



# IDS SELECTION, IMPLEMENTATION, AND TESTING REPORT

Redback Operations

Document Owner: Secure Coding Team  
Deployed By: Mehak and Tushar Sharma

Last Modified By: Mehak

Last Modified on: 18 May 2024

## Contents

<b>Abstract</b> .....	4
<b>1. Introduction</b> .....	5
<b>2. What is an Intrusion Detection System (IDS)?</b> .....	5
<b>3. Types of IDS</b> .....	5
<b>3.1. Network-Based IDS (NIDS)</b> .....	5
<b>3.2. Host-Based IDS (HIDS)</b> .....	5
<b>3.3 Signature-Based IDS</b> .....	6
<b>3.4 Anomaly-Based IDS</b> .....	6
<b>4. Why we need IDS:</b> .....	6
<b>4.1. External Cyber Threats:</b> .....	6
<b>4.2. Insider Threats:</b> .....	6
<b>4.3. Software Vulnerabilities:</b> .....	6
<b>4.4 Emerging Cyber Threats:</b> .....	7
<b>5. Components of IDS Implementation</b> .....	7
<b>5.1. Data Collection</b> .....	7
<b>5.2. Data Analysis</b> .....	8
<b>5.3. Detection Algorithms</b> .....	8
<b>5.4. Alert Generation and Management</b> .....	8
<b>5.5. Response and Mitigation</b> .....	9
<b>6. Existing Intrusion Detection Systems</b> .....	9
<b>6.1. Snort</b> .....	9
<b>6.2. Suricata</b> .....	9
<b>6.3. Zeek</b> .....	10
<b>6.4. Nagios</b> .....	10
<b>6.5. Splunk</b> .....	11
<b>6.6 OSSEC</b> .....	11
<b>7. Comparison of Intrusion Detection System Solutions for Redback Operations</b> .....	12
<b>7.1 Side-by-Side Comparison of existing IDS Solutions</b> .....	12
<b>7.2 Cost Requirements of each IDS Solution:</b> .....	12
7.2.1 Suricata and Snort: .....	12
7.2.2 Suricata with ELK Stack Integration: .....	13
7.2.3 OSSEC: .....	13
7.2.4 Zeek (formerly Bro): .....	14
7.2.5 Snort with Barnyard2 and BASE: .....	14
<b>8. Selection of IDS Solution: Suricata</b> .....	15

<b>9. Deploying Suricata on the Virtual Machine .....</b>	<b>16</b>
<b>9.1 Accessing the VM via Terminal: Step-by-Step Guide .....</b>	<b>16</b>
<b>9.2 Steps Followed to Configure and Integrate Suricata .....</b>	<b>18</b>
<b>9.3 Future Recommendations: For T2 2024 .....</b>	<b>25</b>
<b>10. Conclusion .....</b>	<b>27</b>
Figure 1 connecting to Deakin VPN .....	16
Figure 2 connected to Deakin VPN .....	17
Figure 3 running the ifconfig command .....	19
Figure 4 first edit in the .yaml file .....	20
Figure 5 second edit in the .yaml file.....	21
Figure 6 third edit in the .yaml file .....	21
Figure 7 fourth and crucial edit in the .yaml file .....	22
Figure 8 fifth edit in the .yaml file.....	23
Figure 9 testing Suricata .....	24
Figure 10 successful detection of traffic on Suricata.....	24

Name	Team	Role	Position
Mehak	Secure Coding Team	Research, Implementation and Documentation	Senior
Tushar Sharma	Secure Coding Team	Implementation and Documentation	Senior

## Abstract

Intrusion Detection Systems (IDS) play a pivotal role in safeguarding the security and integrity of Redback Operations' connected fitness devices. As a secure coding team member, understanding IDS principles and best practices is imperative. This research document provides a comprehensive overview of IDS, encompassing its various types, deployment strategies, and recommendations for seamless integration into Redback's projects. By delving into the intricacies of IDS implementation, this document aims to equip Redback with the necessary tools to fortify its software systems against potential cyber threats and uphold the trust of its users.

## 1. Introduction

Redback Operations, a prominent capstone project company within the esteemed School of Information Technology at Deakin University, is dedicated to pioneering cutting-edge connected fitness devices. With a steadfast commitment to safeguarding user data and ensuring product integrity, we embark on a journey to explore and implement robust security measures.

At the forefront of our strategic initiatives lies the proposition to integrate an Intrusion Detection System (IDS). This proactive stance underscores our unwavering dedication to cybersecurity and exemplifies our proactive approach towards mitigating potential threats. By fortifying our defences with an IDS, we not only mitigate risks but also enhance the safety, reliability, and peace of mind associated with our connected fitness technology.

## 2. What is an Intrusion Detection System (IDS)?

An IDS (Intrusion Detection System) is a security mechanism that monitors network traffic, system logs, and user behaviour to detect and respond to unauthorized or malicious activities. It guards by continuously scanning for anomalies and potential security breaches that could compromise the three pillars of cybersecurity which are confidentiality, integrity, and availability of data. Network Administrator can easily detect any malicious activity on the network with the help of IDS. An intruder access and tries to find any useful information and means to unauthorised access for the information inside the network, which can cause disrupt the network or can misuse that information in malicious activities.

By employing algorithms and pattern recognition techniques, Intrusion Detection System can identify the patterns indicating cyber threats and can generate real-time alert with prompt actions. This proactive approach not only enables the detection of external attacks but also helps in uncovering insider threats, thereby ensuring comprehensive security coverage.

Given the nature of our connected fitness devices, IDS serves as a vigilant guardian, continuously monitoring for any suspicious activities that may compromise user data or device functionality. In the context of Redback's connected fitness devices, an IDS plays a pivotal role in upholding our commitment to user privacy and data security, reinforcing our position as a trusted provider of innovative fitness solutions.

## 3. Types of IDS

### 3.1. Network-Based IDS (NIDS)

- Monitors the flow of network traffic at key points (e.g., routers, switches).
- Analyzes packets for anomalies, known attack patterns, and signatures.
- Useful for detecting external threats.

### 3.2. Host-Based IDS (HIDS)

- Installed on individual devices (servers, workstations).
- Monitors system logs, file integrity, and user activities.

- Effective for detecting insider threats.

### 3.3 Signature-Based IDS

- Detect the suspicious activities by existing attack patterns.
- Attack signatures are stored in database.
- System detects if the pattern matches with database, then generate the alert.
- Effective for known threats but fails to detect any new emerging attack.

### 3.4 Anomaly-Based IDS

- Detect the suspicious activities by looking for abnormal behaviour.
- System adds the baseline of normal behaviour and then generate the alert when they detect crossing the baseline.
- Effective for Insider attack and zero-day attacks.

## 4. Why we need IDS:

The threats and risks Redback Operations may face in today's world of hackers and how an Intrusion Detection System (IDS) can mitigate them:

### 4.1. External Cyber Threats:

**Threat:** Redback's connected fitness devices and digital infrastructure are vulnerable to external cyber threats such as unauthorized access, malware attacks, and data breaches.

**Risk:** These threats could compromise user data, disrupt services.

**Mitigation with IDS:** An IDS continuously monitors network traffic and system logs for signs of malicious activity, enabling early detection and response to external threats. By promptly identifying and mitigating security incidents, an IDS helps prevent data breaches and service disruptions, thereby safeguarding Redback's assets and reputation.

### 4.2. Insider Threats:

**Threat:** Internal users can unintentionally or maliciously compromise Redback's systems and data.

**Risk:** Insider threats pose a significant risk to the confidentiality, integrity, and availability of Redback's information assets.

**Mitigation with IDS:** An IDS helps detect abnormal user behaviour and unauthorized access attempts, allowing Redback to identify and address insider threats proactively. By monitoring user activity and system logs, an IDS helps detect and mitigate insider threats before they escalate into serious security incidents.

### 4.3. Software Vulnerabilities:

**Threat:** Software bugs and vulnerabilities in Redback's applications and systems may be exploited by attackers to gain unauthorized access or execute malicious code.

**Risk:** Exploiting software vulnerabilities can lead to data breaches, service disruptions, Application Layer attacks.

**Mitigation with IDS:** An IDS can detect and alert Redback's security team to suspicious network activity indicative of attempts to exploit software vulnerabilities. By identifying and responding to such threats promptly, an IDS helps mitigate the risk posed by software vulnerabilities and strengthens Redback's overall security posture.

#### 4.4 Emerging Cyber Threats:

**Threat:** The cybersecurity landscape is constantly evolving, with attackers employing increasingly sophisticated techniques and tactics to bypass traditional security measures.

**Risk:** Redback may be vulnerable to emerging cyber threats such as Ransomware as Service, Supply chain Attack and Zero-Day Attack that exploit new vulnerabilities or attack vectors.

**Mitigation with IDS:** An IDS equipped with advanced threat detection capabilities helps Redback stay ahead of emerging cyber threats by continuously monitoring for new attack patterns and indicators of compromise. By detecting and responding to emerging threats in real time, an IDS enables Redback to adapt its security defences and mitigate the risk posed by evolving cyber threats effectively.

IDS will provide the protection against cyber threats, safeguarding user data, ensuring product integrity and early threat detection. The main benefit of IDS is that we can monitor network and activities in real time.

### 5. Components of IDS Implementation

IDS systems consist of various components to detect and respond to security threats. Some of the key elements of IDS implementation are:

#### 5.1. Data Collection

Data collection is foundation for IDS's workflow as it provides the necessary information for analysis, to detect the security threats. It consists of several key steps:

- **Network Traffic Capture:** This is where Network-Based IDS (NIDS) examines data packets traveling through the network, typically at critical points like routers and firewalls. By analysing these packets, IDS can identify unusual patterns or suspicious activities.
- **System Log Analysis:** Host-Based IDS (HIDS) focuses on collecting and analysing system logs. This step is crucial for tracking user activities and detecting unusual behaviour, such as unauthorized access or unexpected system changes.
- **Host Monitoring:** This component monitors file integrity and system metrics to identify unauthorized changes. It also tracks CPU usage, memory consumption, and network bandwidth to detect anomalies which adds the extra layer of security.



## 5.2. Data Analysis

Data analysis which is also known as heart of IDS. After gathering data from various sources, the IDS processes and analyses it to detect threats. It works as:

- **Predefined Rule:** This component applies a set of specific rules to identify known patterns of suspicious activity. These rules help the IDS quickly spot behaviours that could indicate a security threat. Some of the rules are:
- **IP-Based Rules:** These rules focus on specific IP addresses or ranges, triggering alerts if there's unexpected communication with known malicious IPs or abnormal traffic from a particular source.
- **Protocol-Based Rules:** These rules monitor network protocols for suspicious behaviour, such as unexpected changes in protocol usage or traffic patterns that deviate from established norms.
- **Rate-Based Rules:** These rules are designed to detect unusually high rates of certain activities, like numerous logins attempts in a short period, which might indicate a brute force attack.
- **Content-Based Rules:** These rules inspect packet content for specific keywords or patterns that could indicate malicious activity, such as SQL injection attempts or cross-site scripting (XSS).

## 5.3. Detection Algorithms

Detection algorithms work by identifying potential malicious activities and security breaches. The two main types are:

- **Signature-Based Detection:** This method compares network traffic patterns to a database of known to attack signatures. It is great for finding common threats, but it may miss new or emerging ones. To use the IDS Signature Package, we can use (<https://www.juniper.net/documentation/us/en/software/junos/idp-policy/topics/topic-map/security-idp-signature-database-for-migration-understanding.html>).
- **Anomaly-Based Detection:** This approach looks for unusual patterns or behaviours using statistical techniques or machine learning. It is effective for spotting unknown threats, but it can sometimes generate false positives if the baseline is not well-defined.

## 5.4. Alert Generation and Management

When an Intrusion Detection System (IDS) identifies suspicious activity, it triggers alerts to prompt immediate response and investigation.

**Real-Time Alert:** This feature sends instant notifications when security breaches or anomalies are detected. Real-time alerts allow security teams to respond quickly, reducing the impact of potential threats.

**Alert Escalation:** In cases where a critical alert is triggered, the system escalates the alert to incident response teams or higher-level security personnel. This ensures that high-priority threats receive the attention they require and are investigated further.

## 5.5. Response and Mitigation

Once an Intrusion Detection System (IDS) generates alerts, the system administrators can take appropriate security actions to address potential threats. This process involves a range of measures aimed at containing, mitigating, or neutralizing the detected risks, which can be Dynamic Packet Filtering, Throttling and Rate Limiting, Disconnecting Compromised Systems.

## 6. Existing Intrusion Detection Systems

There are various open-source IDS platforms which provides collaborative development, providing flexibility and customizability to implement the security system. Some popular examples are:

### 6.1. Snort

Snort is widely used Network-Based IDS (NIDS) which uses a rule-based approach to detect malicious network traffic. Their rules help this system to find packets that checks the malicious activities and will send alert. It has a large community which contributes to its signature database.

**Link:** <https://www.snort.org/>

#### Key Features:

- Analyses network traffic in real-time to detect suspicious activity and potential threats.
- Offers a comprehensive repository of attack signatures contributed by its community, allowing Snort to detect a wide range of known threats.
- Enables users to create custom rules to meet specific security requirements.
- Integrates with various security tools and technologies, providing flexibility in deployment and monitoring.

### 6.2. Suricata

Suricata is a high-performance IDS system with multi-threaded processing capabilities and advanced detection features. It supports both signature-based and anomaly-based detection, and it is also compatible with Snort rule sets. Suricata is designed for large-scale network environments.

**Link:** <https://suricata.io/>

#### Key Features:

- Supports multi-threaded processing, allowing Suricata to handle high traffic volumes with improved performance.
- Offers both signature-based and anomaly-based detection, enabling the detection of a wide range of threats.
- Supports various network protocols, providing deep packet inspection and comprehensive analysis.
- Allows Suricata to use Snort rule sets, providing flexibility and access to a vast repository of attack signatures.

### 6.3. Zeek

Zeek is a network analysis framework with a focus on deep packet inspection for network security monitor. It allows for script-based analysis and integrates with other security tools which provides the flexibility in creating custom detection rules.

**Link:** <https://zeek.org/>

#### **Key Features:**

- Allows users to create custom scripts for specialized detection, offering a high degree of flexibility.
- Analyses network packets at a granular level, providing detailed insights into network activity.
- Supported by an active community that contributes to custom scripts and detection rules.
- Integrates with various security tools and technologies, facilitating a comprehensive security strategy.

### 6.4. Nagios

Nagios is a network monitoring and intrusion detection system designed to track the performance of network devices and services. Nagios is versatile and can be customized to meet various monitoring needs, making it suitable for organizations seeking a comprehensive solution for network monitoring and intrusion detection.

**Link:** <https://www.nagios.com/>

#### **Key Features:**

- Monitors network devices, services, and system resources to ensure optimal performance.
- Generates alerts when network issues or suspicious activities are detected.
- Analyses system logs to detect unusual patterns or behaviours.

## 6.5. Splunk

Splunk's Security Information and Event Management (SIEM) system collects, analyzes, and visualizes data from various security devices, including IDS platforms. It is renowned for its powerful data analytics capabilities and robust alerting features. Splunk can generate real-time alerts and detailed reports, facilitating rapid incident response.

**Link:** <https://www.splunk.com/>

### Key Features:

- Gathers data from multiple security devices and systems, providing a unified view of network activity.
- Allows users to create customized dashboards to monitor security events in real-time.
- Works seamlessly with various security tools and technologies, allowing for a comprehensive security strategy.

## 6.6 OSSEC

OSSEC is host-based intrusion detection system (HIDS) designed to monitor individual computers and servers for suspicious activities. It focuses on identifying potential security threats at the host level, providing comprehensive monitoring of system-level events.

**Link:** <https://www.ossec.net/>

### Key Features:

- Tracks changes to critical system files, ensuring they have not been tampered with or modified without authorization.
- Analyses system logs for unusual patterns or activities that could indicate a security threat.
- Detects rootkits that could compromise system security.
- Keeps an eye on running processes to identify unauthorized or suspicious activity.

## 7. Comparison of Intrusion Detection System Solutions for Redback Operations

Each IDS solution offers distinct features and cost considerations, tailored to meet Redback's specific requirements and budget constraints.

### 7.1 Side-by-Side Comparison of existing IDS Solutions

DS Solution	Features	Ease of Deployment	Maintenance Requirements
<b>Suricata</b>	<ul style="list-style-type: none"> <li>High-performance network IDS/IPS, signature-based detection,</li> <li>Protocol analysis</li> </ul>	Relatively easy using package management tools (apt)	<ul style="list-style-type: none"> <li>Regular updates to rule sets and software versions,</li> <li>Monitoring and tuning for optimal performance</li> </ul>
<b>Snort</b>	<ul style="list-style-type: none"> <li>Widely used IDS/IPS solution,</li> <li>Robust signature-based detection</li> </ul>	Easy installation from Ubuntu repositories (apt)	<ul style="list-style-type: none"> <li>Regular updates to rule sets and software versions,</li> <li>Monitoring and tuning for optimal performance</li> </ul>
<b>Suricata with ELK Stack Integration</b>	<ul style="list-style-type: none"> <li>Centralized log management,</li> <li>Analysis,</li> <li>Visualization,</li> <li>Correlation with events</li> </ul>	Requires additional configuration steps	<ul style="list-style-type: none"> <li>Regular updates to Suricata and ELK Stack components,</li> <li>Proper indexing,</li> <li>Storage management,</li> <li>Log rotation</li> </ul>
<b>OSSEC</b>	<ul style="list-style-type: none"> <li>Host-based IDS,</li> <li>File integrity monitoring, log analysis,</li> <li>Real-time alerting</li> </ul>	Installation and configuration require minimal effort	<ul style="list-style-type: none"> <li>Regular updates to rule sets and software versions,</li> <li>Monitoring and managing agents, ensuring communication with central manager</li> </ul>
<b>Zeek (formerly Bro)</b>	<ul style="list-style-type: none"> <li>Network security monitoring,</li> <li>Protocol analysis, powerful scripting capabilities</li> </ul>	Requires expertise for deployment and customization	<ul style="list-style-type: none"> <li>Regular updates to scripts, plugins, software versions,</li> <li>Monitoring and analyzing network traffic,</li> <li>Tuning for optimal performance</li> </ul>
<b>Snort with Barnyard2 and BASE</b>	<ul style="list-style-type: none"> <li>Combined network IDS/IPS with centralized alert management</li> <li>Web-based analysis</li> </ul>	Installation involves multiple components	<ul style="list-style-type: none"> <li>Regular updates to Snort rules, Barnyard2 configurations, BASE components,</li> <li>Monitoring and managing alerts, database maintenance</li> </ul>

*Table 1 Comparison of different IDS solutions*

### 7.2 Cost Requirements of each IDS Solution:

#### 7.2.1 Suricata and Snort:

- **Initial Deployment:**
  - Both Suricata and Snort are open-source solutions available for free.

- Initial deployment costs primarily involve setup time and expertise required for configuration.
- **Ongoing Maintenance:**
  - Regular updates to rule sets and software versions are necessary for effective threat detection.
  - Internal resources may handle maintenance tasks, minimizing external costs.
- **Overall Cost:**
  - Suricata and Snort offer cost-effective options for network intrusion detection.
  - Ongoing costs primarily involve internal resource allocation for maintenance tasks.

### 7.2.2 Suricata with ELK Stack Integration:

- **Initial Deployment:**
  - ELK Stack components (Elasticsearch, Logstash, Kibana) may require additional infrastructure setup.
  - Costs associated with server resources and potential licensing fees for enterprise-level ELK solutions.
- **Ongoing Maintenance:**
  - Regular updates and maintenance of Suricata and ELK Stack components.
  - Potential costs for additional storage, indexing, and log management.
- **Overall Cost:**
  - Higher initial setup costs compared to standalone Suricata deployment.
  - Ongoing costs may include infrastructure maintenance and potential licensing fees.

### 7.2.3 OSSEC:

- **Initial Deployment:**
  - OSSEC is an open-source solution available for free.
  - Initial deployment costs involve setup time and expertise for agent deployment and central manager configuration.
- **Ongoing Maintenance:**
  - Regular updates to rule sets and software versions.

- Monitoring and managing agents, ensuring communication with the central manager.
- **Overall Cost:**
  - Minimal ongoing costs once deployed, primarily involving internal resource allocation for maintenance tasks.

#### 7.2.4 Zeek (formerly Bro):

- **Initial Deployment:**
  - Zeek is open-source but may entail higher initial setup costs due to its complexity.
  - Deployment requires expertise in network security monitoring and scripting.
- **Ongoing Maintenance:**
  - Regular updates to scripts, plugins, and software versions.
  - Monitoring and analyzing network traffic, tuning for optimal performance.
- **Overall Cost:**
  - Higher initial setup costs due to expertise required.
  - Ongoing costs primarily involve internal resource allocation for maintenance tasks.

#### 7.2.5 Snort with Barnyard2 and BASE:

- **Initial Deployment:**
  - Snort and Barnyard2 are open-source solutions available for free.
  - BASE may require additional infrastructure setup and potential licensing fees for certain components.
- **Ongoing Maintenance:**
  - Regular updates to Snort rules, Barnyard2 configurations, and BASE components.
  - Monitoring and managing alerts, database maintenance for BASE.
- **Overall Cost:**
  - Initial deployment costs involve setup time and potential licensing fees for certain BASE components.
  - Ongoing costs may include infrastructure maintenance and licensing fees for proprietary features.

## 8. Selection of IDS Solution: Suricata

After careful consideration and evaluation of various Intrusion Detection System (IDS) solutions, Redback Operations has finalized **Suricata** as the preferred IDS solution for its network security needs. Several factors contributed to this decision:

**Link:** <https://suricata.io/>

1. **Features:** Suricata offers a comprehensive set of features, including high-performance network intrusion detection, signature-based detection, protocol analysis, and real-time alerting. Its ability to analyse network traffic and detect anomalies aligns well with Redback's requirement for robust security monitoring.
2. **Ease of Deployment:** Suricata is relatively easy to deploy on the Ubuntu server using package management tools such as apt. Its straightforward installation process minimizes the setup time and effort required.
3. **Maintenance Requirements:** Suricata's maintenance requirements are manageable, primarily involving regular updates to rule sets and software versions. With internal resources dedicated to monitoring and tuning for optimal performance, Redback can ensure the continued effectiveness of Suricata in detecting and mitigating security threats.
4. **Cost Considerations:** Suricata offers a cost-effective solution for network intrusion detection, as it is an open-source software available for free. The ongoing maintenance costs primarily involve internal resource allocation for maintenance tasks, minimizing external expenses.
5. **Scalability and Flexibility:** Suricata's scalability and flexibility make it suitable for Redback's evolving security needs. As Redback expands its operations, Suricata can adapt to changes in network architecture and security requirements, providing continuous protection against emerging threats.

In conclusion, Suricata stands out as the ideal IDS solution for Redback Operations due to its robust features, ease of deployment, manageable maintenance requirements, cost-effectiveness, and scalability. By implementing Suricata, Redback can enhance its network security posture and mitigate the risks associated with potential security breaches, thereby safeguarding its operations and reputation.



## 9. Deploying Suricata on the Virtual Machine

### 9.1 Accessing the VM via Terminal: Step-by-Step Guide

1. **Open Terminal:** On your local machine, open the terminal application. This could be Terminal on macOS, Command Prompt on Windows, or any terminal emulator on Linux.
2. **Connect to Deakin VPN (if outside Deakin network):** If you are accessing the VM from outside the Deakin network, you need to connect to the Deakin VPN. Follow the instructions provided by Deakin University for connecting to the VPN using Cisco AnyConnect.

<https://software.deakin.edu.au/2019/04/16/cisco-anyconnect/>

Once connected to the Deakin VPN, you will be prompted to add your Deakin Credentials.

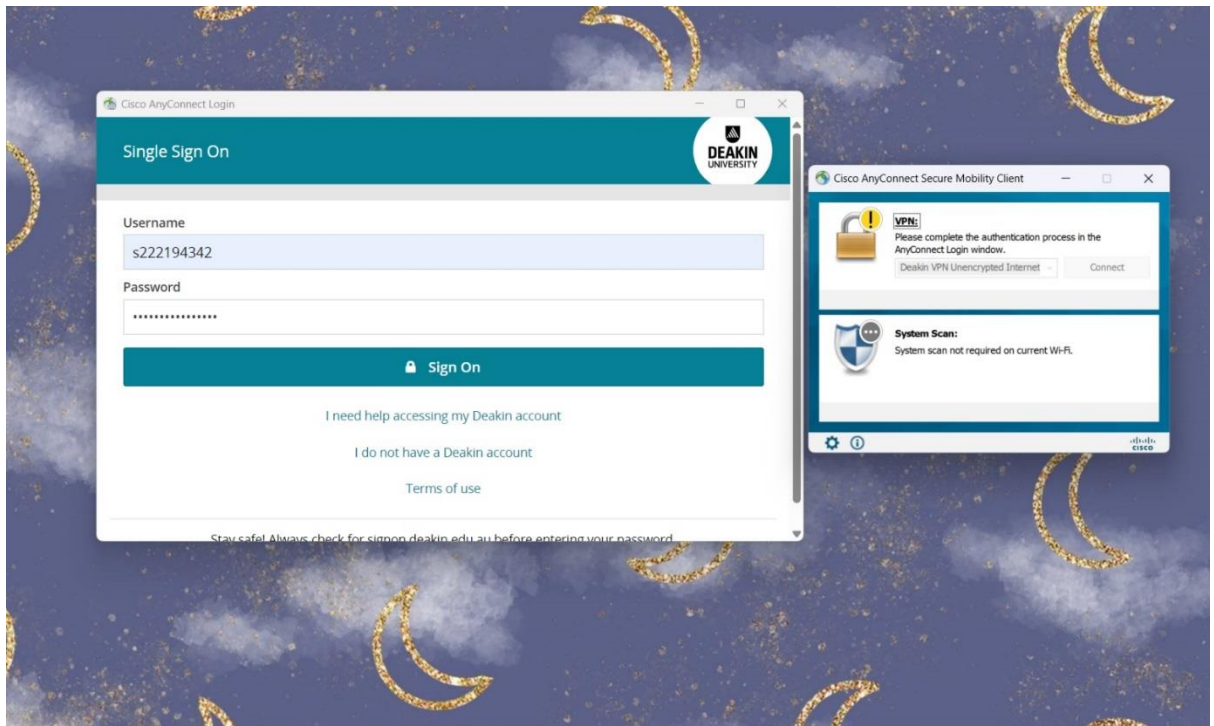


Figure 1 connecting to Deakin VPN

After entering your Deakin username and password and approving the DUO Push notification, you should encounter the following pop-up message.



*Figure 2 connected to Deakin VPN*

3. **SSH into the VM:** Once connected to the VPN (if required), you will need SSH access to connect to the VM. However, if the credentials for accessing the VM are unknown, please contact the appropriate administrator or team member who has access to the credentials. They will provide you with the necessary username and password to log in to the VM.
4. **Obtain Credentials:** Request the username and password required to access the VM from your supervisor, or the designated person responsible for managing the VM.
5. **Use SSH:** Once you have obtained the credentials, return to the terminal, and use the SSH protocol to connect to the VM. Type the following command:

```
ssh [username]@redback.it.deakin.edu.au
```

Replace [username] with the provided username. Press Enter.

6. **Enter Password:** You will be prompted to enter the password for the user. Type the password provided by your administrator and press Enter.  
Note that when typing the password, characters will not be displayed on the screen for security reasons.
7. **Successful Connection:** If the credentials are correct, you will be logged into the VM via SSH and presented with the command prompt.
8. **Perform Operations:** You can now perform various operations on the VM. The presence of the terminal prompt (\$) indicates that you have successfully established a shell session on the VM. You can execute commands and interact with the system through the terminal interface.
9. **Sudo Access:** To proceed with the configuration and installation of Suricata on the virtual machine (VM), it is essential to ensure sudo access for executing administrative tasks. This access allows for seamless implementation of the following steps.

## 9.2 Steps Followed to Configure and Integrate Suricata

To configure and install Suricata on the company's virtual machine (VM), the following steps were undertaken:

Reference: <https://docs.suricata.io/en/latest/index.html>

### 1. Installation of Prerequisites and Suricata Repository Setup:

- Installed **software-properties-common** for managing software repositories.  
`sudo apt-get install software-properties-common`
- Added the Suricata stable PPA repository to ensure installation of the stable version.  
`sudo add-apt-repository ppa:oisf/suricata-stable`
- Updated the package list to include Suricata from the newly added repository.  
`sudo apt update`
- Installed Suricata and **jq** for JSON parsing (optional but useful)  
`sudo apt install suricata jq`

These commands installed necessary tools for managing software repositories, then added the suricata PPA to ensure the installation of stable version of Suricata.

### 2. Verification of Installation:

- Checked Suricata's build information to confirm installation.  
`sudo suricata --build-info`
- Verified Suricata's service status to ensure it was running properly  
`sudo systemctl status suricata`

### 3. Configuration of Network Details:

- Determined the VM's IP address and network interface (**ens160** with IP **10.137.0.149**)  
`Ifconfig`

```
mehak@redback1:~$ ifconfig
br-a8140da369a2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
    inet6 fe80::42:f4ff:fe28:5a7f prefixlen 64 scopeid 0x20<link>
    ether 02:42:f4:28:5a:7f txqueuelen 0 (Ethernet)
    RX packets 45251 bytes 42563474 (42.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 49334 bytes 9644274 (9.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:8cff:fe40:ecaa prefixlen 64 scopeid 0x20<link>
    ether 02:42:8c:40:ec:aa txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5 bytes 526 (526.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

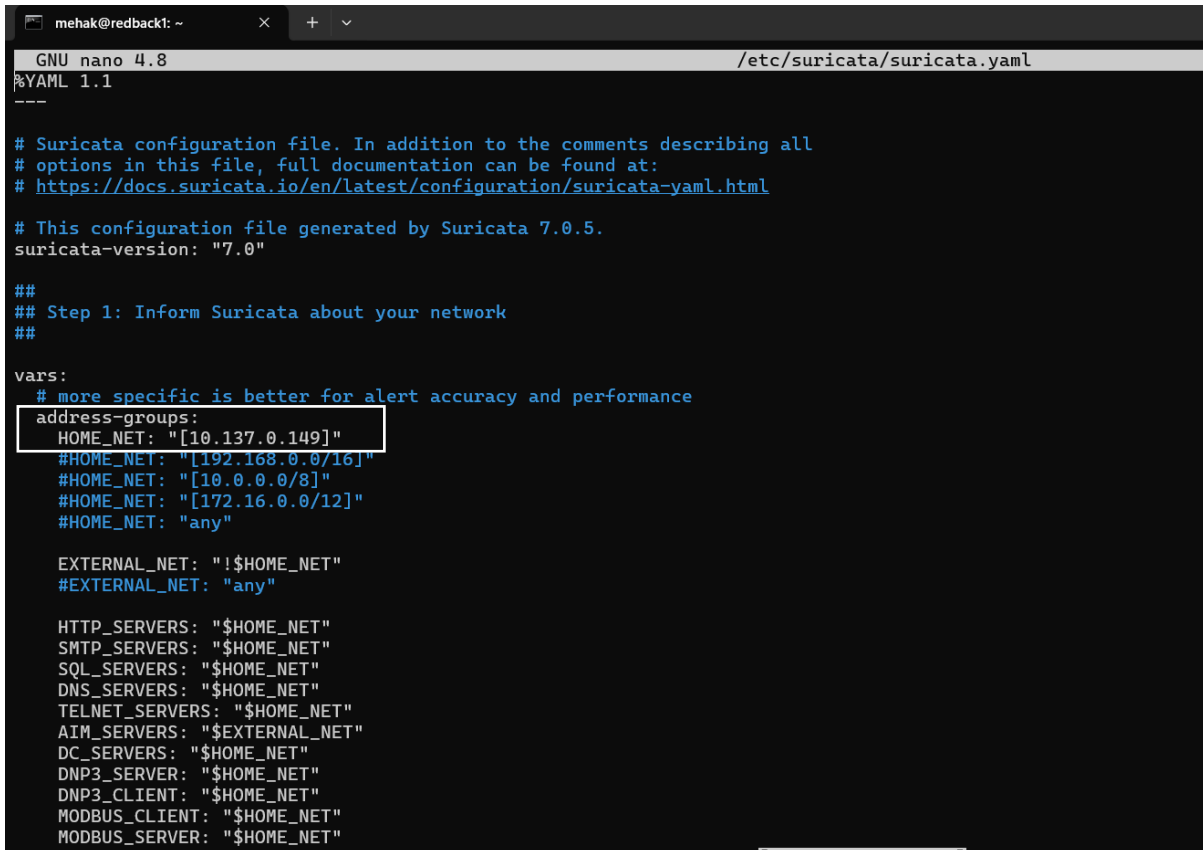
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.137.0.149 netmask 255.255.240.0 broadcast 10.137.15.255
    inet6 fe80::250:56ff:fe8a:58f5 prefixlen 64 scopeid 0x20<link>
    ether 00:50:56:8a:58:f5 txqueuelen 1000 (Ethernet)
    RX packets 11480692 bytes 8880031543 (8.8 GB)
    RX errors 0 dropped 870 overruns 0 frame 0
    TX packets 1074795 bytes 127155095 (127.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 3218 bytes 378549 (378.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3218 bytes 378549 (378.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

veth36355ff: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::4c31:6ff:fe7e:9fae prefixlen 64 scopeid 0x20<link>
    ether 4e:31:06:7e:9f:ae txqueuelen 0 (Ethernet)
    RX packets 86101 bytes 37574751 (37.5 MB)
```

Figure 3 running the ifconfig command

- Open the Suricata configuration file **suricata.yaml** to specify the network details.  
`sudo mousepad /etc/suricata/suricata.yaml`
- It will open the yaml file, now enter the IP address in \$HOME\_NET:  
This process will inform suricata about the network it has to monitor.  
HOME\_NET: "[10.137.0.149]"



```
GNU nano 4.8 /etc/suricata/suricata.yaml
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://docs.suricata.io/en/latest/configuration/suricata-yaml.html

# This configuration file generated by Suricata 7.0.5.
suricata-version: "7.0"

##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[10.137.0.149]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DC_SERVERS: "$HOME_NET"
    DNP3_SERVER: "$HOME_NET"
    DNP3_CLIENT: "$HOME_NET"
    MODBUS_CLIENT: "$HOME_NET"
    MODBUS_SERVER: "$HOME_NET"
```

Figure 4 first edit in the .yaml file

#### 4. Configuration of Packet Capturing:

- Configured **af-packet** and **libpcap** settings to capture packets from the correct interface (**ens160**).

```
mehak@redback1: ~  
GNU nano 4.8 /etc/suricata/suricata.yaml  
##  
# Linux high speed capture support  
af-packet:  
- interface: ens160  
  # Number of receive threads. "auto" uses the number of cores  
  #threads: auto  
  # Default clusterid. AF_PACKET will load balance packets based on flow.  
  cluster-id: 99  
  # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.  
  # This is only supported for Linux kernel > 3.1  
  # possible value are:  
  # * cluster_flow: all packets of a given flow are sent to the same socket  
  # * cluster_cpu: all packets treated in kernel by a CPU are sent to the same socket  
  # * cluster_qm: all packets linked by network card to a RSS queue are sent to the same  
  # socket. Requires at least Linux 3.14.  
  # * cluster_ebpf: eBPF file load balancing. See doc/userguide/capture-hardware/ebpf-xdp.rst for  
  # more info.  
  # Recommended modes are cluster_flow on most boxes and cluster_cpu or cluster_qm on system  
  # with capture card using RSS (requires cpu affinity tuning and system IRQ tuning)  
  # cluster_rollover has been deprecated; if used, it'll be replaced with cluster_flow.  
  cluster-type: cluster_flow  
  # In some fragmentation cases, the hash can not be computed. If "defrag" is set  
  # to yes, the kernel will do the needed defragmentation before sending the packets.  
  defrag: yes  
  # To use the ring feature of AF_PACKET, set 'use-mmap' to yes  
  #use-mmap: yes  
  # Lock memory map to avoid it being swapped. Be careful that over  
  # subscribing could lock your system  
  #mmap-locked: yes  
  # Use tpacket_v3 capture mode, only active if use-mmap is true  
  # Don't use it in IPS or TAP mode as it causes severe latency  
  #tpacket-v3: yes
```

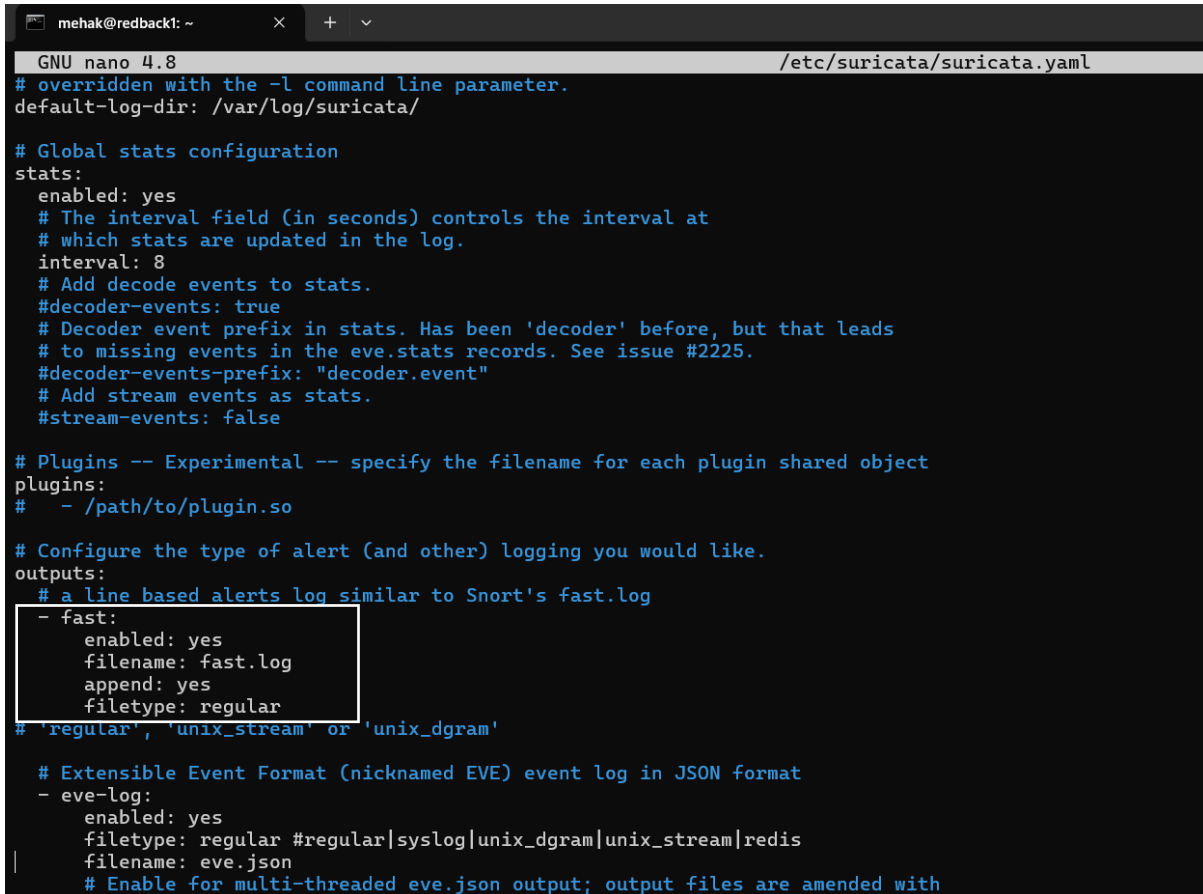
Figure 5 second edit in the .yaml file

```
mehak@redback1: ~  
GNU nano 4.8 /etc/suricata/suricata.yaml  
# Cross platform libpcap capture support  
pcap:  
- interface: ens160  
  # On Linux, pcap will try to use mmap'ed capture and will use "buffer-size"  
  # as total memory used by the ring. So set this to something bigger  
  # than 1% of your bandwidth.  
  #buffer-size: 16777216  
  #bpf-filter: "tcp and port 25"  
  # Choose checksum verification mode for the interface. At the moment  
  # of the capture, some packets may have an invalid checksum due to  
  # the checksum computation being offloaded to the network card.  
  # Possible values are:  
  # - yes: checksum validation is forced  
  # - no: checksum validation is disabled  
  # - auto: Suricata uses a statistical approach to detect when  
  # checksum off-loading is used. (default)  
  # Warning: 'capture.checksum-validation' must be set to yes to have any validation  
  #checksum-checks: auto  
  # With some accelerator cards using a modified libpcap (like Myricom), you  
  # may want to have the same number of capture threads as the number of capture  
  # rings. In this case, set up the threads variable to N to start N threads  
  # listening on the same interface.  
  #threads: 16  
  # set to no to disable promiscuous mode:  
  #promisc: no  
  # set snaplen, if not set it defaults to MTU if MTU can be known  
  # via ioctl call and to full capture if not.  
  #snaplen: 1518  
  # Put default values here  
  - interface: default  
  #checksum-checks: auto
```

Figure 6 third edit in the .yaml file

## 5. Issue Resolution (fast.log filetype specification):

- Addressed an issue where the **fast.log** was not being generated due to missing filetype specification.
  - Specified the filetype in the Suricata configuration to resolve the issue.



```
GNU nano 4.8 /etc/suricata/suricata.yaml
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/

# Global stats configuration
stats:
  enabled: yes
  # The interval field (in seconds) controls the interval at
  # which stats are updated in the log.
  interval: 8
  # Add decode events to stats.
  #decoder-events: true
  # Decoder event prefix in stats. Has been 'decoder' before, but that leads
  # to missing events in the eve.stats records. See issue #2225.
  #decoder-events-prefix: "decoder.event"
  # Add stream events as stats.
  #stream-events: false

# Plugins -- Experimental -- specify the filename for each plugin shared object
plugins:
#   - /path/to/plugin.so

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

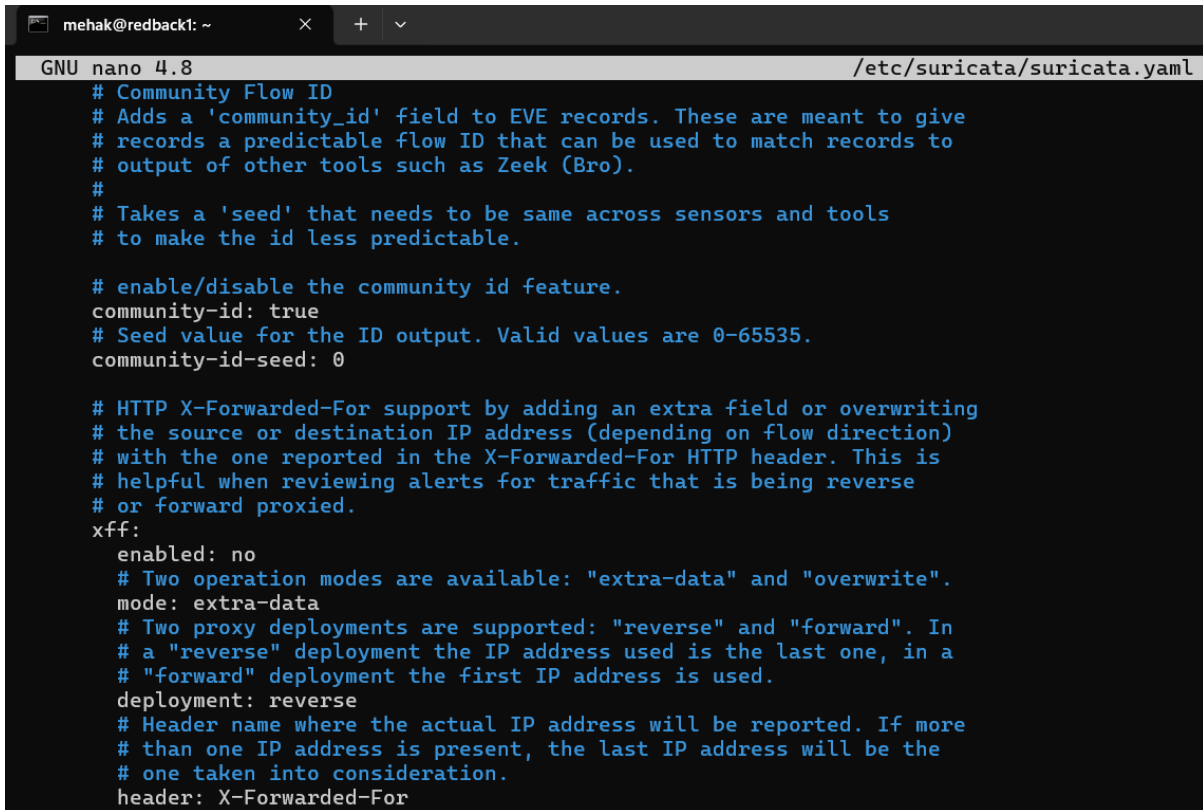
  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
      # Enable for multi-threaded eve.json output; output files are amended with
```

Figure 7 fourth and crucial edit in the .yaml file

## 6. Enablement of Community Flow ID Feature:

- Enabled the community flow ID feature in Suricata to facilitate rule set additions and integration with other tools like Zeek or Snort.





```
GNU nano 4.8 /etc/suricata/suricata.yaml
# Community Flow ID
# Adds a 'community_id' field to EVE records. These are meant to give
# records a predictable flow ID that can be used to match records to
# output of other tools such as Zeek (Bro).
#
# Takes a 'seed' that needs to be same across sensors and tools
# to make the id less predictable.

# enable/disable the community id feature.
community-id: true
# Seed value for the ID output. Valid values are 0-65535.
community-id-seed: 0

# HTTP X-Forwarded-For support by adding an extra field or overwriting
# the source or destination IP address (depending on flow direction)
# with the one reported in the X-Forwarded-For HTTP header. This is
# helpful when reviewing alerts for traffic that is being reverse
# or forward proxied.
xff:
  enabled: no
  # Two operation modes are available: "extra-data" and "overwrite".
  mode: extra-data
  # Two proxy deployments are supported: "reverse" and "forward". In
  # a "reverse" deployment the IP address used is the last one, in a
  # "forward" deployment the first IP address is used.
  deployment: reverse
  # Header name where the actual IP address will be reported. If more
  # than one IP address is present, the last IP address will be the
  # one taken into consideration.
  header: X-Forwarded-For
```

Figure 8 fifth edit in the .yaml file

## 7. Testing Suricata:

- Ran Suricata in test mode to verify functionality without issues.
- Reviewed output files:
  - **fast.log** for intrusion logs
  - **eve.json** for JSON format storage to confirm detection and threshold configurations.



```
mehak@redback1:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
Notice: suricata: This is Suricata version 7.0.5 RELEASE running in SYSTEM mode
Info: cpu: CPUs/cores online: 8
Info: suricata: Running suricata under test mode
Info: suricata: Setting engine mode to IDS mode by default
Info: exception-policy: master exception-policy set to: auto
Info: logopenfile: fast output device (regular) initialized: fast.log
Info: logopenfile: eve-log output device (regular) initialized: eve.json
Info: logopenfile: stats output device (regular) initialized: stats.log
Info: detect: 1 rule files processed. 37330 rules successfully loaded, 0 rules failed, 0
Info: threshold-config: Threshold config parsed: 0 rule(s) found
Info: detect: 37333 signatures processed. 1123 are IP-only rules, 4872 are inspecting packet payload, 31126 inspect application layer, 108 are decoder event only
```

Figure 9 testing Suricata

## 8. Detection of Traffic:

- Successfully detected traffic initiated by other users on the VM.

```
mehak@redback1:~$ sudo tail -f /var/log/suricata/fast.log &
[1] 1157319
mehak@redback1:~$ curl http://testmynids.org/uid/index.html
uid=0(root) gid=0(root) groups=0(root)
05/10/2024-14:35:49.144989 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:56664 -> 18.155.86.83:80
mehak@redback1:~$ 05/10/2024-14:35:49.146425 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
10.137.0.149:56664 -> 18.155.86.83:80
05/10/2024-14:35:51.308695 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:54942 -> 18.155.86.77:80
05/10/2024-14:35:51.308378 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
18.155.86.77:80 -> 10.137.0.149:54942
^C
mehak@redback1:~$ cat /var/log/suricata/fast.log
05/10/2024-14:35:49.144989 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:56664 -> 18.155.86.83:80
05/10/2024-14:35:49.146425 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
18.155.86.83:80 -> 10.137.0.149:56664
05/10/2024-14:35:51.308695 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:54942 -> 18.155.86.77:80
05/10/2024-14:35:51.308378 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
18.155.86.77:80 -> 10.137.0.149:54942
mehak@redback1:~$ 05/10/2024-14:37:17.699148 [**] [1:2210044:2] SURICATA STREAM Packet with invalid timestamp [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 10.137.1.15:9080 -> 10.120.0.79:42641
05/10/2024-14:37:17.724784 [**] [1:2210044:2] SURICATA STREAM Packet with invalid timestamp [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 10.137.1.15:9080 -> 10.120.0.79:42641
05/10/2024-14:37:17.729207 [**] [1:2210044:2] SURICATA STREAM Packet with invalid timestamp [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 10.137.1.15:9080 -> 10.120.0.79:42641
05/10/2024-14:37:17.733521 [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 10.120.0.79:42641 -> 10.137.1.15:9080
05/10/2024-14:37:17.733521 [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 10.120.0.79:42641 -> 10.137.1.15:9080
05/10/2024-14:39:55.854909 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:45512 -> 18.155.86.17:80
05/10/2024-14:39:55.856356 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
18.155.86.17:80 -> 10.137.0.149:45512
05/10/2024-14:39:59.480204 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.1
37.0.149:37284 -> 18.155.86.3:80
05/10/2024-14:39:59.481529 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
```

Figure 10 successful detection of traffic on Suricata

## 9.3 Future Recommendations: For T2 2024

After completing the initial configuration and installation of Suricata on the virtual machine (VM), the next steps typically involve optimizing its configuration, setting up rules for specific threat detection, and integrating it effectively into your network environment. Here are recommended next steps:

### 1. Optimize Suricata Configuration:

- Fine-tune Suricata's configuration (`suricata.yaml`) based on your network environment and security requirements. Pay attention to:
  - Logging settings (`fast.log`, `eve.json` formats).
  - Performance tuning for packet processing and resource utilization.
  - Thresholds and rules management.

### 2. Update Rules and Signatures:

- Ensure Suricata's rule set (`suricata.rules`) is up-to-date to detect the latest threats and vulnerabilities. You can:
  - Enable Emerging Threats or other community rule sets.
  - Customize rules to match company or project specific security policies and network environment.
  - Regularly update and maintain rule sets to stay protected against evolving threats.

### 3. Implement Logging and Monitoring:

- Configure logging mechanisms to capture and store Suricata's alerts and logs effectively. Consider:
  - Setting up alerts and notifications for critical events detected by Suricata.

### 4. Deploy Inline or Passive Mode:

- Decide whether to deploy Suricata in inline mode (IPS) for real-time blocking or passive mode (IDS) for monitoring only. Ensure:
  - Proper testing and validation in the environment before enabling inline mode.
  - Network segmentation and firewall rules adjustment to accommodate Suricata's inline operation.

### 5. Regular Testing and Maintenance:

- Conduct regular testing of Suricata's detection capabilities and performance.

- Schedule maintenance tasks such as updating software versions, rule sets, and configurations.

#### **6. Training and Documentation:**

- Ensure the Cybersecurity team is trained on Suricata's operation, configuration, and troubleshooting.
- Maintain documentation on your Suricata setup, including configuration details, rule modifications, and operational procedures.

#### **7. Review and Improve Security Posture:**

- Continuously review Suricata's logs and alerts to identify trends and potential security gaps.
- Implement improvements based on findings to enhance the company's overall security posture.

By following these steps, we can effectively deploy and maintain Suricata as a robust intrusion detection and prevention system in the company network, helping to safeguard against various cyber threats and security breaches.

## 10. Conclusion

In conclusion, this document has provided a comprehensive exploration of Intrusion Detection Systems (IDS) with a focused examination on Suricata as the chosen solution for Redback Operations. Beginning with an overview of IDS types and their necessity in modern cybersecurity landscapes, we delved into the components essential for effective IDS implementation, emphasizing data collection, analysis, detection algorithms, alert management, and response strategies.

The comparison of prominent IDS solutions, including Snort, Zeek, and OSSEC among others, highlighted Suricata's versatility and robust feature set in detecting and mitigating cyber threats. Through a detailed cost comparison and evaluation of integration capabilities with tools like the ELK stack, it became evident that Suricata offers a cost-effective and scalable solution suitable for Redback Operations' security needs.

The decision to select Suricata was reinforced by its advanced features, open-source nature, and community support, aligning closely with the organization's goals for proactive threat detection and response. Deploying Suricata on a virtual machine was meticulously outlined, ensuring step-by-step guidance for seamless integration and configuration.

Looking ahead, recommendations for T2 2024 include optimizing Suricata's performance, integrating with threat intelligence platforms, establishing robust incident response protocols, and maintaining vigilance through continuous monitoring and updates. By adhering to these practices, Redback Operations can strengthen its cybersecurity posture and effectively mitigate emerging threats.

In conclusion, the adoption of Suricata represents a strategic investment in enhancing network security capabilities, safeguarding organizational assets, and maintaining resilience against evolving cybersecurity challenges.