TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Hendrik Jaanimägi 192846IADB

Modular Threat Detection and Response Framework for Unmanned Vehicle Systems

“Web Applications with C#” coursework

Supervisor: Andres Käver

Tallinn 2024

Author’s declaration of originality

Table of Contents

Table of Figures

1. Introduction

1.1 Background

In the evolving landscape of cybersecurity, the protection of unmanned vehicle systems has emerged as a critical area of focus. These systems, characterized by their modular design and reliance on a myriad of interconnected components, present unique security challenges. The advent of sophisticated cyber threats necessitates the development of advanced Threat Detection and Response Systems (TDRS) tailored to address the nuanced vulnerabilities of these platforms.

1.2 Problem Statement

Modular unmanned vehicle systems, integral to various mission-critical operations, are increasingly targeted by adversaries seeking to exploit network and system vulnerabilities. Traditional security measures often fall short in providing the dynamic, context-aware defences required to protect these complex systems. There is a pressing need for a TDRS that can adapt to the unique operational and security requirements of unmanned vehicles, ensuring the integrity and availability of their mission-critical functions.

1.3 Research Focus

This thesis proposes the development of a TDRS designed specifically for modular unmanned vehicle systems. The proposed system aims to:

- **Detect anomalous network activities**, identifying unauthorized communications between modules, which may signify potential security breaches or intrusion attempts.

- **Implement log-based pattern matching**, leveraging logs from various sources within the vehicle, including modules and the Electronic Control Unit (ECU), to detect signs of malicious activity.

- **Generate and execute response recommendations**, based on predefined rules and the specific context of detected threats, with capabilities for both automatic and manual interventions. This dual approach ensures that while the system can autonomously mitigate certain threats, the operator retains ultimate control over critical response decisions.

1.4 Significance

The development of a dedicated TDRS for modular unmanned vehicle systems addresses a significant gap in the cybersecurity domain. By providing a tailored solution that encompasses both detection and response capabilities, the system enhances the security posture of unmanned vehicles, protecting them against increasingly sophisticated cyber threats. This research not only contributes to the technical body of knowledge in cybersecurity but also has practical implications for the design and operation of secure unmanned vehicle systems.

2. Analysis

2.1 Market Research

1. Overview

This chapter initiates the exploration into the development of a Threat Detection and Response System (TDRS) tailored for modular unmanned vehicle systems, emphasizing a preference for open-source solutions. The objective is to identify existing cybersecurity tools that could serve as components of the TDRS, focusing on capabilities such as network traffic inspection, log collection and parsing, and custom rule management, all while ensuring the possibility for manual operator intervention and automated response actions.

2.1.1 Methodology

The market research employed a dual-focused strategy: First, it sought to uncover commercial off-the-shelf (COTS) solutions that address general cybersecurity needs. Secondly, and more critically, it delved into open-source tools and libraries, given their potential for customization, community-driven improvements, and cost-effectiveness. The evaluation criteria were designed to highlight features crucial for unmanned systems, including:

- Network traffic inspection capabilities, especially on mirrored network ports.
- Advanced log collection and parsing for both standard and bespoke log sources.
- The flexibility of defining and managing custom detection and response rules.
- Provision for human operator intervention in the decision-making process.
- The ability for the system to automatically execute response actions.

2.1.2 Findings

Our investigation into the cybersecurity landscape revealed a rich ecosystem of both COTS and open-source solutions, with a notable emphasis on the latter due to their alignment with the project's core principles. While COTS products offer robust functionalities in network monitoring and log management, they often fall short in terms of adaptability and integration potential for unmanned vehicle systems. Conversely, open-source tools demonstrated considerable promise, owing to their flexibility and the vibrant communities behind them. However, challenges were identified in finding singular or integrated open-source solutions that fully satisfy all specified requirements without extensive customization.

Key observations include:

- **Partial Coverage by Open-Source Tools**: Open-source tools like Snort or Suricata (for IDS) and Elasticsearch or Graylog (for log management) are highly capable in their respective areas but require significant integration effort to work as a cohesive system tailored for unmanned vehicles.

- **Customization and Integration Needs**: Despite the inherent adaptability of open-source solutions, bridging the gap between these discrete tools to form a comprehensive TDRS demands considerable custom development, especially for nuanced unmanned system applications.

- **Operator Intervention**: Few open-source tools natively support workflows that allow for manual intervention by operators in a seamless manner, highlighting an area for targeted development within the TDRS.

2.1.3 Implications

The preference for open-source solutions is validated by their potential for customization and the advantages of community support. However, the research also underscores the absence of a turnkey open-source solution that meets all project criteria. This gap reinforces the project's direction towards integrating and customizing existing open-source tools, aligning with the ethos of supporting open-source software and avoiding the reinvention of proven components.

2.1.4 Conclusion

The market research establishes a solid foundation for the development of a TDRS, confirming the strategic decision to leverage open-source solutions. Despite the availability of numerous tools, the specific needs of unmanned vehicle systems—particularly in terms of system integration, customization, and operator control—necessitate a bespoke approach. This sets the stage for a project that not only contributes to the field of cybersecurity within unmanned systems but also embodies the principles of open-source development.

2.2 Detailed Problem Statement

2.2.1 Introduction

The necessity for an advanced TDRS within modular unmanned vehicle systems is driven by the unique operational environments and threat landscapes these systems navigate. While the "Market Research" has illuminated the landscape of available cybersecurity solutions, with a particular focus on open-source tools, it also highlighted significant gaps between existing capabilities and the specific needs of unmanned systems. This "Detailed Problem Statement" aims to articulate these gaps and challenges in detail, setting the stage for a targeted research and development effort.

2.2.2 The Unique Context of Unmanned Vehicle Systems

Unmanned vehicle systems, characterized by their modular design and reliance on interconnected digital and physical components, present a unique set of security challenges. These systems operate in dynamic environments where the threat of cyber-attacks can have real-world, physical consequences. The complexity of these systems, coupled with the necessity for real-time response capabilities, underscores the need for a TDRS that is both highly adaptable and capable of autonomous operation while allowing for human oversight.

2.2.3 Challenges Identified

1. **Integration of Disparate Open-Source Tools**: The cybersecurity domain offers a plethora of open-source tools, each excelling in specific aspects of threat detection and response. However, the integration of these tools into a cohesive system that addresses the full spectrum of security needs for unmanned vehicle systems remains a significant challenge. This includes ensuring compatibility, maintaining real-time data processing capabilities, and providing a unified interface for system management.

2. **Customization for Specific Threat Scenarios**: Unmanned vehicle systems are subject to a wide range of potential cyber threats, from targeted attacks aiming to disrupt operational integrity to sophisticated attempts to take control of system modules. Developing detection and response mechanisms that can be customized for the specific threat scenarios these systems face, leveraging open-source tools, requires extensive knowledge of both the operational context and the underlying technologies.

3. **Operator Intervention and Automated Responses**: Balancing automated threat responses with the need for human operator intervention presents a complex challenge. The system must be capable of making real-time decisions in critical situations while also providing mechanisms for operators to override or modify these actions when necessary. Achieving this balance within an open-source framework necessitates a deep integration of software components with operational protocols.

4. **Resource Constraints and System Performance**: Unmanned vehicle systems often operate under significant resource constraints, including limitations on computing power, memory, and network bandwidth. Developing a TDRS that is efficient enough to run within these constraints, while still providing comprehensive monitoring and response capabilities, is a critical requirement. The use of open-source solutions must therefore be optimized for performance and resource utilization.

2.2.4 Implications for Development

The detailed exploration of these challenges illustrates the gap between the current state of open-source cybersecurity solutions and the requirements of a TDRS for unmanned vehicle systems. Addressing these challenges requires not just the selection of appropriate tools, but a concerted effort to develop new integrations, customizations, and operational protocols that leverage the strengths of open-source software.

2.2.5 Conclusion

The detailed problem statement underscores the necessity for a project that goes beyond conventional cybersecurity solutions, aiming instead to harness the potential of open-source software to meet the unique demands of modular unmanned vehicle systems. This sets the groundwork for the subsequent phases of this thesis, which will focus on the design, development, and validation of a TDRS that can address these identified challenges, contributing to the safety, reliability, and operational effectiveness of unmanned vehicle systems.

2.3 Comparison of Solutions

2.3.1 Introduction

This section evaluates a selection of open-source tools across essential domains for the development of a Threat Detection and Response System (TDRS) tailored to modular unmanned vehicle systems. Emphasizing the tools' compatibility with Docker containers and their ability to handle JSON input/output, we explore network inspection tools, log collection and parsing tools, lightweight log-based pattern matching and event correlation tools, and automated response software with operator intervention capabilities.

2.3.2 Network Inspection Tools

Tool	Strengths	Weaknesses	Applicability
Suricata	High performance, advanced threat detection, JSON output natively supported	Configuration complexity, resource-intensive	High; Docker-compatible, offers detailed JSON-formatted output for deep analysis and integration
Zeek	Rich network analysis, flexible scripting, supports JSON logs	Learning curve, setup complexity	High; Runs efficiently in Docker, outputs comprehensive JSON logs, facilitating easy integration with other components
Snort3	Next-gen capabilities, improved performance, JSON output support	Transition from Snort2 may require adjustments	High; Docker support is robust, JSON output for alerts and logs aligns with modern data processing pipelines

2.3.3 Log Collection and Parsing Tools

Tool	Strengths	Weaknesses	Applicability
Fluentd	Lightweight, extensive plugin ecosystem, native JSON support	Configuration learning curve	High; Highly Docker-friendly, excels in JSON log collection and parsing, facilitating versatile log management strategies
Filebeat	Part of the Elastic Stack, lightweight, JSON input/output	Primarily tailored for Elasticsearch integration	High; Docker-compatible, efficiently handles JSON logs, ideal for forwarding structured data
Logstash	Powerful processing capabilities, flexible plugin architecture, JSON parsing and output	Resource-intensive compared to others	High; Well-supported within Docker environments, offers robust JSON processing features for complex log data

Lightweight Log-based Pattern Matching and Event Correlation Tools

Tool	**Strengths**	**Weaknesses**	**Applicability**
SEC (Simple Event Correlator)	Efficient at real-time event correlation, light on resources, flexible in handling different log formats	May require complex configuration for advanced correlation scenarios	High; ideal for Docker environments, handles text and regex patterns effectively, and supports JSON for event data
Sagan	Real-time log analysis, integrates with Snort/Suricata rule sets, JSON output	Less known, smaller community	Moderate; suitable for Docker deployment, can correlate events and output in JSON, complementing network inspection tools
Moloch	Full packet capture, indexing, and database-driven search, JSON APIs	High resource usage for full packet capture	Moderate; provides comprehensive network visibility and event correlation, Docker-compatible, with JSON output for integration

2.3.4 Automated Response Software with Operator Intervention

Tool	**Strengths**	**Weaknesses**	**Applicability**
StackStorm	Event-driven automation, extensive integration options, JSON-based workflows	Complexity in setup and management	High; Docker-ready, supports JSON for defining workflows, making it adaptable for automated responses with the option for manual oversight
TheHive	Open-source incident response, integrates with multiple tools, JSON API for communication	Setup complexity, requires integration for full functionality	High; Docker-compatible, offers a JSON-based API for seamless integration with detection and analysis tools

Tool	Strengths	Weaknesses	Applicability
Falco	Behavioral monitoring, JSON output for alerts, Kubernetes and Docker integration	Focuses on container security, may require customization for broader use	High; Ideal for Docker environments, outputs JSON alerts that can be integrated into automated response systems

2.3.5 Conclusion

The selection of tools demonstrates a strong alignment with the project's requirements for Docker compatibility and JSON support, ensuring seamless integration within a containerized architecture and facilitating API-based communication among components. This tailored suite of tools provides a robust foundation for developing a comprehensive and efficient TDRS, capable of sophisticated threat detection, analysis, and response in modular unmanned vehicle systems. The next steps involve detailing the integration strategies for these tools to construct a cohesive system that leverages Docker and JSON for enhanced interoperability and scalability.

2.4 Methods and Tools Selection

2.4.1 Introduction

Selecting the appropriate methods and tools for a Threat Detection and Response System (TDRS) is critical to the success of the project. In this sub-chapter, we rationalize the choices for the TDRS, focusing on tools that excel in Docker environments and support JSON for message exchange.

2.4.2 Rationale for JSON Usage

Interoperability: JSON is universally supported across programming languages and platforms, making it an excellent format for API communication within distributed systems like a TDRS.

Human Readability and Machine Parsability: JSON's format is both human-readable and easily parsed by machines, which is invaluable for development, debugging, and maintaining clear communication protocols between system components.

Lightweight: JSON is less verbose than XML, reducing the amount of data transmitted over the network, which can improve the efficiency of the system, particularly when dealing with the high-volume data characteristic of threat detection systems.

Comparison with XML and Other Formats: XML, while as interoperable as JSON, tends to be more verbose and requires more processing power to parse. Other binary formats like Protocol Buffers offer performance benefits but lack the text-based, human-readable advantage of JSON.

2.4.3 Rationale for Docker Containerization

Isolation: Docker ensures that each tool or service runs in an isolated environment, preventing version conflicts and ensuring consistency across different development, testing, and production environments.

Scalability: Docker containers can be easily scaled up or down, making them ideal for systems that need to adapt to varying loads, such as a TDRS that might experience fluctuating levels of network traffic and log data.
Portability: Docker containers package the application and its dependencies together, allowing for easy portability across different systems and cloud environments.

Development Efficiency: Docker simplifies the setup of development environments, allowing developers to focus on building the TDRS without worrying about inconsistencies between different workstations or deployment targets.

Comparison with Virtual Machines (VMs) and Bare Metal: VMs provide full isolation at the cost of higher overhead, while bare-metal deployments offer performance benefits but lack the isolation, portability, and quick setup and tear-down capabilities of Docker containers.

2.4.4 Selected Tools and Technologies

Based on the above considerations, the following tools have been selected for the TDRS:

- **Network Inspection**: Suricata has been chosen for its Docker compatibility and native JSON output, providing detailed threat analysis that integrates seamlessly with other system components.

- **Log Collection and Parsing**: Fluentd stands out for its lightweight nature, Docker-readiness, and JSON support, offering the flexibility required for the TDRS's log management needs.

- **Event Correlation**: SEC (Simple Event Correlator) is selected for its efficiency in real-time event correlation, ability to handle text and regex patterns, and its support for JSON outputs, aligning with the TDRS's data format preferences.

- **Automated Response**: StackStorm is chosen for its capability to automate complex workflows, Docker compatibility, and JSON-based configuration, which facilitates the creation of responsive and adaptable automated actions with the option for manual intervention.

2.4.5 Conclusion

The selection of methods and tools for the TDRS is driven by the need for efficient data processing, system interoperability, and operational flexibility. The preference for JSON and Docker stems from their widespread support, performance benefits, and alignment with the project's goals of creating a scalable, robust, and maintainable system. These technologies provide a foundation for a TDRS that is well-suited to the dynamic and demanding environment of modular unmanned vehicle systems.

3. Synthesis

…

4. Implementation

…

4.1 Developing the Solution

…

4.2 System Architecture

…

4.2.1 Entity Relationship Diagram

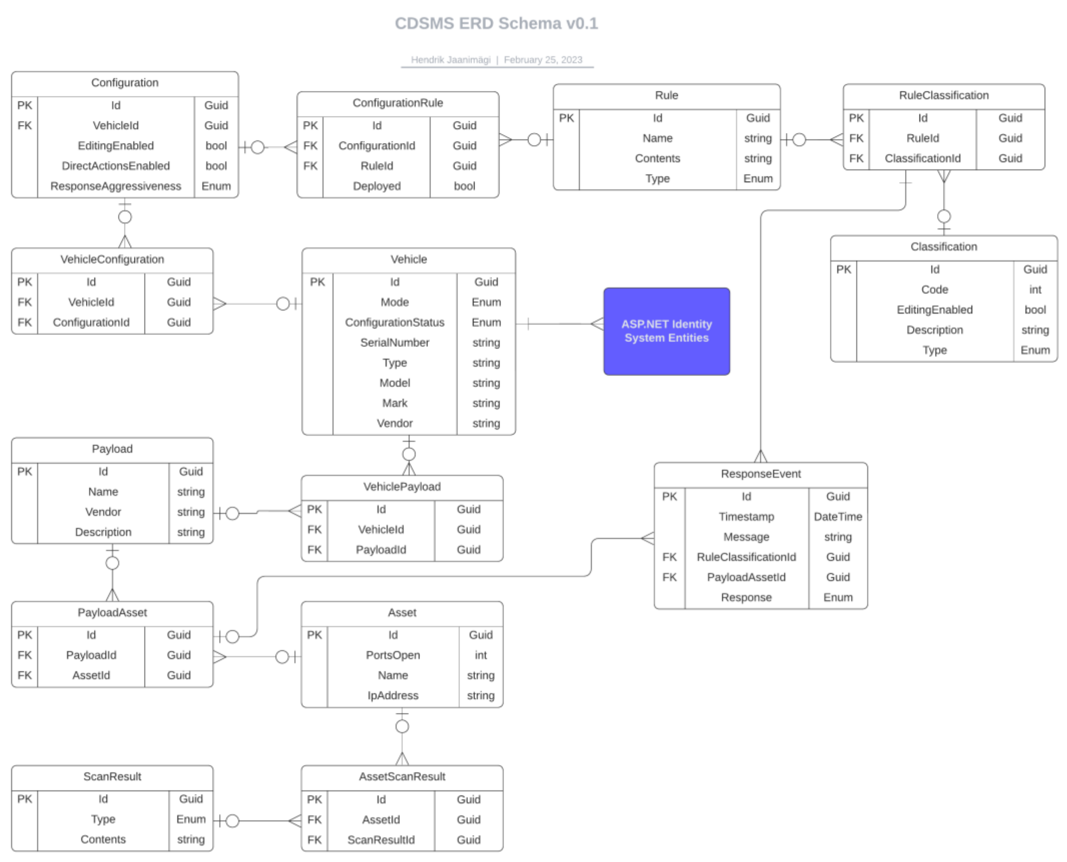


Figure 1. Entity relationship diagram representation

4.2.2 Graphical User Interface

The following figures represent the main client-side "happy flow" of the TDRS within its web interface.

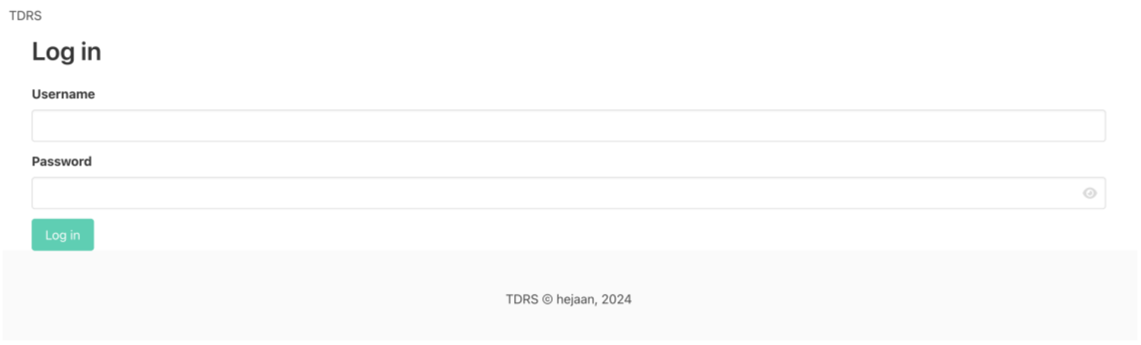
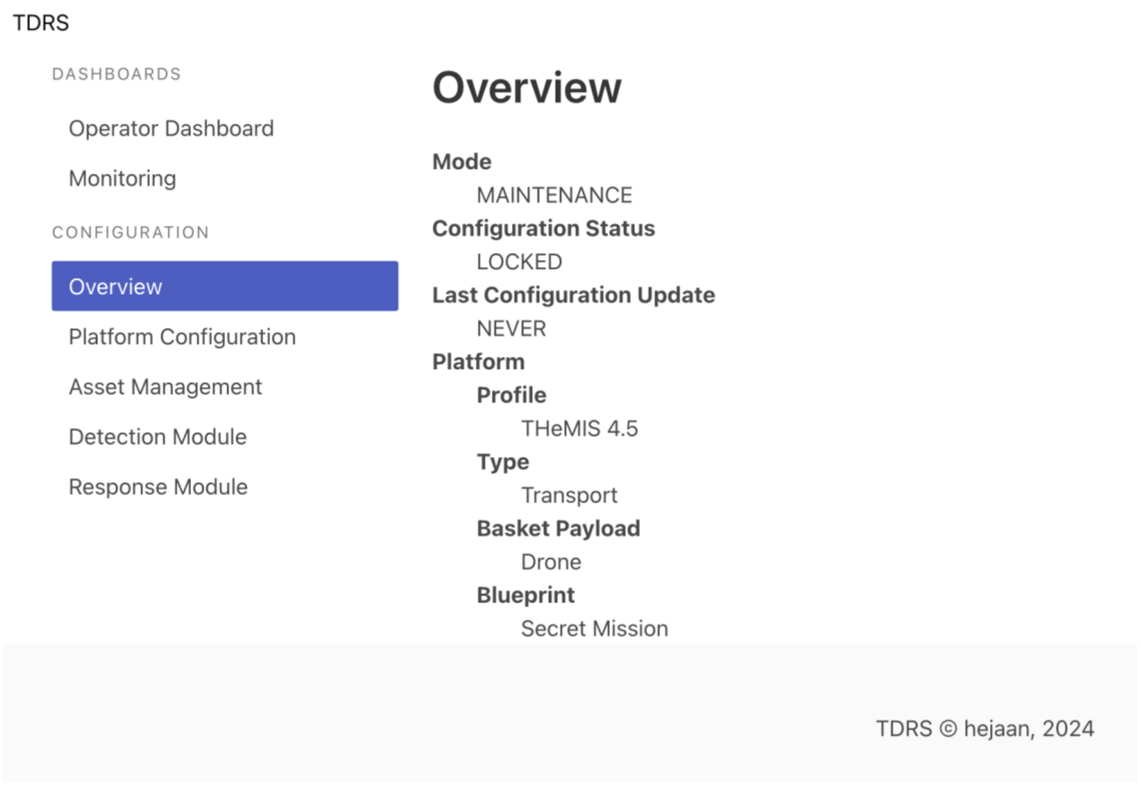


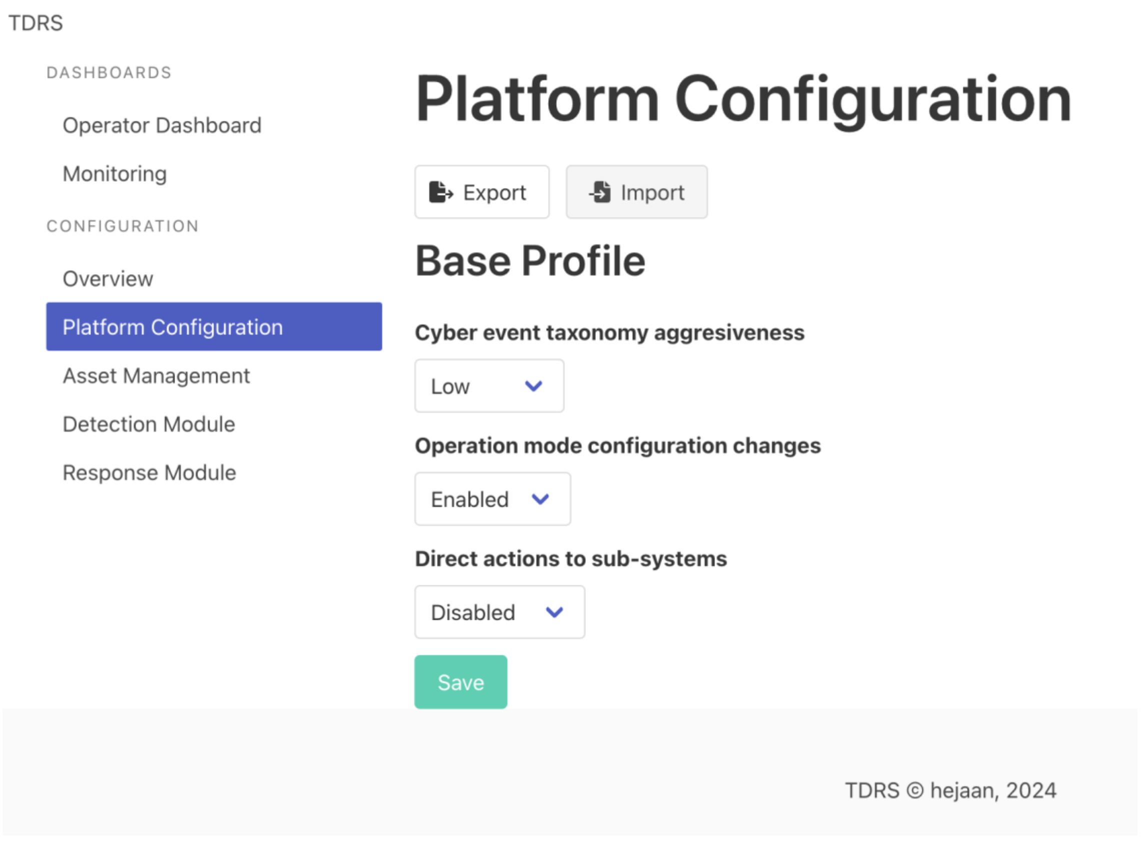
Figure 2. Login view



Overview

Mode
MAINTENANCE
Configuration Status
LOCKED
Last Configuration Update
NEVER
Platform
Profile
THeMIS 4.5
Type
Transport
Basket Payload
Drone
Blueprint
Secret Mission

Figure 3. Vehicle overview


TDRS

DASHBOARDS

Operator Dashboard

Monitoring

CONFIGURATION

Overview

Platform Configuration

Asset Management

Detection Module

Response Module

Platform Configuration

Export | Import

Base Profile

Cyber event taxonomy aggresiveness: Low

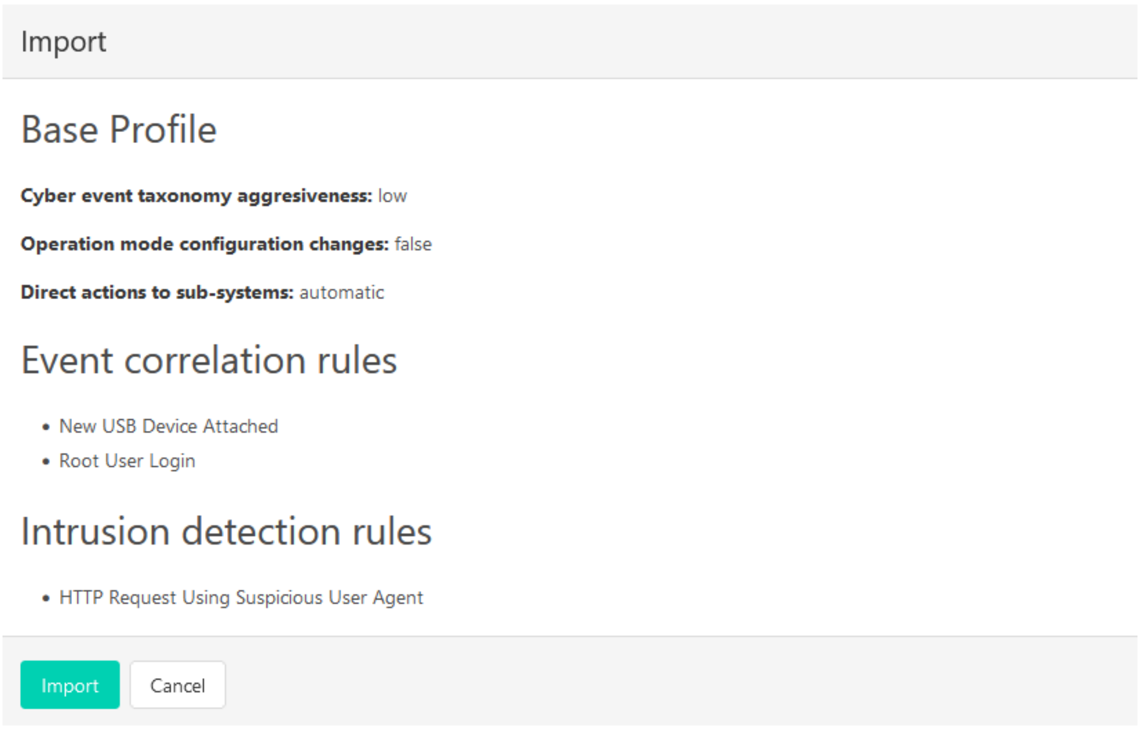
Operation mode configuration changes: Enabled

Direct actions to sub-systems: Disabled

Save

TDRS © hejaan, 2024

Figure 4. Platform configuration view


Import

Base Profile

Cyber event taxonomy aggresiveness: low

Operation mode configuration changes: false

Direct actions to sub-systems: automatic

Event correlation rules

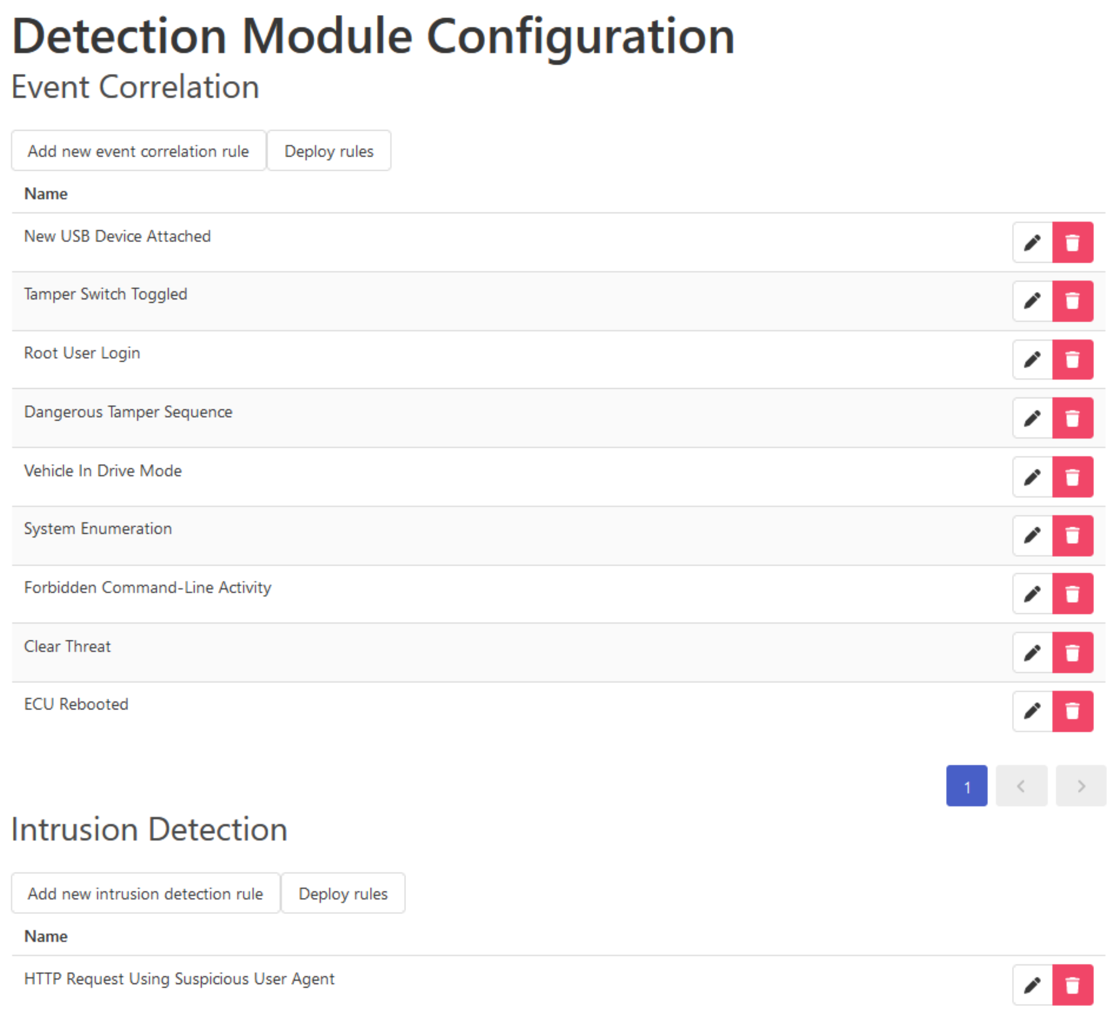
- New USB Device Attached
- Root User Login

Intrusion detection rules

- HTTP Request Using Suspicious User Agent

Import | Cancel

Figure 5. Configuration Import view



Detection Module Configuration

Event Correlation

Add new event correlation rule
Deploy rules

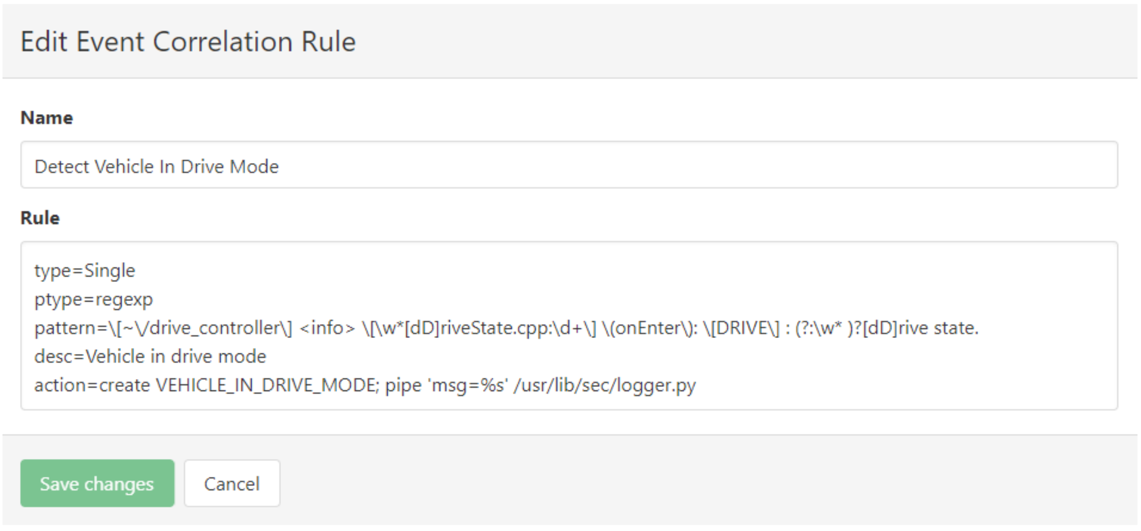
Name
New USB Device Attached
Tamper Switch Toggled
Root User Login
Dangerous Tamper Sequence
Vehicle In Drive Mode
System Enumeration
Forbidden Command-Line Activity
Clear Threat
ECU Rebooted

Intrusion Detection

Add new intrusion detection rule
Deploy rules

Name
HTTP Request Using Suspicious User Agent

Figure 6. Detection Module Configuration view



Edit Event Correlation Rule

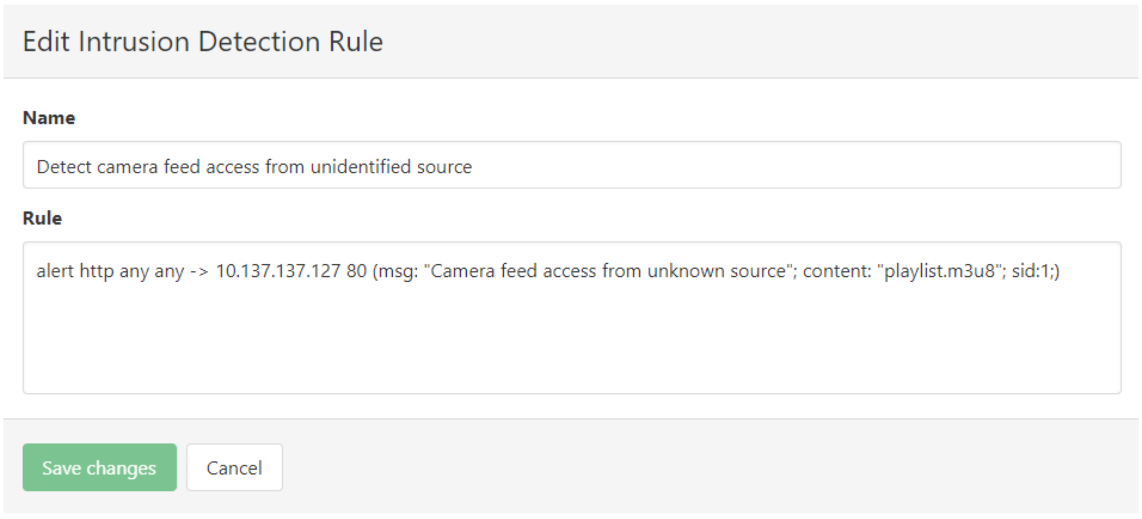
Name

Detect Vehicle In Drive Mode

Rule

```
type=Single
ptype=regexp
pattern=\[~\/drive_controller\] <info> \[\w*[dD]riveState.cpp:\d+\] \(onEnter\): \[DRIVE\] : (?:\w* )?[dD]rive state.
desc=Vehicle in drive mode
action=create VEHICLE_IN_DRIVE_MODE; pipe 'msg=%s' /usr/lib/sec/logger.py
```

Figure 7. Event Correlation Rule Creation view



Edit Intrusion Detection Rule

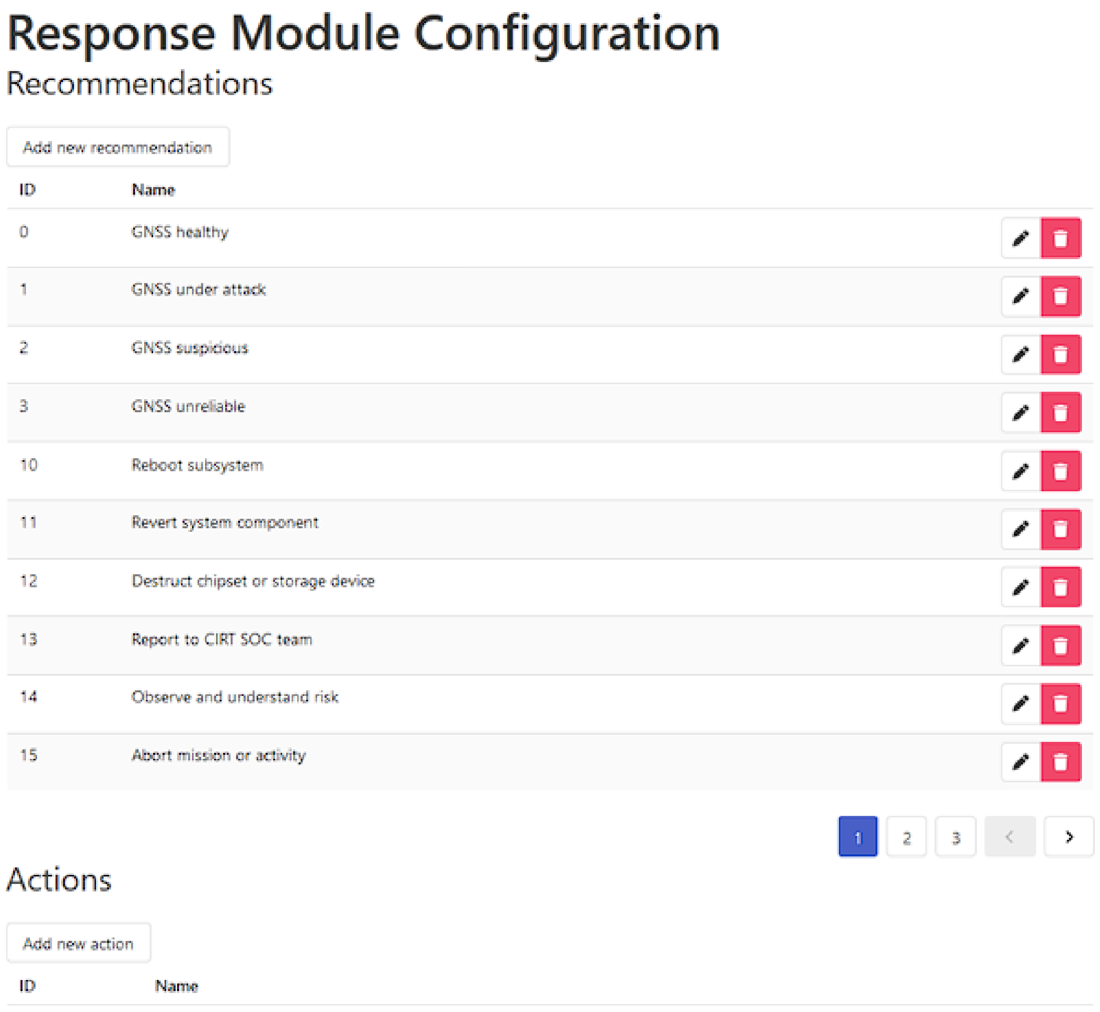
Name

Detect camera feed access from unidentified source

Rule

```
alert http any any -> 10.137.137.127 80 (msg: "Camera feed access from unknown source"; content: "playlist.m3u8"; sid:1;)
```

Figure 8. Intrusion Detection Signature Creation view



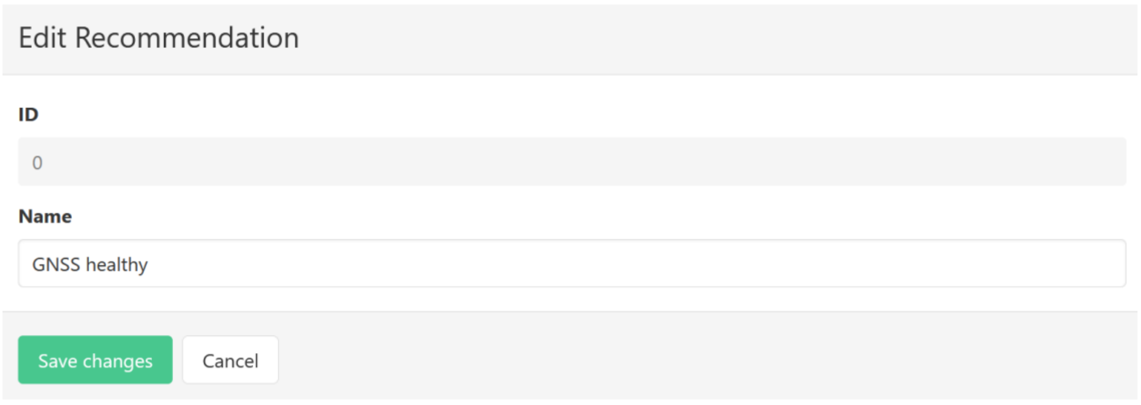
Response Module Configuration

Recommendations

ID	Name
0	GNSS healthy
1	GNSS under attack
2	GNSS suspicious
3	GNSS unreliable
10	Reboot subsystem
11	Revert system component
12	Destruct chipset or storage device
13	Report to CIRT SOC team
14	Observe and understand risk
15	Abort mission or activity

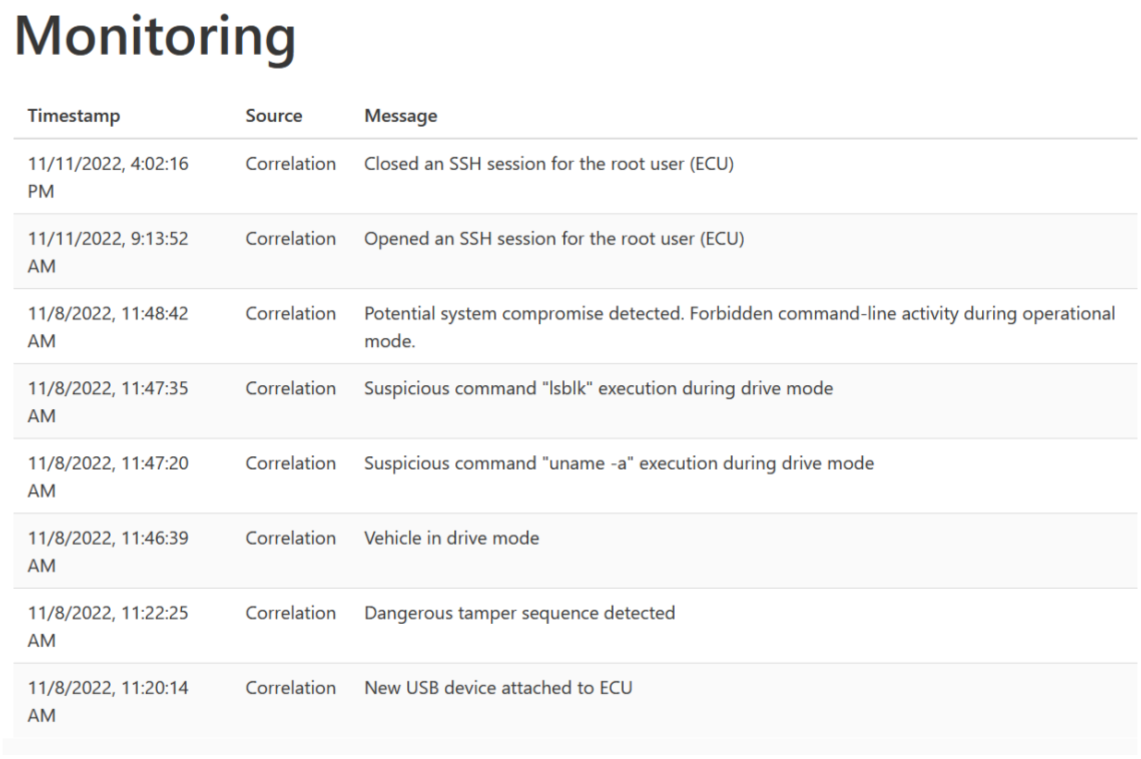
Actions

Figure 9. Response Classification Configuration view


Edit Recommendation

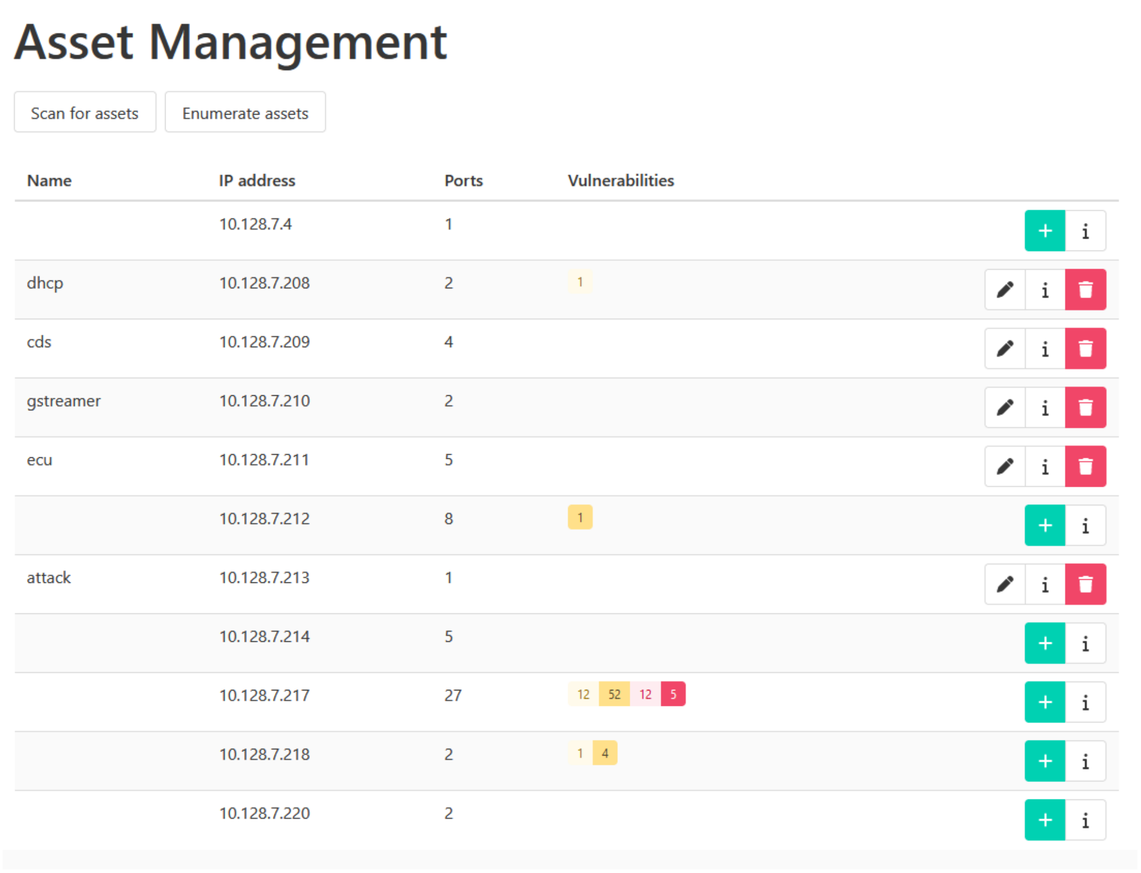
ID	0
Name	GNSS healthy

Figure 10. Response Classification Recommendation modal view


Monitoring

Timestamp	Source	Message
11/11/2022, 4:02:16 PM	Correlation	Closed an SSH session for the root user (ECU)
11/11/2022, 9:13:52 AM	Correlation	Opened an SSH session for the root user (ECU)
11/8/2022, 11:48:42 AM	Correlation	Potential system compromise detected. Forbidden command-line activity during operational mode.
11/8/2022, 11:47:35 AM	Correlation	Suspicious command "lsblk" execution during drive mode
11/8/2022, 11:47:20 AM	Correlation	Suspicious command "uname -a" execution during drive mode
11/8/2022, 11:46:39 AM	Correlation	Vehicle in drive mode
11/8/2022, 11:22:25 AM	Correlation	Dangerous tamper sequence detected
11/8/2022, 11:20:14 AM	Correlation	New USB device attached to ECU

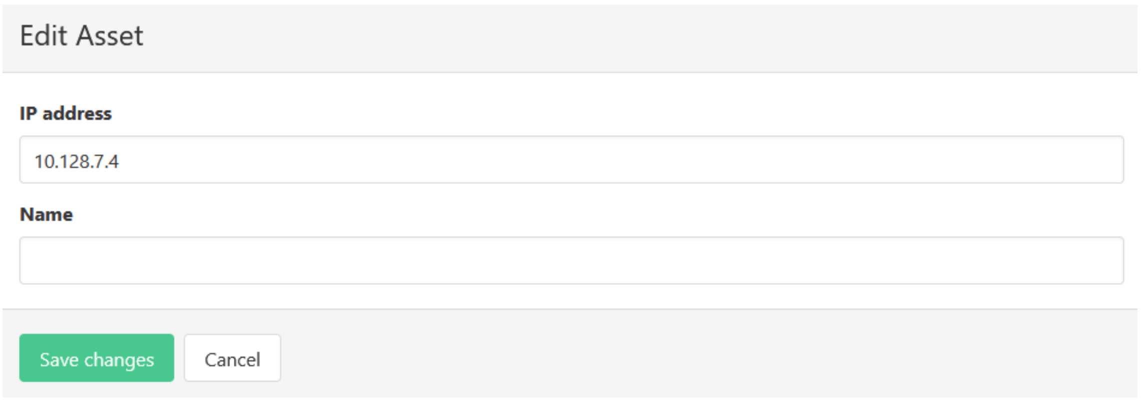
Figure 11. Event Monitoring view



Asset Management

Name	IP address	Ports	Vulnerabilities
	10.128.7.4	1	
dhcp	10.128.7.208	2	1
cds	10.128.7.209	4	
gstreamer	10.128.7.210	2	
ecu	10.128.7.211	5	
	10.128.7.212	8	1
attack	10.128.7.213	1	
	10.128.7.214	5	
	10.128.7.217	27	12 52 12 5
	10.128.7.218	2	1 4
	10.128.7.220	2	

Figure 12. Asset Management view

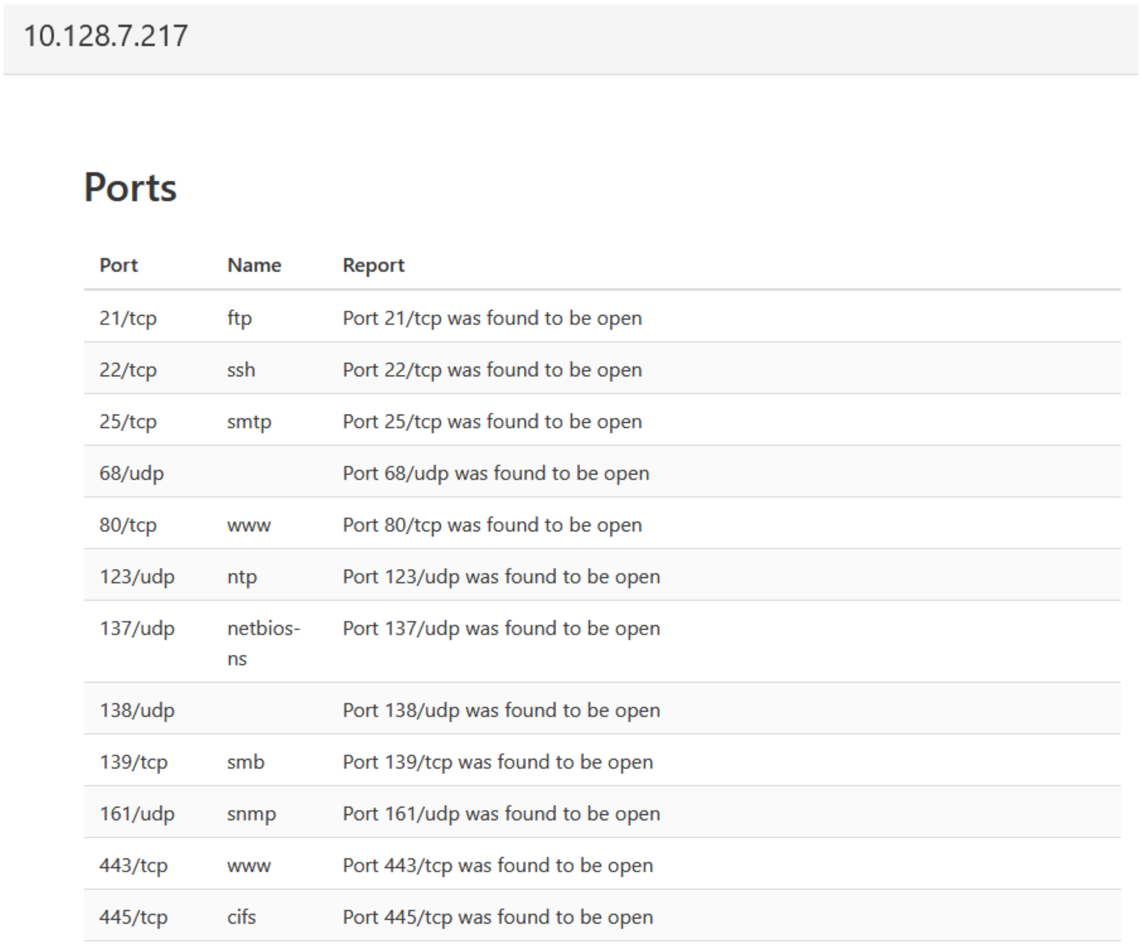


Edit Asset

IP address: 10.128.7.4

Name:

Figure 13. Asset Management modal view

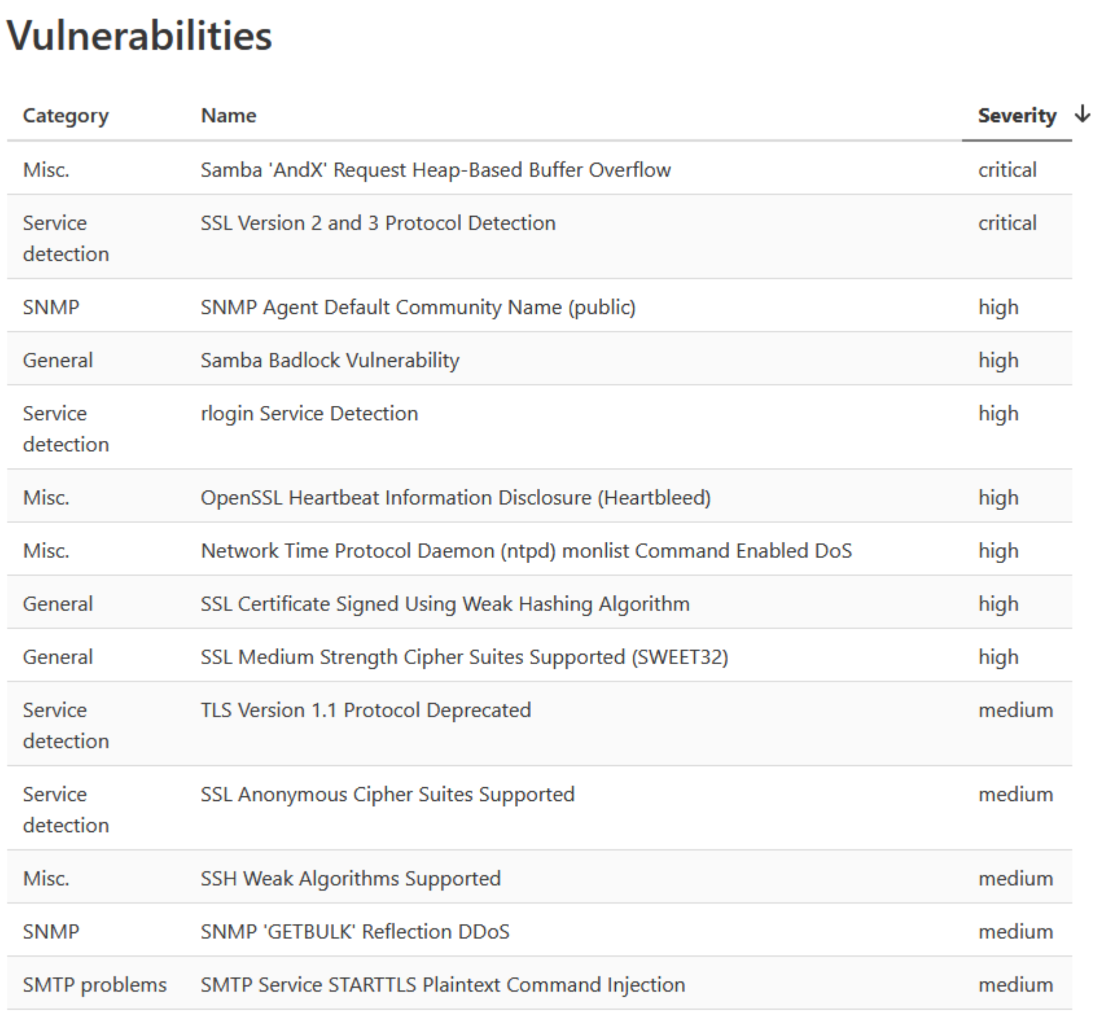


10.128.7.217

Ports

Port	Name	Report
21/tcp	ftp	Port 21/tcp was found to be open
22/tcp	ssh	Port 22/tcp was found to be open
25/tcp	smtp	Port 25/tcp was found to be open
68/udp		Port 68/udp was found to be open
80/tcp	www	Port 80/tcp was found to be open
123/udp	ntp	Port 123/udp was found to be open
137/udp	netbios-ns	Port 137/udp was found to be open
138/udp		Port 138/udp was found to be open
139/tcp	smb	Port 139/tcp was found to be open
161/udp	snmp	Port 161/udp was found to be open
443/tcp	www	Port 443/tcp was found to be open
445/tcp	cifs	Port 445/tcp was found to be open

Figure 14. Asset Management Enumeration modal view – discovered ports



Vulnerabilities

Category	Name	Severity ↓
Misc.	Samba 'AndX' Request Heap-Based Buffer Overflow	critical
Service detection	SSL Version 2 and 3 Protocol Detection	critical
SNMP	SNMP Agent Default Community Name (public)	high
General	Samba Badlock Vulnerability	high
Service detection	rlogin Service Detection	high
Misc.	OpenSSL Heartbeat Information Disclosure (Heartbleed)	high
Misc.	Network Time Protocol Daemon (ntpd) monlist Command Enabled DoS	high
General	SSL Certificate Signed Using Weak Hashing Algorithm	high
General	SSL Medium Strength Cipher Suites Supported (SWEET32)	high
Service detection	TLS Version 1.1 Protocol Deprecated	medium
Service detection	SSL Anonymous Cipher Suites Supported	medium
Misc.	SSH Weak Algorithms Supported	medium
SNMP	SNMP 'GETBULK' Reflection DDoS	medium
SMTP problems	SMTP Service STARTTLS Plaintext Command Injection	medium

Figure 15. Asset Management Enumeration modal view - vulnerabilities

4.3 Challenges and Solutions

…

5. Conclusion

…

5.1 Impact on the Cybersecurity Landscape

…

5.2 Economic Dimension

…

5.3 Further Activities

…

6. Summary

...