

# Malware Analysis

Abhinandan Kainth | Anuneet Anand | Mayank Rawat | Rakshit Singh | Tejas Dubhir

*Malware analysis is the process of understanding the functionality and potential impact of a malicious software. It involves dissecting the malware to comprehend how it works, how it can be identified and how it can be eliminated or defeated.*

Computers and smart devices have become an integral part of our lives. People are dependent on computers for various tasks. Any disruption of digital services or loss of data can create havoc in the lives of individuals and organizations. The criminals and sociopaths have a never ending zeal to design custom softwares aimed at disrupting, damaging or gaining access to a computer system. Such malicious softwares are collectively called Malware and include viruses, trojans, worms, ransomwares, etc.

Malware has always posed a threat to computer users. In recent years, malwares have become more invasive and difficult to detect. There is a growing need to study the working of the latest malware and making our systems resilient to their attacks. Analysing the structure and working of malware helps us understand the vulnerabilities in the systems and sharing this knowledge with others motivates people to develop secure systems.

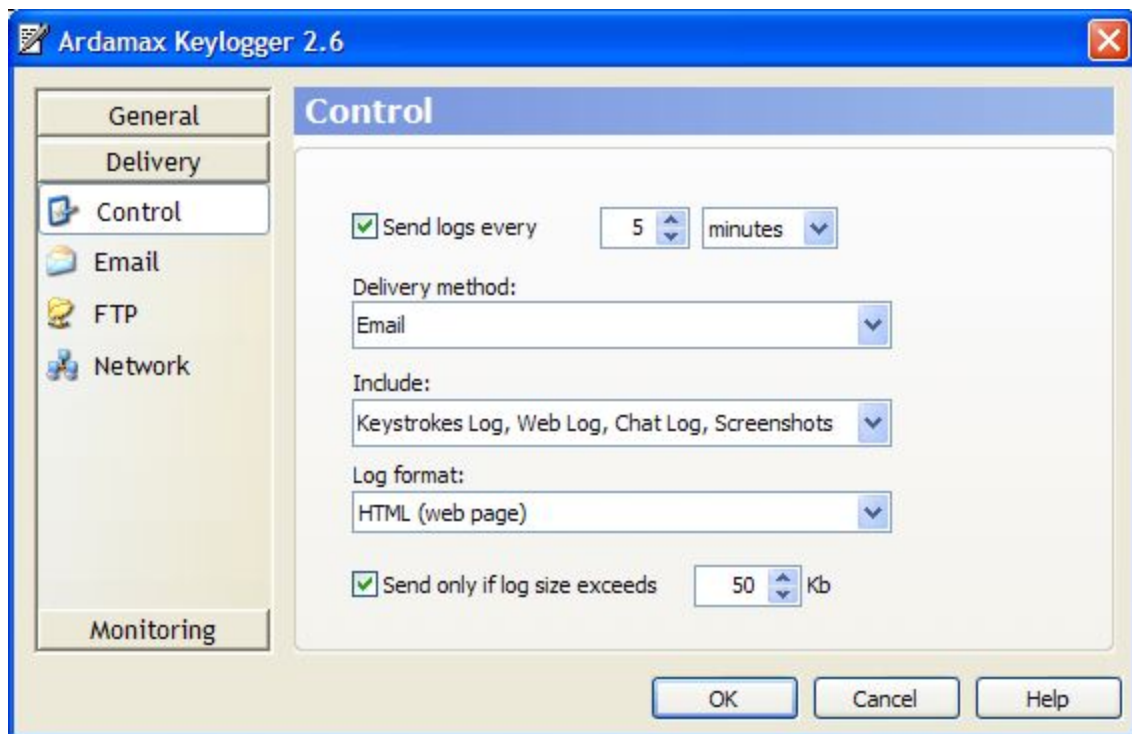
Malware analysis is usually done in two steps :-

- **Static Analysis** : It involves analysing the malware file without actually executing it. The insights are obtained by studying indicators like hashes, strings, IP addresses, headers in the file. The malware might be disassembled to understand the instructions and modifications undertaken by it.
- **Dynamic Analysis** : It involves executing the malware code in a controlled and isolated environment called sandbox. The processes spawned by the malware are studied and network communications are intercepted. Behavioural insights are drawn and analysed.

The purpose of this study is to analyse the working of a keylogger and ransomware. We will analyse the malware with sandboxing, identify their API calls, and try to reverse engineer the mechanism of the malware. We will try to apply static and dynamic analysis and further, try to draw parallels between them.

## Keylogger - Ardamax

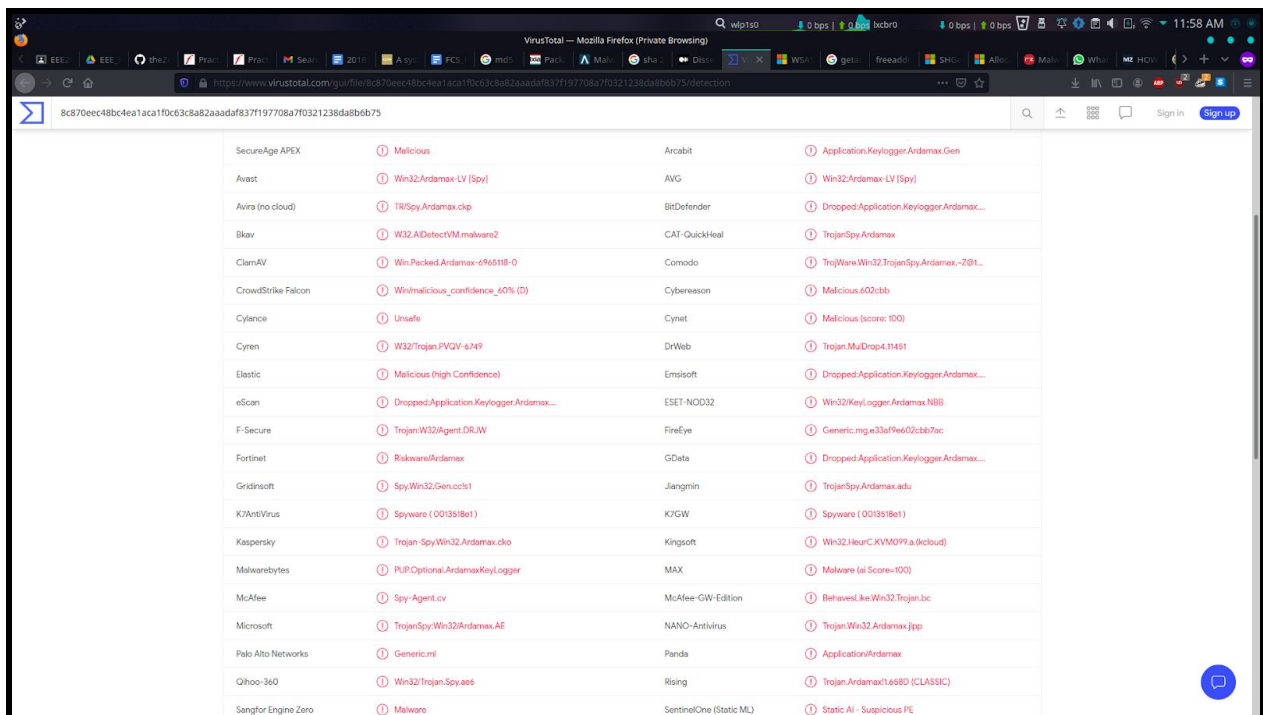
- Ardamax is a keylogger malware which takes periodic screenshots of the screen of the host and records all the logs of the keys pressed.
- It then sends all the recorded data i.e. the key logs and the screenshots through SMTP to a yahoo.mail id which is no longer active, thus the packets are not acknowledged and thus the sending is failed.
- If the host opens its source folder, the process DPBJ.exe which is the main executable process of the Ardamax, is terminated and can be seen in the hidden processes in any process monitor.
- Also, at regular intervals, the logs and screenshots are deleted to avoid suspicion.



Source : <https://www.securitystronghold.com/gates/images/ardamax-keylogger.png>

## Static Analysis

The suspected malware file is run through various antivirus programmes using the website VirusTotal and out of the 68 antivirus engines, 62 flagged it as Ardamax Keylogger, spyware or just as a malware.

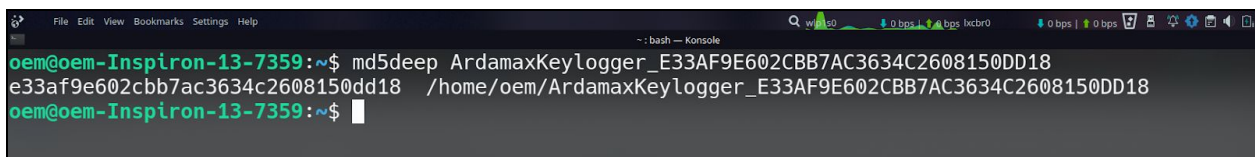


Antivirus Engine	Detection Result
SecureAge APEX	Malicious
Avast	Win32:Ardamax-LV (Spy)
Avira (no cloud)	TR/Spy.Ardamax.ckp
Blav	W32.AIDetect.VM.malware.2
ClamAV	Win.Packed.Ardamax-69651B-Q
CrowdStrike Falcon	Win/malicious_confidence_60% (D)
Cylance	Unsafe
Cyren	W32/Trojan.PQGV-674P
Elastic	Malicious (high Confidence)
eScan	Dropped:Application.Keylogger.Ardamax...
F-Secure	Trojan:Win32/Agent.DR.JW
Fortinet	Riskware/Ardamax
Gridinsoft	Spy.Win32/Genuclot
K7AntiVirus	Spyware (0013518e1)
Kaspersky	Trojan.Spy.Win32.Ardamax.ckp
Malwarebytes	PUP.Optional.ArdamaxKeylogger
McAfee	Spy-Agent.cv
Microsoft	TrojanSpy:Win32/Ardamax.AE
Palo Alto Networks	Generic.mf
Qihoo-360	Win32/Trojan.Spy.a66
Sangfor Engine Zero	Malware
Arcabit	Application.Keylogger.Ardamax.Gen
AVG	Win32:Ardamax-LV (Spy)
BitDefender	Dropped:Application.Keylogger.Ardamax...
CAT-QuickHeal	Trojan.Spy.Ardamax
Comodo	TrojWare.Win32.TrojanSpy.Ardamax-70B1...
Cybereason	Malicious.602cbb
Cynet	Malicious (score: 100)
DrWeb	Trojan.MuDrop.4.11481
Emsisoft	Dropped:Application.Keylogger.Ardamax...
ESET-NOD32	Win32/Keylogger.Ardamax.N8B
FireEye	Generic.mg.e33af9e602cbb7ac
GData	Dropped:Application.Keylogger.Ardamax...
Jiangmin	Trojan.Spy.Ardamax.a6u
K7GW	Spyware (0013518e1)
Kingssoft	Win32.HeurC.KVM099.a(hecloud)
MAX	Malware (ai Score=100)
McAfee-GW-Edition	BehavesLike.Win32.Trojan.bc
NANO-Antivirus	Trojan.Win32.Ardamax.jpp
Panda	Application/Ardamax
Rising	Trojan.Ardamax.1.688D (CLASSIC)
SentinelOne (Static ML)	Static AI - Suspicious PE

Fig. 1: Results for running the malware file through several antivirus programs

Now, the md5deep hash function is used to calculate the hash of the malware. This hash can be used to uniquely identify the malware and acts as a fingerprint for the malware file.

**md5deep hash : e33af9e602cbb7ac3634c2608150dd18**



```
oem@oem-Inspiron-13-7359:~$ md5deep ArdamaxKeylogger_E33AF9E602CBB7AC3634C2608150DD18
e33af9e602cbb7ac3634c2608150dd18 /home/oem/ArdamaxKeylogger_E33AF9E602CBB7AC3634C2608150DD18
oem@oem-Inspiron-13-7359:~$
```

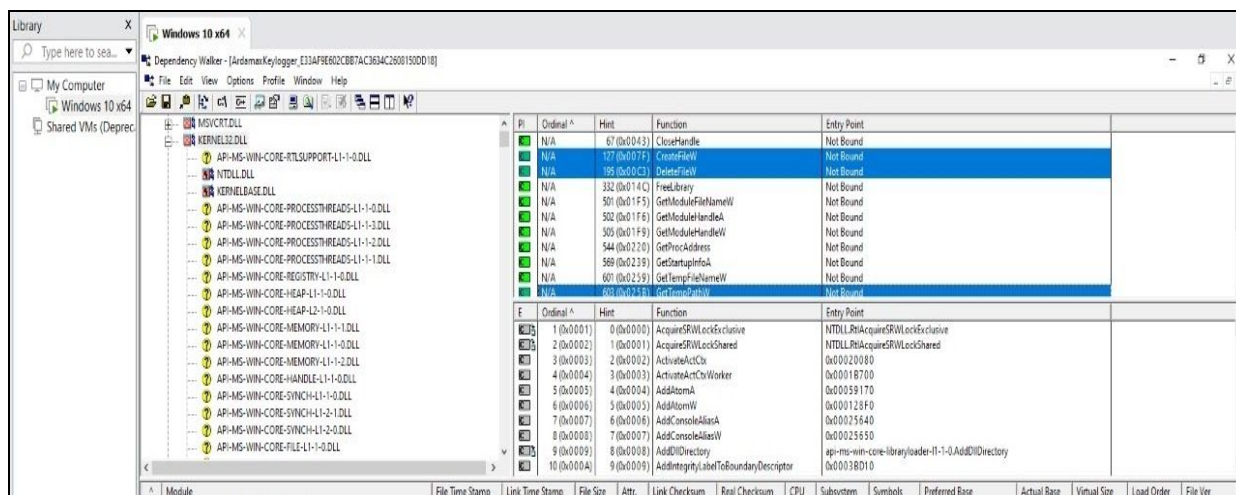
On searching this hash on google, we see that this hash indeed belongs to **Ardamax Keylogger**. On analysing the strings present in the keylogger, some of the interesting observations are:

```
ReadFile
WriteFile
DeleteFileW
FreeLibrary
GetTempFileNameW
CloseHandle
SetFilePointer
CreateFileW
GetModuleFileNameW
GetModuleHandleW
GetTempPathW
GetModuleHandleA
GetStartupInfoA
KERNEL32.dll
MessageBoxW
USER32.dll
?AVILoader@@
?AVICLloader@@
?AV?SCStreamDecompress@VCWin32FileReader@VCWin32FileWriter@VCStreamSimpleCallback@@@
?AVtype_info@@
www.wp
www.wwwwwwwwwp
```

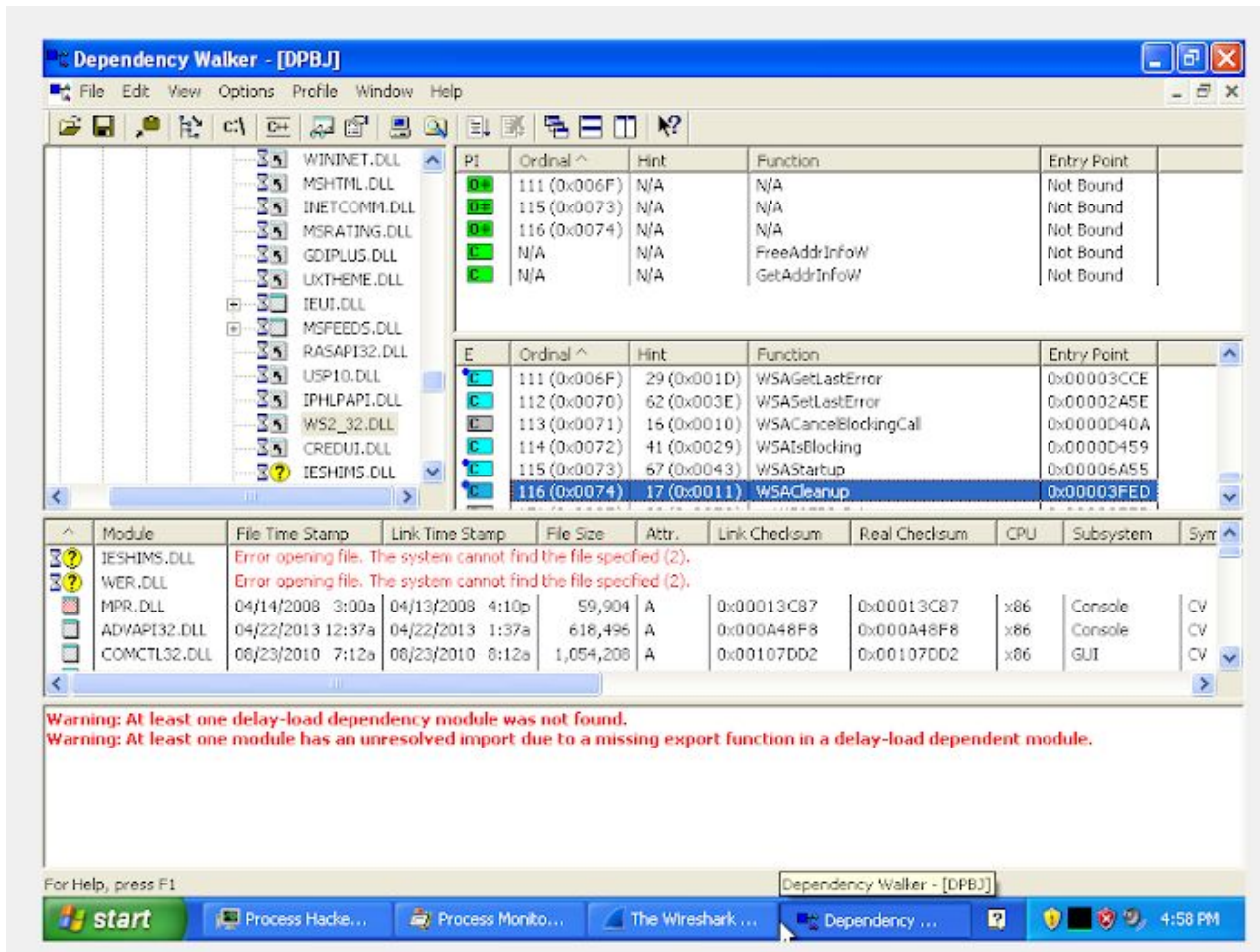
Moving on to further analysis of the malware, the malware is loaded in the *Dependency walker* to see the linked functions present in the executable.

On analysing the imports from *KERNEL32.DLL* we see an interesting function namely, *CreateProcessAsUserA*, the presence of this function in the executable suggests that the **executable may create another process**, so we will watch out for creation of a new process during dynamic analysis.

On further analysis, we see *GetTempPathW*, *CreateFile*, *DeleteFile*, *WriteFile* API calls. We need to check whether the **malware reads or writes to any file in the temp folder**.



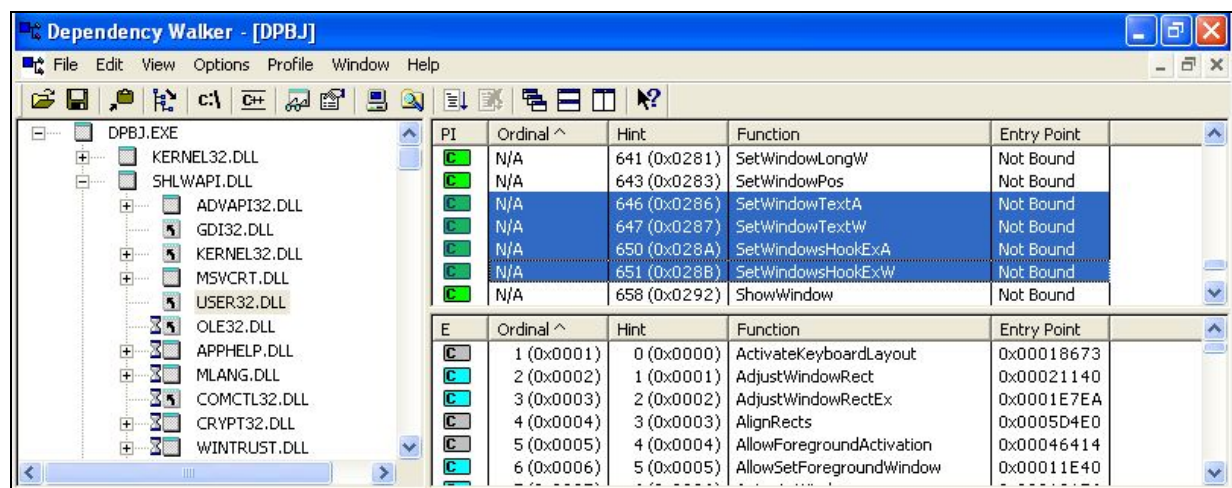




some useful imports from *USER32.DLL* found are-

- **SetCapture** - to capture the mouse input.
- **SetWindowTextW** - to set text in the window.
- **SetWindowsHookExW** - sets a function to be called whenever an event (ex - say mouse input) occurs. Commonly used with keyloggers. Most common way that keyloggers receive keyboard input.





Since most of the antivirus softwares flagged the program as malicious, the malware is probably a keylogger.

## Dynamic analysis

Goal : To observe how the malware executes and what all changes it makes on the host system.

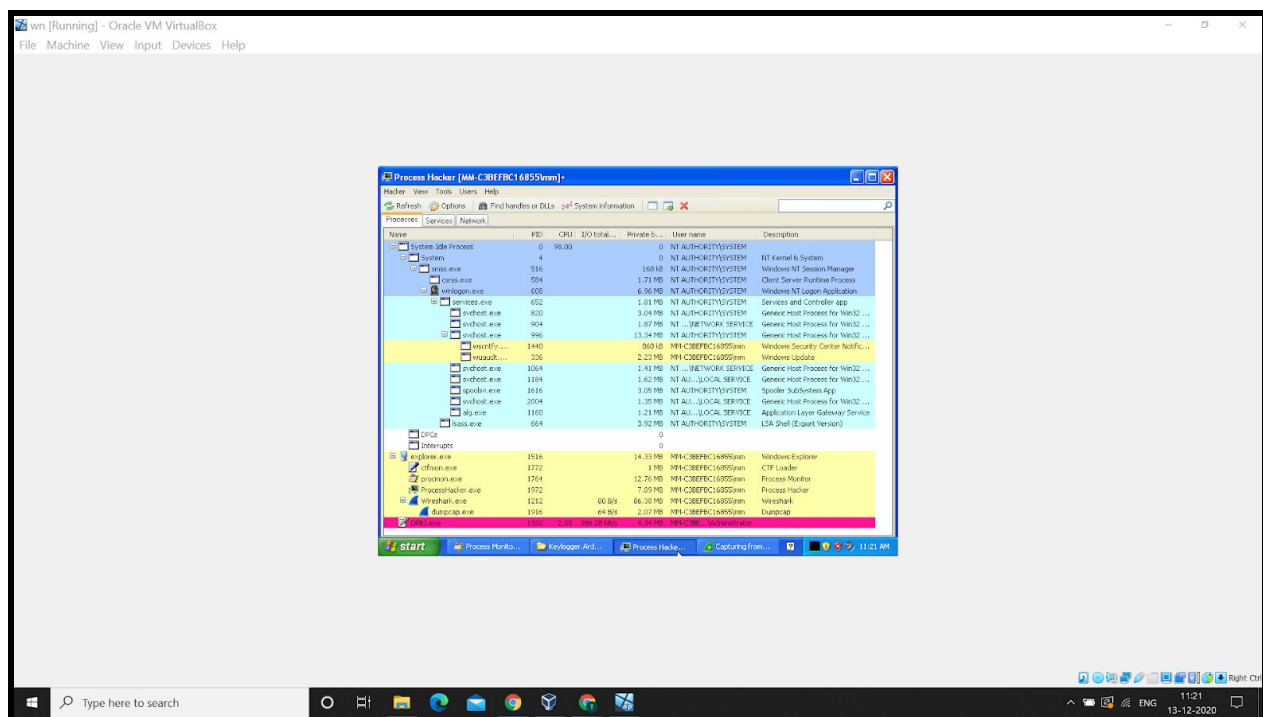
First we activate the process monitors, like *procmom* and *process hacker*, which can track and record all the processes which are being executed and packet tracers like Wireshark to capture the packets sent from the host system by the malware.

We set up a virtual device with *Windows XP* in it so that there are not as many background processes running as the newer versions. And once we close/terminate all the processes and browsers which might send packets outside, we enable the capture process of wireshark and process tracers and execute the malware( here, keylogger).

Here, we analyse the keylogger Ardamax, which along with the keylogs also traces and sends the screenshots of the screen at a regular interval through *SMTP* protocol.

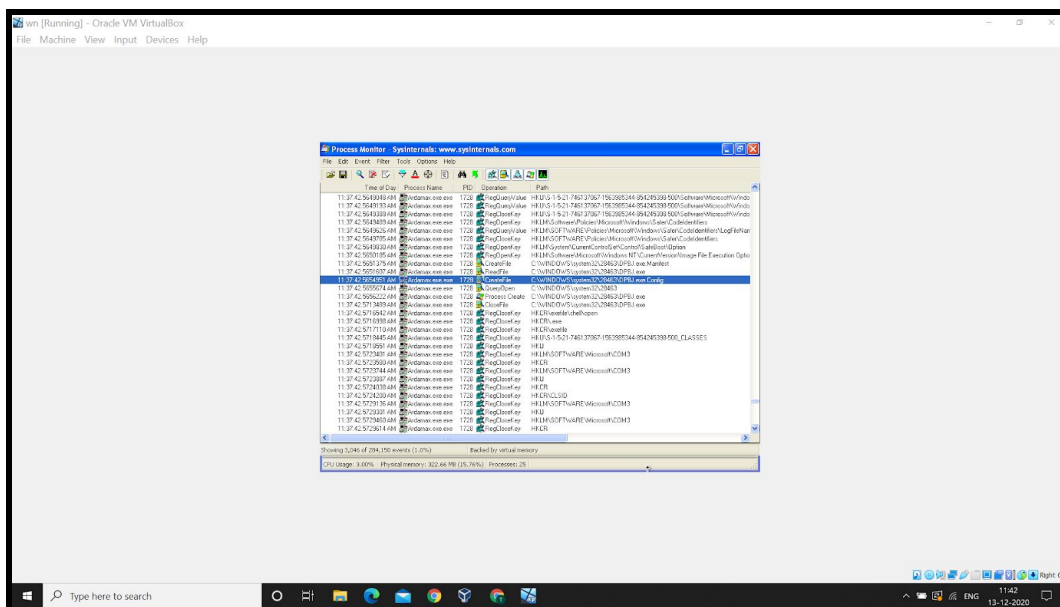
Initially, there is no activity from the keylogger for a few seconds, but once some keys are pressed, the malware starts running and we can see its presence in the process Hacker.

In *Process Hacker* the newer processes are coloured red whereas the already running processes turn blue with time.

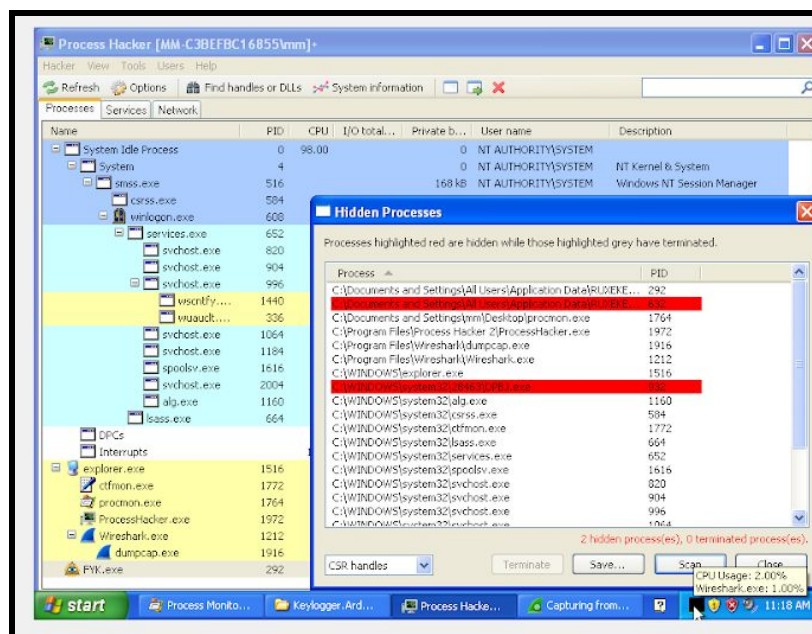




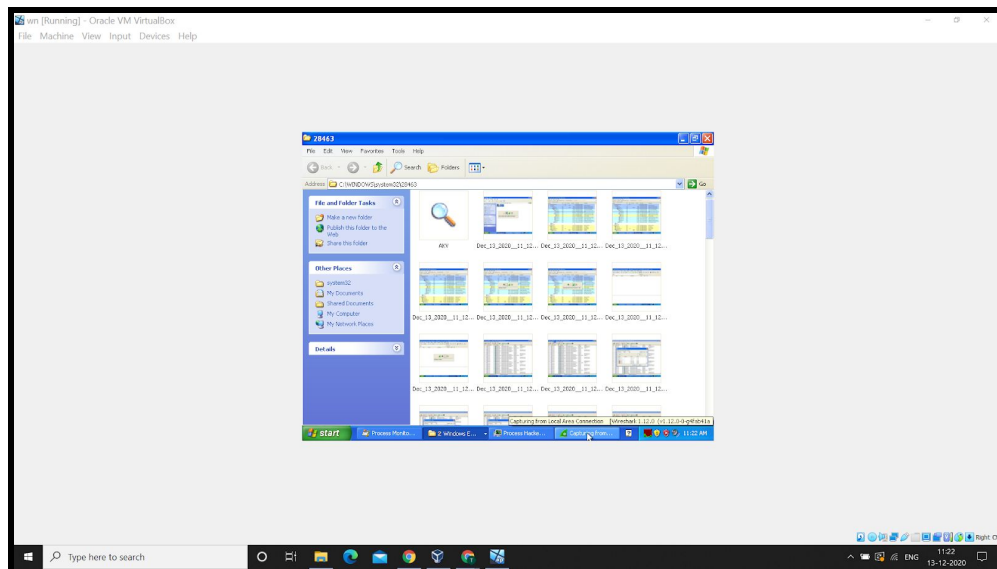
Once the executable file *Ardamax.exe* is run, it forks and the child process is named *PDBJ.exe*. The main executable for the keylogger is *PDBJ.exe* which makes all the moves like recording the key logs and taking continuous screenshots.



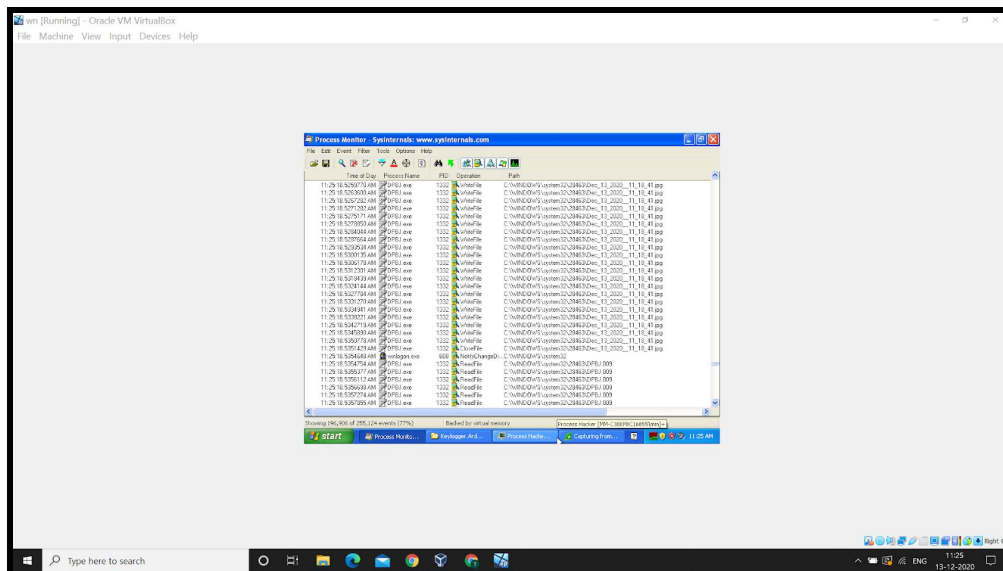
When the keylogger is not capturing the key logs and screenshots, it hides itself and can be traced in the hidden processes. Also, the Ardamax keylogger is run by the name of *PDBJ.exe* and thus can be seen in the active/running processes.



In the *Process Hacker*, we can access the folder in which the process is storing its files. When we open the folder, we can see the media stored in it and after a specific interval, it deletes the screenshots on its own.

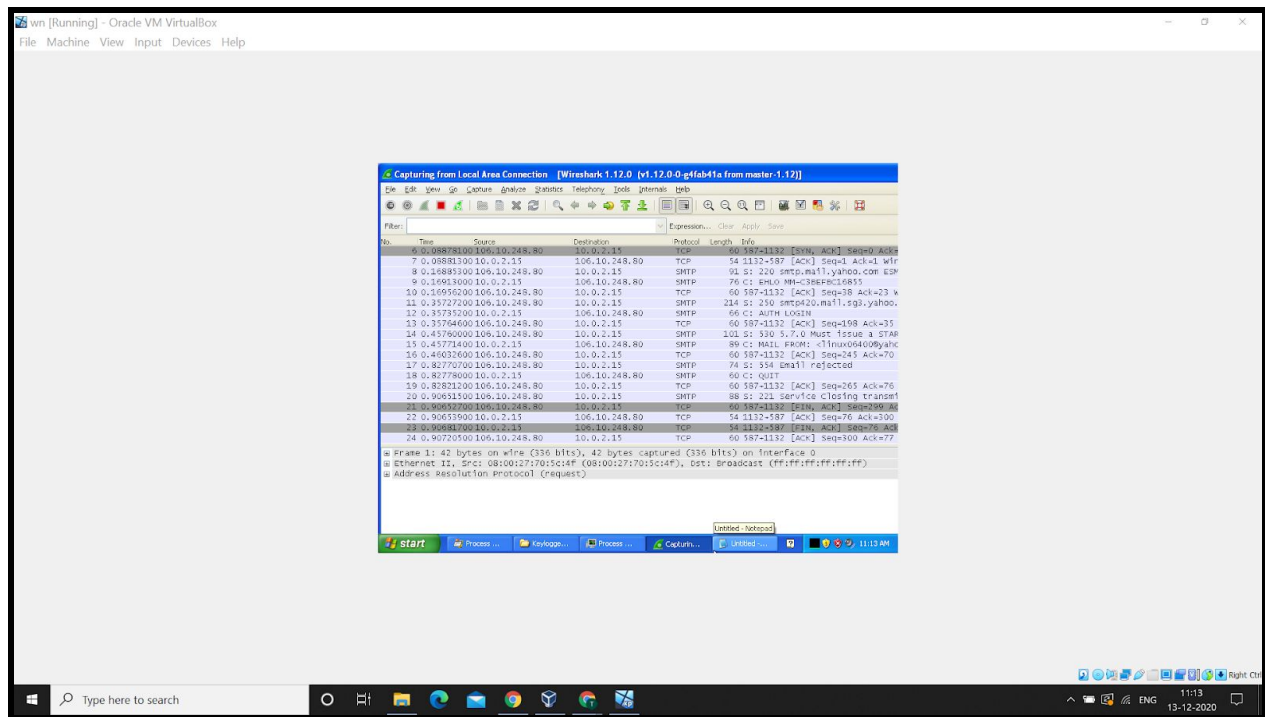


We can confirm that the screenshots are written by the process PDBJ.exe and not already there by finding the “write” operations in the logs of procmon. We can even confirm the destination of the files written which is the same as the root folder of PDBJ.exe.

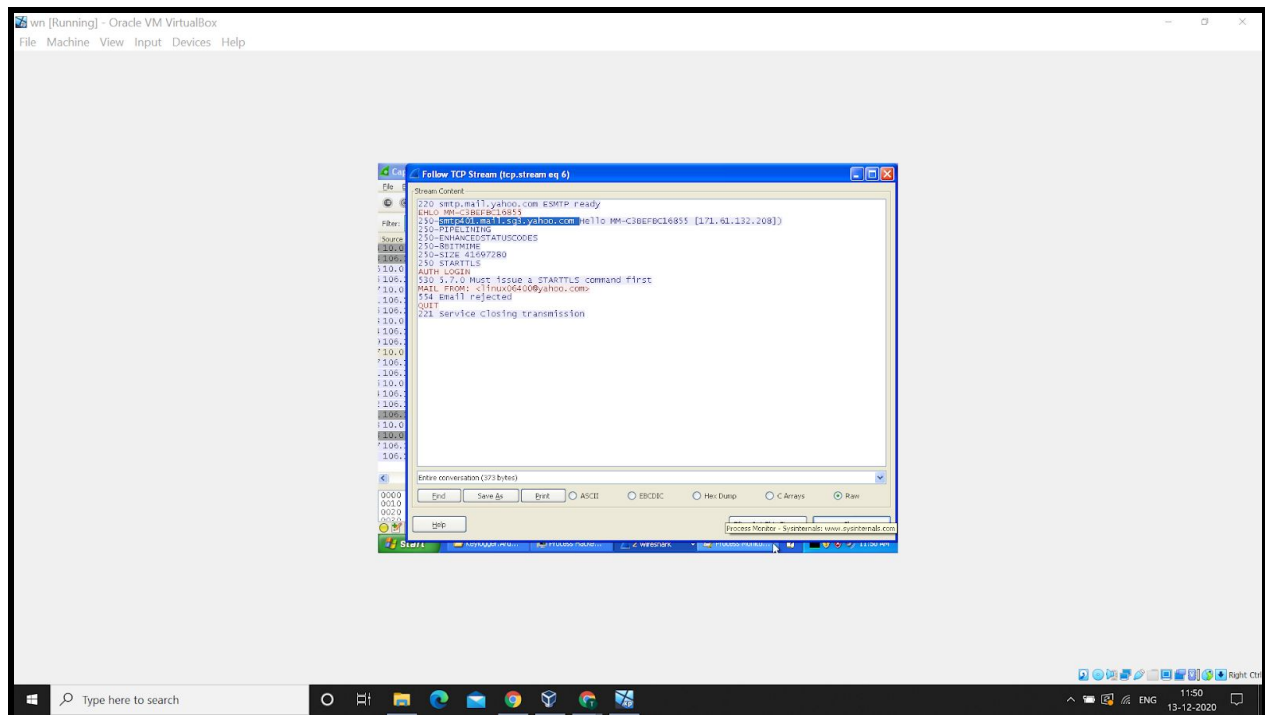


We can confirm that the malware is modifying the contents of the host system stealthily by finding the parent process modifying, erasing and writing temporary files in the temp folder.



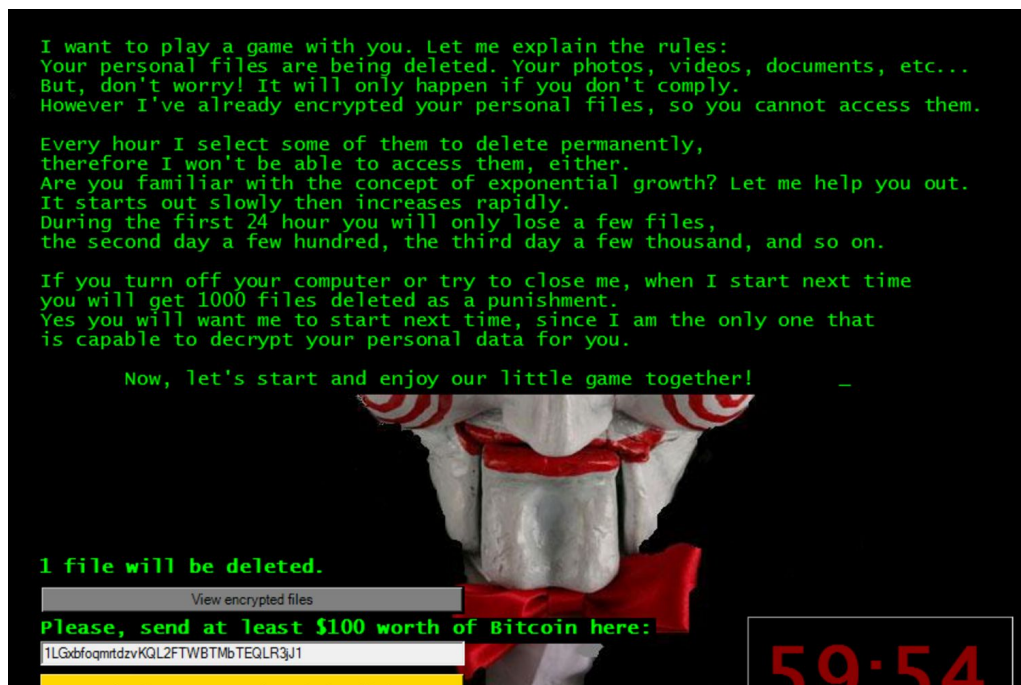


We can see the destination IP address in Wireshark along with a lot of more packet details by following the TCP stream.



## Ransomware - JigSaw

- Jigsaw is an encrypting ransomware malware which was created in 2016. It was initially called "BitcoinBlackmailer" but later became famous as Jigsaw as it featured an image of Billy the Puppet from the Saw film franchise.
- Jigsaw primarily spreads through attachments in spam mails and is activated if the user downloads the attachment.
- Once activated it prompts the user with an intimidating message specifying how the user files have been encrypted and the only way to retrieve them is by paying the required ransom.
- Unlike other ransomwares, JigSaw uses a countdown timer to worsen the situation for the user. Every hour the files get deleted incrementally.
- If the user tries to restart the computer, 1000 files are deleted instantly as a punishment and JigSaw continues to execute.

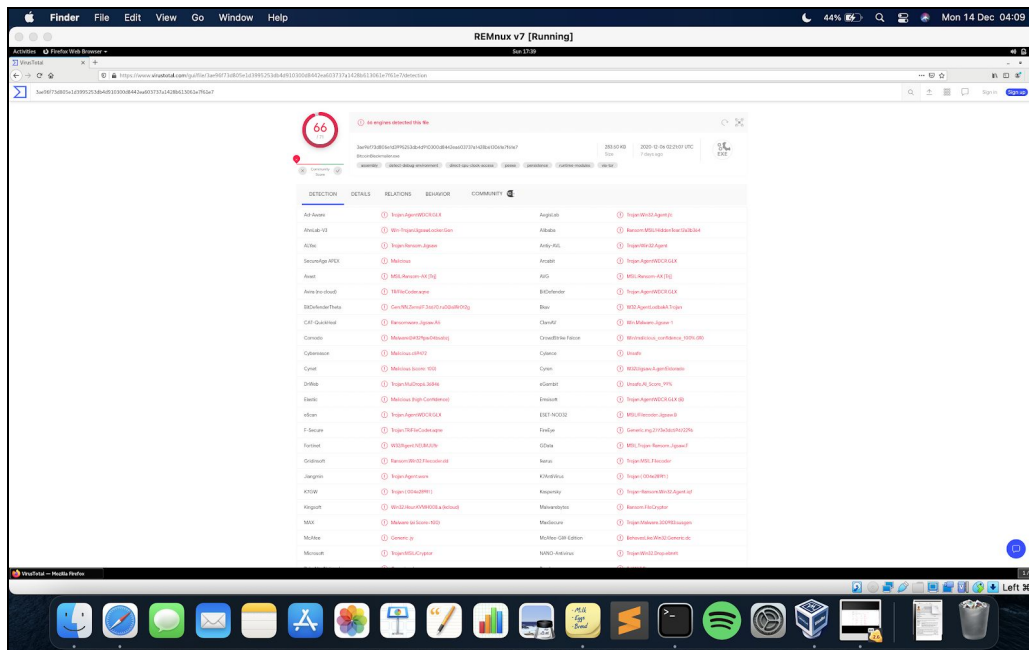


Source: <https://sensorstechforum.com/remove-jigsaw-ransomware-restore-paytounlock-files>



## Static Analysis

The suspected malware file is run through various antivirus programmes using the website VirusTotal and out of the 71 antivirus engines, 66 reported it as a malicious software.



We then calculate the md5deep hash of the jigsaw file. This hash can be used to uniquely identify the malware and acts as a fingerprint for the malware file.

**md5deep hash : 2773e3dc59472296cb0024ba7715a64e**

```
remnux@remnux:~/Downloads$ unzip Ransomware.Jigsaw.zip
Archive:  Ransomware.Jigsaw.zip
[Ransomware.Jigsaw.zip] jigsaw password:
  inflating: jigsaw
remnux@remnux:~/Downloads$ ls
jigsaw  Ransomware.Jigsaw.zip
remnux@remnux:~/Downloads$ md5deep jigsaw
2773e3dc59472296cb0024ba7715a64e  /home/remnux/Downloads/jigsaw
remnux@remnux:~/Downloads$
```



On searching for the hash online, we come across various malware reports for this file. We derive some important insights from the VirusTotal website.

Using the VirusTotal API, we get some more information about the ransomware. Here we note that mscoree library is imported by the jigsaw program.

```
remnux@remnux:~/Downloads$ vt -fi jigsaw | tail -7

[+] Imports
    mscoree.dll
    0x44e000 _CorExeMain

[+] File type
    PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
remnux@remnux:~/Downloads$
```

Now we run strings and pestr on the jigsaw file to identify some crucial parts of the file. We lookup possible IP addresses in the file using a regex and also note the different libraries being used by the jigsaw.

```
remnux@remnux:~/Downloads$ cat malware_strings.txt | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"
4.0.0.0
14.0.0.0
1.0.0.0
remnux@remnux:~/Downloads$ cat malware_strings.txt | grep dll
mscorlib.dll
kernel32.dll
user32.dll
costura.newtonsoft.json.dll.zip
remnux@remnux:~/Downloads$
```

We found an alias name drpbx.exe which might be used by ransomware to hide.

```
{ file = {0}, fi = {1} }}
Congratulations. Your software has been registered. Confirmation code 994759
Email us this code in the chat to active your software. It can take up to 48 hours.
Thank you
Drpbx\drpbx.exe
rfx\firefox.exe
System32Work\
```

Another interesting thing is how the file has been given the name of Firefox while its original name is actually BitcoinBlackmailer.exe . Thus, we should watch out for any Firefox processes in the dynamic analysis.

```
Activities Terminal - Thu 07:50
remnux@remnux: ~/Downloads/static$
KMicrosoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator
14.0.0.0
VS VERSION INFO
VarFileInfo
Translation
StringFileInfo
000004b0
Comments
CompanyName
FileDescription
Firefox
FileVersion
37.0.2.5583
InternalName
BitcoinBlackmailer.exe
LegalCopyright
Copyright 1999-2012 Firefox and Mozilla developers. All rights reserved.
LegalTrademarks
OriginalFilename
BitcoinBlackmailer.exe
ProductName
Firefox
ProductVersion
37.0.2.5583
Assembly Version
37.0.2.5583
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
remnux@remnux: ~/Downloads/static$
```

We observe the bitcoin network through which the ransom is to be paid.

```
Activities Terminal - Thu 07:50
remnux@remnux: ~/Downloads/static$
buttonViewEncryptedFiles
view encrypted files
ucida Sans Unicode
labelCountDown
59:59
labelFilesToDelete
file will be deleted.
FormGame
ain.Properties.Resources
xtensionsToEncrypt
Jigsaw
StartModeDebug
vanityAddresses
http://btc.blockr.io/api/v1/
coin/info/
status
error
data
markets
coinbase
value
address/balance/
balance
DeleteItself.bat
del "{0}"
if exist "{0}" goto del
del %0
fileSystemSimulation
txtTest.txt
am a txt test.
otxtTest.nottxt
am NOT a txt test.
C:\Windows
pIsAwWf23cICQoLDA00De==
ncryptedFileList.txt
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
.zip
{0}. {1}
newtonsoft.json
```

Some other interesting strings in the file are as follows :-

```
System.Threading
NotSupportedException
System.Collections.Generic.IEnumerator<System.String>.Current
System.Collections.IEnumerator.Current
<>c__DisplayClass9_0
<>g__0
<EncryptFiles>b__0
Windows
HWND_TOPMOST
SWP_NOSIZE
SWP_NOMOVE
SetStartup
startupMethod
SetStartupFolder
SetStartupRegistry
exePath
RemoveStartupRegistry
SetWindowPos
user32.dll
hwnd
hwndInsertAfter
uFlags
MakeTopMost
RegistryKey
Microsoft.Win32
StartupMethodType
StartupFolder
Registry
<PrivateImplementationDetails>
FBD2112E56A5379083D53B084795822F604F11FC
__StaticArrayInitTypeSize=16
AssemblyLoader
Costura
nullCache
Dictionary`2
assemblyNames
symbolNames
CS$<>9__CachedAnonymousMethodDelegate1
ResolveEventHandler
CultureToString
culture
ReadExistingAssembly
AssemblyName
name
```

```
mscoree.dll
b0jB
v2.0.50727
#Strings
#GUID
#Blob
iST5
^:T5
y:T5
*<T5
.@T5
:GT5
7P(P
/Z:Z
^hfh
u^h^
BitcoinBlackmailer.exe
<module>
mscorlib
Assembly
System.Reflection
cctor
ResolveEventArgs
System
VirtualProtect
kernel32.dll
ValueType
cctor
Object
Stream
System.IO
<>f__AnonymousType0`2
<file>j__TPar
<ext>j__TPar
<file>i__Field
<ext>i__Field
get_file
get_ext
file
Equals
value
GetHashCode
ToString
IFormatProvider
```

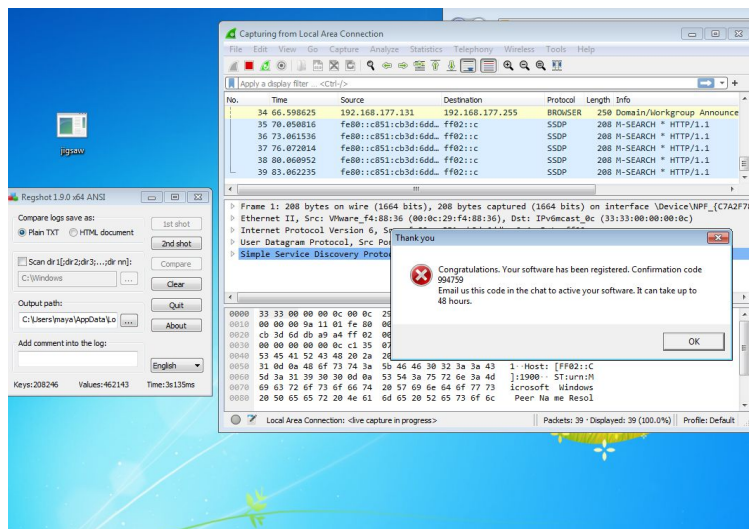
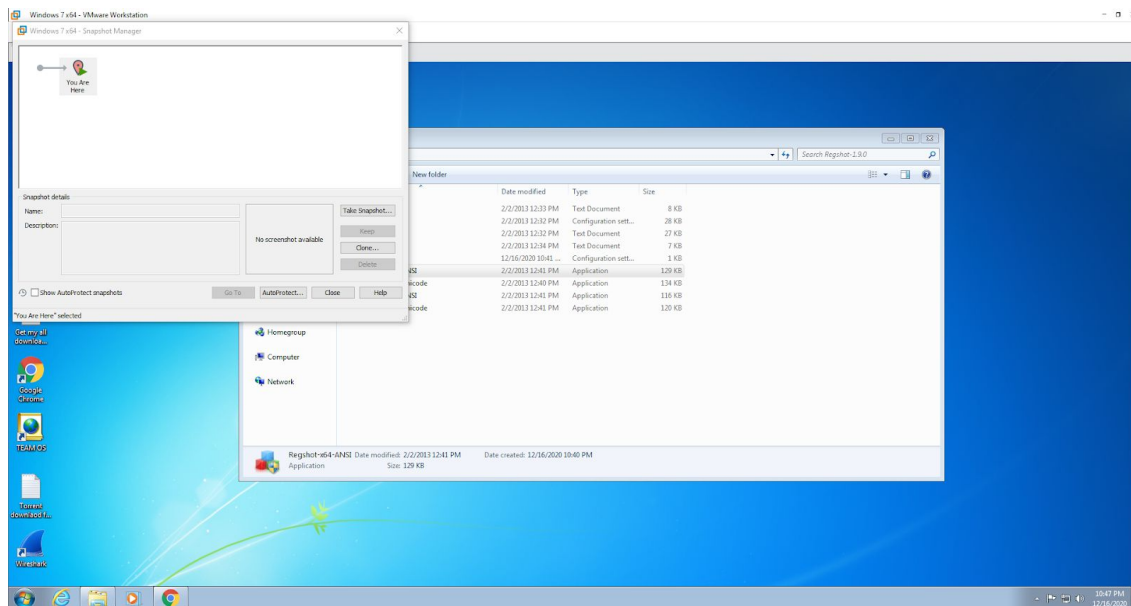
We ran pescan over jigsaw and found two suspicious sections in the file.

```
remnux@remnux:~/Downloads$ pescan -v jigsaw
file entropy: 7.677662 (probably packed)
fpu anti-disassembly: no
imagebase: normal - 0x400000
entrypoint: normal - va: 0x4e00a - raw: 0x46c0a
DOS stub: normal
TLS directory: not found
timestamp: normal - Thu, 31 Mar 2016 06:28:14 UTC
section count: 5
sections
  section
    mmUPp`B  suspicious name, self-modifying
  section
    .text: normal
  section
    .rsrc: normal
  section
    .reloc: small length
  section
    : suspicious name, small length
remnux@remnux:~/Downloads$
```

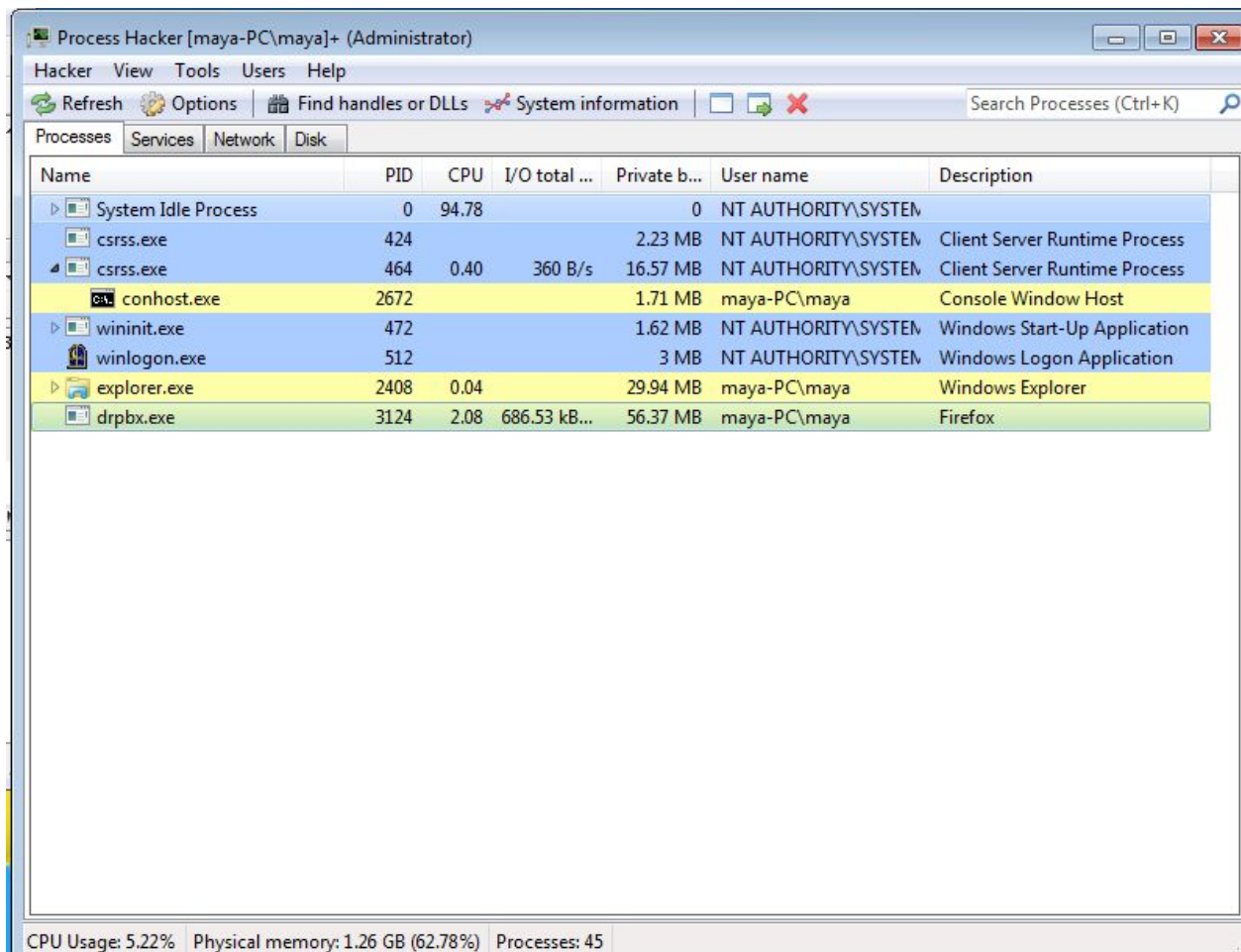
## Dynamic Analysis

We started dynamic analysis to confirm all the things we found in static analysis and try to find if we would be able to undo encryption of the ransomware

We installed the necessary tools for dynamic analysis and created a snapshot of the virtual machine.



This is the activation prompt for the malware as when it starts executing.



Name	PID	CPU	I/O total ...	Private b...	User name	Description
System Idle Process	0	94.78		0	NT AUTHORITY\SYSTEM	
csrss.exe	424			2.23 MB	NT AUTHORITY\SYSTEM	Client Server Runtime Process
csrss.exe	464	0.40	360 B/s	16.57 MB	NT AUTHORITY\SYSTEM	Client Server Runtime Process
conhost.exe	2672			1.71 MB	maya-PC\maya	Console Window Host
wininit.exe	472			1.62 MB	NT AUTHORITY\SYSTEM	Windows Start-Up Application
winlogon.exe	512			3 MB	NT AUTHORITY\SYSTEM	Windows Logon Application
explorer.exe	2408	0.04		29.94 MB	maya-PC\maya	Windows Explorer
drpbx.exe	3124	2.08	686.53 kB...	56.37 MB	maya-PC\maya	Firefox

CPU Usage: 5.22% Physical memory: 1.26 GB (62.78%) Processes: 45

We see a new process created with the name drpbx.exe so as to hide the process. We also see some IO interaction which may be the locking down the system.





KEY IDEAS OF RANSOMWARE:-

LOCKS OUR SYSTEM WITH CONVECTION CRYPTOGRAPHIC ALGORITHM IF WE GET THE KEY WE GET THE DATA BACK.

We have 2 ideas to get the key:-

One is sent after payment is done but there is no such network traffic.

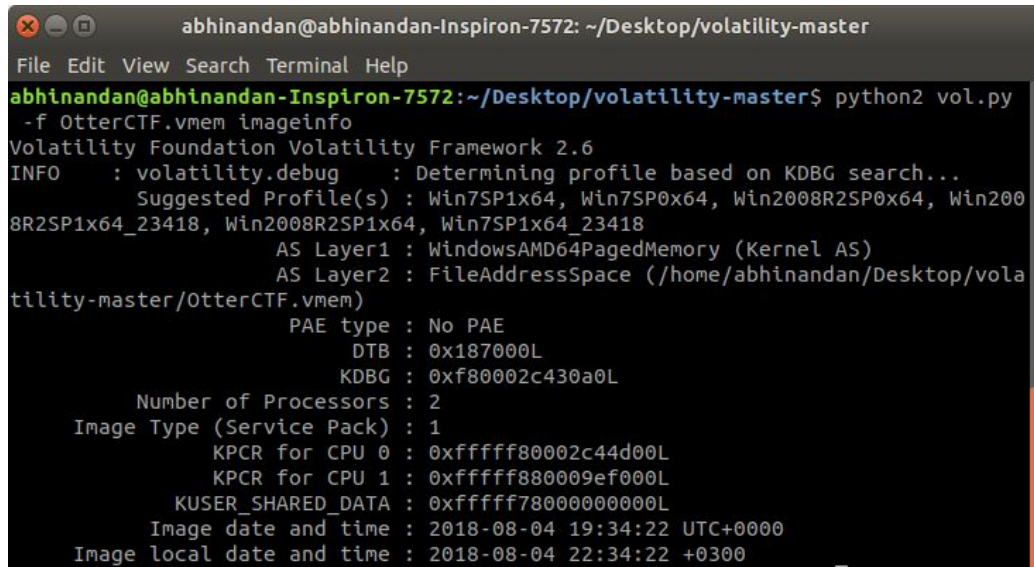
This ransomware has a private key with it so we can use a decompiler for finding the key.

Also looking into the code we can see the Ransomware is using AES.

## Solutions To OtterCTF

### Pre-Analysis

Downloading the .vmem file from OtterCTF and running Volatility Framework.

A terminal window titled 'abhinandan@abhinandan-Inspiron-7572: ~/Desktop/volatility-master' showing the execution of 'python2 vol.py -f OtterCTF.vmem imageinfo'. The output displays system information for a Windows 7 SP1 x64 system, including suggested profiles, AS layers, PAE type, DTB, KDBG, number of processors, image type, KPCR values, and timestamps.

```
abhinandan@abhinandan-Inspiron-7572: ~/Desktop/volatility-master
File Edit View Search Terminal Help
abhinandan@abhinandan-Inspiron-7572:~/Desktop/volatility-master$ python2 vol.py
-f OtterCTF.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win200
8R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
      AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/abhinandan/Desktop/vola
tility-master/OtterCTF.vmem)
      PAE type : No PAE
      DTB : 0x187000L
      KDBG : 0xf80002c430a0L
      Number of Processors : 2
      Image Type (Service Pack) : 1
      KPCR for CPU 0 : 0xffffffff80002c44d00L
      KPCR for CPU 1 : 0xffffffff800009ef000L
      KUSER_SHARED_DATA : 0xffffffff78000000000L
      Image date and time : 2018-08-04 19:34:22 UTC+0000
      Image local date and time : 2018-08-04 22:34:22 +0300
```

The suggested profile is Win7SP1x64. Listing the registries:

```
abhinandan@abhinandan-Inspiron-7572: ~/Desktop/volatility-master
File Edit View Search Terminal Help
abhinandan@abhinandan-Inspiron-7572:~/Desktop/volatility-master$ python2 vol.py
-f OtterCTF.vmem hivelist --profile=Win7SP1x64
Volatility Foundation Volatility Framework 2.6
Virtual Physical Name
-----
0xfffff8a00377d2d0 0x00000000624162d0 \??\C:\System Volume Information\Syscache.
hve
0xfffff8a00000f010 0x000000002d4c1010 [no name]
0xfffff8a000024010 0x000000002d50c010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000053320 0x000000002d5bb320 \REGISTRY\MACHINE\HARDWARE
0xfffff8a000109410 0x0000000029cb4410 \SystemRoot\System32\Config\SECURITY
0xfffff8a00033d410 0x000000002a958410 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a0005d5010 0x000000002a983010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a001495010 0x0000000024912010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a0016d4010 0x00000000214e1010 \SystemRoot\System32\Config\SAM
0xfffff8a00175b010 0x00000000211eb010 \??\C:\Windows\ServiceProfiles\NetworkServ
ice\NTUSER.DAT
0xfffff8a00176e410 0x00000000206db410 \??\C:\Windows\ServiceProfiles\LocalServic
e\NTUSER.DAT
0xfffff8a002090010 0x00000000b92b010 \??\C:\Users\Rick\ntuser.dat
0xfffff8a0020ad410 0x00000000db41410 \??\C:\Users\Rick\AppData\Local\Microsoft\
Windows\UsrClass.dat
```

CTF Problem: Get the computer's IP address.

```
abhinandan@abhinandan-Inspiron-7572: ~/Desktop/volatility-master
File Edit View Search Terminal Help
abhinandan@abhinandan-Inspiron-7572:~/Desktop/volatility-master$ python2 vol.py
-f OtterCTF.vmem netscan --profile=Win7SP1x64
Volatility Foundation Volatility Framework 2.6
Offset(P) Proto Local Address Foreign Address
State Pid Owner Created
0x7d60f010 UDPv4 0.0.0.0:1900 *: *
2836 BitTorrent.exe 2018-08-04 19:27:17 UTC+0000
0x7d62b3f0 UDPv4 192.168.202.131:6771 *: *
2836 BitTorrent.exe 2018-08-04 19:27:22 UTC+0000
0x7d62f4c0 UDPv4 127.0.0.1:62307 *: *
2836 BitTorrent.exe 2018-08-04 19:27:17 UTC+0000
0x7d62f920 UDPv4 192.168.202.131:62306 *: *
2836 BitTorrent.exe 2018-08-04 19:27:17 UTC+0000
0x7d6424c0 UDPv4 0.0.0.0:50762 *: *
4076 chrome.exe 2018-08-04 19:33:37 UTC+0000
0x7d6b4250 UDPv6 ::1:1900 *: *
164 svchost.exe 2018-08-04 19:28:42 UTC+0000
0x7d6e3230 UDPv4 127.0.0.1:6771 *: *
2836 BitTorrent.exe 2018-08-04 19:27:22 UTC+0000
0x7d6ed650 UDPv4 0.0.0.0:5355 *: *
620 svchost.exe 2018-08-04 19:34:22 UTC+0000
0x7d71c8a0 UDPv4 0.0.0.0:0 *: *
868 svchost.exe 2018-08-04 19:34:22 UTC+0000
0x7d71c8a0 UDPv6 :::0 *: *
```

**CTF Problem: Guess the name and address of the game Rick is playing?**

70x7e413d40	TCPv4	708	LunarMS.exe	192.168.202.131:50346	89.64.10.176:10589
70x7e415010	TCPv4	2836	BitTorrent.exe		

LunarMS is the name of the game. And its address is 77.102.199.102

**CTF Problem: Since Rick pastes the password from the clipboard, we have to get the clipboard items.**

Password is M@il\_PrOvid0rs

```
abhinandan@abhinandan-Inspiron-7572: ~/Desktop/volatility-master
File Edit View Search Terminal Help
abhinandan@abhinandan-Inspiron-7572:~/Desktop/volatility-master$ python2 vol.py
-f OtterCTF.vmem clipboard --profile=Win7SP1x64
Volatility Foundation Volatility Framework 2.6
Session WindowStation Format Handle Object
Data
-----
1 WinSta0 CF_UNICODETEXT 0x602e3 0xffffffff900c1ad93f
M@il_PrOvid0rs
1 WinSta0 CF_TEXT 0x10 -----
1 WinSta0 0x150133L 0x2000000000000000 -----
1 WinSta0 CF_TEXT 0x1 -----
1 ----- 0x150133 0xffffffff900c1c1adc
```

**CTF Problem: finding suspicious processes**



Vmware-tray.exe is a child process of Rick and Morty, which is very suspicious.

0xfffffa801b486b30	Rick And Morty	3820	2728	4	185	1	1	2018-08-04	19:32:55	UTC+0000
0xfffffa801a4c5b30	vmware-tray.exe	3720	3820	8	147	1	1	2018-08-04	19:33:02	UTC+0000