WELCOME TO THE OFFICIAL SITE OF

NARNARAYAN SHASTRI INSTITUTE OF TECHNOLOGY

INSTITUTE OF FORENSIC SCIENCES & CYBER SECURITY

AFFILIATED TO

NATIONAL FORENSIC SCIENCES UNIVERSITY

(INSTITUTE OF NATIONAL IMPORTANCE, MHA, GOVT. OF INDIA)

Narnarayan Shastri Institute of Technology

Institute of Forensics Sciences and Cyber Security

M.Sc. Cyber Security

Semester - II (2024-25)

Lab Manual

Subject: Minor Project

Subject Code: CTMSCS SII L5

Name: Upadhyay Sunil Prahladbhai

Enrolment No: 240043001016

WELCOME TO THE OFFICIAL SITE OF
NARNARAYAN SHASTRI INSTITUTE OF TECHNOLOGY
INSTITUTE OF FORENSIC SCIENCES & CYBER SECURITY
AFFILIATED TO
NATIONAL FORENSIC SCIENCES UNIVERSITY
(INSTITUTE OF NATIONAL IMPORTANCE, MHA, GOVT. OF INDIA)

MINOR PROJECT

ON

“CloakSphere: A Framework for IDS Evasion and Pentesting”

Submitted To

Narnarayan shastri institute of technology
Institute Of Forensic Sciences and Cyber Security

Submitted By

Sunil Upadhyay
(240043001016)

Under The Supervision Of
Mr. Akash Khunt
Assistant Professor

Narnarayan Shastri Institute of Technology
Institute Of Forensic Sciences and Cyber Security
(Jetalpur - Ahmedabad)
April 2025

WELCOME TO THE OFFICIAL SITE OF

NARNARAYAN SHASTRI INSTITUTE OF TECHNOLOGY

INSTITUTE OF FORENSIC SCIENCES & CYBER SECURITY

AFFILIATED TO

NATIONAL FORENSIC SCIENCES UNIVERSITY

(INSTITUTE OF NATIONAL IMPORTANCE, MHA, GOVT. OF INDIA)

DECLARATION

I certify that

a. The work contained in the dissertation is original and has been done by myself under the supervision of my supervisor.

b. I have confirmed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

c. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

d. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

Date:

Place: JETALPUR, AHMEDABAD

SUNIL UPADHYAY

240043001016

WELCOME TO THE OFFICIAL SITE OF

NARNARAYAN SHASTRI INSTITUTE OF TECHNOLOGY

INSTITUTE OF FORENSIC SCIENCES & CYBER SECURITY

AFFILIATED TO

NATIONAL FORENSIC SCIENCES UNIVERSITY

(INSTITUTE OF NATIONAL IMPORTANCE, MHA, GOVT. OF INDIA)

CERTIFICATE

THIS IS TO CERTIFY THAT THE WORK CONTAINED IN THE DISSERTATION ENTITLED **CLOAKSPHERE: A FRAMEWORK FOR IDS EVASION AND PENTESTING,** SUBMITTED BY **SUNIL UPADHYAY** FOR THE MINOR PROJECT **MASTER OF SCIENCE IN CYBER SECURITY** TO THE **NARNARAYAN SHASTRI INSTITUTE OF TECHNOLOGY, JETALPUR** IS A RECORD OF BONAFIDE RESEARCH WORKS CARRIED OUT BY HIM UNDER MY DIRECT SUPERVISION AND GUIDANCE.

PLACE: JETALPUR, AHMEDABAD
DATE OF SUBMISSION:

SIGNATURE
FACULTY

SIGNATURE
PRINCIPAL

ACKNOWLEDGEMENTS

I Would like to express my sincere gratitude to all the individuals who have supported and guided me throughout the completion of this dissertation on "**CloakSphere: A Framework for IDS Evasion and Pentesting**". This project has been a challenging and enlightening journey, and it would not have been possible without the encouragement, guidance, and assistance from many people.

First and foremost, I would like to thank my supervisor, Mr. Akash Khunt, for his invaluable guidance, insightful feedback, and constant encouragement throughout this project. His expertise and dedication have been instrumental in shaping this dissertation, and I am deeply grateful for his mentorship.

I am also grateful to the faculty members of the Cyber Security Department for their encouragement and for imparting their knowledge throughout my studies. Their courses and lectures have laid a solid foundation for my research work.

To all those mentioned and to others who have contributed in some way, I extend my deepest gratitude. This dissertation would not have been possible without your collective support.

With Sincere Regards,
Sunil Upadhyay
M.Sc. Cyber Security

ABSTRACT

This practical explores the design and implementation of custom Snort IDS (Intrusion Detection System) rules to detect various network-based attacks and attempts to bypass detection through evasion techniques.

The project, titled **"CloakSphere"**, focuses on simulating real-world intrusion scenarios, including SSH authentication and brute-force attempts, improper input validation, SQL injection, unauthorized file uploads, and reflected cross-site scripting (XSS).

Each detection rule is crafted using Snort's powerful pattern matching capabilities such as content and PCRE (Perl-Compatible Regular Expressions). Additionally, the project investigates techniques attackers use to evade IDS detection, such as Base64 encoding and SQL obfuscation using inline comments.

The aim of this practical is to enhance students' understanding of how intrusion detection systems operate and how attackers attempt to bypass them, while also improving the ability to write and test custom Snort rules in a controlled lab environment.

TABLE OF CONTENT

Chapter 1: Introduction

1.1 Overview of Intrusion Detection Systems (IDS)

- An Intrusion Detection System (IDS) is a tool used in cybersecurity to monitor and analyse network traffic or system activity for signs of unauthorized access, attacks, or policy violations.
- IDS helps identify suspicious behaviour by comparing network data against a known database of threats or by observing unusual patterns in traffic.

There are mainly two types of IDS:

- **Network-based IDS (NIDS):** Monitors entire network traffic.
- **Host-based IDS (HIDS):** Monitors a single host or device.
- IDS tools do not block attacks directly but alert administrators when something unusual is detected.

1.2 Importance of IDS in Modern Cybersecurity

In today's world, where cyberattacks are growing every day, IDS plays an important role in protecting systems and networks. It helps in:

- Detecting potential threats before they cause damage.
- Monitoring ongoing attacks in real-time.
- Alerting system administrators quickly to act.
- Logging important information for future analysis.

Using IDS makes it easier to detect and understand attacks like malware, denial of service (DoS), brute force, and more.

1.3 Problem Statement

- Traditional security methods like firewalls and antivirus software are not enough to detect modern threats. Attackers use advanced techniques to bypass security controls.

- The challenge is to build or use an IDS that can detect these threats efficiently without causing too many false alarms.

- This project aims to explore and enhance an IDS tool like **Snort**, focusing on custom rule creation and testing its effectiveness.

1.4 Scope of the Project

This project is focused on using Snort, a popular open-source IDS. The project includes:

- Installing Snort in a Linux environment.
- Writing and testing rules for different attack types like DoS, port scanning, etc.
- Performing test attacks in a controlled environment to check detection.
- Analysing the logs and results to improve rule performance.

The project does not cover commercial IDS systems or large-scale deployment in enterprise networks.

Chapter 2: Literature Review

2.1 Snort – Lightweight Intrusion Detection for Networks (Martin Roesch)

This paper introduces Snort, an open-source tool designed to monitor network traffic and detect potential threats. Snort is known for being efficient and adaptable, allowing users to define specific patterns to watch for, which helps in identifying new and emerging threats. It's widely used due to its effectiveness and flexibility.

2.2 Evasion and Denial of Service Techniques (Ptacek & Newsham)

This work discusses methods attackers use to avoid detection by IDS like Snort. The authors explain how attackers can manipulate data packets—by breaking them into smaller pieces, altering sequences, or timing transmissions differently—to slip past detection systems. Understanding these tactics is crucial for improving IDS effectiveness.

2.3 Snort Toolkit and Configuration Techniques (Toby Kohlenberg)

This book provides a comprehensive guide on using Snort for both detecting and preventing intrusions. It offers practical advice on setting up, configuring, and optimizing Snort in various network environments. The authors include real-world examples and case studies, making it a valuable resource for those looking to enhance their network security using Snort.

2.4 Enhancing IDS with Custom Rules (Khamphakdee & Benjamas)

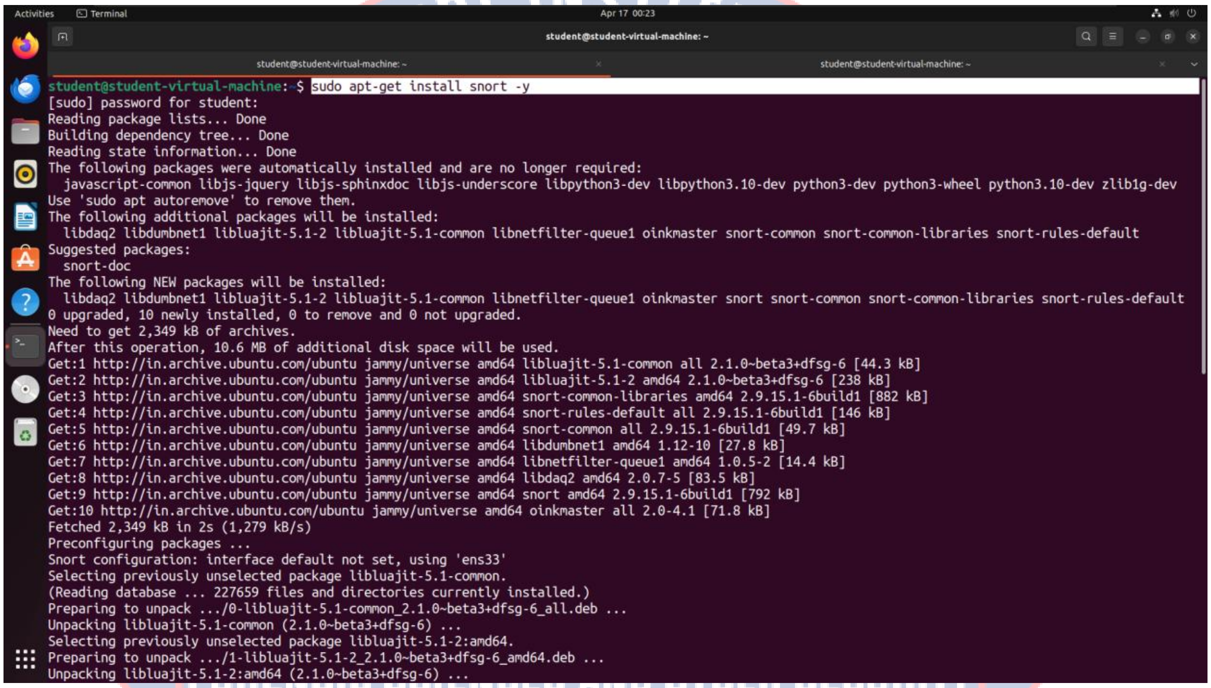
This research focuses on enhancing Snort's ability to detect specific types of attacks known as network probes, where attackers scan networks to find vulnerabilities. The authors suggest changes to existing Snort rules and the creation of new ones to better identify these probing activities. Their findings show that these adjustments can lead to better detection rates and fewer false alarms.

Chapter 3: Project Design & Methodology

3.1 Lab Setup and Components

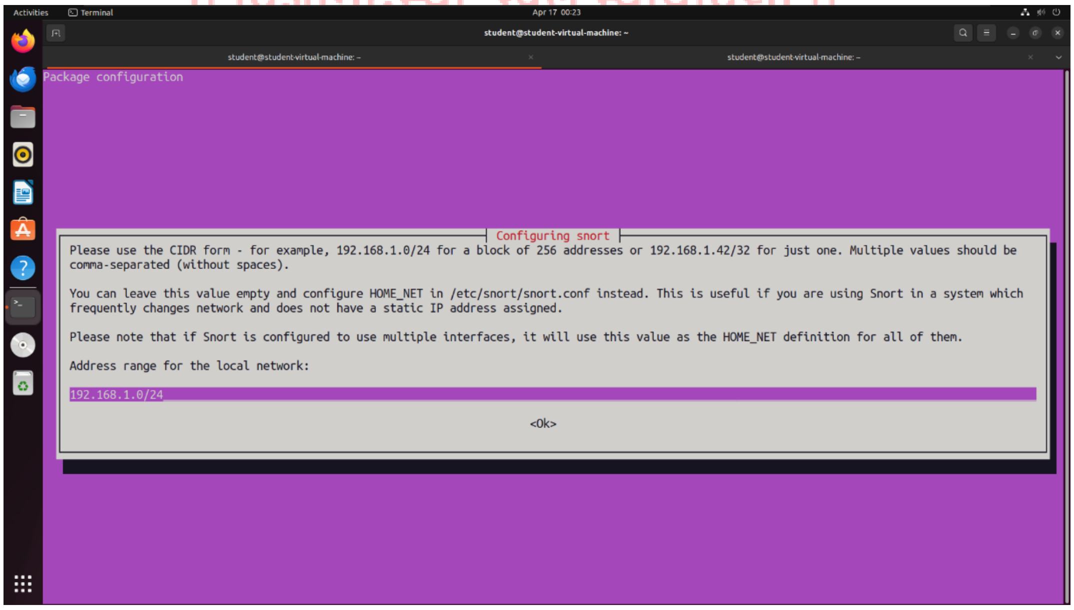
3.1.1 IDS Server Configuration (Ubuntu + Snort 2.9.15)

- Snort can be installed quickly using the package manager on Debian/Ubuntu systems.



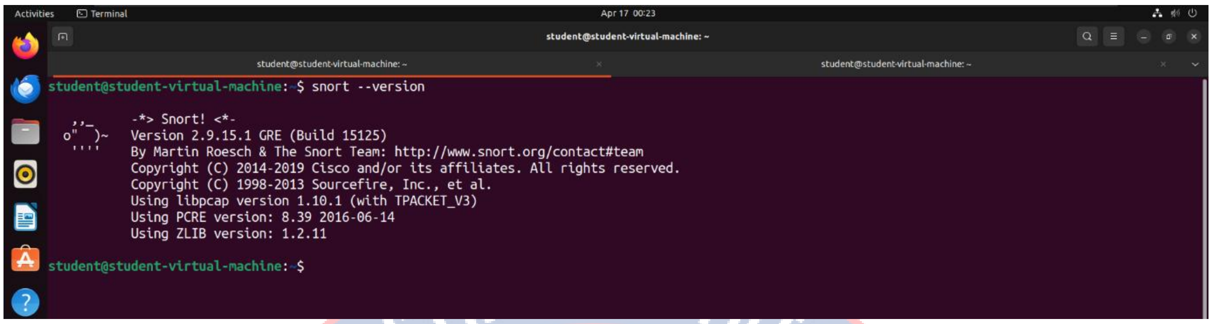
```
student@student-virtual-machine:~$ sudo apt-get install snort -y
[sudo] password for student:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev python3-dev python3-wheel python3.10-dev zlib1g-dev
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libdaq2 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common libnetfilter-queue1 oinkmaster snort-common snort-common-libraries snort-rules-default
Suggested packages:
  snort-doc
The following NEW packages will be installed:
  libdaq2 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common libnetfilter-queue1 oinkmaster snort snort-common snort-common-libraries snort-rules-default
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,349 kB of archives.
After this operation, 10.6 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libluajit-5.1-common all 2.1.0~beta3+dfsg-6 [44.3 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libluajit-5.1-2 amd64 2.1.0~beta3+dfsg-6 [238 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 snort-common-libraries amd64 2.9.15.1-6build1 [882 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 snort-rules-default all 2.9.15.1-6build1 [146 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 snort-common all 2.9.15.1-6build1 [49.7 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libdumbnet1 amd64 1.12-10 [27.8 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libnetfilter-queue1 amd64 1.0.5-2 [14.4 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libdaq2 amd64 2.0.7-5 [83.5 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 snort amd64 2.9.15.1-6build1 [792 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 oinkmaster all 2.0-4.1 [71.8 kB]
Fetched 2,349 kB in 2s (1,279 kB/s)
Preconfiguring packages ...
Snort configuration: interface default not set, using 'ens33'
Selecting previously unselected package libluajit-5.1-common.
(Reading database ... 227659 files and directories currently installed.)
Preparing to unpack .../0-libluajit-5.1-common_2.1.0~beta3+dfsg-6_all.deb ...
Unpacking libluajit-5.1-common (2.1.0~beta3+dfsg-6) ...
Selecting previously unselected package libluajit-5.1-2:amd64.
Preparing to unpack .../1-libluajit-5.1-2_2.1.0~beta3+dfsg-6_amd64.deb ...
Unpacking libluajit-5.1-2:amd64 (2.1.0~beta3+dfsg-6) ...
```

- During installation, you'll be prompted to enter the **IP address range of your network**. For example: 192.168.1.0/24



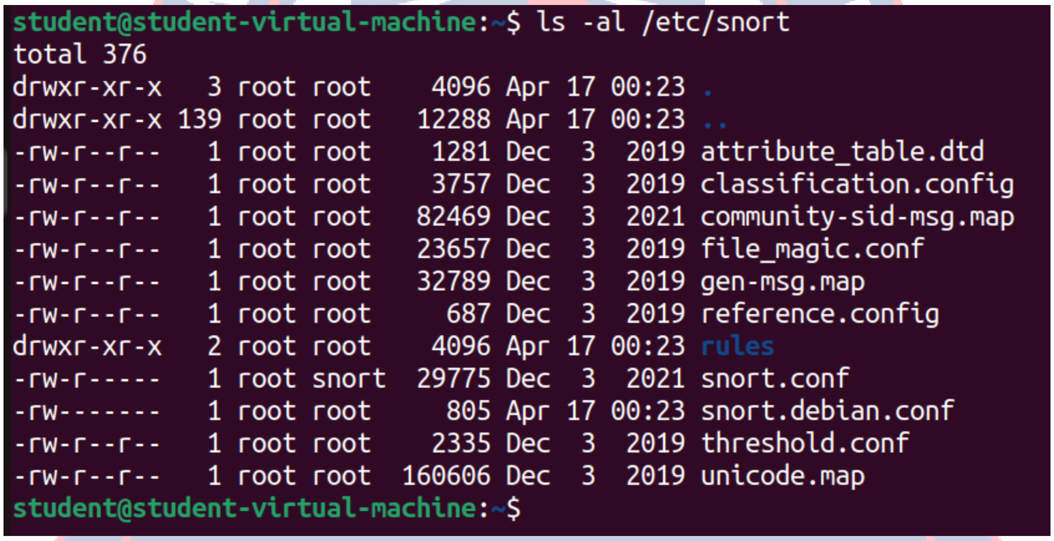
```
Package configuration

                              Configuring snort
Please use the CIDR form - for example, 192.168.1.0/24 for a block of 256 addresses or 192.168.1.42/32 for just one. Multiple values should be
comma-separated (without spaces).

You can leave this value empty and configure HOME_NET in /etc/snort/snort.conf instead. This is useful if you are using Snort in a system which
frequently changes network and does not have a static IP address assigned.

Please note that if Snort is configured to use multiple interfaces, it will use this value as the HOME_NET definition for all of them.

Address range for the local network:

192.168.1.0/24

                                    <Ok>
```

- The version command ensures Snort has been installed successfully.



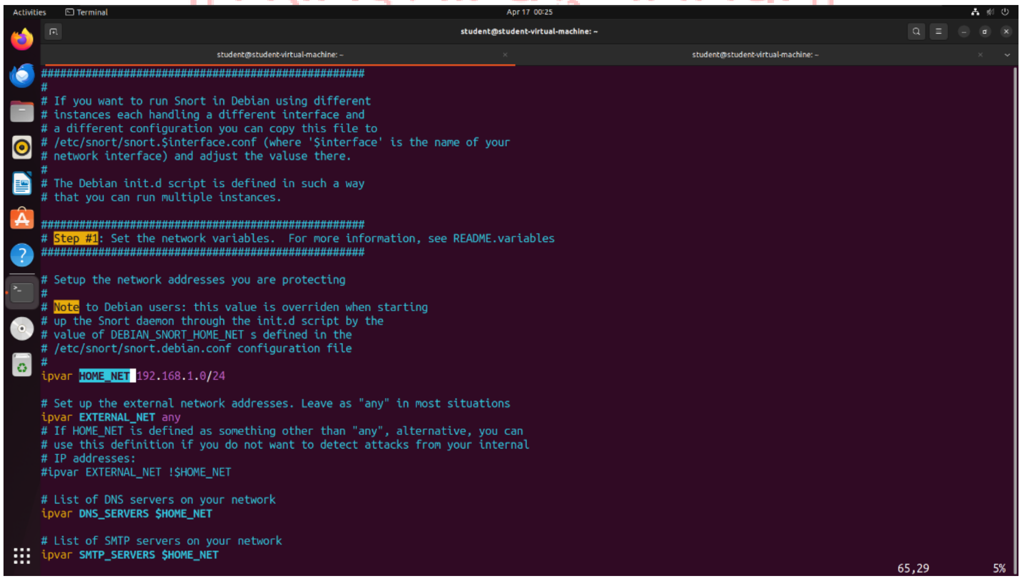
```
student@student-virtual-machine:~$ snort --version

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

student@student-virtual-machine:~$
```

- Snort's configuration files are stored in /etc/snort/ and control how it behaves.



```
student@student-virtual-machine:~$ ls -al /etc/snort
total 376
drwxr-xr-x   3 root root    4096 Apr 17 00:23 .
drwxr-xr-x 139 root root   12288 Apr 17 00:23 ..
-rw-r--r--   1 root root    1281 Dec  3  2019 attribute_table.dtd
-rw-r--r--   1 root root    3757 Dec  3  2019 classification.config
-rw-r--r--   1 root root   82469 Dec  3  2021 community-sid-msg.map
-rw-r--r--   1 root root   23657 Dec  3  2019 file_magic.conf
-rw-r--r--   1 root root   32789 Dec  3  2019 gen-msg.map
-rw-r--r--   1 root root     687 Dec  3  2019 reference.config
drwxr-xr-x   2 root root    4096 Apr 17 00:23 rules
-rw-r-----   1 root snort  29775 Dec  3  2021 snort.conf
-rw-------   1 root root     805 Apr 17 00:23 snort.debian.conf
-rw-r--r--   1 root root    2335 Dec  3  2019 threshold.conf
-rw-r--r--   1 root root  160606 Dec  3  2019 unicode.map
student@student-virtual-machine:~$
```

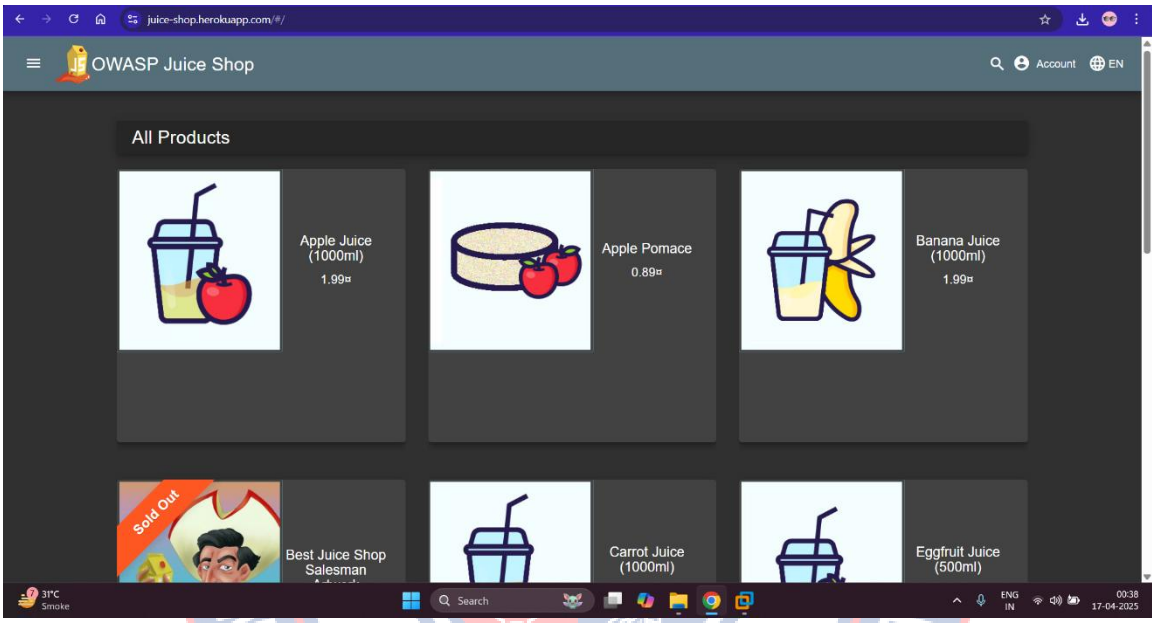
- Edit Snort Configuration to Set HOME_NET



```
###################################################
#
# If you want to run Snort in Debian using different
# instances each handling a different interface and
# a different configuration you can copy this file to
# /etc/snort/snort.$interface.conf (where '$interface' is the name of your
# network interface) and adjust the valuse there.
#
# The Debian init.d script is defined in such a way
# that you can run multiple instances.

###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.1.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET
                                                            65,29         5%
```

3.1.2 Target Web-Server

- Here you can see this is the OWASP juice shop website that we are cloning it and using for the testing purpose.



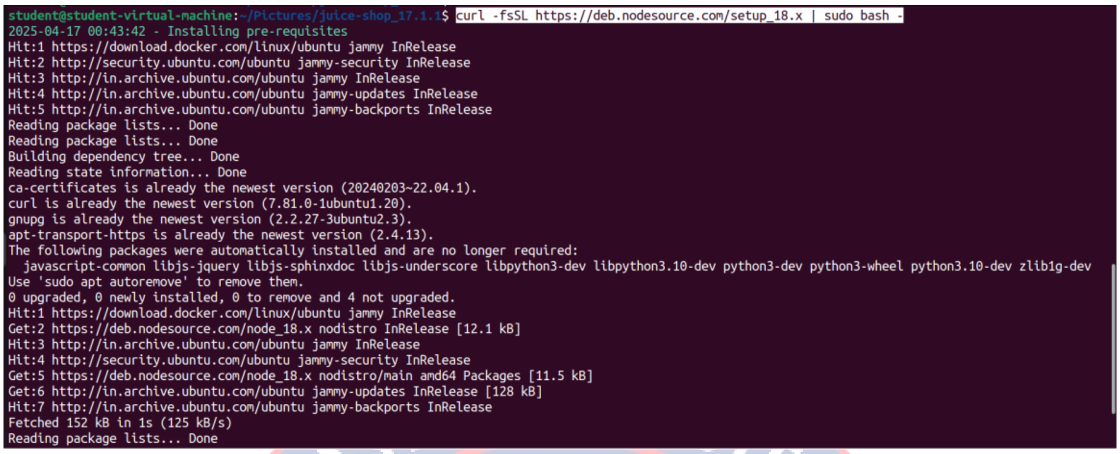
- Download the OWASP Juice Shop .tgz Archive

```
student@student-virtual-machine:~/Pictures$ ls
juice-shop-17.1.1_node18_linux_x64.tgz
student@student-virtual-machine:~/Pictures$ 
```

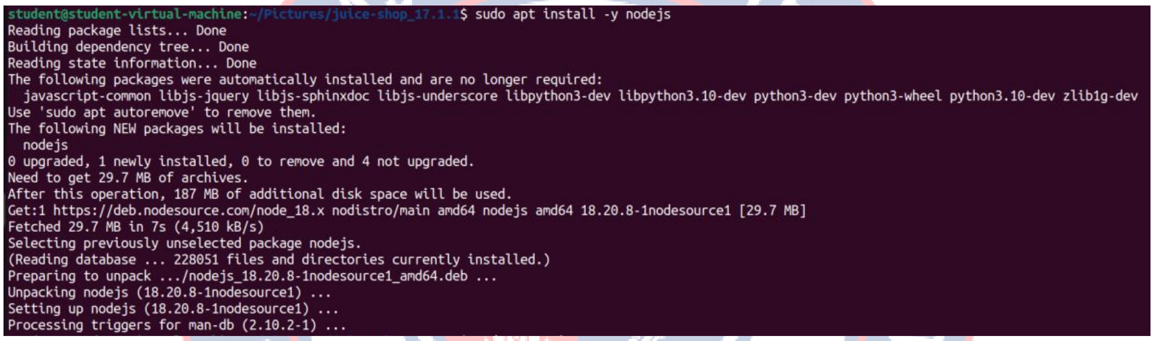
- Extract the Archive

```
student@student-virtual-machine:~/Pictures$ tar -xvzf juice-shop-17.1.1_node18_linux_x64.tgz
```

- Install Dependencies (Node.js, npm, etc.)

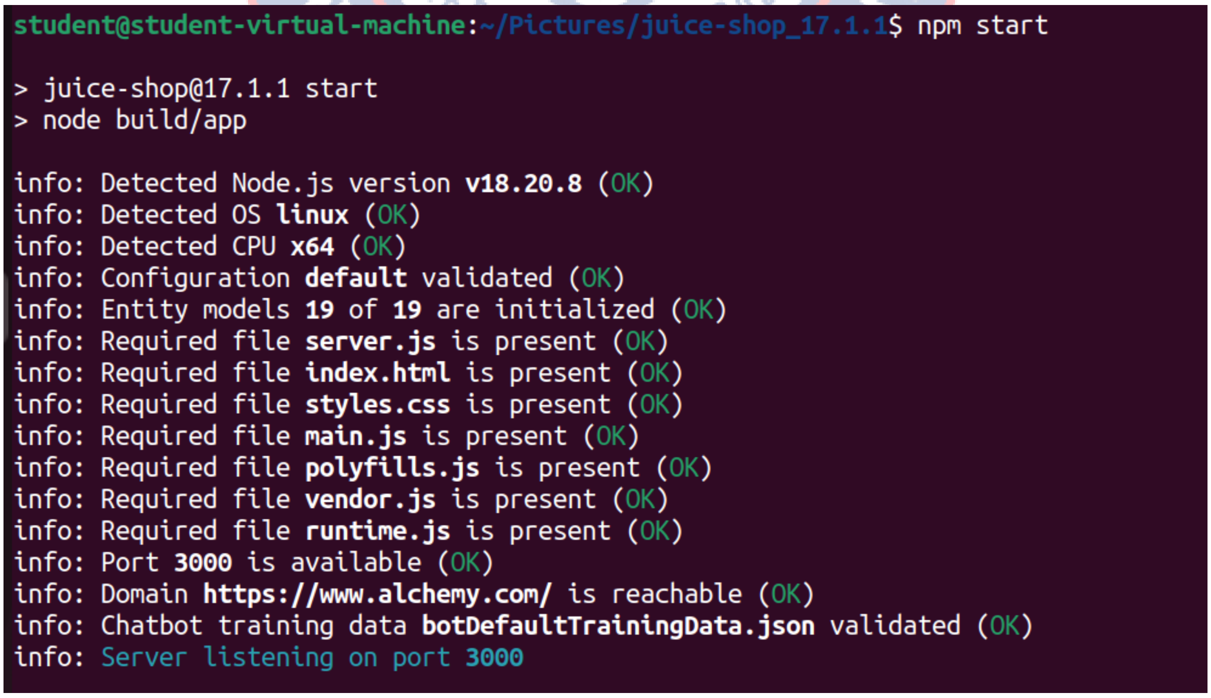


```
student@student-virtual-machine:~/Pictures/juice-shop_17.1.1$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo bash -
2025-04-17 00:43:42 - Installing pre-requisites
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.20).
gnupg is already the newest version (2.2.27-3ubuntu2.3).
apt-transport-https is already the newest version (2.4.13).
The following packages were automatically installed and are no longer required:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev python3-dev python3-wheel python3.10-dev zlib1g-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Get:2 https://deb.nodesource.com/node_18.x nodistro InRelease [12.1 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:5 https://deb.nodesource.com/node_18.x nodistro/main amd64 Packages [11.5 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 152 kB in 1s (125 kB/s)
Reading package lists... Done
```



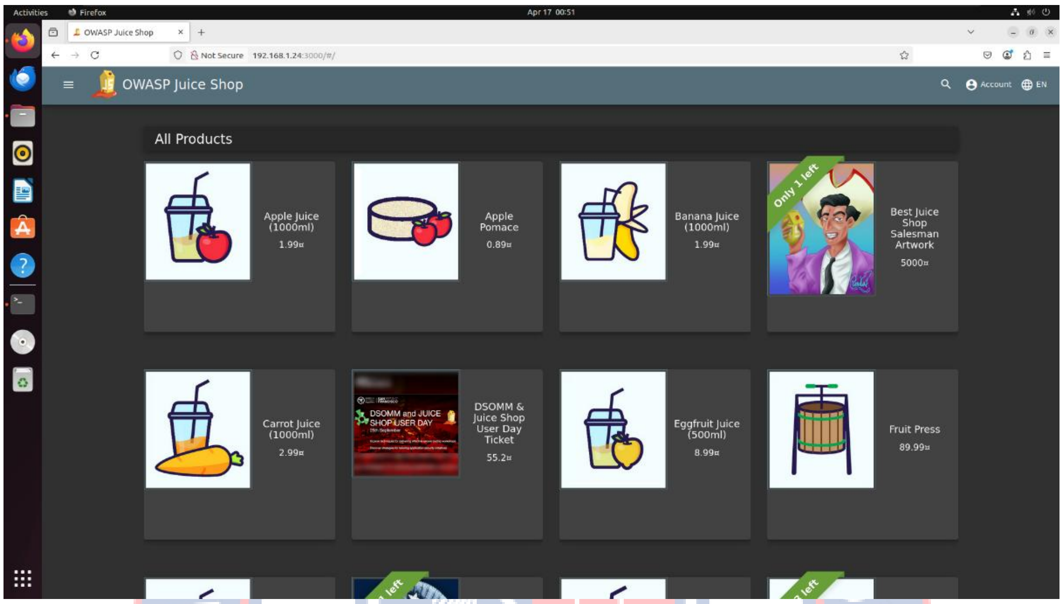
```
student@student-virtual-machine:~/Pictures/juice-shop_17.1.1$ sudo apt install -y nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev python3-dev python3-wheel python3.10-dev zlib1g-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 29.7 MB of archives.
After this operation, 187 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_18.x nodistro/main amd64 nodejs amd64 18.20.8-1nodesource1 [29.7 MB]
Fetched 29.7 MB in 7s (4,510 kB/s)
Selecting previously unselected package nodejs.
(Reading database ... 228051 files and directories currently installed.)
Preparing to unpack .../nodejs_18.20.8-1nodesource1_amd64.deb ...
Unpacking nodejs (18.20.8-1nodesource1) ...
Setting up nodejs (18.20.8-1nodesource1) ...
Processing triggers for man-db (2.10.2-1) ...
```

- Start the Juice Shop Server



```
student@student-virtual-machine:~/Pictures/juice-shop_17.1.1$ npm start

> juice-shop@17.1.1 start
> node build/app

info: Detected Node.js version v18.20.8 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main.js is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file vendor.js is present (OK)
info: Required file runtime.js is present (OK)
info: Port 3000 is available (OK)
info: Domain https://www.alchemy.com/ is reachable (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Server listening on port 3000
```

- Access the Web App via Browser


Activities
Firefox
Apr 17 00:51
OWASP Juice Shop
Not Secure 192.168.1.24:3000/#/
OWASP Juice Shop
Account
EN
All Products
Apple Juice (1000ml) 1.99¤
Apple Pomace 0.89¤
Banana Juice (1000ml) 1.99¤
Only 1 left
Best Juice Shop Salesman Artwork 5000¤
Carrot Juice (1000ml) 2.99¤
DSOMM & Juice Shop User Day Ticket 55.2¤
Eggfruit Juice (500ml) 8.99¤
Fruit Press 89.99¤

3.2 Tools Used

3.2.1 TCPDump

Definition:
TCPDump is a powerful command-line packet analyser used for network traffic monitoring and analysis. It allows users to capture and inspect data packets transmitted over a network in real-time.

Key Features:

- Captures packets at the network interface level.
- Allows filtering of packets using powerful expressions.
- Supports protocols like TCP, UDP, ICMP, ARP, and many more.
- Outputs readable information for analysis.
- Lightweight and widely used in Linux environments.

Use in the Project:
In this project, TCPDump was used to capture network traffic during attack simulations. It helped analyse how packets flow through the network and whether malicious activities (like brute force or DoS) triggered Snort rules properly.

3.2.2 Hydra

Definition:
Hydra (also known as THC-Hydra) is a fast and flexible network login cracker. It supports many protocols and is used to perform brute-force attacks on login pages or network services.

Key Features:

- Supports multiple protocols (FTP, SSH, Telnet, HTTP, HTTPS, SMB, etc.).
- Can use dictionary (wordlist) attacks to guess username/password combinations.
- Highly configurable and fast in execution.
- Open-source and available for Linux.

Use in the Project:
Hydra was used to simulate brute-force attacks against login forms and services. This helped in testing whether Snort's custom rules could detect brute-force attempts and raise alerts accordingly.

Chapter 4: Detection Rule Implementation

4.1 Understanding Snort Rules

4.1.1 Structure and Syntax

Snort rules are the core of how Snort detects suspicious activity. Each rule tells Snort what kind of network traffic to look for and what to do when it finds it.

Basic Structure of a Snort Rule:

action protocol source_IP source_port -> dest_IP dest_port (options)

- **Action** — What Snort should do (e.g,, alert, log, drop).
- **Protocol** — Type of traffic (e.g., TCP, UDP, ICMP).
- **Source IP and Port** — Where the traffic comes from.
- **Destination IP and Port** — Where the traffic is going.
- **Options** — Extra instructions like messages, content patterns, and log details.

4.1.2 Types of Rules

Snort uses different types of rules based on how it should handle the detected traffic. Each rule type serves a specific purpose in the intrusion detection or prevention process. These rules help network administrators monitor, respond to, or block potentially harmful activities.

1. Detection Rules

Detection rules are the most used rules in Snort. These rules are designed to identify suspicious or malicious activity on the network. When a packet matches the conditions in a detection rule, Snort triggers an alert to notify the administrator.

- **Purpose**: To inform about potential attacks like XSS, SQL injection, port scans, brute force attempts, etc.
- **Common usage**: Monitoring critical services and detecting known attack patterns.

2. Prevention Rules

Prevention rules are used when Snort is running in inline mode. In this mode, Snort can actively block or drop malicious packets in real time before they reach their destination. These rules are critical in a proactive security setup, especially in enterprise environments.

- **Purpose**: To not only detect threats but also prevent them from executing.
- **Common usage**: Blocking unauthorized access, malware communication, and exploitation attempts.

3. Logging Rules

Logging rules allow Snort to record specific types of traffic without raising alerts. This is useful when you want to gather data for analysis, debugging, or legal documentation without overwhelming the alert system.

- **Purpose**: To collect and store data for further investigation.
- **Common usage**: Traffic profiling, forensic analysis, or identifying slow attack patterns.

4.2 Types of Detection Techniques

Intrusion Detection Systems (like Snort) use different methods to identify attacks. These methods help detect both known threats and unknown or modified attack patterns. In this project, we focused on two major types of detection techniques used in Snort:

4.2.1 Signature-Based Detection

Signature-based detection, also known as rule-based detection, works by looking for specific patterns of known attacks. These patterns, called signatures, are like digital fingerprints of common threats. Snort compares incoming network traffic with these predefined patterns, and if it finds a match, it triggers an alert.

Key Points:

- Very accurate for detecting known threats.
- Fast and lightweight, as it checks for exact patterns.
- Limited in detecting new or modified attacks (zero-day attacks).

Advantages:

- Easy to implement and manage.
- High accuracy for known threats.

Disadvantages:

- Cannot detect unknown or slightly modified attacks.
- Needs frequent updates to include new signatures.

4.2.2 Regex-Based Detection (Regular Expression Detection)

Regex-based detection uses regular expressions (also called RegEx) to define more flexible and complex patterns. This method is useful when the exact format of an attack is unknown or changes frequently. With regular expressions, Snort can match variations of attack strings, even if attackers try to change the format to bypass detection.

Key Points:

- Uses PCRE (Perl Compatible Regular Expressions) in Snort.
- Can detect a wider range of attacks with just one rule.
- Ideal for detecting encoded, obfuscated, or altered payloads.

Advantages:

- Flexible and powerful.
- Useful against slightly modified attacks or evasion techniques.

Disadvantages:

- Slightly slower than signature-based detection.
- Complex syntax and harder to maintain.

Chapter 5: Attack Simulation & Detection

5.1 SSH Authentication Detection

Purpose of the Rule

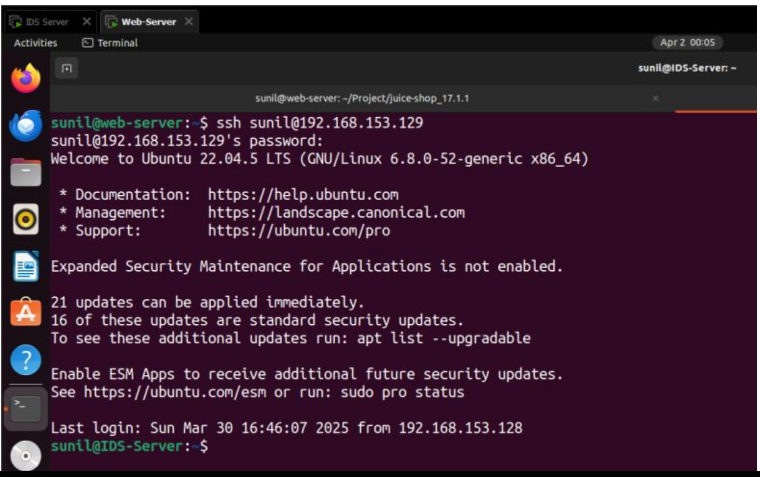
The goal of this rule is to detect SSH authentication attempts within the network. SSH (Secure Shell) is commonly used for remote administration, and monitoring authentication attempts helps identify potential unauthorized access or brute-force attacks.

alert tcp any any -Y $HOME_NET any (msg: "SSH Authentication Attempt Detected !!"; sid:100002; rev:l;)

- **Alert:** Generates an alert when the rule condition matches.
- **Tcp:** Monitors TCP traffic, as SSH operates over TCP.
- **any any:** Checks for traffic from any source IP and port.
- **$HOME NET any:** Targets any destination within the internal network.
- **msg:** Displays a message when an alert is triggered.
- **Sid:** Unique identifier for the rule.
- **rev:** Revision number for version control.

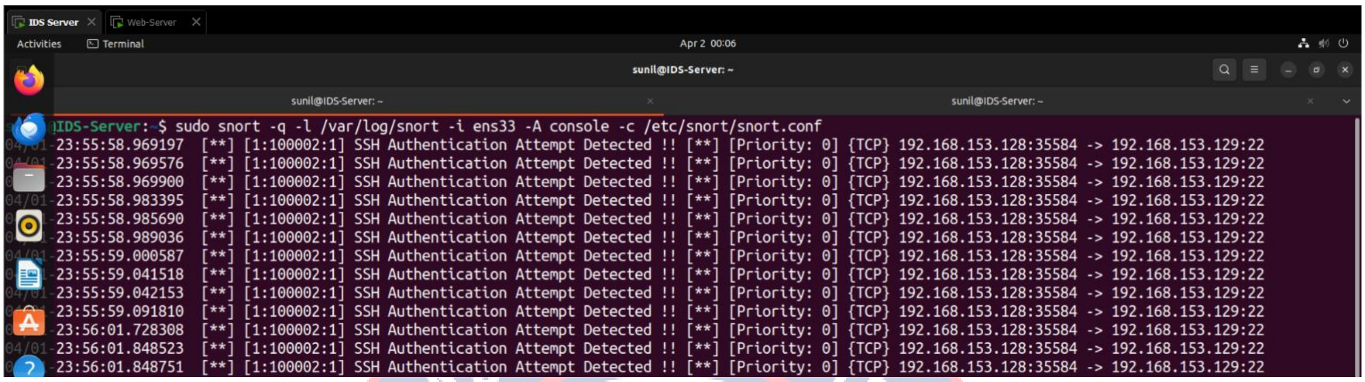
Practical Steps

- Firstly, let’s start to get the SSH connection of another machine.



```
sunil@web-server:~$ ssh sunil@192.168.153.129
sunil@192.168.153.129's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

21 updates can be applied immediately.
16 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Mar 30 16:46:07 2025 from 192.168.153.128
sunil@IDS-Server:~$
```

- Now let's run the snort and check whether the alert is generating or not.



```
IDS-Server:~$ sudo snort -q -l /var/log/snort -i ens33 -A console -c /etc/snort/snort.conf
23:55:58.969197  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:58.969576  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:58.969900  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
04/01-23:55:58.983395  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:58.985690  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:58.989036  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
04/01-23:55:59.000587  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:59.041518  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
04/01-23:55:59.042153  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:55:59.091810  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:56:01.728308  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
04/01-23:56:01.848523  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
23:56:01.848751  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.128:35584 -> 192.168.153.129:22
```

5.2 SSH Brute-Force Attack Detection

Purpose of the Rule

The goal of this rule is to detect SSH brute-force attacks, where an attacker rapidly attempts multiple login attempts using tools like Hydra. Brute-force attacks can lead to unauthorized access, so detecting them is crucial for network security.

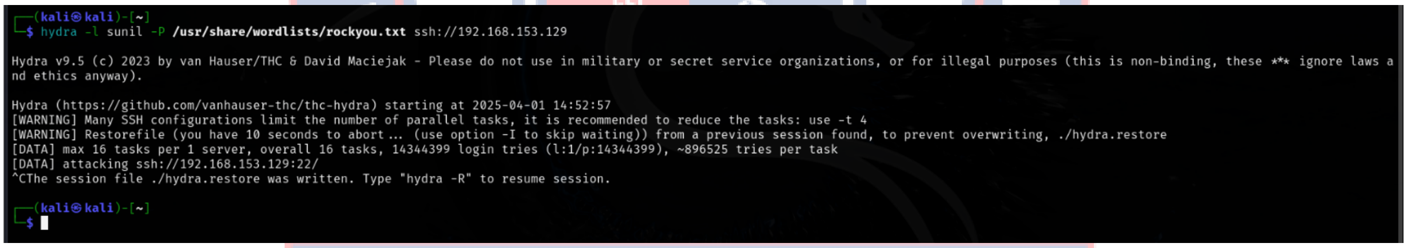
alert tcp any any -> $HOME_NET any (msg: "Possible SSH Brute Force Attempt"; flow:to_server, established; content:"SSH-2.0-"; depth:10; detection_filter:track by_src, count 5, seconds 30; classtype:attempted-admin; sid:100004; rev:1;)

- **flow:to_server, established** → Ensures traffic is directed toward the SSH server and is part of an established connection.
- **content:"SSH-2.0-"** → Matches the SSH protocol version banner, confirming SSH traffic.
- **depth:10** → Limits content inspection to the first 10 bytes to improve efficiency.

- **detection_filter:track by_src, count 5, seconds 30** → Triggers an alert if 5 or more SSH login attempts occur from the same source within 30 seconds.
- **classtype:attempted-admin** → Classifies the alert as an attempted administrative attack.

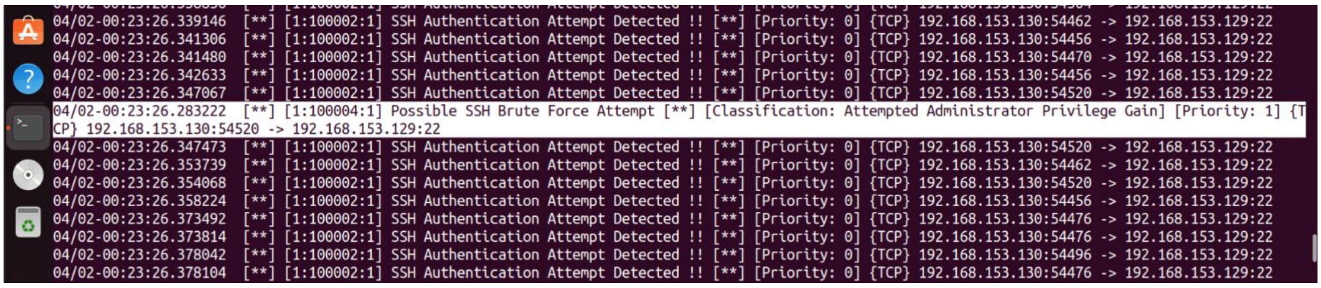
Practical Steps

- Kali Linux is commonly used as an attacker machine due to its built-in penetration testing tools.
- In this scenario, we use Kali Linux to simulate a brute-force attack on a remote system.
- Hydra is a powerful tool used to perform brute-force attacks on various protocols including SSH.
- Run Hydra to send multiple logins attempts to the SSH service on the target:



```
┌──(kali㉿kali)-[~]
└─$ hydra -l sunil -P /usr/share/wordlists/rockyou.txt ssh://192.168.153.129

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws a
nd ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-01 14:52:57
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.153.129:22/
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(kali㉿kali)-[~]
└─$ 
```

- Once Hydra begins its attack, Snort captures and analyses the packets. If detection rules are properly configured, Snort will raise an alert for SSH brute force activity.



```
04/02-00:23:26.339146  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54462 -> 192.168.153.129:22
04/02-00:23:26.341306  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54456 -> 192.168.153.129:22
04/02-00:23:26.341480  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54470 -> 192.168.153.129:22
04/02-00:23:26.342633  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54456 -> 192.168.153.129:22
04/02-00:23:26.347067  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54520 -> 192.168.153.129:22
04/02-00:23:26.283222  [**] [1:100004:1] Possible SSH Brute Force Attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {T
CP} 192.168.153.130:54520 -> 192.168.153.129:22
04/02-00:23:26.347473  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54520 -> 192.168.153.129:22
04/02-00:23:26.353739  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54462 -> 192.168.153.129:22
04/02-00:23:26.354068  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54520 -> 192.168.153.129:22
04/02-00:23:26.358224  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54456 -> 192.168.153.129:22
04/02-00:23:26.373492  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54476 -> 192.168.153.129:22
04/02-00:23:26.373814  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54476 -> 192.168.153.129:22
04/02-00:23:26.378042  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54496 -> 192.168.153.129:22
04/02-00:23:26.378104  [**] [1:100002:1] SSH Authentication Attempt Detected !! [**] [Priority: 0] {TCP} 192.168.153.130:54476 -> 192.168.153.129:22
```

5.3 Improper Input Validation Detection

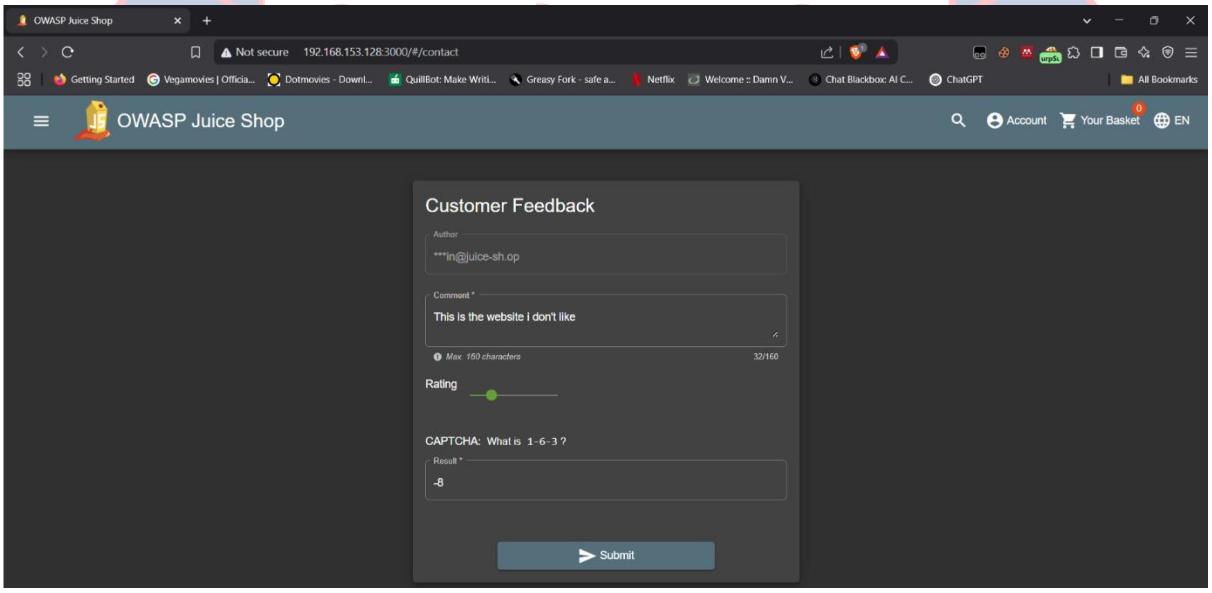
Purpose of the Rule
This rule detects improper input validation in applications, specifically when a user modifies the rating parameter in an HTTP request. If an application is not properly validating user input, attackers can manipulate data, potentially leading to business logic flaws or security vulnerabilities.

alert tcp any any -> any 3000 (msg:"Suspicious Improper Input Validation Detected"; content:"\"rating\":"; pcre:"/\"rating\":\s*[3-9]/"; nocase; sid:1000001; rev:2;)

- **content:""rating":"** → Looks for the rating field in an HTTP request.
- **pcre:"/"rating":\s[3-9]/"*** → Uses a regular expression (RegEx) to detect if the rating value is 3 or higher
- **\s*** → Allows for any amount of whitespace before the number.
- **[3-9]** → Detects ratings from 3 to 9 (modified values).
- **nocase** → Makes the detection case insensitive.

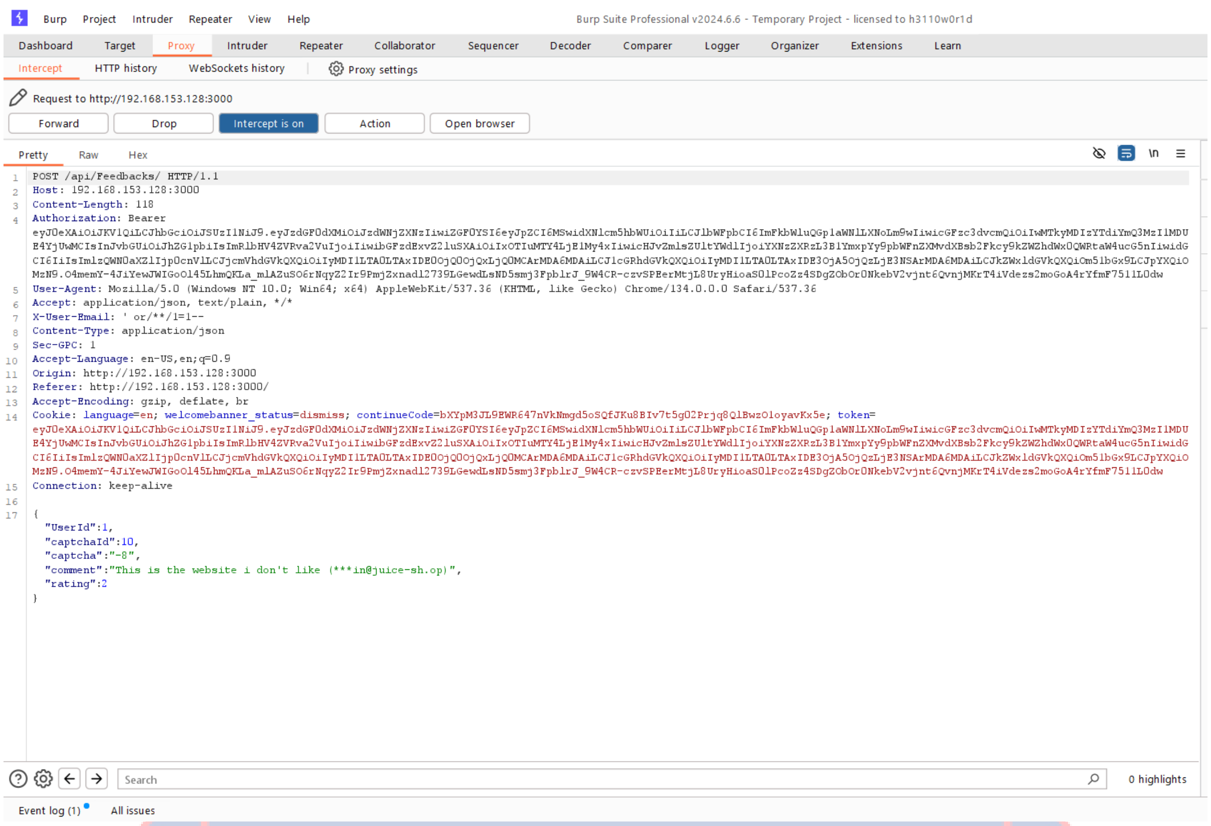
Practical Steps

- Open the Juice Shop web app in a browser and navigate to the Customer Feedback Form.



Customer Feedback
Author
***in@juice-sh.op
Comment *
This is the website i don't like
Max. 160 characters 32/160
Rating
CAPTCHA: What is 1-6-3 ?
Result *
-8
Submit

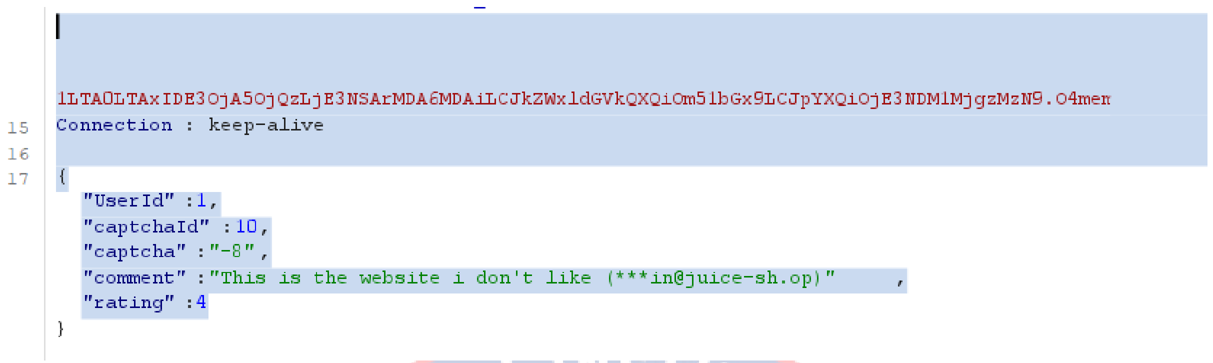
- Make sure Burp Suite (or any HTTP proxy tool) is running in the background with interception enabled.
- Fill in the feedback form with your details and click Submit.
  The request will be intercepted by Burp Suite before it reaches the server.


Burp Project Intruder Repeater View Help
Burp Suite Professional v2024.6.6 - Temporary Project - licensed to h3110w0r1d
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
\n
Search
0 highlights
Event log (1) All issues

Request to http://192.168.153.128:3000

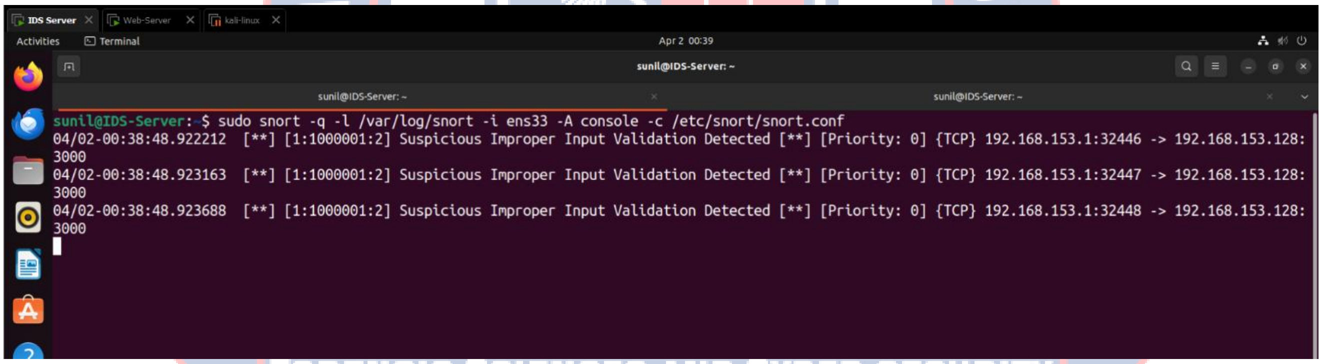
```
POST /api/Feedbacks/ HTTP/1.1
Host: 192.168.153.128:3000
Content-Length: 118
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDU
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG
CI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxIDE0OjQ0OjQxLjQ0MCArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxIDE3OjA5OjQzLjE3NSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiO
MzN9.O4memY-4JiYewJWIGoOl45LhmQKLa_mlAZuSO6rNqyZ2Ir9PmjZxnadl2739LGewdLsND5smj3FpblrJ_9W4CR-czvSPEerMtjL8UryHioaSOlPcoZz4SDgZObOrONkebV2vjnt6QvnjMKrT4iVdezs2moGoA4rYfmF751lLOdw
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: ' or/**/1=1--
Content-Type: application/json
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.153.128:3000
Referer: http://192.168.153.128:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; continueCode=bXYpM3JL9EWR647nVkNmgd5oSQfJKu8BIv7t5gO2Prjq8QlBwzOloyavKx5e; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDU
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG
CI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxIDE0OjQ0OjQxLjQ0MCArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxIDE3OjA5OjQzLjE3NSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiO
MzN9.O4memY-4JiYewJWIGoOl45LhmQKLa_mlAZuSO6rNqyZ2Ir9PmjZxnadl2739LGewdLsND5smj3FpblrJ_9W4CR-czvSPEerMtjL8UryHioaSOlPcoZz4SDgZObOrONkebV2vjnt6QvnjMKrT4iVdezs2moGoA4rYfmF751lLOdw
Connection: keep-alive

{
  "UserId":1,
  "captchaId":10,
  "captcha":"-8",
  "comment":"This is the website i don't like (***in@juice-sh.op)",
  "rating":2
}
```

- In the intercepted HTTP request, you'll observe parameters such as: userId, comment, rating
- Change the rating value from a low value (e.g., 2) to a higher one (e.g., 4) directly in the intercepted request.
- After editing, forward the request to the server.



```
1LTA0LTAxIDE3OjA5OjQzLjE3NSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3NDM1MjgzMzN9.O4men
Connection : keep-alive

{
  "UserId" :1,
  "captchaId" :10,
  "captcha" :"-8",
  "comment" :"This is the website i don't like (***in@juice-sh.op)"     ,
  "rating" :4
}
```

- Snort, running in the background with appropriate detection rules, flags the altered request as suspicious.
- An alert is triggered by indicating "Improper Input Validation", identifying the attempt to bypass normal application logic via parameter tampering.



```
sunil@IDS-Server:~$ sudo snort -q -l /var/log/snort -i ens33 -A console -c /etc/snort/snort.conf
04/02-00:38:48.922212  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32446 -> 192.168.153.128:
3000
04/02-00:38:48.923163  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32447 -> 192.168.153.128:
3000
04/02-00:38:48.923688  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32448 -> 192.168.153.128:
3000
```

5.4 SQL Injection Detection

Purpose of the Rule

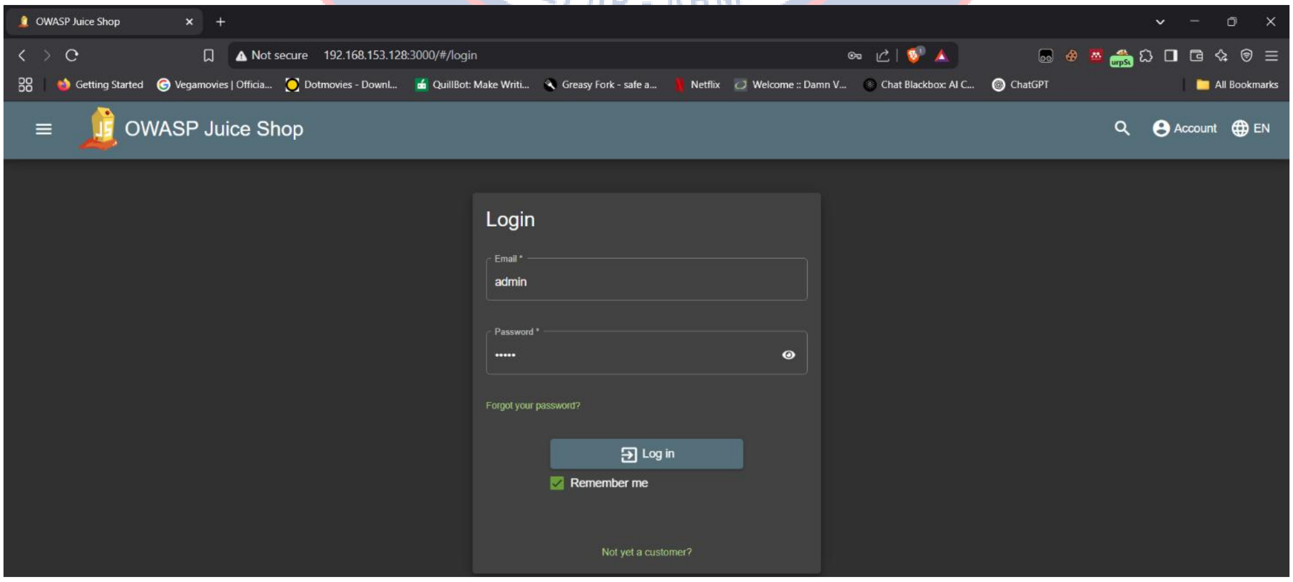
This rule detects SQL Injection (SQLi) attempts, specifically Boolean-based SQL injection where an attacker manipulates the email parameter in an HTTP POST request. SQLi attacks can allow attackers to bypass authentication, extract sensitive data, or manipulate database queries.

alert tcp any any -> 192.168.153.128 3000 (msg:"SQL Injection Attempt Detected"; flow:to_server,established; content:"POST"; http_method; content:"email"; http_client_body; content:"or 1=1"; nocase; sid:1000011; rev:3;)

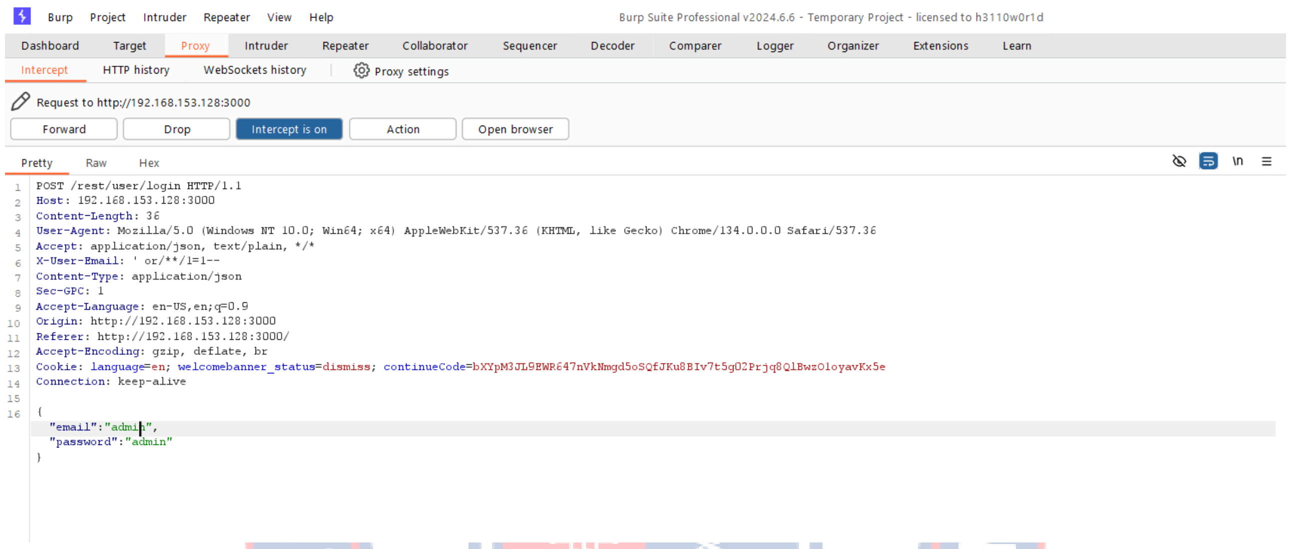
- **content:"POST"; http_method** → Matches HTTP POST requests, commonly used for form submissions.
- **content:"email"; http_client_body** → Ensures the request body contains the email parameter, which is being targeted.
- **content:"or 1=1"; nocase** → Detects the classic SQL injection payload or 1=1, which always evaluates to true, allowing attackers to bypass authentication.

Practical Steps

- Start the Juice Shop application in your browser and navigate to the Login form.
- Enter the default credentials such as admin:admin.


OWASP Juice Shop
Not secure 192.168.153.128:3000/#/login
Login
Email *
admin
Password *
•••••
Forgot your password?
Log in
Remember me
Not yet a customer?

- With Burp Suite (or similar proxy tool) running, submit the login form again to capture the request.
- In the captured request, you'll see parameters like:

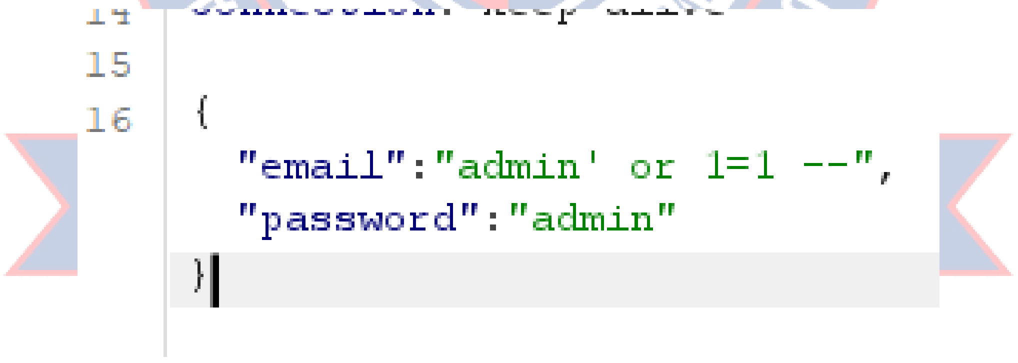

Burp Project Intruder Repeater View Help
Burp Suite Professional v2024.6.6 - Temporary Project - licensed to h3110w0r1d
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex

Request to http://192.168.153.128:3000

```
POST /rest/user/login HTTP/1.1
Host: 192.168.153.128:3000
Content-Length: 36
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: ' or/**/1=1--
Content-Type: application/json
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.153.128:3000
Referer: http://192.168.153.128:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; continueCode=bXYpM3JL9EWR647nVkNmgd5oSQfJKu8BIv7t5g02Prjq8QlBwzOloyavKx5e
Connection: keep-alive

{
  "email":"admin",
  "password":"admin"
}
```

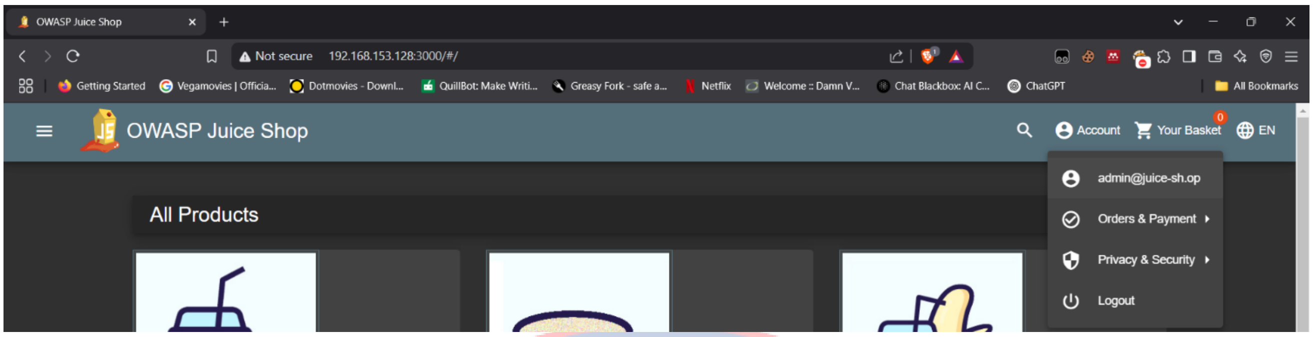
- Replace the email value with a classic SQL injection payload like:

email=' OR 1=1—

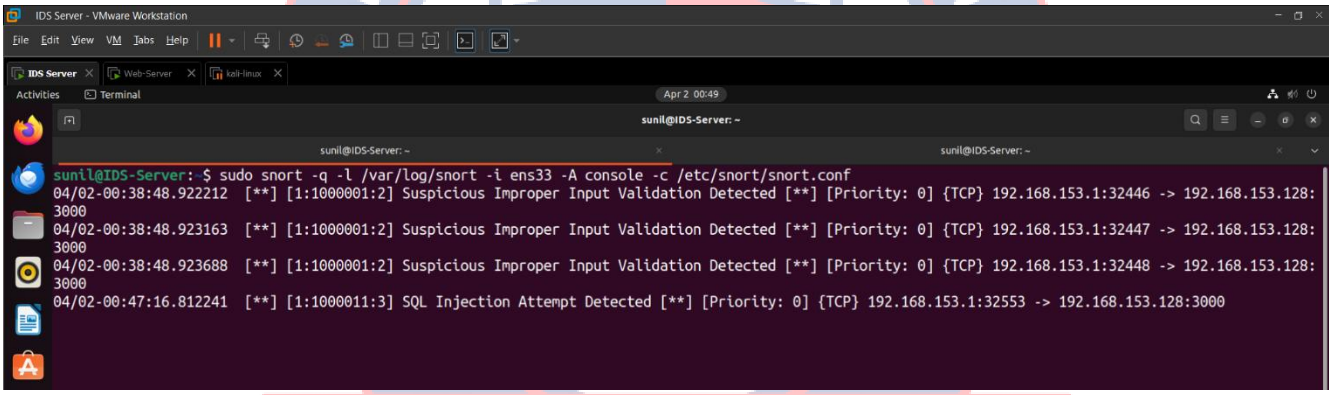


```
{
  "email":"admin' or 1=1 --",
  "password":"admin"
}
```

- Keep the password value anything (it will be ignored due to the injection). Then forward the modified request to the server.
- You'll notice the app logs you in as the administrator without needing the actual password. This confirms a successful SQL injection attack.



- With Snort running in the background, the malicious payload (' OR 1=1--) is detected and flagged as an intrusion.
- This helps identify that someone is trying to exploit SQL injection vulnerabilities on the web server.



```
sunil@IDS-Server:~$ sudo snort -q -l /var/log/snort -i ens33 -A console -c /etc/snort/snort.conf
04/02-00:38:48.922212  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32446 -> 192.168.153.128:
3000
04/02-00:38:48.923163  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32447 -> 192.168.153.128:
3000
04/02-00:38:48.923688  [**] [1:1000001:2] Suspicious Improper Input Validation Detected [**] [Priority: 0] {TCP} 192.168.153.1:32448 -> 192.168.153.128:
3000
04/02-00:47:16.812241  [**] [1:1000011:3] SQL Injection Attempt Detected [**] [Priority: 0] {TCP} 192.168.153.1:32553 -> 192.168.153.128:3000
```

5.5 Unauthorized File Upload Detection

Purpose of the Rule

This rule detects unauthorized file uploads, where users attempt to upload files other than PDF or ZIP. Attackers often upload malicious files (e.g., .xml, .php, .exe) to exploit vulnerabilities in web applications, leading to remote code execution, data leaks, or privilege escalation.

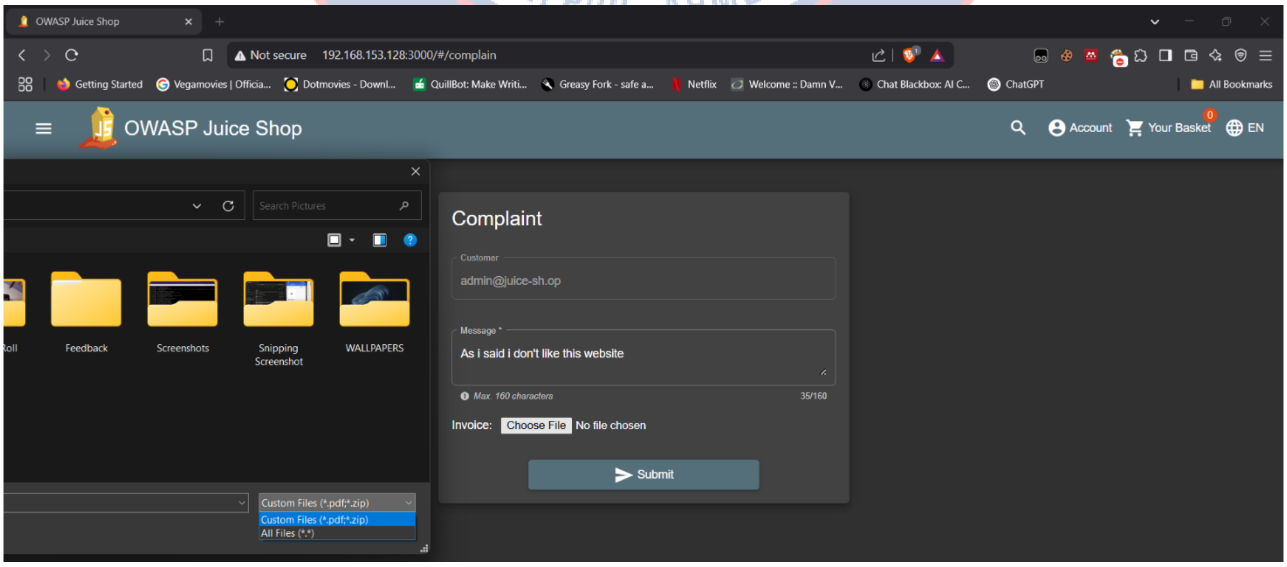
**alert tcp any any -> any any (msg:"Unauthorized file upload detected"; content:"Content-Type:";
nocase; pcre:"/Content-Type:\s*(?!application\/pdf|application\/zip)/";
sid:100002; rev:1;)**

- **pcre:"/Content-Type:\s(?!application/pdf|application/zip)/"*** → Uses a regular expression (RegEx) to check the file type.

- **(?!application\/pdf|application\/zip)** → Negative lookahead to detect files other than PDF or ZIP.

Practical Steps

- Open the Juice Shop web application and navigate to the Complaint Form section.
- The file upload form only allows: **.pdf** files, **.zip** files
- However, the attacker attempts to upload a .xml file (e.g., test.xml) which may contain a malicious payload.



Complaint

Customer: admin@juice-sh.op

Message *: As i said i don't like this website

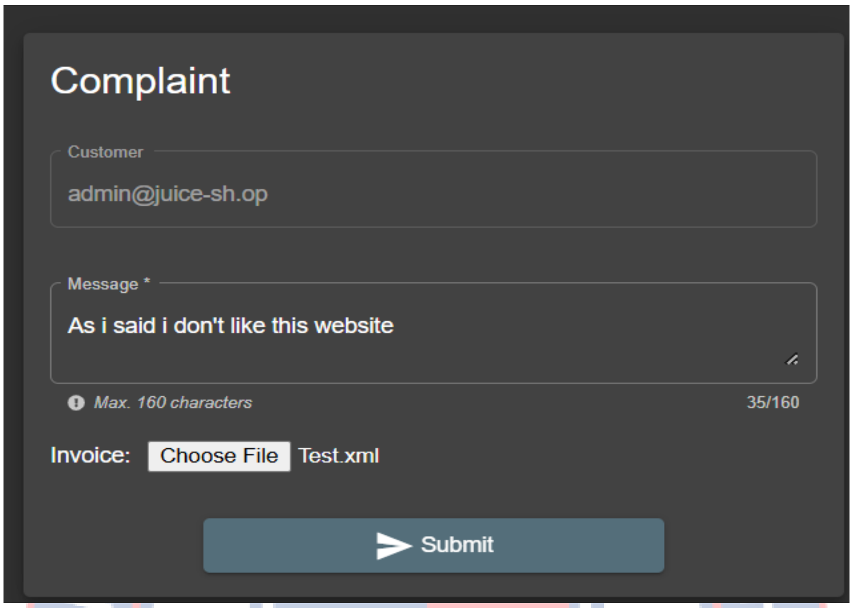
Max. 160 characters 35/160

Invoice: Choose File No file chosen

Submit

Custom Files (*.pdf;*.zip)
Custom Files (*.pdf;*.zip)
All Files (*.*)

- With Burp Suite running in the background, select test.xml and submit the form.



Complaint

Customer
admin@juice-sh.op

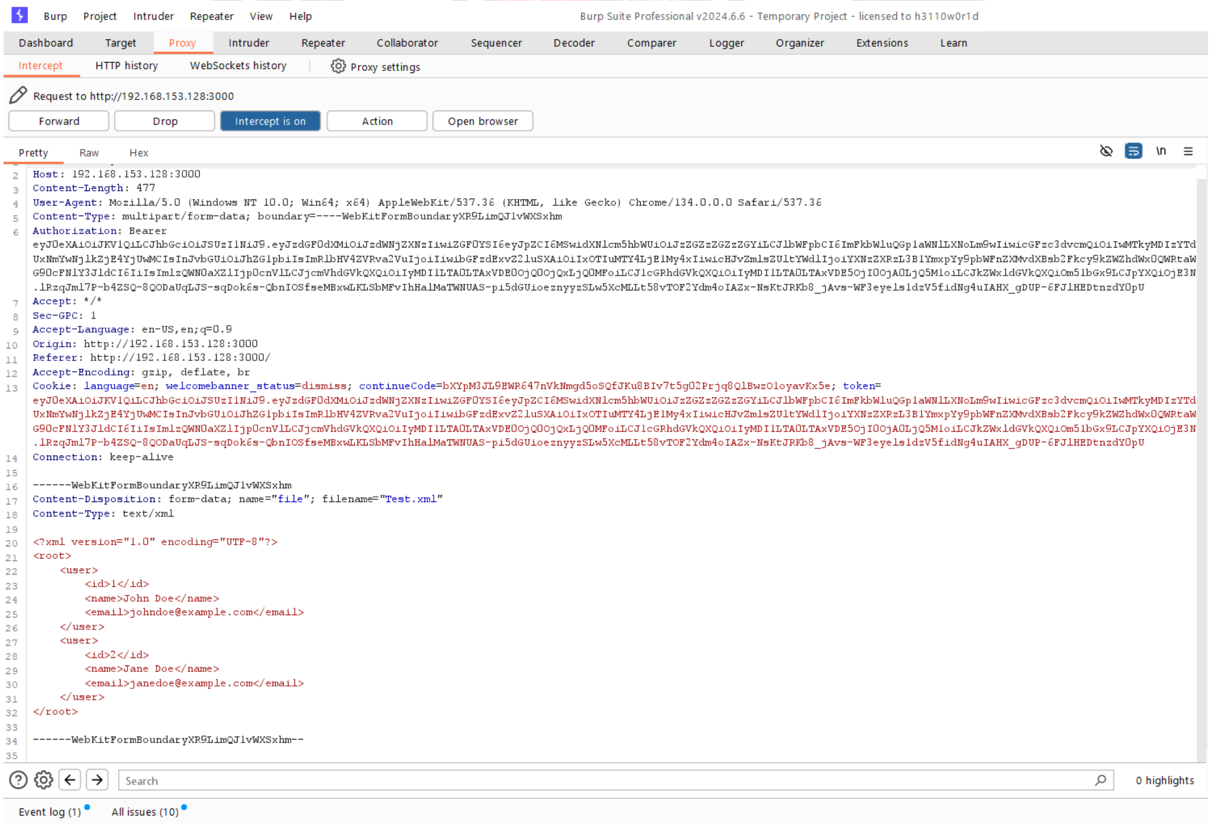
Message *
As i said i don't like this website

Max. 160 characters 35/160

Invoice: Choose File Test.xml

Submit

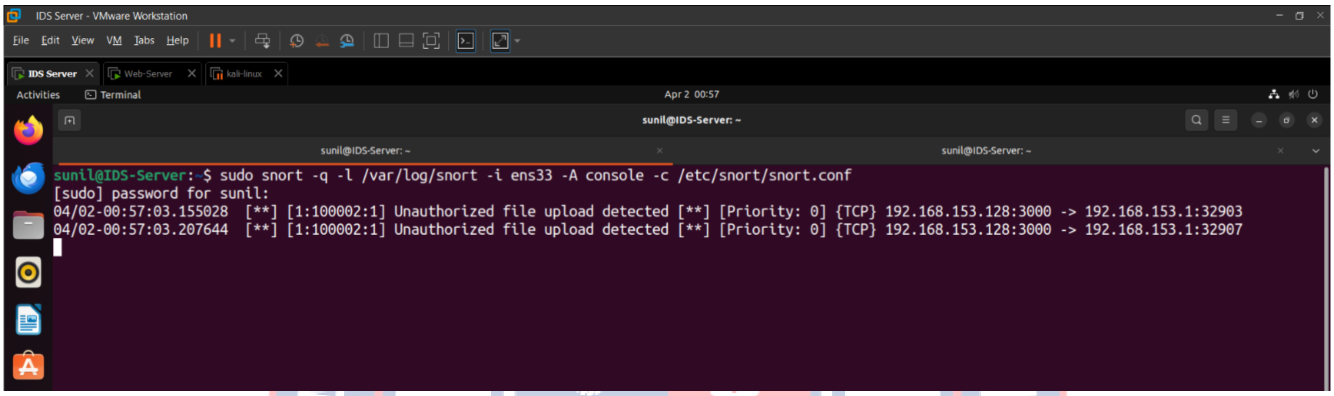
- Burp Suite intercepts the multipart HTTP POST request, revealing the contents of the XML file and the request headers.


Burp Project Intruder Repeater View Help
Burp Suite Professional v2024.6.6 - Temporary Project - licensed to h3110w0r1d
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
Search
0 highlights
Event log (1) All issues (10)

Request to http://192.168.153.128:3000

```
Host: 192.168.153.128:3000
Content-Length: 477
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryXR9LimQJ1vWXSxhm
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJzZGZzZGZzZGYiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTd
UxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW
G90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxVDE0OjQ0OjQxLjQ0MFoiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxVDE5OjI0OjA0LjQ5MloiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3N
.lRzqJml7P-b4ZSQ-8QODaUqLJS-sqDok6s-QbnIOSfseMBxwLKLSbMFvIhHalMaTWNUAS-pi5dGUioeznyyzSLw5XcMLLt58vTOF2Ydm4oIAZx-NsKtJRKb8_jAvs-WF3eyels1dzV5fidNg4uIAHX_gDUP-6FJlHEDtnzdY0pU
Accept: */*
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.153.128:3000
Referer: http://192.168.153.128:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; continueCode=bXYpM3JL9EWR647nVkNmgd5oSQfJKu8BIv7t5gO2Prjq8QlBwzO1oyavKx5e; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJzZGZzZGZzZGYiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTd
UxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW
G90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxVDE0OjQ0OjQxLjQ0MFoiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxVDE5OjI0OjA0LjQ5MloiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3N
.lRzqJml7P-b4ZSQ-8QODaUqLJS-sqDok6s-QbnIOSfseMBxwLKLSbMFvIhHalMaTWNUAS-pi5dGUioeznyyzSLw5XcMLLt58vTOF2Ydm4oIAZx-NsKtJRKb8_jAvs-WF3eyels1dzV5fidNg4uIAHX_gDUP-6FJlHEDtnzdY0pU
Connection: keep-alive

------WebKitFormBoundaryXR9LimQJ1vWXSxhm
Content-Disposition: form-data; name="file"; filename="Test.xml"
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<root>
    <user>
        <id>1</id>
        <name>John Doe</name>
        <email>johndoe@example.com</email>
    </user>
    <user>
        <id>2</id>
        <name>Jane Doe</name>
        <email>janedoe@example.com</email>
    </user>
</root>

------WebKitFormBoundaryXR9LimQJ1vWXSxhm--
```

- Click Forward in Burp Suite to send the request to the server.
- The web server accepts and uploads the XML file successfully, even though it violates the allowed file type policy.
- Snort, running on the network, analyses the request and identifies the .xml upload as unauthorized or suspicious.



```
sunil@IDS-Server:~$ sudo snort -q -l /var/log/snort -i ens33 -A console -c /etc/snort/snort.conf
[sudo] password for sunil:
04/02-00:57:03.155028  [**] [1:100002:1] Unauthorized file upload detected [**] [Priority: 0] {TCP} 192.168.153.128:3000 -> 192.168.153.1:32903
04/02-00:57:03.207644  [**] [1:100002:1] Unauthorized file upload detected [**] [Priority: 0] {TCP} 192.168.153.128:3000 -> 192.168.153.1:32907
```

- This indicates potential misuse of the file upload functionality for delivering malicious files to the server.
- This vulnerability can lead to **Remote Code Execution (RCE)**, **XEE attacks**, or other serious risks if exploited.

Chapter 6: Evasion Techniques

Introduction

Attackers often try to bypass Intrusion Detection Systems (IDS) by using special tricks or methods called evasion techniques. These techniques are designed to hide or disguise malicious traffic so that the IDS fails to detect it. Even though tools like Snort are powerful, they can sometimes be fooled if the attack is modified in a smart way.

Evasion techniques exploit weaknesses in how an IDS analyses packets. By changing the way data is sent—without changing what it does—attackers can sneak past security systems undetected.

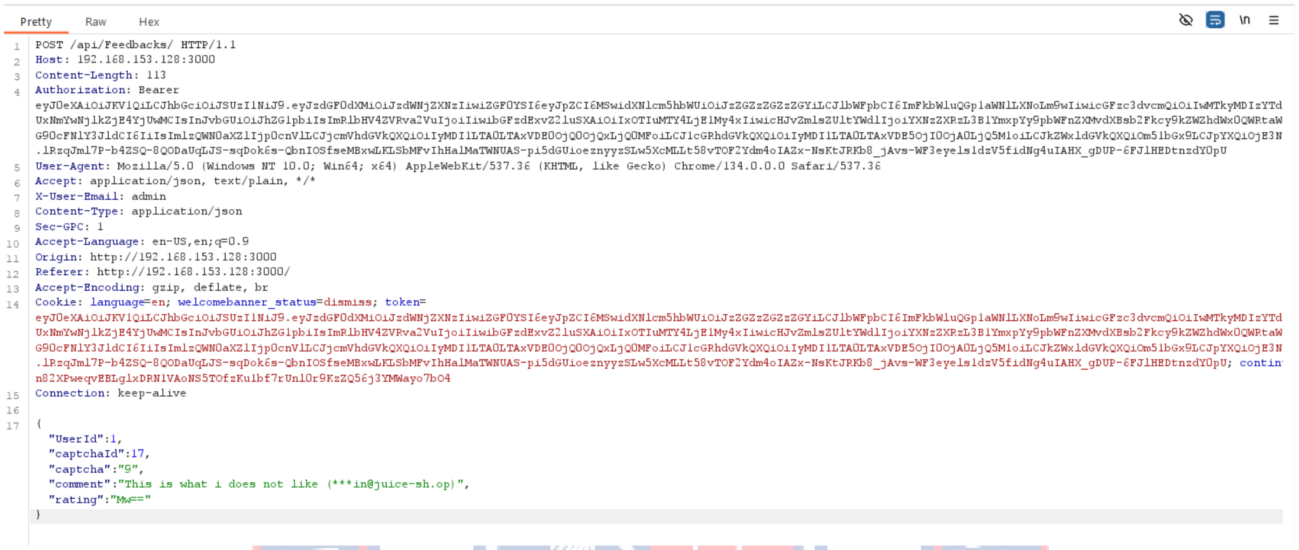
6.1 Base64 Encoding to Bypass Detection

Base64 encoding is a technique used to convert data into a set of readable ASCII characters. It is often used in data transmission to encode binary data into a format that can be easily sent through text-based protocols like HTTP.

However, attackers can use Base64 encoding to hide malicious inputs in a way that makes it difficult for signature-based IDS (like Snort) to detect them. Since the content is encoded, Snort may not recognize the attack pattern unless it has been specifically configured to decode and analyse Base64 content.

Instead of sending raw values in JSON parameters, attackers encode them using Base64. Since Snort's rule is searching for rating values between 3-9, encoding the value hides it from simple content-based detection.

- In this project, we tested how Snort responds to an HTTP request where a numeric rating value was encoded using Base64.



```
POST /api/Feedbacks/ HTTP/1.1
Host: 192.168.153.128:3000
Content-Length: 113
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJzZGZzZGZzZGYiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTd
UxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW
G90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxVDE0OjQ0OjQxLjQ0MFoiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxVDE5OjI0OjA0LjQ5MloiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3N
.lRzqJml7P-b4ZSQ-8QODaUqLJS-sqDok6s-QbnIOSfseMBxwLKLSbMFvIhHalMaTWNUAS-pi5dGUioeznyyzSLw5XcMLLt58vTOF2Ydm4oIAZx-NsKtJRKb8_jAvs-WF3eyels1dzV5fidNg4uIAHX_gDUP-6FJlHEDtnzdY0pU
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: admin
Content-Type: application/json
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.153.128:3000
Referer: http://192.168.153.128:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJzZGZzZGZzZGYiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTd
UxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxOTIuMTY4LjE1My4xIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW
G90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTAxVDE0OjQ0OjQxLjQ0MFoiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTAxVDE5OjI0OjA0LjQ5MloiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3N
.lRzqJml7P-b4ZSQ-8QODaUqLJS-sqDok6s-QbnIOSfseMBxwLKLSbMFvIhHalMaTWNUAS-pi5dGUioeznyyzSLw5XcMLLt58vTOF2Ydm4oIAZx-NsKtJRKb8_jAvs-WF3eyels1dzV5fidNg4uIAHX_gDUP-6FJlHEDtnzdY0pU; contin
n82XPweqvEBLglxDRNlVAoNS5TOfzKulbf7rUnl0r9KzZQ56j3YMWayo7bO4
Connection: keep-alive

{
  "UserId":1,
  "captchaId":17,
  "captcha":"9",
  "comment":"This is what i does not like (***in@juice-sh.op)",
  "rating":"Mw=="
}
```

- Snort was unable to detect this as an improper input because the pattern "rating": 5 did not appear in plain text.

6.2 SQL Injection Obfuscation (Using Comments and Encoding)

SQL Injection (SQLi) is a technique where attackers insert or “inject” malicious SQL queries into input fields to bypass authentication, access sensitive data, or manipulate databases.

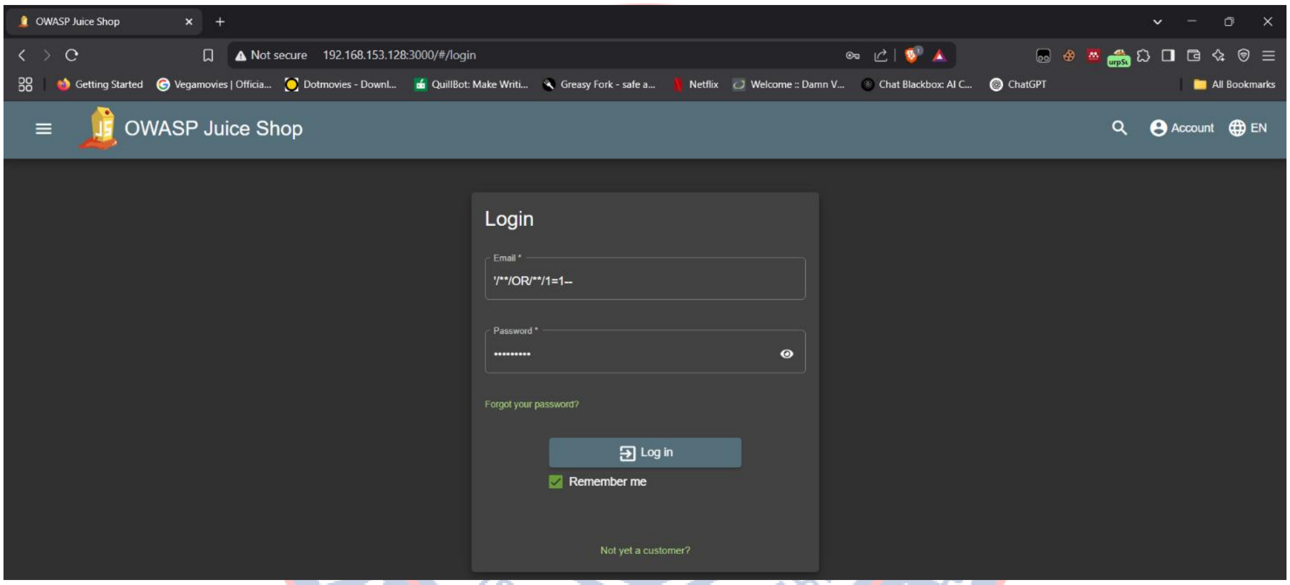
While Snort can detect standard SQL injection patterns like or 1=1, attackers often use obfuscation techniques to bypass such detection.

One common obfuscation method is to insert SQL comments or use encoding to break up the pattern so that the IDS doesn’t recognize it — even though the database still understands the query and executes it successfully.

SQL supports inline comments (/**/), allowing attackers to break up the query to bypass string-based detection.

The Snort rule looking for "or 1=1" won't match because the content is split. The database still executes the query correctly.

- In our project, we simulated an SQL injection attack against the Juice Shop login form by sending a POST request with an obfuscated payload using:



OWASP Juice Shop

Not secure 192.168.153.128:3000/#/login

Login

Email *
'/**/OR/**/1=1--

Password *
•••••••••

Forgot your password?

Log in

✓ Remember me

Not yet a customer?

- Despite the rule for "or 1=1" being active, Snort failed to detect the obfuscated version, demonstrating that basic content-matching rules are not enough to detect well-crafted attacks.

Chapter 7: Recommendation & Future Work

7.1 Enhancing Snort Rules for Better Coverage

Current Limitation:
Many Snort rules rely on basic pattern matching using specific keywords. These rules can miss modified or obfuscated attack payloads.
Recommendation:

- Use regular expressions (PCRE) in rules instead of simple content matches to catch variations of attack patterns.
- Implement threshold rules to detect brute-force attacks over time.
- Continuously update and test rule sets with real-world attack samples.
- Use multiple rules to cover the same attack from different angles (e.g., URL, headers, body).

Outcome:
Improved detection of both known and slightly modified attacks, reducing false negatives.

7.2 Integration of Machine Learning for Anomaly Detection

Current Limitation:
Signature-based IDS like Snort cannot detect unknown or zero-day attacks unless a rule already exists.

Recommendation:
- Combine Snort with machine learning (ML) models that learn normal network behavior and detect deviations.
- Train models use features such as IP, port, request size, frequency, etc.
- Use tools like Scikit-learn, TensorFlow, or PyOD for building anomaly detectors.

Outcome:
Better detection of new threats without needing pre-defined rules. ML adds a second layer of defense.

7.3 Dashboard Design for Real-Time Monitoring (ELK Stack)

Current Limitation:
Snort logs are in text files, which are hard to read and analyse quickly.

Recommendation:

- Integrate Snort with tools like the ELK Stack (Elasticsearch, Logstash, Kibana) or Grafana + Loki.
- Set up a dashboard that shows alerts, attack types, time stamps, and IP addresses in visual format.
- Use filtering and search to quickly find critical events.

Outcome:
Improved visibility and faster response to incidents. Easy to monitor and report security events in real-time.

Chapter 8: Conclusion

- This project aimed to evaluate the effectiveness of the Snort Intrusion Detection System (IDS) in detecting various types of network attacks and understanding how attackers use evasion techniques to bypass detection.

- Several custom rules were written and tested for attacks such as SSH brute-force, SQL injection, improper input validation, unauthorized file uploads, and reflected XSS. Tools like TCPDump helped in capturing traffic, and Hydra was used to simulate brute-force attacks.

- The testing showed that Snort performs well against direct and well-known attack patterns, especially when clear and specific rules are applied. However, it also became clear that simple evasion techniques, like using Base64 encoding or adding comments in SQL queries, can bypass these rules if they are not carefully written.

- This highlights the importance of updating Snort rules regularly, using regex-based detection, and applying threshold filters for improved accuracy. Also, adding visualization tools and machine learning techniques can greatly enhance Snort's detection capabilities.

- In conclusion, while Snort is a powerful and flexible IDS, it must be carefully configured and continuously improved to deal with modern cyber threats. It is best used as part of a layered security system to protect networks more effectively.

Chapter 9: References

1. **Martin Roesch,** ***Snort - Lightweight Intrusion Detection for Networks*****,** Proceedings of LISA '99: 13th Systems Administration Conference, 1999. **https://www.snort.org**

2. **Thomas H. Ptacek & Timothy N. Newsham,** ***Insertion, Evasion, and Denial of Service***: *Eluding Network Intrusion Detection*, Secure Networks Inc., 1998. **https://insecure.org/stf/secnet_ids/secnet_ids.html**

3. **Toby Kohlenberg,** ***Snort Intrusion Detection and Prevention Toolkit***, Syngress Publishing, 2007.

4. **Nattawat Khamphakdee & Nunnapus Benjamas,** ***Improving Intrusion Detection System based on Snort Rules for Network Probe Attack Detection*****,** Journal of Advances in Information Technology, 2021.

5. Snort Official Documentation, Cisco Systems. **https://docs.snort.org**

6. Hydra (THC-Hydra) Documentation, The Hacker's Choice. **https://github.com/vanhauser-thc/thc-hydra**

7. TCPDump Manual Pages **https://man7.org/linux/man-pages/man8/tcpdump.8.html**

8. Regex101 – Online Regex Tester and Debugger **https://regex101.com/**