# Linux Remote Management Protocols [Windows]

lundi 23 septembre 2024 11:22 AM

Windows servers can be managed locally using Server Manager administration tasks on remote servers. Remote management is enabled by default starting with Windows Server 2016. Remote management is a component of the Windows hardware management features that manage server hardware locally and remotely. These features include a service that implements the **WS-Management protocol**, hardware diagnostics and control through baseboard management controllers, and a **COM API and script objects** that enable us to write applications that communicate remotely through the WS-Management protocol.

The main components used for remote management of Windows and Windows servers are the following:

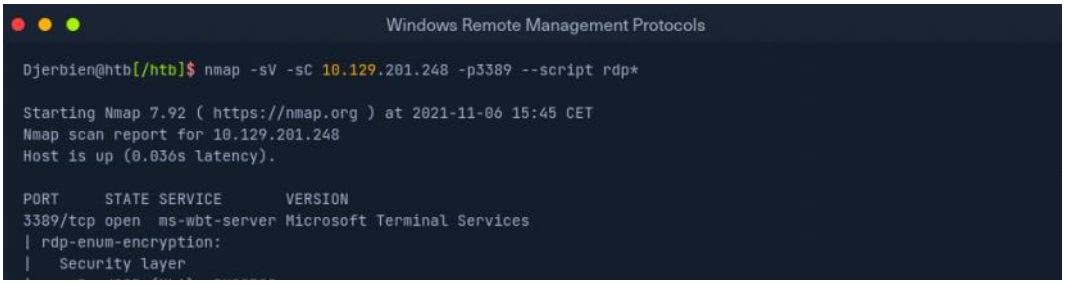
- Remote Desktop Protocol (**RDP**)
- Windows Remote Management (**WinRM**)
- Windows Management Instrumentation (**WMI**)

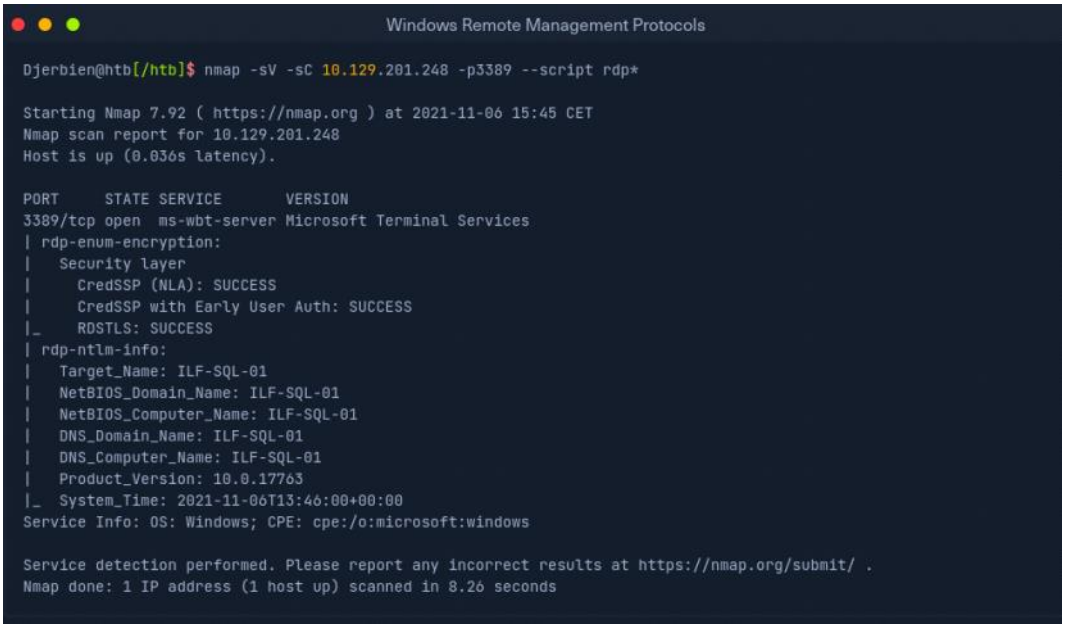
# RDP

| | |
|---|---|
| Abbreviation | The Remote Desktop Protocol (RDP)<br><br>This protocol allows display and control commands to be transmitted via the GUI encrypted over IP networks. |
| Port | RDP works at the application layer in the TCP/IP reference model, typically utilizing **TCP port 3389** as the transport protocol. However, the connectionless **UDP protocol can use port 3389** also for remote administration. |
| Firewall Consideration | For an RDP session to be established, both the network firewall and the firewall on the server must allow connections from the outside. If Network Address Translation (NAT) is used on the route between client and server, as is often the case with Internet connections, the remote computer needs the public IP address to reach the server. In addition, port forwarding must be set up on the NAT router in the direction of the server. |
| Encryption | RDP has **handled Transport Layer Security (TLS/SSL) since Windows Vista**, which means that all data, and especially the login process, is protected in the network by its good encryption. **However**, **many Windows systems do not insist on this but still accept inadequate encryption via RDP Security**. Nevertheless, even with this, an attacker is still far from being locked out because **the identity-providing certificates are merely self-signed by default**. This means that the client cannot distinguish a genuine certificate from a forged one and generates a certificate warning for the user. |
| Installation and connection | The Remote Desktop service **is installed by default on Windows server**s and does not require additional external applications. **This service can be activated using the Server Manager** and comes with the default setting to allow connections to the service **only to hosts with Network level authentication (NLA).** |

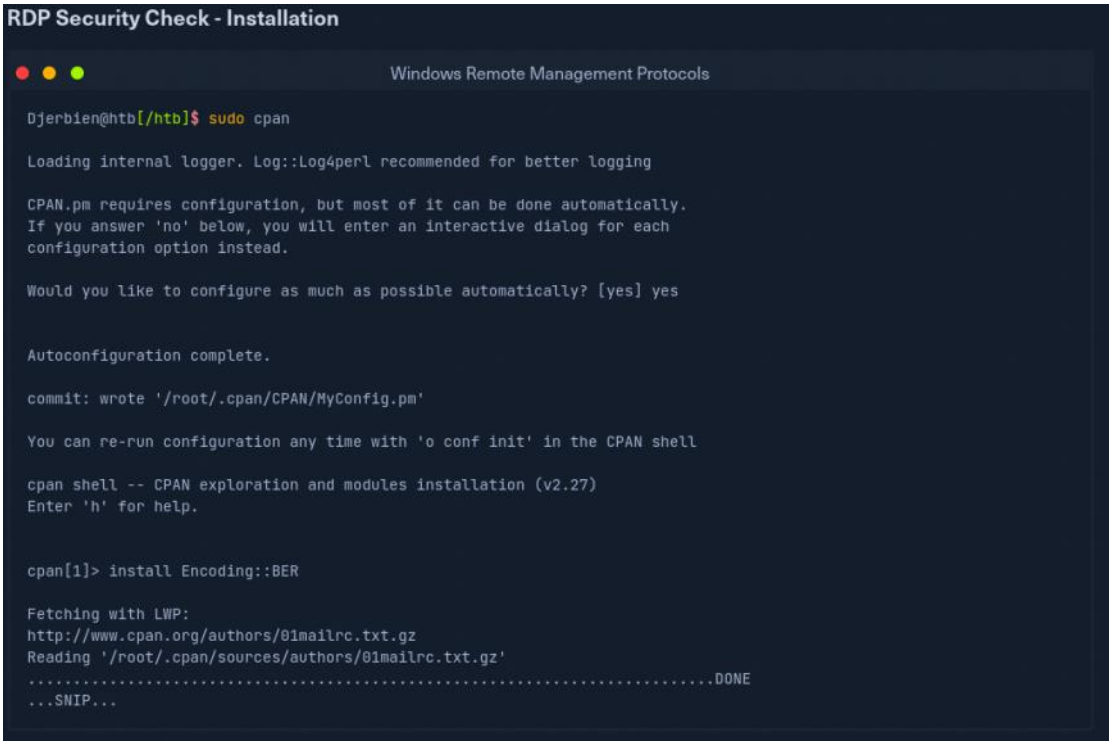
## Footprinting the Service

Scanning the RDP service can quickly give us a lot of information about the host. For example, **we can determine if NLA is enabled on the server or not**, **the product version**, and **the hostname.**






A Perl script named rdp-sec-check.pl has also been developed by Cisco CX Security Labs that **can unauthentically identify the security settings of RDP servers based on the handshakes.**




```
Djerbien@htb[/htb]$ git clone https://github.com/CiscoCXSecurity/rdp-sec-check.git && cd rdp-sec-check
Djerbien@htb[/htb]$ ./rdp-sec-check.pl 10.129.201.248
```

**RDP Security Check**

Windows Remote Management Protocols

```
Djerbien@htb[/htb]$ git clone https://github.com/CiscoCXSecurity/rdp-sec-check.git && cd rdp-sec-check
Djerbien@htb[/htb]$ ./rdp-sec-check.pl 10.129.201.248

Starting rdp-sec-check v0.9-beta ( http://labs.portcullis.co.uk/application/rdp-sec-check/ ) at Sun Nov  7 16:50:32 2021

[+] Scanning 1 hosts

Target:    10.129.201.248
IP:        10.129.201.248
Port:      3389

[+] Checking supported protocols

[-] Checking if RDP Security (PROTOCOL_RDP) is supported...Not supported - HYBRID_REQUIRED_BY_SERVER
[-] Checking if TLS Security (PROTOCOL_SSL) is supported...Not supported - HYBRID_REQUIRED_BY_SERVER
[-] Checking if CredSSP Security (PROTOCOL_HYBRID) is supported [uses NLA]...Supported

[+] Checking RDP Security Layer

[-] Checking RDP Security Layer with encryption ENCRYPTION_METHOD_NONE...Not supported
[-] Checking RDP Security Layer with encryption ENCRYPTION_METHOD_40BIT...Not supported
[-] Checking RDP Security Layer with encryption ENCRYPTION_METHOD_128BIT...Not supported
[-] Checking RDP Security Layer with encryption ENCRYPTION_METHOD_56BIT...Not supported
[-] Checking RDP Security Layer with encryption ENCRYPTION_METHOD_FIPS...Not supported

[+] Summary of protocol support

[-] 10.129.201.248:3389 supports PROTOCOL_SSL    : FALSE
[-] 10.129.201.248:3389 supports PROTOCOL_HYBRID: TRUE
[-] 10.129.201.248:3389 supports PROTOCOL_RDP    : FALSE

[+] Summary of RDP encryption support

[-] 10.129.201.248:3389 supports ENCRYPTION_METHOD_NONE   : FALSE
[-] 10.129.201.248:3389 supports ENCRYPTION_METHOD_40BIT  : FALSE
[-] 10.129.201.248:3389 supports ENCRYPTION_METHOD_128BIT : FALSE
```

Authentication and connection to such RDP servers can be made in several ways. For example, we can connect to RDP servers on Linux using **xfreerdp**, **rdesktop**, or **Remmina** **and interact with the GUI of the server accordingly.**

**Initiate an RDP Session**

Windows Remote Management Protocols

```
Djerbien@htb[/htb]$ xfreerdp /u:cry0l1t3 /p:"P455w0rd!" /v:10.129.201.248

[16:37:47:135] [95319:95320] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[16:37:47:135] [95319:95320] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[16:37:47:135] [95319:95320] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[16:37:47:135] [95319:95320] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
[16:37:47:447] [95319:95320] [INFO][com.freerdp.primitives] - primitives autodetect, using optimized
[16:37:47:453] [95319:95320] [INFO][com.freerdp.core] - freerdp_tcp_is_hostname_resolvable:freerdp_set_last_error_ex resetting error
[16:37:47:453] [95319:95320] [INFO][com.freerdp.core] - freerdp_tcp_connect:freerdp_set_last_error_ex resetting error state
[16:37:47:523] [95319:95320] [INFO][com.freerdp.crypto] - creating directory /home/cry0l1t3/.config/freerdp
[16:37:47:523] [95319:95320] [INFO][com.freerdp.crypto] - creating directory [/home/cry0l1t3/.config/freerdp/certs]
[16:37:47:523] [95319:95320] [INFO][com.freerdp.crypto] - created directory [/home/cry0l1t3/.config/freerdp/server]
[16:37:47:599] [95319:95320] [WARN][com.freerdp.crypto] - Certificate verification failure 'self signed certificate (18)' at stack p
[16:37:47:599] [95319:95320] [WARN][com.freerdp.crypto] - CN = ILF-SQL-01
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - @           WARNING: CERTIFICATE NAME MISMATCH!           @
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - The hostname used for this connection (10.129.201.248:3389)
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - does not match the name given in the certificate:
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - Common Name (CN):
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] -      ILF-SQL-01
[16:37:47:600] [95319:95320] [ERROR][com.freerdp.crypto] - A valid certificate for the wrong name should NOT be trusted!
Certificate details for 10.129.201.248:3389 (RDP-Server):
        Common Name: ILF-SQL-01
        Subject:     CN = ILF-SQL-01
        Issuer:      CN = ILF-SQL-01
        Thumbprint:  b7:5f:00:ca:91:00:0a:29:0c:b5:14:21:f3:b0:ca:9e:af:8c:62:d6:dc:f9:50:ec:ac:06:38:1f:c5:d6:a9:39
The above X.509 certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.


Do you trust the above certificate? (Y/T/N) y

[16:37:48:801] [95319:95320] [INFO][com.winpr.sspi.NTLM] - VERSION ={
[16:37:48:801] [95319:95320] [INFO][com.winpr.sspi.NTLM] -      ProductMajorVersion: 6
[16:37:48:801] [95319:95320] [INFO][com.winpr.sspi.NTLM] -      ProductMinorVersion: 1
[16:37:48:801] [95319:95320] [INFO][com.winpr.sspi.NTLM] -      ProductBuild: 7601
[16:37:48:801] [95319:95320] [INFO][com.winpr.sspi.NTLM] -      Reserved: 0x000000
```

After successful authentication, a new window will appear with access to the server's desktop to which we have connected.

# WinRM

**The Windows Remote Management (WinRM)**

| | |
|---|---|
| SOAP | WinRM uses the **Simple Object Access Protocol (SOAP)** to establish connections to remote hosts and their applications. Therefore, **WinRM must be explicitly enabled and configured starting with Windows 10** |
| Ports | **WinRM** relies on **TCP** ports **5985** and **5986** for communication, with the last **port 5986 using HTTPS**, as ports 80 and 443 were previously used for this task. However, since port 80 was mainly blocked for security reasons, **the newer ports 5985 and 5986 are used today.** |

| | |
|---|---|
| WinRS | Another component that fits WinRM for administration is **Windows Remote Shell (WinRS)**, which lets us execute arbitrary commands on the remote system. **The program is even included on Windows 7 by default**. Thus, with WinRM, it is possible to execute a remote command on another server.<br><br>Services like remote sessions using PowerShell and event log merging require WinRM. **It is enabled by default starting with the Windows Server 2012 version**, but it must first be configured for older server versions and clients, and the necessary firewall exceptions created. |

# Footprinting the Service

As we already know, WinRM uses TCP ports 5985 (HTTP) and 5986 (HTTPS) by default, which we can scan using Nmap. However, often we will see that only HTTP (TCP 5985) is used instead of HTTPS (TCP 5986).

**Nmap WinRM**

Windows Remote Management Protocols

```
Djerbien@htb[/htb]$ nmap -sV -sC 10.129.201.248 -p5985,5986 --disable-arp-ping -n

Starting Nmap 7.92 ( https://nmap.org ) at 2021-11-06 16:31 CET
Nmap scan report for 10.129.201.248
Host is up (0.030s 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
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.34 seconds
```

If we want to find out whether one or more remote servers can be reached via WinRM, we can easily do this with the help of **PowerShell**. The **Test-WsMan** cmdlet is responsible for this, and the host's name in question is passed to it.

**In Linux-based environments**, we can use the tool called **evil-winrm**, another penetration testing tool designed to interact with WinRM.

If we want to find out whether one or more remote servers can be reached via WinRM, we can easily do this with the help of PowerShell. The Test-WsMan cmdlet is responsible for this, and the host's name in question is passed to it. In Linux-based environments, we can use the tool called evil-winrm, another penetration testing tool designed to interact with WinRM.

Windows Remote Management Protocols

```
Djerbien@htb[/htb]$ evil-winrm -i 10.129.201.248 -u Cry0l1t3 -p P455w0rD!

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemente

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Cry0l1t3\Documents>
```

https://learn.microsoft.com/en-us/powershell/module/microsoft.wsman.management/test-wsman?view=powershell-7.2

# WMI

**Windows Management Instrumentation (WMI)**

is **Microsoft's implementation** and also an **extension** of the **Common Information Model (CIM)**, core functionality of the standardized **Web-Based Enterprise Management (WBEM)** for the Windows platform.

WMI allows **read** and **write** access to almost **all settings on Windows systems**. Understandably, this makes it the most critical interface in the Windows environment for the administration and remote maintenance of Windows computers, regardless of whether they are PCs or servers.

WMI **is typically accessed** via **PowerShell**, **VBScript**, or the **Windows Management Instrumentation Console (WMIC)**.

WMI is not a single program but consists of several programs and various databases, also known as repositories.

| | |
|---|---|
| PORT | The initialization of the WMI communication always takes place on **TCP port 135**, and after the successful establishment of the connection, the communication **is moved to a random port**. For example, the program wmiexec.py from the Impacket toolkit can be used for this. |

**WMIexec.py**

Windows Remote Management Protocols

```
Djerbien@htb[/htb]$ /usr/share/doc/python3-impacket/examples/wmiexec.py Cry0l1t3:"P455w0rD!"@10.129.201.248 "hostname"

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
ILF-SQL-01
```

Again, it is necessary to mention that the knowledge gained from installing these services and playing around with the configurations on our own Windows Server VM for gaining experience and developing the functional principle and the administrator's point of view cannot be replaced by reading manuals. Therefore, we strongly recommend setting up your own Windows Server, experimenting with the settings, and scanning these services repeatedly to see the differences in the results.