

Configuration Manual

MSc Research Project
MSc in Cybersecurity

Surya Kant Karanwal
Student ID: 23385405

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Surya Kant Karanwal
Student ID: 23385405
Programme: MSc in Cybersecurity **Year:** 2025-2026
Module: Practicum Part 2
Lecturer: Prof. Khadija Hafeez
Submission Due Date: 28th January 2026
Project Title: A lightweight framework for detecting malicious traffic evasion via DNS over HTTPS

Word Count: 1204 **Page Count:** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Surya Kant Karanwal

Date: 26th January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Surya Kant Karanwal
Student ID: 23385405

1 About Configuration Manual

This configuration manual will showcase all the steps taken to achieve the final goal and development of the artefact. Anyone can use this manual to recreate the proposed model on a PC with 4 GB of RAM and an Intel i3 processor or more.

2 Environment Setup

- 2.1 VMware Workstation Pro was used as a type 2 hypervisor for the setup, which can be downloaded for free from the Broadcom official website. In the experiment, Kali Linux was used as a guest OS for making the sandboxing environment. Kali Linux can be downloaded from the official Kali Linux website. Download its ISO file and

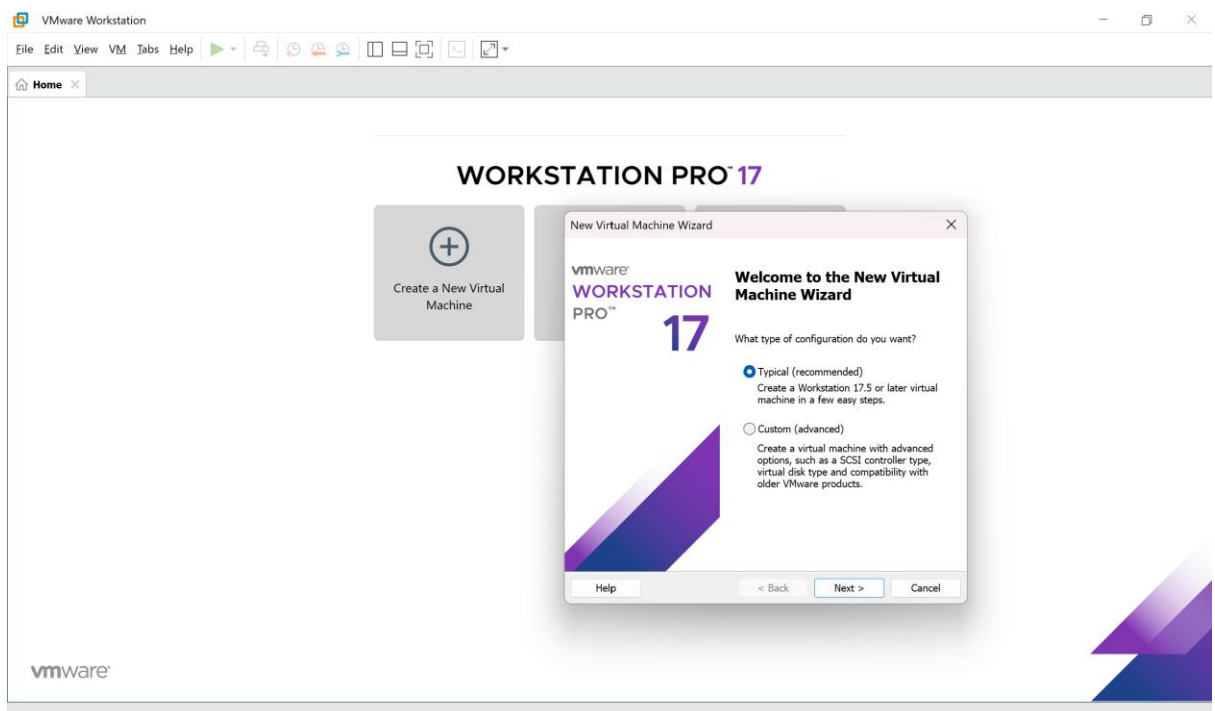


Fig 1. New virtual machine wizard

Open the VMware Workstation, click on Create a New Virtual Machine and follow all the steps of the New Virtual Machine wizard.

Once the the steps are completed, a GUI of kali linux can be accessed in the virutalized environment

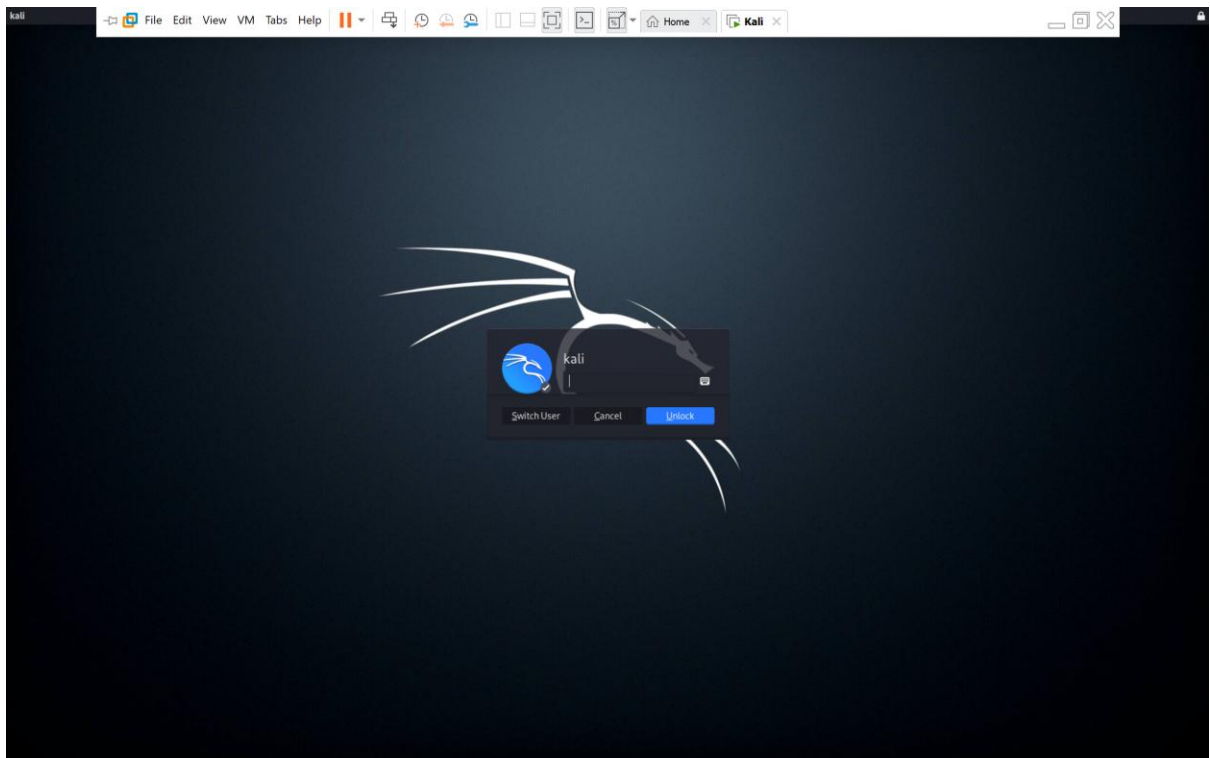


Fig 2. Installed guest OS (Kali Linux)

- 2.2 Once Kali Linux is stable and up and running, you can start by installing all the tools and dependencies of the model, which include Suricata, tshark, Wireshark and Python.

Open CLI and First update the system using this command `#sudo apt update`

To install Suricata use - `#sudo apt install suricata -y`

To install Wireshark use - `#sudo apt install wireshark -y`

To install Tshark use - `#sudo apt install tshark -y`

To install Python use - `#sudo apt install python3 python3-pip -y`

Once everything is installed, they can be verified by looking at the version of each tool, which can be extracted using one simple command `#<name of the tool> --verion`

```
kali@kali: ~  
Session Actions Edit View Help  
$ wireshark --version  
Wireshark 4.4.9.  
  
Copyright 1998-2025 Gerald Combs <gerald@wireshark.org> and contributors.  
Licensed under the terms of the GNU General Public License (version 2 or later).  
This is free software; see the file named COPYING in the distribution. There is  
NO WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
Compiled (64-bit) using GCC 14.3.0, with Glib 2.84.4, with Qt 6.8.2, with  
libpcap, with POSIX capabilities (Linux), with libnl 3, with zlib 1.3.1, without  
zlib-ng, with PCRE2, with Lua 5.4.8, with GnuTLS 3.8.10 and PKCS #11 support,  
with Gcrypt 1.11.2, with Kerberos (MIT), with MaxMind, with nghttp2 1.64.0, with  
nghttp3 1.8.0, with brotli, with LZ4, with Zstandard, with Snappy, with libxml2  
2.14.5, with libsmi 0.4.8, with Minizip 1.3.1, with QtMultimedia, with QtDBus,  
without automatic updates, with binary plugins.  
  
Running on Linux 6.16.8+kali-amd64, with Intel(R) Core(TM) Ultra 5 125H (with  
SSE4.2), with 3883 MB of physical memory, with Glib 2.84.4, with Qt 6.8.2, with  
libpcap 1.10.5 (with TPACKET_V3), with zlib 1.3.1, with PCRE2 10.46 2025-08-27,  
with c-ares 1.34.5, with GnuTLS 3.8.10, with Gcrypt 1.11.2, with nghttp2 1.64.0,  
with nghttp3 1.12.0, with brotli 1.1.0, with LZ4 1.10.0, with Zstandard 1.5.7,  
with libsmi 0.4.8, with LC_TYPE=en_US.UTF-8, binary plugins supported.  
  
$ python --version  
Python 3.13.7  
  
$ tshark --version  
TShark (Wireshark) 4.4.9.  
  
Copyright 1998-2025 Gerald Combs <gerald@wireshark.org> and contributors.  
Licensed under the terms of the GNU General Public License (version 2 or later).  
This is free software; see the file named COPYING in the distribution. There is  
NO WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
Compiled (64-bit) using GCC 14.3.0, with Glib 2.84.4, with libpcap, with POSIX  
capabilities (Linux), with libnl 3, with zlib 1.3.1, without zlib-ng, with  
PCRE2, with Lua 5.4.8, with GnuTLS 3.8.10 and PKCS #11 support, with Gcrypt  
1.11.2, with Kerberos (MIT), with MaxMind, with nghttp2 1.64.0, with nghttp3  
1.8.0, with brotli, with LZ4, with Zstandard, with Snappy, with libxml2 2.14.5,  
with libsmi 0.4.8, with binary plugins.  
  
Running on Linux 6.16.8+kali-amd64, with Intel(R) Core(TM) Ultra 5 125H (with  
SSE4.2), with 3883 MB of physical memory, with Glib 2.84.4, with libpcap 1.10.5  
(with TPACKET_V3), with zlib 1.3.1, with PCRE2 10.46 2025-08-27, with c-ares  
1.34.5, with GnuTLS 3.8.10, with Gcrypt 1.11.2, with nghttp2 1.64.0, with  
nghttp3 1.12.0, with brotli 1.1.0, with LZ4 1.10.0, with Zstandard 1.5.7, with  
libsmi 0.4.8, with LC_TYPE=en_US.UTF-8, binary plugins supported.  
  
$ suricata --version  
suricata: unrecognized option '--version'  
Suricata 8.0.1 ("undefined")  
USAGE: suricata [OPTIONS] [BPF FILTER]  
  
General:  
-v : be more verbose (use multiple times to increase verbosity)  
-c <path> : path to configuration file  
-l <dir> : default log directory  
--include <path> : additional configuration file  
--set name=value : set a configuration value  
--pidfile <file> : write pid to this file  
-T : test configuration file (use with -c)  
--init-errors-fatal : enable fatal failure on signature init error
```

Fig 3. Verification and version of all tools

3 Isolation from the host machine

Once all the tools and dependencies are installed and all the datasets are downloaded. Make some changes to the VMware settings to isolate the guest OS from the host OS.

3.1 Switch to host only mode

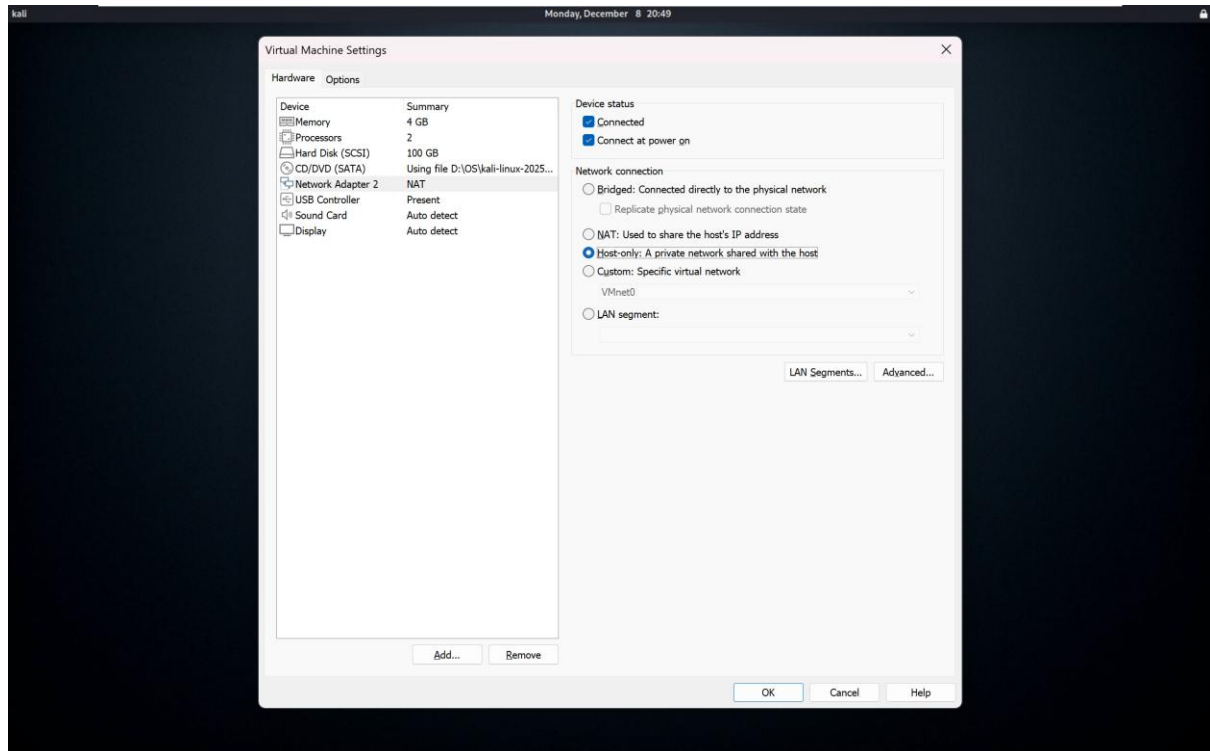


Fig 4. Switching to host only mode

3.2 Disable drag & drop and copy & paste

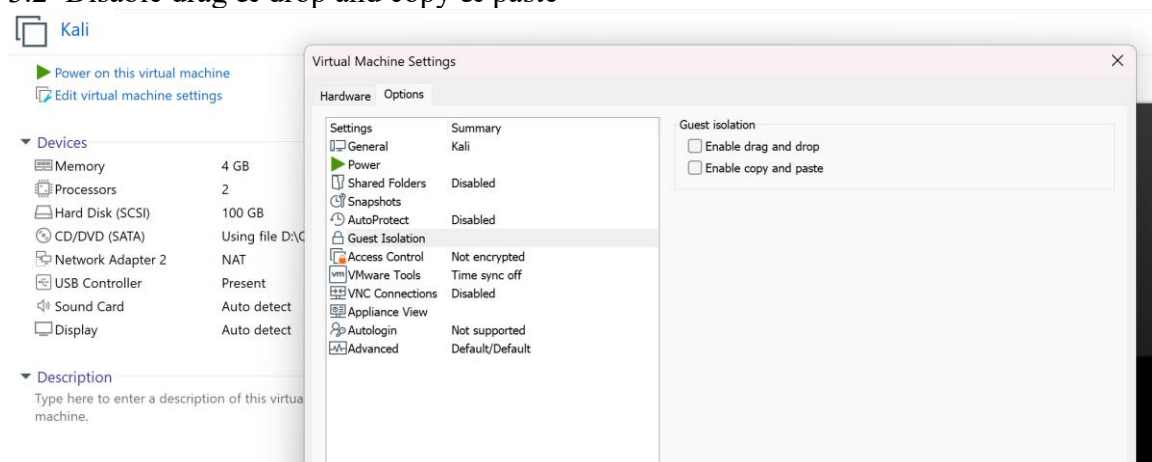


Fig 5. Disabling drag and drop & copy and paste

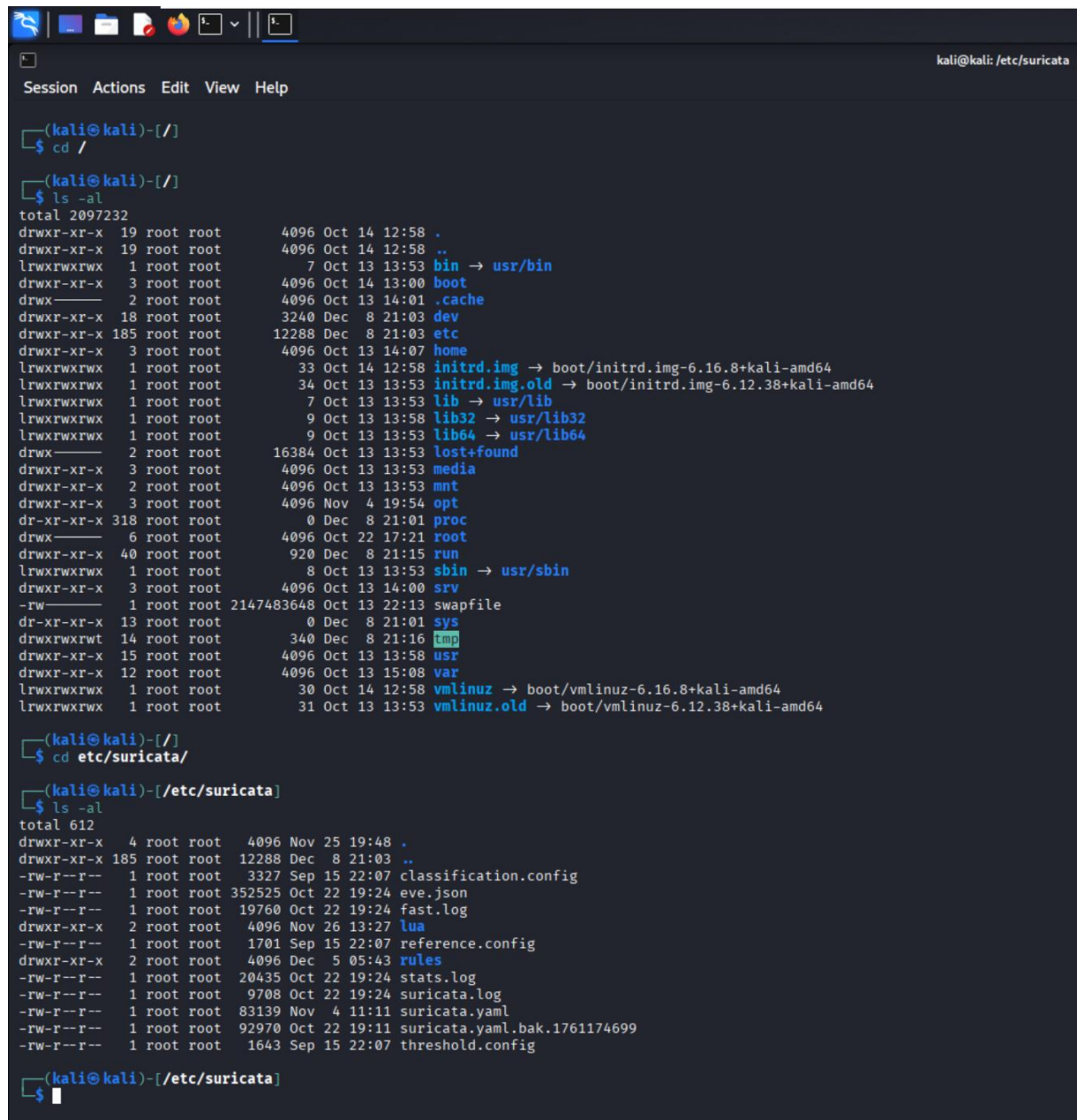
These two steps will do the work. For additional security, disable all network adapters on Kali Linux and disable the shared folder.

4 File location and Suricata changes

Before moving to any other steps, it is very crucial to know the location of Suricata files

- 4.1 When anyone will open the CLI, it will be inside the user directory by default, first, go to the home directory by using `#cd ..` and then again use the same command to enter into backend file system or use `#cd /` command to jump directly into the file system.

For there you will find the Suricata files inside `etc` directory, use the command `#cd etc/suricata`. Then use the `#ls` command to check all the files inside suricata.



```
(kali@kali)-[/]
$ cd /

(kali@kali)-[/]
$ ls -al
total 2097232
drwxr-xr-x 19 root root      4096 Oct 14 12:58 .
drwxr-xr-x 19 root root      4096 Oct 14 12:58 ..
lrwxrwxrwx  1 root root         7 Oct 13 13:53 bin -> usr/bin
drwxr-xr-x  3 root root      4096 Oct 14 13:00 boot
drwx----- 2 root root         2 Oct 13 14:01 .cache
drwxr-xr-x 18 root root     3240 Dec  8 21:03 dev
drwxr-xr-x 185 root root    12288 Dec  8 21:03 etc
drwxr-xr-x  3 root root      4096 Oct 13 14:07 home
lrwxrwxrwx  1 root root         33 Oct 14 12:58 initrd.img -> boot/initrd.img-6.16.8+kali-amd64
lrwxrwxrwx  1 root root         34 Oct 13 13:53 initrd.img.old -> boot/initrd.img-6.12.38+kali-amd64
lrwxrwxrwx  1 root root         7 Oct 13 13:53 lib -> usr/lib
lrwxrwxrwx  1 root root         9 Oct 13 13:58 lib32 -> usr/lib32
lrwxrwxrwx  1 root root         9 Oct 13 13:53 lib64 -> usr/lib64
drwx----- 2 root root    16384 Oct 13 13:53 lost+found
drwxr-xr-x  3 root root      4096 Oct 13 13:53 media
drwxr-xr-x  2 root root      4096 Oct 13 13:53 mnt
drwxr-xr-x  3 root root      4096 Nov  4 19:54 opt
dr-xr-xr-x 318 root root         0 Dec  8 21:01 proc
drwx----- 6 root root         6 Oct 22 17:21 root
drwxr-xr-x 40 root root      920 Dec  8 21:15 run
drwxr-xr-x  1 root root         8 Oct 13 13:53 sbin -> usr/sbin
drwxr-xr-x  3 root root      4096 Oct 13 14:00 srv
-rw----- 1 root root 2147483648 Oct 13 22:13 swapfile
dr-xr-xr-x 13 root root         0 Dec  8 21:01 sys
drwxrwxrwt 14 root root      340 Dec  8 21:16 tmp
drwxr-xr-x 15 root root      4096 Oct 13 13:58 usr
drwxr-xr-x 12 root root      4096 Oct 13 15:08 var
lrwxrwxrwx  1 root root         30 Oct 14 12:58 vmlinuz -> boot/vmlinuz-6.16.8+kali-amd64
lrwxrwxrwx  1 root root         31 Oct 13 13:53 vmlinuz.old -> boot/vmlinuz-6.12.38+kali-amd64

(kali@kali)-[/]
$ cd etc/suricata/

(kali@kali)-[/etc/suricata]
$ ls -al
total 612
drwxr-xr-x  4 root root      4096 Nov 25 19:48 .
drwxr-xr-x 185 root root    12288 Dec  8 21:03 ..
-rw-r--r--  1 root root      3327 Sep 15 22:07 classification.config
-rw-r--r--  1 root root    352525 Oct 22 19:24 eve.json
-rw-r--r--  1 root root     19760 Oct 22 19:24 fast.log
drwxr-xr-x  2 root root      4096 Nov 26 13:27 lua
-rw-r--r--  1 root root      1701 Sep 15 22:07 reference.config
drwxr-xr-x  2 root root      4096 Dec  5 05:43 rules
-rw-r--r--  1 root root     20435 Oct 22 19:24 stats.log
-rw-r--r--  1 root root      9708 Oct 22 19:24 suricata.log
-rw-r--r--  1 root root     83139 Nov  4 11:11 suricata.yaml
-rw-r--r--  1 root root     92970 Oct 22 19:11 suricata.yaml.bak.1761174699
-rw-r--r--  1 root root      1643 Sep 15 22:07 threshold.config
```

Fig 6. Location of important files

- 4.2 Now create your customer rules files inside the rules folder by using `#sudo nano <file name>.rules`. Additionally, a custom log folder can also be created.

- 4.3 Now, before working on the rules, some changes must be made inside the main file of Suricata, which is `suricata.yaml` to activate the functionality of TLS and JA3 tracking. Additionally, the main rule file used for the implementation should be called under `yaml` file.

Enabling TLS functionality

```
# each app-protocol. Warning: VERY verbose

# Plugins -- Experimental -- specify the filename for each plugin shared object
plugins:
  #- /usr/lib/suricata/pfring.so
  #- /usr/lib/suricata/napatech.so
  #- /usr/lib/suricata/ndpi.so
  #- /path/to/plugin.so

# Configure the type of alert (and other) logging you would like.
outputs:
  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: fast.log
      append: yes
      filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular/syslog/unix_dgram/unix_stream/redis
      filename: eve.json
      types:
        - tls:
            enabled: yes
        - alert
        - flow
```

Fig 7. Changes made to enable TLS detection

Enable JA3 functionality

```
app-layer:
  # error-policy: ignore
  protocols:
    telnet:
      enabled: yes
    rfb:
      enabled: yes
      detection-ports:
        dp: 5900, 5901, 5902, 5903, 5904, 5905, 5906, 5907, 5908, 5909
    mqtt:
      enabled: yes
      # max-msg-length: 1 MiB
      # subscribe-topic-match-limit: 100
      # unsubscribe-topic-match-limit: 100
      # Maximum number of live MQTT transactions per flow
      # max-tx: 4096
    krb5:
      enabled: yes
    bittorrent-dht:
      enabled: yes
    snmp:
      enabled: yes
    ike:
      enabled: yes
    tls:
      enabled: yes
      detection-ports:
        dp: 443
      ja3-fingerprints: yes
      ja4-fingerprints: yes
```

Fig 8. Changes made to enable JA3 detection

Adding Rule File, this will help Suricata engine in locating the configured rules

```
default-rule-path: /etc/suricata/rules

rule-files:
- local.rules
```

Fig 9. Added custom rules files into suricata.yaml

5 Dataset Details

The dataset is available in two forms, which are PCAP files and CSV files. All the datasets used in this study have been used in various past renowned research studies, which shows the credibility of the selected datasets. In total, 3 datasets were downloaded, from which 5 Malicious PCAP files, which are dnstt, padcrypt, sison, tcp-over-dns and tuns, and 4 segments of Normal traffic were selected for the final testing and evaluation.

The Fig shows a folder with all PCAP files, and sison PCAP file is opened in Wireshark, which also shows information about JA3 fingerprint.

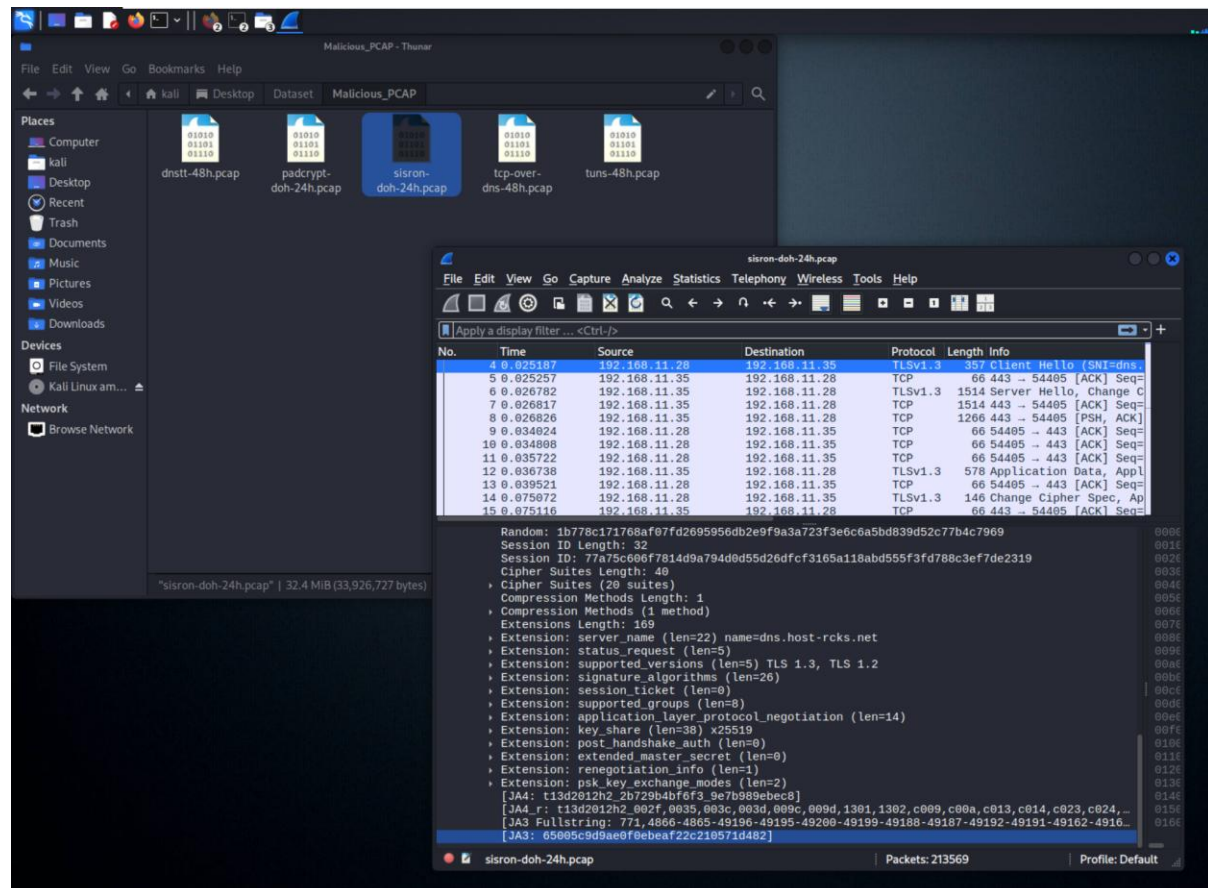
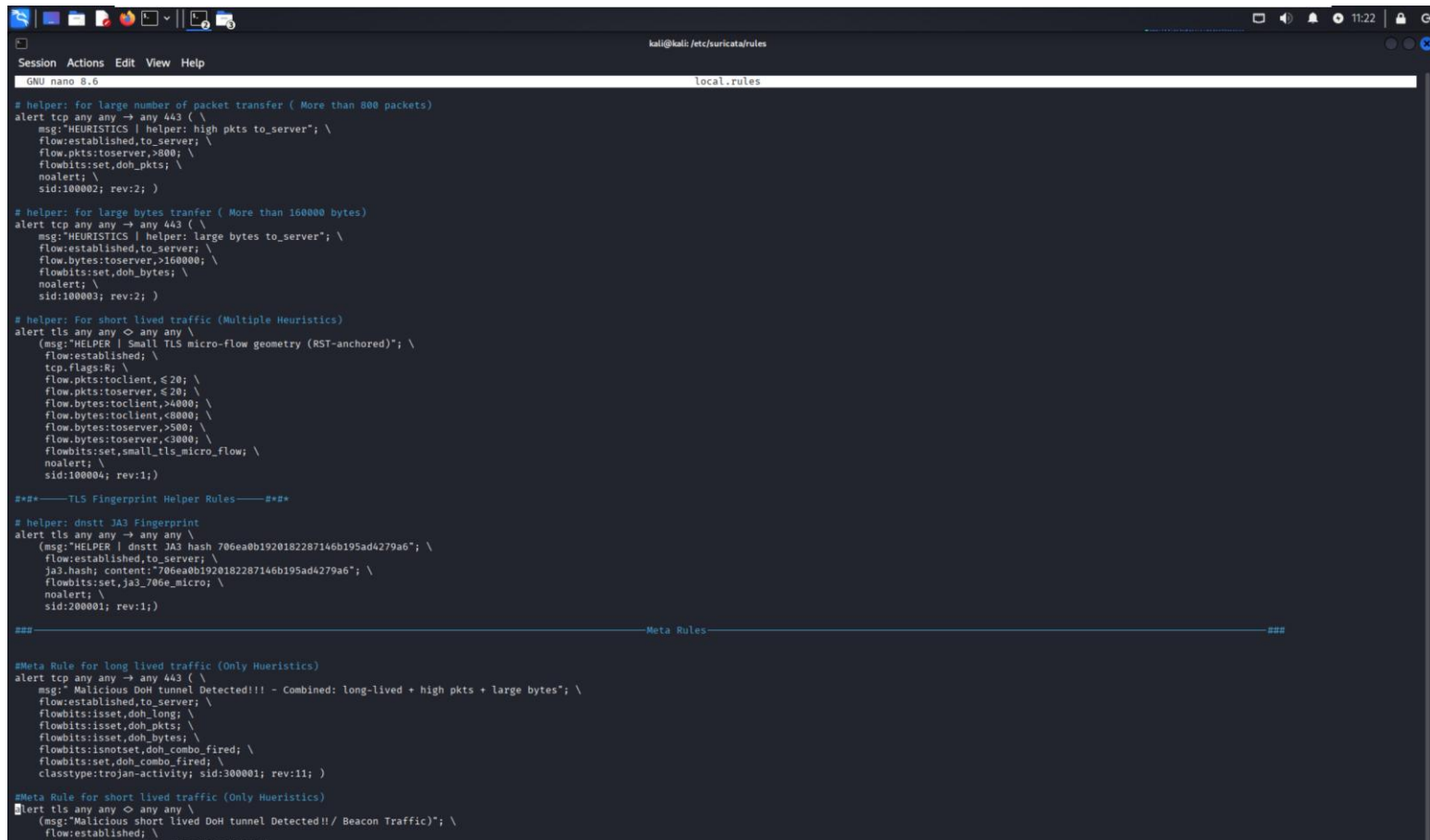


Fig 10. Wireshark Analysis

The datasets can be downloaded from - <https://github.com/doh-traffic-dataset/DoH-Tunnel-Traffic-HKD?tab=readme-ov-file> & <https://www.unb.ca/cic/datasets/dohbrw-2020.html>

6 Rule Creation

Now for the rule creation, open the rules files with superuser permission, in this case, the file name is local.rules, which can be opened using `#sudo nano rules/local.rules`. All the rules can be developed and customised in this file.



```
GNU nano 8.6 local.rules
# helper: for large number of packet transfer ( More than 800 packets)
alert tcp any any -> any 443 ( \
  msg:"HEURISTICS | helper: high pkts to_server"; \
  flow:established,to_server; \
  flow.packets:toserver,>800; \
  flowbits:set,doh_pkts; \
  noalert; \
  sid:100002; rev:2; )

# helper: for large bytes transfer ( More than 160000 bytes)
alert tcp any any -> any 443 ( \
  msg:"HEURISTICS | helper: large bytes to_server"; \
  flow:established,to_server; \
  flow.bytes:toserver,>160000; \
  flowbits:set,doh_bytes; \
  noalert; \
  sid:100003; rev:2; )

# helper: For short lived traffic (Multiple Heuristics)
alert tls any any <> any any \
  (msg:"HELPER | Small TLS micro-flow geometry (RST-anchored)"; \
  flow:established; \
  tcp.flags:R; \
  flow.packets:toclient,<=20; \
  flow.packets:toserver,<=20; \
  flow.bytes:toclient,>4000; \
  flow.bytes:toclient,<8000; \
  flow.bytes:toserver,>500; \
  flow.bytes:toserver,<3000; \
  flowbits:set,small_tls_micro_flow; \
  noalert; \
  sid:100004; rev:1;)

##### TLS Fingerprint Helper Rules #####

# helper: dnstt JA3 Fingerprint
alert tls any any -> any any \
  (msg:"HELPER | dnstt JA3 hash 706ea0b1920182287146b195ad4279a6"; \
  flow:established,to_server; \
  ja3.hash; content:"706ea0b1920182287146b195ad4279a6"; \
  flowbits:set,ja3_706e_micro; \
  noalert; \
  sid:200001; rev:1;)

##### Meta Rules #####

#Meta Rule for long lived traffic (Only Heuristics)
alert tcp any any -> any 443 ( \
  msg:"Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes"; \
  flow:established,to_server; \
  flowbits:isset,doh_long; \
  flowbits:isset,doh_pkts; \
  flowbits:isset,doh_bytes; \
  flowbits:isnotset,doh_combo_fired; \
  flowbits:set,doh_combo_fired; \
  classtype:trojan-activity; sid:300001; rev:11; )

#Meta Rule for short lived traffic (Only Heuristics)
alert tls any any <> any any \
  (msg:"Malicious short lived DoH tunnel Detected!!/ Beacon Traffic"; \
  flow:established; \
  flowbits:isset,small_tls_micro_flow; \
  flowbits:isset,ja3_706e_micro; \
  flowbits:isset,doh_long; \
  flowbits:isset,doh_pkts; \
  flowbits:isset,doh_bytes; \
  flowbits:isset,doh_combo_fired; \
  classtype:trojan-activity; sid:300002; rev:1; )
```

Fig 11. Custom rules configured

```
# helper: for large bytes tranfer ( More than 160000 bytes)
alert tcp any any -> any 443 ( \
  msg:"HEURISTICS | helper: large bytes to_server"; \
  flow:established,to_server; \
  flow.bytes:toserver,>160000; \
  flowbits:set,doh_bytes; \
  noalert; \
  sid:100003; rev:2; )
```

Fig 12. Example of a helper rule

Structure Explanation

1 - alert tcp any any -> any 443

alert <protocol name> <source ip> <source port number> <direction> <destination ip>
<destination port number>

2 - msg:< Write your message>

3 - flow: established, to_server

flow:<look for established TCP connection>, <direction should be client to server>
flow.byts

4 - flow.bytes: toserver, >16000

This flow.bytes is already defined in Suricata documentation to track bytes.

5 - flowbits: set, doh_bytes;

If the condition is matched, this flowbits will keep a record of it and save it into a variable which is doh_bytes.

6 - noalert;

Since this rule is just a helper rule, there is no need to alert on a single threshold

7 - sid 10003; rev:2;

sid is the unique identifier for the rule, and rev keeps track of updates on the specific rule.

```
alert tcp any any -> any 443 ( \
  msg:" Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes"; \
  flow:established,to_server; \
  flowbits:isset,doh_long; \
  flowbits:isset,doh_pkts; \
  flowbits:isset,doh_bytes; \
  flowbits:isnotset,doh_combo_fired; \
  flowbits:set,doh_combo_fired; \
  classtype:trojan-activity; sid:300001; rev:11; )
```

Fig 13. Example of a meta rule

All the flowbits are sent to the meta rule. Once all the required flowbits are received, it will trigger the rule

7 Testing

Once all the setup and rules are ready, testing can be initiated. Now, when the testing will be done on live traffic some steps can vary there, but here these steps will give some idea of testing steps.

Use this to get the information of total session inside a PCAP file=

```
#tshark -r <PCAP file name> -T fields -e tcp.stream | sort -n | uniq | wc -l
```

```
(kali㉿kali)-[~/Desktop/Dataset/Malicious_PCAP]
$ tshark -r siron-doh-24h.pcap -T fields -e tcp.stream | sort -n | uniq | wc -l
38

(kali㉿kali)-[~/Desktop/Dataset/Malicious_PCAP]
$ tshark -r dnstt-48h.pcap -T fields -e tcp.stream | sort -n | uniq | wc -l
1152
```

Fig 14. Tshark command to extract the total number of flow
Use this command to fetch all the JA3 signatures

```
#python3 /home/kali/ja3/python/ja3.py <PCAP file name>
```

[illegible]

Fig 15. Extraction of JA3 fingerprint using Python

This information is fetched just to get an idea of the expected results, for example by looking at the number of flows, the expectations for the alerts will be known.

7.2 Feed the file into Suricata

User this command - **#sudo suricata -r <PCAP file name> -S <PCAP file path> -l <log file path>**

```
(kali@kali)~[~/Desktop/Dataset/Malicious_PCAP]
$ sudo suricata -d /etc/suricata/rules/local.rules -l /var/log/suricata -k none
Notice: Suricata is Suricata version 4.4.1 RELEASE running in OSSE mode [logver:1;suricata.c:1208]
Info: cpu: CPUs/cores online: 2 [UtilCpuPrintSummary:util-cpu.c:149]
Info: suricata: Setting engine mode to IDS mode by default [PostConfigLoadedSetup:suricata.c:12795]
Info: exception-policy: master detection-policy set to: auto [ExceptionPolicyMasterParse:util-exception-policy.c:189]
Info: exception-policy: app-layer-error-policy: ignore (defined via 'exception-policy' master switch) [ExceptionPolicyGetDefault:util-exception-policy.c:288]
Info: app-layer-http: 'default' server has 'request-body-minimal-inspect-size' set to 31539 and 'request-body-inspect-window' set to 3975 after randomization. [HTTPConfigSetDefaultsPhase2:app-layer-http.c:1994]
Info: app-layer-http: 'default' server has 'response-body-minimal-inspect-size' set to 42781 and 'response-body-inspect-window' set to 15854 after randomization. [HTTPConfigSetDefaultsPhase2:app-layer-http.c:12007]
Info: smb: read: max record size: 16777216, max queued chunks 64, max queued size 67108864 [suricata::smb::SMBRegisterSMBParser:smr.rs:15277]
Info: smb: write: max record size: 16777216, max queued chunks 64, max queued size 67108864 [suricata::smb::SMBRegisterSMBParser:smr.rs:15279]
Info: smb: guid: max cache size: 1024 [suricata::smb::SMBRegisterSMBParser:smr.rs:15311]
Info: app-layer-dmz: Protocol detection and parser disabled for DMZ. [RegisterDMZParser:app-layer-dmz.c:1555]
Info: suricata: Preparing unexpected signal handling [InitSignalHandler:suricata.c:12260]
Info: host: allocated 262144 bytes of memory for the host hash... 4096 buckets of size 64 [HostInitConfig:host.c:1249]
Info: host: preallocated 1000 hosts of size 128 [HostInitConfig:host.c:1253]
Info: host: Host memory usage: 382144 bytes, maximum: 33554432 [HostInitConfig:host.c:1277]
Info: core-dump-config: Core dump size set to unlimited. [CoreDumpLoadConfig:util-core-dump-config.c:169]
Info: exception-policy: defrag-memcap-policy: ignore (defined via 'exception-policy' master switch) [ExceptionPolicyGetDefault:util-exception-policy.c:288]
Info: defrag-hash: allocated 3145728 bytes of memory for the defrag hash... 65536 buckets of size 48 [DefragInitConfig:defrag-hash.c:1249]
Info: defrag-hash: preallocated 65535 defrag trackers of size 144 [DefragInitConfig:defrag-hash.c:1277]
Info: defrag-hash: defrag memory usage: 12582768 bytes, maximum: 33554432 [DefragInitConfig:defrag-hash.c:1285]
Info: exception-policy: flow-memcap-policy: ignore (defined via 'exception-policy' master switch) [ExceptionPolicyGetDefault:util-exception-policy.c:288]
Info: flow: flow size 296, memcap allows for 453438 flows. Per hash row in perfect conditions 6 [FlowInitConfig:flow.c:1667]
Info: stream-tcp: stream 'prealloc-sessions': 2048 (per thread) [StreamTcpInitConfig:stream-tcp.c:1521]
Info: stream-tcp: stream 'memcap': 67108864 [StreamTcpInitConfig:stream-tcp.c:1543]
Info: stream-tcp: stream 'midstream' session pickups: disabled [StreamTcpInitConfig:stream-tcp.c:1549]
Info: stream-tcp: stream 'async-one-side': disabled [StreamTcpInitConfig:stream-tcp.c:1557]
Info: stream-tcp: stream 'checksum-validation': disabled [StreamTcpInitConfig:stream-tcp.c:1572]
Info: exception-policy: stream-memcap-policy: ignore (defined via 'exception-policy' master switch) [ExceptionPolicyGetDefault:util-exception-policy.c:288]
Info: exception-policy: stream-midstream-policy: ignore (defined via 'exception-policy' master switch) [ExceptionPolicyGetDefault:util-exception-policy.c:288]
Info: stream-tcp: stream 'inline': disabled [StreamTcpInitConfig:stream-tcp.c:1604]
Info: stream-tcp: stream 'bypass': disabled [StreamTcpInitConfig:stream-tcp.c:1617]
Info: stream-tcp: stream 'reassemble-urgent-policy': oob [StreamTcpInitConfig:stream-tcp.c:1640]
Info: stream-tcp: stream 'checksum-validation': disabled [StreamTcpInitConfig:stream-tcp.c:1667]
Info: stream-tcp: stream 'max-syn-queued': 10 [StreamTcpInitConfig:stream-tcp.c:1681]
Info: stream-tcp: stream 'max-synack-queued': 5 [StreamTcpInitConfig:stream-tcp.c:1694]
Info: stream-tcp: stream 'reassemble-memcap': 26835456 [StreamTcpInitConfig:stream-tcp.c:1715]
Info: stream-tcp: stream 'reassemble-depth': 1048576 [StreamTcpInitConfig:stream-tcp.c:1734]
Info: stream-tcp: stream 'reassemble-toserver-chunk-size': 2610 [StreamTcpInitConfig:stream-tcp.c:1806]
```

Fig 16. Processing of PCAP file through Suricata

7.3 To see the logs use the command #cat <log file name>

```
(kali@kali)~[~/Desktop/DOH-Tunnel-Traffic-HKD/DOH-Pcaps/DOH-Pcaps-48h]
$ cat suricata-logs/fast.log
10/31/2021-20:46:35.135826 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-20:46:35.135826 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36914 -> 192.168.11.16:443
10/31/2021-20:51:35.139694 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-20:46:35.135826 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36918 -> 192.168.11.16:443
10/31/2021-20:51:35.083538 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-20:56:35.086373 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-20:56:35.141285 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-20:56:35.141285 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36924 -> 192.168.11.16:443
10/31/2021-21:01:35.179424 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:01:35.179424 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36934 -> 192.168.11.16:443
10/31/2021-21:01:35.089698 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:06:35.094586 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:06:35.182698 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:06:35.182698 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36936 -> 192.168.11.16:443
10/31/2021-21:11:35.168437 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:11:35.168437 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36946 -> 192.168.11.16:443
10/31/2021-21:11:35.100753 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:16:35.161857 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:16:35.161857 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36948 -> 192.168.11.16:443
10/31/2021-21:16:35.103851 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:21:35.109681 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:21:35.173881 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:21:35.173881 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36956 -> 192.168.11.16:443
10/31/2021-21:26:35.177904 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:26:35.177904 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36960 -> 192.168.11.16:443
10/31/2021-21:26:35.114531 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
10/31/2021-21:31:35.173045 [**] [1:300002:6] Malicious short lived DoH tunnel Detected!! / Beacon Traffic) [**] [Classification: A Network Trojan was detected] [Priority:
10/31/2021-21:31:35.173045 [**] [1:400001:1] Attention Malicious short lived DoH traffic detected!! with matching JA3 706ea0b1920182287146b195ad4279a6 [**] [Classificati
.12:36968 -> 192.168.11.16:443
10/31/2021-21:31:35.110230 [**] [1:300001:11] Malicious DoH tunnel Detected!!! - Combined: long-lived + high pkts + large bytes [**] [Classification: A Network Trojan w
1.16:443
```

Fig 17. Example of alerts generated by the model

7.4 To see the total number of alert generated per rule, use #grep -c <rule s_id> <log files name>

```
(kali@kali)~[~/Desktop/Dataset/Malicious_PCAP]
$ grep -c "300001" suricata-logs/fast.log
574
```

Fig 18. Number of alerts generated by the specific rule

References

Gunter, D. (2022). *Writing Suricata Rules: Understanding the Basic Rule Format - Insane Cyber*. [online] Insane Cyber. Available at: <https://insanecyber.com/writing-suricata-rules-understanding-the-basic-rule-format/> .

Suricata. (n.d.). *Documentation*. [online] Available at: <https://suricata.io/documentation/> .

Inc, S.N. (n.d.). *Suricata Rules*. [online] www.stamus-networks.com. Available at: <https://www.stamus-networks.com/suricata-rules> .

Wireshark (n.d.). *tshark - The Wireshark Network Analyzer 3.2.2*. [online] www.wireshark.org. Available at: <https://www.wireshark.org/docs/man-pages/tshark.html> .