Assignment 1 Question

```
For each sample

a.  What is the name of the malware family?
b.  What method of malware persistence is in use for this sample?
c.  Are there any potential host-based indicators and/or network-based indicators
for the malware?
d.  Any special observations about this sample after it was executed?
e.  Devise/Deploy a method to extract the malware configuration file for similar
samples. Provide a step-by-step explanation and any references that was used.
```

Preamble about structure of A1 subfolder

The main report is A1.md. The relevant analysis files are:

```
*_imports.txt (for IDA imports)
*_strings.txt for strings output
*_reg_win7.txt for regshot output
*.csv for procmont output
*.ipynb for procmon analysis
```

and other relevant output files.

Sample A

a. Family name

XtremeRAT (or XRAT) is a family of Remote Access Trojan. According to FireEye [A1], "The XtremeRAT was developed by “xtremecoder” and has been available since at least 2010. Written in Delphi, the code of XtremeRAT is shared amongst several other Delphi RAT projects including SpyNet, CyberGate, and Cerberus. The RAT is available for free; however, the developer charges 350 Euros for the source code. Unfortunately for xtremecoder, the source code has been leaked online. The current version is Xtreme 3.6, however, there are a variety of “private” version of this RAT available as well. As such, the official version of this RAT and its many variants are used by a wide variety of actors."

A full description of the malware functionalities can be found here -- taken from the XRAT whitepaper

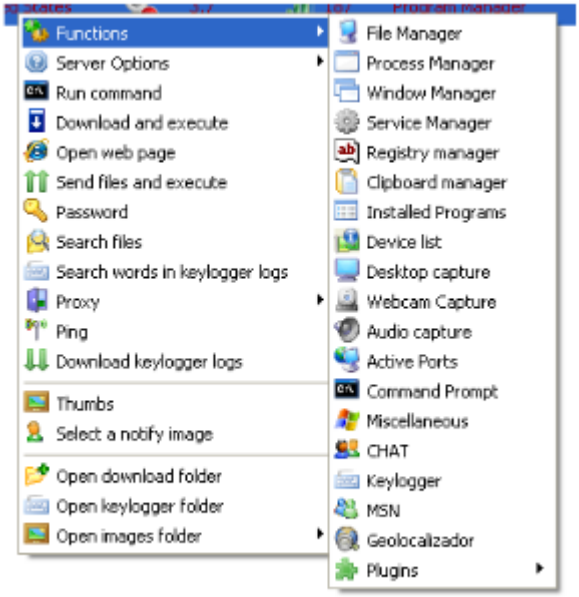


Figure 4: XtremeRAT 3.7 Functions.

a2. Functionalities

It is a RAT so like usually RAT it grants attacker access into the infected machines with a few other functionalities.

I have extracted the imports libraries it uses in a_imports.txt and the strings output in a_strings.txt. Mapping them to the functionalities stated in [A1]

```
Interact with the victim via a remote shell (ShellExecuteW)
Upload/download files (URLDownloadToCacheFileW,FtpPutFileW,Ftp-related functions)
Interact with the registry
(SHDeleteKeyW,SHDeleteValueW,RegQueryValueExA,RegOpenKeyExA,REgCloseKey)
Manipulate running processes and services
(CreateThread,CreateProcessW,CreateFileW,CreateEventA,)
Capture images of the desktop (GetDesktopWindow)
GetClipboard data, GetKeyboardstate (Esp impt since it's arabic keyboard)
Record from connected devices, such as a webcam or microphone
```

I also notice a few interesting things in strings output:

```
SOFTWARE\Borland\Delphi\RTL: it is definitely Delphi
Software\Microsoft\Windows\CurrentVersion\Run: it will run on startup.
Software\Microsoft\Active Setup\Installed Components\; restart; StubPath: it will
attempt funny things during logon.
```

_Author's notes: I will copy output for certain parts under other parts in order to avoid reference pointers pointing everywhere. So there will be duplicates _

b. Persistence

For this section, I run the malware multiple times each with less filters as I notice it inject itselfs into svchost.exe and msedge.exe (my default browser). This is a behaviour already examined by TrendMicro [A2] -- "It injects itself into the following processes as part of its memory residency routine".

The final procmon file I settled with was a_all_events.csv which contains 0 filter.

From my own procmon log + ipynb analysis and the sandbox result *see the notes at the end of this section*, I noticed code injection, hiding a copy of itself and HKEY modification as the main persistence mechanism

b.1 Code Injection

From ipynb+procmon:

```
In [16]: FileCreateEvents = aDF[(aDF["Operation"] == "CreateFile")]
         Sus_files = {}
         for i in range(len(FileCreateEvents)):
             file_created = FileCreateEvents.iloc[i,4]
             for file_ext in ["exe","cfg","dat","html"]:
                 if file_created.endswith(file_ext):
                     #print(f"\x1b[31m{file_ext}\x1b[0m ",end='')
                     #print(f"{FileCreateEvents.iloc[i,1]} -- {FileCreateEvents.iloc[i,4]} ")  #-- {FileCreateEvents.iloc[i,6]}
                     if file_created not in Sus_files.keys():
                         Sus_files[file_created] = (file_ext,i)
             #print({FileCreateEvents.iloc[i,4]} )
         for k,v in Sus_files.items():
             print(f"{FileCreateEvents.iloc[v[1],3]} \x1b[31m{v[0]}\x1b[0m ",end='')
             print(f"{FileCreateEvents.iloc[v[1],1]} -- {FileCreateEvents.iloc[v[1],4]} ")  #-- {FileCreateEvents.iloc[i,6]}

         CreateFile exe Explorer.EXE -- C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_A\Sample A -\s
         erver.exe
         CreateFile exe Explorer.EXE -- C:\Windows\System32\WFS.exe
         CreateFile cfg server.exe -- C:\Users\IEUser\AppData\Roaming\BlackMamba_Mutex.cfg
         CreateFile exe server.exe -- C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_A\Sample A -\svc
         host.exe
         CreateFile exe server.exe -- C:\Windows\System32\svchost.exe
         CreateFile html server.exe -- C:\Users\IEUser\AppData\Local\Temp\x.html
         CreateFile exe server.exe -- C:\Program Files\Internet Explorer\iexplore.exe
```

From sandbox:

Codeinjection	Chained Direct Code Injection	Source: C:\Users\admin\AppData\Local\Temp\Sample A -\server.exe,Target: C:\Windows\System32\svchost.exe	1288 808
Codeinjection	Chained Direct Code Injection	Source: C:\Users\admin\AppData\Local\Temp\Sample A -\server.exe,Target: C:\Program Files\Internet Explorer\iexplore.exe	1288 2436
Codeinjection	Multiple Memory Write With Inline Hook Code Injection	Source: C:\Windows\System32\winlogon.exe,Target: C:\Windows\explorer.exe	388 3080

Notes: I have emailed our profs about the behaviour of malware not executing fully in different environment. I realized that they behave more mildly in a Win10 env compared to Win7 when I compare the regshots result from two machines. This was the replies I got from the profs:

Re: Behaviours of given Sample A in different environment

MA

malware analyst <cecz4069@gmail.com
>
Mon 2/22/2021 12:26 AM
To: #HOANG VIET#

You may wish to try doing the analysis using the Win 7 VM that we provided. Do change the default browser to IE, as I've found through my own testing that Sample A doesn't seem to run completely otherwise.

Regards,
Mr Ho

From: malware analyst <cecz4069@gmail.com>
Sent: Sunday, February 21, 2021 11:10 AM
To: #HOANG VIET# <C180058@e.ntu.edu.sg>
Subject: Re: Behaviours of given Sample A in different environment

1. I would suggest using the default window 7 vm that was provided by us for better and accurate results. Please kindly disable windows defender before you proceed. The network settings should be host-only.

2. Treat the sandbox results like another "experiment" where you have detonated the malware in a safe environment (one of these environments is called chicago PC). There should be consistent results across "these experiments" that you have seen. Try cross verifying on both sandbox results, you should spot something.

b.2 Drop a copy of itself into the system

From ipynb+procmon:

```
In [16]: FileCreateEvents = aDF[(aDF["Operation"] == "CreateFile")]
Sus_files = {}
for i in range(len(FileCreateEvents)):
    file_created = FileCreateEvents.iloc[i,4]
    for file_ext in ["exe","cfg","dat","html"]:
        if file_created.endswith(file_ext):
            #print(f"\x1b[31m{file_ext}\x1b[0m ",end='')
            #print(f"{FileCreateEvents.iloc[i,1]} -- {FileCreateEvents.iloc[i,4]} ")  #-- {FileCreateEvents.iloc[i,6]}
            if file_created not in Sus_files.keys():
                Sus_files[file_created] = (file_ext,i)
    #print({FileCreateEvents.iloc[i,4]} )
for k,v in Sus_files.items():
    print(f"{FileCreateEvents.iloc[v[1],3]} \x1b[31m{v[0]}\x1b[0m ",end='')
    print(f"{FileCreateEvents.iloc[v[1],1]} -- {FileCreateEvents.iloc[v[1],4]} ")  #-- {FileCreateEvents.iloc[i,6]}
host.exe
CreateFile exe server.exe -- C:\Windows\System32\svchost.exe
CreateFile html server.exe -- C:\Users\IEUser\AppData\Local\Temp\x.html
CreateFile exe server.exe -- C:\Program Files\Internet Explorer\iexplore.exe
CreateFile exe server.exe -- C:\Program.exe
CreateFile exe server.exe -- C:\Program Files\Internet.exe
CreateFile exe iexplore.exe -- C:\Windows\InstallDir\Server.exe
CreateFile dat iexplore.exe -- C:\Users\IEUser\AppData\Local\Microsoft\Windows\Temporary Internet Files\counters.dat
CreateFile exe svchost.exe -- C:\Windows\System32\dllhost.exe
```

From sandbox:

File	Failed	C:\Windows\INSTALLDIR	2436
File	Failed	C:\Windows\INSTALLDIR	2436
Folder	Created	C:\Windows\InstallDir	2436
Malicious Alert	Flags	Message: Folder Created	null
File	Failed	C:\Windows\InstallDir\SERVER.EXE	2436
File	Created	C:\Windows\InstallDir\Server.exe	2436

b.3 Modify registry keys

From ipynb+procmon:

IE suspicious HKEY:

```
        else:
            for i in range(len(Events)):
                if Events.iloc[i,1] == pid:
                    print(f"{Events.iloc[i,4]}:{Events.iloc[i,6]}")
#To see all registry events, uncomment the below
#see_events(aDF,[ops for ops in uniq_ops["Operation"] if ops.startswith("Reg")])
#The only interesting ones are RegSetValue
see_events(aDF,["RegSetValue"],"iexplore.exe")
```

```
========================= RegSetValue ==========================
HKCU\Software\BlackMamba_Mutex\ServerStarted:Type: REG_EXPAND_SZ, Length: 40, Data: 20/02/2021 23:43:04
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\HKLM:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKCU\Software\BlackMamba_Mutex\ServerName:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Server.exe
```

svchost suspicious HKEY:

```
see_events(aDF,["RegSetValue"],"svchost.exe")
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\HKLM:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\HKLM:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU:Type: REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Serv
er.exe
```

Other interesting keys:

```
iexplore.exe --RegSetValue -- HKCU\Software\BlackMamba_Mutex\ServerName -- Type:
REG_EXPAND_SZ, Length: 66, Data: C:\Windows\InstallDir\Server.exe
iexplore.exe --RegSetValue -- HKCU\Software\BlackMamba_Mutex\ServerStarted --
Type: REG_EXPAND_SZ, Length: 40, Data: 20/02/2021 23:43:04
```

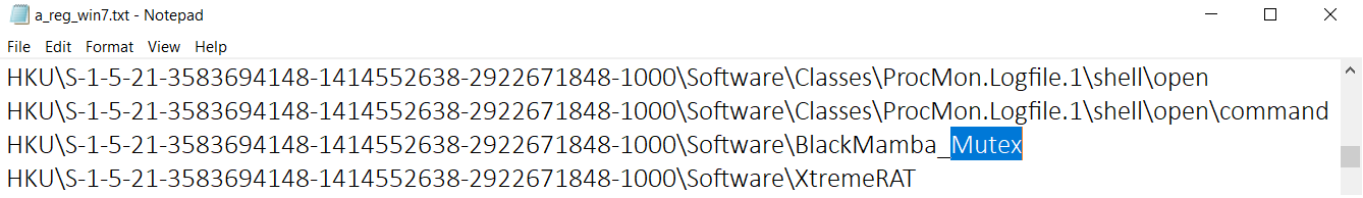
From the strings output we already notices:

```
Software\Microsoft\Windows\CurrentVersion\Run: C:\Windows\InstallDir\Server.exe
(now we know the value)
Software\Microsoft\Active Setup\Installed Components\; restart; StubPath: it will
attempt funny things during logon (not detected)
```

However the malware doesn't seem to modify the second key. The other keys matched up with most (if not all) of the HKEY mod events reported by TrendMicro [A2].

b.4 Mutex

From Regshot:



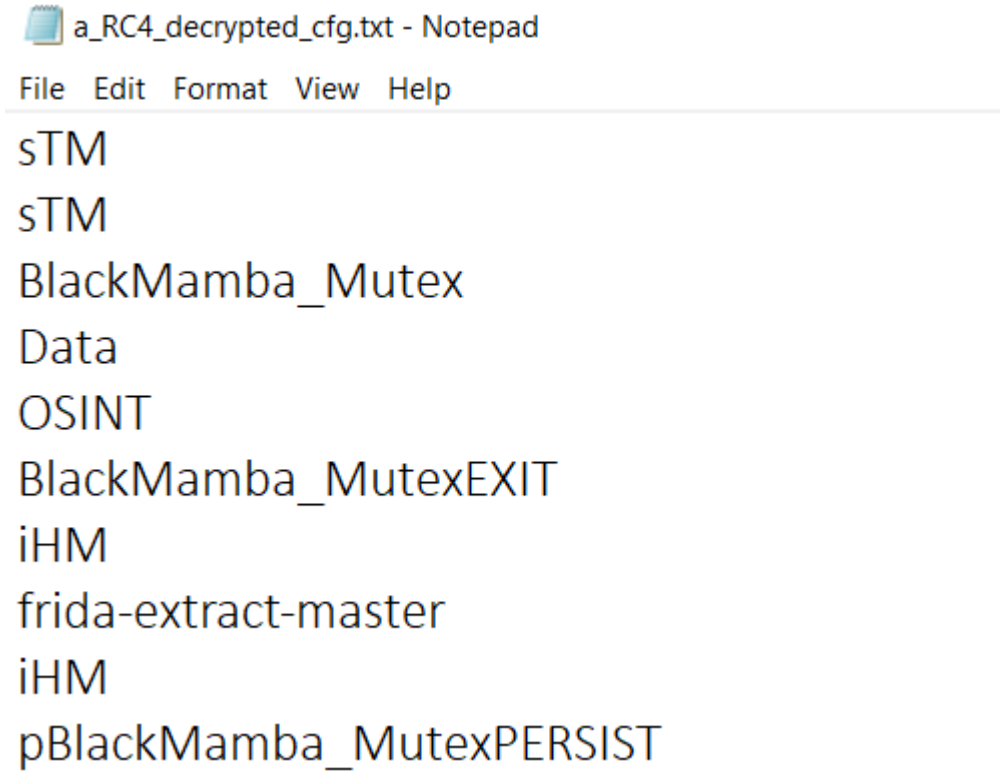
From ipynb+procmon:

```
server.exe --RegSetValue -- HKCU\Software\XtremeRAT\Mutex -- Type: REG_EXPAND_SZ,
Length: 34, Data: BlackMamba_Mutex
iexplore.exe --CreateFile --
C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\BlackMamba_Mutex.xtr
server.exe --CreateFile -- C:\Users\IEUser\AppData\Roaming\BlackMamba_Mutex.cfg
```

From sandbox:

Mutex		\Sessions\1\BaseNamedObjects\BlackMamba_MutexEXIT	808
Mutex		\Sessions\1\BaseNamedObjects\BlackMamba_Mutex	1288
Mutex		\Sessions\1\BaseNamedObjects\BlackMamba_MutexPERSIST	1288

And later on during resource decryption, we also find these:



c. Indicators

c.1 Network based

Later on in section 5 we will explain how we got the C2 key IPs as:

```
C2 server IP: 111.65.40.69
DNS server IP: 112.66.17.12
```

The reason is because fakenet doesn't report anything as said in Notes.

This also holds out with FireEye Sandbox result.

Bot Communication Details

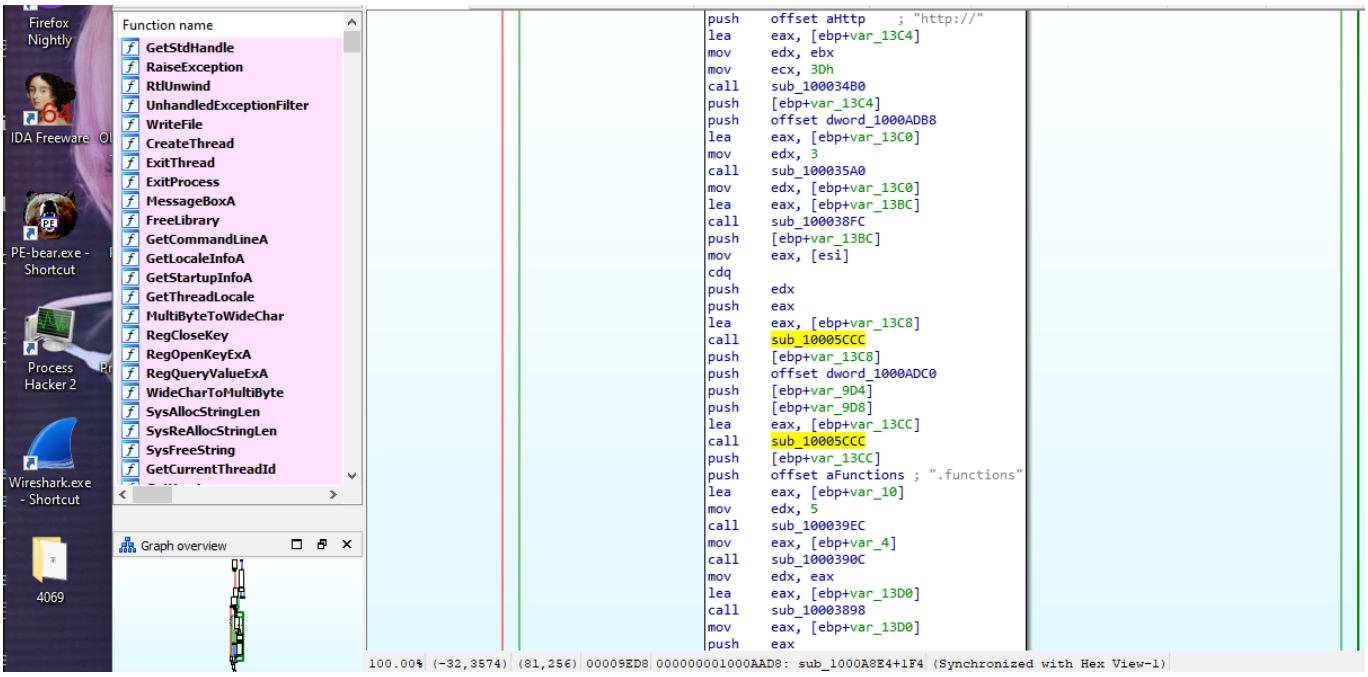
Server DNS Name 111.65.40.69
First Seen At 2021-02-01T09:10:36+00:00
Service Port 81
Signature Name Backdoor.Xtrat

Direction	Command	User Agent	Host	Connection	Pragma	Others
GET	/1234567890.functions HTTP/1.1	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)	111.65.40.69:81	Keep-Alive	undefined	Accept:*/*,Accept-Encoding:gzip, deflate,DNT:1,Cache-Control:no-cache

Server DNS Name 112.66.17.12
First Seen At 2021-02-01T09:10:36+00:00
Service Port 82
Signature Name Backdoor.Xtrat

Direction	Command	User Agent	Host	Connection	Pragma	Others
GET	/1234567890.functions HTTP/1.1	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)	112.66.17.12:82	Keep-Alive	undefined	Accept:*/*,Accept-Encoding:gzip, deflate,DNT:1,Cache-Control:no-cache

And seen in IDA



where it calls the {IP}/1234567890.functions

c.2 Host based

The files events, mutex and HKEY modifications are prime examples of host based indicators. I have selected a few important one:

```
--Language--
Delphi

--RegSetValue--
iexplore.exe HKCU\Software\BlackMamba_Mutex\ServerStarted
iexplore.exe HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\HKLM -- Data:
C:\Windows\InstallDir\Server.exe
iexplore.exe HKCU\Software\Microsoft\Windows\CurrentVersion\Run\HKCU -- Data:
C:\Windows\InstallDir\Server.exe
iexplore.exe HKCU\Software\BlackMamba_Mutex\ServerName --  Data:
C:\Windows\InstallDir\Server.exe
iexplore.exe HKCU\Software\BlackMamba_Mutex\ServerName -- Type: REG_EXPAND_SZ,
Length: 66, Data: C:\Windows\InstallDir\Server.exe
iexplore.exe HKCU\Software\BlackMamba_Mutex\ServerStarted -- Type: REG_EXPAND_SZ,
Length: 40, Data: 20/02/2021 23:43:04

--CreateFile--
cfg iexplore.exe -- --  C:\Users\IEUser\AppData\Roaming\BlackMamba_Mutex.cfg
xtr iexplore.exe -- --
C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\BlackMamba_Mutex.xtr
exe server.exe -- C:\Windows\System32\svchost.exe
html server.exe -- C:\Users\IEUser\AppData\Local\Temp\x.html
exe server.exe -- C:\Program Files\Internet Explorer\iexplore.exe

--Mutex--
```

```
BlackMamba_MutexEXIT
BlackMamba_Mutex
BlackMamba_MutexPERSIST
```

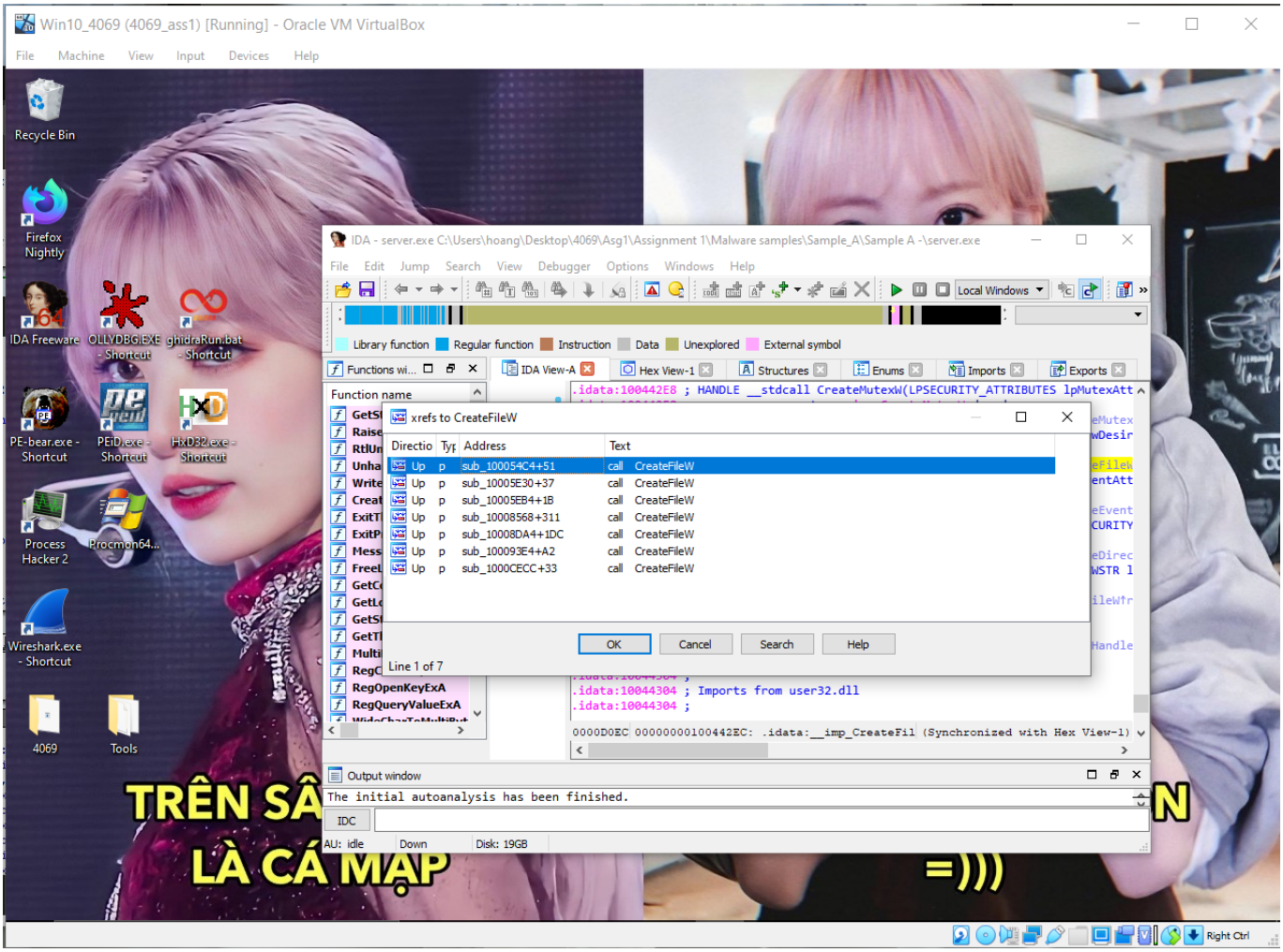
d. Any special observations about this sample after it was executed?

The malware deletes itself in win7 but will stay there in win10. Again due to Notes. This is possibly due to

```
a. Detect itself in a VM environment
b. Persistence mechanism and evasion
```

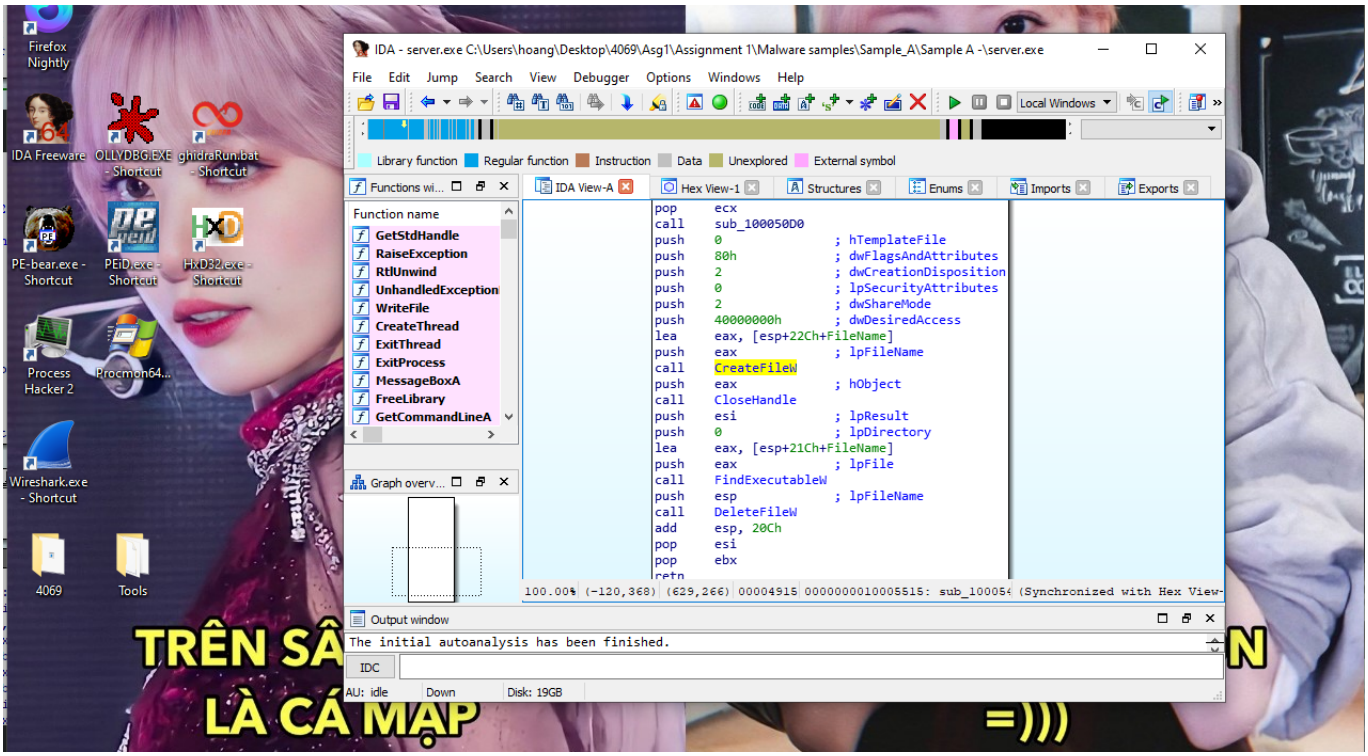
Since it never deletes itself in win10 I guess b. is a more appropriate answer. Code injection traces further proves this point.

Additionally, there's 6 calls to CreateFileW



Setting F2 at all these addresses in OllyDbg, we can only find hit the first breakpoint as the other breakpoints are executed by the infected processes -- svchost and msedge.

And the breakpoint it hit is a junkcode in creating a x.html file that does absolutely nothing



e. Devise/Deploy a method to extract the malware configuration file for similar samples. Provide a step-by-step explanation and any references that was used.

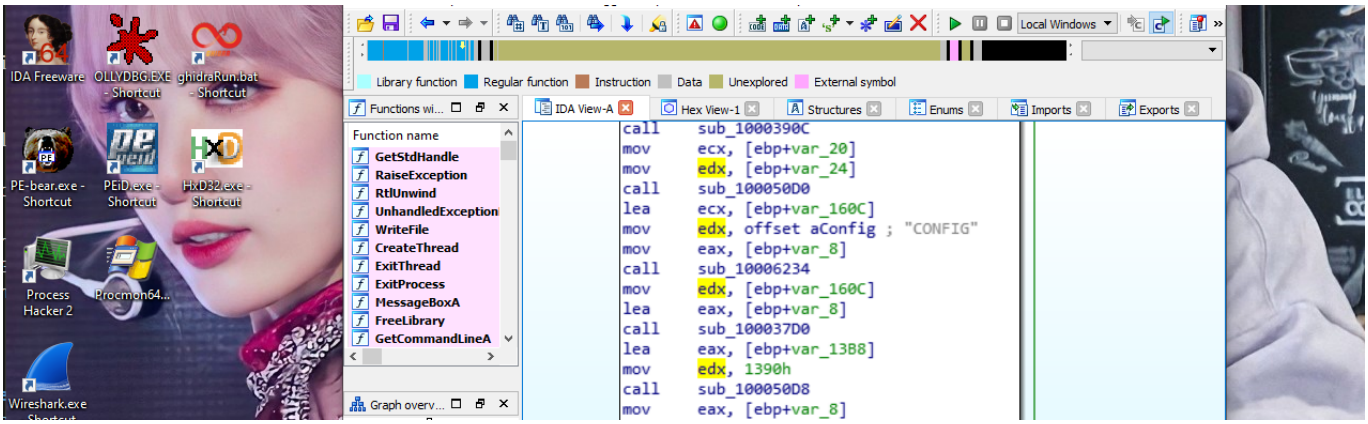
The RAT contains an RC4 encrypted config files that contain several useful information (under %APPDATA%\Roaming\BlackMamba_Mutex.cfg). This can be seen in the encrypted strings in a_strings.txt

For RC4, it is a relatively simple, elegant and vulnerable stream cipher algorithm. I have used it to create a small CTF in my blog start page (The pwd is "sosig"). Malwarelu [A3] has done a brilliant job at detecting the KSA and PKGA in their blogs for XTRAT sample.

There is then two approaches to decrypt this file:

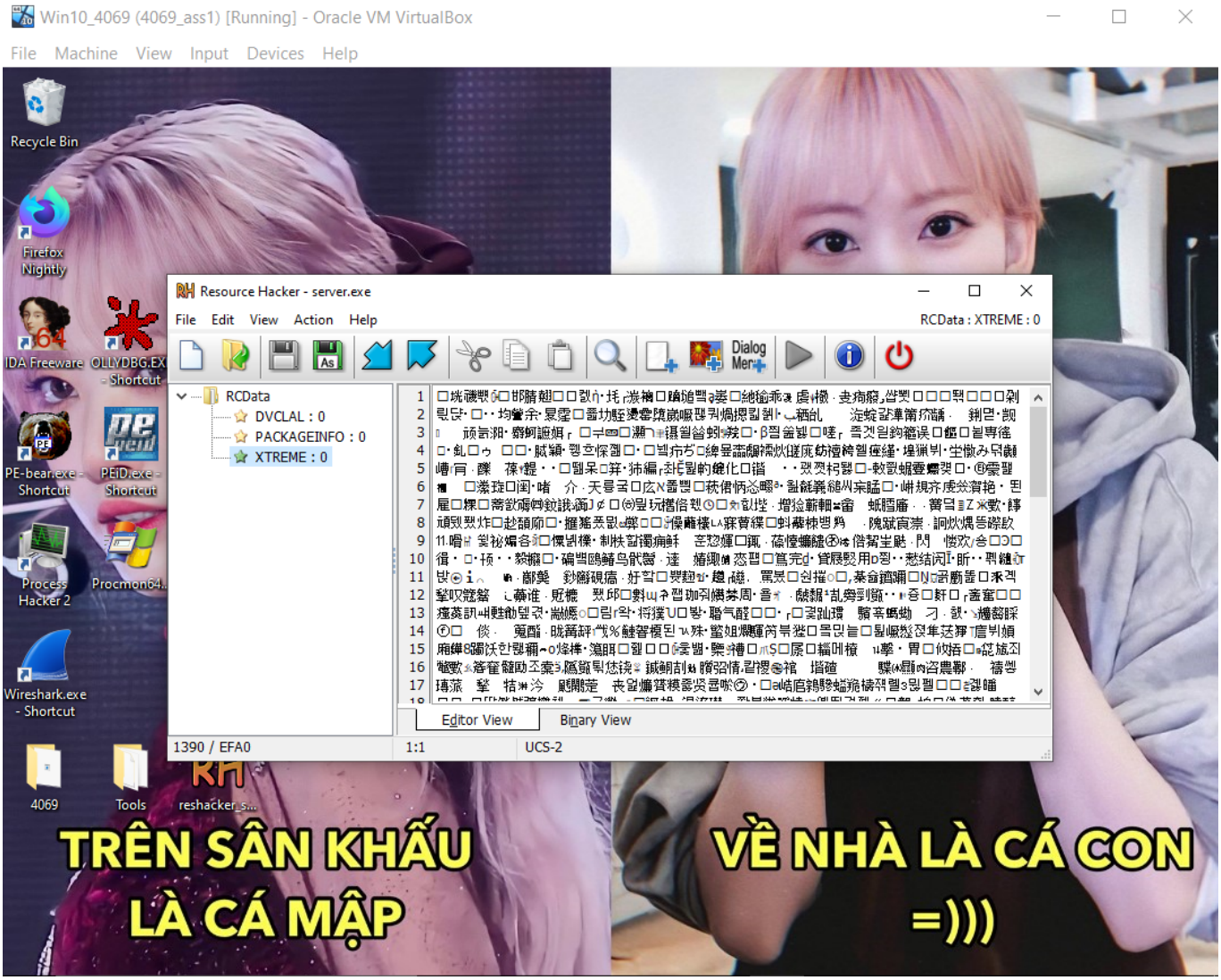
```
1. Ollygdb it till the point it write the file to disk , extract the file/RescHack
the file out. RC4 decrypt using string "CONFIG" and read the content. Decryption
can be done using FireEye XTRAT tools (which is also the same thing that malware
lu used)
2. Use Malwarelu method  of extracting it directly from the resource section, and
the rest is the same. The cool thing is Malwarelu package all of these into a
single python script that is extremely short, so it's more convenient.
```

Why string CONFIG? FireEye said there's two possible strings "CYBERGATEPASS" or "CONFIG". Also in IDA we can tell in IDA that the key is "CONFIG".



A step by step explanation of method 1:'

1. Locate the .cfg files(You may want to modify the malware such that it doesn't delete this file right before it delete itself) OR take it from rsrc section using tools like Resource Hacker
2. Run FireEye tool on it to extract the data.



A step by step explanation of method 2:

The link is [A3]. Just use their xtremerat_config.py on the executable. It is very convenient. I will explain the python file below:

1. Using python pefile module, we locate the resource section directories of the executable

2. Get the directory entry. Then for each of the entries (which will contain a block of 16 strings)
3. Find the relative virtual address of the strings. Since these are almost always differ from its position in disk, we need to use pe.get_memory_mapped_image()[data_rva:data_rva+size] to get the actual data

The output file is a_RC4_decrypted_cfg.txt

We see the following info as mentioned:

```
C2 server IP: 111.65.40.69
DNS server IP: 112.66.17.12
Folder is InstallDir
File is server.exe
Injected process is %DEFAULTBROWSER% & %SERVER%
Mutex created: BlackMamba_Mutex, BlackMamba_MutexEXIT BlackMamba_MutexPERSIST

--misc--
NTU_Education
ConCSA_MA_Team
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
GUID: {5460C4DF-B266-909E-CB58-E32B79832EB2}
Microsoft Terminal Services
ftp.ftpserver.com
```

For the OSINT work, I have read and used references from the following work

```
--notation--
Ax -- referenced
MA -- misc, not referenced, but noteworthy read
```

References for Sample A

A1. FireEye XRAT

A2. TrendMicro XRAT threat encyclopedia

A3. Malware lu analysis

A4. FireEye XTRAT decryptor

Misc 1. Malwarebyte XTRAT backdoor

Misc 2. SANS whitepaper

Misc 3. Cyware

Misc 4. KrebsonSecurity

Sample B

f. What is the name of the malware family?

Hawkeye keylogger. It also loads in other malware like a Bitcoin Wallet Stealer. Goes by the name Phulli.exe, but overall is still very much of a HawkEye variant. This is also reported in [B5]: "Researchers observed the HawkEye keylogger acting as the first-stage downloader for a cryptocurrency miner in a new phishing campaign."

The malware is written in .NET, so have fun IDA it if you are poor. I used .NET Reflector trial version.

g. Functionality of the malware?

It has a few:

```
1.  Steals information and credentials from compromised systems (the original
HawkEye steals from browsers and mail clients, then this variant also steals
minecraft, runescape, steams and bitcoin wallet creds).
2.  Also spread via removable data (usb) and disable taskmgr among other processes
```

Check out here at this tweet. Although the string in its strings output in b_strings.txt

```
C:\Users\Jovan\Documents\Visual Studio
2010\Projects\Stealer\CMemoryExecute\CMemoryExecute\obj\Release\CMemoryExecute.pdb
```

reminds me of Lokibot, I think it's more Hawkeye than Loki.

From strings output we can see the following:

```
    ================================================
            WEB Browser Password Stealer
    ================================================
    ================================================
            Mail Messenger Password Stealer
    ================================================
    ================================================
            Internet Download Manager Stealer
    ================================================
    ================================================
            JDownloader Password Stealer
    ================================================
    ================================================
            WEB Browser Password Stealer
    ================================================

    HawkEye Keylogger | RuneScape Stealer |
    HawkEye Keylogger | BitCoin Stealer |
    HawkEye Keylogger | MineCraft Stealer |
    DisableTaskManager
```

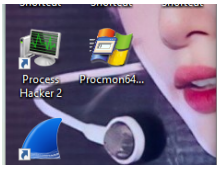
```
    Disablecmd
    Disablemsconfig
    Disablereg
    regedit
    Disablestartup
    HawkEye Keylogger | Keylog Records |
    ==============================================
                      ClipBoard Records
    ==============================================
    **********************************************
                      Keylog Records
    ==============================================
```

The last noteworthy point is that in the NET decompiler, we see a runPE function. This suggests that the malware is capable of running some payload executable PE files.

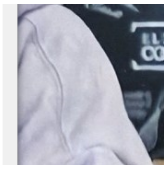
h. Indicators

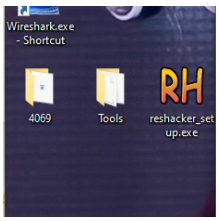
h.1 Network

The following was captured in fakenet

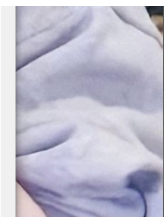



```
02/22/21 05:23:33 PM [      DNS Server] Received A request for domain 'whatismyipaddress.com'.
02/22/21 05:23:33 PM [       Diverter] Windows Update.exe (3308) requested TCP 192.0.2.123:80
02/22/21 05:23:33 PM [   HTTPListener80]  GET / HTTP/1.1
02/22/21 05:23:33 PM [   HTTPListener80]  Host: whatismyipaddress.com

02/22/21 05:23:33 PM [   HTTPListener80]  Connection: Keep-Alive
```






```
02/22/21 05:23:33 PM [   HTTPListener80]
02/22/21 05:23:34 PM [       Diverter] svchost.exe (1292) requested UDP 255.255.255.255:67
02/22/21 05:23:35 PM [       Diverter] svchost.exe (2032) requested UDP 192.168.116.128:53
02/22/21 05:23:35 PM [      DNS Server] Received A request for domain 'smtp.gmail.com'.
02/22/21 05:23:35 PM [       Diverter] Windows Update.exe (3308) requested TCP 192.0.2.123:587
02/22/21 05:23:36 PM [       Diverter] svchost.exe (5648) requested UDP 239.255.255.250:1900
02/22/21 05:23:43 PM [       Diverter] svchost.exe (2032) requested UDP 192.168.116.128:53
02/22/21 05:23:43 PM [      DNS Server] Received A request for domain 'ctldl.windowsupdate.com'.
```



The following calls were made:

```
whatismyipaddress.com
192.0.2.123:587
```

These are efforts to resolve external IP addresses that the malware does not have in its configuration already.

Sandbox result:

00:11:828	DNS Queries	Translated a host name WHATISMYIPADDRESS.COM into an IP address
00:11:843	Socket Activities	Received data from a connected or bound socket
00:11:843	Socket Activities	Sent data on a connected socket

h.2 Host

a. Registry Modification

The only important one is:

```
Windows Update.exe --RegSetValue --
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Windows Update -- Type: REG_SZ,
Length: 100, Data: C:\Users\IEUser\AppData\Roaming\WindowsUpdate.exe
```

which enables its autorun.

```
In [4]: def see_events(df,ops_list,pid=None):
            for op in ops_list:
                print("\x1b[31m==========\x1b[0m",op,"\x1b[31m==========\x1b[0m")#"\x1b[31m\"red\"\x1
                Events = df[(df["Operation"] == op)]
                if pid == None:
                    for i in range(len(Events)):
                        print(f"{Events.iloc[i,1]} --{Events.iloc[i,3]} -- {Events.iloc[i,4]} -- {Events.iloc[i,6]}")
                else:
                    for i in range(len(Events)):
                        if Events.iloc[i,1] == pid:
                            print(f"{Events.iloc[i,1]} --{Events.iloc[i,3]} -- {Events.iloc[i,4]} -- {Events.iloc[i,6]}")
        #To see all registry events, uncomment the below
        #see_events(aDF,[ops for ops in uniq_ops["Operation"] if ops.startswith("Reg")])
        #The only interesting ones are RegSetValue
        see_events(bDF,["RegSetValue"])
```

```
D, Length: 4, Data: 1048576
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASAPI32\FileDirectory -- Type: REG_EX
PAND_SZ, Length: 34, Data: %windir%\tracing
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\EnableFileTracing -- Type: RE
G_DWORD, Length: 4, Data: 0
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\EnableConsoleTracing -- Type:
REG_DWORD, Length: 4, Data: 0
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\FileTracingMask -- Type: REG_
DWORD, Length: 4, Data: 4294901760
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\ConsoleTracingMask -- Type: R
EG_DWORD, Length: 4, Data: 4294901760
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\MaxFileSize -- Type: REG_DWOR
D, Length: 4, Data: 1048576
Windows Update.exe --RegSetValue -- HKLM\SOFTWARE\Microsoft\Tracing\Windows Update_RASMANCS\FileDirectory -- Type: REG_EX
PAND_SZ, Length: 34, Data: %windir%\tracing
Windows Update.exe --RegSetValue -- HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Hidden -- Type: REG_
DWORD, Length: 4, Data: 1
Windows Update.exe --RegSetValue -- HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Windows Update -- Type: REG_SZ, Le
ngth: 100, Data: C:\Users\IEUser\AppData\Roaming\WindowsUpdate.exe
```

b. Files creations:

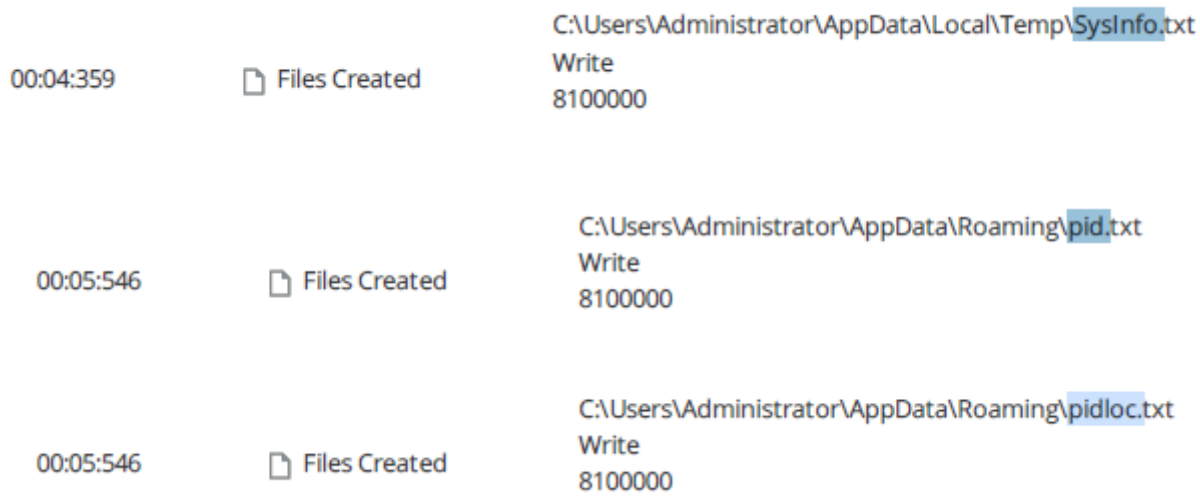
```
        if pid == None:
            #for i in range(len(Events)):
                #print(f"{Events.iloc[i,1]} --{Events.iloc[i,3]} -- {Events.iloc[i,4]}")
            _tmp = list(Events.iloc[:,4].unique())
            for what in _tmp:

                if type(what)== str and ( what.endswith("exe") or what.endswith("txt")):
                    print(what)
see_events_two(bDF,["CreateFile"])
```

```
========== CreateFile : exe ==========
C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_B\Sample_B.exe
C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_B\en-US\Phulli.resources.exe
C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_B\en-US\Phulli.resources\Phulli.resources.exe
C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_B\en\Phulli.resources.exe
C:\Users\IEUser\Desktop\Assignment 1(2)\Assignment 1\Malware samples\Sample_B\en\Phulli.resources\Phulli.resources.exe
C:\Users\IEUser\AppData\Local\Temp\SysInfo.txt
C:\Users\IEUser\AppData\Roaming\Windows Update.exe
C:\Users\IEUser\AppData\Roaming\en-US\Phulli.resources.exe
C:\Users\IEUser\AppData\Roaming\en-US\Phulli.resources\Phulli.resources.exe
C:\Users\IEUser\AppData\Roaming\en\Phulli.resources.exe
C:\Users\IEUser\AppData\Roaming\en\Phulli.resources\Phulli.resources.exe
C:\Users\IEUser\AppData\Roaming\pid.txt
C:\Users\IEUser\AppData\Roaming\pidloc.txt
C:\Users\IEUser\AppData\Roaming\WindowsUpdate.exe
C:\Windows\Microsoft.NET\Framework\v2.0.50727\vbc.exe
C:\Users\IEUser\AppData\Local\Temp\holdermail.txt
C:\Users\IEUser\AppData\Local\Temp\holderwb.txt
```

The more important files are:

```
%temp%/sysinfo.txt -- contains its running path
%appdata%/pid.txt -- contains its pid
%appdata%/pidloc.txt -- same
```

The same can be seen from sandbox results:



So to sum up everything so far on indicators:

```
    ---files---
    %temp%/sysinfo.txt -- contains its running path
    %appdata%/pid.txt -- contains its pid
    %appdata%/pidloc.txt
    %temp%/holdermail.txt -- more on this later
    %temp%/holderwb.txt -- more on this later too

    --keys--
   HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Windows Update
   -- Data: C:\Users\IEUser\AppData\Roaming\WindowsUpdate.exe (the standard
autorun mechanism)

   --network--
   whatismyipaddress (not reliable -- a lot of false alarms)
   beacon activity for windows update.exe for a php page
```

i. Special Observation

Injection:

We can also see the first thing this malware do is to copy itself to Windows Update.exe (shown in Regkey part earlier), delete the original copy then runs off.

```
In [5]: see_events(bDF,["Process Create"])

        ========================= Process Create =========================
        Sample_B.exe --Process Create -- C:\Users\IEUser\AppData\Roaming\Windows Update.exe -- PID: 3988, Command line: "C:\Users\I
        EUser\AppData\Roaming\Windows Update.exe"
        Windows Update.exe --Process Create -- C:\Windows\Microsoft.NET\Framework\v2.0.50727\vbc.exe -- PID: 1212, Command line:
        C:\Windows\Microsoft.NET\Framework\v2.0.50727\vbc.exe /stext "C:\Users\IEUser\AppData\Local\Temp\holdermail.txt"
        Windows Update.exe --Process Create -- C:\Windows\Microsoft.NET\Framework\v2.0.50727\vbc.exe -- PID: 2336, Command line:
        C:\Windows\Microsoft.NET\Framework\v2.0.50727\vbc.exe /stext "C:\Users\IEUser\AppData\Local\Temp\holderwb.txt"
```

A true meme-ful moment:

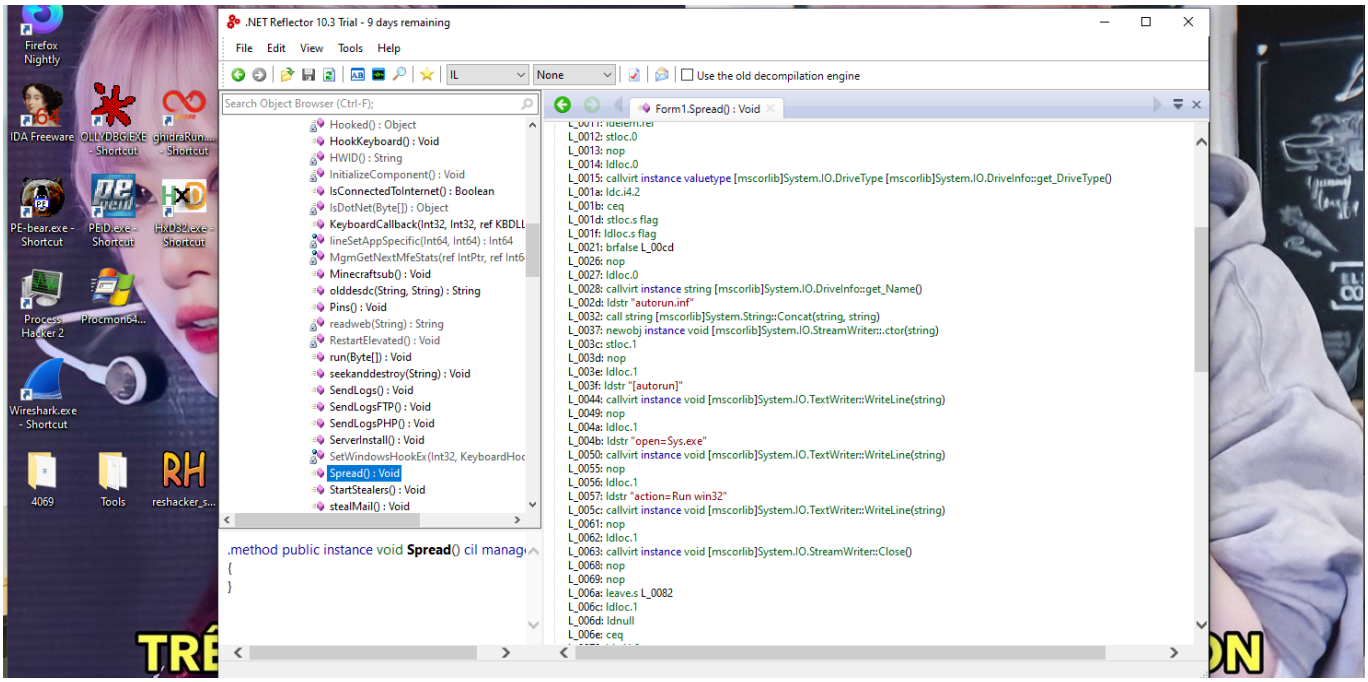
The malware displays a Fake error message: "This is an educational sample..." (nice one there profs). It also delete itself.

Then after that it starts stealing and sending things to its host. The full blown walkthrough of the program has been analyzed by YinzerRE at [B4].

Spreading itself:

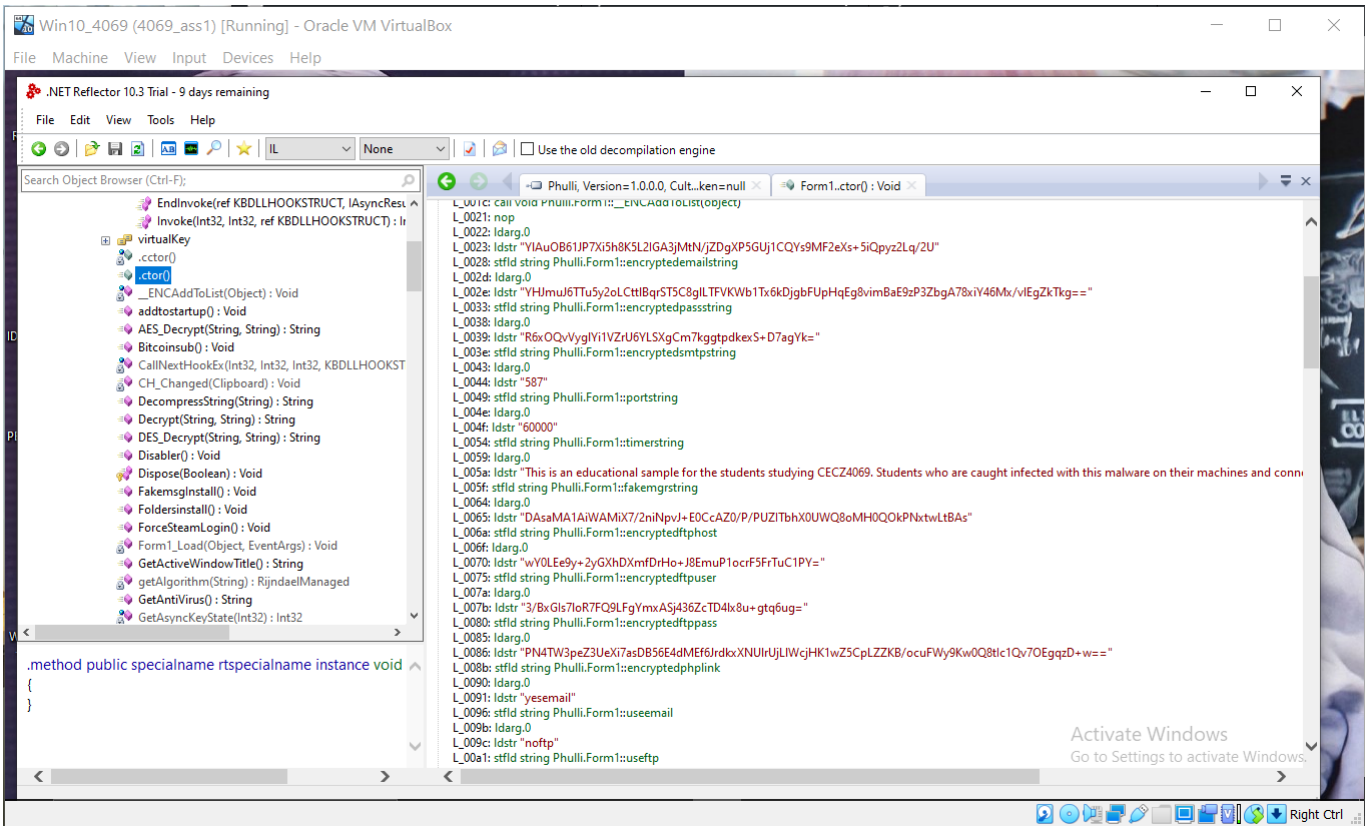
From [B6]: "once the malware is running on the compromised system and if the spreader option is set, it will periodically enumerate all of the connected drives. When a drive is detected of the type DriveType.Removable an Autorun.inf file is created and the malware is copied to the root of the drive as the file Sys.exe."

We can see that here in NET Reflector:



j. Decrypt the config

The encryption are generally done in the constructor (.ctor) function as seen here in the trial NET Reflector (A lot of people use ILSpy too). The strings are AES encrypted (can seen by the Rjindael related struct in the code):



The strings are:

```
    1. encrypted email string:
YlAuOB61JP7Xi5h8K5L2IGA3jMtN/jZDgXP5GUj1CQYs9MF2eXs+5iQpyz2Lq/2U
    2. encrypted pass string:
YHJmuJ6TTu5y2oLCttlBqrST5C8glLTFVKWb1Tx6kDjgbFUpHqEg8vimBaE9zP3ZbgA78xiY46Mx/vlEgZ
kTkg==
    3. encrypted smtp string: R6xOQvVyglYi1VZrU6YLSXgCm7kggtpdkexS+D7agYk=
    4. encrypted ftp host:
DAsaMA1AiWAMiX7/2niNpvJ+E0CcAZ0/P/PUZITbhX0UWQ8oMH0QOkPNxtwLtBAs
    5. encrypted ftp user: wY0LEe9y+2yGXhDXmfDrHo+J8EmuP1ocrF5FrTuC1PY=
    6. encrypted ftp pass: 3/BxGIs7loR7FQ9LFgYmxASj436ZcTD4lx8u+gtq6ug=
    7. encrypted ftp php link:
PN4TW3peZ3UeXi7asDB56E4dMEf6JrdkxXNUlrUjLlWcjHK1wZ5CpLZZKB/ocuFWy9Kw0Q8tIc1Qv7OEgq
zD+w==
```

Using decryption at algorithm by Denialble.org at [B1] using

```
    string secret = "HawkEyeKeylogger";
            #can be "HawkSpySoftware" in some variant
    string salt = "099u787978786";
```

(I guessed prof use a clean sample, it was a wild guess anyway).

Special thanks to Rodel Mendrez@TrustWave and YinzerRE for being very detailed with the decryption, it gave me more insights into the code, but overall I will run using Deniable.org method. They are more or less the

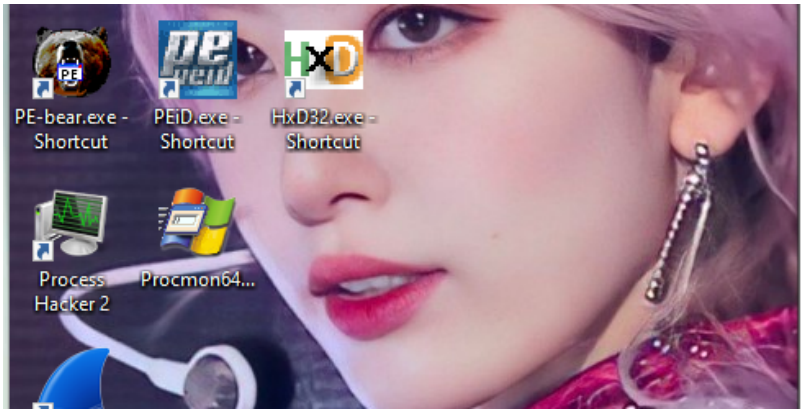
same anyway.

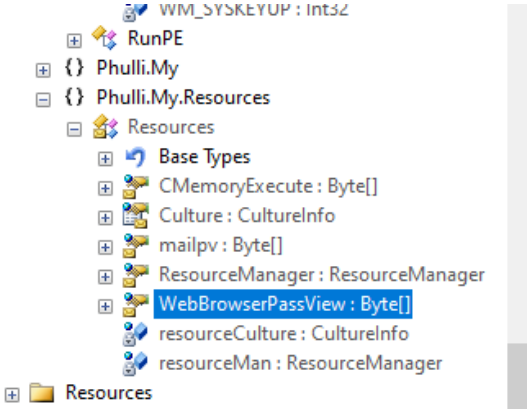
I use an online C# compiler for the process.

Decrypted strings are:

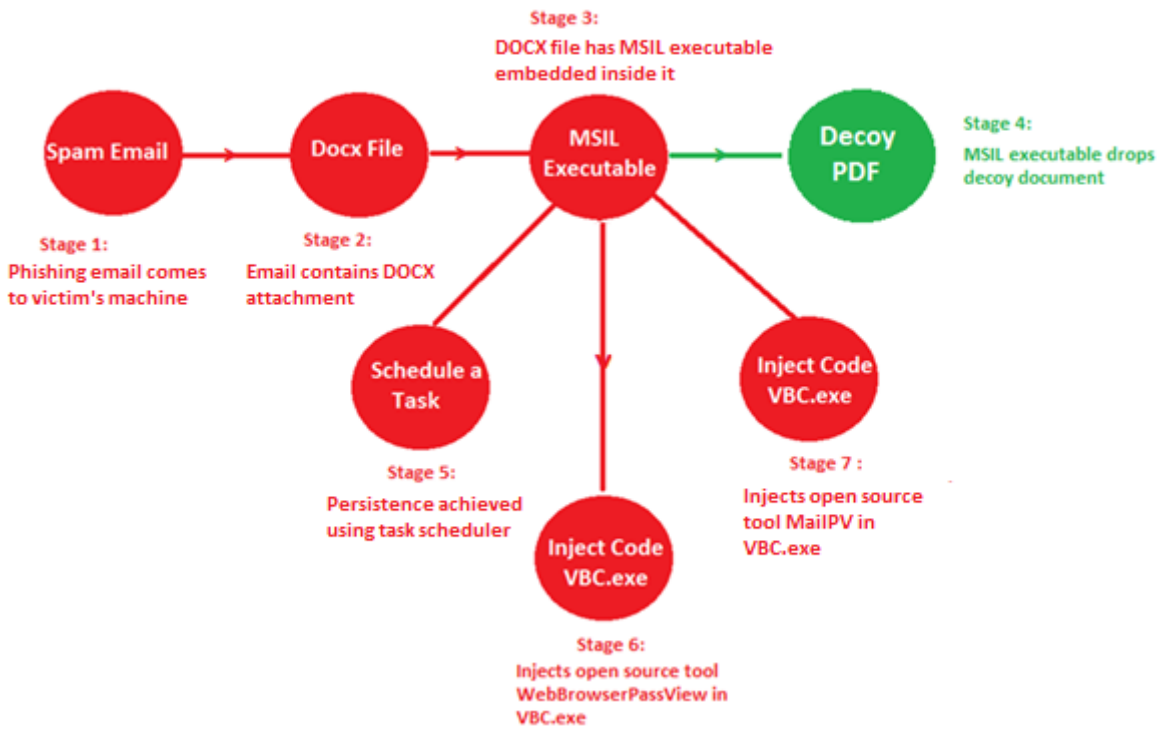
1. email: bossmance4069@gmail.com
2. pass: ThisI$TheAttack3rPa$$w0rd
3. smtp: smtp.gmail.com
4. ftp host: ftp.yourhost.com
5. ftp user: YourUsername
6. ftp pass: YourPassword
7. php site: https://www.site.com/logs.php

There is also some hidden functions at the resource sections





To make sense of these I have kindly borrowed the diagram from FireEye [B8]



The resource are encrypted, but anyway Decrypt class is there to custom decrypt it. FireEye explains:

"After decrypting the resource section, the following files can be extracted:

```
Decoy pdf file.
<Random_Name>.XML-- Contains configuration data for a Windows task creation
CMemoryExecute.dll
WebBrowserPassView.exe
MailPV.exe
```

The xml is used to schedule task in taskmgr in order to attain persistence in infected system. CMemoryExecute.dll is used to load the Mailpv.exe and WebBrowserPassview.exe into the mem of vbc.exe."

The WebBrowserPassView.exe will extract password in web browsers and store in holderwb.txt. MailPV.exe extract password from email clients and store them in holdermail.txt (shown above in files creation section h.2).

References for Sample B

B1. AES Decryptor in C# by Deniable.org

B2. Deniable.org Blog on HawkEye (also includes a trove of other references).

B3. LinkCabin Blog on HawkEye

B4. Yinzer RE Blog

B5. SecIntel News

B6. Golroted technical analysis

B7. Trustwave report

B8. FireEye Report