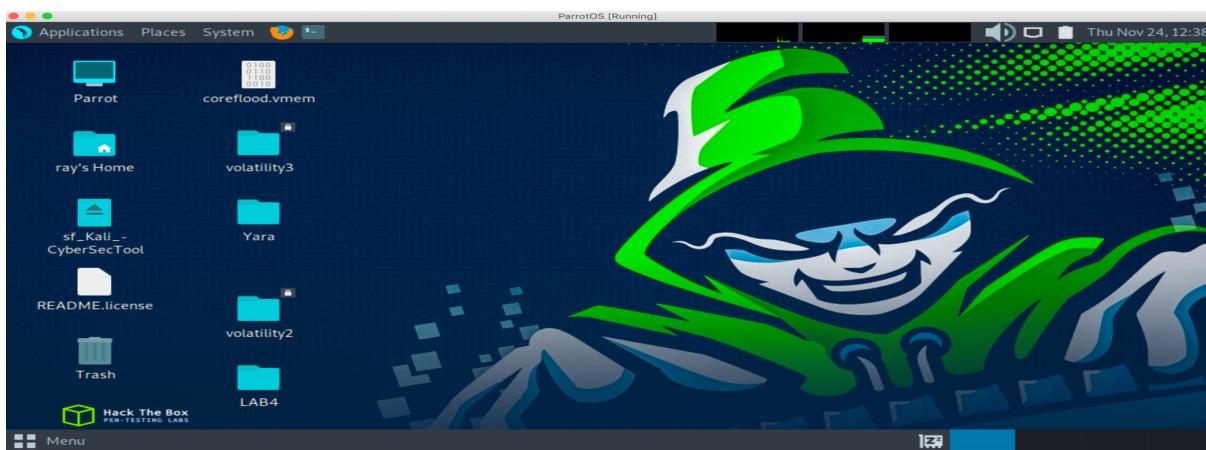


Introduction

In this lab, we are going to perform malware analysis with various detection techniques and find the intended behaviours in each one of the malware programs. Moreover, knowing how to write Yara rules and detect malware through Yara are vital to this lab's objective. And, this lab covers DNS, Hash, Windows API, Persistent maintenance and other malicious indicators.

As to Virtual Machines and Techniques, we use the ParrotOS HTB version, and SSH Port forwarding connects to the machine.



Objective

Objective Malware Files:

1. njRAT-v0.6.4.zip
2. infected.7z
3. Group File **PWN Group**:

PWN	e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.zip
-----	--

4. e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.zip

Extract Password: infected

The extracted files can be observed in the below screenshot.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ ls
Lazarus.yar
Qakbot.yar
RAT_Njrat.yar
crime_wannacry.yar
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.zip
general_rats_malwareconfig.yar
infected.7z
jRAT.yar
malwaresamples
njRAT-v0.6.4
njRAT-v0.6.4.zip
redline_steaлер.yar
[ray@parrot] -[~/Desktop/LAB]
└─$ _
```

Task 1: Create a Yara rule for njRAT malware

Find the clue of Indicators of Compromise (IOC) through Yara rules.

The hash value of njRAT.exe is **0431311b5f024d6e66b90d59491f2563**, and the Virustotal indicates that the hash value has been recognized as a malicious file in 62 out of 71 vendors.

```
[ray@parrot] ~ [~/Desktop/LAB/njRAT-v...]
$ md5sum njRAT.exe
0431311b5f024d6e66b90d59491f2563   njRAT.exe
[ray@parrot] ~ [~/Desktop/LAB/njRAT-v...]
$
```

62 / 71

Community Score

Basic Properties

MD5	0431311b5f024d6e66b90d59491f2563
SHA-1	e9ff4da7e3f2199bc1b16d37d8935cb1b0567ac2c2
SHA-256	fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199
Vhash	EnKSaR.HaCKeR.exe
Authentihash	2950465015151f01011c3952314100
ImpHash	d6fde461f0fecbe9551fcd5db1453c3c687876de8033be73992789f0fb5c4
SSDIEEP	f34d5f2d4577ed6d9ceec516cf5fa744
TLSH	12288+0vxE3J7JO-xPuc/9wvAmv6SAbnzmp2hGnadjFM4ZHOT2:+eXuczPCSGnVjad1
File type	T1D4252A233744FC28C03971F6A0858FD481A24F1E95B645F925B173BAF5FA6C2AC4A3D9
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit Mono/.Net assembly
TrID	Generic CIL Executable (.NET, Mono, etc.) (48%) Win32 Executable MS Visual C++ (generic) (20.5%) Windows screen saver (8.6%) Win64 Executable (generic) (6.9%) Win32 Dynamic Link Library (generic) (4.3%)
File size	959.00 KB (982016 bytes)
PEID packer	.NET.Packer

History

Creation Time	2013-09-27 08:00:20 UTC
First Seen In The Wild	2013-09-27 08:00:20 UTC
First Submission	2013-10-27 11:15:27 UTC
Last Submission	2022-11-21 22:49:08 UTC
Last Analysis	2022-09-20 13:44:41 UTC

Names

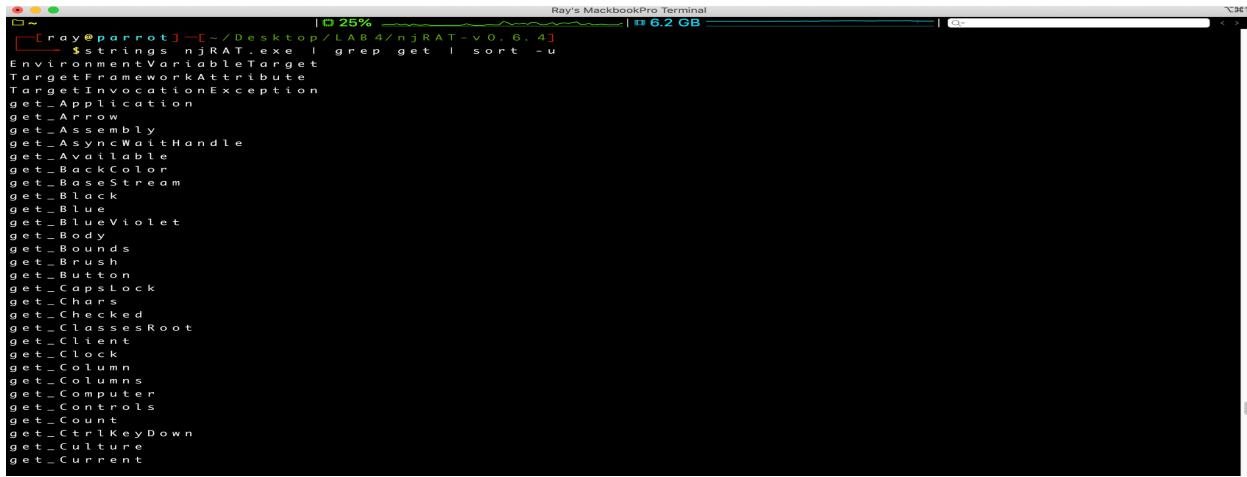
- njRAT.exe

Firstly, we need to know what system of the executable it is via file command.

```
[ray@parrot] ~ [~/Desktop/LAB/njRAT-v...]
$ file njRAT.exe
njRAT.exe: PE 32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
[ray@parrot] ~ [~/Desktop/LAB/njRAT-v...]
$ strings njRAT.exe
```

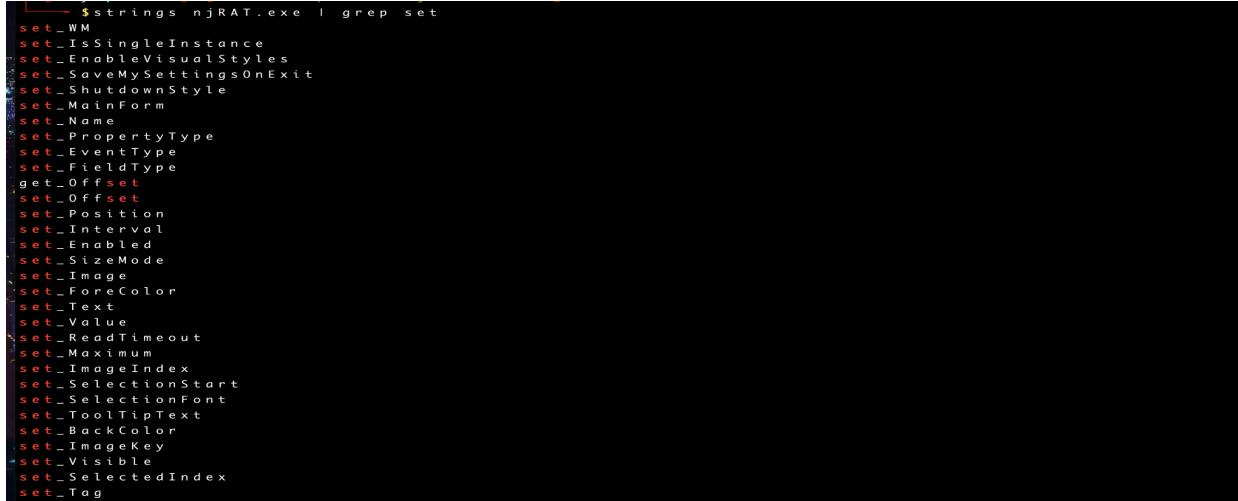
Utilize strings to extract information from the njRAT executable file, and this will show what Windows commands are used and malicious behaviour.

As we can see in the below screenshots, there are many Windows APIs the njRAT executable used [1] [2] [3] [4].

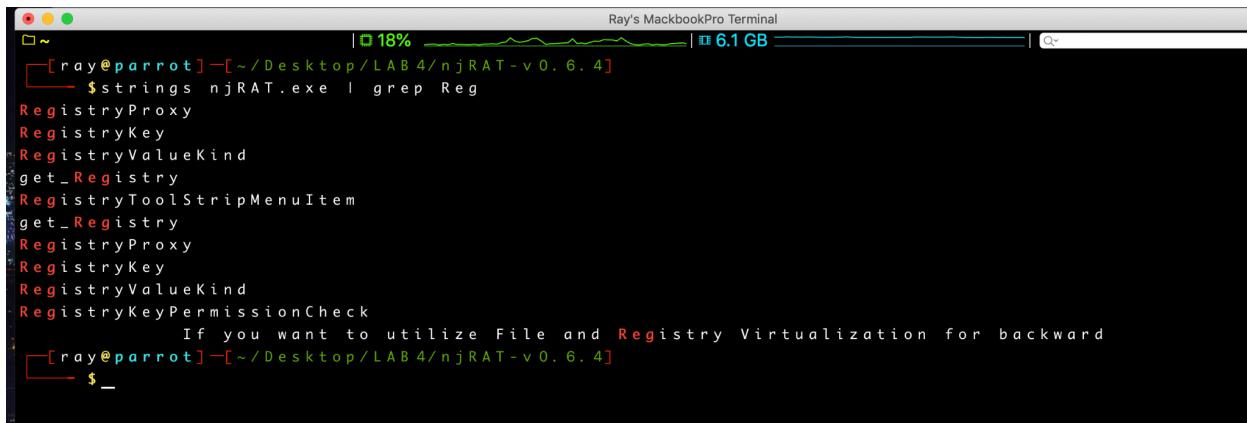


```
Ray's MacBookPro Terminal
└─[ray@parrot]─[~/Desktop/LAB 4/nJRAT-v 0.6.4]
└─$ strings nJRAT.exe | grep get | sort -u
EnvironmentVariableTableTarget
TargetFrameworkAttribute
TargetInvocationException
get_Application
get_Arrow
get_Assembly
get_AsyncWaitHandle
get_Available
get_BackColor
get_BaseStream
get_Block
get_Blue
get_BlueViolet
get_Body
get_Bounds
get_Brush
get_Button
get_CapsLock
get_Chars
get_Checked
get_ClassesRoot
get_Client
get_Clock
get_Column
get_Columns
get_Computer
get_Controls
get_Count
get_CtrlKeyDown
get_Culture
get_Current
```

Windows API Set, Get, Registry File and Tool functions indicate that the attacker might run this executable to connect back to C2 Server.



```
└─$ strings nJRAT.exe | grep set
set_WM
set_IsSingleInstance
set_EnableVisualStyles
set_SaveMySettingsOnExit
set_ShutdownStyle
set_MainForm
set_Name
set_PropertyType
set_EventType
set_FieldType
get_Offset
set_Offset
set_Position
set_Interval
set_Enabled
set_SizeMode
set_Image
set_ForeColor
set_Text
set_Value
set_ReadTimeout
set_Maximum
set_ImageIndex
set_SelectionStart
set_SelectionFont
set_ToolTipText
set_BackColor
set_ImageKey
set_Visible
set_SelectedIndex
set_Tag
```



```
Ray's MacBookPro Terminal
└─[ray@parrot]─[~/Desktop/LAB 4/nJRAT-v 0.6.4]
└─$ strings nJRAT.exe | grep Reg
RegistryProxy
RegistryKey
RegistryValueKind
get_Registry
RegistryToolStripMenuItem
get_Registry
RegistryProxy
RegistryKey
RegistryValueKind
RegistryKeyPermissionCheck
If you want to utilize File and Registry Virtualization for backward
└─[ray@parrot]─[~/Desktop/LAB 4/nJRAT-v 0.6.4]
└─$ _
```

```
[ray@parrot] ~/Desktop/LAB 4/njRAT - v 0.6.4]
$strings njRAT.exe | grep Tool
ToolStrip
ToolStripControlHost
ToolStripDropDown
ToolStripDropDownButton
ToolStripDropDownItem
ToolStripItem
ToolStripItemCollection
ToolStripLabel
ToolStripLayoutStyle
ToolStripMenuItem
ToolStripProgressBar
ToolStripRenderMode
ToolStripStatusLabel
ToolStripTextBox
set_ToolTipText
get_ToolTipText
set_ShowItemToolTips
ToolStripStatusLabel1
RefreshToolStripMenuItem
ToolStripMenuItem1
UPToolStripMenuItem
RunToolStripMenuItem
DeleteToolStripMenuItem
EditToolStripMenuItem
RenameToolStripMenuItem
DownloadToolStripMenuItem
UploadToolStripMenuItem
NewEmptyFileToolStripMenuItem
NewFolderToolStripMenuItem
CopyToolStripMenuItem
PasteToolStripMenuItem
```

As we have the Windows API information, we can write the Yara rule for detecting the njRAT file. Yara rules have a **meta** header which is the owner of this file, a **strings** section that has the string variable for matching purposes in a malware file, and the last **condition** part is to trigger to detect based on the condition.

```

GNU nano 5.4
rule RAT_Detection{
meta:
    author = "Ray"
    description = "njRAT - Remote Access Trojan"
    comment = "Find Indicator of Compromise (IOC) for Windows API"

strings:
$stirng1 = /ToolStrip/
$stirng2 = /ToolStripControlHost/
$stirng3 = /ToolStripDropDown/
$stirng4 = /ToolStripDropDownButton/
$stirng5 = /ToolStripDropDownItem/
$stirng6 = /ToolStripItem/
$stirng7 = /ToolStripItemCollection/
$stirng8 = /ToolStripLabel/
$stirng9 = /ToolStripLayoutStyle/
$stirng10 = /ToolStripMenuItem/
$stirng11 = /set_SendBufferSize/
$stirng12 = /set_ReceiveBufferSize/
$stirng13 = /RegistryProxy/
$stirng14 = /RegistryToolStripMenuItem/
$stirng15 = /RegistryKey/
$stirng16 = /RegistryValueKind/
$stirng17 = /RegistryProxy/
$stirng18 = /get_Regstry/
$stirng19 = /RegistryValueKind/
$stirng20 = /RegistryKeyPermissionCheck/
condition:
    10 of them
}

```

The successful detection result after executing the Yara rule is in terms of the below screenshot. This shows *RAT_Detection*.

```

[ray@parrot] -[~/Desktop/LAB4/njRAT-v0.6.4]
└─$ sudo yara RAT_detection.yara njRAT.exe
RAT_Detection njRAT.exe
[ray@parrot] -[~/Desktop/LAB4/njRAT-v0.6.4]
└─$ 

```

Find the C2 digital footprint:

We only find some numbers with *grep “[0-255].[0-255].[0-255].[0-255]\$”*

```

[ray@parrot] -[~/Desktop/LAB /njRAT-v . . ]
└─$ strings njRAT.exe | grep '[0-255].[0-255].[0-255].[0-255]$'
2.0.0.0
10.0.0.0
10.0.0.0
[ray@parrot] -[~/Desktop/LAB /njRAT-v . . ]
└─$ 

```

If we use `grep "[0-255].[0-255].[0-255].[0-255]"` without \$ at the end, this shows the below result, which could be some infected machines' IP addresses or some windows service versions.

```
/f11d50a3aPADPAB
QSystem.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
QSystem.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
QSystem.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
lSystem.Resources.ResourceReader, mscorelib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=
7a5c561934e089#System.Resources.RuntimeResourceSet
lSystem.Resources.ResourceReader, mscorelib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=
7a5c561934e089#System.Resources.RuntimeResourceSet
fSystem.Drawing.Icon, System.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f
11d50a3ahSystem.Drawing.Bitmap, System.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyTo
n=b03f5f7f11d50a3aPADPAPBj
QSystem.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
QSystem.Drawing, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
2.0.0.0
<assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
<assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
1.0.0.0
1.0.0.0
<assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
-Fnay@connect7-PC:/D/Downloads/Exploit/PAT/Windows/
```

Task 2: Run Yara on malware samples

This Task 2 is to utilize the provided Yara rules to detect malware examples.

Malware Examples:

```
└── $ ls malwareSamples/
03d45dc27bbd683325451ddd8903380113b84581a3e1fa7f7ec0eac6e12595c.dll
129c188a40001fcfc54c92bbe1d88dde350133c2456fa3b4e8effe3b5af702faaff.xls
178a81904017a5b53f378821225ee5d6e436834b1e9e4c9f0ce50805ac36ca37.lnk
2c2e6699405f6fce6adca153c90bdbc58630b10a70b2b92438de04953b5ea12.jar
34e128592e3997a37e00e695a89893560d646f5e078f2d2490df98029fdb8ca7.jar
351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c07c22e.img
38dcfe4f6c31cd0e5c90fc55a2413e3c25342c89b90c42b54cb2a2fe8c9a1c77.exe
4ed978dd7a57e5df732c4a20a738adb245aa389abfad3ed9aa784f57325e990e.js
50e23d069187744a2d3f5d1acfde6506d30e304f0f3d92c57efba9aa061de3a3.jar
76bac32537fe948a8a8b2a4d7cd9877b8d0f603e39298e13c2534c5ef5063e8f.exe
795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c351e5ffbo.dll
835a00d6e7c43db49ae7b3fa12559f23c2920b7530f4d3f960fd285b42b1efb5.rat
8449c227a01dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d9f31901e9.exe
85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d28cd9bababa.exe
8a0675001b5bc63d8389fc7ed80b4a7b0f9538c744350f00162533519e106426.rat
91f43e2dc4437745a310ed3a068457bc090ad11f43b680157ad8b774ad74197.jar
999c88589a40c7321c46d3c53f6c2ca8d0a1ed34601c3c3e2995fd3e066297.exe
9f19c01e8d248937544e926a2f5377fffd5a38f035eb9b3dc692a1cc99394053b.jar
a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat
a6e9679222a133139c4426067330763acc5f8e59f05e1af8636851b0d6aac89.xlsx
a9ecb2c9292cb2d021b122ff5ee1d3f45c672fd75af71e823e524130eb9dd81b.docx
b5e8ed18ebda8beb08e69cd2a602866dcfa8f0aeb20429f4eaf31732c9cc38.exe
cbf7cbc7305bed6abb433ff9b8277c63a2d79dc845d2995adf8cc1c6dd5463dd.jar
cce2dc0e46ba5dd734800c37dc01ef27ea23b912ee98f65e3b5d89f7c7883c64.jar
cd9709bf1c7396f6fe3684b5177fa0890c706ca82e2b98ba58e8d8383632a3c8.rat
cdadac26c09f869e21053ee1a0acf3b2d11df8edd599fe9c377bd4d3ce1c9cd9a9.rat
d61e712d33eb5c948bb64c232292e64add9fb64172163b2eaaa333a017edce3.jar
dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567b14031.iso
df64df82b18e852a3b662b4b26e46a1077fd298c0b9133ba7a8f084b988a4b0f.jar
e2acf723916ce5db6714a17e6d3cf2c95fc1a859de7fbe741a480e679749a86.dll
f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e813776.iso
f86ade6b016aa96bdb40c459b7b3cb413680b891d4436fffa8acc25fa03f0eba0.exe
fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
```

Lazarus.yar:

The result of using Lazarus.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot: ~/Desktop/LAB] $ sudo yara Lazarus.yar -r malwaresamples/
[sudo] password for ray:
EXE_in_LNK malwaresamples//178a81904017a5b53f378821225ee5d6e436834b1e9e4c9f0ce50805ac36ca37.lnk
Encrypted_Office_Document malwaresamples//a9ecb2c9292cb2d021b122ff5ee1d3f45c672fd75af71e823e524
130eb9dd81b.docx
Windows_API_Function malwaresamples//a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a4
87d73a.rat
Windows_API_Function malwaresamples//f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e
813776.iso
Windows_API_Function malwaresamples//351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c
07c22e.img
Windows_API_Function malwaresamples//dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567
b14031.iso
[ray@parrot: ~/Desktop/LAB] $
```

Qakbot.yar:

The result of using the Qakbot.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot: ~/Desktop/LAB] $ sudo yara Qakbot.yar -r malwaresamples/
Windows_API_Function malwaresamples//351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c
07c22e.img
Windows_API_Function malwaresamples//a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a4
87d73a.rat
Windows_API_Function malwaresamples//f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e
813776.iso
Windows_API_Function malwaresamples//dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567
b14031.iso
[ray@parrot: ~/Desktop/LAB] $
```

RAT_Njrat.yar:

The result of using the RAT_Njrat.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ sudo yara RAT_Njrat.yar -r malwaresamples/
Njrat malwaresamples//a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat
Njrat malwaresamples//fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
njrat1 malwaresamples//fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
[ray@parrot] -[~/Desktop/LAB]
└─$ _
```

crime_wannacry.yar:

The result of using crime_wannacry.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ sudo yara crime_wannacry.yar -r malwaresamples/
WannaCry_Ransomware malwaresamples//795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c351e
5ffb0.dll
WannaCry_Ransomware_Gen malwaresamples//795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c
351e5ffb0.dll
WannaCry_Ransomware malwaresamples//8449c227a0a1dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d9f31
901e9.exe
WannaCry_Ransomware_Gen malwaresamples//8449c227a0a1dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d
9f31901e9.exe
WannaCry_Ransomware malwaresamples//85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d28cd9
baaba.exe
WannaCry_Ransomware_Gen malwaresamples//85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d2
8cd9baaba.exe
WannaCry_Ransomware malwaresamples//76bac32537fe948a8a8b2a4d7cd9877b8d0f603e39298e13c2534c5ef50
63e8f.exe
WannaCry_Ransomware malwaresamples//b5e8ed118ebda8beb08e69cd2a602866dc8f0aeb20429f4eaf31732c
9cc38.exe
WannaCry_Ransomware malwaresamples//999c88589a40c7321c46d3ce53f6c2ca8d0a1ed34601c3c33e2995fd3e0
66297.exe
WannaCry_Ransomware_Gen malwaresamples//999c88589a40c7321c46d3ce53f6c2ca8d0a1ed34601c3c33e2995f
d3e066297.exe
WannaCry_Ransomware malwaresamples//03d4a5dc27bbd683325451ddd8903380113b84581a3e1fa7f7ec0eac6e1
2595c.dll
[ray@parrot] -[~/Desktop/LAB]
└─$ _
```

redline_staler.yar:

The result of using the redline_staler.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ sudo yara redline_staler.yar -r malwaresamples/
MALWARE_Win_zgRAT malwaresamples//e2acf723916ce5db6714a17e6d3cf2c95fc1a859de7fbe741a480e679749
a86.dll
MALWARE_Win_RedLine malwaresamples//38dcfe4f6c31cd0e5c90fc55a2413e3c25342c89b90c42b54cb2a2fe8c9
a1c77.exe
MALWARE_Win_NjRAT malwaresamples//fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822
199.rat
INDICATOR_EXE_Packed_Themida malwaresamples//f86ade6b016aa96bdb40c459b7b3cb413680b891d4436ffa8a
cc25fa03f0eba0.exe
[ray@parrot] -[~/Desktop/LAB]
└─$ _
```

general_rats_malwareconfig.yar:

The result of using general_rats_malwareconfig.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ sudo yara general_rats_malwareconfig.yar -r malwaresamples/
MAL_JRAT_Oct 18_1 malwaresamples//d61e712d33eb5c948bb64c232292e64add9fbe64172163b2eaaa333a017edc
e3.jar
RAT_njRat malwaresamples//fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
```

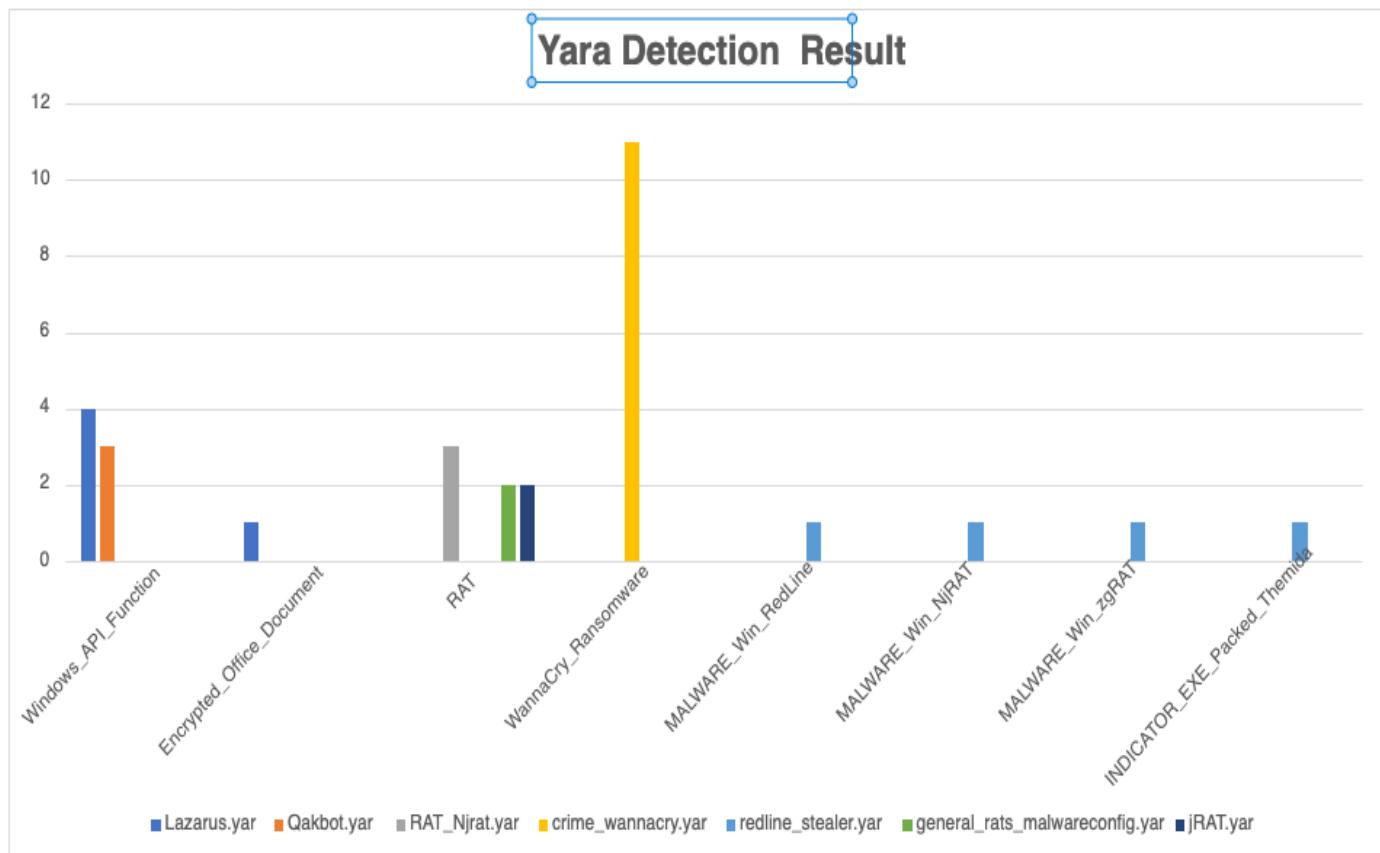
jRAT.yar:

The result of using the jRAT.yar rule with “-r” recursively detects each file of malware example directories.

```
[ray@parrot] -[~/Desktop/LAB]
└─$ sudo yara jRAT.yar -r malwaresamples/
jRat malwaresamples//df64df82b18e852a3b662b4b26e46a1077fd298c0b9133ba7a8f084b988a4b0f.jar
jRat malwaresamples//2c2e6699405f6fce6adca153c90bdbc58630b10a70b2b92438de04953b5ea12.jar
-[ray@parrot] -[~/Desktop/LAB]
└─$ -
```

The bar chart:

If we put the Yara result below, this indicates most malware belongs to WannaCry, which was a well-known ransomware in 2017 [5].



Task 3: Analyse the malware sample

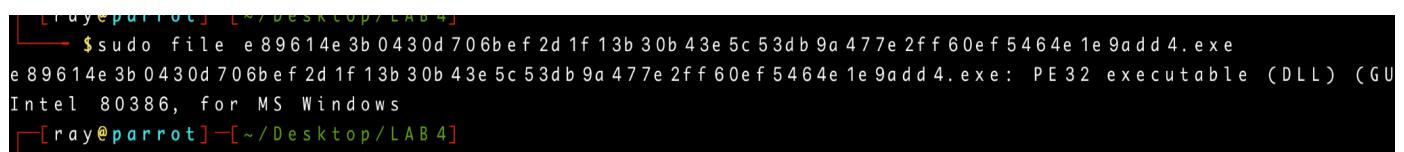
Given the malware and extract the zip file with password:

e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe



e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.zip

Firstly, we have to confirm what file it is and what version of this executable, which is referred to below screenshot.



```
[ray@parrot] -[~/Desktop/LAB 4]$ sudo file e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe: PE32 executable (DLL) (GU
Intel 80386, for MS Windows
[ray@parrot] -[~/Desktop/LAB 4]
```

Then, we collect each of the following information.

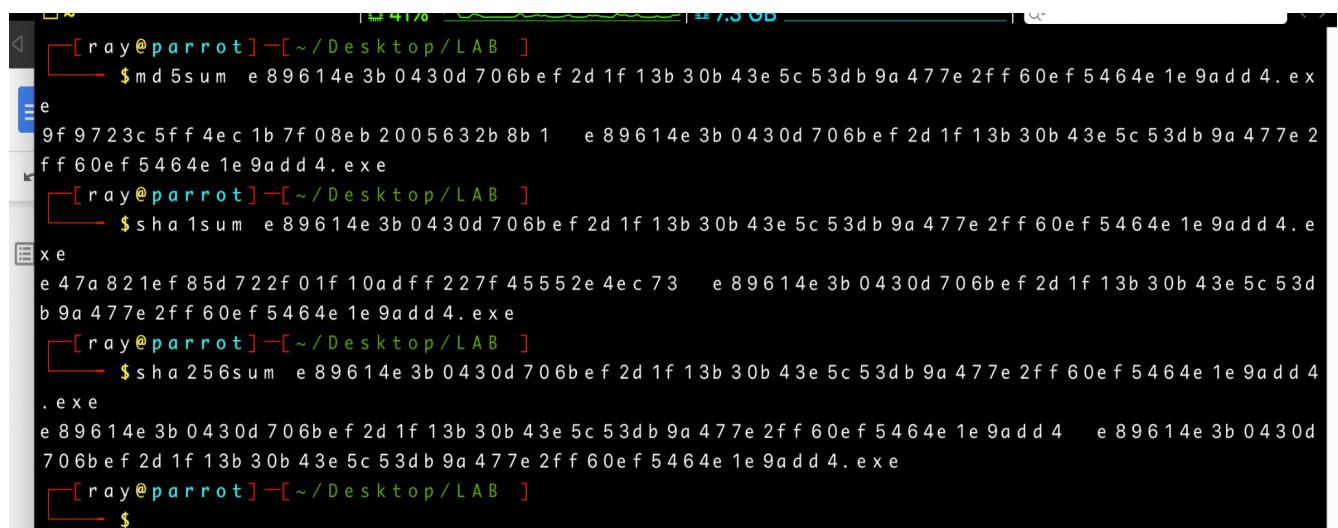
1. Hash Value

md5: 9f9723c5ff4ec1b7f08eb2005632b8b1

sha1sum: e47a821ef85d722f01f10adff227f45552e4ec73

sha256sum:

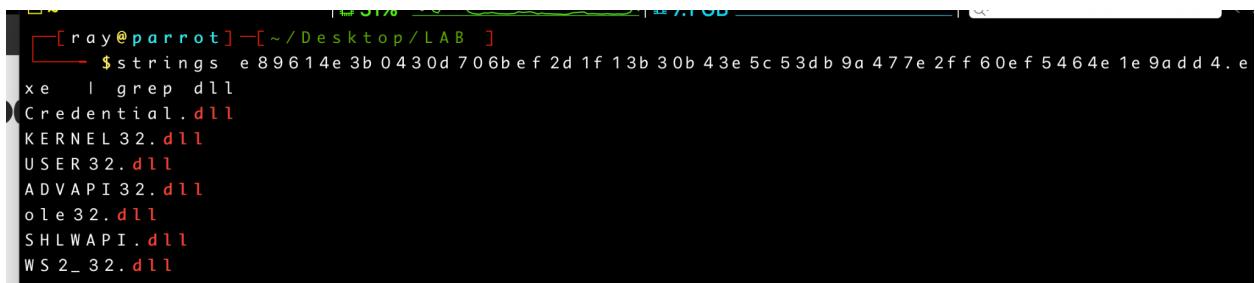
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4



```
[ray@parrot] -[~/Desktop/LAB 4]$ md5sum e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
9f9723c5ff4ec1b7f08eb2005632b8b1 e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
[ray@parrot] -[~/Desktop/LAB 4]$ sha1sum e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
e47a821ef85d722f01f10adff227f45552e4ec73 e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
[ray@parrot] -[~/Desktop/LAB 4]$ sha256sum e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4 e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4.exe
[ray@parrot] -[~/Desktop/LAB 4]$
```

2. Imported DLLs

If we use string to extract the information from malware file, this will show this malware utilizes Credential.dll, KERNEL32.dll, USER32.dll, ADVAPI32.dll, ole32.dll, SHLWAPI.dll, WS2_32.dll. These dll indicates that this malware has some network activities in term os ws2_32.dll, and KernelL21.dll has a various functions call to critical service [6].



```
[ray@parrot] -[~/Desktop/LAB]
$ strings e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60eff5464e1e9add4.e
xe | grep dll
Credential.dll
KERNEL32.dll
USER32.dll
ADVAPI32.dll
ole32.dll
SHLWAPI.dll
WS2_32.dll
```

3. Common Windows API

We can use some dump commands to filter content and see the common windows API in terms of the below screenshot. The command we utilized is “xxd” command to dump the hexadecimal content from the malware file, then pipeline to the “grep” to filter Windows API information.

```

└── $ xx d e 8 9 6 1 4 e 3 b 0 4 3 0 d 7 0 6 b e f 2 d 1 f 1 3 b 3 0 b 4 3 e 5 c 5 3 d b 9 a 4 7 7 e 2 f f 6 0 e f 5 4 6 4 e 1 e 9 a d d 4 . e x e
| grep Get
00014060: 0300 0000 466c 7347 6574 5661 6c75 6500 ....FlsGetValue.
00015d00: 0000 0000 0e00 0000 4765 7443 7572 7265 .....GetCurre
00015d20: 4765 7453 7973 7465 6d54 696d 6550 7265 GetSystemTimePre
00019610: 4765 744e 6174 6976 6553 7973 7465 6d49 GetNativeSystemI
00019620: 5274 6c47 6574 4e74 5665 7273 696f 6e4e RtlGetNtVersionN
0001a120: 0047 6574 4f62 6a65 6374 436f 756e 7400 .GetObjectCount.
0001a400: 4765 744d 6f64 756c 6546 696c 654e 616d GetModuleFileName
0001a430: 7800 5205 536c 6565 7000 5002 4765 744c x.R.Sleep.P.GetL
0001a490: f202 4765 7454 6963 6b43 6f75 6e74 0000 ..GetTickCount..
0001a4f0: 4765 7453 7973 7465 6d49 6e66 6f00 2403 GetSystemInfo.$.
0001a510: 7573 4578 0000 d101 4765 7443 6f6d 7075 useEx....GetCompu
0001a520: 7465 724e 616d 6557 0000 d001 4765 7443 terNameW....GetC
0001a540: a401 4765 7441 4350 0000 8602 4765 744f ..GetACP....GetO
0001a550: 454d 4350 0000 9202 4765 7450 7269 6f72 EMCP....GetPrior
0001a560: 6974 7943 6c61 7373 0000 0902 4765 7443 ityClass....GetC
0001a580: 4765 7454 6872 6561 6450 7269 6f72 6974 GetThreadPriorit
0001a590: 7900 0d02 4765 7443 7572 7265 6e74 5468 y....GetCurrentTh
0001a610: 7957 0000 9d02 4765 7450 726f 6341 6464 yW....GetProcAdd
0001a630: 6172 7900 6702 4765 744d 6f64 756c 6548 ary.g.GetModuleH
0001a640: 616e 646c 6557 0000 7402 4765 744e 6174 andleW...t.GetNat
0001a6b0: 4669 6c65 5700 e302 4765 7454 656d 7050 FileW....GetTempP
0001a6c0: 6174 6857 0000 0e02 4765 7443 7572 7265 atchW....GetCurre
0001a6f0: 5700 0803 4765 7456 6f6c 756d 6549 6e66 W....GetVolumeInf
0001a700: 6f72 6d61 7469 6f6e 5700 1f02 4765 7444 ormationW...GetD
0001a710: 7269 7665 5479 7065 5700 5602 4765 744c riveTypeW.V.GetL
0001a730: 6773 5700 3202 4765 7446 696c 6541 7474 gsW.2.GetFileAtt
0001a770: 4669 6e64 436c 6f73 6500 3b02 4765 7446 FindClose.;.GetF
0001a8d0: 646c 6c00 0700 4765 7441 6464 7249 6e66 dll...GetAddrInf
0001a9b0: be02 4765 7453 7461 7274 7570 496e 666f ..GetStartupInfo
0001a9c0: 5700 0a02 4765 7443 7572 7265 6e74 5072 W....GetCurrentPr
0001a9d0: 6f63 6573 7349 6400 d602 4765 7453 7973 ocessId....GetSys

```

4. VirusTotal Result

Based on the hash file extracted from the malware file, the file is flagged as the malware.

The screenshot shows a malware analysis interface. At the top, there's a search bar with the file hash 'e89614e3b0430d706bef2d1f13b30b43e5c53db9a477e2ff60ef5464e1e9add4'. To the right are icons for search, upload, download, and sign in.

File Details:

- Community Score:** 52 / 71
- Security Vendors' Analysis:** 52 security vendors flagged this file as malicious.
- File Info:** Credential.dll, pedll, Size: 116.00 KB, Date: 2022-10-15 18:29:12 UTC, 1 month ago.
- Category:** DLL

Tab Selection: DETECTION (selected), DETAILS, RELATIONS, BEHAVIOR, COMMUNITY (3)

Security Vendors' Analysis:

Vendor	Signature	Engine	Signature
Ad-Aware	! Trojan.GenericKD.43597666	AhnLab-V3	! Trojan/Win32.Stealer.C4191896
Alibaba	! TrojanPSW:Win32/CryptInject.7e5b5404	ALYac	! Trojan.Agent.Sepulcher
Antiy-AVL	! Trojan/Generic.ASMalwS.4B	Arcabit	! Trojan.Generic.D2993F62
Avast	! Win32:Trojan-gen	AVG	! Win32:Trojan-gen
Avira (no cloud)	! TR/PSW.Agent.wpzys	BitDefender	! Trojan.GenericKD.43597666
BitDefenderTheta	! Gen>NN.ZediaF.34726.hy8@auMSP7ni	Comodo	! Malware@#3i3wuv5vhw1nu
CrowdStrike Falcon	! Win/malicious_confidence_100% (W)	Cylance	! Unsafe
Cynet	! Malicious (score: 100)	Elastic	! Malicious (high Confidence)

Considering that we can not executable the malware file this task, and can not get much more information from the malware file. So, we decide to do the extra task.

Extra Task : Analyse the infected memory affecting from WannaCry via memory analysis tool - Volatility.

We downloaded an infected file from Blue Team training platform, which is <https://blueteamlabs.online/home/challenge/memory-analysis-ransomware-7da6c9244d>

From the SOC objective we have to analysis the infected backup memory after the cyberattack.

Confirm the infected machine which version it uses via *imageinfo* parameter.

```
[ray@parrot] -[~/Desktop]
└─$ python2 volatility2/vol.py -f Ransomware/infected.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1

INFO    : volatility.debug      : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86_2341

P 1x 86
          AS Layer1 : IA32PagedMemoryPae (Kernel AS)
          AS Layer2 : FileAddressSpace (/home/ray/Desktop/Ransomware/infected.vmem)
          PAE type   : PAE
          DTB        : 0x185000L
          KDBG       : 0x82948c28L
  Number of Processors : 1
  Image Type (Service Pack) : 1
          KPCR for CPU 0 : 0x82949c00L
          KUSER_SHARED_DATA : 0xffffdf0000L
  Image date and time  : 2021-01-31 18:24:57 UTC+0000
  Image local date and time : 2021-01-31 13:24:57 -0500
[ray@parrot] -[~/Desktop]
└─$
```

After confirming the system is infected, we can dump process list from the infected memory via *pslist*

```
[ray@parrot] -[~/Desktop]
└─$ python2 volatility2/vol.py -f Ransomware/infected.vmem --profile=Win7SP1x86_2341 pslist

18:02:20 UTC+0000
0x85dc25f8 conhost.exe           2976    404    1     33    1     0 2021-01-31
18:02:20 UTC+0000
0x83ec6800 @WanaDecryptor      3968    2732    1     59    1     0 2021-01-31
18:02:48 UTC+0000
```

As to the below screenshot, it shows the process id 3968 is running WanaDecryptor process, and we can extract the file from this process with *procdump*

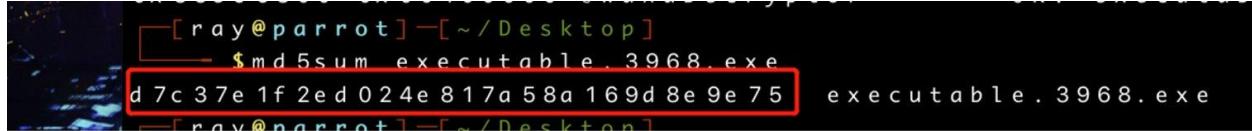
```
[ray@parrot] -[~/Desktop]
└─$ python2 volatility2/vol.py -f Ransomware/infected.vmem --profile=Win7SP1x86_2341 procdump -D . -p 3968
```

The malware file is executable.3968.exe

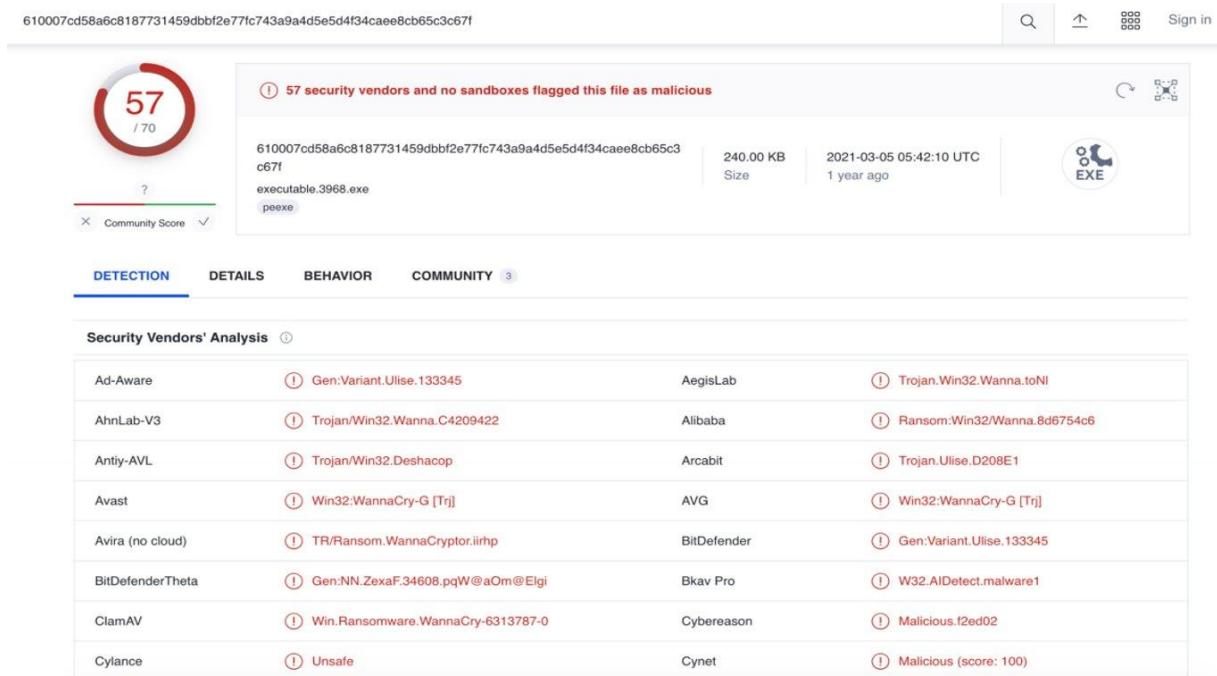
```
└─$ cd crypto.rashny
Process(V) ImageBase  Name
----- 0x83ec6800 0x00400000 @WanaDecryptor
[ray@parrot] -[~/Desktop]
└─$
```

Result
OK: executable.3968.exe

The md5 hash value of the malware is d7c37e1f2ed024e817a58a169d8e9e75, and this is flagged as WannaCry ransomware via VirusTotal



```
[ ray@parrot ]-[ ~/Desktop ]
$ md5sum executable.3968.exe
d 7c 37e 1f 2e d 024e 817a 58a 169d 8e 9e 75  executable.3968.exe
[ ray@parrot ]-[ ~/Desktop ]
```



610007cd58a6c8187731459dbbf2e77fc743a9a4d5e5d4f34cae8cb65c3c67f

57 /70

Community Score

① 57 security vendors and no sandboxes flagged this file as malicious

610007cd58a6c8187731459dbbf2e77fc743a9a4d5e5d4f34cae8cb65c3c67f
executable.3968.exe
peexe

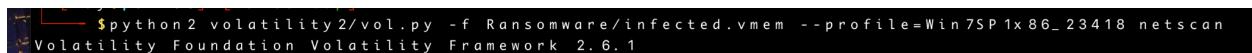
240.00 KB Size | 2021-03-05 05:42:10 UTC 1 year ago | EXE

DETECTION DETAILS BEHAVIOR COMMUNITY 3

Security Vendors' Analysis ①

Vendor	Detection	Vendor	Detection
Ad-Aware	① Gen:Variant.Ulise.133345	AegisLab	① Trojan.Win32.Wanna.toNI
AhnLab-V3	① Trojan/Win32.Wanna.C4209422	Alibaba	① Ransom:Win32/Wanna.8d6754c6
Antiy-AVL	① Trojan/Win32.Deshacop	Arcabit	① Trojan.Ulise.D208E1
Avast	① Win32:WannaCry-G [Trj]	AVG	① Win32:WannaCry-G [Trj]
Avira (no cloud)	① TR/Ransom.WannaCryptor.iirhp	BitDefender	① Gen:Variant.Ulise.133345
BitDefenderTheta	① Gen:NN.Zexaf.34608.pqW@aOm@Elgi	Bkav Pro	① W32.AIDetect.malware1
ClamAV	① Win.Ransomware.WannaCry-6313787-0	Cybereason	① Malicious.f2ed02
Cylance	① Unsafe	Cynet	① Malicious (score: 100)

As for the network connection, we can find there is a suspicious connection after using **netscan** command. The suspicious connection establishing between **192.168.252.143:49190** and **51.75.206.12:9000**



```
[ ray@parrot ]-[ ~/Desktop ]
$ python2 volatility2/vol.py -f Ransomware/infected.vmem --profile=Win7SP1x86_23418 netscan
Volatility Foundation Volatility Framework 2.6.1
```

Offset(P)	Owner	Proto	Local Address Created	Foreign Address	State
0x58c75b8	svchost.exe	TCPv4	0.0.0.0:49153	0.0.0.0:0	LISTENING
36	svchost.exe	TCPv6	:::49153	:::0	LISTENING
36	svchost.exe	TCPv4	192.168.252.143:49165	192.87.28.82:9001	ESTABLISHED
968	taskhsvc.exe	TCPv4	192.168.252.143:49191	8.252.80.254:80	ESTABLISHED
96	svchost.exe	UDPv4	0.0.0.0:5355	*:*	
068	svchost.exe	UDPv4	2021-01-31 18:16:15 UTC+0000	0.0.0.0:0	LISTENING
0x11b80a68	taskhsvc.exe	TCPv4	127.0.0.1:9050	0.0.0.0:0	LISTENING
968	taskhsvc.exe	TCPv4	192.168.252.143:49188	192.168.252.2:80	CLOSED
0x179f09f0	taskhsvc.exe	TCPv4	192.168.252.143:49190	51.75.206.12:9000	ESTABLISHED
968	taskhsvc.exe	UDPV6	::1:53611	*:*	
204	svchost.exe	UDPV6	2021-01-31 18:16:15 UTC+0000		

Conclusion

To summarize this malware detection lab, we have obtained the knowledge how to analyze malware file and how to use Yara rule to detect the file whether it is malware or not. The most important this is we do some extra tasks, which is to perform how to use memory analysis tool with dumping infected virtual memory, and see the suspicious connection.

Reference

1. GrantMeStrength, “Windows API index - win32 apps,” *Win32 apps | Microsoft Learn*. [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list>. [Accessed: 24-Nov-2022].
2. Stevewhims, “Registry functions - win32 apps,” *Win32 apps | Microsoft Learn*. [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-functions>. [Accessed: 24-Nov-2022].

3. Alvinashcraft, “SetFileTime function (fileapi.h) - win32 apps,” *Win32 apps | Microsoft Learn*. [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-setftime>. [Accessed: 24-Nov-2022].
4. Dotnet-Bot, “Toolstriptextbox class (system.windows.forms),” (*System.Windows.Forms*) | *Microsoft Learn*. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.toolstriptextbox?view=windowsdesktop-7.0>. [Accessed: 24-Nov-2022].
5. “WannaCry ransomware attack,” *Wikipedia*, 17-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack. [Accessed: 28-Nov-2022].
6. K. K. Reddy, “Basic static analysis of malware and common DLL,” *Medium*, 28-Dec-2019. [Online]. Available: <https://medium.com/mrx-007/basic-static-analysis-of-malware-and-common-dll-ef9455d49968>. [Accessed: 28-Nov-2022].