



Netradyne Standard Operating Procedure Malware  
Analysis  
v1.0

Internal and Confidential

---

## *TABLE OF CONTENTS*

---

NETRADYNE STANDARD OPERATING PROCEDURE MALWARE ANALYSIS .....	0
<i>Document Control</i> .....	2
<b>1 OVERVIEW .....</b>	<b>3</b>
<b>2 SCOPE .....</b>	<b>3</b>
<b>3 PRE-REQUISITES / ASSUMPTIONS.....</b>	<b>3</b>
<b>4 STEPS TO ACCESS SANDBOX MACHINE .....</b>	<b>3</b>
<b>5 MALWARE ANALYSIS IMPORTANT TERMS .....</b>	<b>3</b>
5.1 TYPES OF MALWARES .....	3
5.2 TYPES OF MALWARE ANALYSIS.....	4
<b>6 BASIC TOOLS AND STEPS FOR MALWARE ANALYSIS .....</b>	<b>4</b>
<b>7 MALWARE SANDBOXING USING HYBRID ANALYSIS.....</b>	<b>12</b>
<b>8 EXAMINING MALWARE FILE USING PE STUDIO (STATIC ANALYSIS) .....</b>	<b>19</b>
<b>9 EXCEPTION.....</b>	<b>23</b>
<b>10 REFERENCES.....</b>	<b>24</b>
10.1 LOGGING .....	24
10.2 IOC.....	24
<b>11 APPENDIX A: DOCUMENT RACI MATRIX .....</b>	<b>25</b>

**Document Control**

<b>Document ID</b>	NDMA2022001
<b>Document Name</b>	Netradyne Standard Operating Procedure Malware Analysis
<b>Document Status</b>	Draft
<b>Document Released Date</b>	04/May/2022
<b>Document Author</b>	Netradyne Infosec Team
<b>Document Content Contributors</b>	Prathamesh Padoskar
<b>Document Signatory</b>	Saravanan Sankaran
<b>Document Owner</b>	Saravanan Sankaran
<b>Document Version</b>	V1.0
<b>Information Classification</b>	Internal

**Document Edit History**

<b>Version</b>	<b>Date</b>	<b>Additions/Modifications</b>	<b>Prepared/Revised By</b>
0.1	28/Apr/2022	Draft version	Prathamesh Padoskar
0.2	04/May/2022	Changes incorporated post initial review	Prathamesh Padoskar
1.0	25/May/2022	Initial Release	Infosec

**Document Review/Approval**

<b>Date</b>	<b>Signatory Name</b>	<b>Organization/Signatory Title</b>	<b>Comments</b>
25/May/2022	Saravanan Sankaran	Senior Director, Infosec & IT	

**Distribution of Final Document**

<b>Name</b>	<b>Organization/Title</b>

## 1 Overview

One of the most widespread threats to cyber security in today's world of unlimited Internet access is malware. In recent times, the malware being designed are with the ability to transform their code and to hide quietly in the systems of the unsuspecting users. Malware analysis is the process of performing analysis of the malware and understanding its actions and behaviour. It is of two types- static and dynamic analysis. Static analysis is performed by observing the source code of the malware and drawing conclusions based on it. Dynamic analysis is the analysis performed by executing the piece of code and noting its actions. Malware analysis is an important and relevant task, for the advanced forms of malware these days are often not even detectable by commonly available anti-virus software.

## 2 Scope

Test Malware and check for relevant IOC create by malware and put appropriate actions on it. Sandbox environment is created; malware can be tested on this environment. Based on analysis of malware we will get IOC, we can use those IOC and submit on various security tools to prevent it from happening again. You can use various tools through Sandbox machine to test malware.

## 3 Pre-requisites / Assumptions

Only authorized users have access to sandbox machine. Members are listed below. Analysis is strictly performed on sandbox machines as they are out of our network.

No.	Name	Access Level
1	Gautam Kumar	Admin Access
2		Admin Access

## 4 Steps to access sandbox machine

Connect to VPN of [vpn-india.netradyne.com](https://vpn-india.netradyne.com).

Once connected to VPN please access terminal server 172.16.20.126 with your credentials.

Through terminal server please access sandbox machine 10.11.13.100 with your credentials.

## 5 Malware Analysis Important Terms

Malware analysis is the process of understanding the behavior and purpose of a suspicious file or URL. The output of the analysis aids in the detection and mitigation of the potential threat. Before you get into what malware analysis is, you first need to understand what malware is. Malware or malicious software is a program or code that is designed to enable unauthorized access to a computer or network. Malware can come in many forms.

### 5.1 Types of Malwares

- **Viruses:** A self-replicating malicious program that attaches itself to legitimate files. In some cases, viruses do not damage the victim's computer, but is solely created for humor or to disrupt productivity.
- **Ransomware:** Malicious software created to encrypt the victim's data, preventing the victim from being able to access their files in exchange for money.

- **Rootkit:** A malicious software that allows threat actors to remotely connect to a compromised computer and control it without the victim's knowledge.
- **Trojan:** Malware that is hidden inside seemingly legitimate software and can cause damage to the victim's computer and/or steal data.
- **Worm:** A type of malware that is designed to self-replicate and spread to other computers consuming most of or all of the bandwidth on a victim's network.
- **Spyware:** A form of malware that covertly runs in the background of a victim's device collecting their data.

## 5.2 Types of Malware Analysis

- **Statical or Static property analysis:** This method involves analyzing the binary without executing it. Static analysis allows the analyst to extract metadata from the binary. This technique reveals limited but useful information about the binary that will aid in further analysis. Static analysis also involves analyzing the malicious code itself. The analyst will disassemble and/or debug the suspect binary and investigate the code to get a better understanding of the program's functionality.
- **Dynamic or fully automated malware analysis:** Dynamic or behavioral analysis involves executing the binary to get a visual understanding of how the binary functions. It is highly recommended that you perform dynamic analysis in an isolated environment. This technique is useful but does not uncover all the functionalities of the malicious binary.
- **Manual Code reversing:** In this type involves reverse engineering of the malicious file. this process is made especially difficult because you must circumvent everything the malware does to prevent you from understanding it. code packing, code obfuscation, anti-debugging techniques. Reverse engineering is time consuming and complex. but it is the only way to have complete insight into piece of malware.

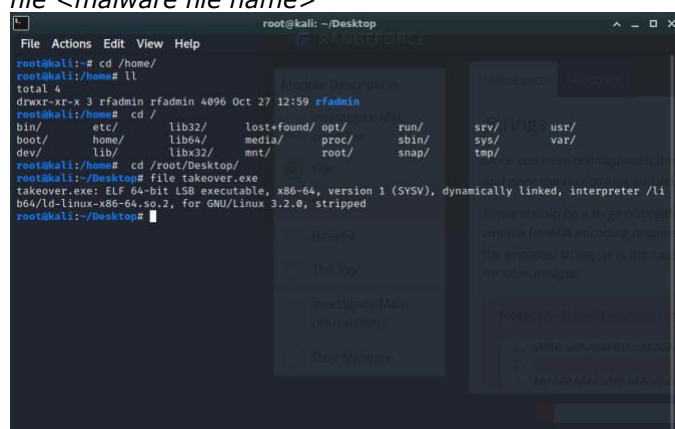
## 6 Basic tools and steps for malware analysis

### 6.1 using staticl approach

In this scenario we have takeover.exe malware file and we will be performing analysis on this file.

1) Investigate malware file type: Determining if the binary has a file type of portable executable (PE) will help determine which operating system the malware is targeting. It is not recommended to rely on file extensions alone. We will be using file command to find file type.

*file <malware file name>*



```

root@kali: ~/Desktop
File Actions Edit View Help
root@kali:~# cd /home/
root@kali:~# ll
total 4
-rwxr-xr-x 3 rfadmin rfadmin 4096 Oct 27 12:59 rfadmin
root@kali:~# cd /
bin/      etc/      lib32/    lost+found/  opt/      run/      srv/      usr/
boot/     home/     lib64/    media/       proc/     sbin/     sys/      var/
dev/      lib/      libx32/   mnt/         root/     snap/     tmp/
root@kali:~# cd /root/Desktop/
root@kali:~/Desktop# file takeover.exe
takeover.exe: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, stripped
root@kali:~/Desktop#

```

Can conclude An ELF 64-bit LSB executable and linux executable.

## 2) Strings

Once you have distinguished the file type, you can go ahead and use strings to find and print the text strings embedded in the file.

There should be a large noticeable Base64 string embedded in the file. Sometimes, when a Base64 encoding requires padding, it adds one or two = characters to the end of the encoded string, as is the case in this file. Copy that whole string and save it into a file for later analysis.

**Note:** The Base64-encoded string's first and last lines look like this:

```
1. aHNxcwcwAAAAAP6YhcAAACAAAAAAAEABEA0AIBAAQAAADgAAAAAAAAAK0FAAAAAAApQUAAAAAAD/
2. ...
3. AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

Often, SHA-256 or other hashing algorithms are used to create a unique signature for a malware file. The sha256sum utility can be used for that.

command to pull strings: strings <file>

Command to generate hash: sha256sum <file>

3) Base64: Now that you have created a new file with the Base64 encoding saved into it, it is time to decode it and see what it entails. The base64 utility is handy for these kinds of situations.

command: base64 -d <file>

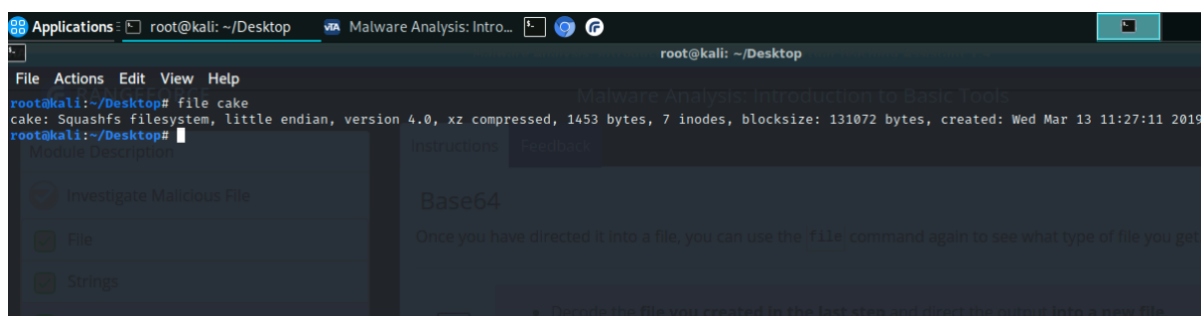
If you just try to decode it, you get a bunch of gibberish. This could indicate that you are dealing with binary. Try directing the output into a file.

command: base64 -d <file> > <new file>

Once you have directed it into a file, you can use the file command again to see what type of file you get.



```
root@kali: ~/Desktop
File Actions Edit View Help
root@kali:~/Desktop# base64 -d check.txt
hsqs...
...
VZname: takeover
version: '0.1'
summary: Do something really nasty with the target server
description: 'See https://github.com/nasty.git'
architectures:
- amd64
confinement: devmode
grade: devel
...
root@kali:~/Desktop# base64 -d check.txt > cake
root@kali:~/Desktop#
```



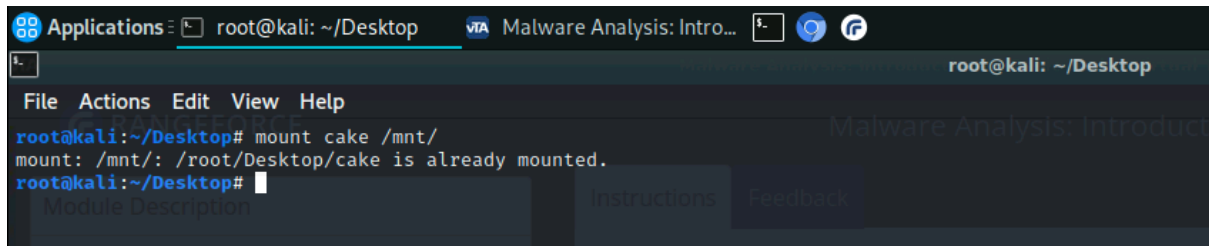
```
Applications: root@kali: ~/Desktop Malware Analysis: Intro...
root@kali:~/Desktop# file cake
cake: Squashfs filesystem, little endian, version 4.0, xz compressed, 1453 bytes, 7 inodes, blocksize: 131072 bytes, created: Wed Mar 13 11:27:11 2019
root@kali:~/Desktop#
```

4) The Key: It seems that there is a compressed read-only file system encased in the file. You can

easily mount it to the OS file system using mount. The /mnt directory is usually used as a mountpoint.

mount <device> <mountpoint> # The device can also be a file

Now you have a bunch of new files to investigate. Try finding the key from the file content.

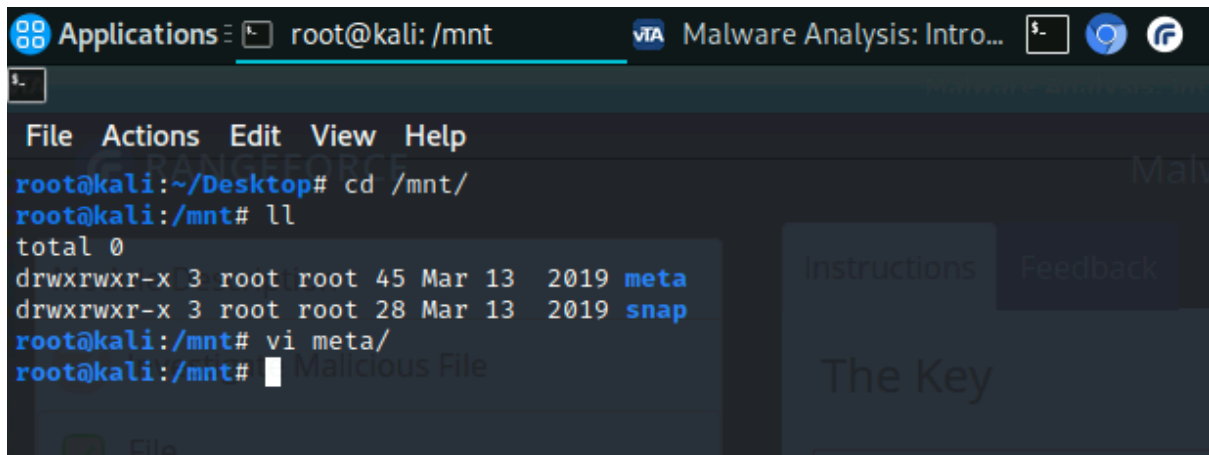


```

Applications root@kali: ~/Desktop Malware Analysis: Intro...
root@kali: ~/Desktop
File Actions Edit View Help
root@kali:~/Desktop# mount cake /mnt/
mount: /mnt/: /root/Desktop/cake is already mounted.
root@kali:~/Desktop#

```

now go to /mnt the open meta.

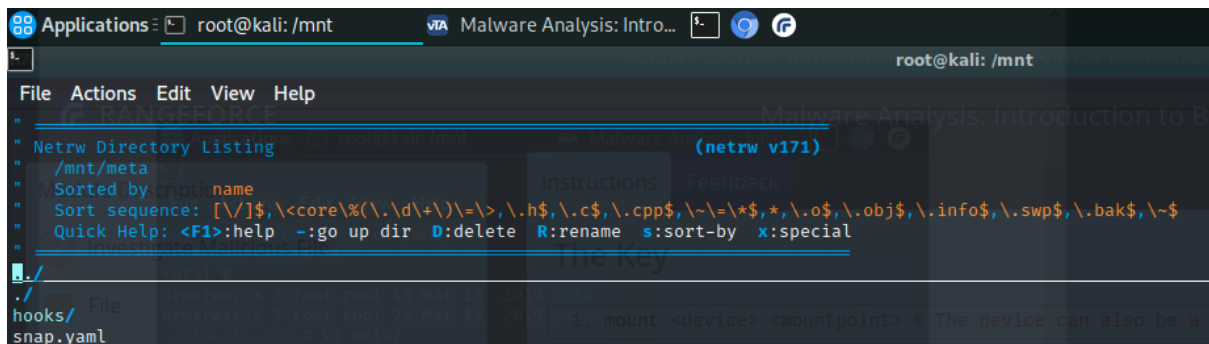


```

Applications root@kali: /mnt Malware Analysis: Intro...
root@kali: /mnt
File Actions Edit View Help
root@kali:~/Desktop# cd /mnt/
root@kali:/mnt# ll
total 0
drwxrwxr-x 3 root root 45 Mar 13 2019 meta
drwxrwxr-x 3 root root 28 Mar 13 2019 snap
root@kali:/mnt# vi meta/
root@kali:/mnt#

```

You can investigate in both files hooks and snap.yml. I found key in hooks / install\*.



```

Applications root@kali: /mnt Malware Analysis: Intro...
root@kali: /mnt
File Actions Edit View Help
* Netrw Directory Listing (netrw v171)
* /mnt/meta
* Sorted by: dname
* Sort sequence: [/]/$, \<core\%(\\.d\\+\\)=\\>\\.h$,\\.c$,\\.cpp$,\\-\\=\\*$,*,\\.o$,\\.obj$,\\.info$,\\.swp$,\\.bak$,\\-\\$
* Quick Help: <F1>:help --go up dir D:delete R:rename S:sort-by X:special
./
./
hooks/ File
snap.yml

```

```

root@kali: /mnt
File Actions Edit View Help
! /bin/bash

KEY=${1:'TewvyidnexictOsvilabquojTeirOfIcDotfогrydWacgeyThyrodmyntAnCues'}
fileExts=(
    "*.py" "*.txt" "*.cpp" "*.png" "*.jpg" "*.sh" "*.pyc" \
    "*.key" "*.php" "*.css" "*.js" "*.tiff" "*.tff" "*.pl" \
    "*.ini" "*.xml" "*.gpg" "*.enc" "*.lst" \
    "*.propertis" "*.acl" "*.gz" "*.tar" "*.bz2" "*.gif" \
    "*.doc*" "*.xls*" "*.pdf" "*.java" "*.swf" "*.jar" \
    "*.json" "*.ppt*" "*.pst" "*.bat" "*.exe" "*.x" "*.pm" \
    "*.aps*" "*.cgi" "*.htm*" "*.dll" "*.class" "*.mov" \
    "*.flv" "*.mp4" "*.mp3" "*.wav" "*.mov" "*.ogg" "*.md" \
    "*.yaml" "*.pem" "*.gpg" "*.sql" "*.vim" "*.csv" "*.bak" \
    "*.rb" "*.h" "*.c" "*.log" "*.pdf" "*.log.*")

#fileList=("/root/Documents/")

```

The ps -ef command with its parameters is handy for finding out which processes are running on your workstation.



```

root@kali: /mnt/meta/hooks# ss -antp
State      Recv-Q    Send-Q    Local Address:Port    Peer Address:Port    Process
LISTEN     0          4096      0.0.0.0:5355          0.0.0.0:*             users(("systemd-resolve",pid=304,fd=12))
LISTEN     0          4096      127.0.0.53:53         0.0.0.0:*             users(("systemd-resolve",pid=304,fd=19))
LISTEN     0          128       0.0.0.0:22            0.0.0.0:*             users(("sshd",pid=437,fd=3))
ESTAB      0          0         192.168.6.1:56718     35.238.100.217:443    users(("chromium",pid=928,fd=219))
ESTAB      0          0         192.168.6.1:44048     10.216.56.164:2297    users(("koox7eeghi0eagh",pid=587,fd=5))
LISTEN     0          4096      [::]:5355            [::]:*                users(("systemd-resolve",pid=304,fd=14))
LISTEN     0          128       [::]:22              [::]:*                users(("sshd",pid=437,fd=4))
LISTEN     0          2         [::]:13350           [::]:*                users(("xrdp-sesman",pid=417,fd=7))
LISTEN     0          2         [::]:3389            [::]:*                users(("xrdp",pid=447,fd=11))
ESTAB      0          0         [::ffff:192.168.6.1]:3389 [::ffff:192.168.6.254]:42060 users(("xrdp",pid=588,fd=12))

root@kali: /mnt/meta/hooks# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address     State       PID/Program name
tcp        0      0 0.0.0.0:5355       0.0.0.0:*          LISTEN      304/systemd-resolve
tcp        0      0 127.0.0.53:53      0.0.0.0:*          LISTEN      304/systemd-resolve
tcp        0      0 0.0.0.0:22         0.0.0.0:*          LISTEN      437/sshd: /usr/sbin
tcp        0      0 192.168.6.1:56718 35.238.100.217:443 ESTABLISHED 928/chromium --type
tcp        0      0 192.168.6.1:44048 10.216.56.164:2297 ESTABLISHED 587/koox7eeghi0eagh
tcp6       0      0 [::]:5355         [::]:*             LISTEN      304/systemd-resolve
tcp6       0      0 [::]:22           [::]:*             LISTEN      437/sshd: /usr/sbin
tcp6       0      0 [::]:13350        [::]:*             LISTEN      417/xrdp-sesman
tcp6       0      0 [::]:3389         [::]:*             LISTEN      447/xrdp
tcp6       0      0 192.168.6.1:3389 192.168.6.254:42060 ESTABLISHED 588/xrdp

```

Now you can find for which Ip malware is trying to connect then port and full path of malware.

```

root@kali: /mnt/meta/hooks# ss -antp
State      Recv-Q    Send-Q    Local Address:Port    Peer Address:Port    Process
LISTEN     0          4096      0.0.0.0:5355          0.0.0.0:*             users(("systemd-resolve",pid=304,fd=12))
LISTEN     0          4096      127.0.0.53:53         0.0.0.0:*             users(("systemd-resolve",pid=304,fd=19))
LISTEN     0          128       0.0.0.0:22            0.0.0.0:*             users(("sshd",pid=437,fd=3))
ESTAB      0          0         192.168.6.1:56718     35.238.100.217:443    users(("chromium",pid=928,fd=219))
ESTAB      0          0         192.168.6.1:44048     10.216.56.164:2297    users(("koox7eeghi0eagh",pid=587,fd=5))
LISTEN     0          4096      [::]:5355            [::]:*                users(("systemd-resolve",pid=304,fd=14))
LISTEN     0          128       [::]:22              [::]:*                users(("sshd",pid=437,fd=4))
LISTEN     0          2         [::]:13350           [::]:*                users(("xrdp-sesman",pid=417,fd=7))
LISTEN     0          2         [::]:3389            [::]:*                users(("xrdp",pid=447,fd=11))
ESTAB      0          0         [::ffff:192.168.6.1]:3389 [::ffff:192.168.6.254]:42060 users(("xrdp",pid=588,fd=12))

root@kali: /mnt/meta/hooks# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address     State       PID/Program name
tcp        0      0 0.0.0.0:5355       0.0.0.0:*          LISTEN      304/systemd-resolve
tcp        0      0 127.0.0.53:53      0.0.0.0:*          LISTEN      304/systemd-resolve
tcp        0      0 0.0.0.0:22         0.0.0.0:*          LISTEN      437/sshd: /usr/sbin
tcp        0      0 35 192.168.6.1:56718 35.238.100.217:443 ESTABLISHED 928/chromium --type
tcp        0      0 10.216.56.164:2297 ESTABLISHED 587/koox7eeghi0eagh
tcp6       0      0 [::]:5355         [::]:*             LISTEN      304/systemd-resolve
tcp6       0      0 [::]:22           [::]:*             LISTEN      437/sshd: /usr/sbin
tcp6       0      0 [::]:13350        [::]:*             LISTEN      417/xrdp-sesman
tcp6       0      0 [::]:3389         [::]:*             LISTEN      447/xrdp
tcp6       0      0 192.168.6.1:3389 192.168.6.254:42060 ESTABLISHED 588/xrdp

root@kali: /mnt/meta/hooks# ps -ef | grep 587
root    587      1  0 00:57 ?        00:00:00 /var/jalaWa/aHo6gi/ga40ng/koox7eeghi0eagh5eiNg
root    15136  1138  0 10:53 pts/0  00:00:00 grep 587

```

6) Stop Malware: Now that you have all the information you need about the malware, it's time to stop its activities. Just killing the process is not always enough. Try going for the source itself.

Command

# To remove a file

rm <file>

# To kill a process

kill -9 <pid>

```

root@kali: /mnt/meta/hooks# rm -rf /var/jalaWa/aHo6gi/ga40ng/koox7eeghi0eagh5eiNg
root@kali: /mnt/meta/hooks# kill -9 587
root@kali: /mnt/meta/hooks#

```

## 6.2 using Dynamic approach

Dynamic or behavioral analysis involves executing a malware sample in a virtual environment and analyzing how it affects the target system. There are multiple tasks that need to be done during analysis.

### Task/Tools

- Simulating internet services: Using Flare-Fakenet-NG or INetSim to provide the malware a simulation of the services it requires to operate.

```
=== INetSim main process started (PID 21034) ===
Session ID:      21034
Listening on:    127.0.0.1
Real Date/Time:  2021-01-26 19:29:57
Fake Date/Time:  2021-01-26 19:29:57 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 21036)
* finger_79_tcp  - started (PID 21048)
* daytime_13_tcp - started (PID 21053)
* ntp_123_udp    - started (PID 21047)
* syslog_514_udp - started (PID 21050)
* time_37_udp    - started (PID 21052)
* daytime_13_udp - started (PID 21054)
* quotd_17_tcp   - started (PID 21059)
* quotd_17_udp   - started (PID 21060)
* chargen_19_tcp - started (PID 21061)
* chargen_19_udp - started (PID 21062)
* irc_6667_tcp   - started (PID 21046)
* dummy_1_udp    - started (PID 21064)
* discard_9_tcp  - started (PID 21057)
* discard_9_udp  - started (PID 21058)
* echo_7_tcp     - started (PID 21055)
* dummy_1_tcp    - started (PID 21063)
* echo_7_udp     - started (PID 21056)
* tftp_69_udp    - started (PID 21045)
* http_80_tcp    - started (PID 21037)
* pop3s_995_tcp  - started (PID 21042)
* ftps_990_tcp   - started (PID 21044)
* https_443_tcp  - started (PID 21038)
* ftp_21_tcp     - started (PID 21043)
* pop3_110_tcp   - started (PID 21041)
* ident_113_tcp  - started (PID 21049)
* time_37_tcp    - started (PID 21051)
* smtps_465_tcp  - started (PID 21040)
* smtp_25_tcp    - started (PID 21039)
done.
Simulation running.
```

Monitoring the malware to understand how it behaves and its effects on the system is important during dynamic analysis. There are multiple forms of monitoring malware activity, which include network monitoring, process monitoring, registry monitoring, and file system monitoring.

- **Network monitoring:** Monitoring live network traffic to and from the system during malware execution. (Wireshark, Flare-Fakenet-NG, INetSim)

In this below image we see trojan connecting to C2 server.

3	8.994398	192.168.1.50	192.168.1.100	DNS	75 Standard query 0x7cee A miledaughter.ru
4	8.994578	192.168.1.100	192.168.1.50	ICMP	103 Destination unreachable (Port unreachable)
5	8.994626	192.168.1.50	192.168.1.100	DNS	75 Standard query 0x7cee A miledaughter.ru
6	8.994756	192.168.1.100	192.168.1.50	ICMP	103 Destination unreachable (Port unreachable)
7	8.994786	192.168.1.50	192.168.1.100	DNS	75 Standard query 0x7cee A miledaughter.ru
8	8.994930	192.168.1.100	192.168.1.50	ICMP	103 Destination unreachable (Port unreachable)
9	8.994966	192.168.1.50	192.168.1.100	DNS	75 Standard query 0x7cee A miledaughter.ru
10	8.995092	192.168.1.100	192.168.1.50	ICMP	103 Destination unreachable (Port unreachable)
11	8.995125	192.168.1.50	192.168.1.100	DNS	75 Standard query 0x7cee A miledaughter.ru
12	8.995251	192.168.1.100	192.168.1.50	ICMP	103 Destination unreachable (Port unreachable)

In below image trojan is attempting to connect suspicious urls

262	212.752177512	192.168.1.50	192.168.1.100	DNS	77 Standard query 0x4c33 A obsp.comodoca.com
263	212.752224345	192.168.1.100	192.168.1.50	ICMP	105 Destination unreachable (Port unreachable)
264	212.762898418	192.168.1.50	192.168.1.100	DNS	76 Standard query 0x9461 A crl.comodoca.com
265	212.762925544	192.168.1.100	192.168.1.50	ICMP	104 Destination unreachable (Port unreachable)
266	212.763192210	192.168.1.50	192.168.1.100	DNS	76 Standard query 0x9461 A crl.comodoca.com
267	212.763291527	192.168.1.100	192.168.1.50	ICMP	104 Destination unreachable (Port unreachable)
268	212.763480595	192.168.1.50	192.168.1.100	DNS	76 Standard query 0x9461 A crl.comodoca.com
269	212.763490251	192.168.1.100	192.168.1.50	ICMP	104 Destination unreachable (Port unreachable)
270	212.763734445	192.168.1.50	192.168.1.100	DNS	76 Standard query 0x9461 A crl.comodoca.com

Upon further investigation of the C2 server (miledaughter.ru) and the suspicious URLs (ocsp.comodoca.com, crt.comodoca.com) a few sources (howtoremove.guide, how-to-remove.com) claim that this trojan happens to be a browser hijacker that is delivered upon the installation of seemingly legitimate software. Although when visiting the comodoca.com, the site offers SSL certificates. As for the C2 server, after running a whois query it shows that the domain registrar is 101DOMAIN-RU and it was created May 3, 2020. ThreatCrowd also displays a graph showing the different variants of this malware and their origins.

- **Process Monitoring:** Examining the malware process activity during malware execution. (Process Hacker, Process Monitor, Noriben)
- **Registry monitoring:** Monitors whether registry keys have been modified/accessed and what registry data has been read/written to. (Noriben, Process Monitor)
- **File system monitoring:** Monitoring the file system activity in real-time during malware execution. (Noriben, Process Monitor)

Below is Noriben's report after executing the malware. As you can see, there are multiple registry changes, but the most intriguing registry change is "[RegSetValue]"

```
Explorer.EXE:5436 >
```

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\Reader

RevisionInfo\6540C835-B904-5629-60B1-FAD36E0062D1 = 1 7 竿+脇獅地鴉鑲

獮習据鑣<揔勛 ㄊㄨˋ ㄏㄨㄢˋ 敵据>揔烈 ㄊㄨˋ ㄌㄧㄝˋ ㄓㄨˋ 捡梯窰械璵牯鑣<揔 ㄊㄨˋ 釁 ㄌㄞˊ 兪 ㄈㄨˊ 筭 ㄓㄨˋ ㄒ

垢莖圻箝㊦秋<sup>ㄑㄩ</sup>藎<sup>ㄋㄧˋ</sup>～(助)以<sup>ㄇㄣˊ</sup>口<sup>ㄎㄨ</sup>咄<sup>ㄉㄨ</sup>汨<sup>ㄇㄣˊ</sup>整<sup>ㄓㄥ</sup>摺<sup>ㄓ</sup>笠<sup>ㄌㄧˋ</sup>††𡵚<sup>ㄙ</sup>傲<sup>ㄜ</sup>擗<sup>ㄆㄛ</sup>樓<sup>ㄌㄠ</sup>塗<sup>ㄊㄨ</sup>舫<sup>ㄈㄤ</sup>•†††𡵚<sup>ㄙ</sup>舳<sup>ㄔㄨ</sup>杓<sup>ㄅㄞ</sup>湏<sup>ㄒㄨ</sup>樓<sup>ㄌㄠ</sup>塗<sup>ㄊㄨ</sup>舫

[illegible]

Running the registry value in Google Translate reveals that the text is Chinese. The program was able to translate a portion of the text to renewing the enemy's data, picking up the stubborn data, and production and treatment. There were no interesting data entries logged for under the Network Traffic and Unique Host tabs since the malware was connecting to a simulated network. You can view the full list of changed registries [here](#).

```

1. -=] Sandbox Analysis Report generated by Noriben v1.8.4
2. -=] Developed by Brian Baskin: brian @@ thebaskins.com @bbaskin
3. -=] The latest release can be found at https://github.com/Rurik/Noriben
4.
5. -=] Execution time: 47.90 seconds
6. -=] Processing time: 0.48 seconds
7. -=] Analysis time: 0.69 seconds
8.
9. Processes Created:
10. =====
11. [CreateProcess] Explorer.EXE:5436 > "%UserProfile%\Desktop\salford1\salford1.exe "
    [Child PID: 2632]
12. [CreateProcess] svchost.exe:812 > "% WinDir%\SysWOW64\DllHost.exe
    /Processid:{776DBC8D-7347-478C-8D71-791E12EF49D8}" [Child PID: 2496]
13.
14. File Activity:
15. =====
16. [CreateFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache [File no longer
    exists]
17. [CreateFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache [File no longer
    exists]
18. [RenameFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-S-1-5-
    18.dat =>
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-S-1-
    5-18.dat
19. [RenameFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-
    FontSet-S-1-5-18.dat =>
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-
    FontSet-S-1-5-18.dat
20. [CreateFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-
    FontSet-S-1-5-18.dat [File no longer exists]
21. [CreateFile] svchost.exe:1648 >
    % WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-S-1-
    5-18.dat [File no longer exists]
22.
23. Registry Activity:
24. =====
25. [RegSetValue] System:4 >
    HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-
    4279892692-2277359461-354179757-1001\SequenceNumber = 44
26. [RegSetValue] Explorer.EXE:5436 >
    HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderR
    evisionInfo\6540C835-B904-5629-60B1-FAD36E0062D1 = 1 ▯ 𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈𐮉𐮊𐮋𐮌𐮍𐮎𐮏𐮐𐮑𐮒𐮓𐮔𐮕𐮖𐮗𐮘𐮙𐮚𐮛𐮜𐮝𐮞𐮟𐮠𐮡𐮢𐮣𐮤𐮥𐮦𐮧𐮨𐮩𐮪𐮫𐮬𐮭𐮮𐮯𐮰𐮱𐮲𐮳𐮴𐮵𐮶𐮷𐮸𐮹𐮺𐮻𐮼𐮽𐮾𐮿𐯀𐯁𐯂𐯃𐯄𐯅𐯆𐯇𐯈𐯉𐯊𐯋𐯌𐯍𐯎𐯏𐯐𐯑𐯒𐯓𐯔𐯕𐯖𐯗𐯘𐯙𐯚𐯛𐯜𐯝𐯞𐯟𐯠𐯡𐯢𐯣𐯤𐯥𐯦𐯧𐯨𐯩𐯪𐯫𐯬𐯭𐯮𐯯𐯰𐯱𐯲𐯳𐯴𐯵𐯶𐯷𐯸𐯹𐯺𐯻𐯼𐯽𐯾𐯿𐰀𐰁𐰂𐰃𐰄𐰅𐰆𐰇𐰈𐰉𐰊𐰋𐰌𐰍𐰎𐰏𐰐𐰑𐰒𐰓𐰔𐰕𐰖𐰗𐰘𐰙𐰚𐰛𐰜𐰝𐰞𐰟𐰠𐰡𐰢𐰣𐰤𐰥𐰦𐰧𐰨𐰩𐰪𐰫𐰬𐰭𐰮𐰯𐰰𐰱
```



<p>坳箝Θ秋ꠞꠞ 捷~(財)𐄂Q†咄汨整≤撰笠††∠獾倣概櫻塗舂••††∠牯杓湮櫻塗舂欲𐄂撰          𐄂𐄂†††糊惛整櫻塗舂欲𐄂撰𐄂𐄂†††產𐄂稽瑣湮卮惛整櫻塗舂••†素素;</p>
<p>27. [RegSetValue] Explorer.EXE:5436 &gt;          HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderR          evisionInfo\6540C835-B904-5629-60B1-FAD36E0062D1 = 1 ▯ 𐄂†𐄂𐄂地獺鑣獺          惛据鑣≤撰勳𐄂††𐄂煥敵据≥撰烈𐄂𐄂†∠捡梯窠穢琰牯鑣≤撰∠𐄂 /•𐄂𐄂𐄂𐄂𐄂𐄂𐄂𐄂𐄂𐄂𐄂          坳箝Θ秋ꠞꠞ 捷~(財)𐄂Q†咄汨整≤撰笠††∠獾倣概櫻塗舂••††∠牯杓湮櫻塗舂欲𐄂撰          𐄂𐄂†††糊惛整櫻塗舂欲𐄂撰𐄂𐄂†††產𐄂稽瑣湮卮惛整櫻塗舂••†素素;</p>
<p>28. [RegSetValue] svchost.exe:812 &gt;          HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-          4279892692-2277359461-354179757-          1001\Device\HarddiskVolume2\Windows\System32\dlhhost.exe = D1 18 56 1F F4 ED          D6 01 00 00 00 00 00 00 00 00</p>
<p>29. [RegSetValue] ctfmon.exe:5312 &gt;          HKCU\SOFTWARE\Microsoft\Input\TypingInsights\Insights = 02 00 00 00 07 1D E8          C1 31 CC 83 60 A3 D6 D9 C1</p>
<p>30. ...</p>
<p>31.</p>
<p>32.</p>
<p>33. Network Traffic:</p>
<p>34. =====</p>
<p>35. [UDP] svchost.exe:2388 &gt; 192.168.1.100:53</p>
<p>36. [UDP] svchost.exe:2388 &gt; 169.254.38.149:53</p>
<p>37. [UDP] svchost.exe:2388 &gt; ff02:0:0:0:0:1:3:5355</p>
<p>38. [UDP] System:4 &gt; 192.168.1.100:137</p>
<p>39. [UDP] System:4 &gt; 169.254.38.149:137</p>
<p>40. [UDP] 192.168.1.100:53780 &gt; svchost.exe:7048</p>
<p>41.</p>
<p>42. Unique Hosts:</p>
<p>43. =====</p>
<p>44. 169.254.38.149</p>
<p>45. 192.168.1.100</p>
<p>46. ff02</p>

## 7 Malware Sandboxing using Hybrid Analysis

Malware sandboxing is a technique utilized by cybersecurity professionals to analyze malicious and potentially dangerous software. The idea behind it is rather simple — create a secure "walled" environment (a virtual machine, which in this context is called a sandbox) and let the malware do whatever it was supposed to do. This way defenders can collect indicators of compromise (or IOCs for short) without the risk of infecting their systems.

here are many great automated sandbox solutions that allow you to submit your malware samples for analysis online. With the help of tools like Hybrid Analysis, Cuckoo and AnyRun, you can submit malicious files and URLs for analysis for free and get results within minutes.

## 7.1 Submitting Malware for Analysis

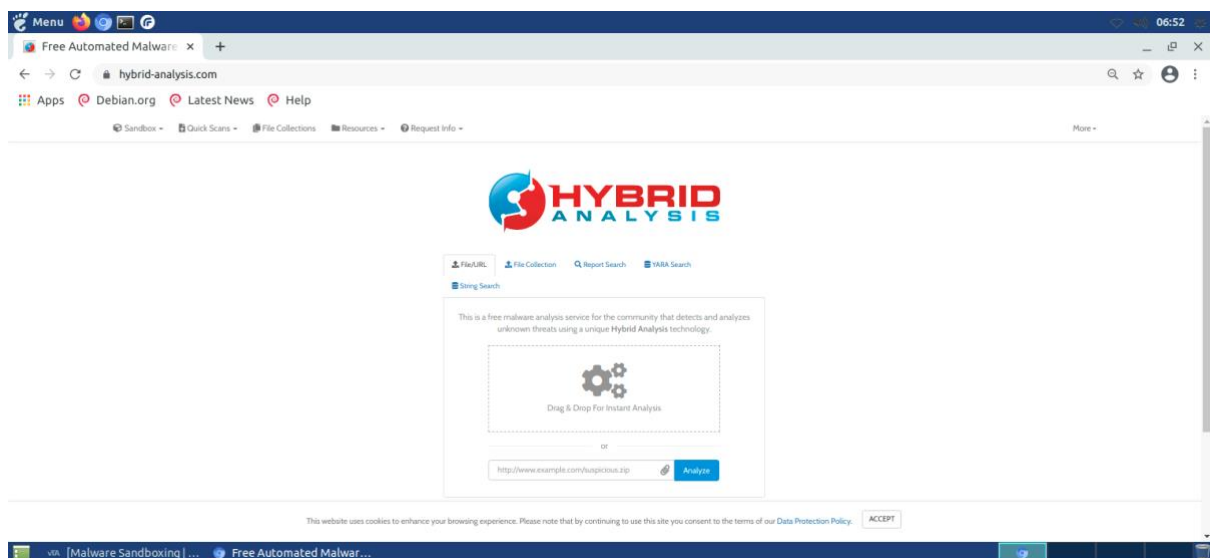
There are many sandbox solutions, both commercial and free. We will be using hybrid analysis for sandboxing. Hybrid Analysis allows you to analyze both files and URLs for malicious activity. It cross-references dozens of security intelligence providers to gather information on potential IOCs and related flavors of malware.

Steps for submitting malware

- Open Hybrid-analysis.com in your browser.
- Drag and drop malware sample
- Complete captcha and select analysis environment on which you want to test.
- Click to generate public report.

After submitting a file for analysis, the tool automatically redirects you to the Analysis Overview page. This is where you can find all the base information regarding the file, including:

*file type,*  
*SHA256 hash,*  
*antivirus detection score,*  
*general assessment of whether the file is malicious or not.*



### Analysis Environments

Name: x64\_install.bin

Size: 4.3MiB

Type: pease executable

MIME: application/x-dosexec

SHA256: 9f4cb65bd6f8a6...582b4774f6c84

Available:


- ☒ Windows 7 32 bit
- ☐ Windows 7 32 bit (HWP Support)
- ☐ Windows 7 64 bit
- ☐ Linux (Ubuntu 16.04, 64 bit)
- ☐ Android Static Analysis
- ☐ Quick Scan

There are no files in the processing queue.

Currently, the average processing time per sample is 8 minutes and 5 seconds.

<< Back
Runtime Options
Generate Public Report

Results will appear as follows.


Sandbox
Quick Scans
File Collections
Resources
Request Info

### Analysis Overview

Submission name: setup\_x86\_x64\_install.bin

Size: 4.3MiB

Type: pease executable

MIME: application/x-dosexec

SHA256: 9f4cb65bd6f8a6...582b4774f6c84

Operating System: Windows

Last Anti-Virus Scan: 12/01/2021 08:54:12 (UTC)

Last Sandbox Report: 12/06/2021 21:17:39 (UTC)

**malicious**

Threat Score: 100/100

AV Detection: 65%

Labeled as: jak.Generic

[Link](#) [Twitter](#) [Facebook](#) [Email](#)

#### Analysis Overview

- Anti-Virus Scanner Results
- Related Hashes
- Falcon Sandbox Reports (2)
- Incident Response
- Community (0)

[Back to top](#)

### Anti-Virus Results

**CrowdStrike Falcon**

100%

Static Analysis and ML

Last Update: 12/01/2021 08:54:12 (UTC)

[View Details](#)

[Visit Vendor](#)

[GET STARTED WITH A FREE TRIAL](#)

**MetaDefender**

25%

Multi Scan Analysis

Last Update: 12/01/2021 08:54:12 (UTC)

[View Details](#)

[Visit Vendor](#)

**VirusTotal**

71%

Multi Scan Analysis

Last Update: 12/01/2021 08:54:12 (UTC)

[View Details](#)

[Visit Vendor](#)

## Related Hashes

Invid  
Com  
Back

### Files extracted during detonation

Name	Sha256	Verdict
file	ea8cce26f5348e13395c7b4a713b28a7801c1a27b67bb860b82063c4276ald	malicious
file	b5b2c802acac154a31c2ad67b0d971d481db8887a939173b54ec2a933792daa9	malicious
file	339e4d129303e36ef518c9ef738119007a8b01b3cb3b15f1cee56495c4e40747	malicious
file	d5959e437e58709c5e5e7a923efe7351b28bedef15cb00cd9fdb4e5e955b2a13	malicious
file	83664d9745f175b770b960a253e5efc0ff4ee06b72083fa8be2bbf801328d3e	malicious
SunO6964bbec839856df.exe	82c816980fe9b0de916fc1954a2ekdb51011770179418f15a2e84656962eb67	malicious
file	5da618d0d5419251a1735057b27f9a5188e2ddd44153ce35ce69caaf678f26a8	malicious
file	93a17aa366e45b5d4c87a6273cd20a6657a729516831f486ca9478dff44f83b	malicious
file	d838cfaf7b197d6c3379e2c5daf269cc422a09df556de6ca08fe174b4906b3b6	malicious
file	282eb8e7745f9396a2551817e90afbdfe54a77c427c3050d0cc638fb2f50dc3	malicious
file	6e14451c400fc83136bbe8d08d404b038aebd2a7dff1a18c145581b8cc0d78cc0d	malicious
file	36cc42294d2eac9e45fa38919a7ald1f8cb5af6f68ed2d5e9563bd522f48bc4a	malicious
file	d5cceb40a76ec2c82cac45cc208a778269e743f1a825ef881533b85d6cd31	malicious
Udp_1_.exe	2975e8edd52d5266521d56cdd2e8fe293c341d6a082321c0e4974a8678cf6c6b	malicious
file	af620f48d534f6db07e31fa18182cbf78bf4b9c9128657a779094cdd8fe4a25	malicious
_shfoldr.dll	9884e9dfb4f8a873ccbd81f8ad0ae25776d2348d027d811a56475e028360d87	whitelisted
foradvertisingweb_1_.exe	342038018ad62031807f31a0cfaef169a4db81d297b1fd4b8e4c963f67ea50	malicious

## Falcon Sandbox Reports

### MALICIOUS

setup\_x86\_x64\_install.bin

Analyzed on: 11/23/2021 14:52:11 (UTC)

Environment: Windows 7 64 bit

Threat Score: 100/100

AV Detection: 71% Trojan.Fabookie

Indicators: 11 0 0

Network:

### MALICIOUS

x64\_install.bin

Analyzed on: 12/06/2021 21:17:39 (UTC)

Environment: Windows 7 32 bit

Threat Score: 100/100

AV Detection: 71% Trojan.Fabookie

Indicators: 22 0 0

Network:

### FALCON SANDBOX TECHNOLOGY

**Strong Hybrid Analysis: Powered by Falcon Sandbox**

Upgrade to a Falcon Sandbox license and gain full access to all features, IOCs and behavioral analysis.

**Easily Deploy and Scale**

Process up to 25,000 files per month with Falcon Sandbox Private Cloud or select an unlimited license with the On-Prem Edition.

**Extensive Coverage**

Expanded support for file types, operating systems and export file formats.

**Unparalleled Customization**

Import custom virtual machine images and customize settings to mirror your real-world environment.

[Learn more](#)



## Incident Response

**Risk Assessment**

<b>Remote Access</b>	Reads terminal service related keys (often RDP related)
<b>Spyware</b>	Contains ability to open the clipboard POSTs files to a webserver
<b>Persistence</b>	Modifies System Certificates Settings Spawns a lot of processes Writes data to a remote process
<b>Fingerprint</b>	Queries kernel debugger information Queries process information Queries sensitive IE security settings Queries the display settings of system associated file extensions Queries the internet cache settings (often used to hide footprints in index.dat or internet cache) Reads the active computer name Reads the windows installation language Tries to identify its external IP address
<b>Evasive</b>	Marks file for deletion Possibly tries to evade analysis by sleeping many times Possibly tries to implement anti-virtualization techniques Tries to sleep for a long time (more than two minutes)
<b>Network Behavior</b>	Contacts 31 domains and 30 hosts. <a href="#">View all details</a>

**MITRE ATT&CK™ Techniques Detection**


We found MITRE ATT&CK™ data in 2 reports, on average each report has 53 mapped indicators. [View all details](#)

Falcon 5  
Incident  
Commu  
Back to t

## 7.2 Examining the result

Once the sandbox analysis results appear in the Falcon Sandbox Reports section (located right below the antivirus section of the page), you will see a generated report, like so:

MALICIOUS

 **setup\_x86\_x64\_install.bin**


**Analyzed on:** 11/23/2021 14:52:11 (UTC)

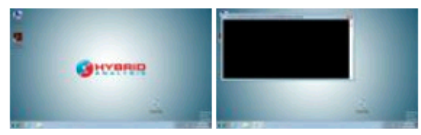
**Environment:** Windows 7 64 bit

**Threat Score:** 100/100

**AV Detection:** 71% Trojan.Fabookie

**Indicators:** 17 46 33

**Network:** 



## 1) Incident Response

The first section of the report is titled Incident Response and consists of two parts:

- Risk Assessment that gives you the summary of what the malware is doing on the infected machine;
- MITRE ATT&CK mapping of the techniques used by the malicious file (you can click the View all details button to see the full mapping).

This section alone is enough to give you an idea of the TTPs used by the malware. Further sections of the report, however, are very useful for gathering more specific information, such as pinpointing the attacker's C2 servers and process/file names created and modified by the malware.

## Incident Response

### Risk Assessment

<b>Remote Access</b>	Reads terminal service related keys (often RDP related)
<b>Spyware</b>	Contains ability to open the clipboard
<b>Persistence</b>	Spawns a lot of processes Writes data to a remote process
<b>Fingerprint</b>	Queries kernel debugger information Queries sensitive IE security settings Queries the display settings of system associated file extensions Queries the internet cache settings (often used to hide footprints in index.dat or internet cache) Reads the active computer name Reads the windows installation language Tries to identify its external IP address
<b>Evasive</b>	Marks file for deletion Possibly tries to evade analysis by sleeping many times Possibly tries to implement anti-virtualization techniques Tries to sleep for a long time (more than two minutes)
<b>Network Behavior</b>	Contacts 8 domains and 3 hosts. <a href="#">View all details</a>

### MITRE ATT&CK™ Techniques Detection

This report has 48 indicators that were mapped to 31 attack techniques and 9 tactics. [View all details](#)

## 2) Indicators

In the indicators section of the report, you will find IOCs. This is split into two parts.

Malicious Indicators, Suspicious Indicators and informative.

The Malicious Indicators part includes all the activities that can be considered red flags right away, such as running PowerShell commands and pulling files from remote hosts. Note that all indicators in the list are collapsible and will give you more specific information if you click on them. For instance, the file that is being used as an example in this module has been detected to have an embedded VBA macro, which is represented in the following indicator:

Malicious Indicators <span>17</span>	
<b>External Systems</b>	
Sample was identified as malicious by a large number of Antivirus engines	▼
Sample was identified as malicious by at least one Antivirus engine	▼
<b>General</b>	
The analysis extracted a file that was identified as malicious	▼
The analysis spawned a process that was identified as malicious	▼
<b>Installation/Persistence</b>	
Allocates virtual memory in a remote process	▼
Writes data to a remote process	▼
<b>Network Related</b>	
Malicious artifacts seen in the context of a contacted host	▼
Tries to identify its external IP address	▼

## 3) Suspicious Indicators:

The Suspicious Indicators part contains activities or techniques utilized by the malware that seem questionable, but the analysis tool is less certain regarding their maliciousness. Do keep in mind, however, that some of these techniques can seem quite obviously malicious to a human analyst, so don't just skip through this part of the report. For example, you can find clearly malicious indicators like the following one hidden in the Suspicious Indicators part:

## Anti-Reverse Engineering

Cmdline arguments contain obfuscated URL(s) ^

**details** [http://eddietravelmarigoldcatba.com -> URL extracted from cmd.exe's cmdline arguments \(Show Process\)](#), [http://school3.webhawksittesting.com -> URL extracted from cmd.exe's cmdline arguments \(Show Process\)](#)  
[http://www.wmdcustoms.com -> URL extracted from cmd.exe's cmdline arguments \(Show Process\)](#), [http://www.yogananda-palermo.org -> URL extracted from cmd.exe's cmdline arguments \(Show Process\)](#)  
[http://klanift.cba.pl -> URL extracted from cmd.exe's cmdline arguments \(Show Process\)](#)

**source** Monitored Target

**relevance** 10/10

## 4) Informative Indicators

Informative part of the indicators section contains the list of all detected techniques and activities performed by the malware. This part can be especially extensive, so normally you would just briefly look through it because chances are the tool has caught and already marked all the notoriously bad indicators in the previous part of the indicators list

Informative <span style="float: right;">33</span>	
<b>Anti-Reverse Engineering</b>	
Contains ability to register a top-level exception handler (often used as anti-debugging trick)	▼
PE file contains zero-size sections	▼
<b>Environment Awareness</b>	
Contains ability to enumerate files inside a directory	▼
Contains ability to query local/system time	▼
Contains ability to query machine time	▼
Contains ability to query the machine version	▼
Contains ability to query volume size	▼
Contains ability to read software policies	▼
Makes a code branch decision directly after an API that is environment aware	▼
Queries volume information	▼
Reads the registry for installed applications	▼

## Hybrid Analysis:

The Hybrid Analysis section of the report can seem cryptic at first, but it provides a great insight into the malware's workflow. It presents all the processes ran by the malware in a tree view sorted in chronological order, so you can trace it every step of the way.

Here, the malware sample is trying to download some files from several remote hosts with the help of an obfuscated PowerShell command:

## Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 31 processes in total (System Resource Monitor).

You can also click on every process to investigate it further. Hybrid Analysis records all the API calls, registry, and file manipulations committed by each process spawned by the malware.

## 8 Examining malware file using PE studio (Static Analysis)

### 8.1 Basic Information

When you first open a file in pestudio, you will see a window like this:

Open the mystery.exe file on your desktop using pestudio and look at some of the initially displayed information. To do this, launch pestudio using the shortcut on your desktop and either drag mystery.exe into the window or use the application menu to open it.

## 8.2 Hashes

When examining a potentially malicious file for the first time, it can be handy to start by grabbing the file's hash. You can think of a hash in this context as a fingerprint for the file. When you change even a single bit of data in a file, the hash of the file will be different. The results of three different algorithms displayed by pestudio (MD5, SHA1, SHA256) are effectively interchangeable for this purpose.

It is also possible to calculate a hash only for certain parts of the file, such as the headers or the imported libraries, also known as an imphash. This can be useful when you are trying to compare different files that you suspect are related. When malware authors release new versions, they might leave some parts of the file unchanged. Calculating hashes on specific parts of the file is something that pestudio also does for you.

You can use a file's hash to search online in VirusTotal, Hybrid Analysis, or any other similar databases or sandboxes. Pestudio will automatically search for the hash on VirusTotal, and if there is a match, pestudio will display the results directly.

By querying databases like this, you can sometimes quickly find out what you are dealing with, rendering further investigation unnecessary.

## 8.3 File Type:

Generally, pestudio will parse file type information out for you. For example, pestudio will attempt to detect if you are dealing with an executable and for which architecture it was compiled. However, it is good practice to be able to verify some of this information manually. Additionally, if pestudio cannot parse out any meaningful information, you might be able to see why by looking at the contents of the file directly.

For this purpose, pestudio conveniently shows you the first bytes of a file. The file you are currently examining is an executable, and you can see the start of the MZ header as you would expect. However, if you were to see a different signature or random data at the beginning of the file, then you would know to proceed accordingly.

For example, if someone hands you a file and says that it is malicious, but it looks like it is just random data, then maybe they have made a mistake, or perhaps the file contains encrypted data that is somehow used by some malicious application. This tells you what questions to ask going forward.

## 8.3 Compilation Time

The compilation timestamp can potentially be removed or faked by the developer but is still worth examining. This information can tell you how recently the sample was created — this can be important if you are dealing with an intrusion and are trying to understand if an attacker is still active. For instance, if the sample was compiled the day before, the attacker is likely actively attempting to use this tool.

The compilation time is also useful if you are examining multiple related malware samples and are trying to establish a timeline. If you have many associated samples, it can be easier to start your analysis with the older samples. They might be easier to understand if the malware has evolved to be more complex over time.

## 8.4 Entropy

Entropy, in simple terms, is a measure of randomness and is a good indicator for determining if the file is somehow compressed or packed. A packed executable or an encrypted data file will have a more random-seeming structure than a regular executable file containing CPU instructions, i.e., encrypted data will have higher entropy. Therefore, if you see those parts of the file or the entire file have high entropy, you could guess that compressed or encrypted data is stored within.

Note: Zero entropy means no randomness, e.g., a file filled with zeroes, and the maximum value of eight represents a completely random sequence of bits.

It is essential to determine if you are dealing with an executable that is somehow packed because then you know that you must get it unpacked before you can meaningfully analyze it any further. Until then, all the functionalities will be hidden from you for any static analysis. You can also get an indication of whether the executable is packed by examining the imports. This topic will be covered in the following steps.

Note: Entropy is not always immediately telling since a packed executable can have similar entropy to one that has not been packed. Entropy is just one indicator.

## 8.5 Libraries and Functions

The information about libraries and functions is probably the most telling of what the malware might want to accomplish in broad terms.

Note: In pestudio, you will see a column called blacklist in various views. The items in this column are not inherently malicious. This only means that the item or functionality is commonly related to malware or is potentially dangerous in some way.

library (15)	blacklist (1)	type (1)	functions (153)	description
kernel32.dll	-	implicit	30	Windows NT BASE API Client DLL
user32.dll	-	implicit	1	Multi-User Windows USER API Client DLL
advapi32.dll	-	implicit	5	Advanced Windows 32 Base API
ws2_32.dll	x	implicit	14	Windows Socket 2.0 32-Bit DLL
msvcp140.dll	-	implicit	41	n/a
vcruntime140_1.dll	-	implicit	1	n/a
vcruntime140.dll	-	implicit	12	n/a
api-ms-win-crt-r...	-	implicit	18	n/a
api-ms-win-crt-s...	-	implicit	17	n/a
api-ms-win-crt-fi...	-	implicit	3	n/a
api-ms-win-crt-s...	-	implicit	4	n/a
api-ms-win-crt-h...	-	implicit	4	n/a
api-ms-win-crt-e...	-	implicit	1	n/a
api-ms-win-crt-...	-	implicit	1	n/a
api-ms-win-crt-l...	-	implicit	1	n/a

pestudio 9.20 - Malware Initial Assessment - www.winitor.com [c:\users\contosoadmin\desktop\mystery.exe]

file settings about

c:\users\contosoadmin\desktop\mystery.exe

indicators (41)	functions (153)	blacklist (22)	type (1)	ordinal (12)	library (15)
<ul style="list-style-type: none"> <li>virustotal (warning)</li> <li>dos-header (64 bytes)</li> <li>dos-stub (184 bytes)</li> <li>rich-header (10)</li> <li>file-header (Nov.2021)</li> <li>optional-header (GUI)</li> <li>directories (7)</li> <li>sections (98.20%)</li> <li>libraries (15) *</li> <li>functions (153) *</li> <li>exports (n/a)</li> <li>tls-callbacks (n/a)</li> <li>.NET (n/a)</li> <li>resources (manifest) *</li> <li>strings (858)</li> <li>debug (Nov.2021)</li> <li>manifest (asInvoker)</li> <li>version (n/a)</li> <li>certificate (n/a)</li> <li>overlay (n/a)</li> </ul>	<ul style="list-style-type: none"> <li>Sleep</li> <li>CloseHandle</li> <li>CreateProcessA</li> <li>MultiByteToWideChar</li> <li>GetModuleFileNameA</li> <li>PeekNamedPipe</li> <li>GetFileAttributesA</li> <li>CreateDirectoryA</li> <li>RtlLookupFunctionEntry</li> <li>InitializeListHead</li> <li>GetSystemTimeAsFileTime</li> <li>CreatePipe</li> <li>FormatMessageA</li> <li>WriteFile</li> <li>SetHandleInformation</li> <li>ReadFile</li> <li>CopyFileA</li> <li>GetCurrentThreadId</li> <li>GetCurrentProcessId</li> <li>QueryPerformanceCounter</li> </ul>	<ul style="list-style-type: none"> <li>-</li> <li>-</li> <li>x</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>x</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>x</li> <li>-</li> <li>-</li> <li>-</li> <li>x</li> <li>x</li> <li>-</li> </ul>	<ul style="list-style-type: none"> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> <li>implicit</li> </ul>	<ul style="list-style-type: none"> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> <li>-</li> </ul>	<ul style="list-style-type: none"> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> <li>kernel32.dll</li> </ul>

### 8.5.1 Libraries:

In the libraries view, you will see a list of libraries imported by this executable. This will give you an indication of what type of functionality the sample might have. There is a short description of the library in the description field, but you can always search online for more information or look ahead at the functions view which can sometimes be more descriptive since you can see what functions the library provides.

Note: The exports view is what you will want to look at if you analyze a DLL file. This view will tell you what functionality the DLL exposes.



## 8.5.2 Functions:

The functions view lists the external functions potentially used by the program and is relatively self-explanatory. Pestudio will also mark the functions more commonly abused or potentially dangerous as denylisted.

You can see that as ws2\_32.dll was marked as denylisted, all of the functions provided by that DLL are also marked as such. Additionally, in the functions view, you can see that this library implements networking functionality more easily.

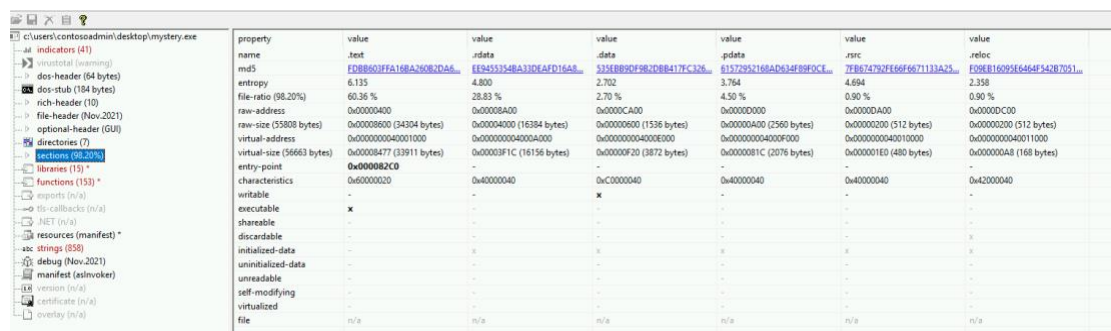
Note: If you want more details on a particular function, you can search for it in Microsoft Docs.

If you only see a few functions listed here or only a few libraries imported, it could mean that the program is packed or is resolving the addresses of the libraries and functions by itself. For example, here is the list of functions shown for a UPX packed executable:

## 8.6 Other Information

### 8.6.1 Headers and Sections

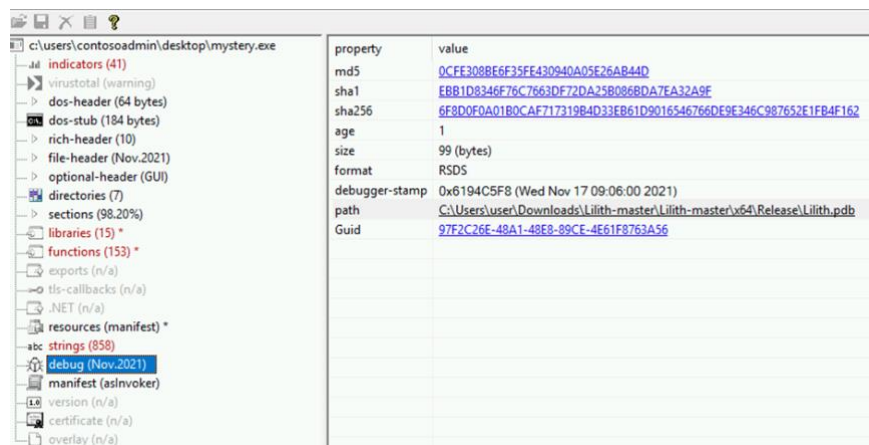
This information is usually not all that relevant for initial analysis. The relevant aspects here are parsed out by pestudio and displayed for you on the initial summary view. However, if you need to examine the file's headers or sections more closely, pestudio will also display that information in more detail as well.



property	value	value	value	value	value	value
name	.text	.rdata	.data	.pdata	.zsrc	.reloc
md5	<a href="#">FDBB60FFA16BA750B2DA6...</a>	<a href="#">EE455354BA33DEAFD16A8...</a>	<a href="#">535EBB9DF9B2DBB417FC326...</a>	<a href="#">61572952168AD634F8F8FC...</a>	<a href="#">7F8B74792FE66F667113A43...</a>	<a href="#">F08E16095E6464F542B7051...</a>
entropy	6.135	4.800	2.702	3.764	4.694	2.358
file-ratio (98.20%)	60.36 %	28.83 %	2.70 %	4.50 %	0.90 %	0.90 %
raw-address	0x00000400	0x00008A00	0x0000CA00	0x0000D000	0x0000DA00	0x0000DC00
raw-size (5808 bytes)	0x00006000 (24304 bytes)	0x00004000 (16384 bytes)	0x00006000 (1536 bytes)	0x0000A000 (2560 bytes)	0x00002000 (512 bytes)	0x00002000 (512 bytes)
virtual-address	0x0000000040001000	0x000000004000A000	0x000000004000E000	0x000000004000F000	0x0000000040010000	0x0000000040011000
virtual-size (5663 bytes)	0x00000477 (33911 bytes)	0x00003F1C (16156 bytes)	0x00000F20 (3872 bytes)	0x0000081C (2076 bytes)	0x000001E0 (480 bytes)	0x000000A8 (168 bytes)
entry-point	0x00000B2C0	-	-	-	-	-
characteristics	0x00000020	0x40000040	0xC0000040	0x40000040	0x40000040	0x42000040
writable	-	-	X	-	-	-
executable	X	-	-	-	-	-
shareable	-	-	-	-	-	-
discardable	-	-	-	-	-	-
initialized-data	-	-	X	-	-	-
uninitialized-data	-	-	-	-	-	-
unreadable	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-
virtualized	-	-	-	-	-	-
file	n/a	n/a	n/a	n/a	n/a	n/a

### 8.6.2 Debug Information

Specific information that is of particular interest in the debug view is the PDB string displayed in the path field. You don't need to understand how the PDB file is used. In this context, it is only important to understand that this path can tell you what the developer calls this program or project. This information, similarly, to the compilation timestamp, is useful when trying to understand the broader context of the sample you are analyzing.



property	value
md5	<a href="#">0CFE308BE6F25FE430940A05E26A844D</a>
sha1	<a href="#">EBB1D8346F76C7663DF72DA25B086BDA7EA32A9F</a>
sha256	<a href="#">6F8D0F0A01B0CAF717319B4D33EB61D9016546766DE9E346C987652E1FB4F162</a>
age	1
size	99 (bytes)
format	RSDS
debugger-stamp	0x6194C5F8 (Wed Nov 17 09:06:00 2021)
path	<a href="#">C:\Users\user\Downloads\Liith-master\Liith-master\v64\Release\Liith.pdb</a>
Guid	<a href="#">97F2C26E-48A1-48E8-B9CF-4E61F8763A56</a>

### 8.6.3 Strings

Sometimes taking a quick look at the strings that the executable stores in the open can give some

information about the nature of the file you are dealing with. For example, does the executable store strings that hint functionality (e.g., CMD session opened), or does it store anything that looks like a C2 address?

Usually, these types of strings are stored encrypted, but sometimes you can get lucky. What can you see in the strings view for this executable?

	encoding (2)	size (bytes)	file-offset	blacklist (10)	hint (102)	value (858)
indicators (41)	ascii	15	0x00009250	-	utility	CMD is not open
virustotal (warning)	ascii	4	0x00009364	-	utility	kill
dos-header (64 bytes)	ascii	19	0x00009438	-	utility	CMD session opened,
dos-stub (184 bytes)	ascii	18	0x00009470	-	utility	CMD session closed
rich-header (10)	ascii	9	0x00009680	-	url-pattern	10.55.0.1
file-header (Nov.2021)	ascii	23	0x0000B918	-	rtti	? ? Lockit@std@@QFAA@XZ
optional-header (GUI)	ascii	24	0x0000B932	-	rtti	? ? Lockit@std@@QFAA@H@Z
directories (7)	ascii	49	0x0000B94E	-	rtti	? ? Getgloballocale@locale@std@@CAPFAV Locimp@12@XZ
sections (98.20%)	ascii	30	0x0000B982	-	rtti	? ? Xout_of_range@std@@YAXPEBD@Z
libraries (15) *	ascii	46	0x0000B9A4	-	rtti	? ? Id@?Scodecvt@DDU_Mbstatet@@@std@@ZV@locale@?@Z
functions (153) *	ascii	36	0x0000B9D6	-	rtti	? ? Fiopen@std@@YAPEAU iobuf@@PEBDH@Z
exports (n/a)	ascii	70	0x0000B9FE	-	rtti	? ? Xlength_error@std@@YAXPEBD@Z
tls-callbacks (n/a)	ascii	74	0x0000BA20	-	rtti	? ? Getcat@?Scodecvt@DDU_Mbstatet@@@std@@SA KPEAPERBFacet@locale@?@PERV42...
.NET (n/a)	ascii	73	0x0000BA6E	-	rtti	? ? getloc@?Sbasic_streambuf@DU?Schar_traits@D@std@@@std@@QFAA?AV@locale@?@XZ
resources (manifest) *	ascii	73	0x0000BABA	-	rtti	? ? unshift@?Scodecvt@DDU_Mbstatet@@@std@@QFAHAFAU_Mbstatet@@PEAD1AEAPE...
strings (858)	ascii	57	0x0000BB06	-	rtti	? ? ?Sbasic_streambuf@DU?Schar_traits@D@std@@@std@@IFAA@XZ
debug (Nov.2021)	ascii	61	0x0000BB42	-	rtti	? ? Init@?Sbasic_streambuf@DU?Schar_traits@D@std@@@std@@IFAA@XZ
manifest (asInvoker)	ascii	58	0x0000BB82	-	rtti	? ? clear@?Sbasic_ios@DU?Schar_traits@D@std@@@std@@QFAAXH_N@Z
version (n/a)	ascii	80	0x0000BBC0	-	rtti	? ? in@?Scodecvt@DDU_Mbstatet@@@std@@QFAHAFAU_Mbstatet@@PEBD1AEAPEBDPE...
certificate (n/a)						
overlay (n/a)						

## 8.6.4 Indicators

The indicators view is a feature specific to pestudio. Pestudio will try to parse out features of an executable that might indicate that it is malicious. It can be useful to start by looking here and seeing anything interesting that pestudio has extracted. You should be able to see here that pestudio has parsed out the IP address and the PDB path that you can also find in the strings and debug views.

The indicators feature can also be useful in the sense that if there are a lot of red flags here, then there is a higher chance that you are dealing with something that is indeed malicious. However, keep in mind that if tools such as pestudio do not report anything suspicious, then that does not automatically mean that everything is fine. It might be, but it is your task to make that assessment.

	indicator (41)	detail	level
indicators (41)	The file references string(s)	type: blacklist, count: 10	1
virustotal (warning)	The file imports symbol(s)	type: blacklist, count: 22	1
dos-header (64 bytes)	The file references a URL pattern	url: 10.55.0.1	1
dos-stub (184 bytes)	The file references blacklist library(ies)	count: 1	2
rich-header (10)	The file imports anonymous function(s)	count: 12	2
file-header (Nov.2021)	The file references debug symbols	file: C:\Users\user\Downloads\Liith-master\Liith-m...	3
optional-header (GUI)	The file checksum is invalid	checksum: 0x00000000	3
directories (7)	The file references a group of API	type: diagnostic, count: 4	3
sections (98.20%)	The file references a group of API	type: file, count: 22	3
libraries (15) *	The file references a group of API	type: data-exchange, count: 4	3
functions (153) *	The file references a group of API	type: execution, count: 14	3
exports (n/a)	The file references a group of API	type: dynamic-library, count: 4	3
tls-callbacks (n/a)	The file references a group of API	type: input-output, count: 2	3
.NET (n/a)	The file references a group of API	type: registry, count: 10	3
resources (manifest) *	The file references a group of API	type: network, count: 6	3
strings (858)	The file references a group of API	type: memory, count: 10	3
debug (Nov.2021)	The file references a group of API	type: exception, count: 6	3
manifest (asInvoker)	The file references a group of API		

## 9 Exception

Exception to this procedure must be approved through the Netradyne Exception Process.



## 10 References

### 10.1 Logging

A ticket should be raised for each and every malware analysis  
Ticket can be raised through [service desk+](#) portal

### 10.2 IOC

Every IOC found during analysis, we have to block on different  
Please raise ticket for same through [service desk+](#) portal

## 11 Appendix A: Document RACI Matrix

Role/Activity	Document Owner/Functional Area Lead	Document Contributor	ND Leadership	Functional Area Team	InfoSec	All ND Member(s)
Ensure document is kept current	A	R	I, C	R, C	C	I
Ensure stakeholders are kept informed	A	R	-	R	C	-
Ensure document contains all relevant information	A	R	I, C	R, C	C	I
Ensure document adheres to document governance policy	A, R	R	I	R, C	R, C	I
Provide SME advice	I, R	A, R	I	R, C	I, C	I
Gathering and adding document contents	I	A, R	I, C	R, C	C	I
Document Approval	A	R	I, R	I	I, R	I

### Key

<b>R</b>	<b>Responsible</b>
<b>A</b>	<b>Accountable</b>
<b>C</b>	<b>Consulted</b>
<b>I</b>	<b>Informed</b>