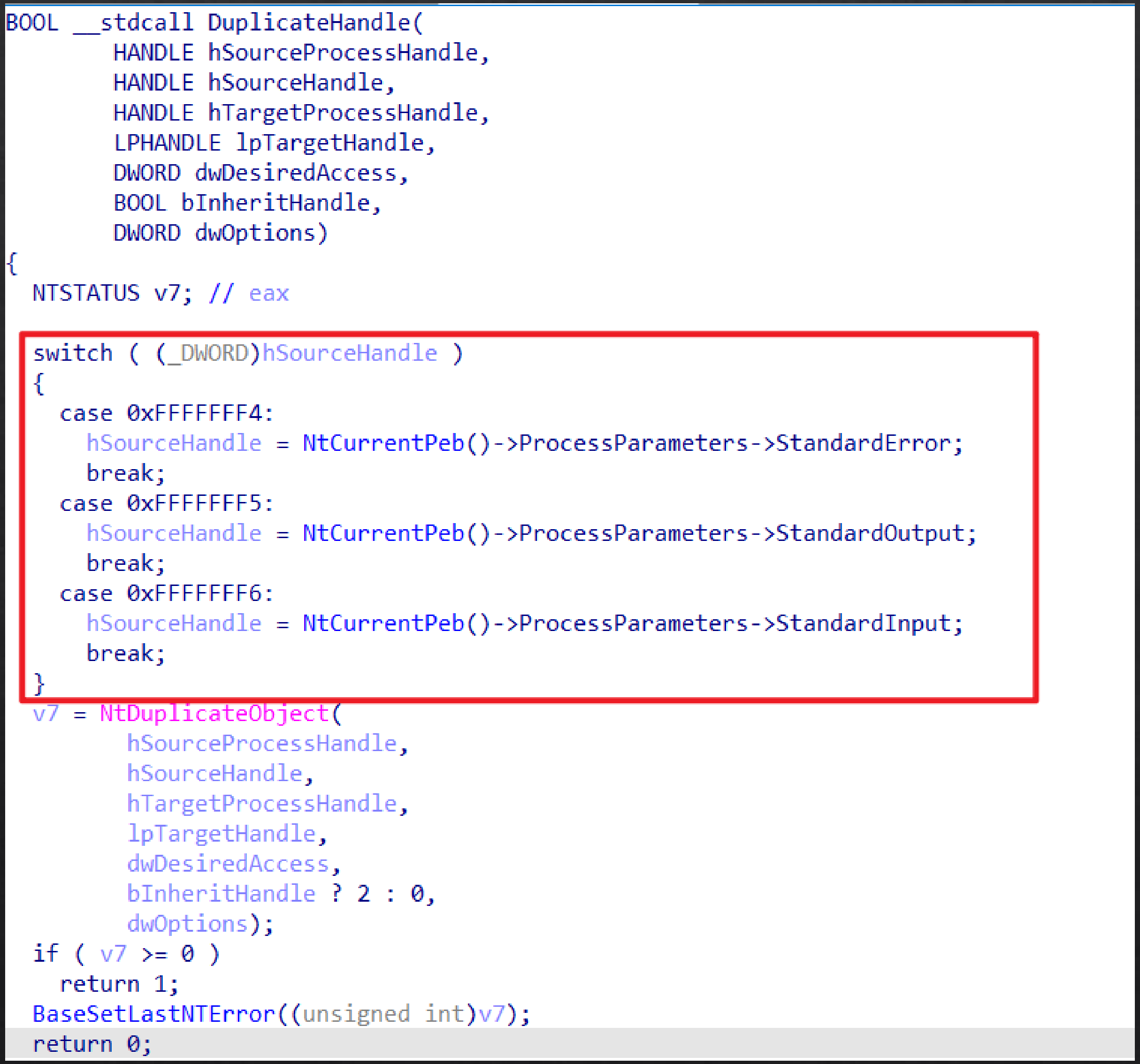
Duplicates an object handle.

Syntax

```
BOOL DuplicateHandle(
  [in]  HANDLE   hSourceProcessHandle,
  [in]  HANDLE   hSourceHandle,
  [in]  HANDLE   hTargetProcessHandle,
  [out] LPHANDLE lpTargetHandle,
  [in]  DWORD    dwDesiredAccess,
  [in]  BOOL     bInheritHandle,
  [in]  DWORD    dwOptions
);
```

Special Handle Value:

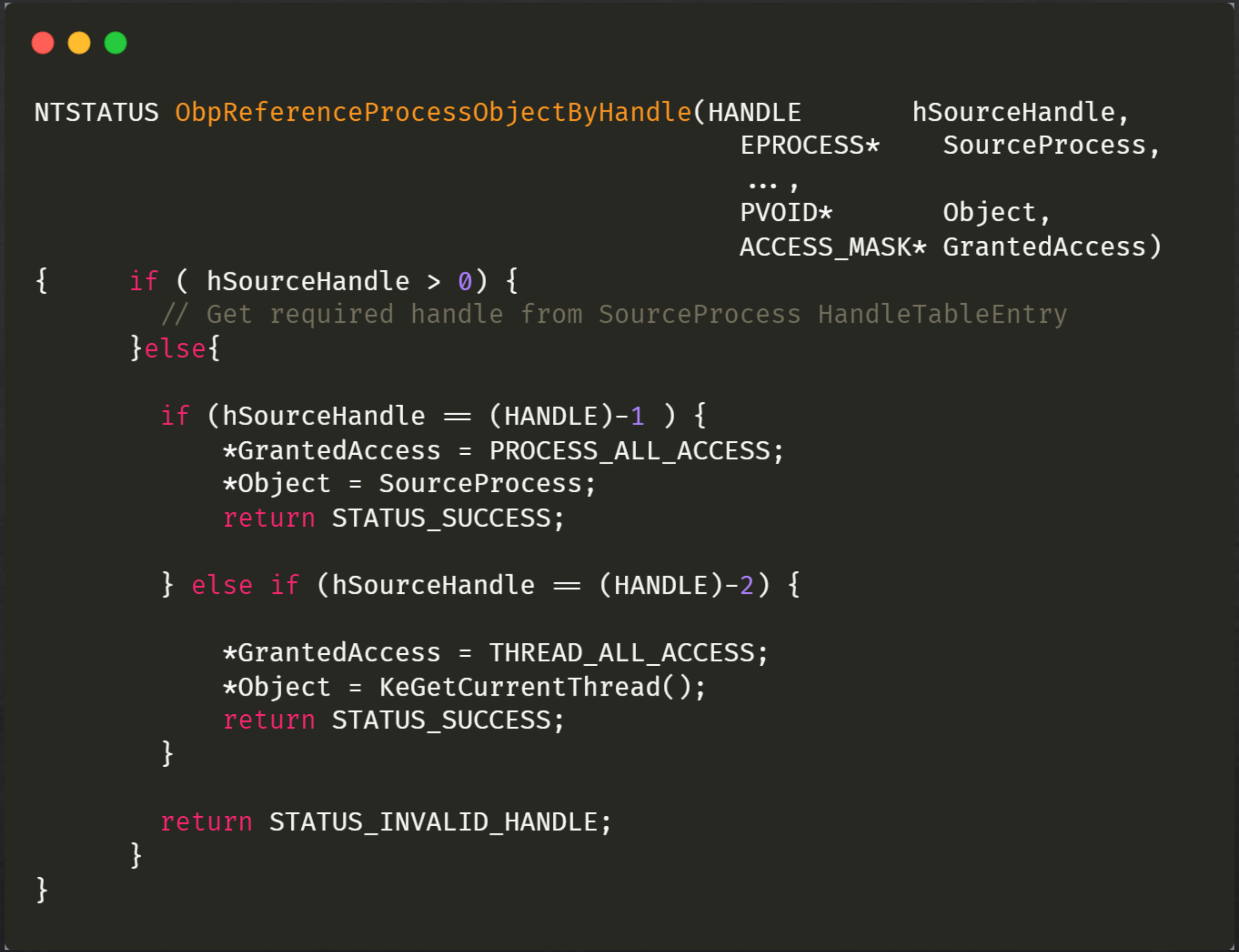
CurrentProcess	:	-1
CurrentThread	:	-2
StandardInput	:	-10
StandardOutput	:	-11
StandardError	:	-12



```
BOOL __stdcall DuplicateHandle(
        HANDLE hSourceProcessHandle,
        HANDLE hSourceHandle,
        HANDLE hTargetProcessHandle,
        LPHANDLE lpTargetHandle,
        DWORD dwDesiredAccess,
        BOOL bInheritHandle,
        DWORD dwOptions)
{
  NTSTATUS v7; // eax

  switch ( (_DWORD)hSourceHandle )
  {
    case 0xFFFFFFF4:
      hSourceHandle = NtCurrentPeb()->ProcessParameters->StandardError;
      break;
    case 0xFFFFFFF5:
      hSourceHandle = NtCurrentPeb()->ProcessParameters->StandardOutput;
      break;
    case 0xFFFFFFF6:
      hSourceHandle = NtCurrentPeb()->ProcessParameters->StandardInput;
      break;
  }
  v7 = NtDuplicateObject(
         hSourceProcessHandle,
         hSourceHandle,
         hTargetProcessHandle,
         lpTargetHandle,
         dwDesiredAccess,
         bInheritHandle ? 2 : 0,
         dwOptions);
  if ( v7 >= 0 )
    return 1;
  BaseSetLastNTError((unsigned int)v7);
  return 0;
```

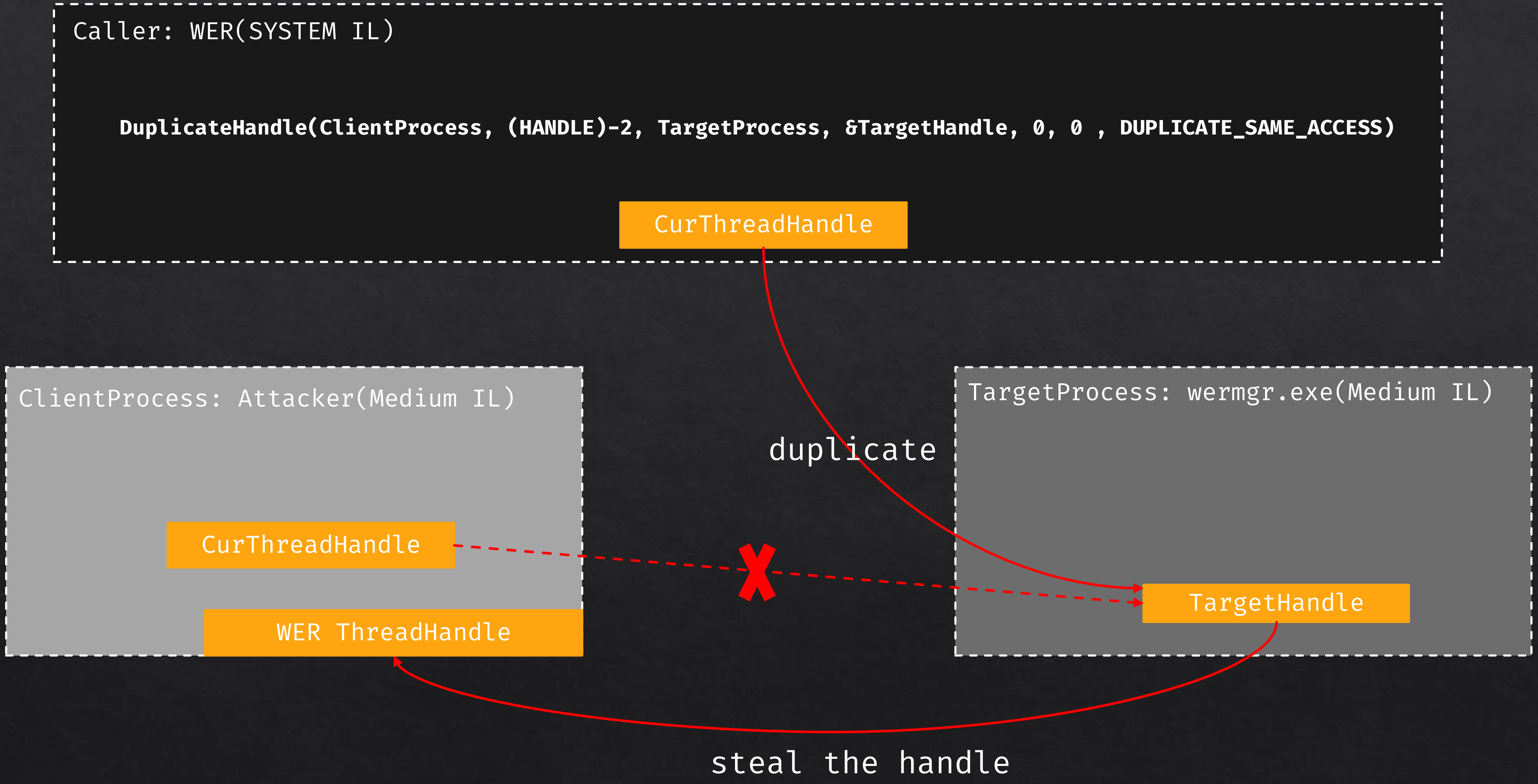
How to process special handle value?

- GetCurrentProcess() --> (HANDLE)-1
- GetCurrentThread() --> (HANDLE)-2



```
NTSTATUS ObpReferenceProcessObjectByHandle(HANDLE       hSourceHandle,
                                             EPROCESS*    SourceProcess,
                                             ... ,
                                             PVOID*       Object,
                                             ACCESS_MASK* GrantedAccess)
{     if ( hSourceHandle > 0) {
        // Get required handle from SourceProcess HandleTableEntry
      }else{

        if (hSourceHandle == (HANDLE)-1 ) {
            *GrantedAccess = PROCESS_ALL_ACCESS;
            *Object = SourceProcess;
            return STATUS_SUCCESS;

        } else if (hSourceHandle == (HANDLE)-2) {

            *GrantedAccess = THREAD_ALL_ACCESS;
            *Object = KeGetCurrentThread();
            return STATUS_SUCCESS;
        }

        return STATUS_INVALID_HANDLE;
      }
}
```

- Handle > 0 : Object from source process
- Handle == -1 : Source process object
- Handle == -2 : **Caller thread object**



```
DuplicateHandle(ClientProcess, (HANDLE)-2, TargetProcess, &TargetHandle, 0, 0 , DUPLICATE_SAME_ACCESS)
```

Exploit Leaked Handle

Process Handle with ***PROCESS_ALL_ACCESS***

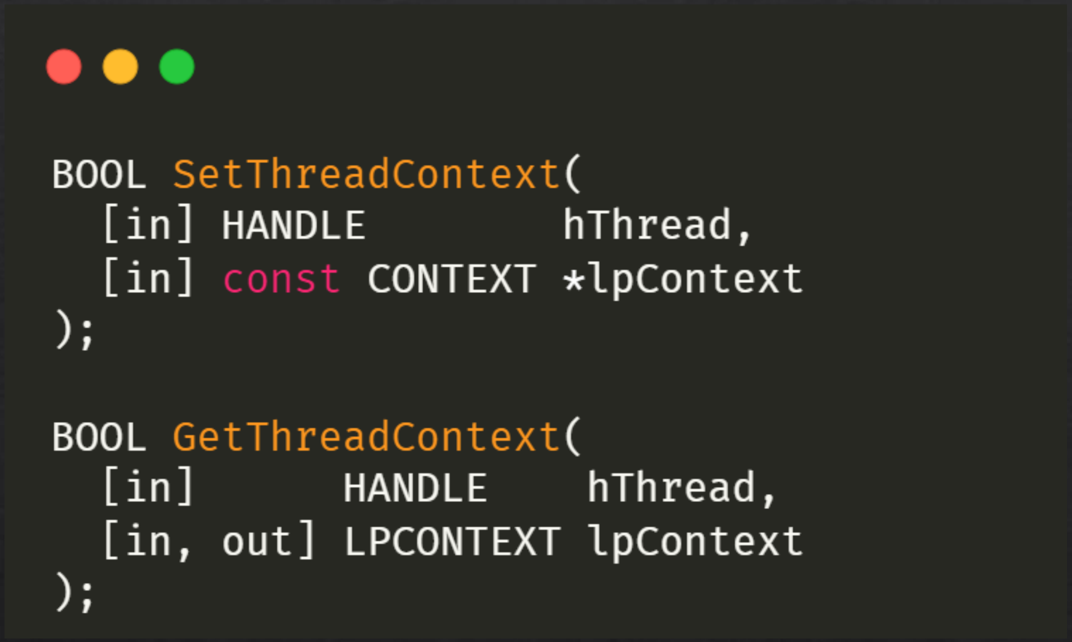
- VirtualAllocEx -> WriteProcessMemory -> CreateRemoteThread -> EOP

Thread Handle with ***THREAD_ALL_ACCESS***

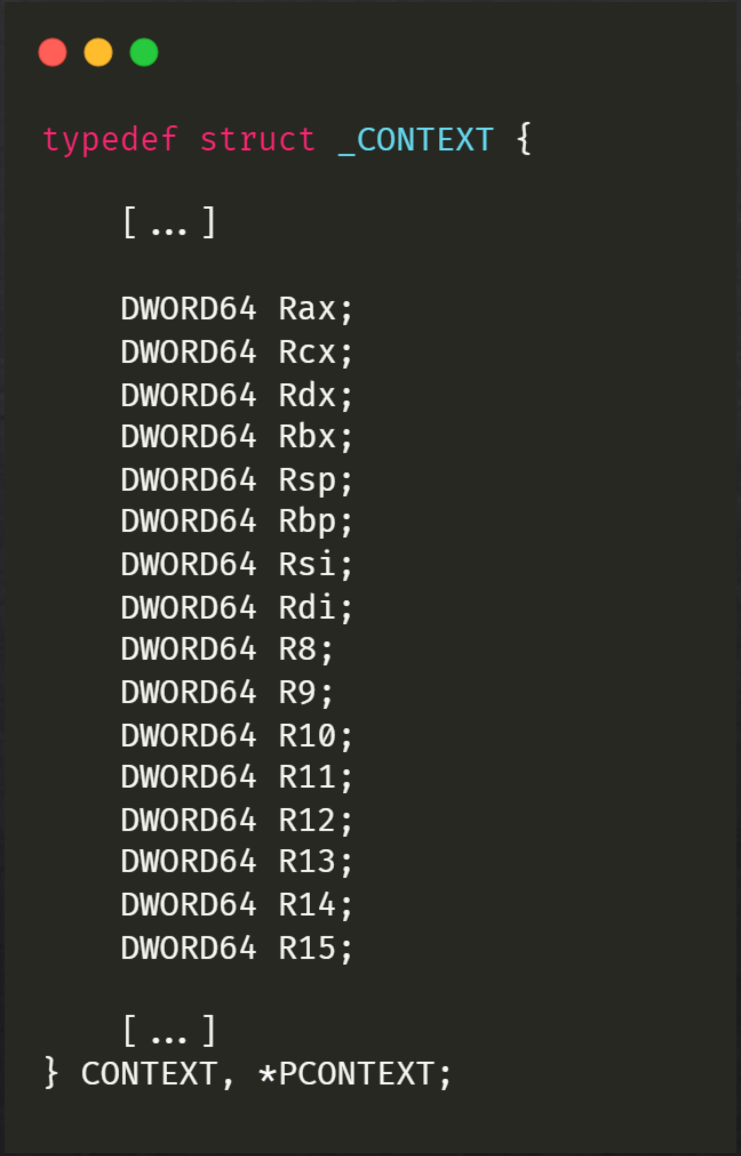
- No directly memory read/write
- No directly memory allocate

Exploit SYSTEM Thread Handle

Read/Write Thread Register

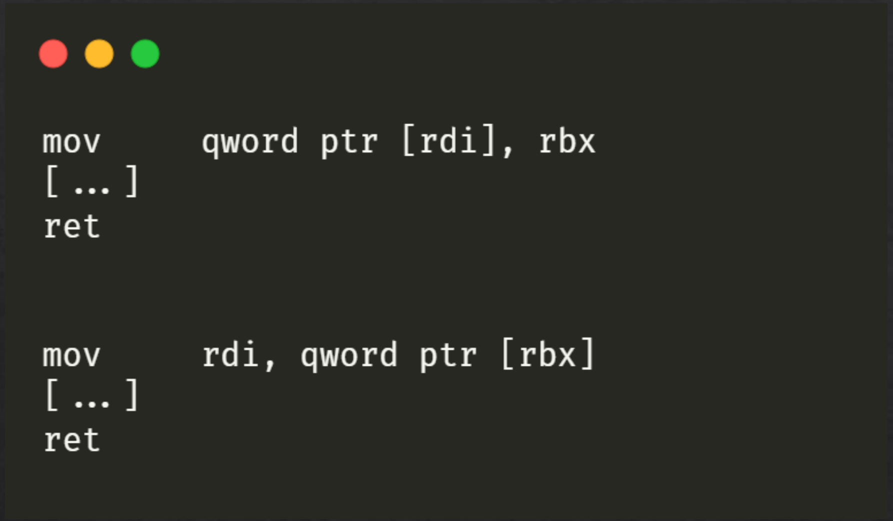


```
BOOL SetThreadContext(
  [in] HANDLE        hThread,
  [in] const CONTEXT *lpContext
);

BOOL GetThreadContext(
  [in]      HANDLE    hThread,
  [in, out] LPCONTEXT lpContext
);
```



```
typedef struct _CONTEXT {

    [ ... ]

    DWORD64 Rax;
    DWORD64 Rcx;
    DWORD64 Rdx;
    DWORD64 Rbx;
    DWORD64 Rsp;
    DWORD64 Rbp;
    DWORD64 Rsi;
    DWORD64 Rdi;
    DWORD64 R8;
    DWORD64 R9;
    DWORD64 R10;
    DWORD64 R11;
    DWORD64 R12;
    DWORD64 R13;
    DWORD64 R14;
    DWORD64 R15;

    [ ... ]
} CONTEXT, *PCONTEXT;
```

Find the ROP Gadget to get Read-What-Where and Write-What-Where primitive



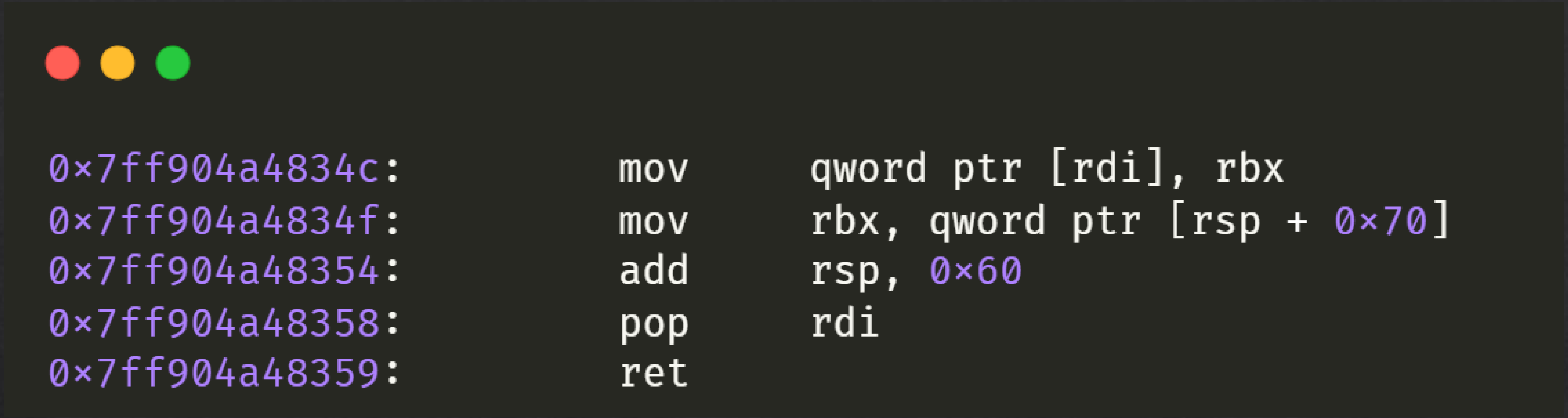
```
mov     qword ptr [rdi], rbx
[ … ]
ret
```

Arbitrary Write

```
mov     rdi, qword ptr [rbx]
[ … ]
ret
```

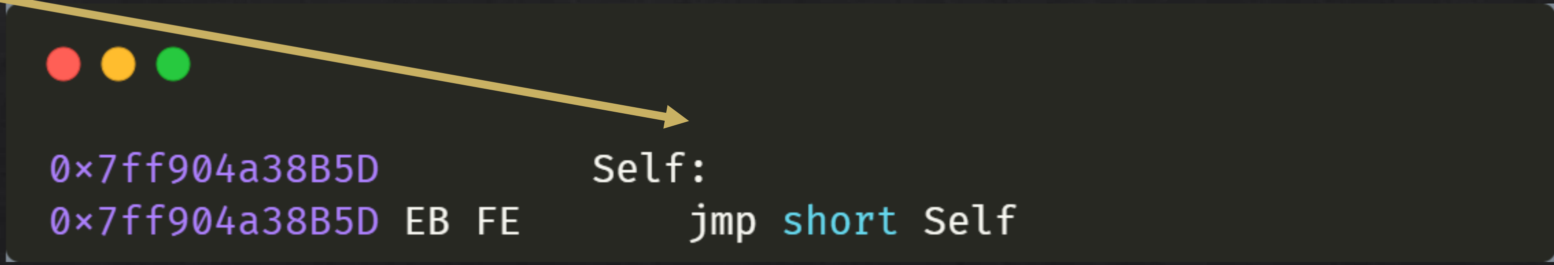
Arbitrary Read

Write the return address to the stack



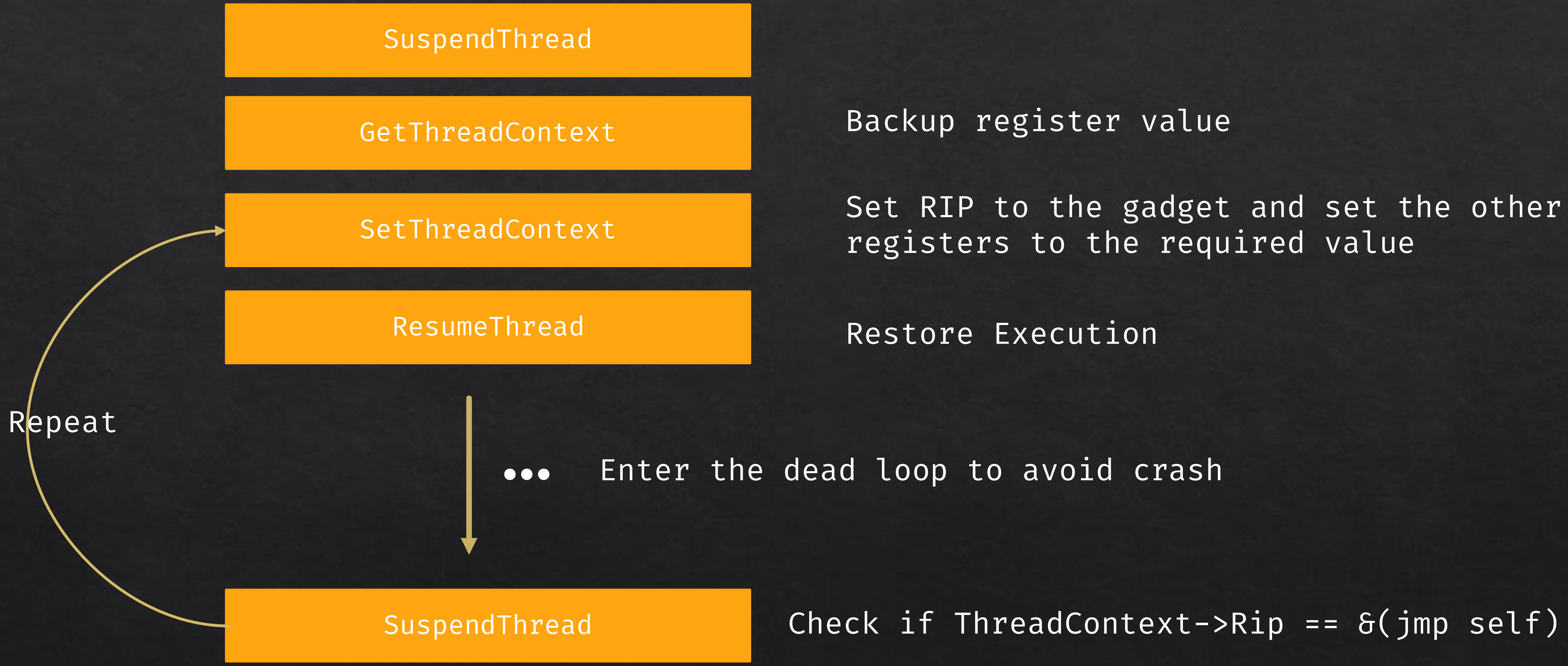
```
0x7ff904a4834c:         mov     qword ptr [rdi], rbx
0x7ff904a4834f:         mov     rbx, qword ptr [rsp + 0x70]
0x7ff904a48354:         add     rsp, 0x60
0x7ff904a48358:         pop     rdi
0x7ff904a48359:         ret
```

- rdi = rsp – 0x60 – 0x8
- rbx = return address



```
0x7ff904a38B5D              Self:
0x7ff904a38B5D EB FE            jmp short Self
```

Steps to complete a WWW primitive



SuspendThread

GetThreadContext — Backup register value

SetThreadContext — Set RIP to the gadget and set the other registers to the required value

ResumeThread — Restore Execution

••• Enter the dead loop to avoid crash

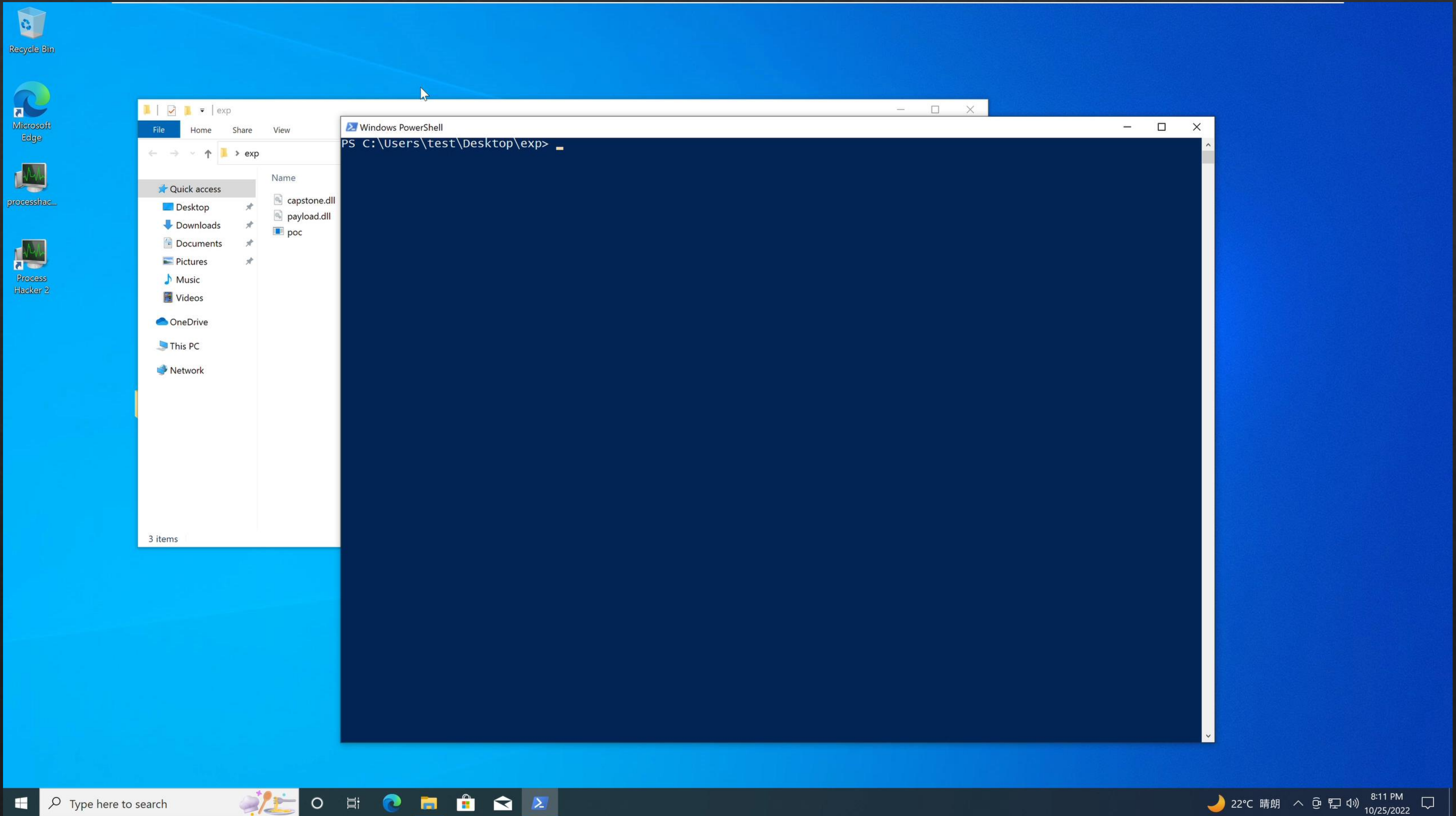
SuspendThread — Check if ThreadContext->Rip == &(jmp self)

Repeat

From WWW to code execution

- Write malicious DLL path into the stack
- Call LoadLibraryW with the path

Demo Time



```
PS C:\Users\test\Desktop\exp> _
```

Conclusion

- More and more services enable mitigations for path redirection attack
- Path Redirection Attacks with Symlink will become less and less
- It’s time to hunt for other type of logic bugs

Reference

- Exploiting Errors in Windows Error Reporting (BlueHatIL 2020) Gal De Leon
- Exploiting a Leaked Thread Handle - Project Zero James Forshaw

Thanks for listening!

Any Questions?