Scientific Research Publishing

Journal of Software Engineering and Applications, 2024, 17, 396-420
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

A Framework for Cybersecurity Alert Distribution and Response Network (ADRIAN)

Akarshita Shankar, Vijay Madisetti

School of Cybersecurity and Privacy, Georgia Institute of Technology, Atlanta, USA
Email: vkm@gatech.edu

How to cite this paper: Shankar, A. and Madisetti, V. (2024) A Framework for Cybersecurity Alert Distribution and Response Network (ADRIAN). *Journal of Software Engineering and Applications*, **17**, 396-420.
https://doi.org/10.4236/jsea.2024.175022

Received: April 13, 2024
Accepted: May 28, 2024
Published: May 31, 2024

Abstract

Security Information and Event Management (SIEM) platforms are critical for organizations to monitor and manage their security operations centers. However, organizations using SIEM platforms have several challenges such as inefficiency of alert management and integration with real-time communication tools. These challenges cause delays and cost penalties for organizations in their efforts to resolve the alerts and potential security breaches. This paper introduces a cybersecurity Alert Distribution and Response Network (Adrian) system. Adrian introduces a novel enhancement to SIEM platforms by integrating SIEM functionalities with real-time collaboration platforms. Adrian leverages the uniquity of mobile applications of collaboration platforms to provide real-time alerts, enabling a two-way communication channel that facilitates immediate response to security incidents and efficient SIEM platform management. To demonstrate Adrian's capabilities, we have introduced a case-study that integrates Wazuh, a SIEM platform, to Slack, a collaboration platform. The case study demonstrates all the functionalities of Adrian including the real-time alert distribution, alert customization, alert categorization, and enablement of management activities, thereby increasing the responsiveness and efficiency of Adrian's capabilities. The study concludes with a discussion on the potential expansion of Adrian's capabilities including the incorporation of artificial intelligence (AI) for enhanced alert prioritization and response automation.

Keywords

SIEM Platforms, Alert Distribution, Incident Response Automation, SIEM Management, Collaboration Platform

1. Introduction

Security Information and Event Management (SIEM) platforms assist organiza-

tions in data collection, policies, notifications on events, and data consolidation and correlation. These platforms help in providing real-time visibility of the organization's security systems, maintaining event logs from various sources, correlating the logs from the sources using predefined rules, etc. However, despite their critical importance, SIEM platforms face several challenges that limit their effectiveness and operational efficiency [1].

Some of the problems that hinder their efficacy include:

- There are alerts that are generated for any suspicious log activity performed in the host machine. These alerts that are managed and visualized through the dashboards configured in the SIEM platform, requiring the DevSecOps team members to manually log in and review the generated alerts. This process is time-consuming and not efficient especially because most of the open-source SIEM platforms do not have a mobile application and are hosted on servers.
- Another issue with these platforms is that the alerts that are generated are not sent to any collaborating platform such as Microsoft Teams, Slack, or CatchUp. Due to this, DevSecOps team not only miss out on real-time updates on the alerts, but it also leads to a delay in the alert response.
- There is usually an overwhelming volume of alerts generated for every activity performed in the system including deletion of a file or installing a new verified software, leading to an increase in the number of low-quality alerts generated. The cluttering of the low-quality alerts causes the team to miss out on important, high-quality alerts that would require immediate action.
- Lastly, to perform any action in the SIEM platforms, the DevSecOps team members are required to login to the server-based platform and perform actions. The actions include checking the alerts generated by their system, auditing the roles and the accesses for each user, inspecting the rules and policies, etc.

To tackle this problem, we propose ADRN (Alert Distribution and Response Network) or Adrian. Adrian can integrate any SIEM platform used by the organization with their collaboration platform, thereby addressing the identified issues by enhancing functionality and leveraging the presence of mobile applications of collaboration platforms. Examples of SIEM platforms include Wazuh, OpenCTI, and MISP, while collaboration platforms include Slack, Microsoft Teams, CatchUp, or Google Meet.

The primary goal of Adrian is to improve the SIEM platforms by enhancing its various functionalities. The advantage of Adrian is that it eliminates the need for users to always be available by their server-based system. Most of the open-source SIEM platforms do not have a mobile application, but all the collaboration platforms do have, which Adrian uses to its advantage.

The objectives of Adrian are as follows:

- Establishing a 2-way communication between SIEM platforms and collaboration tools.

- Sending alerts from SIEM to the collaboration platform.
- Analyzing the quantity of alerts generated, visually represent the data, and send the visual representation to the collaboration platform.
- Enabling SIEM platform management related activities through the collaboration platform.

Figure 1 represents the various functionalities of Adrian.

Benefits of Adrian for DevSecOps Team

The advantages of Adrian for the DevSecOps teams are as follows:

- Real-time notifications on the security incidents through their organization's collaboration platform. SIEM platforms are not available as a mobile app, but their collaboration platform would be. This means that the DevSecOps team need not be checking their SIEM platform for notifications all the time.
- Alerts generated and sent to their collaboration platform contain the severity, type, and other relevant information on the incident. The alert messages are customized to include only the details that the team would like to view. This customized notification would be hard for them to miss.
- The alerts are segregated based on the alert level. This segregation would help the team focus on the high priority incidents. This targeted communication with alert categorization would help in resolving the incident faster.
- Collaboration platforms are shared by team members. This provides a shared medium where the entire team is aware of any incident and can easily collaborate.
- The necessity to login to the SIEM platform would be eliminated as the interaction from their collaboration platform would be helpful for managing activities in their SIEM platforms.
- Alert statistics would be generated and would be visually represented and sent to a separate channel or task in their collaboration platform, which would be publicly available for anyone in the organization. Using the statistics, the team could decide if there is a requirement to login. If anyone from the higher management team would like to view the statistics, they could subscribe to this public channel. The separation of the alerts from the alert statistics would prevent the teams from higher management from receiving unnecessary and irrelevant information on the alerts.

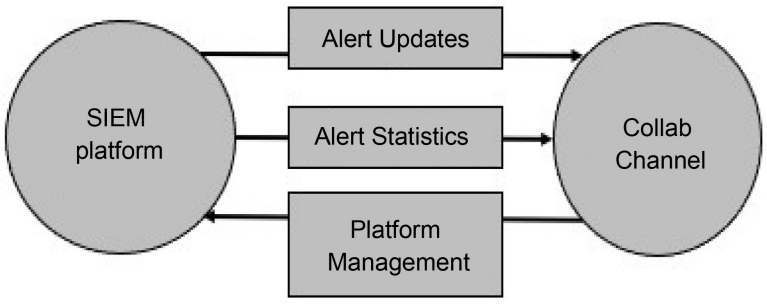

Figure 1. Overview of Adrian's functionalities.

Adrian's design principles and implementation details are demonstrated through a case study. For the case study, we have chosen Wazuh, which is a well-known open-source SIEM platform and Slack, a popular collaboration platform used by several organizations, for the DevSecOps team. This case study not only demonstrates Adrian's practical application, but also highlights its potential of helping organizations to better manage and respond to security alerts.

2. Literature Review & Related Work

It is imperative to effectively manage SIEM platforms in order to safeguard organizational information from cyber threats. Recent research highlights the evolution and challenges faced in these platforms, there by emphasizing on advancements like Adrian. This literature review summarizes the most essential points from various papers, establishing the groundwork for Adrian's contribution to the field.

1) Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures

This comprehensive review [1], discussed in this paper dives into the presence and usage of SIEM systems across various critical infrastructures and emphasizing the need for more robust cybersecurity defenses. In this paper, the authors have analyzed the evolution of SIEM technologies, starting from their origin to their contemporary tools. Using their analysis, the authors have derived a significant trend would require SIEM technologies to address the need for real-time, scalable, and adaptable security alert systems. This enhancement aligns with the development of Adrian. Adrian's innovative and scalable approach focuses on the real-time alert updates. Adrian's two-way communication channel symbolizes a more responsive, interactive, and user-friendly security operations framework.

2) Challenges and Directions in Security Information and Event Management (SIEM)

In this paper [2], the authors articulate the operational and technical challenges faced by organizations when implementing and managing SIEM solutions. Some of the challenges that the authors discuss in the paper include the volume of low alerts and false positives generated, the lack of relevant information on alerts, the delay in receiving and resolving a SIEM alert, etc. Adrian's design directly addresses these hurdles through its simplified alert management and response processes. The introduction of the middleware in Adrian's solution not only provides real-time alert updates, but also enhances reliability through its queueing mechanism, ensuring that the alerts are processed efficiently even during communication interruptions. Moreover, Adrian's approach on alert customization and categorization reduces the cognitive load on the security analysts and the DevSecOps team, allowing teams to focus on genuine threats more effectively.

3) Critical Capabilities for Security Information and Event Management

The study discussed in [3], provides an exhaustive evaluation of leading open-source SIEM systems in terms of their performance metrics, functionalities, and adaptability to the current cybersecurity requirements. The findings from the authors' analysis are the highlight on the importance of flexibility in SIEM platforms. They emphasize the need for alert customization and integration with other tools which would in turn improve the organization's response and security requirements. Adrian's architecture is designed to improve the capabilities of traditional SIEM platform through its novel integration with external platforms. It also has the capability of alert customization which addresses the shortcomings discussed in the paper.

The reviewed literature provides insights to evolution of SIEM technologies and the need for real-time alert and user-centric option. Adrian emerges as a response to these challenges by leveraging advancements in communication technology and middleware solutions.

SIEM platforms play a pivotal role in enabling organizations to detect, analyze, and respond to cybersecurity threats. Despite their critical importance, organizations face several challenges as seen in the previous section. Recent analyses, such as the Forrester report [4], on the Total Economic Impact of IBM Security QRadar SIEM identifies multiple benefits experienced by IBM's customers through the usage of SEIM tools. The benefits underscored by the report are in alignment to Adrian. The benefits are as follows:

- Enhanced Visibility and Productivity: The Ferrester report includes reports of IBM customers stating the improved visibility into their security environments and increased productivity among security teams. Similarly, Adrian aims to enhance SIEM platforms through the integrations with collaboration platforms like Microsoft Teams, Slack, etc., that would enhance the visibility and operational efficiency. The real-time alert distribution and customization features of Adrian specifically address these aspects, potentially providing similar benefits for visibility and productivity.
- Reduce False Positives and Improved Prioritization: The Ferrester report talks about IBM QRadar customers experiencing reduced false positives and were able to prioritize threats more effectively. Adrian's approach to alert customization and alert categorization directly aligns this benefit. Adrian helps in reducing the clutter of low-priority alerts by categorizing the alerts based on the alert severity level. This feature allows the security teams to focus on high priority alerts, thereby potentially reducing false positives and improving threat prioritization.
- Improved Compliance and Risk Management: As per the Ferrester's report, IBM users mentioned that the tools were found to improve the compliance and risk management. Adrian's enhanced management capabilities, including the bidirectional communication feature allow for direct integration with the SIEM platform through collaboration tools. This feature ensures timely responses to compliance-related alerts and facilitating easier audit trails.

- Scability and Flexibility: One of the benefits of IBM's SIEM tools highlights involves the scalability and flexibility to meet the evolving security needs of organizations. Adrian's design principles focus on flexibility and scalability.

Based on the information provided in the Ferrester report [4], the benefits align with the design principles of Adrian that are discussed above.

3. Adrian's Approach and Design

3.1. Introduction to Adrian's Approach

As discussed previously, some of the common challenges in SIEM platforms is the platform management, real-time communication, alert customization, etc. To address these challenges, Adrian provides solutions for SIEM platform management, real-time communication on alerts, alert customization, etc. Apart from resolving the challenges discussed, Adrian also has new features that introduces a paradigm shift in how security alerts are managed and responded to across platforms. This innovative approach adds more values to SIEM platforms and to the overall cybersecurity field.

The design of Adrian is based on the principles of reliability, customization, categorization, and bidirectional communication. Before delving into the design principles of Adrian, it is important to introduce the architecture of Adrian. The architecture of Adrian revolves around a seamless integration between existing SIEM platforms and wisely used collaboration tools. The integration is enabled through a middleware. The middleware layer ensures that the data transmission is reliable between the two platforms even during downtimes or high-traffic scenarios.

The middleware is a robust message broker that would handle the communication between the SIEM platform and the collaboration tool. Not only does this framework preserve the confidentiality and integrity of the message, but also ensure that the alerts are categorized and prioritized before being sent to the collaboration tool. These mechanisms assist in focusing on high-priority alerts, which require immediate attention.

Adrian's innovative bidirectional communication design is one of its core features that sets it apart from traditional alert management systems. Apart from just alert distribution, Adrian integrates the platforms to enable SIEM management activities through the collaboration tool.

3.2. Design Principles

- **Reliability**—Adrian prevents data loss in scenarios where the collaboration platform is down. This reliability is possible because of the introduction of middleware queueing mechanism. If the collaboration platform is down, then Adrian's queue would have all the alerts stored and then send it to the collaboration platform when it is functioning. This makes Adrian a reliable solution.

- **Customization**—Adrian's solution resolves the issue of alert customization by including only the relevant fields and modifying it to a more readable format. The cleaned data is then sent to the collaboration platform. This alert customization mechanism would help the DevSecOps teams to interpret the alerts.
- **Categorization**—Apart from alert customization, Adrian has the capability for alert categorization also. Based on the severity level of the alert, it categorizes the alerts into 3 categories—low, medium, and high alerts. This would be helpful for organization as the DevSecOps team can primarily focus on the high priority alerts.
- **Alert Statistics**—Another advantage of Adrian is the generation and communication of the alert statistics. Adrian not only calculates the alert statistics, but also provides a visual representation of the generated statistics and sends to a separate channel in the organization's collaboration platform. Adrian also ensures that this communication is performed in a separate channel so that the statistics is not confused with the alerts. These statistics enable security teams to identify trends, allocate resources efficiently and improve overall security posture.
- **Bidirectional Communication**—The most important feature of Adrian is its two-way communication channel. Adrian would not only provide communication from the SIEM platform to the collaboration platform, but also provide SIEM platform management activities through a channel in the collaboration platform. Adrian's bidirectional channel allows a more dynamic interaction between the two platforms, thereby significantly shortening response times and enhancing operational agility.

The above advantages help in making Adrian a robust, reliable, and scalable solution. These design principles not only significantly reduce the response time to incidents, but also enables a more dynamic and interactive approach to security management. Adrian was primarily built to bridge the gaps and provide a more comprehensive solution.

3.3. Technical Architecture

The proposed solution, Adrian, is divided into two phases. The first phase involves alert distribution from the SIEM platform to the collaboration platform. The second phase involves response network from the collaboration platform to the SIEM platform.

Unlike the traditional approach of integrating two systems using APIs, Adrian uses a more reliable and scalable approach. It uses a middleware queueing mechanism that would not only connect SIEM platform with the collaboration platform but also regularize the traffic flow between the two platforms. **Figure 2** provides a high-level architecture of Adrian through middleware where the message broker would be the queueing mechanism that would connect the SIEM platform to the collaboration platform.

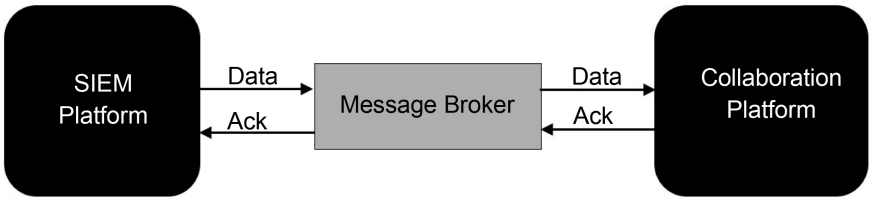

Figure 2. High-level architecture for Alert Distribution.

The message broker such as RabbitMQ or ActiveMQ helps in the asynchronous communication between the two platforms. Without waiting for the acknowledgement from the collaboration platform, the message broker would send an acknowledgement to the SIEM platform so that the SIEM platform can continue with the other allocated tasks.

The message broker would also help in prioritizing the alerts through Adrian's alert customization and alert categorization features. This queueing mechanism would ensure efficient processing and delivery to the appropriate channels in the collaboration platform, thereby making Adrian a more reliable and scalable solution.

For Phase 2 of Adrian *i.e.*, the communication from the collaboration platform to the SIEM platform can be completed through a flexible, modular interface. This communication is novel and has not been developed previously. The developed application would be the interface that connects the collaboration platform to the organization's SIEM platform. Figure 3 is the high-level architecture of the response network.

The modular interface, as described in Figure 3, could be an application that would adapt to different communication platforms and SIEM systems. The application could consist of webhooks, APIs or custom plugins based on the requirement of the organization.

The above technical architecture of Adrian shows that the solution is modular, scalable, and designed to meet the evolving needs of cybersecurity operations. The solution consists of relevant and useful features such as alert customization and alert categorization. It also consists of new features such as generating and visualizing alert statistics and enabling a bidirectional communication.

4. Implementation of Adrian: Case Study

To demonstrate the usefulness and effectiveness of the Adrian approach, we present a case-study that focuses on integrating Wazuh, an open-source SIEM platform, with Slack, a widely used communication platform. This case study serves not only as a demonstration of Adrian's capabilities, but also as an illustration of its potential to enhance the alert management systems in cybersecurity operations.

4.1. Overview of Design Choices

In Adrian's design principles, we emphasized on reliability, efficiency, and

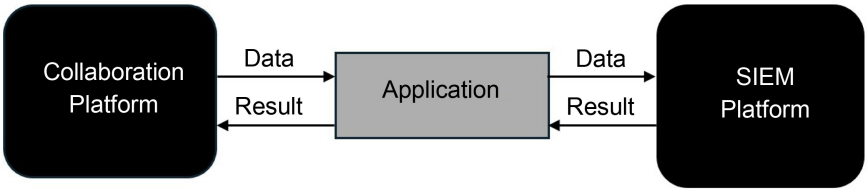

Figure 3. High-level architecture for response network.

scalability, which persuaded our choices of communication models and the specific use of Rest APIs. We decided that our model should perform asynchronous communication which would be facilitated by RabbitMQ, a message broker that is used for a publish/subscribe scenario. Using this approach, the SIEM platform continue its alert processing and generation without waiting for acknowledgements from the collaboration platform. This non-blocking feature of asynchronous communication is vital during high alert volume periods. By incorporating this feature, the model ensures that alert management is not hindered by network latency or downtimes of the collaboration platform. Moreover, the producer-consumer (a.k.a. publisher-subscriber) mechanism implemented is effective in scenarios where alerts need to be distributed across various channels. This model also supports decoupling of alerts produced from consumers, thereby enhancing the system's modularity and scalability.

For integrating the SIEM platform (Wazuh) with the collaboration platform (Slack), we chose Rest APIs due to their ubiquity and ease of use. The APIs eased a straightforward communication that can be easily managed and scaled. The stateless feature of Rest APIs supports scalable environments by simplifying server design and facilitating easier adjustments. The stateless feature is where each request contains all the necessary data to be understood independently.

These design choices were crucial for ensuring that Adrian remains robust and efficient. It should be capable of handling real-time data and adapt to the various organizational environments.

4.2. Objectives of the Case Study

The primary objectives of this case study include the following:

- To demonstrate the capabilities of Adrian by implanting its core principles. The demonstration would include middleware usage for reliability, alert customization, alert categorization, and bidirectional communication for interactive alert management.
- To reiterate the practical benefits of integrating SIEM platforms with collaboration platforms for improved alert response times, enhanced team collaboration and increased operational efficiency.
- To showcase the adaptability of Adrian across various tools and platforms, thereby emphasizing its potential for broader application in the cybersecurity field.

Adrian was implemented in two phases. The first phase was a communication

channel from Wazuh to Slack and the second phase was a communication channel from Slack to Wazuh.

4.3. Programming Language; Platform Selection

For completing both the phases of Adrian, Python was the programming language we used. The reason for choosing Python was because of the several libraries that it consists of that helped in connecting various interfaces. The following are the libraries used:

- pika—used for connecting any interface with RabbitMQ.
- slacksdk—used for connecting any interface with Slack.
- matplotlib.pyplot—used for generating the alert statistics bar graph.

These libraries help in a seamless integration between Wazuh and Slack and also provide a bidirectional communication.

For connecting Wazuh to Slack to send the alerts and alert statistics, this case study of Adrian uses RabbitMQ as its message broker.

RabbitMQ is a proven reliable broker queue that is used by organizations such as Netflix and Salesforce. It decouples the processes (in this case, Slack and Wazuh), increases the reliability and scability of the solution [5]. RabbitMQ provides reliability by providing an asynchronous communication. In addition to its basic functionalities, it also has an acknowledgement feature where it would send an acknowledgement to the appropriate platform after a message has been put in its queue or when a message is being sent from its queue. It provides various routing mechanisms through its concept of Exchange. Adrian uses Direct Exchange which helps in separating alerts and alerts statistics.

For this project, we have chosen Wazuh and Slack. The reason for choosing Wazuh was because it is a popular open-source threat prevention, detection, and response platform [6]. It is primarily used for data collection, indexing, aggregation, and data analysis from various sources including intrusion detection, suspicious behaviors, and threat detection. Some of the core functionalities of Wazuh include log data analysis, intrusion detection, incident response, etc.

We used Slack because of the channel feature that is available in the platform. The Slack channels have been made public intentionally so that anyone, with the required permissions, can subscribe to the alert channels or the alert statistics channel to view the information. However, the solution can be expanded to include other platforms such as Teams or CatchUp.

For developing the Slack application that would connect to Wazuh through an interactive channel, we used Flask and ngrok [7].

Flask is a microweb framework coded in Python that does not require any specific libraries or tools. It supports use cases for form validation, upload handling, and several common framework related tools. It is lightweight, modular, scalable, flexible making it an optimal solution for creating web applications and hosting web services.

Ngrok is a cross-platform application that creates a security tunnel for data

from the internet to the local server on our local machines. It permits exposure of web server to the internet without much effort making it particularly useful for web development, allowing developers to share their local sites without the need to deploy those sites to a public server, testing applications, etc [7].

4.4. Implementation Details

Phase 1: Wazuh to Slack Communication

The queueing mechanism chosen for Adrian is RabbitMQ [5]. RabbitMQ is an open-source message broker that is proven to be useful in a producer-consumer scenario. Using the RabbitMQ solution and decoupling Wazuh and Slack, Adrian would provide an asynchronous communication. The detailed diagram of Phase 1 of Adrian is shown in **Figure 4**.

For categorization of the alerts based on its alerts level, Adrian categorizes the alerts based on its severity level and inserting them into low-alerts-queue, medium-alerts-queue, and high-alerts-queue respectively in RabbitMQ. Alerts with severity level < 5 are put into the low-alert-queue. Alerts with severity level ≥ 5 and level < 7 are put into the medium-alerts-queue. Alerts with severity level ≥ 7 are put into high-alerts-queue [8].

The segregation is performed using Exchanges in RabbitMQ. Exchanges (aka Topics in other broker queues) assist in connecting the producer to different queues using a routing key [9]. Routing Key is specified by the producer while transmitting the message to RabbitMQ. There are different types of Exchanges; Adrian uses Direct Exchange [10]. After receiving the message and the routing key the direct exchange, based on the configured routing key, inserts the message in the appropriate queue [11]. The detailed diagram is depicted in **Figure 5**.

Alert categorization is performed on the consumer side as well. To avoid cluttering of a single Slack channel, Adrian segregates the messages into three different slack channels for the alerts [11]. The messages from the low-alerts-queue are published to the low-alerts-slack-channel. A similar concept is followed for the medium and high alerts. Apart from the three channels in Slack, Adrian uses a fourth channel to publish the alert statistics [12] [13]. This helps in separating

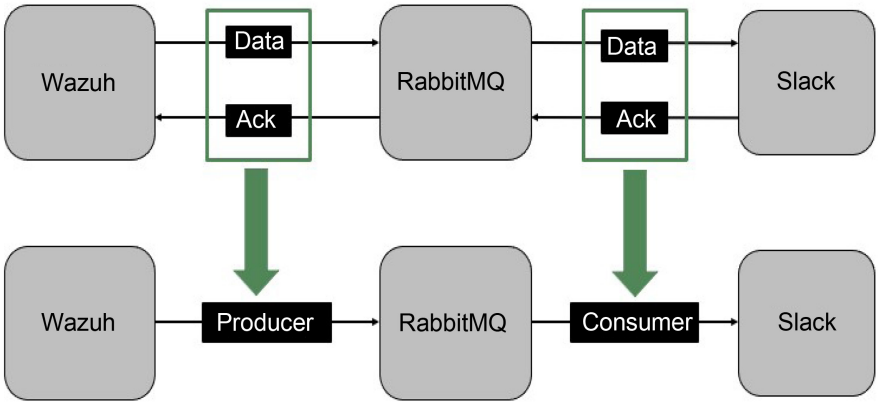

Figure 4. Phase 1 of Adrian—Alert distribution.

the technical specifics and the statistical information of the alerts [14]. The detailed diagram is depicted in Figure 6.

Figure 7 shows the queues that are configured in RabbitMQ. The queues are wazuh-alerts-high, wazuh-alerts-medium, and wazuh-alerts-low [15].

Figure 8 shows the binding of the routing keys and the queues through exchange. Adrian uses Direct Exchange. The routing key for wazuh-alerts-high queue is wazuh-alerts-high. The routing key for wazuh-alerts-low queue is wazuh-alerts-low. The routing key for wazuh-alerts-medium queue is wazuh-alerts-medium.

Phase 2: Slack to Wazuh Communication

For phase 2 of Adrian i.e., connecting Slack with Wazuh. The Slack application provides an interactive Slack channel that would provide various management related activities to be performed on Wazuh. Each button corresponds to an activity that can be obtained from Wazuh [14]. Figure 9 is the block diagram provides a detailed visual representation of the response network architecture [16].

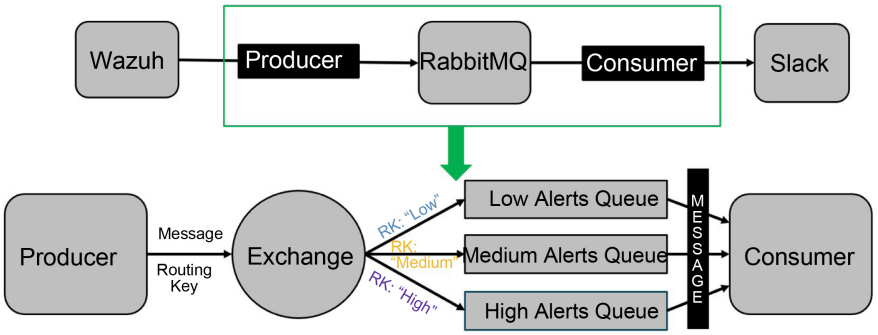

Figure 5. Detailed diagram of Alert distribution.

Figure 6. Screenshot of the direct exchange used by Adrian.

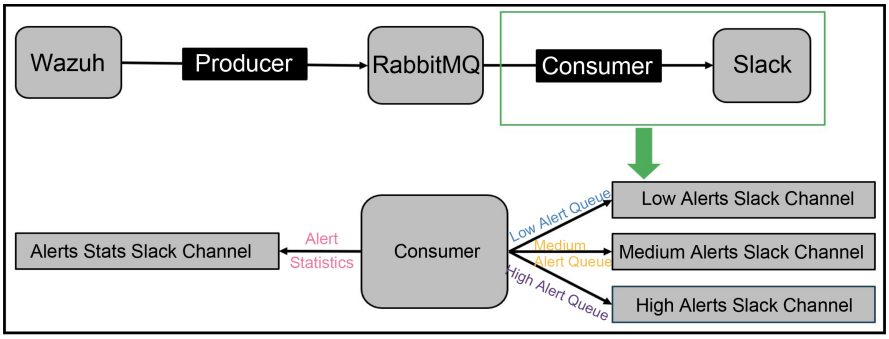

Figure 7. Detailed diagram of alert categorization.

Figure 8. Screenshot of the RabbitMQ queues.

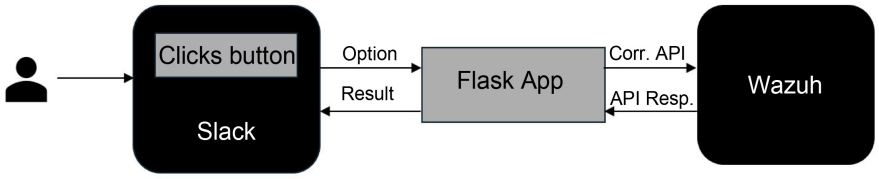

Figure 9. Phase 2 of Adrian—Response network.

An important aspect of Adrian's innovative approach is its alignment with the FCAPS model [16], a network management framework that was developed by International Organization for Standardization (ISO). By categorizing network management tasks into five domains, FCAPS provides a comprehensive blueprint for network operations. This model is adeptly used by Adrian to enhance SIEM functionality and team responsiveness. The full form and the details of the FCAPS model is:

- **Fault Management**—helps in identifying and correcting any malfunctions.
- **Configuration Management**—helps in monitoring the system configurations.
- **Accounting Management**—helps in managing the users, roles, etc.
- **Performance Management**—helps in monitoring system performance.
- **Security Management**—helps in protecting the system by checking the se-

curity rules.

For each of the components of the FCAPS model [16], we have chosen two APIs. The APIs chosen are as follows. This is a comprehensive list and can be expanded to include more API functionalities for each category.

1) Fault Management

- Fault Status—returns the faults that the agent has been allocated. It includes the source, details, and status of the fault. It also provides the first detected timestamp and the last updated timestamp of the fault.
- Agent Upgrade Status—returns the status of an upgraded agent, if any. If any agent was upgraded, it provides the status of the upgrade and details on which component of the agent was upgraded.

2) Configuration Management

- List rules—returns the top 10 rules that Wazuh uses for identifying malicious activities. There are over 6500 rules that are configured. Hence, I chose the top 10. This number is configurable and can be modified based on the requirements.
- List decoders—returns the top 10 decoder rules and configurations that Wazuh uses for parsing and decoding the data that it receives from various sources. There are approximately 500 decoder rules, I have chosen the top 10. This field is configurable and can be modified based on the requirements.

3) Accounting Management

- List users—returns the users present in the system, along with their roles and username.
- List roles—returns the different roles configured in the system along with the name of the role and the policies that have been applied to it.

4) Performance Management

- SCA scan results—returns the SCA scan results of Wazuh. SCA stands for Security Configuration Assessment. It runs the scan for CIS benchmark for Ubuntu and provides the number of checks that passed, failed and were invalid. It also provides the start time and end time of the scan.
- Manager logs—returns the Wazuh manager logs that include a description, level, and timestamp of the logs.

5) Security Management

- Security policies—list the top 10 security policies of Wazuh. It lists the name and details of the policy, the resources, and users where the policy is implemented. There are over 5000 security policies. I have chosen the top 10 policies.
- MITRE References—lists the top 10 references that are used for identifying MITRE ATT&CK by Wazuh. The results include the MITRE ID, source, description, and the reference link for each attack pattern. There are approximately 650 references. I have chosen the top 10 policies. This field is configurable and can be modified based on the requirements.

The API and the formatting of the API responses is included in the Flask app.

The Flask app creates an interactive Slack channel that initially gives the user five options to choose from [17]. The options include the high-level configurations of FCAPS. After the user selects one of the five options, the Flask app provides two more options in the submenu. The options in the submenu include the APIs that were previously mentioned. After obtaining a choice, the app connects to Wazuh and fetches its response [18]. The response is formatted to include only the relevant details and in a more organized format and then published to the Slack channel [19]. The Flask app flowchart is depicted in **Figure 10**. **Figure 11** includes all the details of the APIs used in the FCAPS model of Adrian [20].

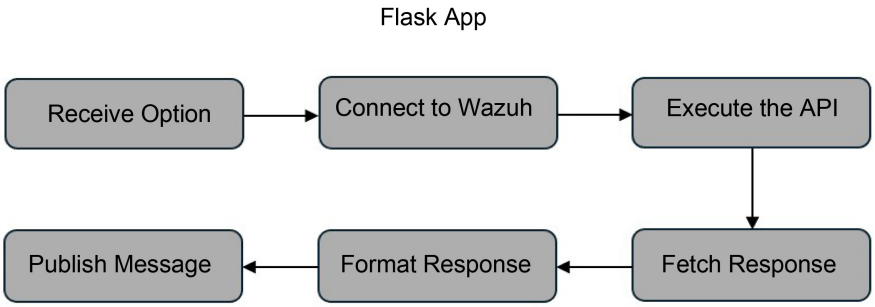

Figure 10. Overview of the activities executed by the Flask application.

FCAPS Model	Functionality	Description	Generic API Call	Query Parameter
Fault Management	Get Fault Status	Returns the faults details that the agent is allocated	rootcheck/{agent_id}	select, status, offset, limit, sort, search
	Get Agent Upgrade Status	Return the agents upgrade results	agents/upgrade_results	select, status, offset, limit, sort, search
Configuration Management	List Rules	Return a list containing information about each rule including the definition, rule group, description, status, etc.	/rules	rule_ids, group, level, filename, tsc, mitre
	List Decoders	Returns data on the all the decoders in ossec.conf file. The data includes name, route, file, etc., about the decoder.	/decoders	decoder_names, filename, status, select, sort, limit
Accounting Management	List Users	Returns data on all the users in SIEM platform.	/security/users	user_ids, select, sort, offset, limit
	List Roles	Returns roles configured along with the policies and users mapped to the role.	/security/roles	roles_ids, select, search, offset, limit
Performance Management	SCA Scan Results	Returns the security SCA database of an agent	/sca/{agent_id}	agent_id, name, description
	Get Manager Logs	Returns the last 2000 log entries	/manager/logs	level, tag, select, search, offset
Security Management	Get Security Policies	Lists all the policies in the system	/security/policies	policy_ids, limit, offset, search
	Get Mitre References	Returns the references from the MITRE database	/mitre/references	reference_ids, limit, offset, sort

Figure 11. APIs used for the FCAPS model.

5. Results and Discussion of the Case Study

This section provides screenshots of the implementation of Adrian's integration of Wazuh with Slack. The outcomes of the implementation resulted in several enhancements to the traditional alerts management and response mechanisms of SIEM platforms. The key improvements are as follows:

- Real-Time Alert Distribution: Adrian provided real-time notifications on security incidents directly through the Slack channels. This significant improvement would ensure that the DevSecOps teams would promptly be informed of incidents without the requirements of constantly monitoring the SIEM platform.
- Alert Customization and Categorization: The introduction of the RabbitMQ allowed customization and categorization of the alerts based on its severity levels. This feature significantly reduced the clutter in the notification channels and enabled the teams to focus on high Enhanced Response Capabilities: The bidirectional communication enables the DevSecOps teams to perform management-related activities directly through the Slack channel. This streamlined the response process.

The above improvements are implemented and screenshots with relevant explanations have been included in the next section.

5.1. Phase 1: Wazuh to Slack Communication

Phase 1 of Adrian covered the communication from Wazuh to Slack. While Figure 12 is the screenshot of the original alert that was before Adrian cleaned up the alert, Figure 13 shows the alert that Adrian customized to only include the relevant fields and categorized it as a low alert based on the severity level.

Figure 14 and Figure 15 present examples of medium and high alert generated in Wazuh and sent through Adrian to the corresponding Slack channels. These screenshots illustrate Adrian's capability for alert prioritization and categorization.

Figure 16 and Figure 17 is an illustration of Adrian sending the alert statistics


```
{"timestamp":"2024-02-14T17:47:52.879-0500","rule":{"level":3,"description":"PAM
: Login session opened.","id":"5501","mitre":{"id":["T1078"],"tactic":["Defense
Evasion","Persistence","Privilege Escalation","Initial Access"],"technique":["Va
lid Accounts"]},"firedtimes":2,"mail":false,"groups":["pam","syslog","authentica
tion_success"],"pci_dss":["10.2.5"],"gpg13":["7.8","7.9"],"gdpr":["IV_32.2"],"hi
paa":["164.312.b"],"nist_800_53":["AU.14","AC.7"],"tsc":["CC6.8","CC7.2","CC7.3"
]},"agent":{"id":"000","name":"akarshita-VirtualBox"},"manager":{"name":"akarshi
ta-VirtualBox"},"id":"1707950872.2244","full_log":"Feb 14 17:47:52 akarshita-Vir
tualBox su: pam_unix(su:session): session opened for user root(uid=0) by akarshi
ta(uid=0)","predecoder":{"program_name":"su","timestamp":"Feb 14 17:47:52","host
name":"akarshita-VirtualBox"},"decoder":{"parent":"pam","name":"pam"},"data":{"s
rcuser":"akarshita","dstuser":"root(uid=0)","uid":"0"},"location":"/var/log/auth
.log"}
```


Figure 12. Alert before Adrian's modification.

Figure 13. Alert after Adrian's modification.

Figure 14. Medium alert notification in slack.

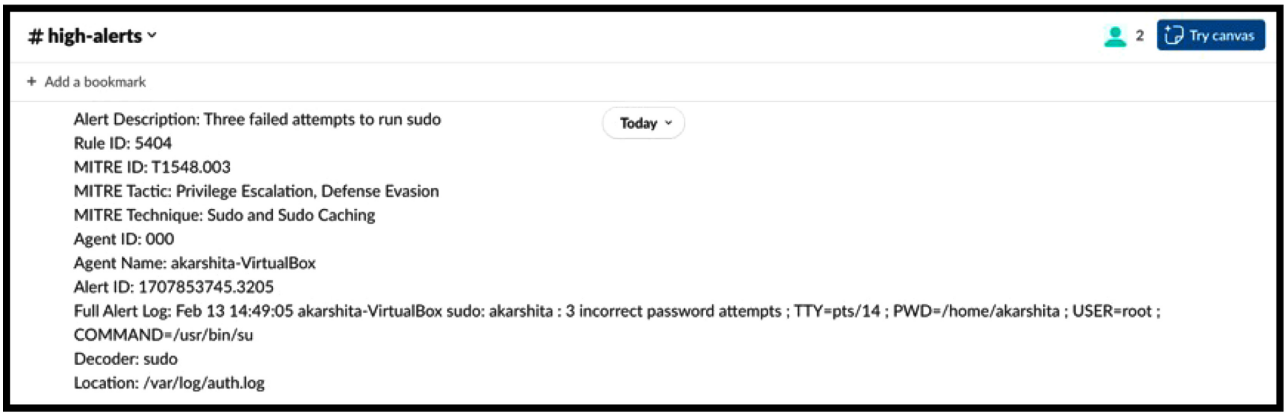

Figure 15. High alert notification in slack.

to the teams in a timespan of 24 hours. Figure 16 was sent on 3rd March and Figure 17 was sent on 4th March. The figures indicate that there was a dip in the number of low alerts from 50 to 30. There is a marginal dip in the medium alerts as well. However, the number of high alerts has increased which could be a cause of concern for the team. In such instances, the team would have to login to investigate the issue further.

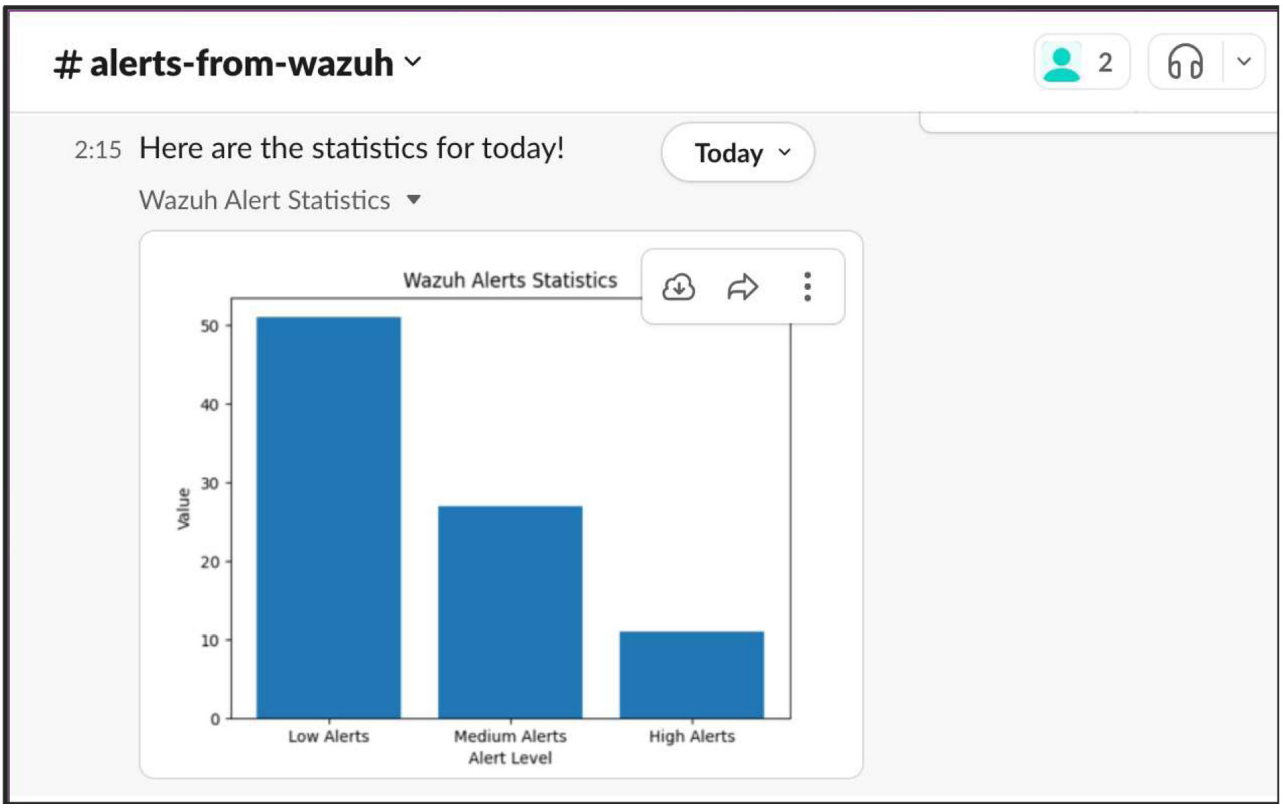

Figure 16. Exemplary visualization.

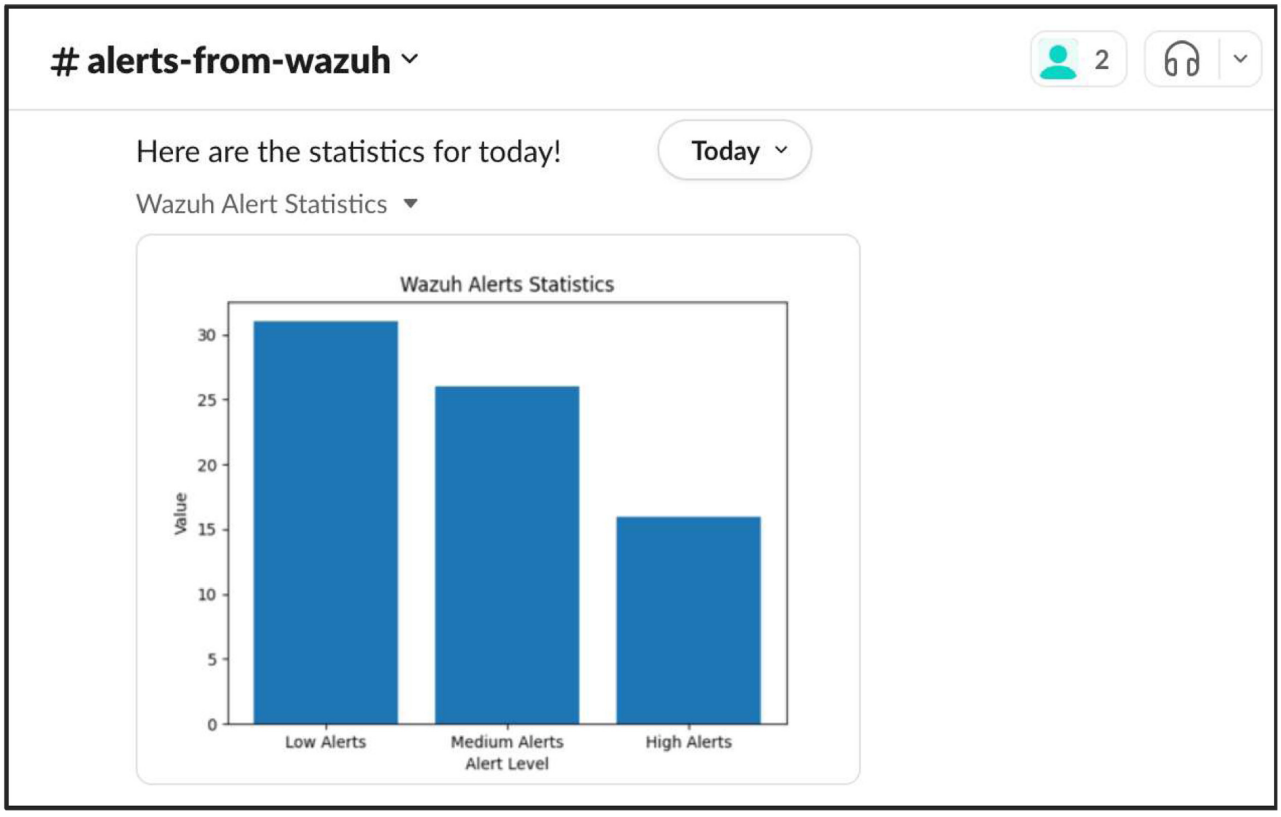

Figure 17. Exemplary visualization.

5.2. Phase 2: Slack to Wazuh Communication

Figure 18 shows the initial message of the Flask app in the slack channel. As per the FCAPS model, it displays the high-level menu for the user to select the option.

The submenu for each option is shown in Figure 19. The flowchart represents all the submenu options that the user can choose from based on the chosen model. The images in the flowchart represent the options displayed to the user from the submenu.

Figure 20 represents the Fault Status and Agent Upgrade Status. Figure 20 (top image) provides the details of the fault that the agent has been allocated. Figure 20 (bottom image) represents the agents that were upgraded. In cases where there is no agent to be upgraded, it displays the message depicted in the diagram.

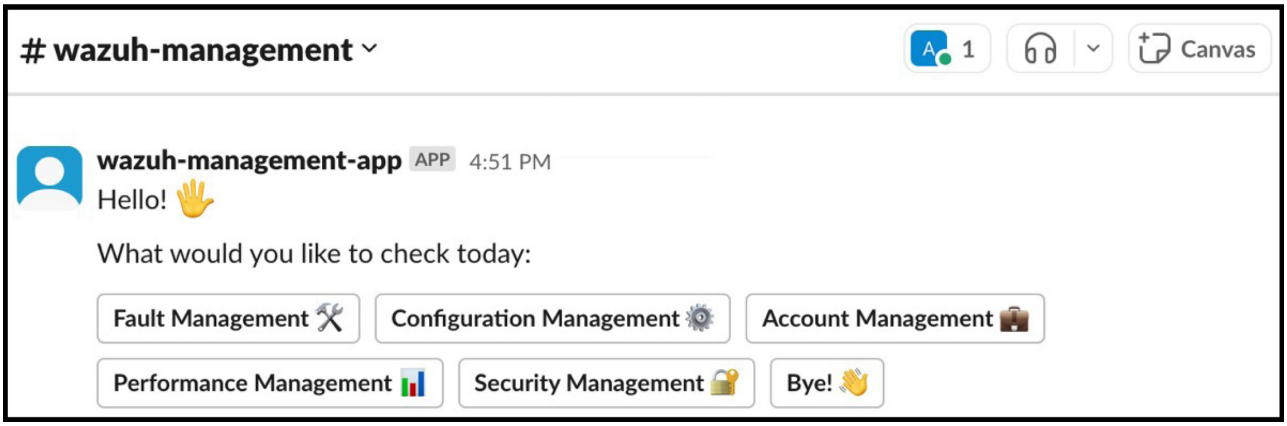

Figure 18. Initial user menu in the slack channel.

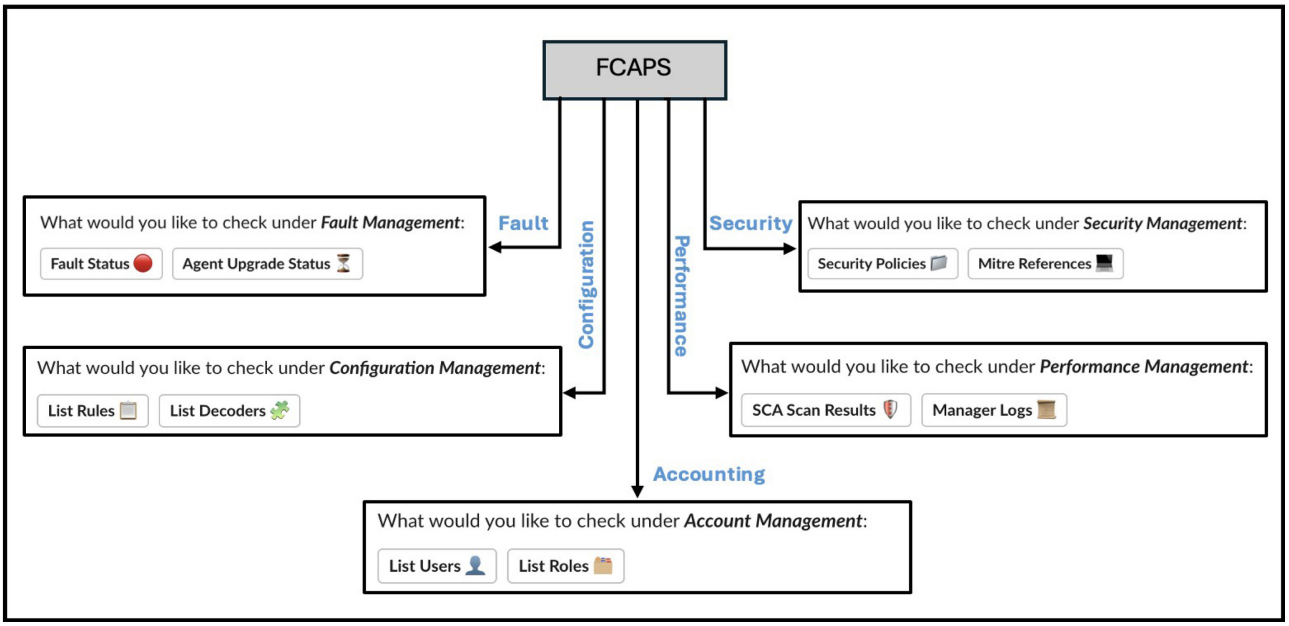

Figure 19. Screenshot of individual messages for each category of FCAPS.

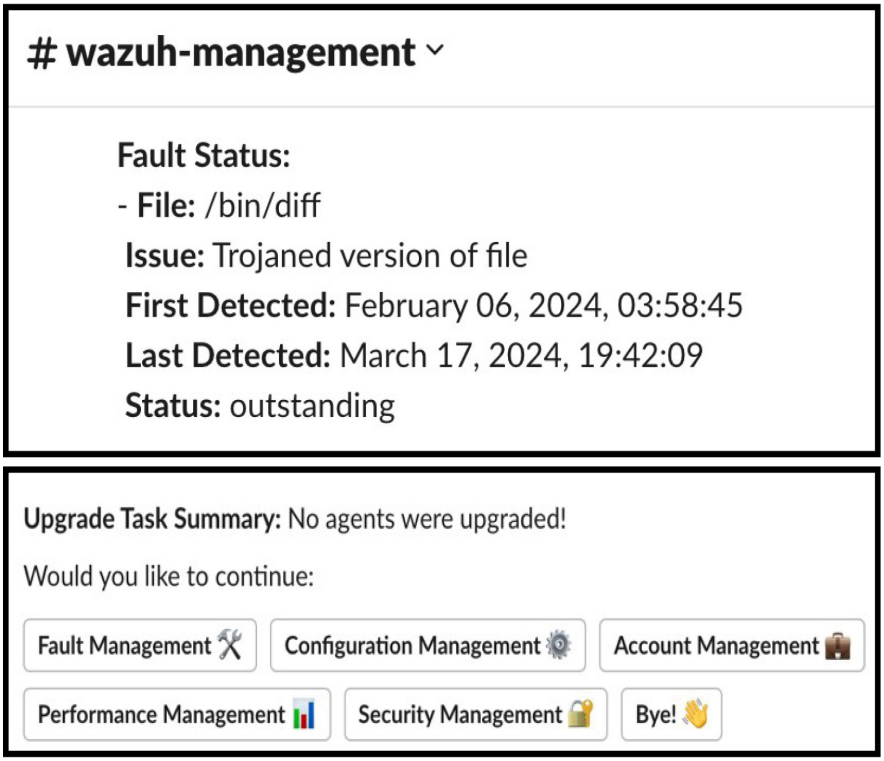

Figure 20. Fault Management submenu options.

Figure 21 represents the Rules and Decoders. The figure show only one rule and one decoder, but Wazuh consists of over 6500 rules and approximately 600 decoders. Figure 21 (top image) shows the required details on each rule including the name and directory of the rule. Figure 21 (bottom image) shows the details of the decoder including the name and the pattern that is used by Wazuh for parsing and decoding data.

Figure 22 represents the Users and Roles. Figure 22 (top image) provides all the users and the roles they are mapped to. Figure 22 (bottom image) provides all the roles and the policies that each role is mapped to. Together, users can map the users to their policies.

Figure 23 represents the SCA Scan Results and Manager Logs. Figure 23 (top image) provides the total number of checks that have passed, failed, and considered invalid. Since the host machine is an Ubuntu machine, SCA scan was performed for CIS Benchmarks for Ubuntu. Figure 23 (bottom image) provides the latest Wazuh manager logs.

Figure 24 represents the Security Policies and MITRE References. Figure 24 (top image) provides all the details on the security policies that are configured in Wazuh. Figure 24 (bottom image) represents one of the MITRE references that Wazuh uses for identifying MITRE ATT&CK in the host machine. The image represents a limited set of security policies and MITRE references, but there are 250+ security policies and 100+ MITRE references that are configured in Wazuh.

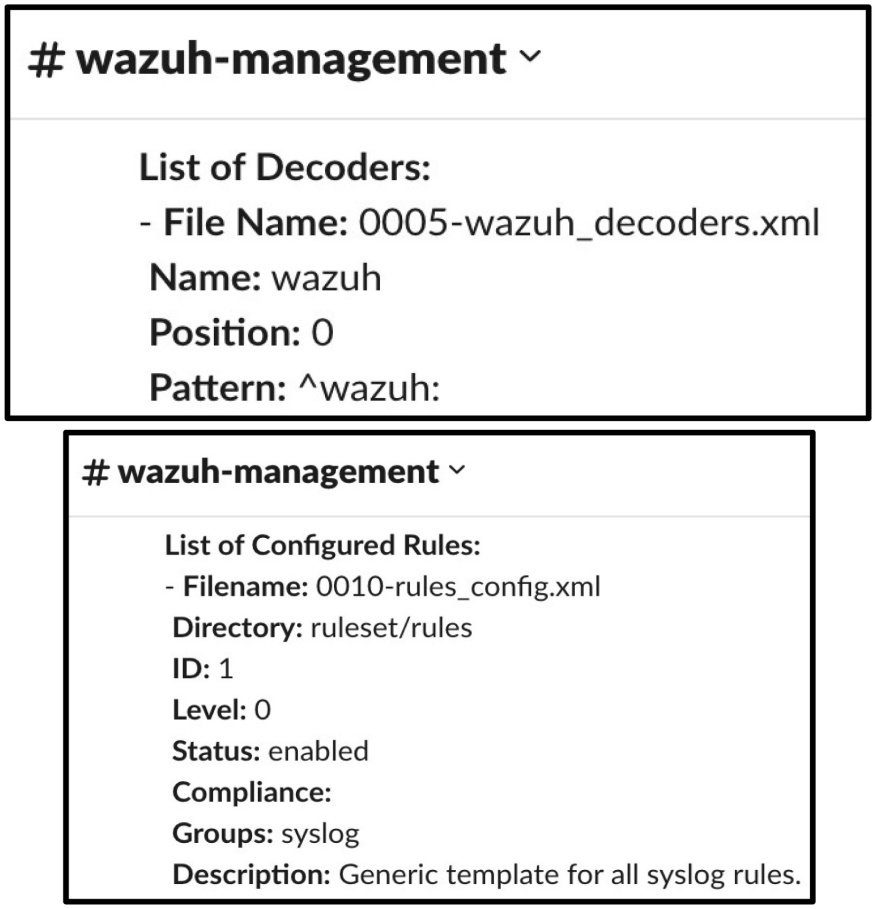

Figure 21. Configuration management submenu options.

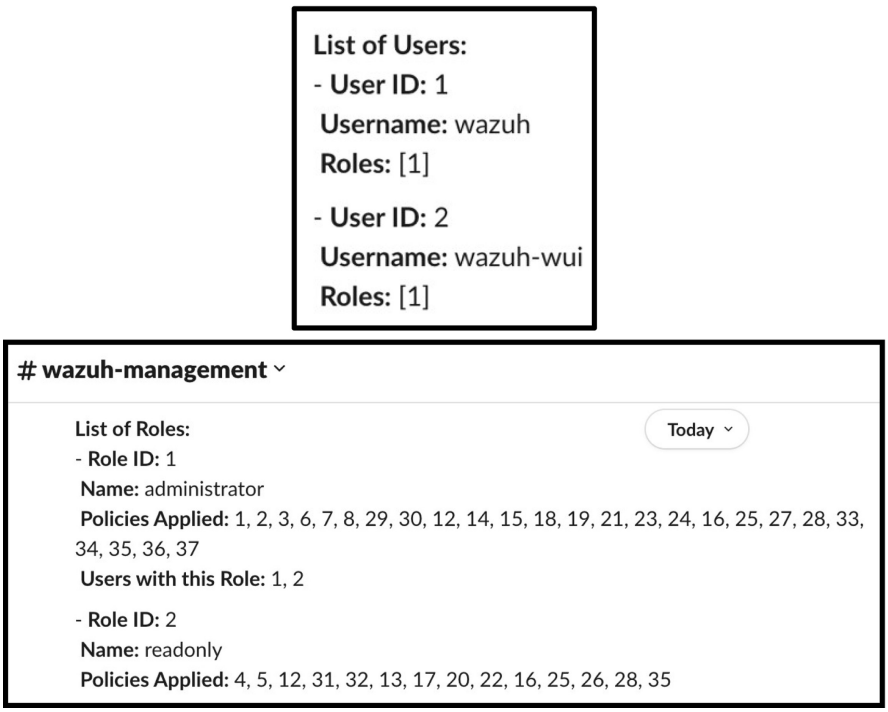

Figure 22. Accounting management submenu options.

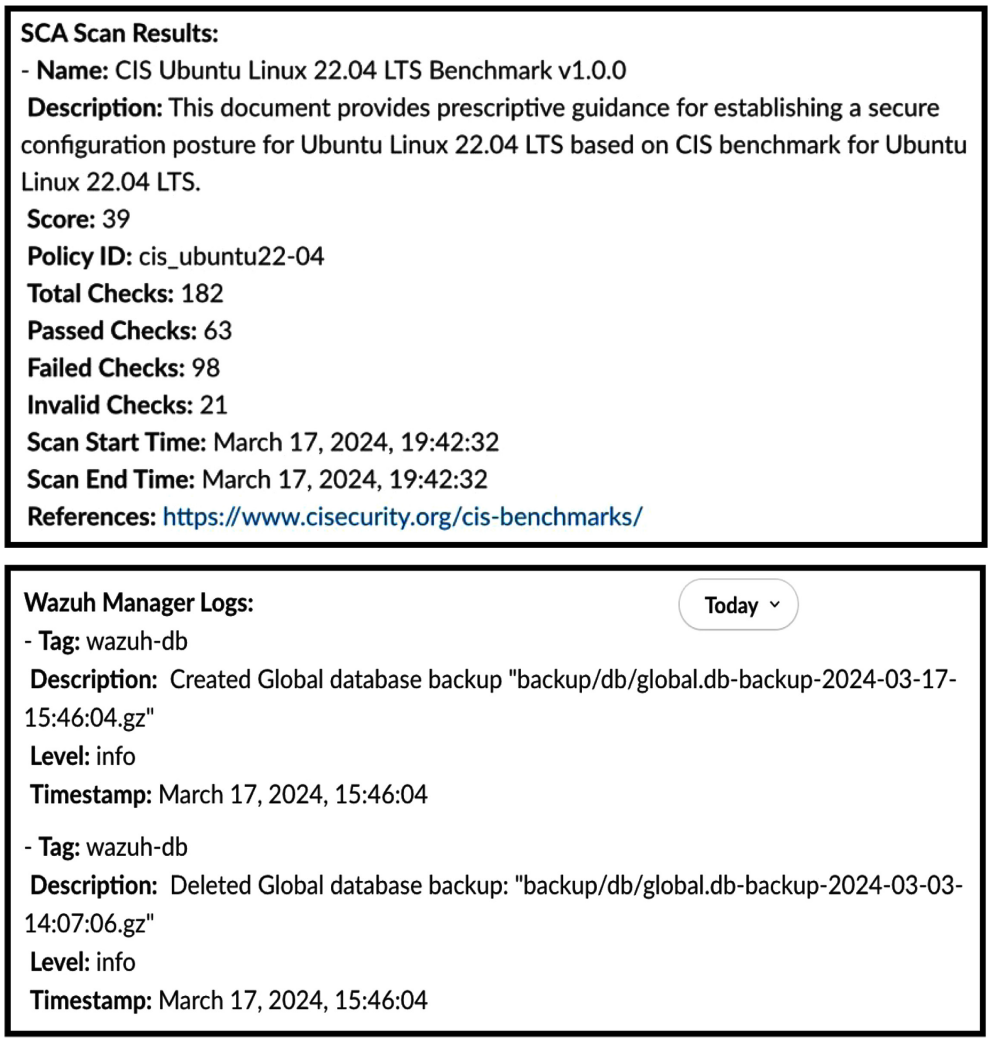

Figure 23. Performance Management submenu options.

5.3. Discussion

This case study of Adrian clearly demonstrates the system's innovative approach to improving SIEM platforms' alert management and response mechanisms. By streamlining the flow of information between Wazuh and Slack, Adrian not only enhances the real-time alert distribution but also introduces a higher level of alert customization, alert categorization, and operational responsiveness.

6. Conclusion and Future Work

The results of our initial implementation of Adrian pave ways for several avenues for future research and development:

- **Platform Expansion**: The presented case study of Adrian is primarily focused on integrating Wazuh with Slack. However, it can be expanded to include other software such as Microsoft Teams, Lark, Google Meet, etc. Similarly, it can also incorporate OpenCTI, MISP, etc. This expansion would be useful in exploring Adrian's adaptation.

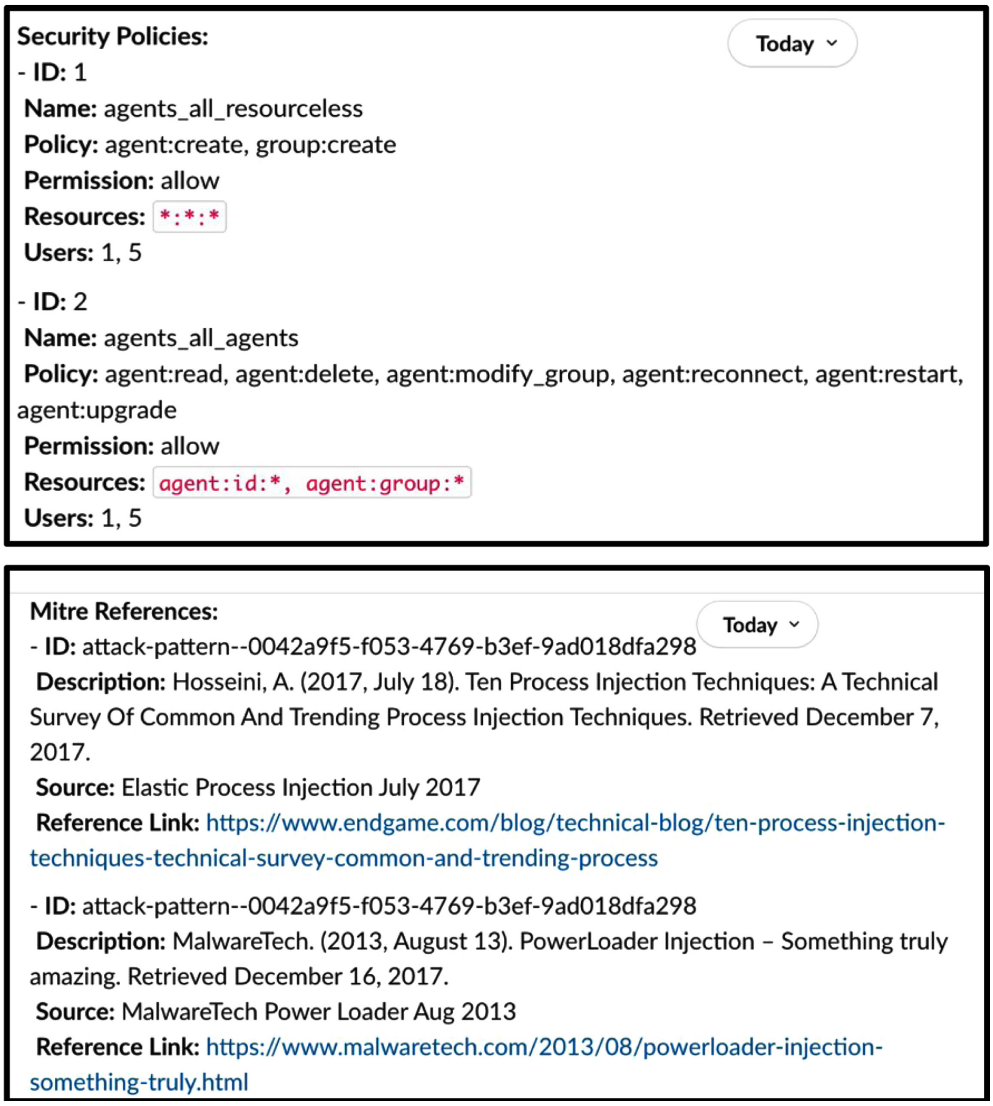

Figure 24. Security Management submenu options.

- **Advanced Features**: Incorporating artificial intelligence and machine learning techniques to enhance Adrian's alert prioritization and response automation capabilities could further improve the operational efficiency. The research in this path could lead to the development of intelligent modules within Adrian that would learn from the past incidents and optimize alert handling and response strategies.
- **Usability and User Experience**: User studies could be conducted to gather feedback on Adrian's interface and functionalities. The recommendation would be to collect feedback from the DevSecOps team members who could provide insights into usability improvements.

In conclusion, Adrian represents a significant advancement in streamlining security alert management and enhancing the response network for SIEM platforms. This integration ensures that the critical alerts are not only promptly delivered to the teams, but also the ability to respond and mitigate security inci-

dents. The separation of alerts from alert statistics enables the teams to take fast decisions and actions.

Furthermore, Adrian offers a scalable and a flexible solution to the security incident monitoring and response challenges. We are optimistic about Adrian's potential to contribute to a more security and efficient operational environment.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cinque, M., Cotroneo, D. and Pecchia, A. (2018) Challenges and Directions in Security Information and Event Management (SIEM). *IEEE International Symposium on Software Reliability Engineering Workshops* (*ISSREW*), Memphis, 15-18 October 2018, 95-99. https://doi.org/10.1109/ISSREW.2018.00-24

[2] González-Granadillo, G., González-Zarzosa, S., Diaz, R. (2021) Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors*, **21**, 4759. https://doi.org/10.3390/s21144759

[3] Sadowski, G., Kavanagh, K. and Bussa, T. (2020) Critical Capabilities for Security Information and Event Management. Gartner, 10-30. https://www.exclusivenetworks.com/se/wp-content

[4] Forrester Total Economic Impact (2023) The Total Economic Impact of IBM Security QRadar SIEM.

[5] RabbitMQ (2024) Installing RabbitMQ on Debian and Ubuntu. https://www.rabbitmq.com/docs/install-debian

[6] Wazuh (2024) Official Documentation. https://documentation.wazuh.com/current/user-manual/api/configuration.html

[7] Ngrok (2024) Slack + Ngrok. https://ngrok.com/partners/slack

[8] Grant Pennington (2020) RabbitMQ Tutorial and Python Demo. https://www.youtube.com/watch?v=wDv1iCMjypg

[9] CloudAMQP (2021) RabbitMQ Explained-Exchanges. https://www.youtube.com/watch?v=o8eU5WiO8fw

[10] Ram N Java (2023) RabbitMQ Direct Exchange Explained. https://www.youtube.com/watch?v=YDqlwRrno0w

[11] Soumil Shah (2020) Starting with RabbitMQ Using Python. https://www.youtube.com/watch?v=eSN0otKzYOE&list=PLL2hlSFBmWwy8lhnj11FVJldKsZm66oq1

[12] Geeksforgeeks (2023) Data Visualization with Python. https://www.geeksforgeeks.org/data-visual/

[13] Ian Webster (2019) How to Send Dynamic Charts with Slack Bot. https://quickchart.io/documentation/send-charts-with-slack-bot/

[14] Wazuh (2024) Integration with Third-Party APIs. https://documentation.wazuh.com/current/user-manual/manager/manual-integration.html

[15] Real Python (2020) Getting Started with the Slack API Using Python and Flask. https://realpython.com/getting-started-with-the-slack-api-using-python-and-flask/

[16] Splunk (2023) Common Information Model Add-on Manual. https://docs.splunk.com/Documentation/CIM/5.3.1 /User/Alerts

[17] Fireship (2020) How to Build a Slack App. https://www.youtube.com/watch?v=25ArxpK48tU

[18] Rohan Singh (2023) How to Run Python Flask App Online Using Ngrok? https://www.tutorialspoint.com/how-to-run-python-flask-app-online-using-ngrok

[19] Slack API (2024) Building an App with Bolt for Python. https://api.slack.com/start/building/bolt-python

[20] Peter Baumgartner (2016) Creating Slack Command with Python and Flask. https://pmbaumgartner.github.io/blog/slack-commands-with-python-and-flask/