National College of Ireland

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 form the official Kali Linux website. Download its ISO file and

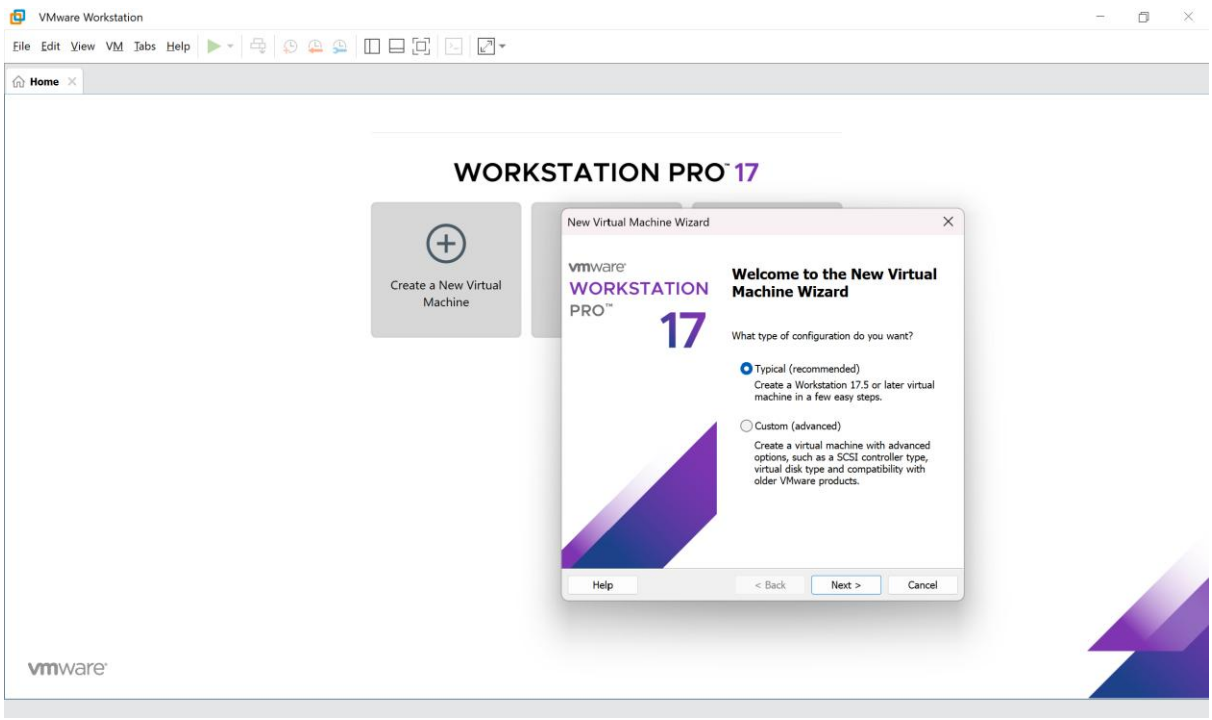

Fig 1. New virtual machine wizard

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

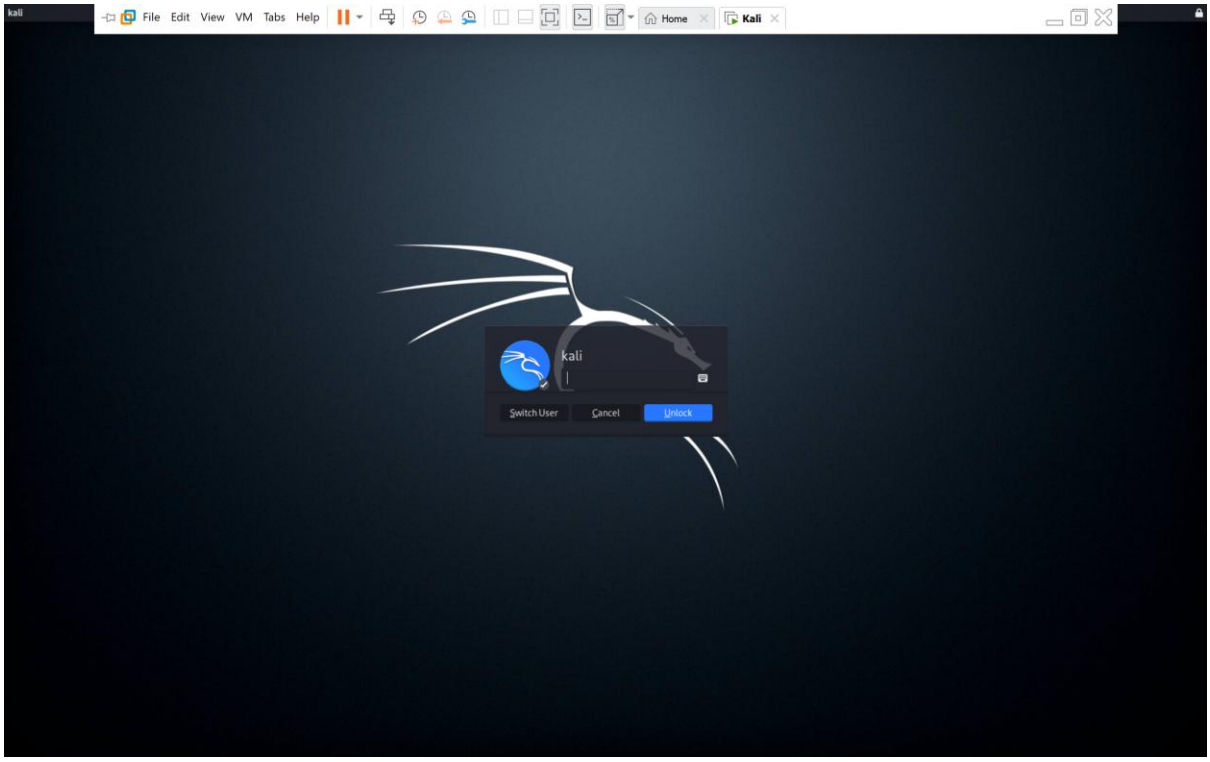

Fig 2. Installed guest OS (Kali Linux)

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

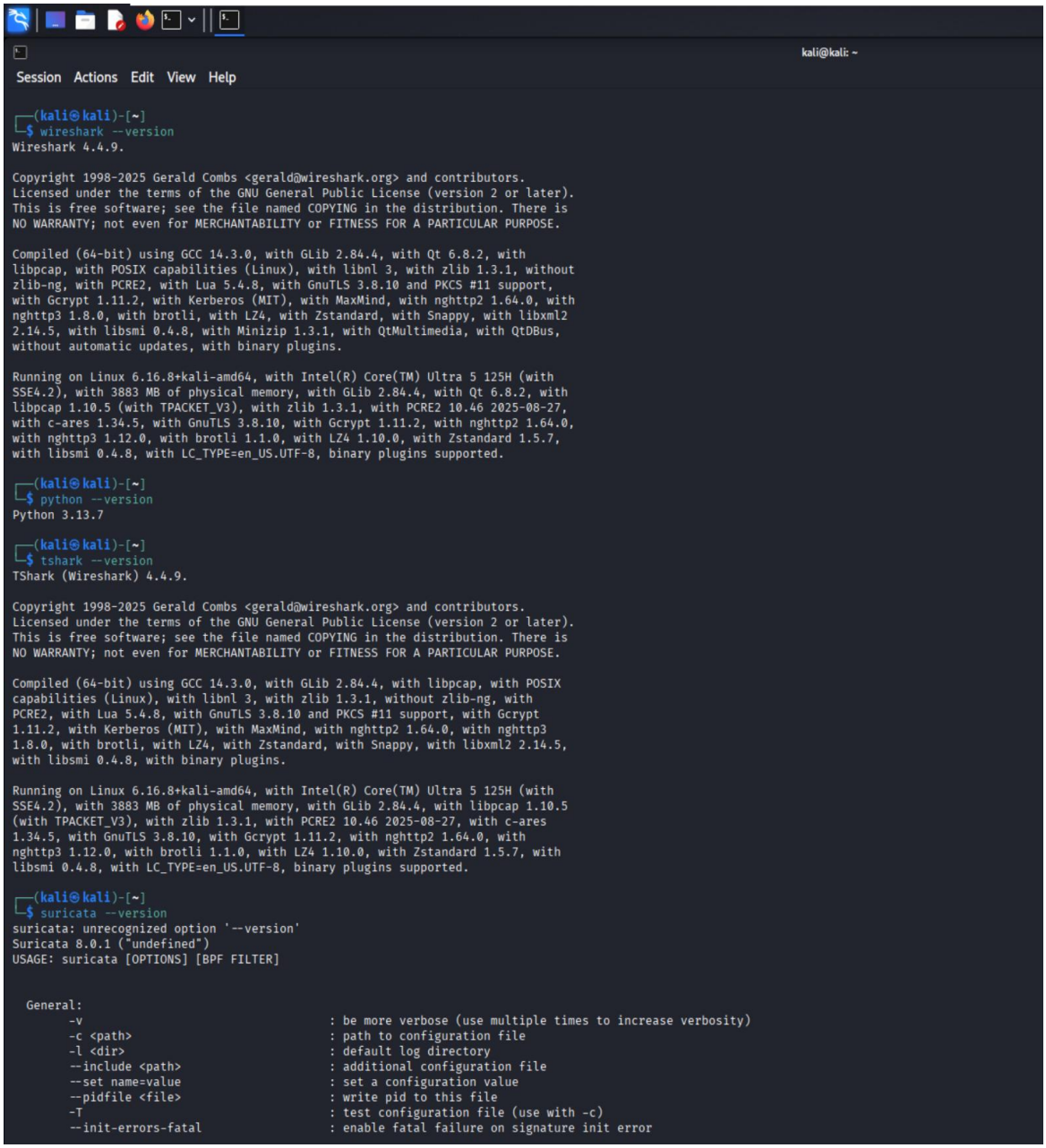

Fig 3. Verification and version of all tools

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

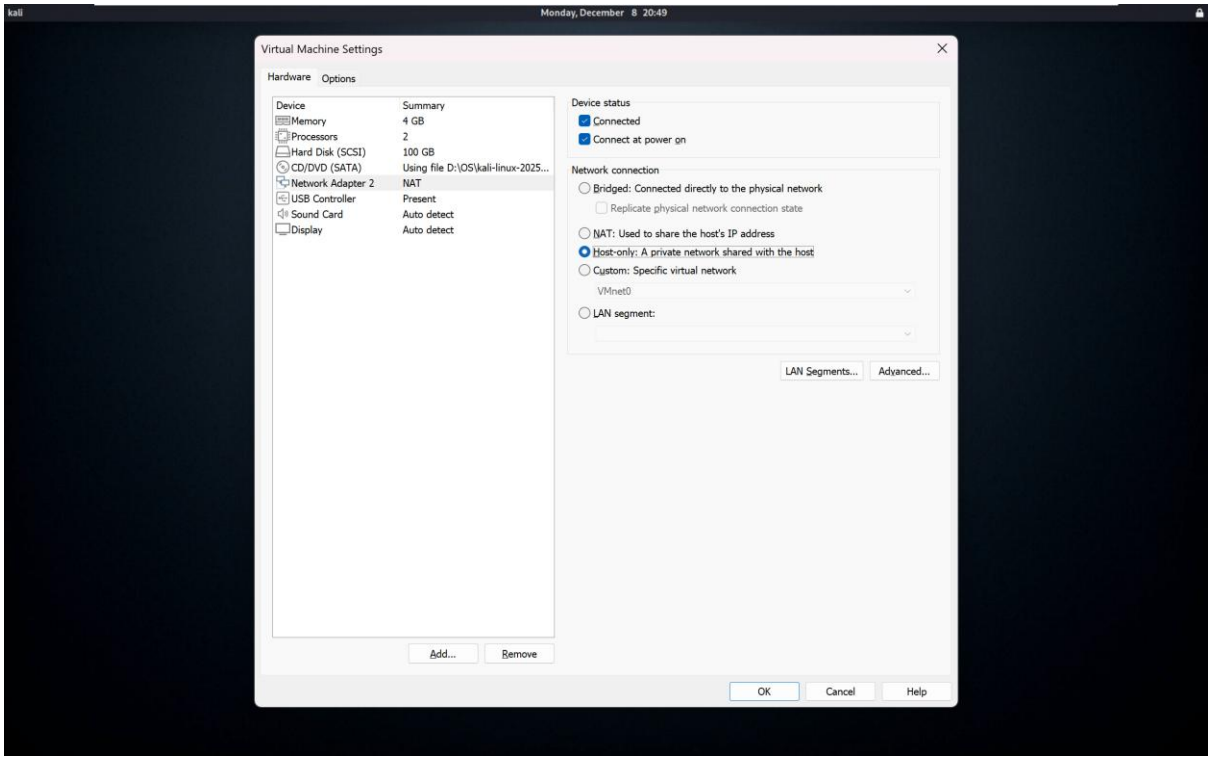

Fig 4. Switching to host only mode

3.2 Disable drag & drop and copy & paste

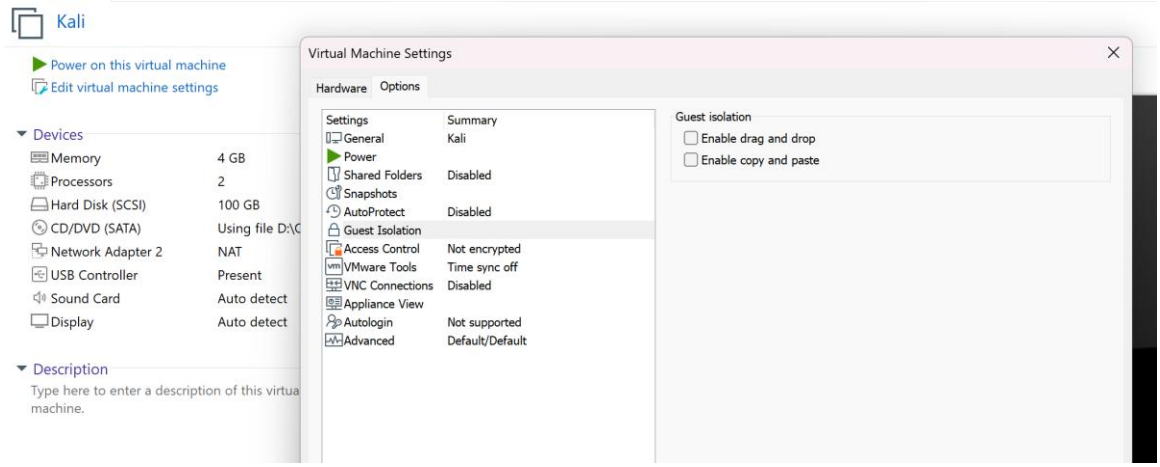

Fig 5. Disabling drag and drop & copy and paste

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.

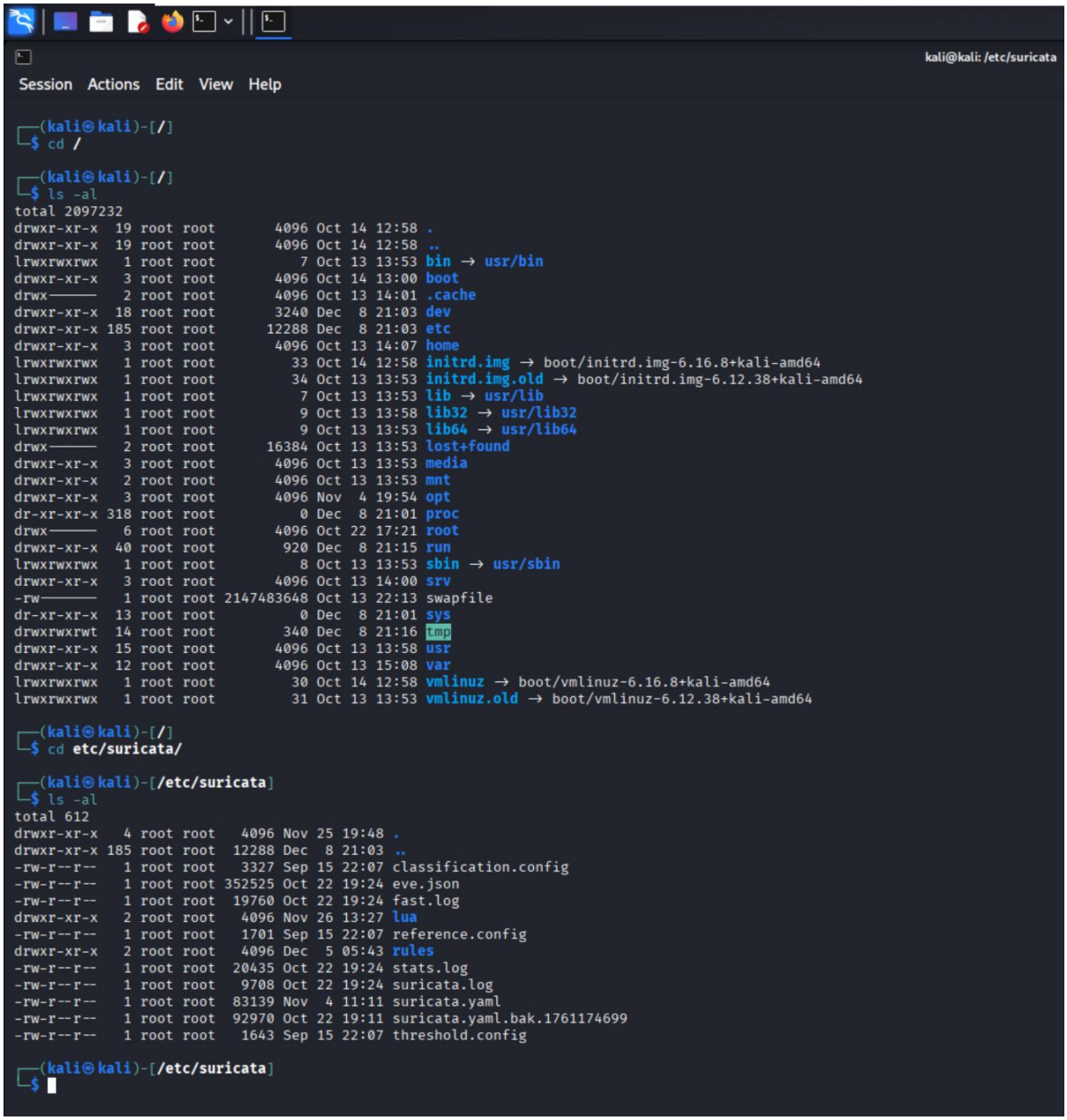

Fig 6. Location of important files

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-proto. 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, sisron, 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 sisron PCAP file is opened in Wireshark, which also shows information about JA3 fingerprint.

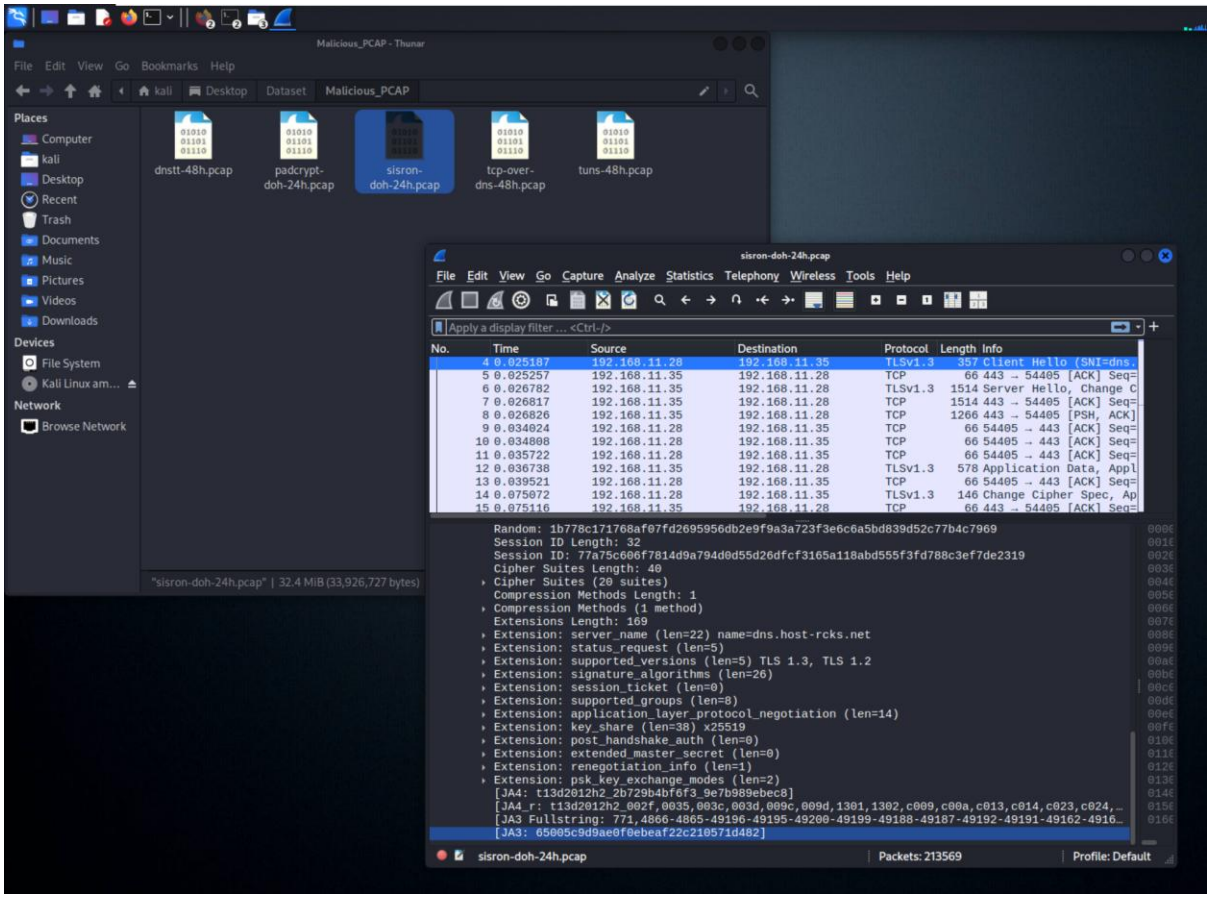

Fig 10. Wireshark Analysis

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.

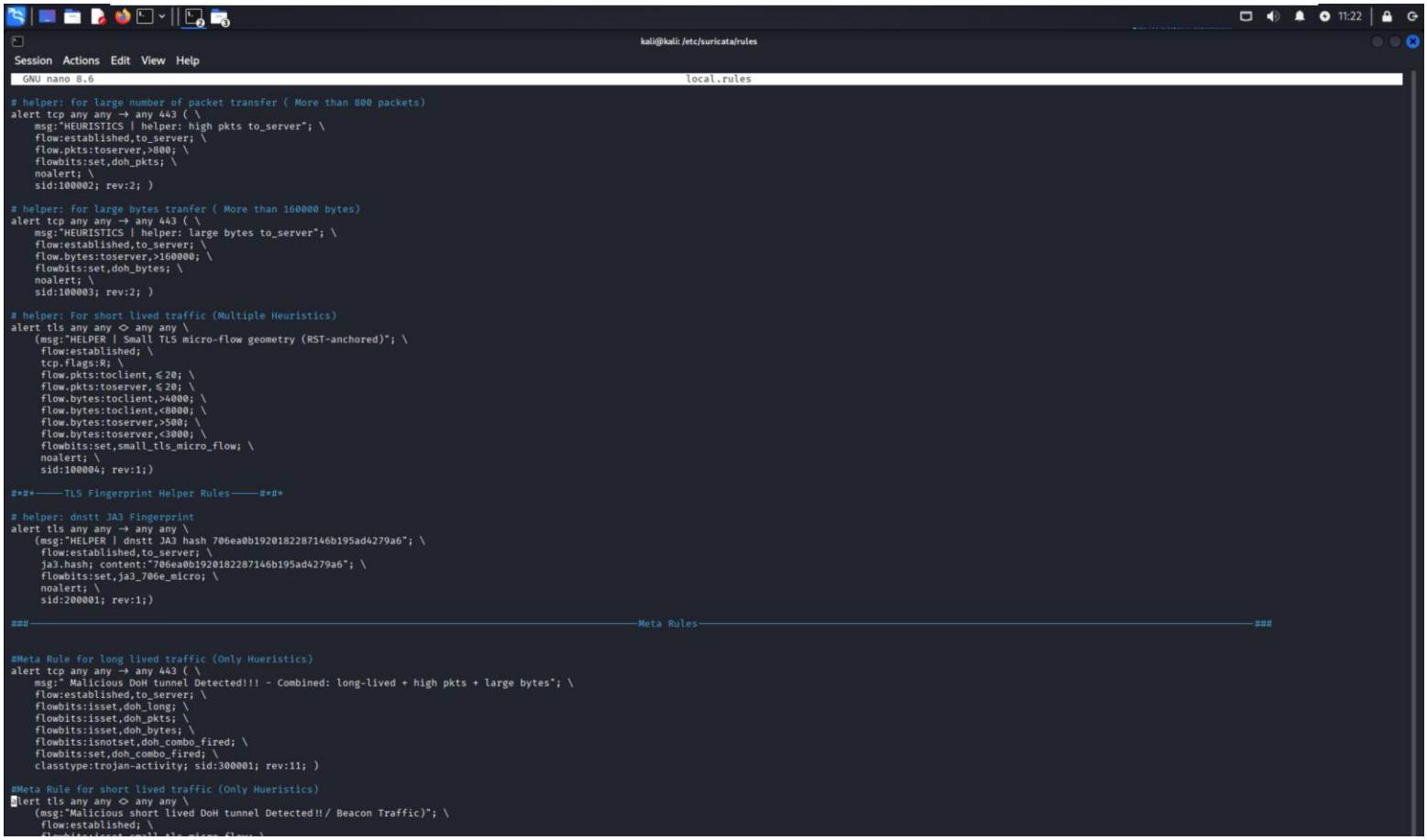

Fig 11. Custom rules configured

```
# 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> <soruce port number> <direction> <destination ip> <destionation port number>

2 - msg:< Write your message>

3 - flow: established, to_server

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

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.

7.1 Information extraction from a PCAP file, before feeding it into Suricata.

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 sisron-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>

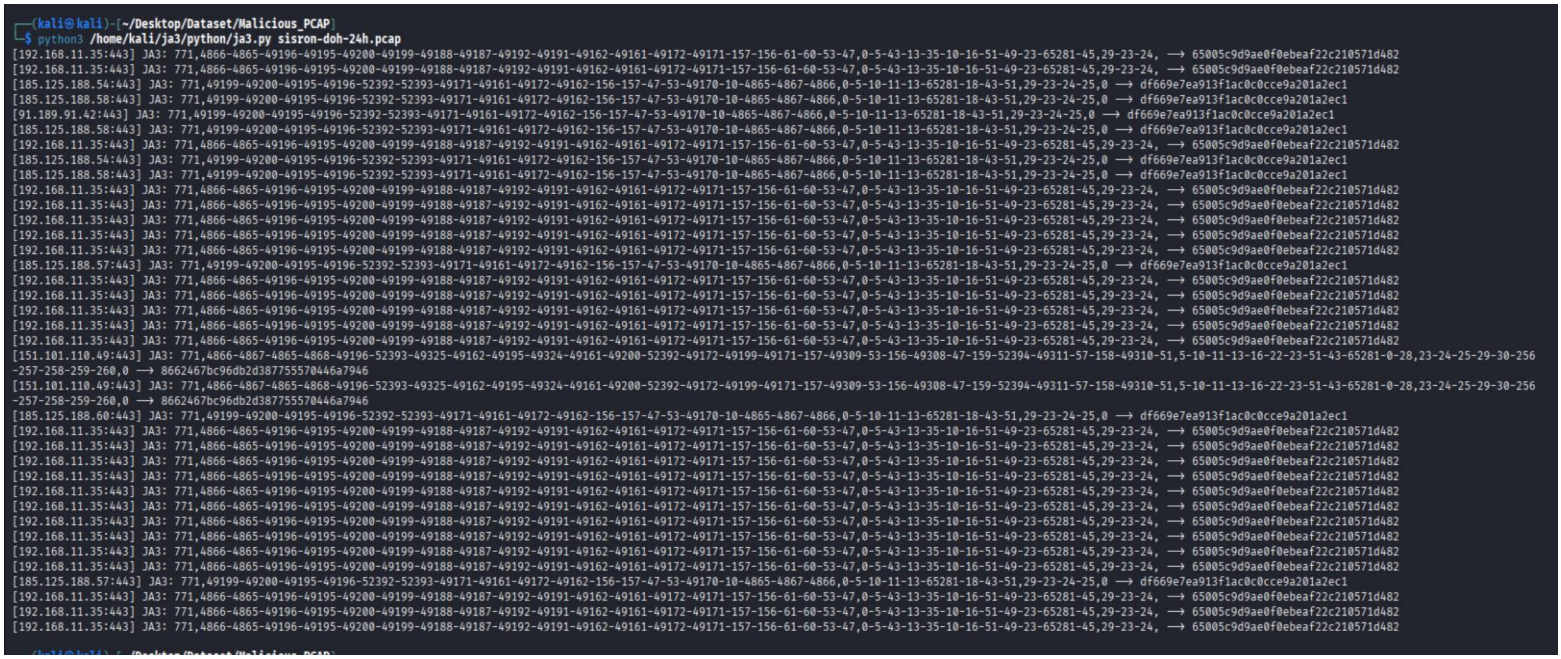

Fig 15. Extraction of JA3 fingerprint using Python

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>**

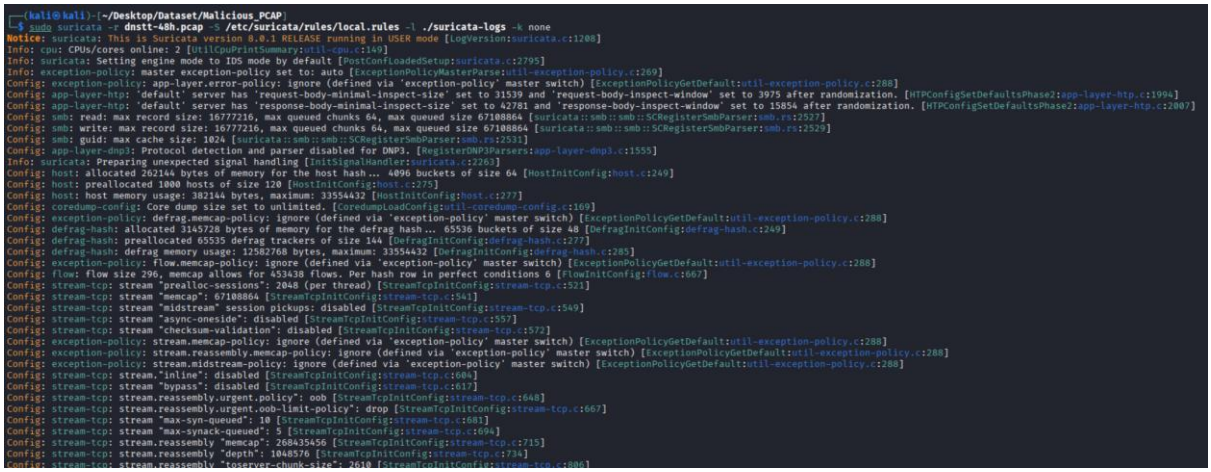

Fig 16. Processing of PCAP file through Suricata

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

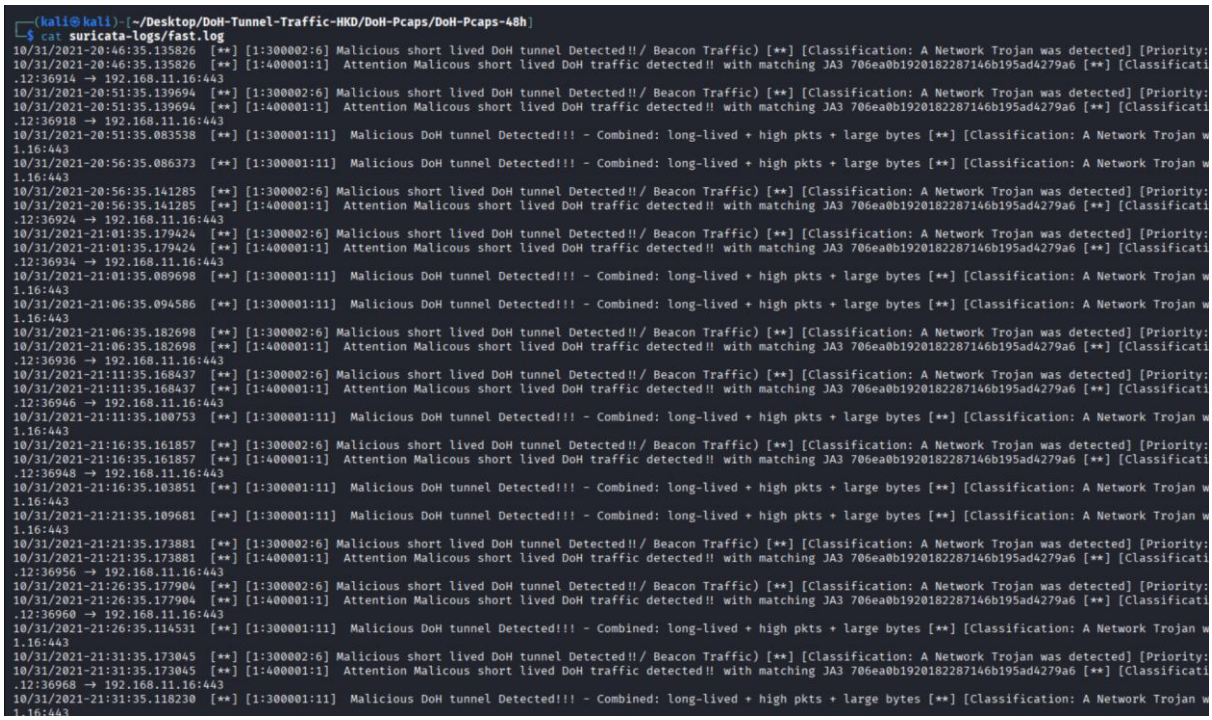

Fig 17. Example of alerts generated by the model

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

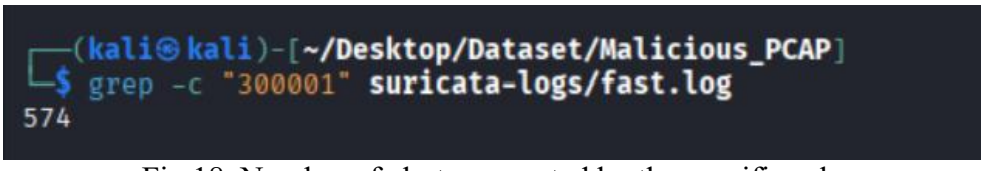

Fig 18. Number of alerts generated by the specific rule

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 .