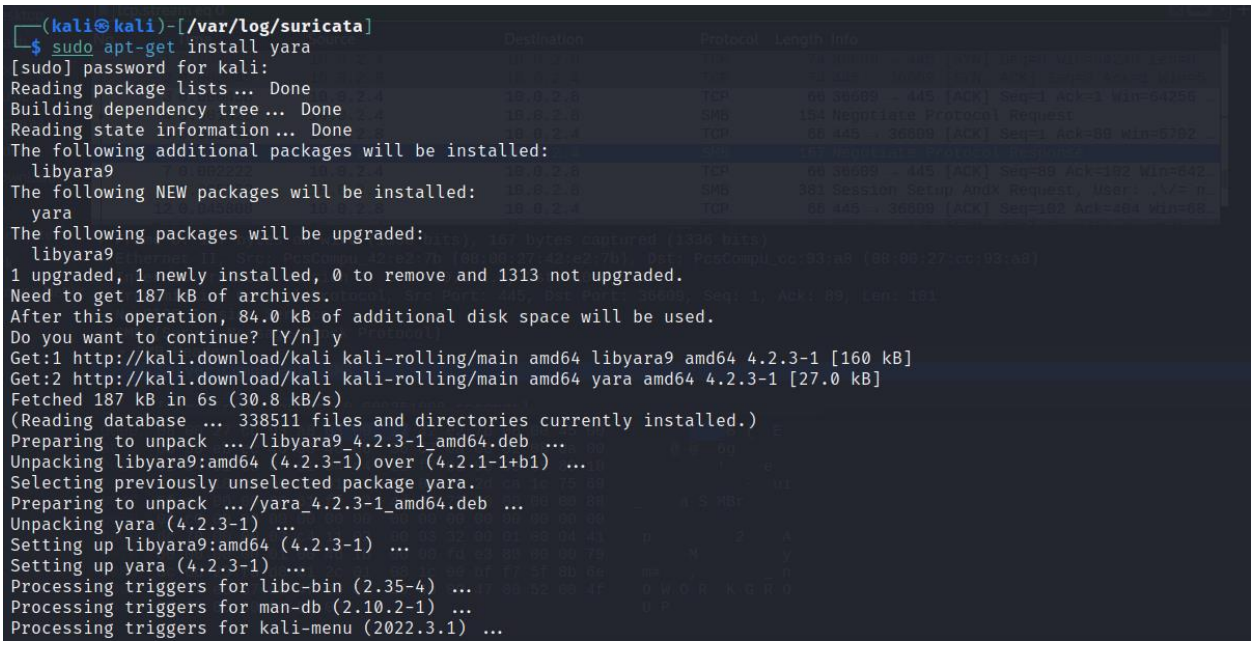
Report

for

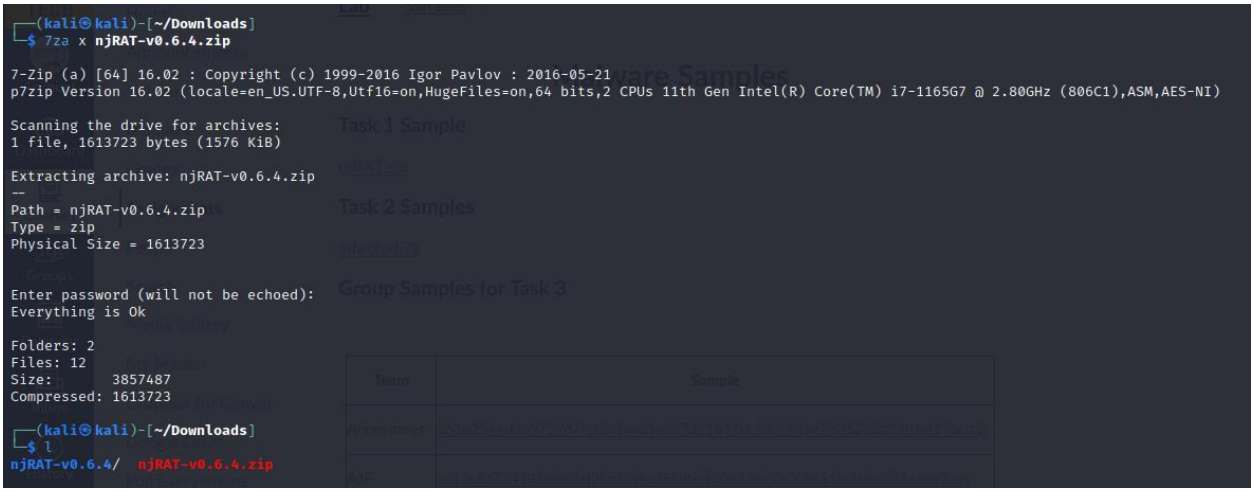
Malware Detection

Task 1: Create a Yara rule for njRAT malware

In the first step, we need to install yara which is a pattern-matching software, and one of its capabilities is detecting malwares.

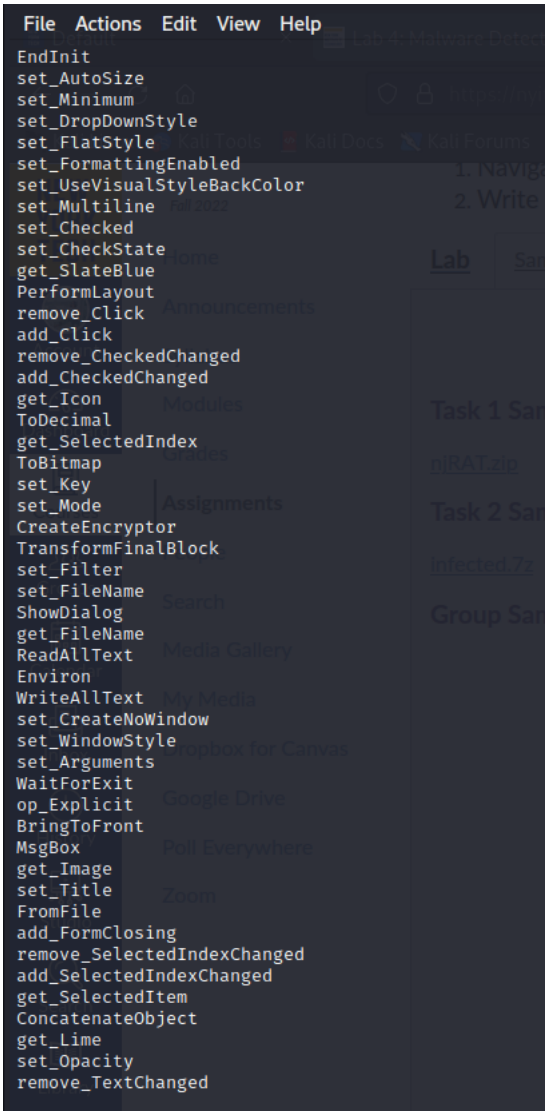


Then we extracted the sample malware named “njRAT-v0.6.4.zip ” with the below commands:



Once we unzipped the given file, we can see all the files inside it as shown in the picture below:

After that, with the command “strings” we can check which APIs this malware used



Then it will help us to use them for writing our Yara rules and put these API-name in string part, and in the condition part if this rule detects 10 of them and raise errors.

```
rule njrat_detection
{
        meta:
                description = "Yara rule for njRAT detection Task1"
                author = "Team PH Pegah and Hanieh"
        strings:
                $string1=/GetModules/
                $string2=/Get_Assembly/
                $string3=/get_Parent/
                $string4=/GetTypes/
                $string5=/IntPtr/
                $string6=/Read/
                $string7=/BitConverter/
                $string8=/Space/
                $string9=/StrDup/
                $string10=/ToInt32/
                $string11=/Zero/
                $string12=/CompressionMode/
                $string13=/System.Reflection/
                $string14=/DirectoryInfo/
        condition:
                10 of them
}

```

After running our Yara rule, the name of the rule and file which is malware was printed. This shows, at least 10 out of 14 strings (function calls) are used in the malware.

```
┌──(kali㉿kali)-[~/Downloads/njRAT-v0.6.4]
└─$ l
GeoIP.dat  Mono.Cecil.dll  NAudio.dll  njRAT.exe  Plugin/  stub.il  Stub.manifest  Task1-rule.yara

┌──(kali㉿kali)-[~/Downloads/njRAT-v0.6.4]
└─$ yara Task1-rule.yara njRAT.exe
njrat_detection njRAT.exe
```

Task2: Run Yara on malware samples

For this task, we have given a file which is contained a bunch of different malware samples and some Yara rules. like task1, we again used unzip command (7za) to extract the file. In the figure below we can see them all in our directory:

Then need to run each Yara rule on the “malwaresamples” file and see what is the result of each yara command.

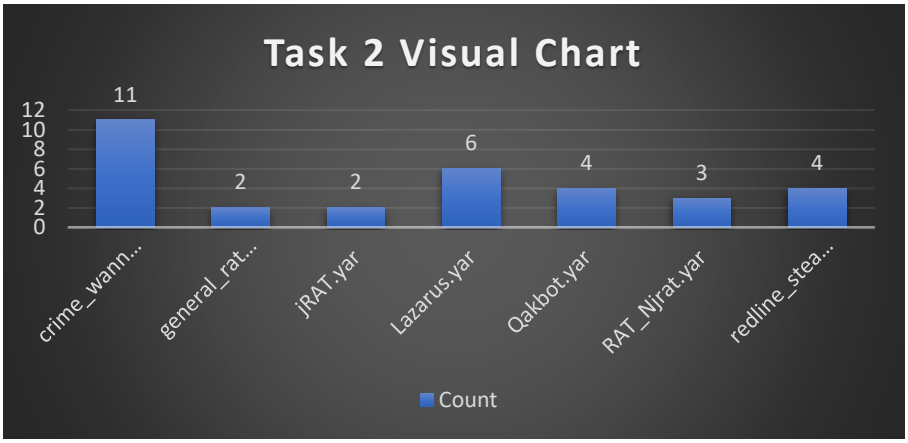
Below pictures shows all of them :

```
┌──(kali㉿kali)-[~]
└─$ yara crime_wannacry.yar malwaresamples
WannaCry_Ransomware malwaresamples/999c88589a40c7321c46d3ce53f6c2ca8d0a1ed34601c3c33e2995fd3e066297.exe
WannaCry_Ransomware_Gen malwaresamples/999c88589a40c7321c46d3ce53f6c2ca8d0a1ed34601c3c33e2995fd3e066297.exe
WannaCry_Ransomware malwaresamples/76bac32537fe948a8a8b2a4d7cd9877b8d0f603e39298e13c2534c5ef5063e8f.exe
WannaCry_Ransomware malwaresamples/03d4a5dc27bbd683325451ddd8903380113b84581a3e1fa7f7ec0eac6e12595c.dll
WannaCry_Ransomware malwaresamples/795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c351e5ffb0.dll
WannaCry_Ransomware_Gen malwaresamples/795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c351e5ffb0.dll
WannaCry_Ransomware malwaresamples/85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d28cd9baaba.exe
WannaCry_Ransomware_Gen malwaresamples/85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d28cd9baaba.exe
WannaCry_Ransomware malwaresamples/8449c227a0a1dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d9f31901e9.exe
WannaCry_Ransomware_Gen malwaresamples/8449c227a0a1dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d9f31901e9.exe
WannaCry_Ransomware malwaresamples/b5e8ed118ebda8bebd08e69cd2a602866dca8f0aebe20429f4eaf31732c9cc38.exe

┌──(kali㉿kali)-[~]
└─$ yara general_rats_malwareconfig.yar malwaresamples
RAT_njRat malwaresamples/fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
MAL_JRAT_Oct18_1 malwaresamples/d61e712d33eb5c948bb64c232292e64add9fbe64172163b2eaaa333a017edce3.jar

┌──(kali㉿kali)-[~]
└─$ yara jRAT.yar malwaresamples
jRat malwaresamples/2c2e6699405f6fece6adca153c90bdbc58630b10a70b2b92438de04953b5ea12.jar
jRat malwaresamples/df64df82b18e852a3b662b4b26e46a1077fd298c0b9133ba7a8f084b988a4b0f.jar

┌──(kali㉿kali)-[~]
└─$ yara Lazarus.yar malwaresamples
Windows_API_Function malwaresamples/a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat
Encrypted_Office_Document malwaresamples/a9ecb2c9292cb2d021b122ff5ee1d3f45c672fd75af71e823e524130eb9dd81b.docx
Windows_API_Function malwaresamples/f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e813776.iso
Windows_API_Function malwaresamples/351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c07c22e.img
EXE_in_LNK malwaresamples/178a81904017a5b53f378821225ee5d6e436834b1e9e4c9f0ce50805ac36ca37.lnk
Windows_API_Function malwaresamples/dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567b14031.iso

┌──(kali㉿kali)-[~]
└─$ yara Qakbot.yar malwaresamples
Windows_API_Function malwaresamples/351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c07c22e.img
Windows_API_Function malwaresamples/a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat
Windows_API_Function malwaresamples/f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e813776.iso
Windows_API_Function malwaresamples/dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567b14031.iso

┌──(kali㉿kali)-[~]
└─$ yara RAT_Njrat.yar malwaresamples
Njrat malwaresamples/fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
njrat1 malwaresamples/fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
Njrat malwaresamples/a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat

┌──(kali㉿kali)-[~]
└─$ yara redline_stealer.yar malwaresamples
INDICATOR_EXE_Packed_Themida malwaresamples/f86ade6b016aa96bdb40c459b7b3cb413680b891d4436ffa8acc25fa03f0eba0.exe
MALWARE_Win_RedLine malwaresamples/38dcfe4f6c31cd0e5c90fc55a2413e3c25342c89b90c42b54cb2a2fe8c9a1c77.exe
MALWARE_Win_NjRAT malwaresamples/fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
MALWARE_Win_zgRAT malwaresamples/e2acf723916ce5db6714a17e6d3cf2c95fca1a859de7fbe741a480e679749a86.dll
```

On the other hand, if we check the inside of “malwaresamples” by “l” command we can see all these results are in it as below red lines :

```
┌──(kali㉿kali)-[~]
└─$ cd malwaresamples

┌──(kali㉿kali)-[~/malwaresamples]
└─$ l
03d4a5dc27bbd683325451ddd8903380113b84581a3e1fa7f7ec0eac6e12595c.dll
129c188a40001cfc54c92bbe1d88dde350133c2456fa3b4e8efe3b5af702faff.xls
178a81904017a5b53f378821225ee5d6e436834b1e9e4c9f0ce50805ac36ca37.lnk
2c2e6699405f6fece6adca153c90bdbc58630b10a70b2b92438de04953b5ea12.jar
34e128592e3997a37e00e695a89893560d646f5e078f2d2490df98029fdb8ca7.jar
351025529c0a38aa351e96c58143f41798f1dd26be05431aae60ca092c07c22e.img
38dcfe4f6c31cd0e5c90fc55a2413e3c25342c89b90c42b54cb2a2fe8c9a1c77.exe
4ed978dd7a57e5df732c4a20a738adb245aa389abfad3ed9aa784f57325e990e.js
50e23d069187744a2d3f5d1acfde6506d30e304f0f3d92c57efba9aa061de3a3.jar
76bac32537fe948a8a8b2a4d7cd9877b8d0f603e39298e13c2534c5ef5063e8f.exe
795742e194ad35b73172bf15bf5f8379b2e8c82a1548ec59c5e935c351e5ffb0.dll
835a00d6e7c43db49ae7b3fa12559f23c2920b7530f4d3f960fd285b42b1efb5.rat
8449c227a0a1dadbc8e1f81bbf6cdf3669727864c9a2f309a224a1d9f31901e9.exe
85aea2af28cb7f0d72911be0a8c52917334c5234682a257b3d001d28cd9baaba.exe
8a0675001b5bc63d8389fc7ed80b4a7b0f9538c744350f00162533519e106426.rat
91f43e2dca4437745a310ed3a068457bc090ad11f43b680157ad8b774ad74197.jar
999c88589a40c7321c46d3ce53f6c2ca8d0a1ed34601c3c33e2995fd3e066297.exe
9f19c01e8d248937544e926a2f5377ffd5a38f035eb9b3dc692a1cc99394053b.jar
a1b65f18c7e882b1606a4ef9387d8988e6fd755d7d03214b677ad528a487d73a.rat
a6e96799222a133139c4426067330763acc5f8e59f05e1af8636851b0d6aac89.xlsx
a9ecb2c9292cb2d021b122ff5ee1d3f45c672fd75af71e823e524130eb9dd81b.docx
b5e8ed118ebda8bebd08e69cd2a602866dca8f0aebe20429f4eaf31732c9cc38.exe
cbf7cbc7305bed6abb433ff9b8277c63a2d79dc845d2995adf8cc1c6dd5463dd.jar
cce2dc0e46ba5dd734800c37dc01ef27ea23b912ee98f65e3b5d89f7c7883c64.jar
cd9709bf1c7396f6fe3684b5177fa0890c706ca82e2b98ba58e8d8383632a3c8.rat
cdadc26c09f869e21053ee1a0acf3b2d11df8edd599fe9c377bd4d3ce1c9cda9.rat
d61e712d33eb5c948bb64c232292e64add9fbe64172163b2eaaa333a017edce3.jar
dc20873b80f5cd3cf221ad5738f411323198fb83a608a8232504fd2567b14031.iso
df64df82b18e852a3b662b4b26e46a1077fd298c0b9133ba7a8f084b988a4b0f.jar
e2acf723916ce5db6714a17e6d3cf2c95fca1a859de7fbe741a480e679749a86.dll
f1bd53092088ec6c35205a381df1360d145f03c6cc11185218dff5013e813776.iso
f86ade6b016aa96bdb40c459b7b3cb413680b891d4436ffa8acc25fa03f0eba0.exe
fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199.rat
```

The below chart shows the number of sample files matched by each rule. The horizontal line shows the names of Yara rules and the vertical line shows the number of matched sample files.

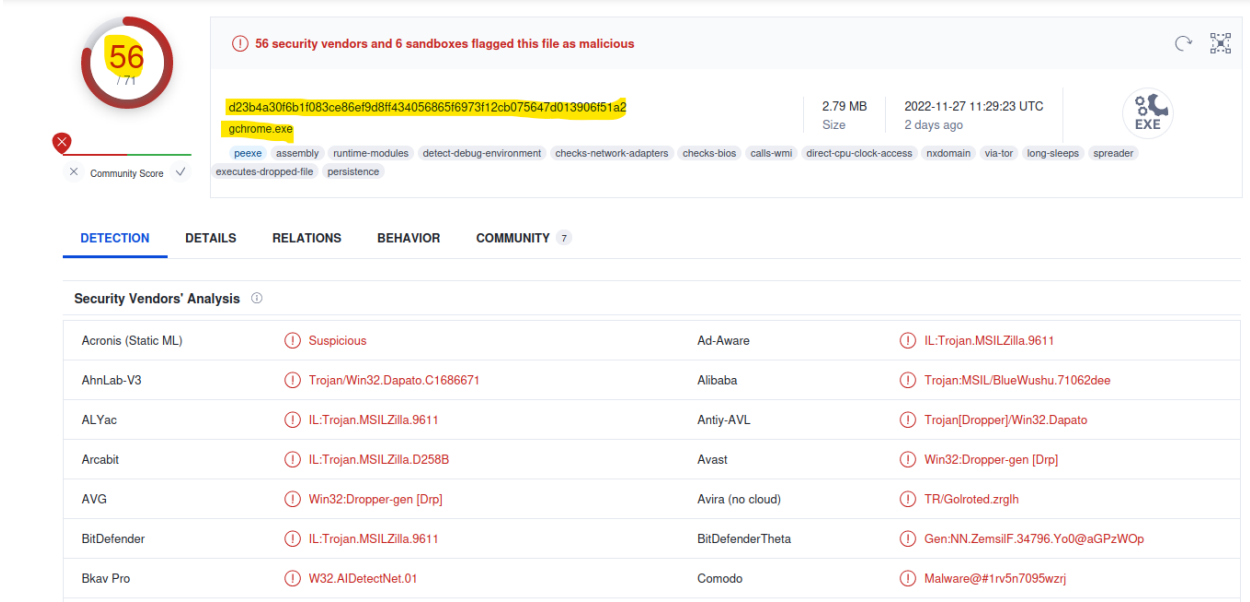


Task 3: Analyse the malware sample

In this task, we as group “PH” has been given a unique malware sample. Again we unzipped it.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ 7za x d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.zip

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs AMD Ryzen 5 5625U with Radeon Graphics
A50F00),ASM)

Scanning the drive for archives:
1 file, 2573164 bytes (2513 KiB)

Extracting archive: d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.zip
--
Path = d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.zip
Type = zip
Physical Size = 2573164


Enter password (will not be echoed):
Everything is Ok

Size:       2930176
Compressed: 2573164
```

There is a site named https://www.virustotal.com. We uploaded this file to this site to see if it shows any details about the malware. As we can get through the site, it shows that 56 security vendors and 6 sandboxes flagged this file as malicious.



The below image shows, this malware is a Trojan horse for the Chrome browser.

History

Creation Time	2016-11-21 12:16:27 UTC
First Seen In The Wild	2016-11-21 17:46:27 UTC
First Submission	2016-11-21 18:19:48 UTC
Last Submission	2022-11-18 20:27:18 UTC
Last Analysis	2022-11-18 20:27:18 UTC

Names

d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe
gchrome.exe
chrome.exe
d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2
Proteus.exe
pl.exe
49FD4020BF4D7BD23956EA892E6860E9
Proteus.....exe
Proteus....exe

Signature Info

Signature Verification

⚠ File is not signed

File Version Information

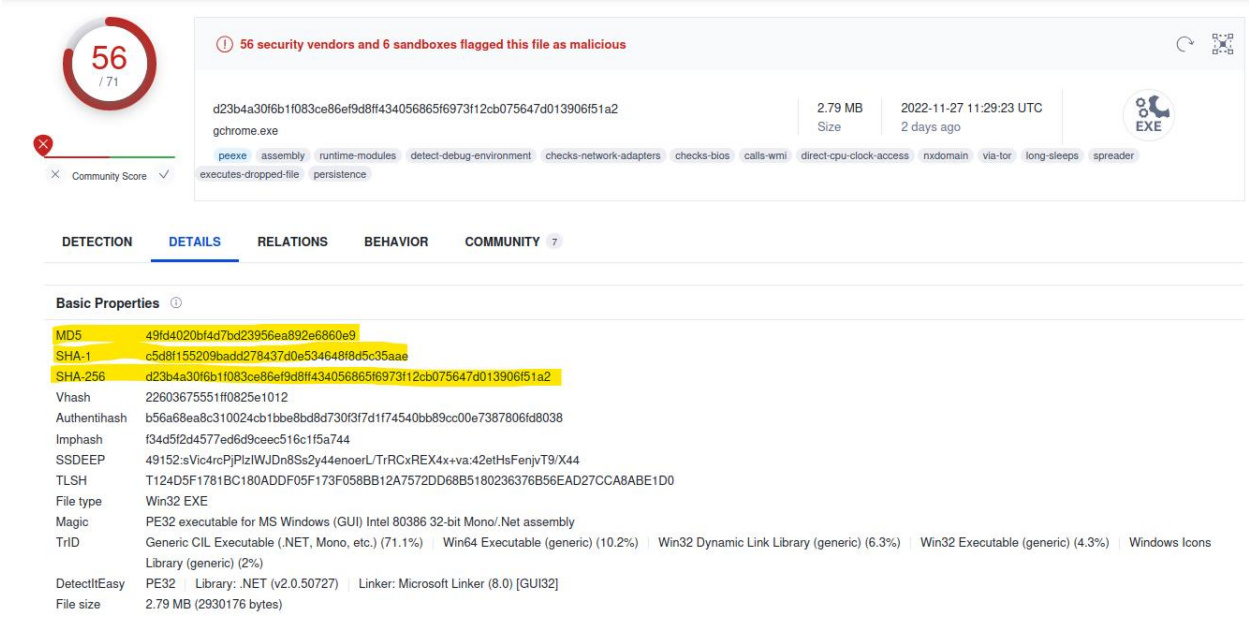
Copyright	Copyright 2016 All rights reserved.
Product	Google Chrorne
Description	Google Chrorne
Original Name	gchrome.exe
Internal Name	gchrome.exe
File Version	55.0.2840.99

3.1: Hashes - md5, sha1sum, sha256sum

In this step, we created hash values of the malware using md5, sha1 and sha2 algorithms. Then we compared them with the provided values from the website.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ md5sum d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe
49fd4020bf4d7bd23956ea892e6860e9  d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe

┌──(kali㉿kali)-[~/Downloads]
└─$ sha1sum d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe
c5d8f155209badd278437d0e534648f8d5c35aae  d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe

┌──(kali㉿kali)-[~/Downloads]
└─$ sha256sum d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe
d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2  d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f
51a2.exe
```

The figure below shows that hash values are the same and we conclude that the malware we consider is the same as that considered by the website.



3.2: Yara rule

For yara rules, we checked in “Malpedia” site and get the rule for this malware from there. Should mentioned that the other name of this malware is “Proteus”

As below, we have downloaded this rule and extracted them as well. The name of the rule is win_proteus_auto.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ 7za x win.proteus.yar.zip

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (806C1),ASM,AES-NI)

Scanning the drive for archives:
1 file, 1631 bytes (2 KiB)

Extracting archive: win.proteus.yar.zip
--
Path = win.proteus.yar.zip
Type = zip
Physical Size = 1631

Everything is Ok

Size:       6035
Compressed: 1631

┌──(kali㉿kali)-[~/Downloads]
└─$ l
crime_wannacry.yar                                                  general_rats_malwareconfig.yar  Lazarus.yar       njRAT-v0.6.4.zip  redline_stealer.yar
d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe  infected.7z                     malwaresamples/   Qakbot.yar        win.proteus_auto.yar
d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.zip  jRAT.yar                        njRAT-v0.6.4/     RAT_Njrat.yar     win.proteus.yar.zip

┌──(kali㉿kali)-[~/Downloads]
└─$
```

```
rule win_proteus_auto {

    meta:
        author = "Felix Bilstein - yara-signator at cocacoding dot com"
        date = "2020-05-30"
        version = "1"
        description = "autogenerated rule brought to you by yara-signator"
        tool = "yara-signator v0.4.0"
        signator_config = "callsandjumps;datarefs;binvalue"
        malpedia_reference = "https://malpedia.caad.fkie.fraunhofer.de/details/win.proteus"
        malpedia_rule_date = "20200529"
        malpedia_hash = "92c362319514e5a6da26204961446caa3a8b32a8"
        malpedia_version = "20200529"
        malpedia_license = "CC BY-SA 4.0"
        malpedia_sharing = "TLP:WHITE"

    /* DISCLAIMER
     * The strings used in this rule have been automatically selected from the
     * disassembly of memory dumps and unpacked files, using yara-signator.
     * The code and documentation / approach is published here:
     * https://github.com/fxb-cocacoding/yara-signator
     * As Malpedia is used as data source, please note that for a given
     * number of families, only single samples are documented.
     * This likely impacts the degree of generalization these rules will offer.
     * Take the described generation method also into consideration when you
     * apply the rules in your use cases and assign them confidence levels.
     */


    strings:
        $sequence_0 = { e74a 6d b0b2 a1???????? 71fb }
            // n = 5, score = 100
            //   e74a                | out                 0x4a, eax
            //   6d                  | insd                dword ptr es:[edi], dx
            //   b0b2                | mov                 al, 0xb2
            //   a1????????          |
            //   71fb                | jno                 0xffffffd

        $sequence_1 = { b388 d957ef 1d9edbe60e e8???????? de9fab02e7c2 }
            // n = 5, score = 100
            //   b388                | mov                 bl, 0x88
            //   d957ef              | fst                 dword ptr [edi - 0x11]
            //   1d9edbe60e          | sbb                 eax, 0xee6db9e
            //   e8????????          |
            //   de9fab02e7c2        | ficomp              word ptr [edi - 0x3d18fd55]

        $sequence_2 = { c9 3a98b3bc15b5 7eb8 68aa362158 }
            // n = 4, score = 100
            //   c9                  | leave
            //   3a98b3bc15b5        | cmp                 bl, byte ptr [eax - 0x4aea434d]
            //   7eb8                | jle                 0xffffffba
            //   68aa362158          | push                0x582136aa
```

```
        $sequence_3 = { 19c4 3e31de fd 1576ea9453 }
            // n = 4, score = 100
            //   19c4                | sbb                 esp, eax
            //   3e31de              | xor                 esi, ebx
            //   fd                  | std
            //   1576ea9453          | adc                 eax, 0x5394ea76

        $sequence_4 = { 6d b0b2 a1???????? 71fb 70d9 2d6c0f17e9 4f }
            // n = 7, score = 100
            //   6d                  | insd                dword ptr es:[edi], dx
            //   b0b2                | mov                 al, 0xb2
            //   a1????????          |
            //   71fb                | jno                 0xfffffffd
            //   70d9                | jo                  0xffffffdb
            //   2d6c0f17e9          | sub                 eax, 0xe9170f6c
            //   4f                  | dec                 edi

        $sequence_5 = { a1???????? 71fb 70d9 2d6c0f17e9 4f 19c4 3e31de }
            // n = 7, score = 100
            //   a1????????          |
            //   71fb                | jno                 0xfffffffd
            //   70d9                | jo                  0xffffffdb
            //   2d6c0f17e9          | sub                 eax, 0xe9170f6c
            //   4f                  | dec                 edi
            //   19c4                | sbb                 esp, eax
            //   3e31de              | xor                 esi, ebx

        $sequence_6 = { a2???????? 1cff 5c 9f 12ff }
            // n = 5, score = 100
            //   a2????????          |
            //   1cff                | sbb                 al, 0xff
            //   5c                  | pop                 esp
            //   9f                  | lahf
            //   12ff                | adc                 bh, bh

        $sequence_7 = { 1bff 61 a2???????? 1cff 5c 9f 12ff }
            // n = 7, score = 100
            //   1bff                | sbb                 edi, edi
            //   61                  | popal
            //   a2????????          |
            //   1cff                | sbb                 al, 0xff
            //   5c                  | pop                 esp
            //   9f                  | lahf
            //   12ff                | adc                 bh, bh

        $sequence_8 = { 95 b388 d957ef 1d9edbe60e e8???????? }
            // n = 5, score = 100
            //   95                  | xchg                eax, ebp
            //   b388                | mov                 bl, 0x88
            //   d957ef              | fst                 dword ptr [edi - 0x11]
            //   1d9edbe60e          | sbb                 eax, 0xee6db9e
            //   e8????????          |
```

```
        $sequence_9 = { 71fb 70d9 2d6c0f17e9 4f }
            // n = 4, score = 100
            //   71fb                | jno                 0xfffffffd
            //   70d9                | jo                  0xffffffdb
            //   2d6c0f17e9          | sub                 eax, 0xe9170f6c
            //   4f                  | dec                 edi

    condition:
        7 of them and filesize < 5898240
}
```

3.3: Common Windows API used (if any is present)

No API

3.4: Network communication, if any are present (URLs and suspicious IPs)

Network Communication ⓘ

DNS Resolutions

+ 140.215.184.52.in-addr.arpa
+ 16.155.190.20.in-addr.arpa
+ 4.4.8.8.in-addr.arpa
+ 80.69.35.23.in-addr.arpa
+ prda.aadg.msidentity.com
+ proteus-network.ml
+ wpad

IP Traffic

114.114.114.114:53 (UDP)
13.107.39.203:80 (TCP)
20.99.133.109:443 (TCP)
20.99.184.37:443 (TCP)
218.85.157.99:53 (UDP)
23.216.147.62:443 (TCP)
52.251.79.25:443 (TCP)
a83f:8110:2f6a:9cff:2d68:9aff:2b66:98ff:53 (UDP)

3.5: Persistence mechanism (if is present)

Moving over to registry items, a lot of registry writes were performed by Proteus making it tough to sort out what was garbage and what was not. The most interesting item that I found was **C:\Users\admin\AppData\Roaming\chrome.exe** being written to key **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run**. This is used as a persistence mechanism to ensure that the application runs on startup.

Persistence TA0003

Registry Run Keys / Startup Folder T1547.001
ⓘ Creates an autostart registry key

Persistence TA0003

Registry Run Keys / Startup Folder T1547.001
⚠ Write self start registry common

3.6: Imported DLLs (Dynamically Loaded Libraries)

Imports

- mscoree.dll
  - _CorExeMain

3.7: Dropped files (if present)

Dropped Files (12)

Scanned	Detections	File type	Name
2022-10-30	0 / 70	Win32 DLL	Tamir.SharpSsh.dll
2022-11-18	59 / 72	Win32 EXE	d23b4a30f6b1f083ce86ef9d8ff434056865f6973f12cb075647d013906f51a2.exe
2021-11-30	52 / 67	Win32 EXE	chrome.exe
?	?	file	059B6748030EA8BE6FF9B34169BEAA61DF8D7756514A54C13C61F20F4F1F6DD5
?	?	file	4D8CFE0690BC07A25F70627DCC6B20A93F449176A49453179052F62421FE1718
?	?	file	4d8cfe0690bc07a25f70627dcc6b20a93f449176a49453179052f62421fe1718
?	?	file	758260026ca7c14b98c81c36c1f37a84cb69b5ccd24c9967b8b83d7d52be601f
?	?	file	9F1DD5CA1A61CBA953238C0B8F96F808436762078CC5AACE854D4E5CBDA4B744
?	?	file	9f1dd5ca1a61cba953238c0b8f96f808436762078cc5aace854d4e5cbda4b744
?	?	file	A884C47812B7DF78D5E7B4C6DE0036D4D12ACC4A8A9190B71063BA67DA6461C5
?	?	file	D23B4A30F6B1F083CE86EF9D8FF434056865F6973F12CB075647D013906F51A2
?	?	file	a884c47812b7df78d5e7b4c6de0036d4d12acc4a8a9190b71063ba67da6461c5

3.8: DNS info (if is present)

Domain	Detections	Created	Registrar
140.215.184.52.in-addr.arpa	0 / 96	-	-
16.155.190.20.in-addr.arpa	0 / 96	-	-
4.4.8.8.in-addr.arpa	0 / 96	-	-
80.69.35.23.in-addr.arpa	0 / 95	-	-
prda.aadg.msidentity.com	0 / 96	2016-03-21	MarkMonitor Inc.
proteus-network.ml	3 / 96	-	-

Contacted IP Addresses (7)

IP	Detections	Autonomous System	Country
114.114.114.114	2 / 96	174	CN
13.107.39.203	0 / 96	8068	US
20.99.133.109	0 / 96	8075	US
20.99.184.37	0 / 96	8075	US
218.85.157.99	0 / 96	4134	CN
23.216.147.62	0 / 96	20940	US
52.251.79.25	0 / 96	8075	US

3.9: Research on suspicious activities

Proteus malware is an example of a cyber threat which is multi-functional menace:

3.9.1: Proteus steels cryptocurrency

According to Bleeping Computer, Proteus capable of mining fo cryptocurrency by using SHA256 miner, CPUMiner and ZCashMiner, to steal Bitcoin, Litecoin, Zcash, and so on....

3.9.2: Proteus turns the computer into a SOCKS proxy

Proteus can enable the computer act as a socket secure proxy which can relays traffic between a client and a server for any type of network protocol. So that any malicious traffic can pass through an infected system. So that it will change your endpoints to an open door for cybercriminals. So that any other types of malwares can be downloaded and executed easily.

3.9.3: Proteus verifies stolen credentials

One of the most interesting and useful purposes of this malware is testing the validity of stolen login credentials.it is typically used by e-commerce sites.

3.9.4: Proteus logs keystrokes

Proteus is able to steal login credentials, verify them, use them to extract PII and then once it needed, route that PII to a remote client. It means it can work as a stealing machine.

Conclusion

Now after doing this Lab, we are able to install Yara in a Kali Linux machine and use it as malware detection by writing Yara Rules. Also, we learn how to apply rules if there is malicious file, how to extract information and see the matched between rules and the files

Furthermore, we learned how to analyze malwares and obtain useful information such as network communication, persistent mechanisms and so on from some websites as we performed in the 3rd task.