Assignment- 3

Harshit Rai
2017152

Part - I

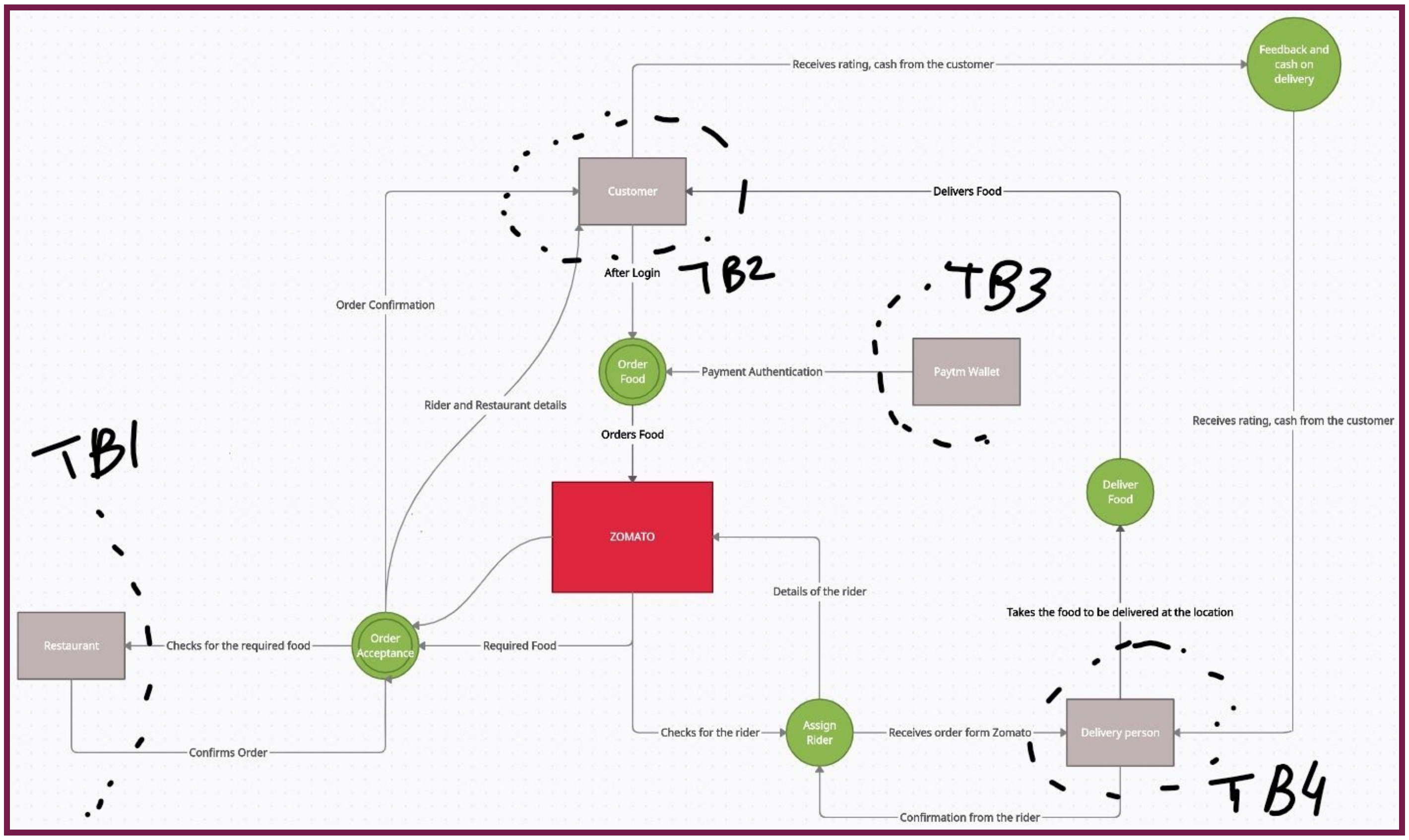
Question- 1

List of entry and exit points	● The nearest Amazon store will act as the main base station. ● There will be other base stations between the source to the destination. ● The destination base station.
Purpose of each entry and exit point	● The main Base station at amazon: It will serve the purpose of product attachment to the drone. Also, controlling/instructions of the drone. ● The intermediate base stations will ensure that the product is safe, look into technical issues if any. ● The destination base station will serve the purpose of the product finally delivery, or product return.
List of assets	● The product to be delivered. ● The drone. ● The valuable amazon's data in the drone (map or user's sensitive information).
Employee/drone handlers trust levels	● All the qualified employees are registered in amazon's database to operate the drone. ● The employees at various base stations have a valid Identification card to operate or change the settings of the drone.
Two use scenarios	● Employees at Amazon headquarters will maintain a flight recorder (flight's black box) which will keep track of the employee's drone simulation details. ● All the other employees serve the purpose of handling and maintaining the drone. Rest will work as before (before drone situation).
External security notes	● Customers have to enter an authentication pin in their mobile to make sure that they only ordered the product and no one else.

	● Drone can also be equipped with a facial recognition technique, to match the user's and the receiver's facial features. ● Drone will record the whole process of product delivery and pin entering. ● Educating and training customers about drone handling and security.
Internal security notes	● Location-based access control (LBAC) is implemented by the Amazon internal servers. ● Every flight's details (black box) are monitored by the amazon headquarters.
Any Three threats in the order of severity	● The adversaries could plant a bomb or some other threats in the product, drone by rerouting the drone by hacking it (High severity). ● They could hack into amazon's headquarters and stole, erase, temper all of their data (Medium severity). ● The adversaries could steal the product or the sensitive data of the user and amazon (Low severity).
Discoverability of the 3 threats	● If drones fly at comparatively low levels, people can catch or even destroy it by throwing various items at it. ● Drones are quite fragile and vulnerable to attacks while delivering products. Rainy or windy weather also plays a role in product delivery. ● Drones can be hacked if they are implemented on a normal security basis as nowadays people are more likely to get educated on their own and hack stuff.
Damage potential of the 3 threats	● Attackers can use the drone to attack the people, unauthorised surveillance or even data collection. ● If attackers are able to hack into the headquarter, it could result in a catastrophe for both employees and the customers. As leaking customers sensitive data can rip down the reputation of the company. ● Frequent product stealing, damaging the drone could lead to a huge loss to Amazon.
Mitigation strategy for all 3 threats	● Higle level of security must be implemented in the Amazon headquarters servers and drones. They must be inspected and monitored on a regular

	basis. ● Drone's GPS coordinates will get dynamically updated in the headquarters, if the drone gets lost or takes more time in delivery. ● Multi-factor authorization system must be enabled in the app and the drone. All the data must be end-to-end encryption using a strong hash function.
ROI for all 3 threats	● High. Harm to human lives can never be justified with money. ● Medium. Data can be collected again but it will consume time and resources. ● Low. Products can be manufactured again by the company, if they are stolen or damaged.

Question- 2


Receives rating, cash from the customer
Feedback and cash on delivery
Customer
Delivers Food
TB2
After Login
TB3
Order Confirmation
Order Food
Payment Authentication
Paytm Wallet
Rider and Restaurant details
Orders Food
Receives rating, cash from the customer
TB1
Deliver Food
ZOMATO
Details of the rider
Takes the food to be delivered at the location
Restaurant
Checks for the required food
Order Acceptance
Required Food
Checks for the rider
Assign Rider
Receives order form Zomato
Delivery person
Confirms Order
TB4
Confirmation from the rider

The data flow diagram shown above depicts the single Zomato food ordering cycle. Customer orders food -> restaurant accepts the order -> rider assigned -> food delivered. Various elements of the diagram is explained in detail below.

External Entities: customer, restaurant, delivery person, paytm wallet.
Process: assign rider, deliver food, feedback and COD.
Multiple Process: order food, order acceptance.
Trust Boundaries: TB1, TB2, TB3, TB4.
Data store: Zomato.

Process (Green circle)

The **assign rider** process ensures Zomato to check for an available rider/delivery person and give them the order accordingly. The **deliver food** process ensures the delivery person to reach the customer's delivery address and deliver the food. The **feedback and COD** ensures that the customer pays and gives a review to the rider.

Multiple Process (Green double circle)

The **order food** ensures that the customer is allowed to order food after logging into the Zomato app. The **order acceptance** ensures that the restaurant confirms the order and gives rider restaurant details to the customer.

Data Store (Red Rectangle)

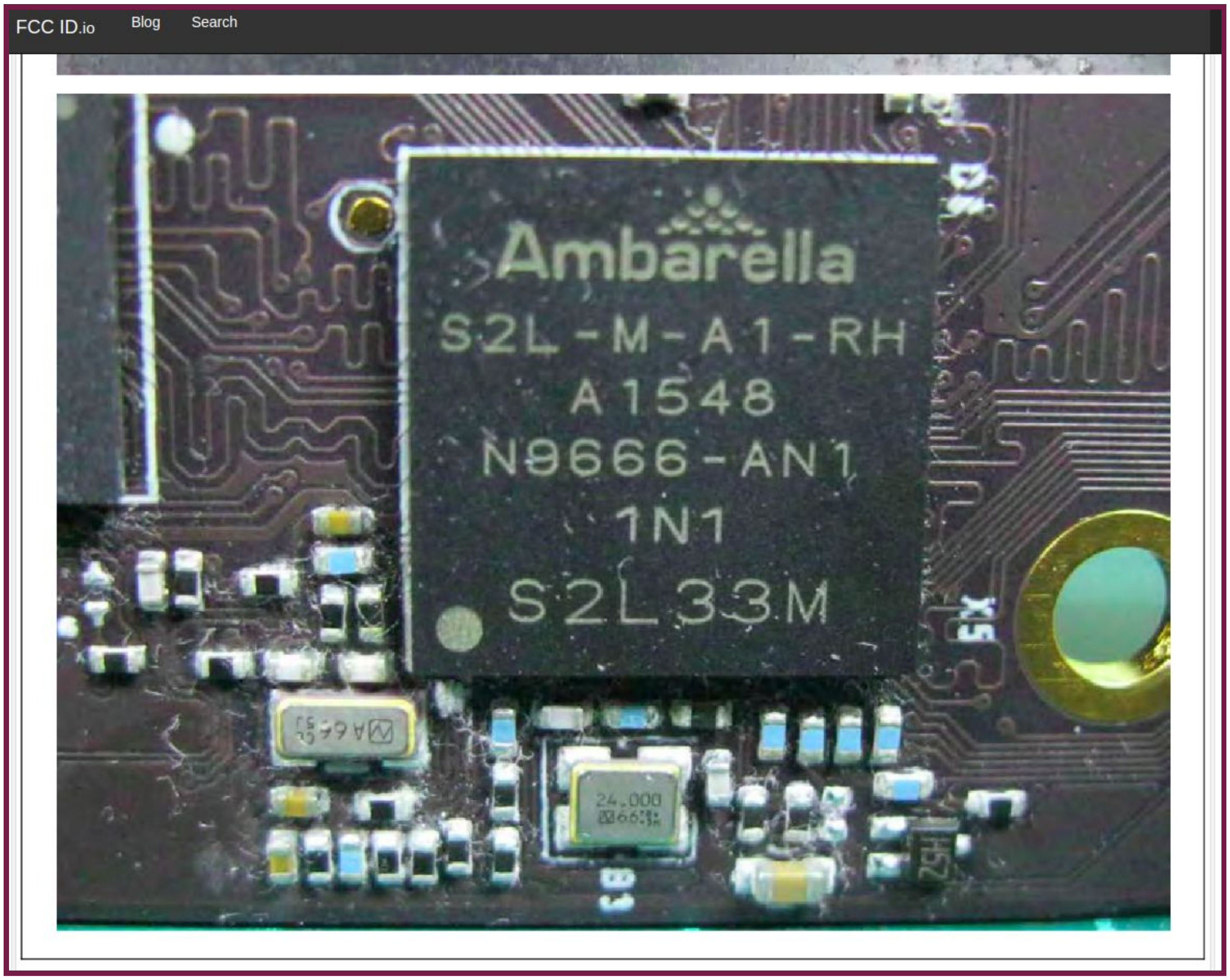
Zomato acts as the data store by storing the details of restaurants, delivery persons and customer information.

Trust Boundaries (Dotted line) and External Entities (Silver Rectangle)

Trust boundaries are added between the external entity and the process. **Customers** can hack the Zomato app and steal various information from the Zomato database. They are limited by **trust boundary 2**. **Delivery person** can eat the food to be delivered or add something in the food. They are limited by **trust boundary 4**. **Paytm wallet** can be hacked, malware or trojan can be installed in these softwares to steal money or other financial fraud could also happen. They are limited by **trust boundary 3**. **Restaurant** employees can also harm Zomato in many ways like giving false order confirmations, charge high prices in the name of Zomato, or social engineer the delivery person, in the restaurant's benefit. They are limited by **trust boundary 1**.

Question- 3

A) https://fccid.io/ZTM-SVI-1609-5/Internal-Photos/Internal-Photos-3183106



B) IC Number: **S2L-M-A1-RH.**
Ambarella chip is a low power consuming, highly efficient image processing, and computer vision system-on-chip. As it can perform various artificial intelligence tasks like object detection and person detection etc, they are widely used in cameras of autonomous vehicles, security cameras, and other robotics applications. In the given application, it is used for face detection and tracking, object recognition, motion detection, and various other image processing and video encoding processes.

C)

- DDR3 SDRAM: **NT5CB128M16FP-DII**. Packaging style: **BGA.**
- NAND EEPROM: **TC58BVG1S3HBAI4**. Packaging style: **Tray.**
- Single-band 1x1 Wi-Fi: **88W8801-NMD2.** Packaging style: **48-pin QFN.**

Question- 4

<u>Fingerprint</u>

Pros: The security is not so good but the usability is quite notable. During this pandemic period, this feature has helped almost everyone. This feature can be used even at low lighting conditions.

Cons: It unlocks the phone even when the person is wearing a mask, but as we sanitize our hands quite frequently and wear gloves, the usability decreases and the users have to suffer. Also, it fails to unlock when our fingers are sweaty or dirty. I have faced this situation a lot.

<u>Face ID</u>

Pros: Apple's Face ID is extremely secure, it would take approximately 1 in 1 million random people to falsely unlock it. It is obviously better than fingerprint in terms of security. The feature is so outstanding that it unlocks the phone almost instantly when the person's face is shown. The false-positive rate is quite impressive.

Cons: The main drawback of Face ID is that the person is not able to unlock the phone while they are wearing a mask, it has become more prominent in this pandemic period. Also, the users are not able to just take a look at their phone (to watch time or something) without unlocking the phone. The camera angle and light also play an important role while unlocking the phone.

Conclusion: There is always a tradeoff between security and usability, and this is an up to mark example of it. Both the features have their own pros and cons. After considering this pandemic period and the normal period, I would favor the Face ID more. Because of its extreme security and efficiency. Although it can not unlock when the person is wearing a mask, they can still use a fingerprint. But when we have only a fingerprint and the phone fails to unlock, we have to rely on the passcode (very less secure as compared to both of the above features). Face ID is more robust as compared to fingerprint.

Question- 5

A) 2 ways by which Teachable moment can be used in security design/systems are:
   - Teachable moment can be used in security by designing a dynamic view of memory. Like what happens in the memory while we download a malware, how it spreads in the system by downloading multiple malwares. This will help the user's to get a bigger picture of how memory is linked with networking. They can be encouraged towards sandbox systems through this.
   - Teachable moment can be used in security by designing a popup window while downloading malicious software from a website. For instance, the pop up should show a descriptive information of the downloaded file. Like its extension, size, the website's details, status after running a virus scan, etc. The proper explanation with minimum technical terms like what would happen if we run the file. Users can be encouraged towards memory forensics through this.

B) A website take down means that the ISP which is hosting the website has banned or forbidden it. This can happen when the website is containing some malicious or illegal stuff in it. The website can also be taken down from the internet if it has a direct or indirect link with malware, virus, and phishing or DDoS attack.

<u>The process involved in taking down a website:</u>

   - As we know the website's **domain name**, we can look up online for its **IP address.**
   - Then we have to contact the **DNS Host,** to report that a website under their domain name is part of a phishing or malware attack.
   - We can also make use of the **"Google Safe Browsing"** services, to report a phishing, malware website.

C) Embedded training can be used in security problems in the following manner:
   - Monthly sessions of cybersecurity can be organized in various institutions and organizations to make the general public aware of various cybercrimes and how to avoid them cleverly. Basic tips and ideas can be given to them in order to avoid or how to recognize malware or virus.
   - A 4-weeks course can be arranged by the government of the cybersecurity department. In which the person will be taught all the basics of computer security and other things which they are interested in learning.

Question- 6

Emotet and Agent Tesla are very intelligent malware. More specifically, Agent Tesla is spyware (keylogger or information stealer). Once they get installed in your system, it downloads various other malware. It spreads like a worm in your system and tries to steal your sensitive information. Their greatest defense mechanism which helps them to **deceive antivirus software** is that they themselves are not harmful, but they contain codes written **(macros)** in some hidden or gibberish language. Which further lets them access the internet and contact the attacker. As they are **dynamic malware,** they maintain a **command-and-control server** (adversary can command the infected system), which helps them in regular updation. The malware changes repeatedly in order to get deceived by antivirus software.

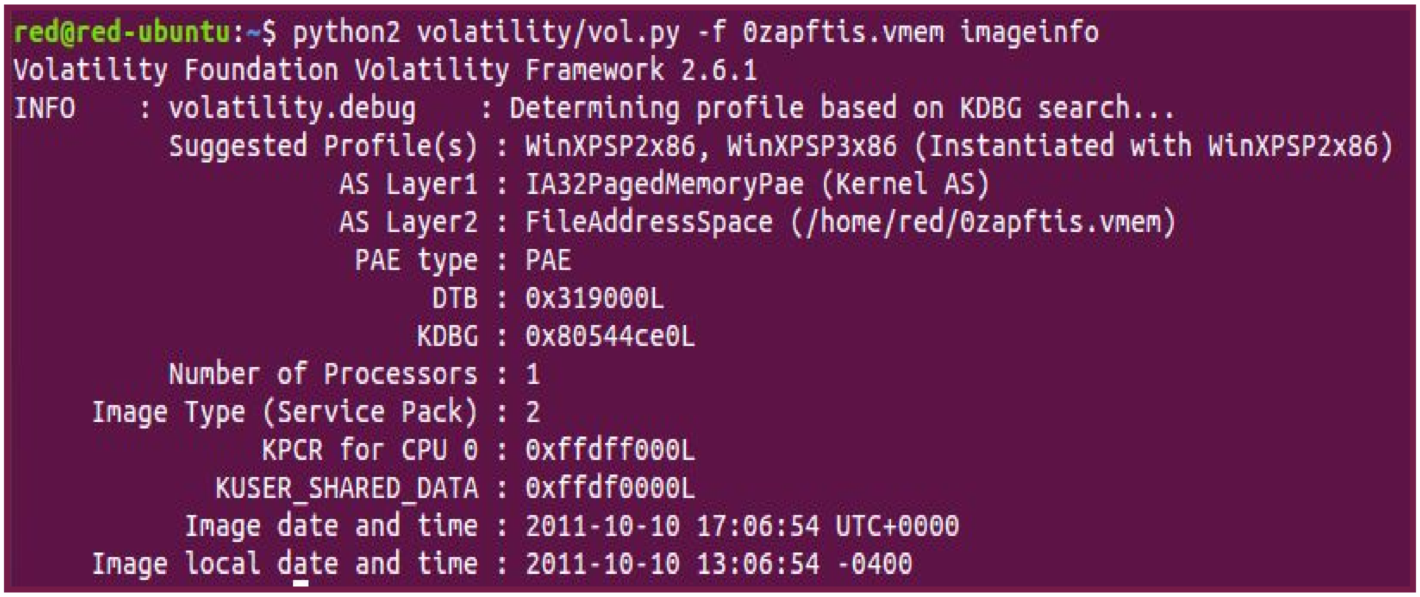
It is hard to detect Agent Tesla using **static analysis** because they are very clever malware. They know when they are being investigated via **sandbox**. Also, as this malware bypasses the antivirus (by tampering with the macros), we need to do a more intelligent investigation. That is done by doing the **memory analysis.**

Emotet's stealthiness can be increased by making it more immune to sandbox systems. This can be done by also making it a **RAT (Remote Access Trojan)** malware. Emotet can learn techniques like **screen monitoring, collecting user's inputs, etc** from Agent Tesla.

Part - II

Question- 7

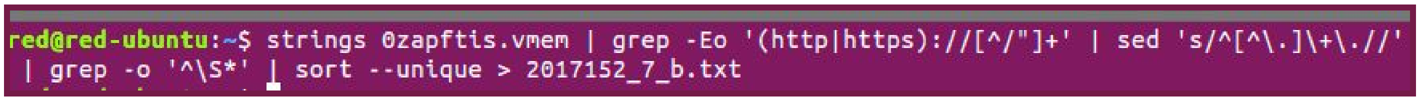
A)



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/red/0zapftis.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2011-10-10 17:06:54 UTC+0000
     Image local date and time : 2011-10-10 13:06:54 -0400
```

Platform: WinXPSP2x86, WinXPSP3x86

B)

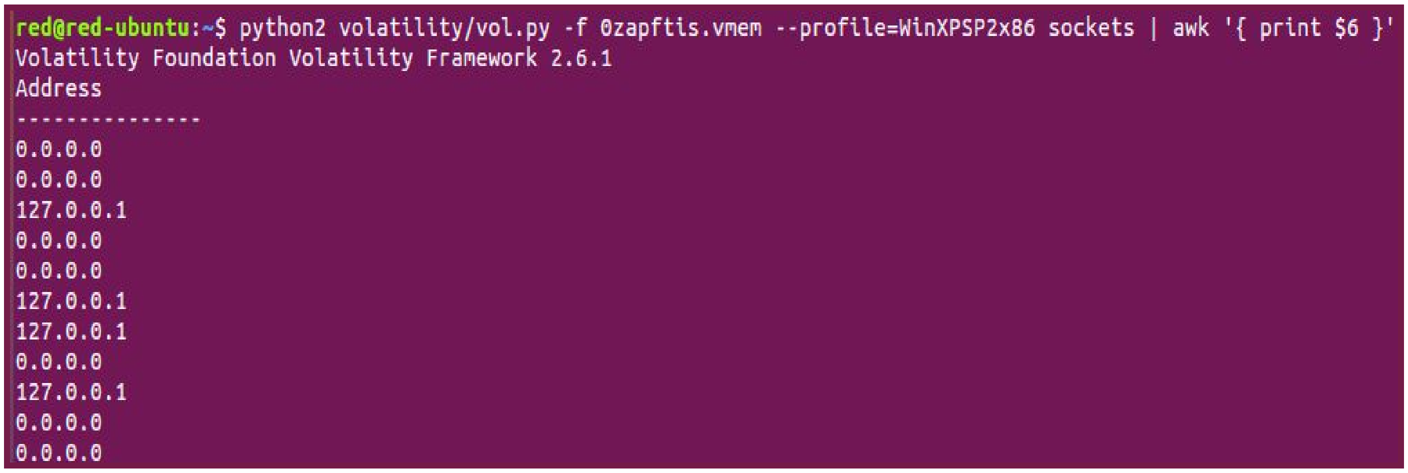


```
red@red-ubuntu:~$ strings 0zapftis.vmem | grep -Eo '(http|https)://[^/"]+' | sed 's/^[^\.]\+\.//'
 | grep -o '^\S*' | sort --unique > 2017152_7_b.txt
```

Command: strings 0zapftis.vmem | grep -Eo '(http|https)://[^/"]+' | sed 's/^[^\.]\+\.//' | grep -o '^\S*' | sort --unique > 2017152_7_b.txt

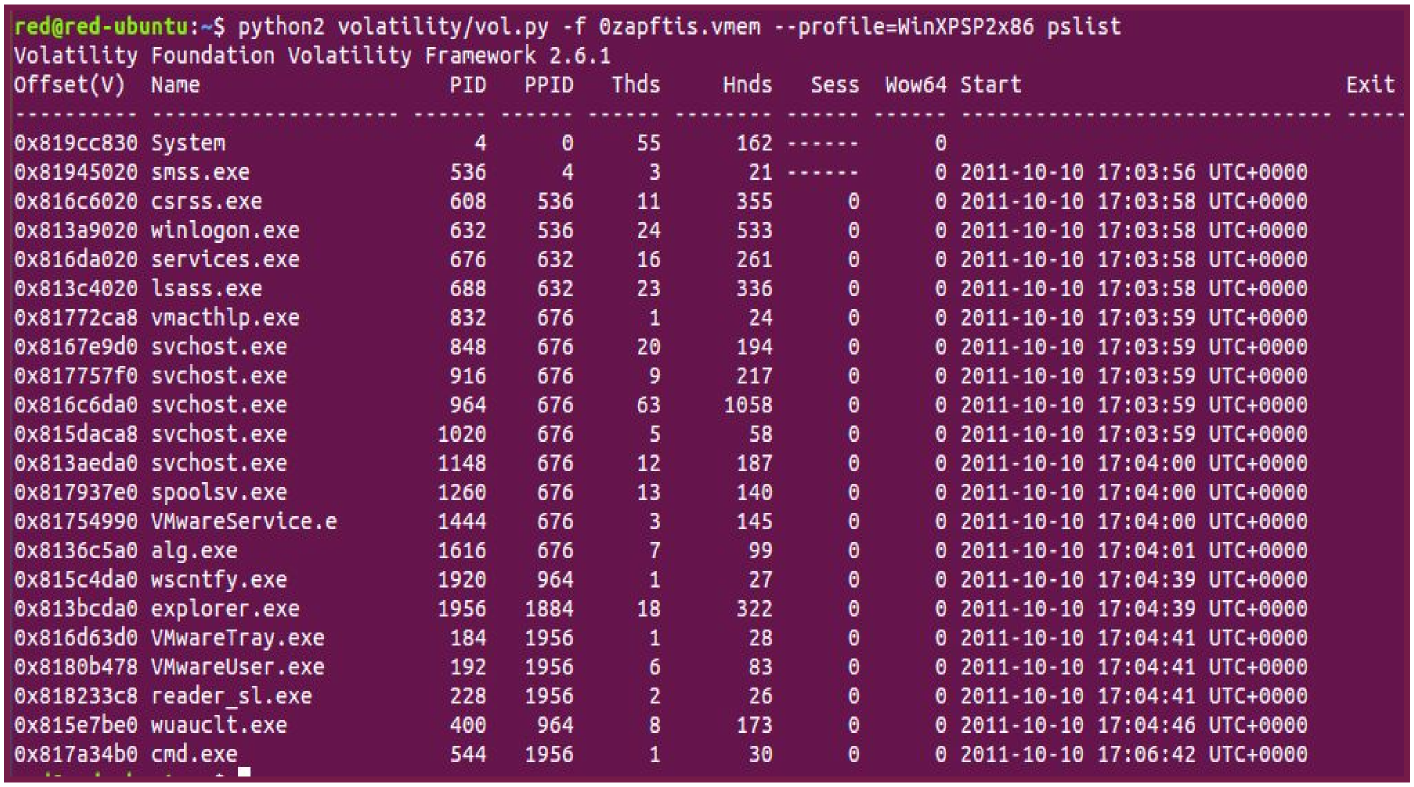
No website is suspicious. As because I checked all the websites in the .txt file on this website. It showed that all the websites are Safe.

C)



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 sockets | awk '{ print $6 }'
Volatility Foundation Volatility Framework 2.6.1
Address
---------------
0.0.0.0
0.0.0.0
127.0.0.1
0.0.0.0
0.0.0.0
127.0.0.1
127.0.0.1
0.0.0.0
127.0.0.1
0.0.0.0
0.0.0.0
```

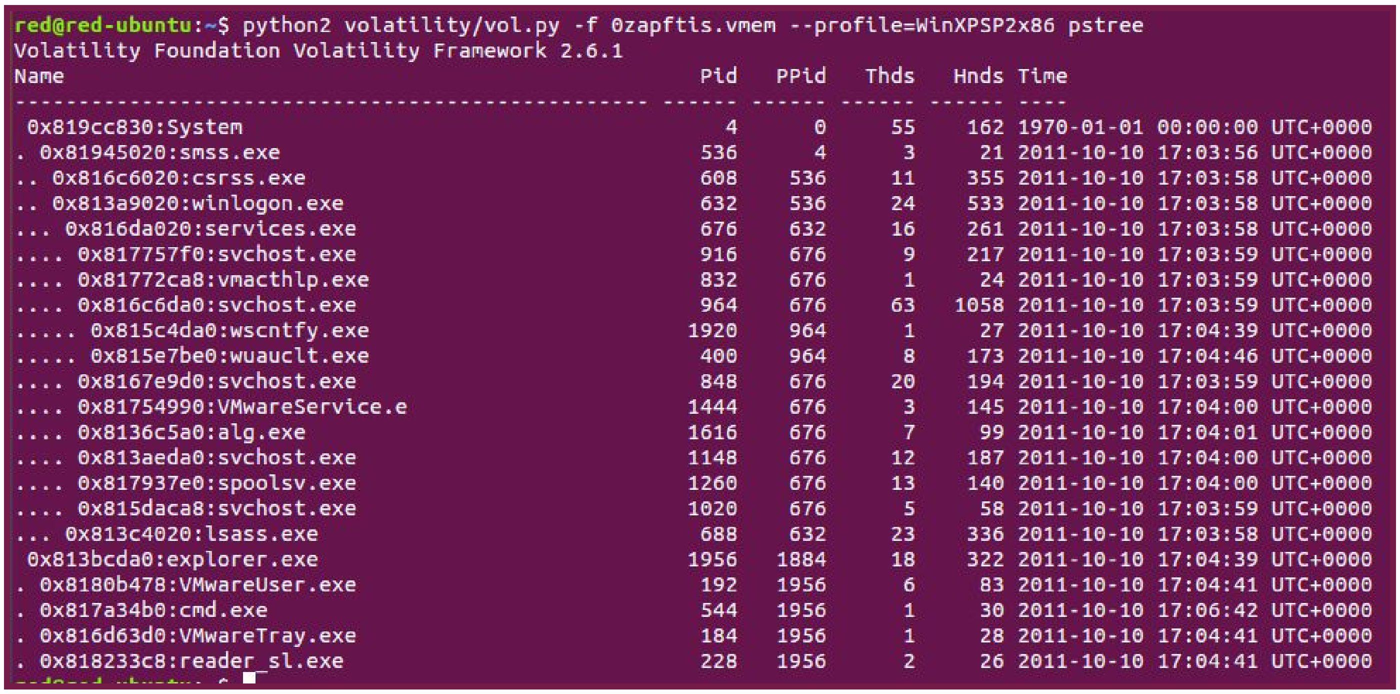
D)



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ -----
0x819cc830 System                    4      0     55      162 ------      0
0x81945020 smss.exe                536      4      3       21 ------      0 2011-10-10 17:03:56 UTC+0000
0x816c6020 csrss.exe               608    536     11      355      0      0 2011-10-10 17:03:58 UTC+0000
0x813a9020 winlogon.exe            632    536     24      533      0      0 2011-10-10 17:03:58 UTC+0000
0x816da020 services.exe            676    632     16      261      0      0 2011-10-10 17:03:58 UTC+0000
0x813c4020 lsass.exe               688    632     23      336      0      0 2011-10-10 17:03:58 UTC+0000
0x81772ca8 vmacthlp.exe            832    676      1       24      0      0 2011-10-10 17:03:59 UTC+0000
0x8167e9d0 svchost.exe             848    676     20      194      0      0 2011-10-10 17:03:59 UTC+0000
0x817757f0 svchost.exe             916    676      9      217      0      0 2011-10-10 17:03:59 UTC+0000
0x816c6da0 svchost.exe             964    676     63     1058      0      0 2011-10-10 17:03:59 UTC+0000
0x815daca8 svchost.exe            1020    676      5       58      0      0 2011-10-10 17:03:59 UTC+0000
0x813aeda0 svchost.exe            1148    676     12      187      0      0 2011-10-10 17:04:00 UTC+0000
0x817937e0 spoolsv.exe            1260    676     13      140      0      0 2011-10-10 17:04:00 UTC+0000
0x81754990 VMwareService.e        1444    676      3      145      0      0 2011-10-10 17:04:00 UTC+0000
0x8136c5a0 alg.exe                1616    676      7       99      0      0 2011-10-10 17:04:01 UTC+0000
0x815c4da0 wscntfy.exe            1920    964      1       27      0      0 2011-10-10 17:04:39 UTC+0000
0x813bcda0 explorer.exe           1956   1884     18      322      0      0 2011-10-10 17:04:39 UTC+0000
0x816d63d0 VMwareTray.exe          184   1956      1       28      0      0 2011-10-10 17:04:41 UTC+0000
0x8180b478 VMwareUser.exe          192   1956      6       83      0      0 2011-10-10 17:04:41 UTC+0000
0x818233c8 reader_sl.exe           228   1956      2       26      0      0 2011-10-10 17:04:41 UTC+0000
0x815e7be0 wuauclt.exe             400    964      8      173      0      0 2011-10-10 17:04:46 UTC+0000
0x817a34b0 cmd.exe                 544   1956      1       30      0      0 2011-10-10 17:06:42 UTC+0000
```

All the processes in the 2nd column were running at that time.

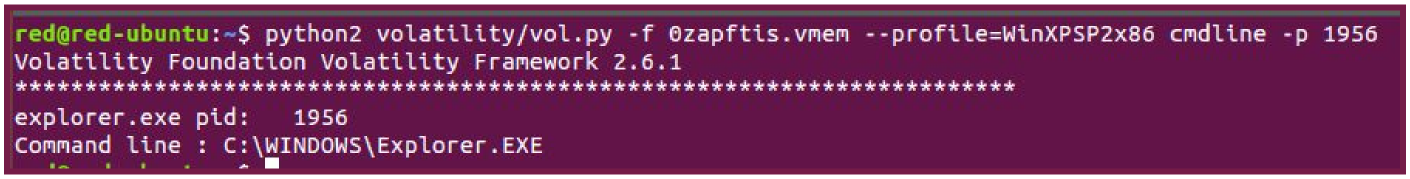
E)



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x819cc830:System                                      4      0     55    162 1970-01-01 00:00:00 UTC+0000
. 0x81945020:smss.exe                                 536      4      3     21 2011-10-10 17:03:56 UTC+0000
.. 0x816c6020:csrss.exe                               608    536     11    355 2011-10-10 17:03:58 UTC+0000
.. 0x813a9020:winlogon.exe                            632    536     24    533 2011-10-10 17:03:58 UTC+0000
... 0x816da020:services.exe                           676    632     16    261 2011-10-10 17:03:58 UTC+0000
.... 0x817757f0:svchost.exe                           916    676      9    217 2011-10-10 17:03:59 UTC+0000
.... 0x81772ca8:vmacthlp.exe                          832    676      1     24 2011-10-10 17:03:59 UTC+0000
.... 0x816c6da0:svchost.exe                           964    676     63   1058 2011-10-10 17:03:59 UTC+0000
..... 0x815c4da0:wscntfy.exe                         1920    964      1     27 2011-10-10 17:04:39 UTC+0000
..... 0x815e7be0:wuauclt.exe                          400    964      8    173 2011-10-10 17:04:46 UTC+0000
.... 0x8167e9d0:svchost.exe                           848    676     20    194 2011-10-10 17:03:59 UTC+0000
.... 0x81754990:VMwareService.e                      1444    676      3    145 2011-10-10 17:04:00 UTC+0000
.... 0x8136c5a0:alg.exe                              1616    676      7     99 2011-10-10 17:04:01 UTC+0000
.... 0x813aeda0:svchost.exe                          1148    676     12    187 2011-10-10 17:04:00 UTC+0000
.... 0x817937e0:spoolsv.exe                          1260    676     13    140 2011-10-10 17:04:00 UTC+0000
.... 0x815daca8:svchost.exe                          1020    676      5     58 2011-10-10 17:03:59 UTC+0000
... 0x813c4020:lsass.exe                              688    632     23    336 2011-10-10 17:03:58 UTC+0000
 0x813bcda0:explorer.exe                             1956   1884     18    322 2011-10-10 17:04:39 UTC+0000
. 0x8180b478:VMwareUser.exe                           192   1956      6     83 2011-10-10 17:04:41 UTC+0000
. 0x817a34b0:cmd.exe                                  544   1956      1     30 2011-10-10 17:06:42 UTC+0000
. 0x816d63d0:VMwareTray.exe                           184   1956      1     28 2011-10-10 17:04:41 UTC+0000
. 0x818233c8:reader_sl.exe                            228   1956      2     26 2011-10-10 17:04:41 UTC+0000
```

The processes with PID 192, 544, 184, and 228 have the same parent, which has PID 1956. The parent process (1956) belongs to the **explorer.exe** program.

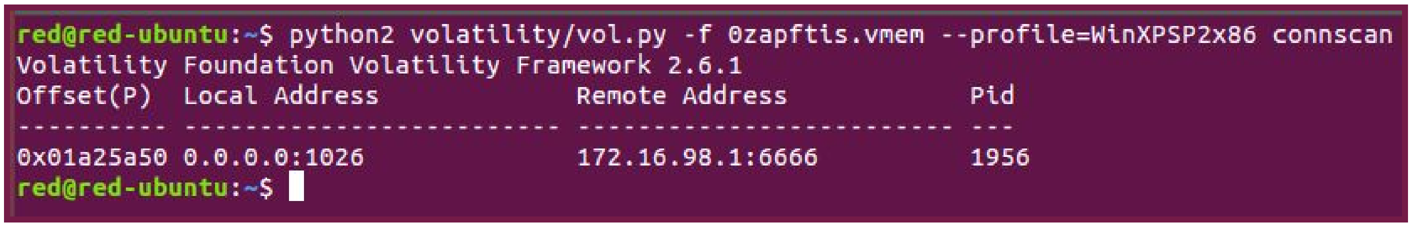
F)



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 cmdline -p 1956
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
explorer.exe pid:   1956
Command line : C:\WINDOWS\Explorer.EXE
```

C:\WINDOWS\Explorer.EXE, invoked the above 4 processes.

G)

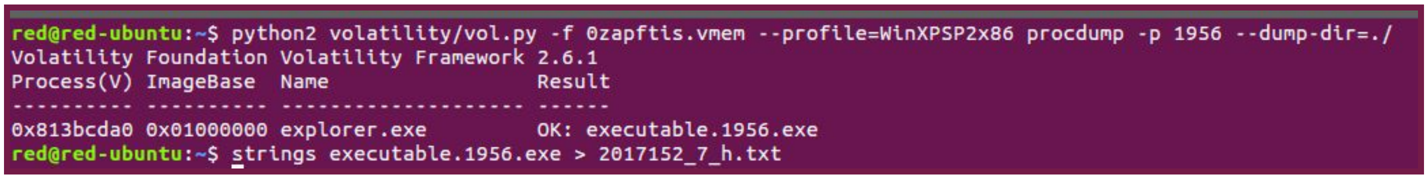


```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x01a25a50 0.0.0.0:1026              172.16.98.1:6666          1956
red@red-ubuntu:~$
```

The process with PID 1956 seems suspicious because out of all the processes only this process is using an active TCP connection. Also, it is receiving information from port 6666. On searching the details of TCP port 6666, I found out that this port is not secure. A high number of malicious activities like a virus or trojan horses had been shared through this port in the past. This port can be a real threat to us.

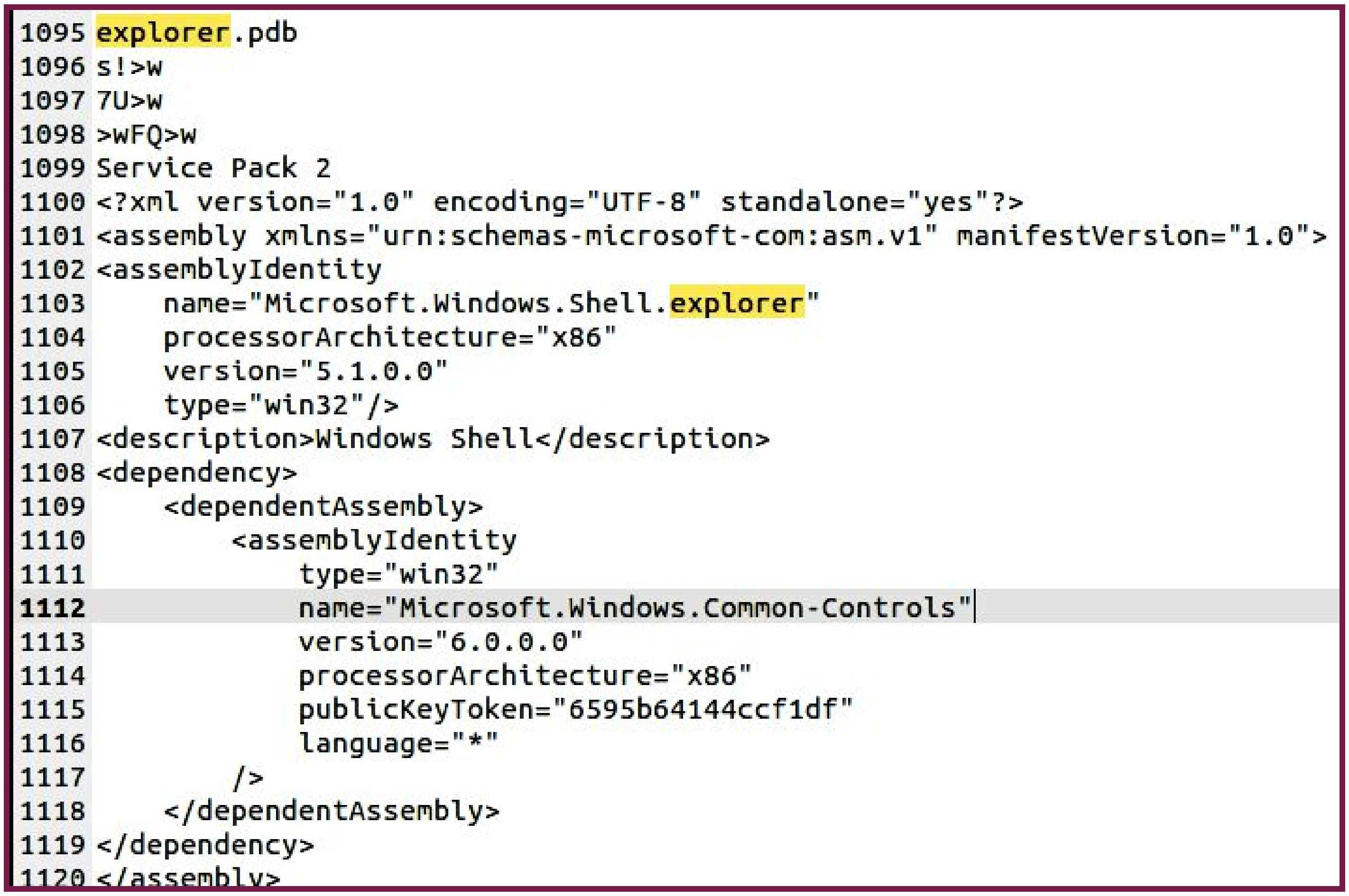
H)

Method 1



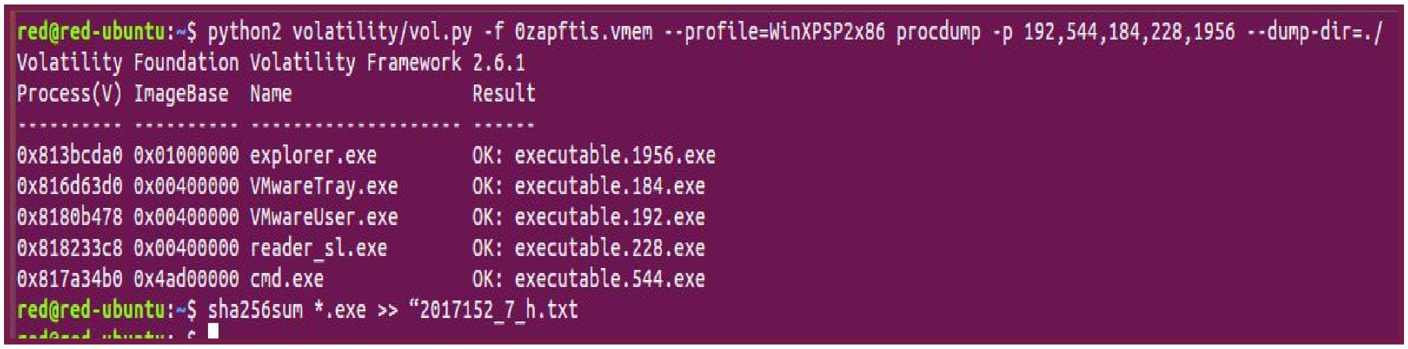
```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 procdump -p 1956 --dump-dir=./
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x813bcda0 0x01000000 explorer.exe         OK: executable.1956.exe
red@red-ubuntu:~$ strings executable.1956.exe > 2017152_7_h.txt
```

I took the process dump of process 1956 (explorer.exe) and saved its strings output in a .txt file. Below is a portion of code of the .txt file. Here we can see that this process uses **“Windows powershell”.** Powershell is commonly used by malwares in order to get access to the internet. There is a possibility that this file contains trojan because the following code snippet contains a malicious code. It can be a malicious code of macros.



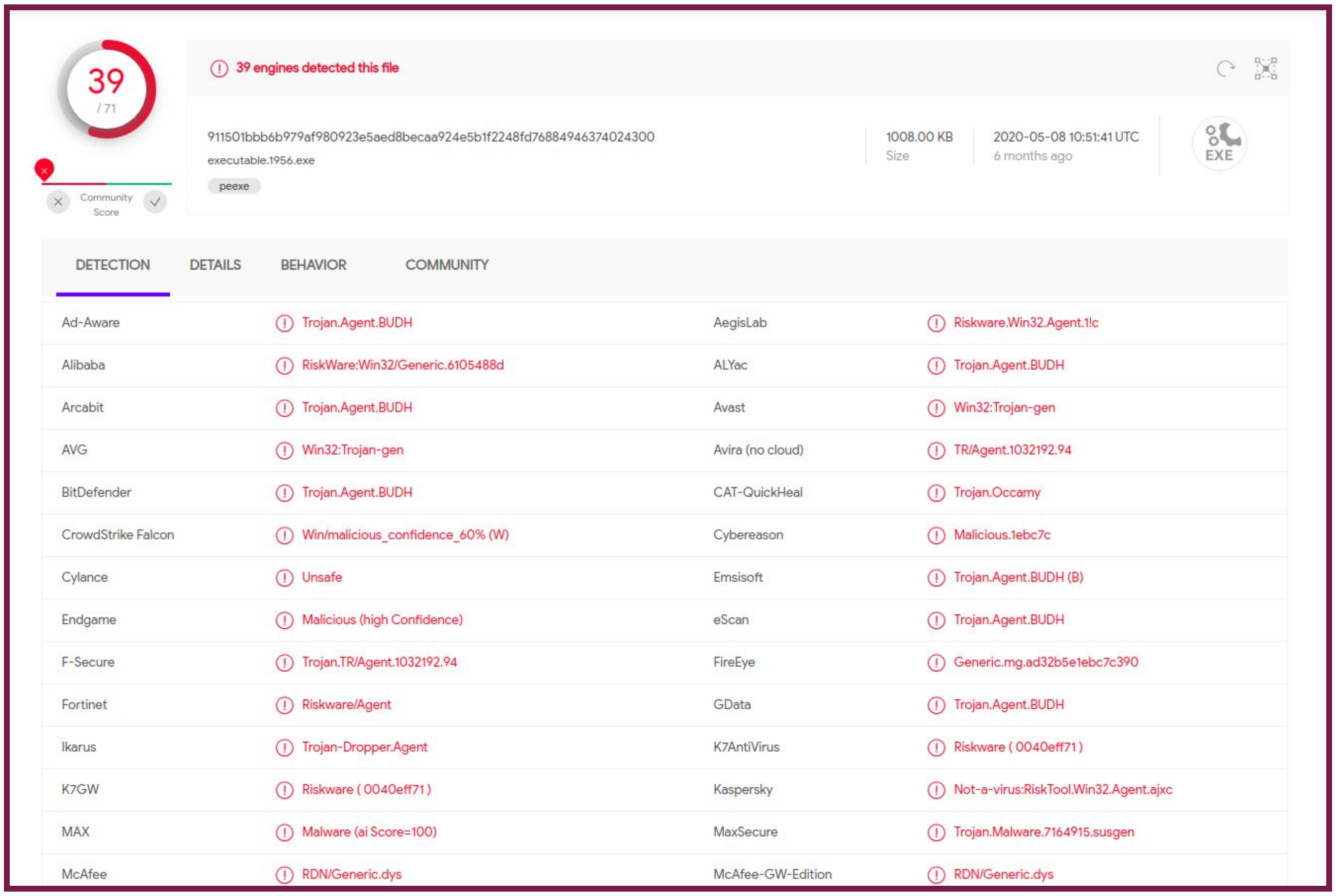
```
explorer.pdb
s!>w
7U>w
>wFQ>w
Service Pack 2
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
    name="Microsoft.Windows.Shell.explorer"
    processorArchitecture="x86"
    version="5.1.0.0"
    type="win32"/>
<description>Windows Shell</description>
<dependency>
    <dependentAssembly>
        <assemblyIdentity
            type="win32"
            name="Microsoft.Windows.Common-Controls"
            version="6.0.0.0"
            processorArchitecture="x86"
            publicKeyToken="6595b64144ccf1df"
            language="*"
        />
    </dependentAssembly>
</dependency>
</assembly>
```

Method 2



```
red@red-ubuntu:~$ python2 volatility/vol.py -f 0zapftis.vmem --profile=WinXPSP2x86 procdump -p 192,544,184,228,1956 --dump-dir=./
Volatility Foundation Volatility Framework 2.6.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x813bcda0 0x01000000 explorer.exe         OK: executable.1956.exe
0x816d63d0 0x00400000 VMwareTray.exe       OK: executable.184.exe
0x8180b478 0x00400000 VMwareUser.exe       OK: executable.192.exe
0x818233c8 0x00400000 reader_sl.exe        OK: executable.228.exe
0x817a34b0 0x4ad00000 cmd.exe              OK: executable.544.exe
red@red-ubuntu:~$ sha256sum *.exe >> "2017152_7_h.txt
```

Firstly, I created the dump files for all the suspicious processes. Secondly, I found out their "SHA-256" hash codes. Then I checked on an online virus database website VirusTotal, which told me that processes with PID 1956, 544, and 228 are **Trojan Horses.** The processes with PID 184 and 192 are safe. I have also attached the snapshots from the website.



39 / 71

Community Score

39 engines detected this file

911501bbb6b979af980923e5aed8becaa924e5b1f2248fd76884946374024300
executable.1956.exe
peexe

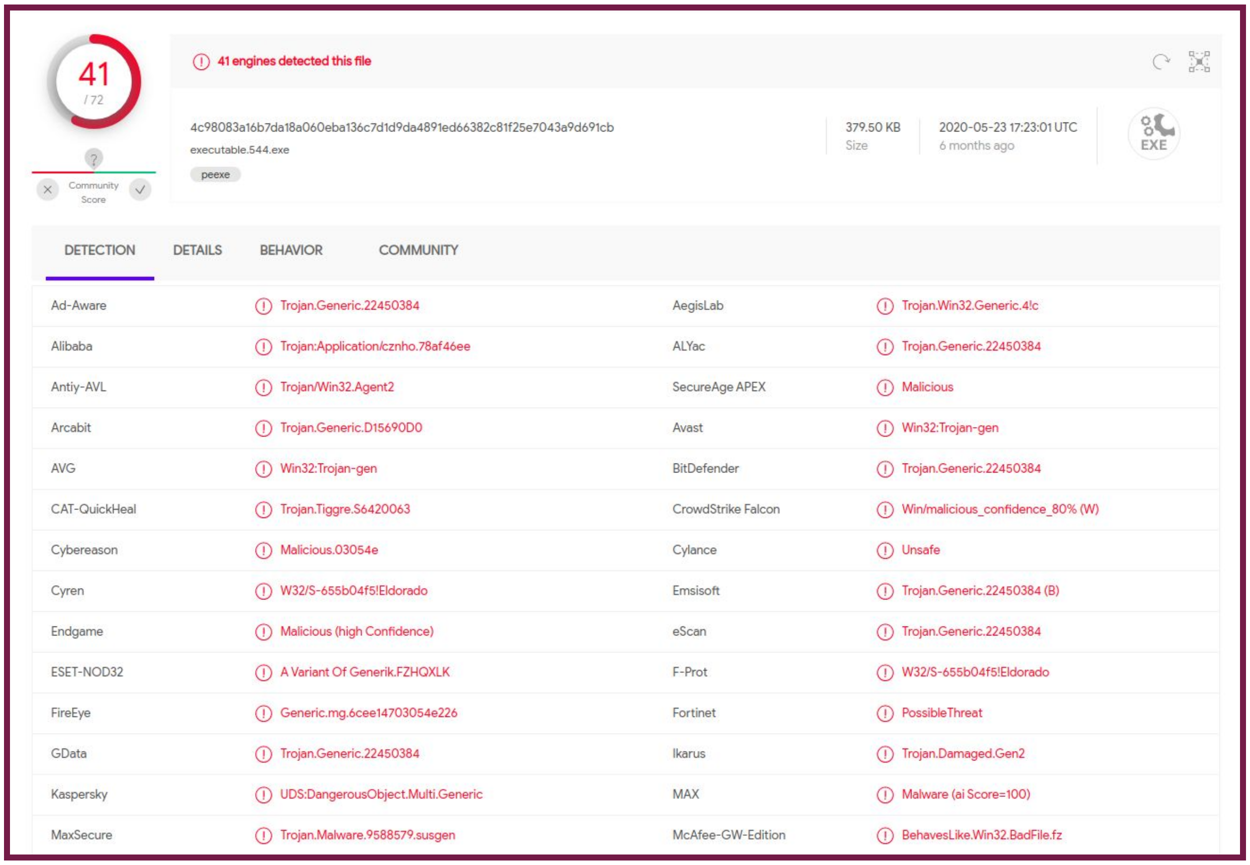
1008.00 KB Size

2020-05-08 10:51:41 UTC 6 months ago

EXE

DETECTION DETAILS BEHAVIOR COMMUNITY

Ad-Aware	Trojan.Agent.BUDH	AegisLab	Riskware.Win32.Agent.1!c
Alibaba	RiskWare:Win32/Generic.6105488d	ALYac	Trojan.Agent.BUDH
Arcabit	Trojan.Agent.BUDH	Avast	Win32:Trojan-gen
AVG	Win32:Trojan-gen	Avira (no cloud)	TR/Agent.1032192.94
BitDefender	Trojan.Agent.BUDH	CAT-QuickHeal	Trojan.Occamy
CrowdStrike Falcon	Win/malicious_confidence_60% (W)	Cybereason	Malicious.1ebc7c
Cylance	Unsafe	Emsisoft	Trojan.Agent.BUDH (B)
Endgame	Malicious (high Confidence)	eScan	Trojan.Agent.BUDH
F-Secure	Trojan.TR/Agent.1032192.94	FireEye	Generic.mg.ad32b5e1ebc7c390
Fortinet	Riskware/Agent	GData	Trojan.Agent.BUDH
Ikarus	Trojan-Dropper.Agent	K7AntiVirus	Riskware (0040eff71)
K7GW	Riskware (0040eff71)	Kaspersky	Not-a-virus:RiskTool.Win32.Agent.ajxc
MAX	Malware (ai Score=100)	MaxSecure	Trojan.Malware.7164915.susgen
McAfee	RDN/Generic.dys	McAfee-GW-Edition	RDN/Generic.dys



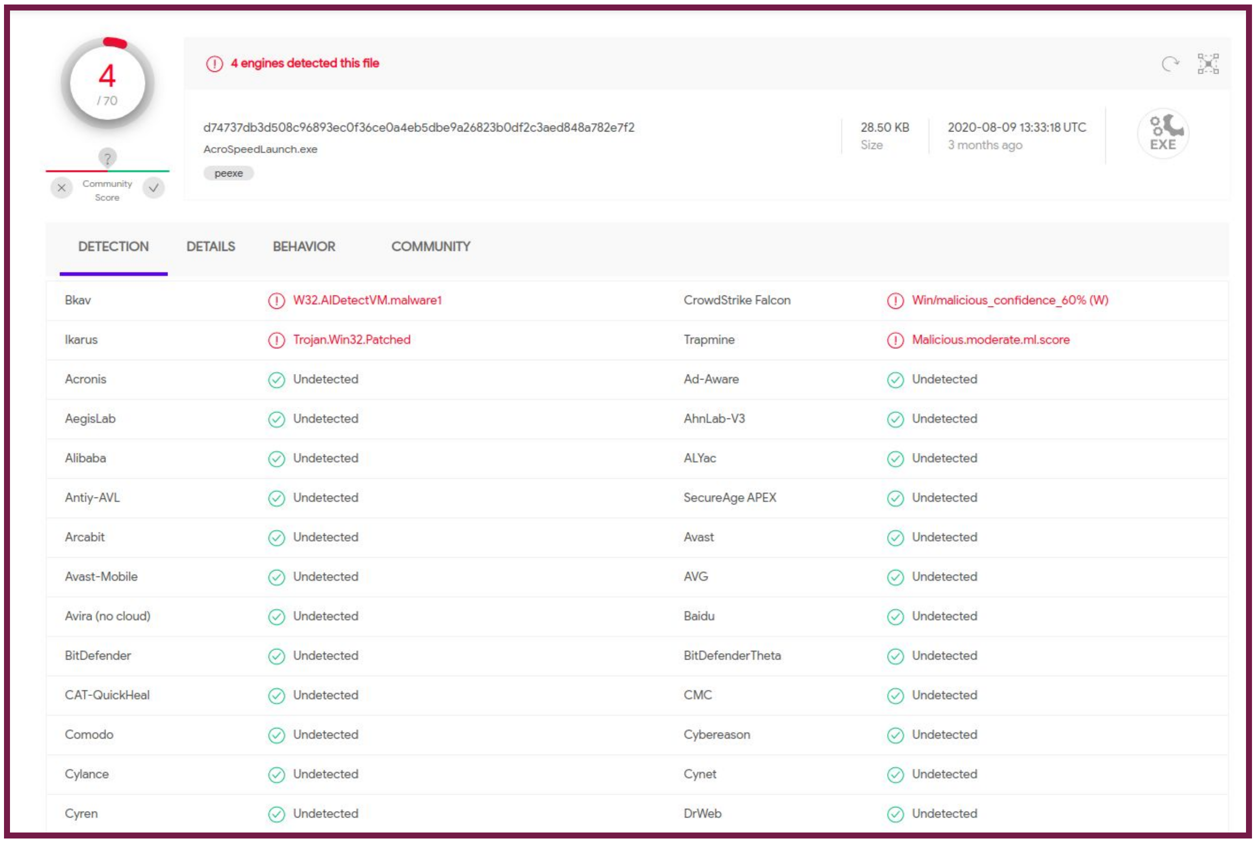
41 engines detected this file

4c98083a16b7da18a060eba136c7d1d9da4891ed66382c81f25e7043a9d691cb
executable.544.exe
peexe

379.50 KB
Size

2020-05-23 17:23:01 UTC
6 months ago

Ad-Aware	Trojan.Generic.22450384	AegisLab	Trojan.Win32.Generic.4!c
Alibaba	Trojan:Application/cznho.78af46ee	ALYac	Trojan.Generic.22450384
Antiy-AVL	Trojan/Win32.Agent2	SecureAge APEX	Malicious
Arcabit	Trojan.Generic.D15690D0	Avast	Win32:Trojan-gen
AVG	Win32:Trojan-gen	BitDefender	Trojan.Generic.22450384
CAT-QuickHeal	Trojan.Tiggre.S6420063	CrowdStrike Falcon	Win/malicious_confidence_80% (W)
Cybereason	Malicious.03054e	Cylance	Unsafe
Cyren	W32/S-655b04f5!Eldorado	Emsisoft	Trojan.Generic.22450384 (B)
Endgame	Malicious (high Confidence)	eScan	Trojan.Generic.22450384
ESET-NOD32	A Variant Of Generik.FZHQXLK	F-Prot	W32/S-655b04f5!Eldorado
FireEye	Generic.mg.6cee14703054e226	Fortinet	PossibleThreat
GData	Trojan.Generic.22450384	Ikarus	Trojan.Damaged.Gen2
Kaspersky	UDS:DangerousObject.Multi.Generic	MAX	Malware (ai Score=100)
MaxSecure	Trojan.Malware.9588579.susgen	McAfee-GW-Edition	BehavesLike.Win32.BadFile.fz



4 engines detected this file

d74737db3d508c96893ec0f36ce0a4eb5dbe9a26823b0df2c3aed848a782e7f2
AcroSpeedLaunch.exe
peexe

28.50 KB
Size

2020-08-09 13:33:18 UTC
3 months ago

Bkav	W32.AIDetectVM.malware1	CrowdStrike Falcon	Win/malicious_confidence_60% (W)
Ikarus	Trojan.Win32.Patched	Trapmine	Malicious.moderate.ml.score
Acronis	Undetected	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	SecureAge APEX	Undetected
Arcabit	Undetected	Avast	Undetected
Avast-Mobile	Undetected	AVG	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected
BitDefender	Undetected	BitDefenderTheta	Undetected
CAT-QuickHeal	Undetected	CMC	Undetected
Comodo	Undetected	Cybereason	Undetected
Cylance	Undetected	Cynet	Undetected
Cyren	Undetected	DrWeb	Undetected

References

- S2L IP Camera Processor
- TC58BVG1S3HBAI4 Kioxia America, Inc. | Integrated Circuits (ICs)
- https://www.nxp.com/docs/en/data-sheet/88W8801DS.pdf
- https://www.nanya.com/en/Product/3767/NT5CB128M16FP-DII