

“Intrusion Detection & Prevention System using Suricata”

By

Mukul Kumar	18BEC1197
Vishnu Puligadda	18BEC1345
Nikhil Anand	18BEC1346
Ashish Tiwari	18BEC1221
Suryansh Sohgaure	18BEC1333

A project report submitted to

Dr. S. REVATHI

SCHOOL OF ELECTRONICS ENGINEERING

In partial fulfilment of the requirements for the course of

CSE3502 – INFORMATION SECURITY MANAGEMENT

in

B. Tech. ELECTRONICS AND COMMUNICATION

ENGINEERING

Vandalur – Kelambakkam Road

CHENNAI - 600127

June 2021

BONAFIDE CERTIFICATE

This is to certify that the Project work titled “**Intrusion Detection and Prevention System using Suricata**” is being submitted by **Mukul Kumar (18BEC1197), Vishnu Puligadda (18BEC1345), Nikhil Anand (18BEC1346), Ashish Tiwari (18BEC1221), and Suryansh Sohgaura (18BEC1333)** as a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University.

Dr. S. REVATHI

Associate Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai-600127

TABLE OF CONTENT

SERIAL NO.	CONTENT	PAGE NO.
01	Abstract	4
02	Acknowledgement	5
03	Introduction to Project & Suricata	6-11
04	Suricata Installation	12-22
05	Suricata Configuration as IDS/IPS	22-24
06	Customized Rule Setup	24-26
07	IDS Testing based upon Inbuilt Rulesets & Customized Rulesets	26-32
08	IPS Testing based upon Inbuilt Rulesets	32-33
09	Machine Learning Approach to build IDS/IPS - Introduction	34
10	Machine Learning Model Dataset Description	35-36
10	Machine Learning Model Result Discussion	37
11	Overall Project Result Discussion & Conclusion	38
12	Team member's Biodata	39

ABSTRACT

With the thriving technology and the great increase in the usage of computer networks, the risk of having these network to be under attacks have been increased. Number of techniques have been created and designed to help in detecting and/or preventing such attacks. One common technique is the use of Network Intrusion Detection / Prevention Systems NIDS. Today, number of open sources and commercial Intrusion Detection Systems are available to match enterprises requirements but the performance and usage difficulty of these Intrusion Detection Systems is still the main concern.

In this project, to overcome the performance issue, we have used Suricata a high performance network security monitoring engine and designed a host based IDS and IPS system, deployed it on Ubuntu server and to overcome usage difficulty issue, we have shown each and every step in detail from Suricata installation to default or customized ruleset based testing.

Also, deploying IDS system is not enough as it is passive monitoring solution for detecting cybersecurity threats to an organization. To avoid IDS vs IPS problem we have deployed both the solutions to protect the organization assets and servers.

Instead of using open source commercial tools for Intrusion detection & prevention, we have also developed a Machine Learning model to overcome false positive issue in these tools. Our model test accuracy is 99.969% which means we have reduced false positive issue up to a great extent.

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. Sivasubramanian A**, Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Dr. Vetrivelan P** for his support throughout the course of this project.

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. S. Revathi**, Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

INTRODUCTION - Project

Intruders are always ready and determined to cause data breaches, install malware, and steal sensitive information. Research shows that websites are hit with 22 cyber-attacks in a day, on average. Embracing cyber threat prevention and detection technologies is significant to mitigating threats. Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS) are two technologies used in threat protection.

Through this project we have deployed an IDS and IPS solution deployed it into our network using a high performance open source network threat detection engine called Suricata. And verified its working by performing different types of attacks also configured the system to work on user defined rulesets as well.

Also, in this project we have developed a machine learning model to overcome false positive issue, and can be easily recognized in network threat detection tools. We have compared the result of several classifier machine learning algorithms and finally able to achieve testing accuracy of 99.969%.

INTRODUCTION - Suricata

Suricata is a free and open source, mature, fast and robust network threat detection engine. The Suricata engine is capable of real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing. Suricata inspects the network traffic using a powerful and extensive rules and signature language, and has powerful Lua scripting support for detection of complex threats.

The Suricata project and code is owned and supported by the Open Information Security Foundation (OISF), a non-profit foundation committed to ensuring Suricata's development and sustained success as an open source project.

WORKING OF SURICATA

Suricata works by getting one packet at a time from the system. These are then pre-processed, after which they are passed to the detection engine. Suricata can use pcap for this in IDS mode, but can also connect to a special feature of Linux, named `nfnetlink_queue`. This queue is a way for the kernel to copy a packet to a userspace process (in this case Suricata). Then it waits for this userspace process to issue a verdict on this packet, with 'accept' and 'drop' being the two possible verdicts.

Suricata works with rules. These rules can have actions like 'alert', 'log', etc. Suricata IPS introduces three new actions specific to IPS mode, 'drop', 'sdrop' and 'reject' (sdrop is silent drop). As you read above, the only two possible verdict for a packet are 'accept' and 'drop', so how can Suricata reject a packet? This is done in two steps-

1. the packet is dropped using the 'drop' verdict.
2. Suricata sends an icmp error or tcp-reset to the offending ipaddress.

COMPANIES USING SURICATA

Who uses Suricata?

Some of the companies that use Suricata include:

Company	Website	Country	Revenue	Company Size
Code42	code42.com	United States	50M-100M	200-500
City of Seattle	seattle.gov	United States	>1000M	>10000
Carnegie Mellon University	cmu.edu	United States	>1000M	>10000
Hewlett Packard Enterprise Company	hpe.com	United States	>1000M	>10000

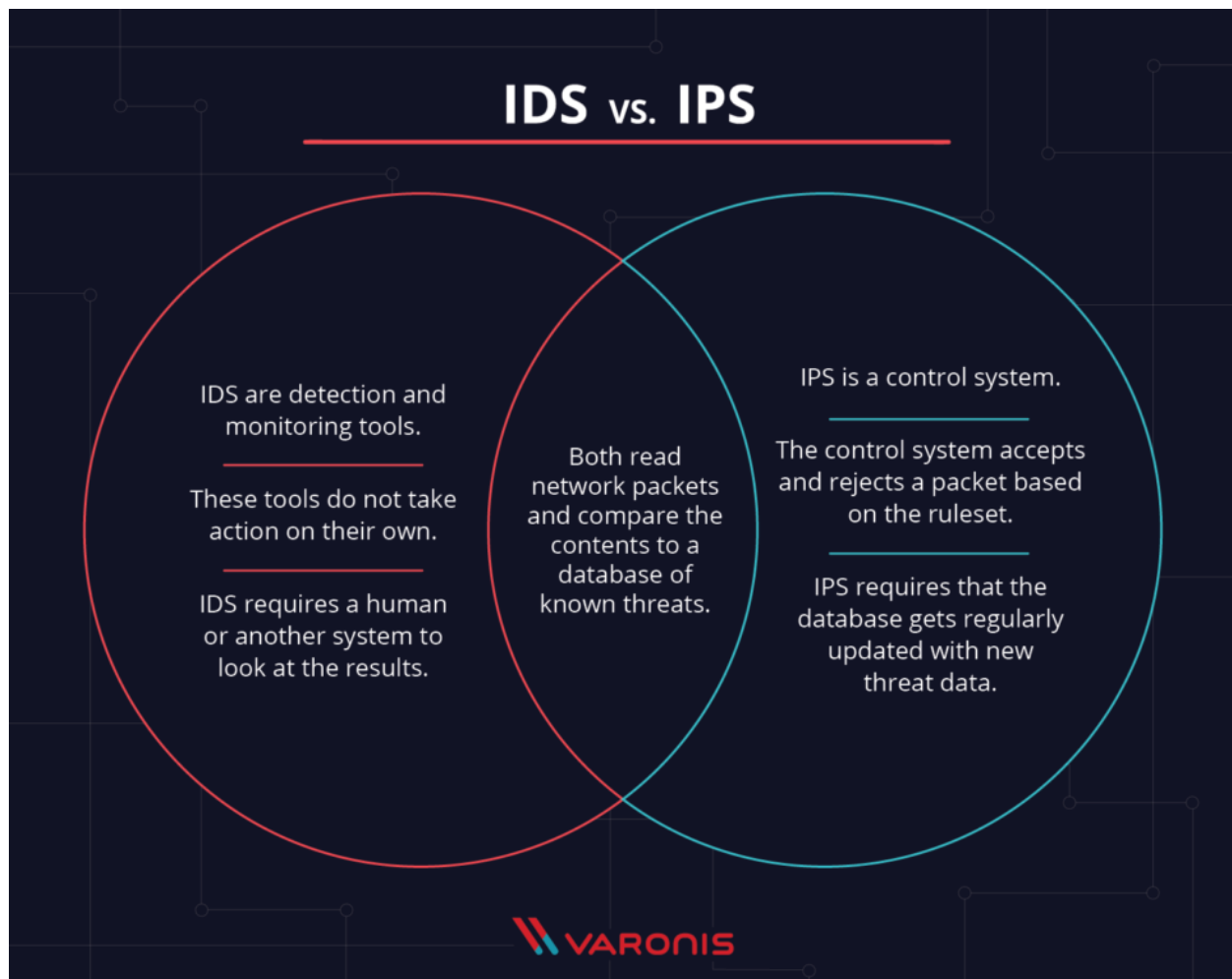
Intrusion Detection Systems (IDS): analyze and monitor network traffic for signs that indicate attackers are using a known cyber threat to infiltrate or steal data from your network. IDS systems compare the current network activity to a known threat database to detect several kinds of behaviors like security policy violations, malware, and port scanners.

Intrusion Prevention Systems (IPS): live in the same area of the network as a firewall, between the outside world and the internal network. IPS proactively deny network traffic based on a security profile if that packet represents a known security threat.

DIFFERENCE BETWEEN IDS and IPS

The main difference between them is that IDS is a monitoring system, while IPS is a control system.

IDS doesn't alter the network packets in any way, whereas IPS prevents the packet from delivery based on the contents of the packet, much like how a firewall prevents traffic by IP address.



Four main types of IPS exist:

1. **Network:** Analyze and protect traffic on your network.
2. **Wireless:** Observe anything happening within a wireless network and defend against an attack launched from there.
3. **Network behavior:** Spot attacks that involve unusual traffic on your network.

4. **Host-based:** Scan events that occur within a host you specify.

Five main types of IDS exist.

1. **Network:** Choose a point on your network and examine all traffic on all devices from that point.
2. **Host:** Examine traffic to and from independent devices within your network, and leave all other devices alone.
3. **Protocol-based:** Place protection between a device and the server, and monitor all traffic that goes between them.
4. **Application protocol-based:** Place protection within a group of servers and watch how they communicate with one another.
5. **Hybrid:** Combine some of the approaches listed above into a system made just for you.

Both IDS/IPS read network packets and compare the contents to a database of known threats. The primary difference between them is what happens next. IDS are detection and monitoring tools that don't take action on their own. IPS is a control system that accepts or rejects a packet based on the ruleset.

IDS requires a human or another system to look at the results and determine what actions to take next, which could be a full time job depending on the amount of network traffic generated each day. IDS makes a better post-mortem forensics tool for the CSIRT to use as part of their security incident investigations.

The purpose of the IPS, on the other hand, is to catch dangerous packets and drop them before they reach their target. It's more passive than an IDS, simply requiring that the database gets regularly updated with new threat data.

It plays a very significant role in Cybersecurity system-

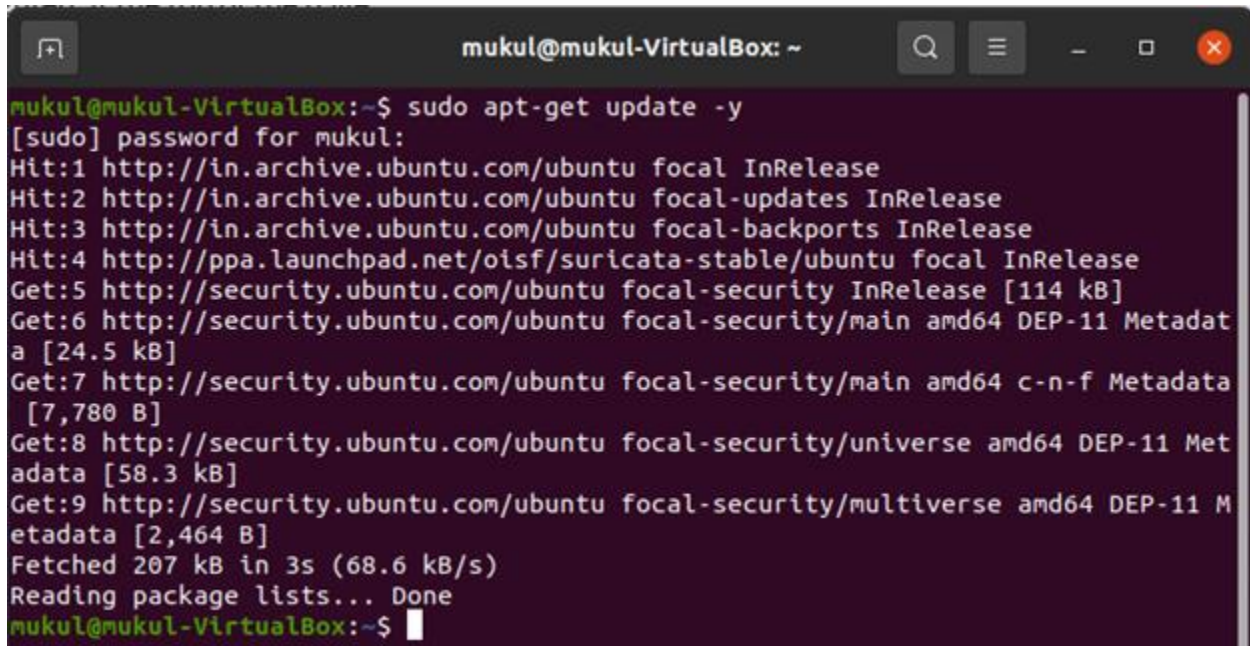
Automation: IDS/IPS systems are largely hands-off, which makes them ideal candidates for use in the current security stack. IPS provides the peace of mind that the network is protected from known threats with limited resource requirements.

Compliance: Part of compliance often requires proving that you have invested in technologies and systems to protect data. Implementing an IDS/IPS solution checks off a box on the compliance sheet and addresses a number of the CIS Security controls. More importantly, the auditing data is a valuable part of compliance investigations.

Policy enforcement: IDS/IPS are configurable to help enforce internal security policies at the network level. For example, if you only support one VPN, you can use the IPS to block other VPN traffic.

SURICATA INSTALLATION

sudo apt-get update -y - command to update the base system with the latest available packages



```
mukul@mukul-VirtualBox: ~  
mukul@mukul-VirtualBox:~$ sudo apt-get update -y  
[sudo] password for mukul:  
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease  
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease  
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease  
Hit:4 http://ppa.launchpad.net/oisf/suricata-stable/ubuntu focal InRelease  
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]  
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [24.5 kB]  
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [7,780 B]  
Get:8 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [58.3 kB]  
Get:9 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]  
Fetched 207 kB in 3s (68.6 kB/s)  
Reading package lists... Done  
mukul@mukul-VirtualBox:~$
```

sudo apt-get upgrade - command is used to download package information from all configured sources

apt-get install rustc cargo make libpcrc3 libpcrc3-dbg libpcrc3-dev build-essential autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-config -y - to install required dependencies

Dependencies used for installation of suricata-

libjansson

Jansson is a C library for encoding, decoding and manipulating JSON data.

It features:

- * Simple and intuitive API and data model
- * Comprehensive documentation
- * No dependencies on other libraries
- * Full Unicode support (UTF-8)
- * Extensive test suite

This package contains the development files for jansson.

libpcap

Python libpcap module is a low-level binding for *libpcap* C library.

It is an effort to allow python programs full access to the API provided by the well known *libpcap* Unix C library and by its implementations provided under Win32 systems by such packet capture systems as: Npcap, WinPcap

libpcap is a lightweight Python package, based on the *ctypes* library.

It is fully compliant implementation of the original C *libpcap* from 1.0.0 up to 1.9.0 API and the *WinPcap*'s 4.1.3 libpcap (1.0.0rel0b) API by implementing whole its functionality in a clean Python instead of C.

Libpcre3

This is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language.

New packages should use the newer pcre2 packages, and existing packages should migrate to pcre2.

This package contains the development files, including headers, static libraries, and documentation.

Libnet

libnet provides a portable framework for low-level network packet writing and handling.

libnet features portable packet creation interfaces at the IP layer and link layer, as well as a host of supplementary functionality.

Using libnet, quick and simple packet assembly applications can be whipped up with little effort. With a bit more time, more complex programs can be written (Traceroute and ping were easily rewritten using libnet and libpcap).

This package contains the shared library.

Libyaml

LibYAML covers *presenting* and *parsing* processes. Thus LibYAML defines the following two processors:

- *Parser*, which takes an input stream of bytes and produces a sequence of parsing events.
- *Emitter*, which takes a sequence of events and produces a stream of bytes.

The processes of parsing and presenting are inverse to each other. Any sequence of events produced by parsing a well-formed YAML document should be acceptable by the Emitter, which should produce an equivalent document. Similarly, any document produced by emitting a sequence of events should be acceptable for the Parser, which should produce an equivalent sequence of events.

The job of *resolving* implicit tags, *composing* and *serializing* representation trees, as well as *constructing* and *representing* native objects is left to applications and bindings. Although some of these processes may be covered in the latter releases, they are not in the scope of the initial release of LibYAML.

Libmagic

The File::LibMagic module is a simple perl interface to libmagic from the file package (version 4.x or 5.x). You will need both the library (*libmagic.so*) and the header file (*magic.h*) to build this Perl module.

Zlib

zlib is a library implementing the deflate compression method found in gzip and PKZIP. This package includes the development support files.

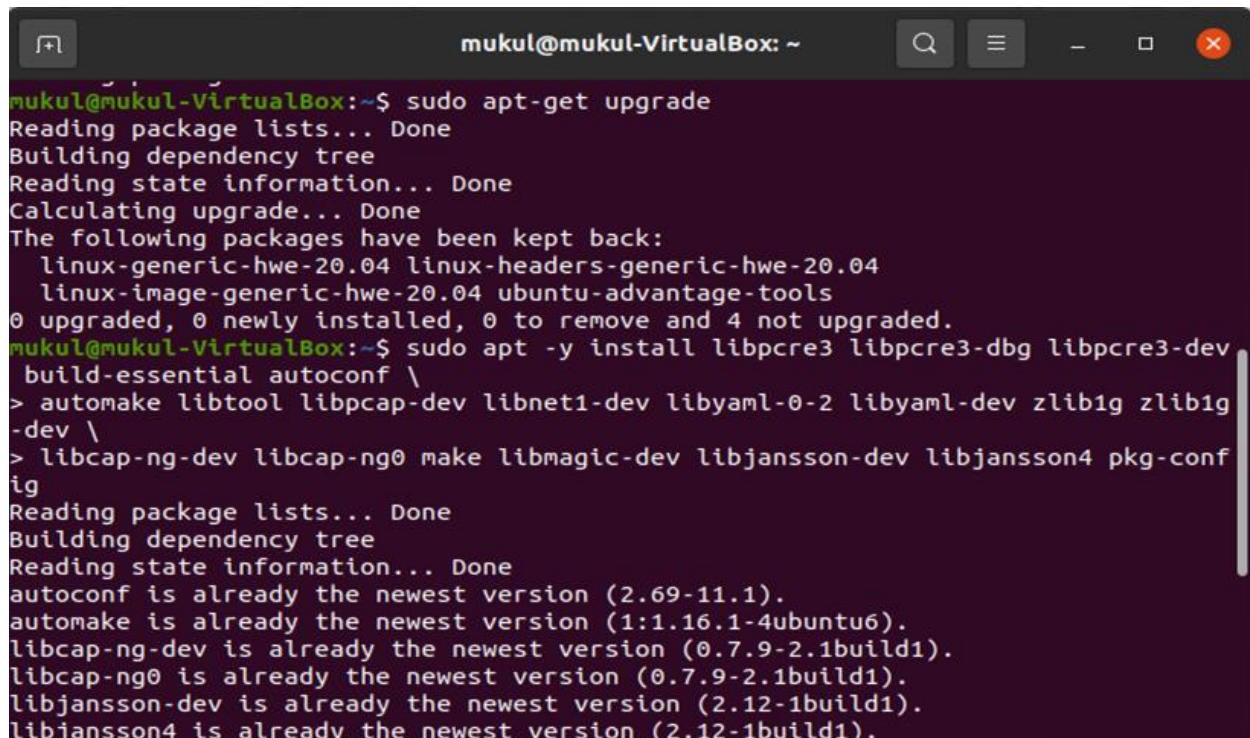
Libnetfilter_queue

libnetfilter_queue is a userspace library providing an API to packets that have been queued by the kernel packet filter. It is part of a system that deprecates the old ip_queue / libipq mechanism.

This package provides development files and static libraries.

libnfnetlink

libnfnetlink is the low-level library for netfilter related kernel/userspace communication. It provides a generic messaging infrastructure for in-kernel netfilter subsystems (such as nfnetlink_log, nfnetlink_queue, nfnetlink_conntrack) and their respective users and/or management tools in userspace.

A terminal window titled 'mukul@mukul-VirtualBox: ~' with standard Ubuntu window controls. The terminal shows the output of 'sudo apt-get upgrade' and 'sudo apt -y install libpcrc3 libpcrc3-dbg libpcrc3-dev build-essential autoconf \> automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev \> libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-config'. The first command lists packages kept back and shows 0 upgrades. The second command lists several packages already at their newest versions.

```
mukul@mukul-VirtualBox:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  linux-generic-hwe-20.04 linux-headers-generic-hwe-20.04
  linux-image-generic-hwe-20.04 ubuntu-advantage-tools
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
mukul@mukul-VirtualBox:~$ sudo apt -y install libpcrc3 libpcrc3-dbg libpcrc3-dev
build-essential autoconf \
> automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g
-dev \
> libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-conf
ig
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version (2.69-11.1).
automake is already the newest version (1:1.16.1-4ubuntu6).
libcap-ng-dev is already the newest version (0.7.9-2.1build1).
libcap-ng0 is already the newest version (0.7.9-2.1build1).
libjansson-dev is already the newest version (2.12-1build1).
libjansson4 is already the newest version (2.12-1build1).
```

By default, Suricata functions as an intrusion detection system (IDS). If we want to include intrusion prevention system (IPS) functionality, then we will need to install some more packages in our system.

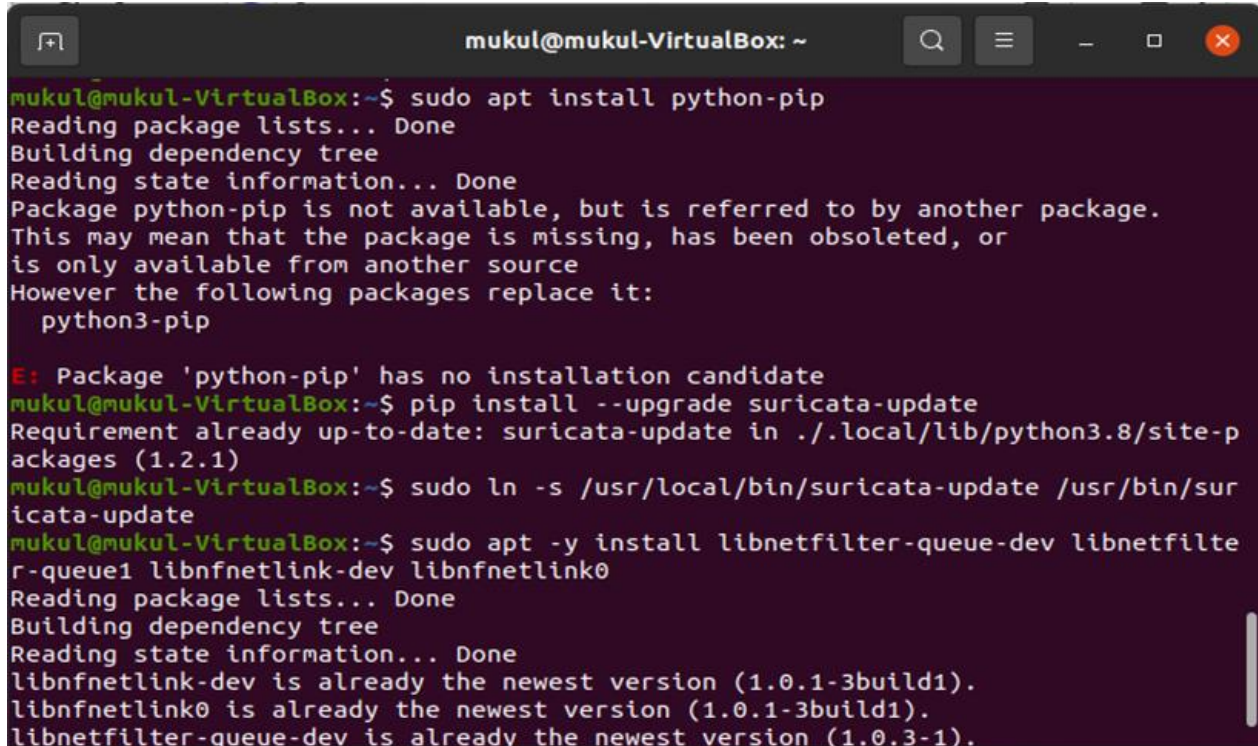
apt-get install libnetfilter-queue-dev libnetfilter-queue1 libnfnetlink-dev libnfnetlink0 -y - to install required dependencies for IPS functionality

Once all the packages are installed, we will need to install the suricata-update tool to update the Suricata rules.

apt-get install python3-pip - to install pip for Python 3 and all the dependencies for building Python modules

pip3 install --upgrade suricata-update – to globally making it available to all users or it can install suricata-update into your home directory for use by your user

ln -s /usr/local/bin/suricata-update /usr/bin/suricata-update – to update the suricata



```
mukul@mukul-VirtualBox: ~  
mukul@mukul-VirtualBox:~$ sudo apt install python-pip  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Package python-pip is not available, but is referred to by another package.  
This may mean that the package is missing, has been obsoleted, or  
is only available from another source  
However the following packages replace it:  
python3-pip  
  
E: Package 'python-pip' has no installation candidate  
mukul@mukul-VirtualBox:~$ pip install --upgrade suricata-update  
Requirement already up-to-date: suricata-update in ./local/lib/python3.8/site-p  
ackages (1.2.1)  
mukul@mukul-VirtualBox:~$ sudo ln -s /usr/local/bin/suricata-update /usr/bin/sur  
icata-update  
mukul@mukul-VirtualBox:~$ sudo apt -y install libnetfilter-queue-dev libnetfilte  
r-queue1 libnfnetlink-dev libnfnetlink0  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
libnfnetlink-dev is already the newest version (1.0.1-3build1).  
libnfnetlink0 is already the newest version (1.0.1-3build1).  
libnetfilter-queue-dev is already the newest version (1.0.3-1).
```

Install Suricata

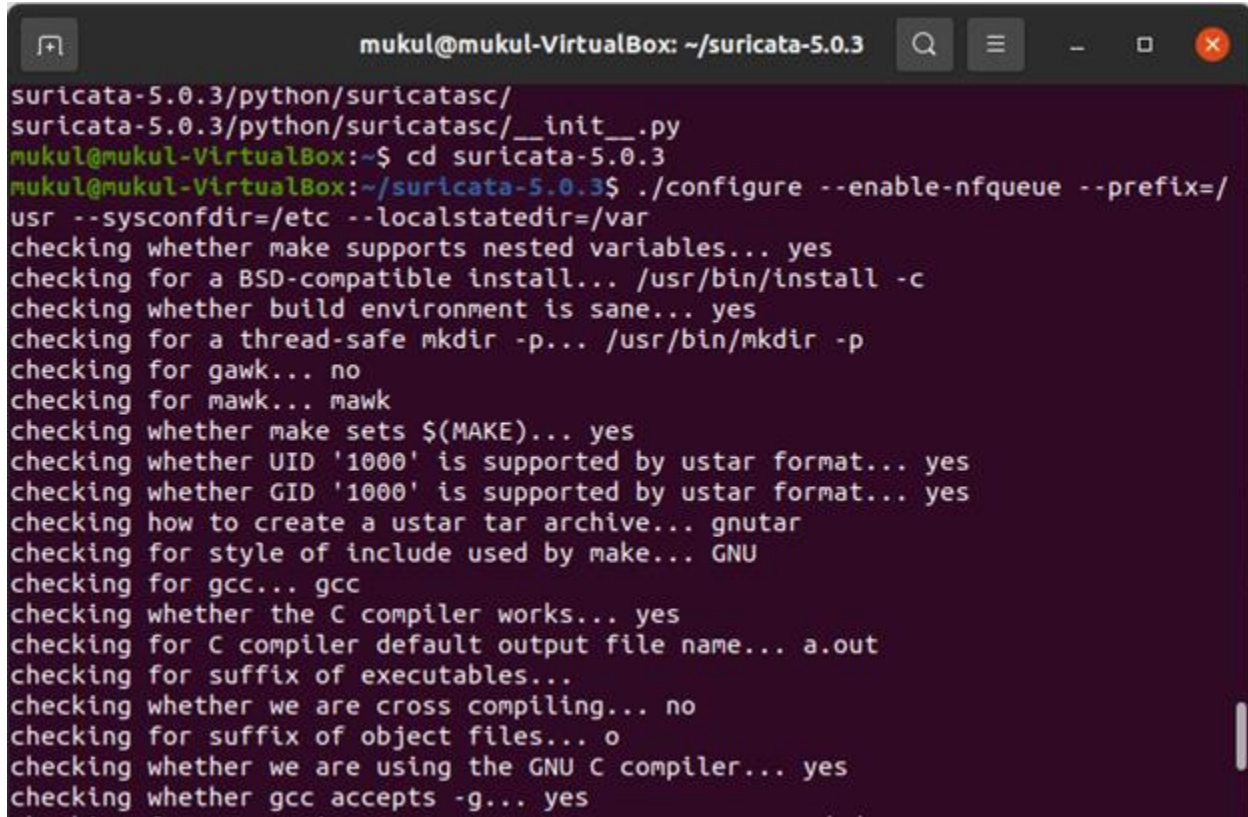
wget <https://www.openinfosecfoundation.org/download/suricata-5.0.3.tar.gz> - to download the latest version of Suricata from their official website

tar -xvzf suricata-5.0.3.tar.gz -to extract the downloaded file

```
mukul@mukul-VirtualBox: ~  
mukul@mukul-VirtualBox:~$ wget https://www.openinfosecfoundation.org/download/suricata-5.0.3.tar.gz  
--2021-06-03 18:12:40-- https://www.openinfosecfoundation.org/download/suricata-5.0.3.tar.gz  
Resolving www.openinfosecfoundation.org (www.openinfosecfoundation.org)... 52.14.249.179, 2600:1f16:db2:4f00:da9d:37d6:e8b9:9802  
Connecting to www.openinfosecfoundation.org (www.openinfosecfoundation.org)|52.14.249.179|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 23744731 (23M) [application/x-gzip]  
Saving to: 'suricata-5.0.3.tar.gz'  
  
suricata-5.0.3.tar. 100%[=====] 22.64M 468KB/s in 53s  
  
2021-06-03 18:13:35 (434 KB/s) - 'suricata-5.0.3.tar.gz' saved [23744731/23744731]  
  
mukul@mukul-VirtualBox:~$ tar -xvzf suricata-5.0.3.tar.gz  
suricata-5.0.3/  
suricata-5.0.3/depcomp  
suricata-5.0.3/configure.ac  
suricata-5.0.3/Makefile.am  
suricata-5.0.3/suricata-update/  
suricata-5.0.3/suricata-update/.travis.yml
```

```
mukul@mukul-VirtualBox: ~  
mukul@mukul-VirtualBox:~$ tar -xvzf suricata-5.0.3.tar.gz  
suricata-5.0.3/  
suricata-5.0.3/depcomp  
suricata-5.0.3/configure.ac  
suricata-5.0.3/Makefile.am  
suricata-5.0.3/suricata-update/  
suricata-5.0.3/suricata-update/.travis.yml  
suricata-5.0.3/suricata-update/Makefile.am  
suricata-5.0.3/suricata-update/requirements.txt  
suricata-5.0.3/suricata-update/CHANGELOG.md  
suricata-5.0.3/suricata-update/tests/  
suricata-5.0.3/suricata-update/tests/classification.config  
suricata-5.0.3/suricata-update/tests/test_util.py  
suricata-5.0.3/suricata-update/tests/emerging.rules.zip  
suricata-5.0.3/suricata-update/tests/sid-msg.map  
suricata-5.0.3/suricata-update/tests/test_rule.py  
suricata-5.0.3/suricata-update/tests/test_main.py  
suricata-5.0.3/suricata-update/tests/test_net.py  
suricata-5.0.3/suricata-update/tests/emerging-current_events.rules  
suricata-5.0.3/suricata-update/tests/emerging.rules.tar.gz.md5  
suricata-5.0.3/suricata-update/tests/docker-ubuntu-1604/  
suricata-5.0.3/suricata-update/tests/docker-ubuntu-1604/Makefile  
suricata-5.0.3/suricata-update/tests/docker-ubuntu-1604/README.md
```


cd suricata-5.0.3 - to change the directory to the extracted directory
./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc --localstatedir=/var - to configure it

A terminal window titled 'mukul@mukul-VirtualBox: ~/suricata-5.0.3' with standard window controls. The terminal shows the following commands and output:

```
suricata-5.0.3/python/suricatasc/  
suricata-5.0.3/python/suricatasc/__init__.py  
mukul@mukul-VirtualBox:~$ cd suricata-5.0.3  
mukul@mukul-VirtualBox:~/suricata-5.0.3$ ./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc --localstatedir=/var  
checking whether make supports nested variables... yes  
checking for a BSD-compatible install... /usr/bin/install -c  
checking whether build environment is sane... yes  
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p  
checking for gawk... no  
checking for mawk... mawk  
checking whether make sets $(MAKE)... yes  
checking whether UID '1000' is supported by ustar format... yes  
checking whether GID '1000' is supported by ustar format... yes  
checking how to create a ustar tar archive... gnutar  
checking for style of include used by make... GNU  
checking for gcc... gcc  
checking whether the C compiler works... yes  
checking for C compiler default output file name... a.out  
checking for suffix of executables...  
checking whether we are cross compiling... no  
checking for suffix of object files... o  
checking whether we are using the GNU C compiler... yes  
checking whether gcc accepts -g... yes
```

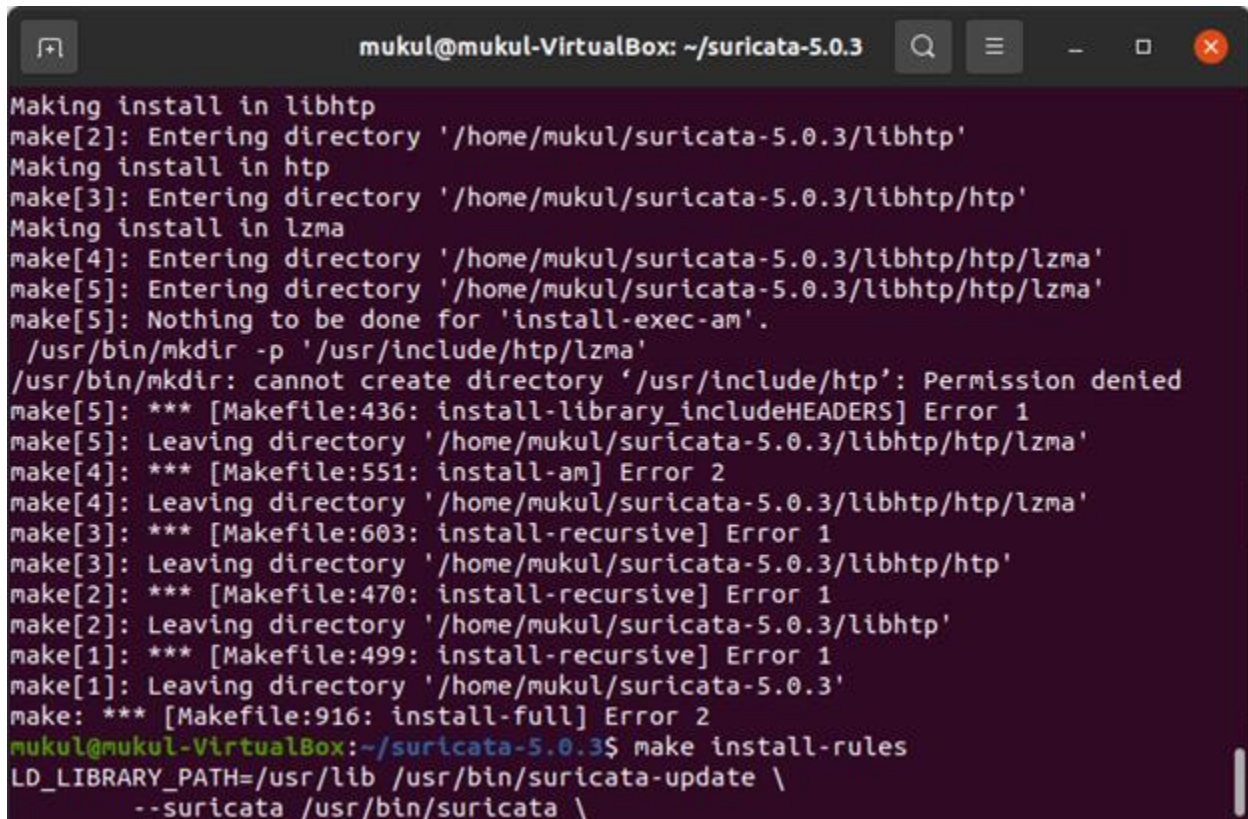
make - to install the suricata

make install-full - to install initial configuration files

```
mukul@mukul-VirtualBox: ~/suricata-5.0.3
copying suricata/update/configs/drop.conf -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
copying suricata/update/configs/disable.conf -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
copying suricata/update/configs/enable.conf -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
copying suricata/update/configs/modify.conf -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
copying suricata/update/configs/update.yaml -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
copying suricata/update/configs/threshold.in -> /home/mukul/suricata-5.0.3/suricata-update/lib/suricata/update/configs
running build_scripts
creating /home/mukul/suricata-5.0.3/suricata-update/scripts-3.8
copying and adjusting bin/suricata-update -> /home/mukul/suricata-5.0.3/suricata-update/scripts-3.8
changing mode of /home/mukul/suricata-5.0.3/suricata-update/scripts-3.8/suricata-update from 664 to 775
make[2]: Leaving directory '/home/mukul/suricata-5.0.3/suricata-update'
make[2]: Entering directory '/home/mukul/suricata-5.0.3'
make[2]: Leaving directory '/home/mukul/suricata-5.0.3'
make[1]: Leaving directory '/home/mukul/suricata-5.0.3'
mukul@mukul-VirtualBox:~/suricata-5.0.3$ make install-full
```

```
mukul@mukul-VirtualBox: ~/suricata-5.0.3
make[2]: Leaving directory '/home/mukul/suricata-5.0.3'
make[1]: Leaving directory '/home/mukul/suricata-5.0.3'
mukul@mukul-VirtualBox:~/suricata-5.0.3$ make install-full
make install
make[1]: Entering directory '/home/mukul/suricata-5.0.3'
Making install in libhttp
make[2]: Entering directory '/home/mukul/suricata-5.0.3/libhttp'
Making install in http
make[3]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http'
Making install in lzma
make[4]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[5]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[5]: Nothing to be done for 'install-exec-am'.
/usr/bin/mkdir -p '/usr/include/http/lzma'
/usr/bin/mkdir: cannot create directory '/usr/include/http': Permission denied
make[5]: *** [Makefile:436: install-library_includeHEADERS] Error 1
make[5]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[4]: *** [Makefile:551: install-am] Error 2
make[4]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[3]: *** [Makefile:603: install-recursive] Error 1
make[3]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http'
make[2]: *** [Makefile:470: install-recursive] Error 1
make[2]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp'
make[1]: *** [Makefile:499: install-recursive] Error 1
```


make install-rules - to install all rules



```
mukul@mukul-VirtualBox: ~/suricata-5.0.3
Making install in libhttp
make[2]: Entering directory '/home/mukul/suricata-5.0.3/libhttp'
Making install in http
make[3]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http'
Making install in lzma
make[4]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[5]: Entering directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[5]: Nothing to be done for 'install-exec-am'.
/usr/bin/mkdir -p '/usr/include/http/lzma'
/usr/bin/mkdir: cannot create directory '/usr/include/http': Permission denied
make[5]: *** [Makefile:436: install-library_includeHEADERS] Error 1
make[5]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[4]: *** [Makefile:551: install-am] Error 2
make[4]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http/lzma'
make[3]: *** [Makefile:603: install-recursive] Error 1
make[3]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp/http'
make[2]: *** [Makefile:470: install-recursive] Error 1
make[2]: Leaving directory '/home/mukul/suricata-5.0.3/libhttp'
make[1]: *** [Makefile:499: install-recursive] Error 1
make[1]: Leaving directory '/home/mukul/suricata-5.0.3'
make: *** [Makefile:916: install-full] Error 2
mukul@mukul-VirtualBox:~/suricata-5.0.3$ make install-rules
LD_LIBRARY_PATH=/usr/lib /usr/bin/suricata-update \
--suricata /usr/bin/suricata \
```

Note: By default, all rules are located at /var/lib/suricata/rules/suricata.rules

cat /var/lib/suricata/rules/suricata.rules - to see all rules

```
mukul@mukul-VirtualBox: ~/suricata-5.0.3
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET WEB_SPECIFIC_APPS Boonex Dolphin explain Parameter Cross Site Scripting Attempt"; flow:established,to_server; http.uri; content: "/explain.php?"; nocase; content: "explain"; nocase; pcre: "/explain\x3d\.(script|onmouse[a-z]+|onkey[a-z]+|onload|onunload|ondragdrop|onblur|onfocus|onclick|ondblclick|onsubmit|onreset|onselect|onchange|style\x3d)/i"; reference: url, packetstormsecurity.org/files/view/98408/Dolphin7.0.4-xss.txt; reference: bugtraq, 46337; classtype: web-application-attack; sid: 2012370; rev: 5; metadata: affected_product Web_Server_Applications, attack_target Web_Server, created_at 2011_02_24, deployment Datacenter, signature_severity Major, tag XSS, tag Cross_Site_Scripting, updated_at 2020_09_13;)
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET WEB_SPECIFIC_APPS Boonex Dolphin relocate Parameter Cross Site Scripting Attempt"; flow:established,to_server; http.uri; content: "/modules/boonex/custom_rss/post_mod_crss.php?"; nocase; content: "relocate"; nocase; pcre: "/relocate\x3d\.(script|onmouse[a-z]+|onkey[a-z]+|onload|onunload|ondragdrop|onblur|onfocus|onclick|ondblclick|onsubmit|onreset|onselect|onchange|style\x3d)/i"; reference: url, packetstormsecurity.org/files/view/98408/Dolphin7.0.4-xss.txt; reference: bugtraq, 46337; classtype: web-application-attack; sid: 2012371; rev: 5; metadata: affected_product Web_Server_Applications, attack_target Web_Server, created_at 2011_02_24, deployment Datacenter, signature_severity Major, tag XSS, tag Cross_Site_Scripting, updated_at 2020_09_13;)
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET WEB_SPECIFIC_APPS ColldUserGroup LibraryID Parameter Blind SQL Injection Attempt"; flow:established,to_server; http.method; content: "GET"; http.uri; content: "/index.cfm?"; nocase; co
```

SURICATA CONFIGURATION

The default Suricata configuration file is located at `/etc/suricata/suricata.yaml`. We will need to configure it to protect our internal network. We can do it by editing the file:

nano /etc/suricata/suricata.yaml -to open suricata.yaml file

```
HOME_NET: "[192.168.1.0/24]"
EXTERNAL_NET: " !$HOME_NET"
```

Note: In the command above, we have replaced 192.168.1.0/24 with our internal network.


```
mukul@mukul-VirtualBox: ~/suricata-5.0.3
mukul@mukul-VirtualBox: ~/suricata-... x mukul@mukul-VirtualBox: ~ x
GNU nano 4.8 /etc/suricata/suricata.yaml Modified
%YAML 1.1
---
# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml
##
## Step 1: inform Suricata about your network
##
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.1.103]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

sudo tail /var/log/suricata/suricata.log – to open suricata log file

```
ashish@ashish-VirtualBox:~$ sudo tail /var/log/suricata/suricata.log
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for mqtt.subscribe.topic
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for mqtt.unsubscribe.top
ic
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for icmpv4.hdr
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for tcp.hdr
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for udp.hdr
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for icmpv6.hdr
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for ipv4.hdr
23/5/2021 -- 20:07:45 - <Perf> - using shared mpm ctx' for ipv6.hdr
23/5/2021 -- 20:07:45 - <Config> - IP reputation disabled
23/5/2021 -- 20:07:45 - <Config> - Loading rule file: /var/lib/suricata/rules/s
uricata.rules
```

sudo suricata -c /etc/suricata/suricata.yaml -I enp0s3 --init-errors-fatal – to start suricata engine in one terminal

```
ashish@ashish-VirtualBox:/etc/suricata$ sudo suricata -c /etc/suricata/suricata
.yaml -i enp0s3 --init-errors-fatal
24/5/2021 -- 10:29:10 - <Notice> - This is Suricata version 6.0.2 RELEASE runni
ng in SYSTEM mode
24/5/2021 -- 10:30:11 - <Notice> - all 1 packet processing threads, 4 managemen
t threads initialized, engine started.
```

tail -n 50 fast.log – to capture only 50 packets

```
ashish@ashish-VirtualBox:/var/log/suricata$ tail -n 50 fast.log
```

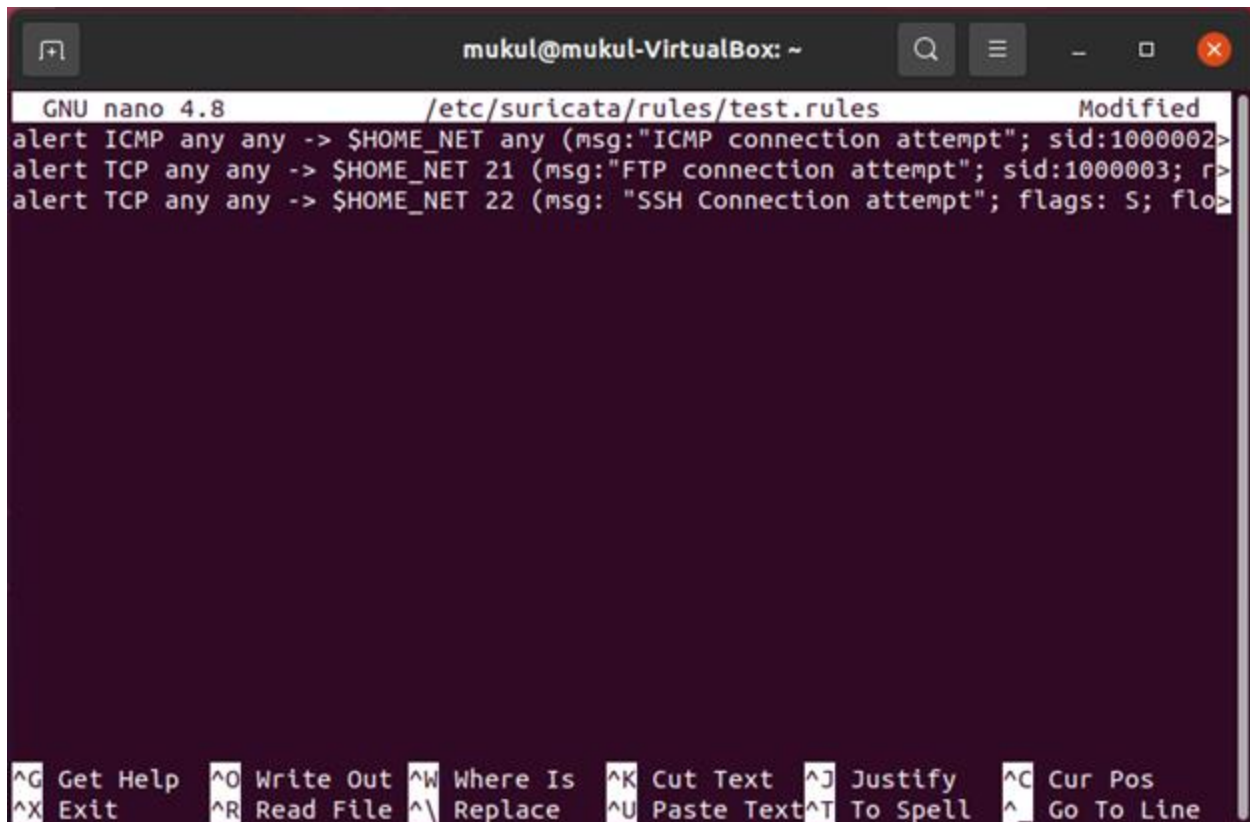
tail -f fast.log – to capture all packet live

```
ashish@ashish-VirtualBox:/var/log/suricata$ tail -f fast.log
```

CUSTOMIZED RULE SETUP

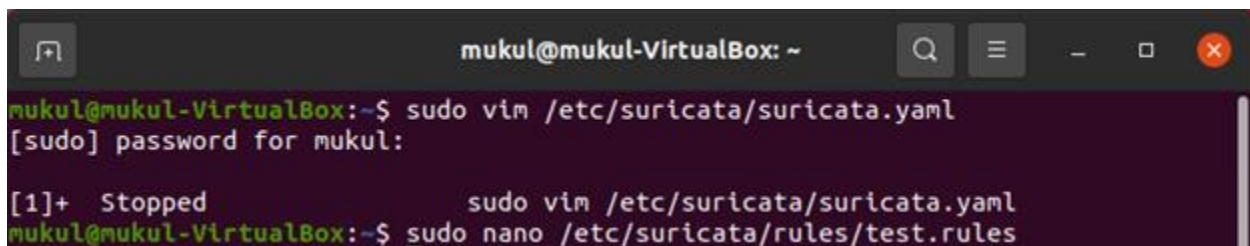
sudo nano /etc/suricata/rules/test.rules –to create a rule file in rules directory with name test.rules

In this test.rules file we have added three types of alert rules for different connection attempts such as **ICMP connection attempt**, **FTP connection attempt** and **SSH connection attempt** in required ruleset format.



```
mukul@mukul-VirtualBox: ~
GNU nano 4.8 /etc/suricata/rules/test.rules Modified
alert ICMP any any -> $HOME_NET any (msg:"ICMP connection attempt"; sid:1000002; r
alert TCP any any -> $HOME_NET 21 (msg:"FTP connection attempt"; sid:1000003; r
alert TCP any any -> $HOME_NET 22 (msg: "SSH Connection attempt"; flags: S; flo
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

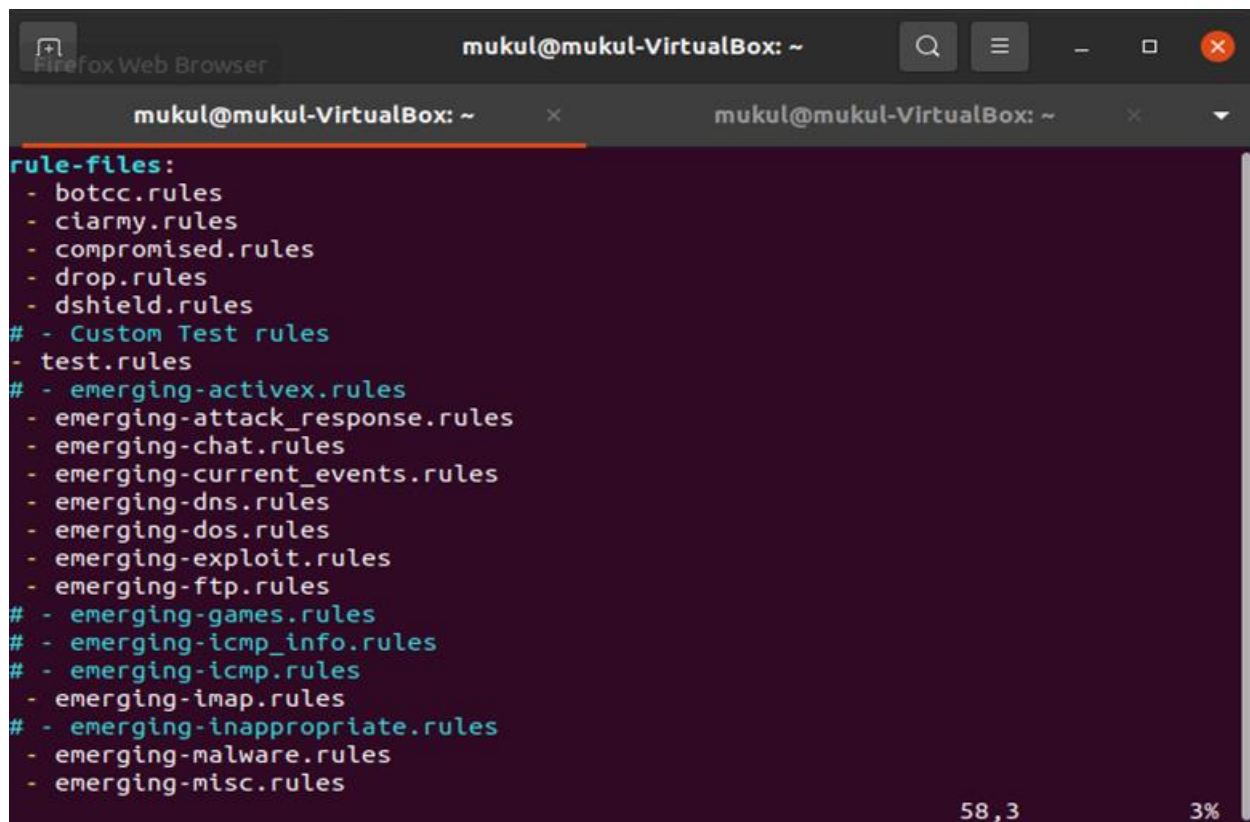
sudo vim /etc/suricata/suricata.yaml – to open suricata configuration file to add test.rules file under rules-files section so that this file to get executed during program compilation



```
mukul@mukul-VirtualBox: ~
mukul@mukul-VirtualBox:~$ sudo vim /etc/suricata/suricata.yaml
[sudo] password for mukul:

[1]+  Stopped                  sudo vim /etc/suricata/suricata.yaml
mukul@mukul-VirtualBox:~$ sudo nano /etc/suricata/rules/test.rules
```

test.rules file got added under **rule-files** section with identifier comment Custom Test Rules



The screenshot shows a terminal window titled 'mukul@mukul-VirtualBox: ~'. The terminal displays a list of rule files under the heading 'rule-files:'. The list includes: botcc.rules, ciarmy.rules, compromised.rules, drop.rules, dshield.rules, Custom Test rules (commented out), test.rules, emerging-activex.rules, emerging-attack_response.rules, emerging-chat.rules, emerging-current_events.rules, emerging-dns.rules, emerging-dos.rules, emerging-exploit.rules, emerging-ftp.rules, emerging-games.rules (commented out), emerging-icmp_info.rules (commented out), emerging-icmp.rules (commented out), emerging-imap.rules, emerging-inappropriate.rules (commented out), emerging-malware.rules, and emerging-misc.rules. The terminal also shows a status bar at the bottom right with '58,3' and '3%'.

```
mukul@mukul-VirtualBox: ~  
rule-files:  
- botcc.rules  
- ciarmy.rules  
- compromised.rules  
- drop.rules  
- dshield.rules  
# - Custom Test rules  
- test.rules  
# - emerging-activex.rules  
- emerging-attack_response.rules  
- emerging-chat.rules  
- emerging-current_events.rules  
- emerging-dns.rules  
- emerging-dos.rules  
- emerging-exploit.rules  
- emerging-ftp.rules  
# - emerging-games.rules  
# - emerging-icmp_info.rules  
# - emerging-icmp.rules  
- emerging-imap.rules  
# - emerging-inappropriate.rules  
- emerging-malware.rules  
- emerging-misc.rules  
58,3 3%
```

IDS TESTING BASED ON DEFAULT RULESETS

RESULT 1

Go to the remote system and perform a simple ping attack test against the Suricata server using the *ping server_ip* command as shown below


```

root@kali:~# ping 192.168.43.42
PING 192.168.43.42 (192.168.43.42) 56(84) bytes of data.
64 bytes from 192.168.43.42: icmp_seq=1 ttl=64 time=0.457 ms
64 bytes from 192.168.43.42: icmp_seq=2 ttl=64 time=0.540 ms
64 bytes from 192.168.43.42: icmp_seq=3 ttl=64 time=0.699 ms
64 bytes from 192.168.43.42: icmp_seq=4 ttl=64 time=0.589 ms
64 bytes from 192.168.43.42: icmp_seq=5 ttl=64 time=0.450 ms
64 bytes from 192.168.43.42: icmp_seq=6 ttl=64 time=0.383 ms
64 bytes from 192.168.43.42: icmp_seq=7 ttl=64 time=0.491 ms
64 bytes from 192.168.43.42: icmp_seq=8 ttl=64 time=0.524 ms
64 bytes from 192.168.43.42: icmp_seq=9 ttl=64 time=0.490 ms
64 bytes from 192.168.43.42: icmp_seq=10 ttl=64 time=0.473 ms
64 bytes from 192.168.43.42: icmp_seq=11 ttl=64 time=0.465 ms
64 bytes from 192.168.43.42: icmp_seq=12 ttl=64 time=0.550 ms
64 bytes from 192.168.43.42: icmp_seq=13 ttl=64 time=0.659 ms
64 bytes from 192.168.43.42: icmp_seq=14 ttl=64 time=0.686 ms
64 bytes from 192.168.43.42: icmp_seq=15 ttl=64 time=0.741 ms
64 bytes from 192.168.43.42: icmp_seq=16 ttl=64 time=0.567 ms
64 bytes from 192.168.43.42: icmp_seq=17 ttl=64 time=0.557 ms
64 bytes from 192.168.43.42: icmp_seq=18 ttl=64 time=0.563 ms
64 bytes from 192.168.43.42: icmp_seq=19 ttl=64 time=0.499 ms
64 bytes from 192.168.43.42: icmp_seq=20 ttl=64 time=0.457 ms

```

LOG FILE: *ICMP_INFO PING classified as Misc activity detected using default ruleset*

```

05/16/2021-18:05:49.691795  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:50.692412  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:51.705617  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:52.704916  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:53.720112  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:54.743761  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:55.744230  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:56.744580  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:57.749623  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:58.750618  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:05:59.750215  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:06:00.750822  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0
05/16/2021-18:06:01.777447  [**] [1:2100366:8] GPL ICMP_INFO PING *NIX [**] [Classification: Misc activ
ty] [Priority: 3] {ICMP} 192.168.43.168:8 -> 192.168.43.42:0

```

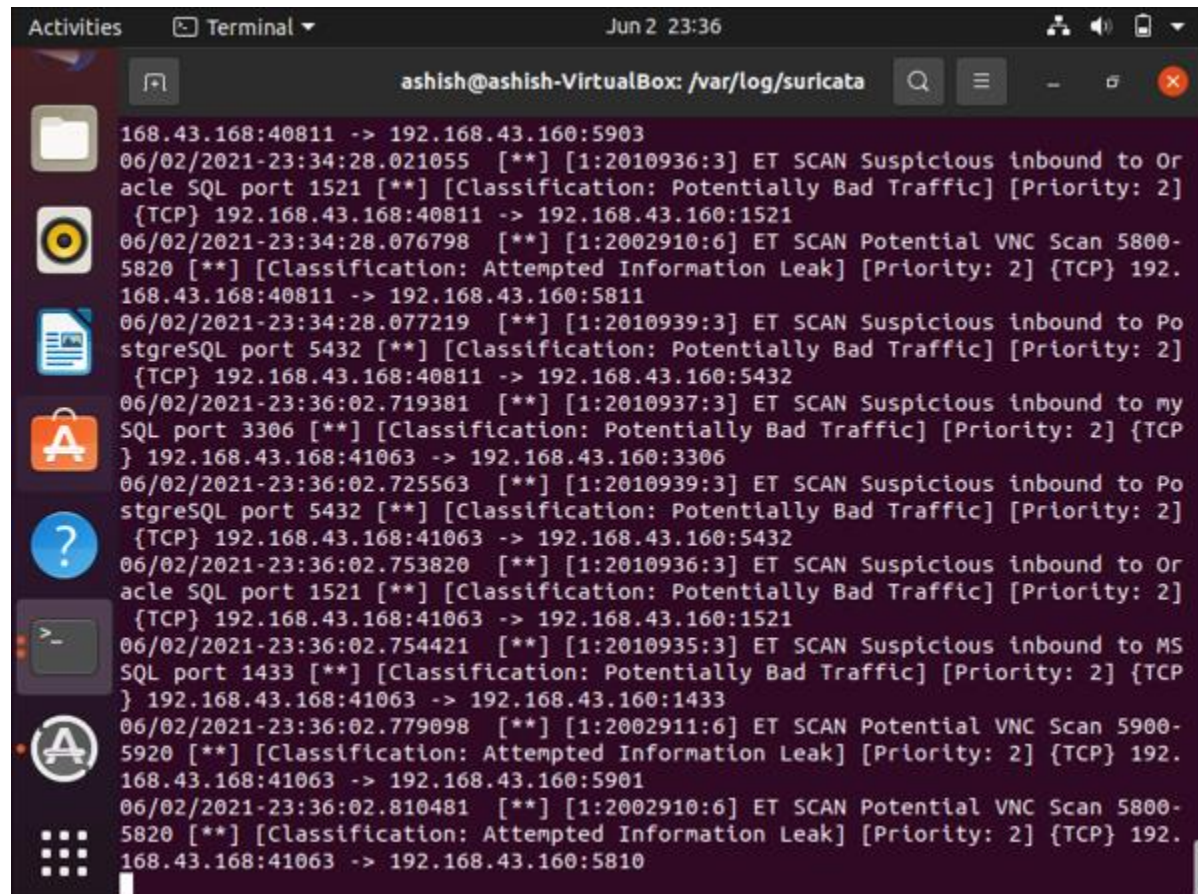

RESULT 2

Go to the remote system and perform a single IP address port scanning using ***nmap -sV server_ip*** command as shown below

```
root@kali:~# nmap -sV 192.168.43.160
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-02 23:36 IST
Nmap scan report for ashish-VirtualBox (192.168.43.160)
Host is up (0.00066s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
23/tcp    open  telnet  Linux telnetd
MAC Address: 08:00:27:43:91:9D (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.55 seconds
```

LOG FILE: *ET Scan Suspicious inbound to Oracle SQL port classified as Potentially Bad Traffic detected using default ruleset*

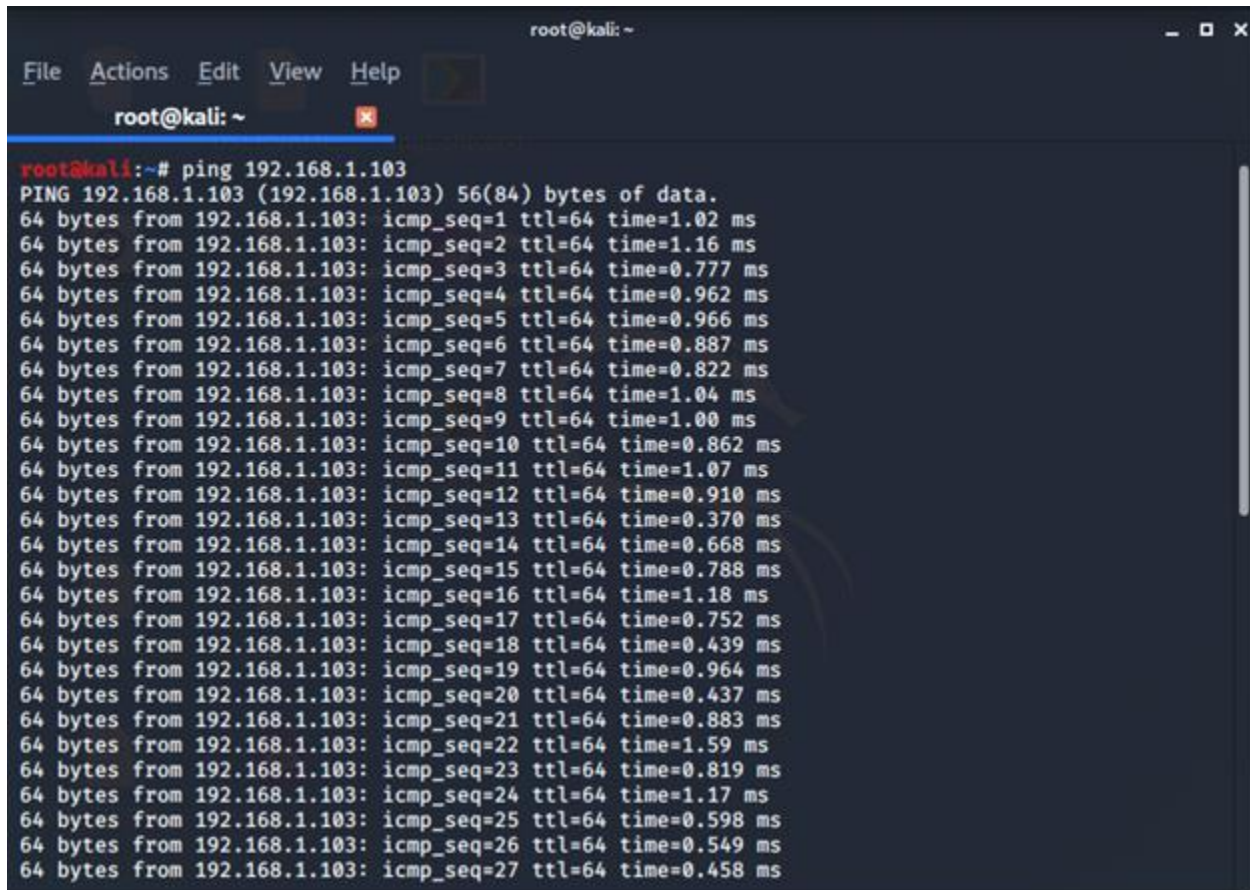


```
ashish@ashish-VirtualBox: /var/log/suricata
168.43.168:40811 -> 192.168.43.160:5903
06/02/2021-23:34:28.021055  [**] [1:2010936:3] ET SCAN Suspicious inbound to Oracle SQL port 1521 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:40811 -> 192.168.43.160:1521
06/02/2021-23:34:28.076798  [**] [1:2002910:6] ET SCAN Potential VNC Scan 5800-5820 [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.43.168:40811 -> 192.168.43.160:5811
06/02/2021-23:34:28.077219  [**] [1:2010939:3] ET SCAN Suspicious inbound to PostgreSQL port 5432 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:40811 -> 192.168.43.160:5432
06/02/2021-23:36:02.719381  [**] [1:2010937:3] ET SCAN Suspicious inbound to my SQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:3306
06/02/2021-23:36:02.725563  [**] [1:2010939:3] ET SCAN Suspicious inbound to PostgreSQL port 5432 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:5432
06/02/2021-23:36:02.753820  [**] [1:2010936:3] ET SCAN Suspicious inbound to Oracle SQL port 1521 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:1521
06/02/2021-23:36:02.754421  [**] [1:2010935:3] ET SCAN Suspicious inbound to MS SQL port 1433 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:1433
06/02/2021-23:36:02.779098  [**] [1:2002911:6] ET SCAN Potential VNC Scan 5900-5920 [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:5901
06/02/2021-23:36:02.810481  [**] [1:2002910:6] ET SCAN Potential VNC Scan 5800-5820 [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.43.168:41063 -> 192.168.43.160:5810
```

IDS TESTING BASED ON CUSTOMIZED RULESETS

RESULT 1

Go to the remote system and perform a simple ping attack test against the Suricata server using the *ping server_ip* command as shown below



```
root@kali: ~  
File Actions Edit View Help  
root@kali: ~  
root@kali:~# ping 192.168.1.103  
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.  
64 bytes from 192.168.1.103: icmp_seq=1 ttl=64 time=1.02 ms  
64 bytes from 192.168.1.103: icmp_seq=2 ttl=64 time=1.16 ms  
64 bytes from 192.168.1.103: icmp_seq=3 ttl=64 time=0.777 ms  
64 bytes from 192.168.1.103: icmp_seq=4 ttl=64 time=0.962 ms  
64 bytes from 192.168.1.103: icmp_seq=5 ttl=64 time=0.966 ms  
64 bytes from 192.168.1.103: icmp_seq=6 ttl=64 time=0.887 ms  
64 bytes from 192.168.1.103: icmp_seq=7 ttl=64 time=0.822 ms  
64 bytes from 192.168.1.103: icmp_seq=8 ttl=64 time=1.04 ms  
64 bytes from 192.168.1.103: icmp_seq=9 ttl=64 time=1.00 ms  
64 bytes from 192.168.1.103: icmp_seq=10 ttl=64 time=0.862 ms  
64 bytes from 192.168.1.103: icmp_seq=11 ttl=64 time=1.07 ms  
64 bytes from 192.168.1.103: icmp_seq=12 ttl=64 time=0.910 ms  
64 bytes from 192.168.1.103: icmp_seq=13 ttl=64 time=0.370 ms  
64 bytes from 192.168.1.103: icmp_seq=14 ttl=64 time=0.668 ms  
64 bytes from 192.168.1.103: icmp_seq=15 ttl=64 time=0.788 ms  
64 bytes from 192.168.1.103: icmp_seq=16 ttl=64 time=1.18 ms  
64 bytes from 192.168.1.103: icmp_seq=17 ttl=64 time=0.752 ms  
64 bytes from 192.168.1.103: icmp_seq=18 ttl=64 time=0.439 ms  
64 bytes from 192.168.1.103: icmp_seq=19 ttl=64 time=0.964 ms  
64 bytes from 192.168.1.103: icmp_seq=20 ttl=64 time=0.437 ms  
64 bytes from 192.168.1.103: icmp_seq=21 ttl=64 time=0.883 ms  
64 bytes from 192.168.1.103: icmp_seq=22 ttl=64 time=1.59 ms  
64 bytes from 192.168.1.103: icmp_seq=23 ttl=64 time=0.819 ms  
64 bytes from 192.168.1.103: icmp_seq=24 ttl=64 time=1.17 ms  
64 bytes from 192.168.1.103: icmp_seq=25 ttl=64 time=0.598 ms  
64 bytes from 192.168.1.103: icmp_seq=26 ttl=64 time=0.549 ms  
64 bytes from 192.168.1.103: icmp_seq=27 ttl=64 time=0.458 ms
```

LOG FILE: *ICMP connection attempt message displayed & attempt classified as ICMP detected using customized ruleset*


```

] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:29:16.853322  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
  activity] [Priority: 3] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:29:17.876885  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc
  activity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:17.876885  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0
] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:17.876885  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
  ity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:17.876944  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0
] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:29:17.876944  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
  activity] [Priority: 3] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:29:18.877557  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc
  activity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:18.877557  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0
] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:18.877557  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
  ity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:29:18.877580  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0
] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:29:18.877580  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
  activity] [Priority: 3] {ICMP} 192.168.1.103 -> 192.168.1.104

```

RESULT 2

Go to the remote system and perform a simple ftp connection attempt and test against the Suricata server using the *ftp server_ip* command as shown below

```

File  Actions  Edit  View  Help
root@kali: ~
root@kali:~# ftp 192.168.1.103
ftp: connect: Connection refused
ftp> ^Z
[3]+  Stopped                  ftp 192.168.1.103
root@kali:~#

```

LOG FILE: *FTP connection attempt message displayed & attempt classified as TCP detected using customized ruleset*

```

activity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:30:26.676561  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[ICMP] 192.168.1.104 -> 192.168.1.103
06/03-22:30:26.676561  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
ity] [Priority: 3] {ICMP} 192.168.1.104 -> 192.168.1.103
06/03-22:30:26.676597  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[ICMP] 192.168.1.103 -> 192.168.1.104
06/03-22:30:26.676597  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
activity] [Priority: 3] {ICMP} 192.168.1.103 -> 192.168.1.104
06/03-22:30:43.957218  [**] [1:9998:1] FTP connection attempt [**] [Priority: 0]
{TCP} 192.168.1.104:60596 -> 192.168.1.103:21
06/03-22:30:49.809017  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[IPV6-ICMP] fe80::8056:781f:af6e:68da -> ff02::16
06/03-22:30:49.824875  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[IPV6-ICMP] fe80::8056:781f:af6e:68da -> ff02::16
06/03-22:30:49.825585  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[IPV6-ICMP] fe80::8056:781f:af6e:68da -> ff02::16
06/03-22:30:49.825797  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[IPV6-ICMP] fe80::8056:781f:af6e:68da -> ff02::16
06/03-22:30:50.212516  [**] [1:9999:1] ICMP connection attempt [**] [Priority: 0]
[IPV6-ICMP] fe80::8056:781f:af6e:68da -> ff02::16
06/03-22:31:49.868544  [**] [1:9998:1] FTP connection attempt [**] [Priority: 0]
{TCP} 192.168.1.104:60598 -> 192.168.1.103:21

```

RESULT 3

Go to the remote system and perform a simple ssh connection attempt and test against the Suricata server using the *ssh server_ip* command as shown below

```

File Actions Edit View Help
root@kali: ~
root@kali:~# ssh 192.168.1.103
ssh: connect to host 192.168.1.103 port 22: Connection refused
root@kali:~#

```

LOG FILE: *SSH connection attempt message displayed & attempt classified as TCP detected using customized ruleset*


```

06/03-22:33:41.964542  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:41.967874  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:42.056425  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:42.059778  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:42.063525  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:42.067173  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:33:42.070649  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [C
lassification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.
250:1900
06/03-22:34:00.046860  [**] [1:10001:1] SSH connection attempt [**] [Priority: 0
] {TCP} 192.168.1.104:59364 -> 192.168.1.103:22

```

IPS TESTING BASED ON DEFAULT RULESETS

Go to the remote system and perform a simple DDoS attack and test against the Suricata server using the hping3 tool with ***hping -S -p 80 -flood -rand-source 192.168.43.42*** command as shown below

```

root@kali:~# hping3 -S -p 80 --flood --rand-source 192.168.43.42
HPING 192.168.43.42 (eth0 192.168.43.42): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.43.42 hping statistic ---
958561 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```


LOG FILE: *ET DROP Spamhaus DROP packet dropped and classified as Misc Attack detected using default ruleset*

Also we can verify it from Kali a total of 958561 packets transmitted and 0 packets received, it means all packets are dropped.

```
ASHISH@ubuntu:~$ tail -f /var/log/suricata/fast.log
05/16/2021-00:27:31.850180  [**] [1:2400001:2889] ET DROP Spamhaus DROP Listed Traffic Inbound group 2 [
**] [Classification: Misc Attack] [Priority: 2] {TCP} 42.136.174.120:22635 -> 192.168.43.42:80
05/16/2021-00:27:31.876786  [**] [1:2400014:2889] ET DROP Spamhaus DROP Listed Traffic Inbound group 15
[**] [Classification: Misc Attack] [Priority: 2] {TCP} 147.16.149.102:23621 -> 192.168.43.42:80
05/16/2021-00:27:32.358104  [**] [1:2400012:2889] ET DROP Spamhaus DROP Listed Traffic Inbound group 13
[**] [Classification: Misc Attack] [Priority: 2] {TCP} 134.33.154.116:35027 -> 192.168.43.42:80
05/16/2021-00:27:33.693046  [**] [1:2016778:8] ET DNS Query to a *.pw domain - Likely Hostile [**] [Clas
sification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.43.42:57261 -> 192.168.43.1:53
05/16/2021-00:27:35.348867  [**] [1:2016778:8] ET DNS Query to a *.pw domain - Likely Hostile [**] [Clas
sification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.43.42:42369 -> 192.168.43.1:53
05/16/2021-17:14:45.371885  [**] [1:2022973:1] ET POLICY Possible Kali Linux hostname in DHCP Request Pa
cket [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {UDP} 0.0.0.0:68 -> 255.
255.255.255:67
05/16/2021-17:45:21.203286  [**] [1:2027865:4] ET INFO Observed DNS Query to .cloud TLD [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.43.42:44663 -> 192.168.43.1:53
05/16/2021-17:45:30.338068  [**] [1:2027865:4] ET INFO Observed DNS Query to .cloud TLD [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.43.42:40005 -> 192.168.43.1:53
05/16/2021-17:45:48.428346  [**] [1:2027865:4] ET INFO Observed DNS Query to .cloud TLD [**] [Classifica
tion: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.43.42:45338 -> 192.168.43.1:53
05/16/2021-17:45:55.293735  [**] [1:2210054:1] SURICATA STREAM excessive retransmissions [**] [Classific
ation: Generic Protocol Command Decode] [Priority: 3] {TCP} 44.231.216.202:443 -> 192.168.43.42:54172
05/16/2021-17:57:34.798242  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classificat
ion: Generic Protocol Command Decode] [Priority: 3] {TCP} 103.71.26.124:443 -> 192.168.43.42:44660
```

MACHINE LEARNING APPROACH TO BUILD IDS/IPS SYSTEM

INTRODUCTION

Intrusion Detection System is a software application to detect network intrusion using various machine learning algorithms. IDS monitors a network or system for malicious activity and protects a computer network from unauthorized access from users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between ‘bad connections’ (intrusion/attacks) and ‘good (normal) connections’.

DATASET USED - KDD Cup 1999 Data

Link to dataset: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

ABOUT DATASET

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 the Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between “bad” connections, called intrusions or attacks, and “good” normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

According to the provided dataset they divided attacks into four main categories

DOS: denial-of-service, e.g. syn flood;

R2L: unauthorized access from a remote machine, e.g. guessing password;

U2R: unauthorized access to local super user (root) privileges, e.g., various “buffer overflow” attacks;

Probing: surveillance and another probing, e.g., port scanning.

DATASET DESCRIPTION - Data Files

- **kddcup.names:** A list of features.
- **kddcup.data.gz:** The full data set
- **kddcup.data_10_percent.gz:** A 10% subset.
- **kddcup.newtestdata_10_percent_unlabeled.gz**
- **kddcup.testdata.unlabeled.gz**
- **kddcup.testdata.unlabeled_10_percent.gz**
- **corrected.gz:** Test data with corrected labels.
- **training_attack_types:** A list of intrusion types.
- **typo-correction.txt:** A brief note on a typo in the data set that has been corrected

MODEL DEVELOPMENT WORKFLOW - (Python Language)

Step 1: Data Reading

Step 2: Data Preprocessing

Step 3: Modelling (Models Implementation & Training)

Step 4: Model Testing and Result Comparison

VARIOUS APPLIED ALGORITHMS - Gaussian Naive Bayes, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression and Gradient Boosting Classifier.

MODEL ANALYSIS APPROACH - We have applied various classification algorithms that are mentioned above on the KDD dataset and compared their results to build a predictive model.

PYTHON CODE - Google drive link

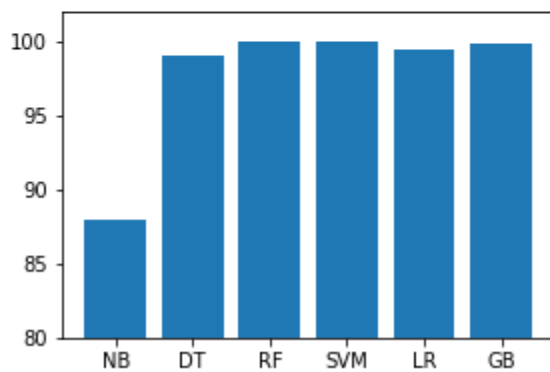
https://drive.google.com/file/d/1YjV-CeX-aOdTRyCk2XmGz_INmmZX6_ir/view?usp=sharing

MODEL TESTING RESULT

TESTING ACCURACY

```
names = ['NB','DT','RF','SVM','LR','GB']
values = [87.903,99.052,99.969,99.879,99.352,99.771]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)
```

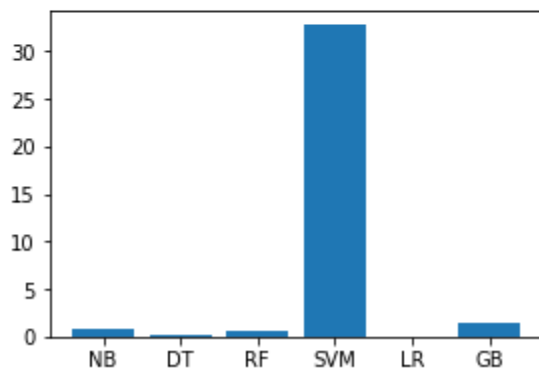
<BarContainer object of 6 artists>



TESTING TIME

```
names = ['NB','DT','RF','SVM','LR','GB']
values = [0.79089,0.10471,0.60961,32.72654,0.02198,1.41416]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.bar(names,values)
```

<BarContainer object of 6 artists>



MACHINE LEARNING MODELS RESULT DISCUSSION

Our desired parameters to test the validity of trained model are Testing Accuracy and Testing Time. We are chasing a model which satisfy the criteria of highest accuracy and lowest testing time. After implementing different classifier algorithm and model training we have got a minimum of 87.903% testing accuracy using Gaussian Naive Bayes classifier algorithm and a highest of 99.969% testing accuracy using Support Vector Machine classifier algorithm. On the other hand, a maximum testing time 32.7265 ms is taken by Support Vector Machine algorithm while a minimum testing time 0.10471 ms is taken by Decision Tree algorithm.

MACHINE LEARNING MODELS INFERENCE

From the above analysis of different models the Support Vector Machine model gives highest testing accuracy but consumes a large amount of testing time. But after proper detailed analysis we can conclude that the Decision Tree model best fits our data considering both accuracy and time complexity.

CONCLUSION

We have successfully, designed and deployed IDS and IPS system using Suricata on Ubuntu 20.04 server and tested the working of the system by performing different types of attacks, ICMP, FTP and SSH connection attempts and port scanning methods using our remote attacker host Kali Linux. We observed that our developed system is able to drop or alert the user based upon the defined default and user defined rulesets.

Also, we have successfully explored Machine Learning approach to develop IDS and IPS system. We have trained a Machine Learning Model which has very high accuracy testing accuracy of 99.969% which can overcome the issue of false positive in network security monitoring engine up to a great extent.

REFERENCES

- [1] <https://enlyft.com/tech/products/suricata>
- [2] <https://www.varonis.com/blog/ids-vs-ips/>
- [3] <https://suricata.readthedocs.io/en/suricata-6.0.0/quickstart.html>
- [4] <https://kifarunix.com/install-and-setup-suricata-on-ubuntu-18-04/>
- [5] https://www.researchgate.net/publication/236146583_Quantitative_Analysis_of_Intrusion_Detection_Systems_Snort_and_Suricata
- [6] https://www.researchgate.net/publication/301668778_A_Comparative_Study_of_Performance_of_Open_Source_IDSIPS_Snort_and_Suricata
- [7] https://www.researchgate.net/publication/241701294_A_Performance_Analysis_of_Snort_and_Suricata_Network_Intrusion_Detection_and_Prevention_Engines
- [8] <https://ieeexplore.ieee.org/document/7946818>

BIODATA

Name: Mukul Kumar

Reg_no: 18BEC1197

Email: mukul.kumar2018@vitstudent.ac.in



Name: Nikhil Anand

Reg_no: 18BEC1346

Email: nikhil.anand2018@vitstudent.ac.in



Name: Ashish Tiwari

Reg_no: 18BEC1221

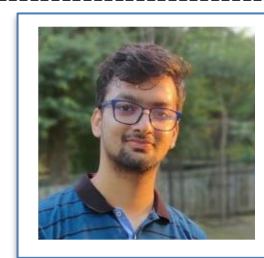
Email: ashish.kumar2018@vitstudent.ac.in



Name: Suryansh Sohgaure

Reg_no: 18BEC1333

Email: suryansh.sohgaure2018@vitstudent.ac.in



Name: Vishnu Puligadda

Reg_no: 18BEC1345

Email: vishnu.puligadda2018@vitstudent.ac.in

