

Name	Candidate number
Daniel Peter Mohr	115
Eirik Sunde	194
Kevin Kristian Adamsrød	224
Martin Røed Flyum	282
Patrik Abrahamsen	336

Emnekode: BAO302

Emnenavn: Bachelorprosjekt

Oppdragsgiver: SmartSecLab, SEIT, Kristiania

Innleveringsdato: 22.05.2024

Antall side: 51

Antall ord: 11628

Tilgjengelighet: Fri

Høyskolen Kristiania

Tittel på prosjektet - Norsk: **Trusselinformasjon og Oppdagelse av Anomaliteter i Web-servere: Planlegging og implementering av et spesialtilpasset logg-basert system for administrasjon av sikkerhetsinformasjon og hendelser**

Title of the project - English: **Threat Intelligence and Anomaly Detection in Web Servers: Design and Implementation of a Custom-made Log-based Security Information and Event Management System**



Semester Våren 2024

Denne bacheloroppgaven er gjennomført som en del av utdannelsen ved Høyskolen Kristiania. Høyskolen er ikke ansvarlig for oppgavens metoder, resultater, konklusjoner eller anbefalinger.

Summary

The team has developed "Glitnir," a custom Security Information and Event Management (SIEM) system for real-time threat detection in web servers. The system improves anomaly detection by integrating External Threat Intelligence feeds from sources such as ThreatFox, AlienVault, and AbuseIPDB. Achieved using a custom software written in Go to process IP addresses.

Glitnir's modular and scalable architecture allows for handling increasing data volumes and integrating new functionalities as needed. The system allows for static log analysis in combination with real-time threat intelligence to identify security breaches, thereby reducing the risks of cyberattacks such as DDoS and brute force attacks. Loki and Grafana are incorporated for better visualization, log data management and actionable insights.

This Thesis is presented in Chapters 1 to 7 with references made to respective Sections and Subsections. At the end of the thesis, the references are stated as "Bibliography" (Bbl. nr), "External Documentation" (ED. nr), Abbreviations, Appendixes A to B, and finally "Tables and Figures".

Acknowledgments

First and foremost, we sincerely thank Tomas Sandnes, our primary supervisor, for his unwavering support, insightful feedback, and invaluable mentorship throughout the process of this thesis. His expertise and enthusiasm for the subject have been a constant source of inspiration and guidance.

We are also immensely grateful to Nikola Gavric, who has played a pivotal role as our supervisor. His pragmatic advice, thorough understanding of the subject, and willingness to share his knowledge have been instrumental in the successful progression of this project.

Special thanks are due to Andrii Shalaginov, who has been responsible for our bachelor's degree program. His dedication to academic excellence and commitment to fostering a supportive learning environment have enriched our educational experience at Kristiania University College.

We would also like to acknowledge the entire teaching collegium at Kristiania University College for their contributions to our academic growth. Their collective expertise has greatly influenced our understanding, approach, and process contained in this thesis.

Lastly, we thank our friends and family for their unwavering support, patience, and belief in our abilities. Their encouragement and assistance have been a constant source of motivation throughout this academic journey.

The Team Presentation.

Team Members and Primary Roles	Description
Daniel Peter Mohr Project Manager: Main point of contact for communications and Threat Detection	<ul style="list-style-type: none"> - Serve as the primary liaison between the team and external parties, ensuring clear and effective communication. - Collaborate with the team and facilitate a risk-level assessment of various anomalies detected. - Lead the development of the underlying methodology to detect, analyze, and further improve the detection of security incidents and anomalies, coordinating with the “Threat Detection and System Administration Team” to understand the impact and root cause.
Eirik Sunde Scrum Master: Documentation Lead, Meeting Coordinator	<ul style="list-style-type: none"> - Oversee the creation and maintenance of all project documentation, ensuring accuracy, clarity, and timeliness. - Organize and facilitate team meetings, set the agenda, and ensure all relevant issues are discussed and handled. - Develop and maintain a systematic approach to document the meeting minutes, project progress, and decision-making processes. - Ensure that all documentation and visual data representations adhere to project standards and are easily accessible to all team members.
Kevin Adamsrød Development team member: System Architect	<ul style="list-style-type: none"> - Manage and maintain the project’s technical infrastructure, including servers, databases, and network systems. - Ensure the reliable performance of the system by monitoring system health, conducting regular maintenance, and applying reasonable updates and patches. - Collaborate with the Development Team to provide necessary technical support and guidance in system-related aspects of the project. - Document system configurations, changes, and procedures to maintain a clear record and facilitate the sharing of knowledge among the team members.

<p>Martin Flyum Development team member: Team Coordinator & System Developer</p>	<ul style="list-style-type: none"> - Design, develop, and oversee the customer and team interface of the applications. - Coordinate with team members to gather and disseminate project updates, feedback, and important notices. - Develop and set up an environment suited for penetration testing and log generation. - Provide insights for post-operation reviews and collaborate with the rest of the team to ensure continuous improvement of anomaly detection.
<p>Patrik Abrahamsen Development team member: Threat Detection and Mitigation</p>	<ul style="list-style-type: none"> - Coordinate and collaborate with the team to implement advanced strategies for detecting potential security threats using log analysis, monitoring tools, and threat intelligence. - Analyze detected threats to assess their potential impact and urgency, providing actionable insights for response and mitigation, and collaborate with the Incident Response Coordination team to prioritize and categorize threats to assess their urgency and potentially implement mitigations.
<p>Tomas Sandnes Hanne Sørum, Nikola Gavric and Andrii Shalaginov: Representatives of Kristiania University College, Product Owner and Stakeholder</p>	<ul style="list-style-type: none"> - Board Members - Faculty Leaders - Professors - Academic Directors - Counselors

Table of Content

1. Introduction.....	7
1.1 Enhancing Web Server Security Through Log Analysis.....	7
1.2 The Aim of this Thesis.....	7
1.3 The Inquiry Formulated in this Thesis.....	8
1.4 The Significance of this project.....	8
1.5 Scope and Limitations.....	8
1.6 Technical Software Components Terminology.....	9
1.6.1 Script.....	9
1.6.2 Module.....	9
1.6.3 Component.....	9
2. Literature Review.....	9
3. Project Management.....	11
3.1 Scrum.....	11
3.1.1 Team Roles.....	11
3.1.2 Sprint Planning.....	11
3.1.3 Daily Stand Up.....	14
3.1.4 Weekly Planning.....	14
3.1.5 Sprint Reviews and Retrospectives.....	15
3.1.6 Notion and Kanban.....	15
3.2 Documentation and Communication.....	17
3.3 Project Scheduling.....	17
3.4 Resource Management.....	18
3.4.1 Hardware and Software.....	19
3.4.2 Human Resources.....	19
3.5 Risk Management.....	19
3.5.1 Risk Exposure and Mitigation Strategies.....	19
4. Project Methodology.....	20
4.1 Description of the Research Design and Approach.....	20
4.1.1 Data Collection.....	20
4.1.2 Data Processing.....	21
4.1.3 Data Analysis and Alerting.....	22
4.2 Details of the Data Collection Process.....	23
4.2.1 Go Modules.....	23
4.2.2 The Security Information and Event Management.....	25
4.3 Tools and Techniques Explained.....	26
4.3.1 Log Analysis.....	26
4.3.2 Anomaly Detection.....	26

5. System Architecture and Implementation.....	29
5.1 Description of the system setup.....	29
5.1.1 Log Collection Environment.....	29
5.1.2 Mounted drive.....	29
5.1.3 Log Receiver and Analysis Environment.....	29
5.2 Explanation of Procedures.....	31
5.2.1 Log Collection Process.....	31
5.2.2 Log Collection Pipeline.....	31
5.2.3 From Logs to Insights: Analysis and Visualization.....	31
5.2.4 Bridge Middleware and Modular Architecture.....	32
5.3 Implementation of Anomaly Detection Features.....	33
The Anomaly Detection Process:.....	33
5.3.1 Go Modules.....	33
5.3.2 Indicators of Compromise.....	38
6. Results and Discussion.....	39
6.1 Key Findings Summary.....	39
6.1.1 Accomplishments.....	39
6.1.2 Functionalities.....	39
6.1.3 Implications.....	40
6.1.4 Limitations and Issues.....	41
6.1.5 Project Execution.....	42
6.1.6 General Data Protection Regulation (GDPR).....	42
6.1.7 Risk Management.....	42
6.1.8 The coding to attain successful Anomaly Detection.....	43
7. Conclusion and Recommendations.....	43
7.1 Conclusions.....	43
7.2 Recommendations.....	44
Bibliography (Bbl.).....	45
External Documentation (ED.).....	48
Appendix:.....	50
Appendix A.....	50
Appendix B.....	51

1. Introduction

1.1 Enhancing Web Server Security Through Log Analysis

The ever-growing reliance on web servers powering countless websites and applications is crucial to our daily lives, yet makes the servers prime targets for cyberattacks. These attacks range from data breaches, as detailed in IBM's 2024 report^(Bbl. 1), to the rising incidence of denial-of-service attacks, which are becoming more frequent and sophisticated according to a recent report by Radware (Dyrek, 2024)^(Bbl. 2). Such vulnerabilities can have significant impacts, underscoring the increasing importance of ensuring web server security through constant vigilance and the development of innovative solutions.

This thesis addresses anomaly detection on a Linux Nginx web^(ED. 1) server by leveraging a Security Information and Event Management (SIEM) system. SIEMs analyze complex web server log-data to identify potential breaches. Traditional methods often struggle with high false positives as they lack real-time capabilities, hindering timely responses. Research shows that over 92% of false positives in intrusion detection systems stem from management policies, not threats (Cheng-Yuan et al., 2012)^(Bbl. 3), and such inefficiency highlights the need for more advanced, adaptive solutions.

We deployed a Linux Virtual Machine (VM) hosting an Nginx web server integrated with a custom SIEM system within a cloud environment to address these limitations. To enhance log analysis capabilities, the configuration enabled a more precise and timely identification of potential threats. Ultimately, the objective is to bolster web server security by improving the effectiveness of anomaly detection, thereby mitigating the risks and impacts of cyber threats.

1.2 The Aim of this Thesis

The aim of this thesis is to develop and evaluate a prototype anomaly detection system for a Linux-based Nginx web server, analyze the system and service logs to identify potential security threats and explore the integration of External Threat Intelligence (ETI) feeds to enhance the system's capabilities. Through this project, the team will gain insight into the feasibility and effectiveness of such a system for real-time threat detection.

1.3 The Inquiry Formulated in this Thesis

The team has postulated the following Inquiry:

“How can enhanced anomaly detection techniques be applied to the system and service logs of an Nginx web server in a Linux environment to improve the identification and mitigation of cybersecurity threats?”

1.4 The Significance of this project.

This thesis contributes to the ongoing exploration of effective methods for web server anomaly detection. By investigating the integration of SIEM systems with emerging threat intelligence, this thesis aims to provide insights into the potential of this approach for real-time threat detection and mitigation.

The team may contribute valuable data points to developing a more comprehensive understanding of anomaly detection systems. Additionally, this project reinforces the importance of log analysis in cybersecurity practices.

1.5 Scope and Limitations

Our project explores SIEM and threat intelligence integration for anomaly detection in web servers built on a cloud-based Nginx server and a corresponding SIEM system to simulate real-world interactions. This comprehensive environment generates and analyzes logs, allowing us to assess the effectiveness of ETI feeds in enhancing anomaly detection.

Custom software for supporting anomaly detection scripts will also be developed to improve threat identification further. Real-time metrics and visualizations will be used to understand the server's security posture under attack scenarios.

Reliance on a cloud environment introduces external dependencies and potential network stability concerns. Additionally, simulated attacks may not fully represent real-world threats, and the SIEM's rules and initial anomaly detection algorithms might require further refinement.

1.6 Technical Software Components Terminology

1.6.1 Script

Throughout this project, the team will refer to scripts as defined in The Portable Operating System Interface (POSIX)^(Bbl. 4), meaning a text file with a set of commands sequentially read and executed by an interpreter, such as the shell.

1.6.2 Module

The term ‘module’ is used to refer to a self-contained, custom-built, or open-source integrated functional component for the bridge, named “Bivrost”. The modular approach to the system architecture facilitates greater visibility in the program control flow, which expedites debugging and ensures ease of future expandability. By design, due to the nature of the ever advancing and rapidly evolving cyber security threat landscape, there is a need for constant iterations on both sides (adversary vs. defender).

1.6.3 Component

In relation to the software solution as a whole developed for this project, a component is any piece of software, such as a file system, database, script, or executable. As long as the component is self-contained, with the purpose of contributing to the system as a whole.

2. Literature Review

The team sought to provide a custom SIEM, solidly founded on existing research, yet built and expanded on the latest techniques and information that the team had attained during the Cyber Security course at Kristiania University College. Inspiration for our work on this thesis has been taken from the literature referenced below. We addressed technical risks through established testing procedures and quality assurance practices.

Overview of Existing Research into Anomaly Detection.

The design of our thesis is inspired by the five external documents below. Some of these documents extend beyond the scope of our thesis, yet we found them of value as we tested the possibilities for Machine Learning early on. Please note that these studies are taken as inspiration, not citations.

Doc. 1 An Empirical Investigation of Practical Log Anomaly Detection for Online Service Systems^(Bbl. 5)

This paper provides an overview of log parsing techniques, an important step in log analysis for anomaly detection. It compares various log parsing methods and underlines the importance of selecting a technique that aligns with the specific characteristics of the logs generated by Nginx servers.

Doc. 2 Detecting Anomalies in System Logs^(Bbl. 8)

The document introduces an open-source log management tool, emphasizing the necessity of real-time anomaly detection. The study's exploration of Artificial Ignorance and its application in filtering log messages is highly relevant to our goal of developing a SIEM system capable of discerning potential threats.

Doc. 3 Experience Report: System Log Analysis for Anomaly Detection^(Bbl. 9)

The report takes six log-based anomaly detection methods and evaluates them by bridging the gap between academic research and industry practice. The paper offers a systematic comparison and the release of an open-source toolkit that can significantly influence our project's choice of anomaly detection techniques and how we implement them.

Out of Scope

We had the intention of using the content presented in the following documents, however, time constraints prevented us from doing this.

Doc. 4 Anomaly Detection in Log Files Using Machine Learning Techniques^(Bbl. 6)

This paper discusses the application of machine learning algorithms for detecting significant events in log files. The effectiveness of methods such as the Local Outlier Factor, Random Forest, and Term Frequency Inverse Document Frequency (TFIDF) in identifying anomalies could provide a theoretical basis for our thesis's approach to leveraging machine learning for enhanced log analysis in the event of such implementation.

Doc. 5 Anomaly Detection for Linux System Log^(Bbl. 7)

By evaluating workflow-based and Principal Component Analysis (PCA)-based methods, this document highlights the adaptability and efficiency of machine learning in parsing and analyzing logs for anomaly detection.

3. Project Management

This section delves into the strategies and tools employed to guide the development and execution of this project. We will begin by exploring the chosen methodology, Scrum, section 3.1, emphasizing how it facilitated iterative development and efficient teamwork. Following this, section 3.2, (page. 17) focuses on Documentation and Communication, outlining the methods used to capture project details and maintain clear communication channels with the stakeholders. Next, section 3.3, (page. 17) examines Project Scheduling, detailing the techniques used to create a realistic timeline for project milestones and deliverables. section 3.4, (page. 18) dives into Resource Management, exploring how human and material project resources were allocated and utilized effectively. Finally, section 3.5, (page. 19) explores the strategies employed for Risk Management, highlighting how potential challenges were identified, mitigated, and managed throughout the project's lifecycle.

3.1 Scrum

In the ever-evolving realm of software development, staying at the forefront is essential for success. Scrum, a methodology that has significantly reshaped the industry, was chosen to elevate our development strategies. During our first key in-person meeting, we realized the need for an iterative approach, leading to the implementation of Scrum. While Agile and Scrum are distinct methodologies, Scrum is an Agile framework that facilitates collaboration and efficiency in software development and testing. Scrum-based development projects are broken into small builds called sprints, which consist of three components: product backlogs, sprint backlogs, and sprint goals. With each sprint, a specific function is defined, developed, and tested(Simplilearn, 2024)^(Bbl. 11).

3.1.1 Team Roles

A scrum normally consists of three primary roles: the Product Owner, The Scrum Master, and the Development Team. In our case, the Product Owner is Kristiania University College, the Scrum Master is Eirik, and finally, the Development Team (See 'The Group Presentation', page. 3-4).

3.1.2 Sprint Planning

Sprint Planning kicks off the sprint. Sprint planning aims to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team (West, 2019)^(Bbl. 12). We used the three 'Ws' and the two 'puts'; The What, The How, The Who, The Inputs and The Outputs.

The What

'The what' handles the objectives and Backlog items. At the initial meeting held between the external supervisor for the Product Owner and the representatives of the team, the bachelor assignment and the details around it were shared with the group. The group had the opportunity to ask any question in order to clarify any immediate doubt concerning the scope and limitations.

The How

‘The how’ is oriented around work planning. After handling the objectives and backlog items, the team held a key meeting where time frames, work schedules, milestones, and sprint goals were discussed and agreed upon. We decided on six sprints, with one to three milestones for each. These milestones were then broken down into smaller tasks, referred to as ‘a story’, before breaking the stories down to tasks.

The Who

Scrum emphasizes collaboration, particularly during sprint goal definition. Ideally, the Product Owner defines goals based on desired value delivery while the development team clarifies their ability to achieve them. However, in our case, initial communication with the Product Owner was limited due to two factors:

- **Tight Deadlines:**
The Product Owner's workload managing multiple student groups made frequent meetings challenging.
- **Unique Project Nature:**
Being the first of its kind for the Cybersecurity Bachelor's program, the project lacked a clear roadmap compared to previous projects. The team was therefore granted more autonomy in defining project scope and milestones, but it also necessitated effective communication to ensure alignment with the Product Owner's expectations.

After receiving the initial proposal, the team opted for a more challenging assignment, demanding a more strict but realistic timeframe and sprint management. The team proposed a project lifecycle of six sprints (see Figure 1, page 13).

Aa Sprint name	⚙ Sprint status	📅 Dates
🚀 <u>Sprint 1 - Requirements</u>	● Past	January 8, 2024 → January 31, 2024
🚀 <u>Sprint 2 - System Setup</u>	● Last	January 29, 2024 → February 16, 2024
🚀 <u>Sprint 3 - Dev Phase I</u>	● Past	February 5, 2024 → March 1, 2024
🚀 <u>Sprint 4 - Dev Phase II</u>	● Past	March 4, 2024 → April 19, 2024
🚀 <u>Sprint 5 - Testing and Optimization</u>	● Current	April 22, 2024 → May 3, 2024
🚀 <u>Sprint 6 - Project finalization</u>	● Future	May 6, 2024 → May 15, 2024

Fig. 1: The six sprints of the Bachelor Project

The Inputs

‘The Inputs’ refers to the essential elements considered at the beginning of sprint planning. These inputs, such as backlog items, user stories, technical dependencies, etc. guided the team in determining what can realistically be achieved in the upcoming sprints, ensuring alignment with overall project goals and available resources. As the team looked at work done, intermittently we could estimate the necessary work capacity for the future sprints. This meant that the team considered the existing work completed in the previous sprint and the team's current workload when planning the upcoming sprints. The agreed approach helped to avoid overloading team members and ensured realistic commitments.

In our case, the team undertook a bachelor assignment that was more comprehensive than the one originally proposed. Throughout the project, we monitored the team's capacity to handle the increased workload and resource demands, especially in consideration of the available bandwidth at the cloud provider, Digital Ocean.

The Outputs

While ‘The How’ tackles work planning and ‘The Inputs’ handle the essential elements considered at the beginning of sprint planning, ‘The Outputs’ centers around the sprint planning results. This demanded a focus on two themes: Sprint Goals and Sprint Backlog.

During the early planning phase, we discussed what we would need for the entire project, then split them into milestones, which we discussed during ‘The How’. To define each sprint’s goals with certainty, we first needed to understand what had to be completed before beginning the next sprint. To analyze logs, we first needed to collect them. Similarly, to compare metrics of 'normal operations' with 'penetration operations,' we had to establish baseline metrics before evaluating them under stress and duress.

The Why

Additionally, there is ‘The Why’. While ‘The Why’ is not a part of the Three W’s and the Two Puts, it remains an important anchor in this bachelor project. Beyond the "what" and "how" of sprint planning lies the fundamental question: “Why did we make certain choices?”

The initial project scope presented an opportunity for a more in-depth learning experience. We opted to propose a more challenging project, prioritizing valuable knowledge gained even if it meant an increased workload and potentially more complex execution. Once again, due to the unique nature of this thesis, we relied on our understanding of the project goals and internal discussions to move forward with increased team autonomy.

Finally, we considered the optimal sprint structure. Shorter sprints offer more frequent opportunities for course correction, while longer sprints can benefit from sustained focus. Inspired by the principles outlined in LeanWisdom^(Bbl. 13), we balanced agility with focus by choosing shorter sprints for phases with higher uncertainty, such as the early stages of development, and opting for longer sprints for well-defined tasks.

3.1.3 Daily Stand Up

Based on the experience and knowledge gained from an earlier software project, the group committed to including three types of meetings. First, the Daily stand-up was conducted as a semi-mandatory meeting, and it rarely went unattended. There was no specific agenda for the Daily stand-up, which only had one primary objective. To have everyone gathered in front of the workspace, share some laughter, and begin the day with enthusiasm. As a group, we felt that this approach was instrumental in generating the motivation and energy required to complete this Bachelor's project.

3.1.4 Weekly Planning

Weekly planning was a second type of meeting. It was a mandatory meeting at 10 AM each Monday. Everyone joined together online in a digital meeting. We knew the importance of this meeting was paramount, as it laid the foundation for the work that would be addressed this week and summarized the previous week.

The Scrum Master usually posted an agenda for the Monday Weekly planning early each Monday. The agenda addressed important topics, such as ‘Last week—the good and the bad’, and ‘what are we doing this week?’. We handled important reminders, such as delivery dates and meetings, allowing us to address any unforeseen situations and then consider rescheduling or changing the order of tasks as necessary. All meetings were documented thoroughly on Notion and in an independent management document, where our “mantra and guiding principles” were written down.

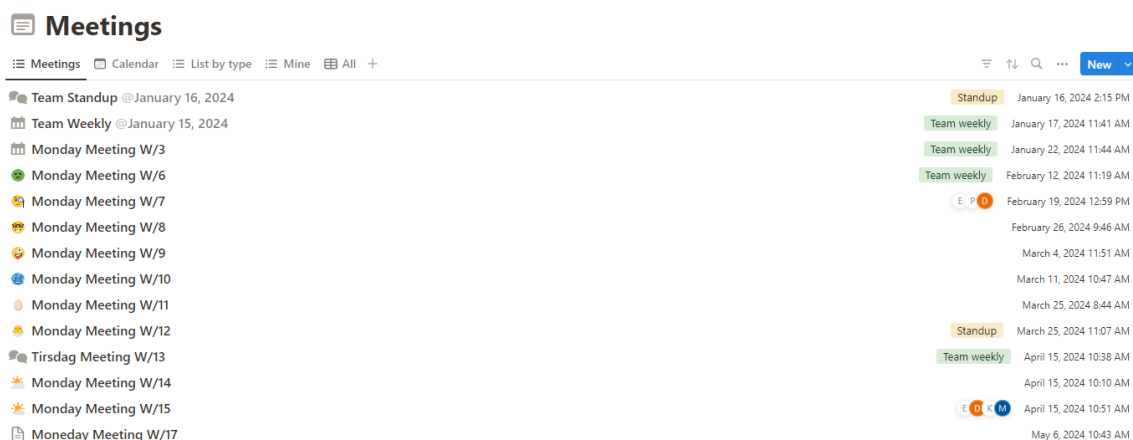


Fig. 2: Our Weekly Meetings well documented throughout the project lifecycle

3.1.5 Sprint Reviews and Retrospectives

The Sprint Reviews and Retrospectives were a third type of meeting, and the most important one. These meetings covered a series of important topics. The two main topics were; what is left in the backlog for the newly finished sprint and the milestones and goals set for this sprint. To address the previous Sprint, we used something called the 4 Ls. The 4 Ls is a retrospective technique where team members identify what they loved, loathed, learned, and longed for in a project or sprint of work. This allowed us to reflect on our work and use what we have learned to improve as a team and avoid errors and mistakes from previous sprints.

3.1.6 Notion and Kanban

Trello's Kanban-style boards initially facilitated task management by providing clear visibility into project progress and individual responsibilities. However, as the project's complexity grew, we transitioned to Notion. Notion offered a more robust framework for managing extensive documentation, integrating databases, and maintaining a better record of notes, timelines, and milestones. Notion consolidated these functionalities within a single platform.

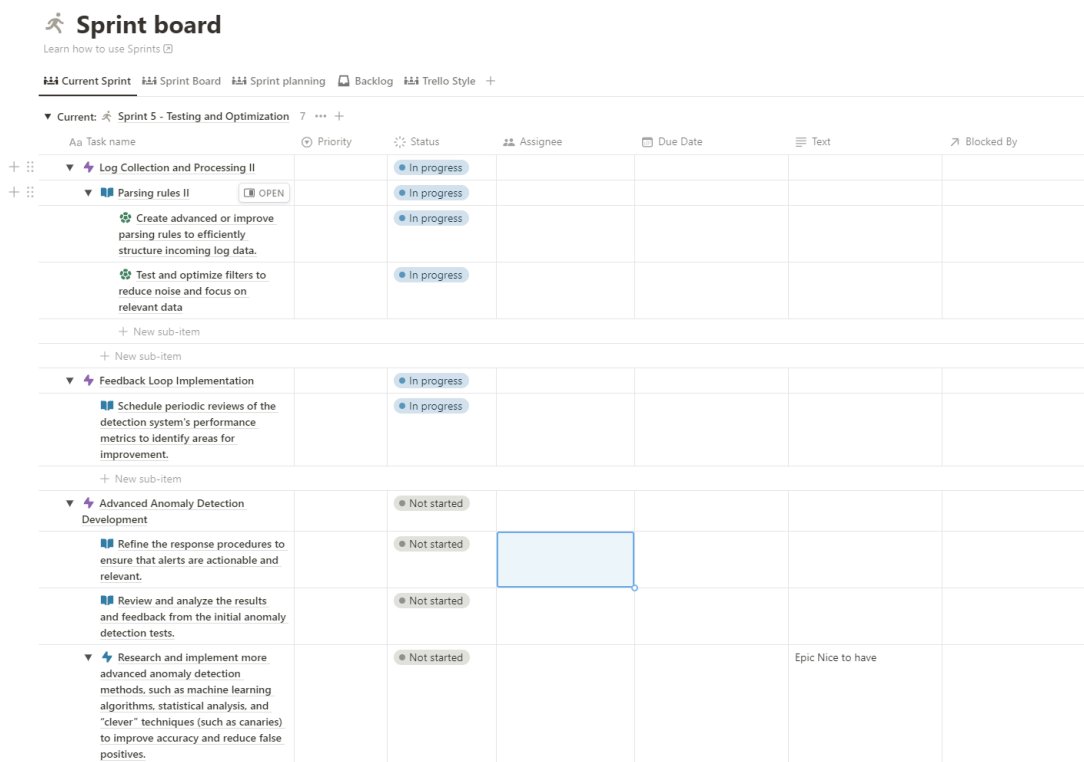


Fig. 3: Example on how our Notion Sprint Board looked like before the start of a sprint

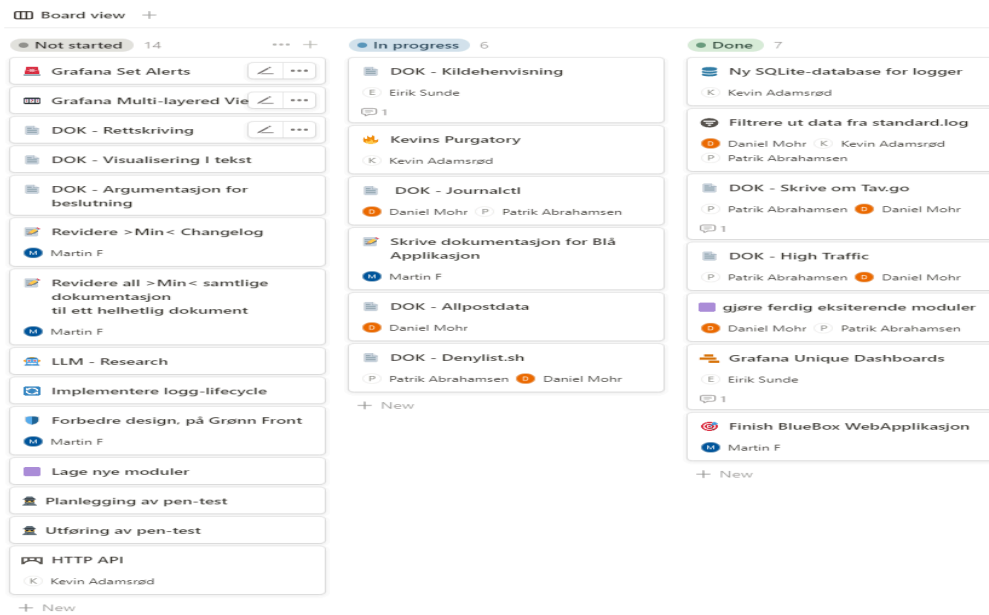


Fig. 4: How our Notion Kanban style looked like during the final stages of development

3.1.7 Visual Brainstorming - MIRO

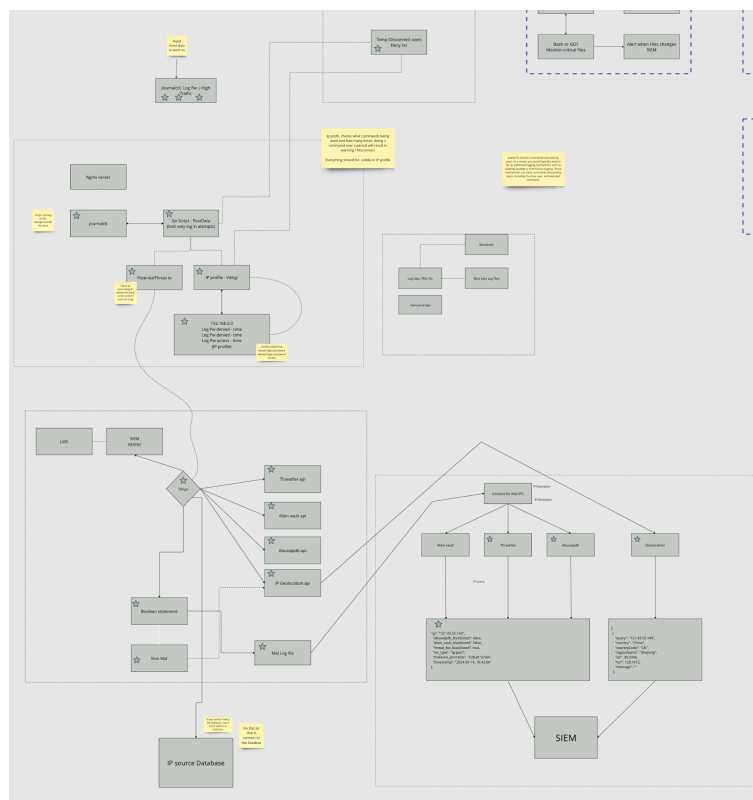


Fig. 5: Miro Board project during the early phases.

Miro is a tool that assists in planning strategies and layouts during our brainstorming sessions. At the start of our project, the team used Miro to collaborate in a digital workspace, enabling real-time idea sharing and various brainstorming techniques. Employing Miro helped enhance our concepts and efficiently structure the framework of our thesis from the beginning.

3.2 Documentation and Communication

The team members divided the work, generally following their respective responsibilities. Generating one product in sections requires thorough communication between the team members. As the work progressed, the members continuously updated and disseminated the documentation. The “Team Weekly Stand-Ups” presented the latest updates each Monday. Further, the latest work was noted and explained on the Notion Board for all to be informed. The communication was done online through “Discord”.

Further, to stay on course with the Product Owner's requirements, the team requested monthly Zoom meetings with the Product Owner. The Product Owner, however, did not make specific demands yet let the team devise the direction, content, and scope of the thesis project based on their knowledge gained throughout the CyberSecurity course. This shared strategy was in accordance with the purpose of the thesis, whereby the students showed their creativity and knowledge in the thesis design, and the team expanded the task to become a comprehensive custom SIEM for an Anomaly Detection and Threat Warning system.

3.3 Project Scheduling

The project followed a planned and phased approach to achieve our ultimate goal: developing a robust, custom-made SIEM system. Each phase is built upon the previous, ensuring steady progress.

Project Timeline:

- Initial Planning and Setup (January 5th - January 19th):
 - Defined project scope and objectives, transitioning to DigitalOcean for a scalable cloud solution.
 - Established foundational elements: role assignments, initial risk assessment, and project description.
 - The comprehensive team meeting on January 8th was a critical milestone, ensuring project alignment.
- Research and Training (January 22nd - January 31st):
 - Equipped the team with essential knowledge and skills in Nginx, Linux, SIEM systems, and Go programming language.
 - This phase included structured learning sessions, hands-on practice, and resource compilation for a solid technical foundation.

- System Setup (January 29th - February 16th):
 - Established the testing and development environment, configured the cloud infrastructure, and installed necessary tools and applications.
 - This phase focused on building the infrastructure for the SIEM and web server environment.
- Development Phase I (February 5th - March 1st):
 - Created the SIEM solution's core components, initiated log collection, and developed initial anomaly detection scripts.
- Development Phase II (March 4th - April 19th):
 - Advanced anomaly detection development, analyzed opportunities for penetration test results for system refinement, expanded log collection and processing capabilities and implemented a feedback loop for continuous improvement.
- Testing and Optimization (April 22nd - May 3rd):
 - This phase finalized the system, optimized performance, and addressed outstanding issues.
 - The goal was to ensure the system met all predefined standards and objectives.
- Project Finalization (May 6th - May 15th):
 - Completed comprehensive project documentation, conducted final system testing, and finalized the GitHub repository, preparing the system for deployment.
- Delivery and Presentation Preparation (May 20th):
 - The project concluded with delivering all project artifacts and preparing for a presentation (scheduled after May 22nd).

3.4 Resource Management

Effective resource management involves strategically selecting technological groundwork and the required human resources. For our custom SIEM system, this meant carefully considering the hardware and software necessary to support its functionalities and the skills needed to operate it effectively. These strategic decisions align with best practices in project resource management, as discussed in the article ‘What is Project Resource Management?’ (Brown, 2024)^(Bbl. 14), which highlights the importance of aligning resources with project demands to enhance efficiency and success.

3.4.1 Hardware and Software

This project necessitated a reliable hardware foundation for advanced anomaly detection and data processing capabilities. DigitalOcean's cloud infrastructure provided this foundation through the deployment of scalable VMs. These VMs hosted a Linux-based Nginx web server and a custom SIEM system, which fulfilled the project's specific requirements. The cloud solution offered flexibility, enabling real-time adaptation to evolving demands while ensuring high availability and consistent performance.

Additionally, Miro emerged as an invaluable tool for brainstorming and strategic planning. Its interactive whiteboard allowed us to better understand and plan the dynamic mapping of ideas and project workflows. The visual collaboration tool simplified complex project elements into manageable tasks, enhancing the team's understanding and efficiency (See Figure 5, page. 16).

3.4.2 Human Resources

A diverse team of five individuals drove this project, each bringing unique skills essential to our success:

- One member, an expert in server system administration, ensured our infrastructure was robust and secure.
- Another combined expertise in servers, databases, and front-end development, creating a powerful yet user-friendly system.
- A member specialized in penetration testing and security, fortifying our defenses against potential threats.
- An experienced conditioner and security analyst provided a sharp eye for detail, identifying and solving issues before they escalated.
- The final member, with extensive experience in documentation and organization, kept the project on track and ensured effective communication and collaboration.

3.5 Risk Management

A risk assessment was conducted at the project outline to address potential challenges. The assessment identified technical vulnerabilities (e.g., system failures, integration problems, coding errors, hardware limitations), operational risks (e.g., resource limitations, communication breakdowns, sickness, time constraints, team conflicts), and misalignments (missing requirements, missing documentation, and dependencies). Based on these findings, a detailed risk management plan was developed, outlining specific mitigation strategies for each identified risk. (See: Risk Plan, Appendix A, page. 50).

3.5.1 Risk Exposure and Mitigation Strategies

The risk plan addressed each risk with targeted actions. Further, the team developed comprehensive Application Programming Interface (API) documentation to ensure compatibility before the system integration. Coding the anomaly detection and handling system was a critical factor in the project. The team employed continuous manual and automatic testing and system audits to ensure desired results. Although the systems are cloud-based, hardware capacity may be critical when handling extensive traffic.

Throughout the project the team has worked closely with the internal supervisor and the Stakeholders (Faculty representatives of the Product Owner) to avoid compliance failure of the task requirements. Close cooperation and mandating the documentation as a part of the development process has been prioritized. The team has maintained an open and cordial method of cooperation to avoid conflicts over the direction of the project as a whole. Each team member presided over their section of the project

4. Project Methodology

In addressing the complexity of log anomaly detection in web server environments, it is crucial to consider the challenges highlighted in recent empirical studies, which reveal significant gaps in current systems' ability to manage diverse and complex data patterns effectively (Nengwen et al., 2021)^(Bbl. 5). Our methodology is designed to overcome these limitations by integrating enhanced anomaly detection capabilities with domain-specific insights.

4.1 Description of the Research Design and Approach

4.1.1 Data Collection

The team focused on utilizing Indicators of Compromise (IoCs) to detect security issues in web server logs. IoCs serve as clues or red flags that signal potentially malicious activities. Users are alerted whenever a request is made to the server by monitoring the access logs. An example of an anomaly is observing HyperText Transfer Protocol (HTTP) 200 status codes for paths that should not be typically accessible, indicating unauthorized access attempts or successful exploitation of vulnerabilities like directory traversal attacks, error logs identifying misconfigurations, failed access attempts, or operational faults. While these issues might not always indicate malicious intent, they can expose vulnerabilities that could be exploited. Persistent or suspicious error patterns, such as repeated 'file not found' errors or upstream service (Nginx Upstream)^(ED. 2) failures, usually mean someone is trying to probe or compromise the server.

Attention to logs of failed Secure Sockets Layer (SSL) handshakes is another important part, as these can reveal attempts to bypass encrypted communications, flagging potential Man-in-the-Middle (MitM) attacks or highlighting problems with certificates. Examples could be expiration, misconfiguration, or vulnerabilities to attacks like SSL stripping or the infamous Heartbleed flaw.^(ED. 3)

Lastly, the team evaluated the outgoing network traffic against a list of known threat actors' Internet Protocol (IP) addresses' IoCs to see if our server is talking to any known threats. If it does, it could mean that the server is being controlled by someone else or that information is leaking.

Log Type	Information Collected	Potential Security Concerns
Access Logs	<ul style="list-style-type: none"> - Time of request - Requested resource path - User-agent (browser information) - IP address - HTTP status code (200 = success, 404 = not found) 	<ul style="list-style-type: none"> - Unauthorized access attempts (unusual paths, high volume of requests) - Successful exploitation of vulnerabilities (e.g., directory traversal attacks)
Error Logs	<ul style="list-style-type: none"> - Time of error - Error message - Affected component (database, file system) 	<ul style="list-style-type: none"> - Misconfigurations in the server or applications - Failed access attempts (repeated login attempts) - Operational faults that could be exploited (e.g., persistent "file not found" errors)
SSL Handshake Logs	<ul style="list-style-type: none"> - Time of handshake attempt - Cipher suite used - Certificate details (validity, issuer) - Error code (if the handshake fails) 	<ul style="list-style-type: none"> - Man-in-the-middle (MitM) attacks (attempts to bypass encryption) - Certificate issues (expiration, misconfiguration, vulnerabilities) - Attacks like SSL stripping or Heartbleed (depending on the error code)

Tbl. 1. Overview of log types, data captured, and associated security concerns.

Data Anonymization

As the analysis of web server logs is essential for security, it is imperative to prioritize user privacy, as this is increasingly recognized as a crucial matter in corporate behavior.

During the testing, the team ensured minimized, identifiable information by only collecting IP addresses submitted to existing trusted ETI APIs. For the continued data collection, the team anonymized the IP addresses using hashing or truncation techniques. Specifically, the truncation technique involved replacing a portion of the IP address, typically the least significant bits, with zeros. This method made an endpoint non-identifiable while retaining enough information for utility in security analysis. An "endpoint" is any device that connects to a network, such as computers, smartphones, or Internet of Things (IoT) devices. These endpoints can be entry points for security threats, making their protection vital for network security. The potential threat list will also maintain anonymization through Classless Inter-Domain Routing (CIDR)(Ellingwood, 2021)^(Bbl. 15) notation or hashed IP storage. Additionally, the data retention policy will dictate the duration of storage of anonymized data before its secure disposal. The approach aligned with advanced techniques discussed in the literature, such as those emphasizing the balance between privacy and utility in network trace anonymization using methods including truncation (Aleroud et al., 2016)^(Bbl. 16).

Companies are adapting to data anonymization by refining their data management practices to better align with consumer expectations and regulatory demands. Businesses also enhance their reputation and customer relationships by transparently managing data and respecting user privacy (Anant et al., 2020)^(Bbl. 17).

4.1.2 Data Processing

This section delves into the strategies employed to manage the substantial volume of log data generated by the system. As established, effective log management is a cornerstone of robust anomaly detection (Xu Zhang et al., 2019)^(Bbl. 10). To achieve this, the team took a multi-step approach that involved using several tools and carefully configuring them.

Streamlined Log Aggregation and Indexing

The initial step involves gathering and consolidating logs from diverse sources, including Nginx web server logs and the system journal managed by 'journalctl', which is further covered in subsection 4.2.1 - Go Modules (page. 23). The journal captures a wide range of operational data, encompassing access attempts, system events, and error messages, which are crucial for monitoring, as outlined by Crowdstrides 'What is Log Aggregation' (Sharif, 2022).^(Bbl. 18)

Promtail, a log shipper designed for seamless integration with Loki, is a central component in this context^(ED. 5, 6, 7, 8). It efficiently collects these logs, enriches them with relevant metadata like timestamps and source identifiers, and then forwards them to Loki for indexing.

Unlike conventional log aggregation tools that index the entire content of every log entry, Loki takes a more discerning approach by focusing only on indexing the metadata associated with each log, significantly enhancing search speed and reducing storage costs. The optimized indexing method proves particularly advantageous when dealing with the massive datasets commonly encountered in security analysis, as it streamlines the indexing process and empowers more efficient querying of vast log volumes. This process ultimately aids in the identification of potential security threats with improved accuracy, as detailed in Özgür Özkök's article on Grafana Loki^(Bbl. 19).

Consistent and Usable Data - Parsing and Normalization

The project utilizes a two-step approach for efficient analysis of security-related data. Initially, Nginx logs capture specific details in a standardized format, acting like a structured diary to ease future log analysis and comparison.

After log capture, Loki uses this format to categorize and organize data. It allows users to merge data streams from various sources for a comprehensive system view, aiding easy analysis and comparison across different components.

4.1.3 Data Analysis and Alerting

The system is designed to leverage existing resources while adapting to future updates. To improve the web server's security, the team will employ threat intelligence feeds, including open source vendors (External Documentation, page. 48) to get IoCs. These IoCs help the user to spot and address security threats early on. We will touch more upon IoC and its technical use in subsection 5.3.2 - Indicators of Compromise, (page. 38).

4.2 Details of the Data Collection Process

4.2.1 Go Modules

High Traffic Detection^(Bbl. 20, 21, 22)

The project implemented scripts designed to classify IP addresses as potential threats. Each classification is recorded in a result file named "*nmap_detection*," facilitating easy analysis.

The script focuses on detecting unusual network traffic patterns. It continuously monitors incoming traffic and logs instances of unusually high traffic volume, further aiding in identifying potentially suspicious IP addresses that warrant additional investigation.

The script utilizes tshark, a command-line tool for network traffic analysis. Tshark focuses solely on capturing synchronized (SYN) packets from remote machines (excluding the server's IP address to avoid self-monitoring). The script defines a customizable packet count and velocity threshold to avoid detection. Users can specify this threshold based on their system's capacity and expected traffic patterns. The script effectively identifies potential threats by continuously monitoring packet volume and dynamically adjusting based on network conditions.

JournalCTL Systemd's Log Detection^(Bbl. 22)

The JournalCTL-monitoring Go module assists in real-time monitoring and analyzing system logs in a remote server.

The first JournalCTL Go module performs a Secure Shell Protocol (SSH) in real-time, collecting records back to the Local Machine if there have been three or more failed SSH attempts^(Bbl. 24). The IP information is stored in a plain text file.

The second JournalCTL Go module connects via SSH to the server and collects the history of all attempted SSH authentications that have failed more than three times. The Go module captures detailed information about such events, including IP addresses, timestamps, number of attempts, and the severity level, and stores such information at the local machine in the *potentialThreat.yaml* files.

Standard Log Detection.

Standard Log is a Go module developed to capture and analyze potential *payloads*, such as brute-force attacks. The logs are parsed to find patterns of malicious activity. The program then generates reports based on the findings. This process allows us to process log files and detect irregularities in the patterns of log entries.

The important part of this component is its feature to run concurrently so the user can execute the parts of the program independently. One is real-time monitoring and capturing into a standard log file, "potentialthreat.txt," as intermediate storage. The Second part involves parsing through the file to verify integrity and putting it in a YAML (Yet Another Markup Language) file.

The log will be split into two files, one where all the *payloads* are stored so the user can further process the files and one for the *non-payloads*, allowing the user to process logged files that are classified as *payloads*. The data of the YAML file will be further subject to analysis for possible threat vendors. This is not to be confused with the *malicious* and *non_malicious* JSON files, which will be addressed later in this subsection.

ThreatAnalysis.go

“ThreatAnalysis” is a Go module that analyzes threats. The module operates by extracting and comparing IP addresses from the JSON (JavaScript Object Notation) files containing the potential non-threat ones (*non_threat_ips.json*) and the YAML files containing the potential threats (*HighTrafficDetection* and *payloads.yaml*).

The module then identifies and compares commonly known IP addresses from the two previously named data sources. It generates a YAML-formatted report listing the common IP Addresses. IP addresses may be recorded in external sources as potential threats or non-threats, and vice versa. The YAML file assists in visualizing such cases.

Threat API Vendor (ETI)^(ED. 9, 10, 11, 12)

To effectively assess the reputation of IP addresses, the Go module ThreatAPIVendor utilizes data from four reputable threat intelligence sources:

- AbuseIPDB API
- AlienVault Open Threat Exchange (OTX) API
- Threat Fox API
- IP Geolocation API

The functions of these APIs are further explained in Appendix B.

By combining data from these diverse sources, the IP Reputation Checker gains a comprehensive view of an IP's reputation, including the maliciousness of the IP addresses and their geolocation. The multi-source approach enhances the accuracy and reliability of the reputation assessment, allowing for better decision-making.

Data Sources and Collection

The IP Reputation Checker gathers data from two primary sources:

- **ETI APIs:** The checker integrates with various threat intelligence APIs to obtain information about known malicious IP addresses. This allows a preliminary assessment of an IP's reputation by checking it against pre-existing threat intelligence databases. The checker can identify potentially malicious IPs based on their past activity by comparing the retrieved data with the threat intelligence feed.

- **System Logs:** The checker also collects log data from system journals using journalCTL accessed via SSH. The log data is then analyzed to identify suspicious activity. Specifically, the checker focuses on login attempts and flags IPs with over three failed login attempts within a certain timeframe. These potentially compromised credentials could indicate malicious activity, and the identified IPs are added to the “potentialThreat” and stored in a file named "potentialThreat.txt."

Developing an Intelligent IP Reputation Checker

The project developed an IP Reputation Checker module written in Go. This module analyzes IP addresses against various threat intelligence sources (e.g., AbuseIPDB, AlienVault, ThreatFox) to assess their potential malicious intent. These sources were chosen for their reliability, extensive coverage, and large user communities contributing threat data.

The script leverages the Go-modules' strengths to manage API calls and data parsing efficiently. API keys are securely stored using environment variables, and data structures are defined to organize information and handle JSON responses effectively.

For optimal performance, a concurrent function calls the “goroutines” and executes queries to ETI providers simultaneously. Additionally, a serialization and file storage caching mechanism optimizes geolocation data retrieval, while mutexes ensure thread-safe access during caching.

Extracted threat data is categorized as malicious or non-malicious and stored in JSON files. These files bridge the application and an SQLite3^(ED, 8) database, where a background process populates the database tables. The system seamlessly integrates new data with existing information. Utility functions handle file operations and data persistence.

4.2.2 The Security Information and Event Management

Glitnir: A Custom SIEM Solution

Glitnir, our custom SIEM system, was built from scratch to collect, analyze, and visualize log data effectively. Glitnir operates on a separate machine to handle and analyze log data, ensuring isolation and optimal performance and security for log processing activities. To access log data, Glitnir is granted read permissions on the designated locations, internally referred to as the "Blue Box".

Streamlined Log Collection and Parsing

Glitnir collects logs from various sources, including system logs, Nginx access logs, and potentially network logs. The collection process leverages Promtail, a component of the Loki project. Promtail acts as a log shipper, efficiently scraping logs from various sources. These scraped logs are then forwarded to the Loki server for further processing.

Centralized Log Aggregation and Labeling

Once collected, logs are sent to Loki, a powerful log aggregation system inspired by Prometheus. Unlike Prometheus, which focuses on collecting and analyzing metrics, Loki specializes in logs. Loki employs a push model for log collection, where sources actively send logs to the server. Upon receiving them, Loki

parses and enriches logs with valuable metadata, such as timestamps and source identification labels. These labels allow for efficient filtering and searching of logs within the system.

Visualization and Alerting for Enhanced Monitoring

The processed logs are readily available for viewing within a dedicated Grafana dashboard. The integration allows for user-friendly visualization of log data, enabling security personnel to identify patterns and potential security events quickly. Furthermore, Loki incorporates a built-in alerting component service called "ruler". This ruler allows for continuous evaluation of log queries. Based on the results of these queries, the system can trigger actions like sending alerts through integration with Prometheus Alert Manager or Grafana's built-in alerting features. The comprehensive alerting capability empowers security teams to respond promptly to potential threats.

Detailed Infrastructure and Configuration Guide

The Github Repo provides a detailed guide for setting up Glitnir on a DigitalOcean droplet. The guide includes information about hardware specifications and network configurations required for optimal performance (<https://github.com/adaalrf/kamomille/tree/main/Dokumentasjon/Technical/SetupGuides>). Typically, Glitnir runs on an Ubuntu 23.10 x64 system with minimal resource requirements (1 vCPU, 4 GB RAM, and 25 GB disk space).

Secure Communication and Firewall Configurations

The documentation also covers essential firewall configurations. These configurations prioritize blocking unnecessary traffic while allowing communication over private IP addresses within the same Virtual Private Cloud (VPC) network.

4.3 Tools and Techniques Explained

4.3.1 Log Analysis

Centralized and Streamlined Log Storage

At the core lies Grafana Loki, a horizontally scalable log storage solution. Loki efficiently ingests, stores, and indexes logs from various sources, including Nginx access logs and system journals. Promtail then streamlines the collection process. It gathers logs from diverse sources, enriches them with valuable metadata like timestamps and source information, and then forwards them to Loki for indexing and querying.

4.3.2 Anomaly Detection

High Traffic Detection ^(Bbl. 20, 21, 22)

There are significant advantages to using a shell script that detects high traffic on an Nginx server. This script helps optimize resource allocation and improve cybersecurity defenses.

This functionality is crucial for mitigating SYN flood attacks, where attackers overload the server with SYN packets, potentially causing service denial by overwhelming server resources. If SYN packet counts drop below a defined threshold, the script halts the capture to conserve resources and prevent exploitation during such attacks.

After stopping the capture, the script transfers the captured YAML file—a human-readable data serialization format—from the remote machine to the local system, performing error checks to ensure data integrity. This method enhances the capability to monitor and manage high-traffic situations on the Nginx server effectively.

JournalCTL^(Bbl. 22)

The team will attain anomaly detection by extracting SSH login information of denied attempts, utilizing the “journalctl” command-line utility. The Procedure involves invoking “journalCTL” with specific parameters and filtering criteria to isolate SSH-related log entries. One of the filtering criteria entails identifying instances where access denial occurs repeatedly. Exceeding the threshold of three occurrences, the Go module will parse the relevant data into a YAML file.

The JournalCTL provides enhanced system security monitoring. It tracks failed login attempts with details and identifies potential security breaches. It further dynamically updates the threat severity based on the attempt frequency, assisting in prioritizing the proposed responses and effectively allocating resources. The JournalCTL provides a shutdown handling feature that prevents data loss and maintains data integrity.

The benefits are a proactive security posture, operational efficiency, and data-driven decisions. YAML file format is chosen due to its readability and ease of parsing. It describes the overall login attempts within the specified timeframe and includes timestamps for initial and final occurrences.

The script was applied to align ourselves with industry standards regarding monitoring threat detection and prevention. Anomaly detection is a big part of cybersecurity, and this script allows the team to go from reactive security to more proactive threat detection and prevention, ensuring the script runs automatically and categorizes threats and non-threats, which helps streamline the security processes and reduce the potential for human errors. Collecting data into a YAML report makes readability and potential investigation easier.

Standard Log

The standard log Go module is designed to enhance the security of web servers, attained by actively monitoring the log files for signs of brute force attacks and other unauthorized access attempts. The detection and reporting process is automated, assisting the user in quickly responding to potential threats. When continuous monitoring and detection are required, the module can run in the background and still provide a reliable service.

Potential brute force login attack detection system

Brute force attacks are a common hacking technique where attackers try to guess a user's password by systematically trying many possible combinations. Anomaly detection systems can be used to identify these attacks by looking for patterns of failed login attempts that deviate from normal behavior.

Threat Analysis

This threat analysis Go module aims to automate the threat analysis process, saving the user's resources. The analysis is conducted by identifying and comparing the information recorded with the IP addresses at different data sources. When comparing such information, the module may detect common patterns and overlaps, indicating suspiciousness when comparing such information activity. The generated YAML report file will contain information about the common IP addresses, facilitating the investigation and the possible mitigation of potential threats and security risks.

Threat API Vendors

The user may aggregate data from multiple sources by using multiple APIs^(ED. 9, 10, 11, 12). The information thereby attained enables the system to provide a more comprehensive assessment of the reputation of the respective IP Addresses, enhancing the probability of risk mitigation.

The further use of "goroutines" and "channels" permits the modules to handle multiple IP addresses efficiently, which is important when using real-time threat intelligence systems.

The module also contains Caching Mechanisms, Persistent Storage, and Error Handling functions. These functions, respectively, can:

- reduce the number of API calls, speeding up the response time,
- saves the cache and the results, providing durability and quick access to stored data,
- handles potential errors during network requests and data processing, assuring reliability and integrity.

5. System Architecture and Implementation

5.1 Description of the system setup

5.1.1 Log Collection Environment

Area	Description
Operation System	Debian 12 (latest)
Focus	Solely dedicated to log collection (minimalist approach)
Logs Captured	Requests (errors, access), including POST request bodies
Storage	External virtual drive mounted via SSHFS (Siem user) on the VPC network
Persistence	Drive stores data even when dismounted

Tbl. 2. Dedicated Debian log system: Capture details and virtual drive storage strategy

5.1.2 Mounted drive

Mounted Log Storage

An external virtual storage drive mounted onto this system allows for persistent log storage. A dedicated user account named "SIEM" utilizes the Secure Shell Protocol File System (SSHFS) to access the drive over the Virtual Private Cloud network, enabling near real-time log transfer.

Persistence

The mounted drive operates as a separate unit, ensuring log data is retained even when dismounted. The functionality safeguards the collected logs during system maintenance or potential failures.

5.1.3 Log Receiver and Analysis Environment

The receiver system (SIEM) serves as the log analysis and visualization environment. The system hosts a user interface (dashboard/Grafana) that provides the user with a convenient platform for monitoring system status, analyzing log data, and visualizing the results of those analyses.

The SIEM system accesses logs stored on the mounted virtual drive through a secure connection established via SSHFS over the Virtual Private Cloud network. This continuous log stream originates from the dedicated Nginx server responsible for log collection.

Log Processing and Visualization Pipeline

The collected logs are processed and visualized through a dedicated pipeline. Promtail acts, as mentioned in subsection 4.1.2 (page. 22), as the initial processing stage, labeling and collecting metadata on the logs. The capability allows it to recognize a wide range of logs, including our custom normalized format. Following Promtail, Loki indexes the logs using Log Query Language (LogQL). Finally, visualization and data manipulation occur within Grafana. This versatile platform empowers users to create informative visualizations and transformations based on the collected statistics, metrics, and analyzed log data. Additionally, Grafana plugins simplify the process of testing log formats and uploading data to the SIEM server.

Bridge Modules and Data Visualization

The “sqlite-web” web server plays a crucial role in data visualization within the SIEM dashboard. The server retrieves data from various modules integrated into the bridge, such as the Threat Intel module. These modules are designed with a modular architecture, enabling a "plug-and-play" integration approach. This modularity allows the bridge to support a wide range of functionalities, catering to diverse user needs. The SIEM dashboard allows users to configure these modules for optimal system performance.

Proprietary Bridge Middleware

The bridge acts as an intermediary layer within the SIEM system, enabling seamless interaction between the data acquisition layer comprised of tools like Promtail (for log collection) and Loki (for log storage), the visualization layer utilizing Grafana for data exploration, and the system core, which encompasses the "bridge" itself and its functionalities as both the front-end and back-end. Additionally, various modules and their corresponding subsystems interact with the bridge.

Key Functionalities of the Bridge Middleware

The bridge middleware serves several critical functions within the SIEM system. First, it is responsible for normalizing log data from diverse sources to ensure consistency and facilitate analysis. Furthermore, it stores normalized logs within SQLite3 databases, enabling efficient retrieval and interaction with the data by the system's web interface.

Secondly, the bridge serves an important role in promoting modularity and scalability. New features can easily be incorporated as modules within the bridge, making system expansion and resource management efficient among diverse functionalities. The modular design allows for adding features beyond the core functionalities of log collection and analysis offered by tools like Promtail and Loki in Linux Environments.

5.2 Explanation of Procedures

5.2.1 Log Collection Process

The SIEM system leverages a dedicated log collection environment running on Debian 12. The created environment captures a broad spectrum of log data, including error reports, access logs, and POST request bodies. This approach can prioritize efficient log collection without using unnecessary overhead.

As mentioned in subsection 5.1.2, (page. 29) Mounted Drive, the logs are collected and stored on a separate virtual drive mounted through SSHFS under a dedicated Virtual Private Server named “SIEM.” This configuration centralizes the log storage, making it secure and persistent. The mounted SSHFS allows near real-time log transfer over our Virtual Private Cloud network, ensuring data integrity even during disconnections or maintenance activities.

5.2.2 Log Collection Pipeline

The collected logs are fed into a defined processing pipeline. Initially, Promtail acts as a versatile log parser, recognizing and organizing a variety of log formats, including custom-designed normalized logs subsection 4.1.2 - Log receiver and analysis environment (page. 22). During this processing phase, Promtail enriches log data by adding labels and collecting relevant metadata and labels, making organization and identification easier.

Then, once Promtail’s processing is done, Loki takes over. Loki is a highly performant log storage solution that leverages its LogQL for efficient log indexing. This combination enables rapid retrieval and in-depth analysis of the collected logs within our SIEM system.

5.2.3 From Logs to Insights: Analysis and Visualization

The SIEM system has two purposes. It acts as both the analysis environment and as the host for the user interface.

Grafana transforms into a versatile platform for crafting detailed visualizations and manipulating data derived from processed logs. It acts as the final stage in the pipeline, ingesting raw data and transforming it into actionable insights. Through intuitive dashboards, Grafana allows users to monitor system health and visualize log analysis results.

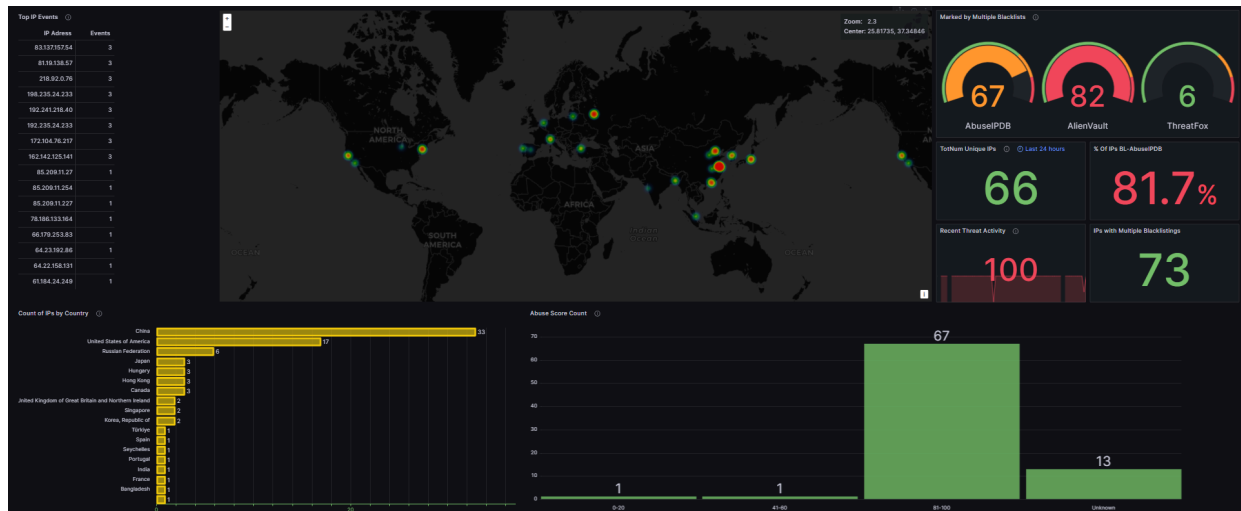


Fig. 5 Image: Grafana dashboard. Showcases Geo location, anomaly detection hits, and statistics.



Fig. 6 Image: Grafana Dashboard. Showcases resources used on the Digital Ocean SIEM

5.2.4 Bridge Middleware and Modular Architecture

At the core of the custom SIEM system lies a proprietary bridge middleware that is the foundation for robust data management. The bridge ensures data consistency by enforcing a normalized format across diverse log sources. The normalized data is subsequently stored within SQLite3 databases, making retrieval and analysis efficient.

The bridge is a central hub, facilitating seamless communication between the data acquisition layer (Promtail for collection, Loki for storage) and the visual layer (Grafana). The modular architecture makes integrating additional functionalities in the future easier without worrying about scalability.

5.3 Implementation of Anomaly Detection Features

The Anomaly Detection Process:

The detection process contains the following elements:

- Network Analysis / High Traffic Detection: Monitoring High Traffic and detecting the high thresholds.
- The JournalCTL (1) monitoring in real-time.
- The JournalCTL (2) executes fact-checking for undetected IP addresses.
- The Standard Log Go modules for enhanced security,
- Threat Analysis, for analyzing and comparing log information
- Threat API Vendors, for the use of multiple APIs for sending requests to uncover threats and non-threats from outside data sources.

5.3.1 Go Modules

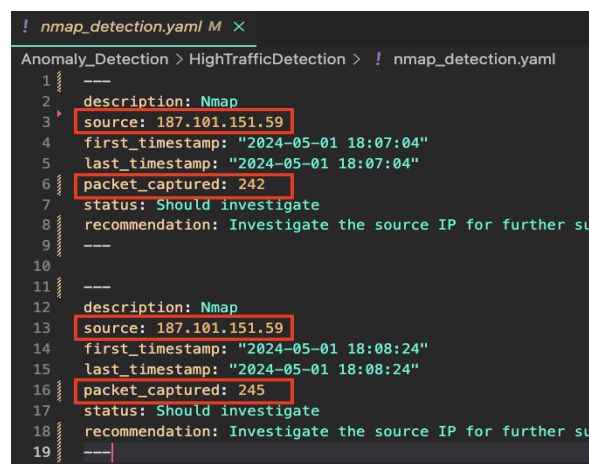
High Traffic Detection^(Bbl. 20, 21, 22)

A script is written to monitor the network traffic to accomplish the Network Analyses / High Traffic Detection. A tool named “tshark command line packet analyzer” is used. The purpose is to analyze the traffic and detect high traffic volumes. It will reveal potential Network Mapper (Nmap) scans. When high traffic is encountered, the following process takes place:

- Unusual, high-threshold SYN packages are detected.
- Source (IP addresses), first and last timestamps, amount of packet count, the status of packages, and recommendations are logged and saved in a designated YAML file named “HighTrafficDetection”.

Unique IP addresses that may be potential threats are identified and stored in the ‘PotentialThreat.txt’ file for further analysis. The process is achieved through a remote SSH connection to the server. The connection allows the script to define variables and execute tshark commands (Figure below).

To ensure the script doesn't consider its own traffic a potential threat, it excludes its server IP address from the results analysis.



```
! nmap_detection.yaml M X
Anomaly_Detection > HighTrafficDetection > ! nmap_detection.yaml
1  ---
2  description: Nmap
3  source: 187.101.151.59
4  first_timestamp: "2024-05-01 18:07:04"
5  last_timestamp: "2024-05-01 18:07:04"
6  packet_captured: 242
7  status: Should investigate
8  recommendation: Investigate the source IP for further su
9  ---
10
11 ---
12 description: Nmap
13 source: 187.101.151.59
14 first_timestamp: "2024-05-01 18:08:24"
15 last_timestamp: "2024-05-01 18:08:24"
16 packet_captured: 245
17 status: Should investigate
18 recommendation: Investigate the source IP for further su
19 ---
```

Fig. 7: Network traffic detection and analysis derived from tshark.

JournalCTL ^(Bbl. 22)

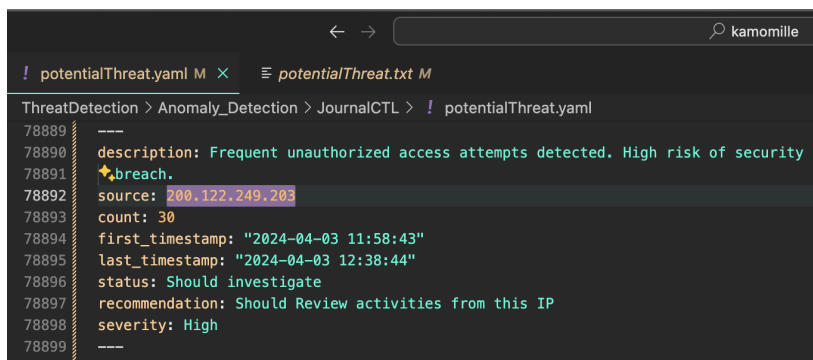
The JournalCTL Go module utilizes SSH to execute the journalCTL command remotely, and it will record detailed timestamps. The execution of the command is within a Bash shell. It facilitates using sshpass for password inputs and sudo for elevated permissions.

The JournalCTL handles Log Parsing and Data Aggregation. The output is continuously parsed in real-time, and a regular expression identifies the lines that indicate failed password attempts. This results in the extraction of timestamps and IP Addresses. Each IP address is stored as a key in a map with a corresponding IPRecord.

The application checks for new logs at regular intervals, set to make updates every 5 seconds. When the application is interrupted, a cleanup sequence is initiated, and the data is stored in the YAML file.

```
Feb 02 11:47:20 debian-s-1vcpu-1gb-fra1-01 sshd[426221]: pam_unix(sshd:auth): authentication failure: logname= uid=0 euid=0 tty=ssh ru
Feb 02 11:47:22 debian-s-1vcpu-1gb-fra1-01 sshd[426221]: Failed password for root from 200.122.249.203 port 41232 ssh2
Feb 02 11:47:23 debian-s-1vcpu-1gb-fra1-01 sshd[426223]: Unable to negotiate with 103.98.107.62 port 34751: no matching host key type
Feb 02 11:47:24 debian-s-1vcpu-1gb-fra1-01 sshd[426221]: Received disconnect from 200.122.249.203 port 41232:11: Bye Bye [preauth]
Feb 02 11:47:24 debian-s-1vcpu-1gb-fra1-01 sshd[426221]: Disconnected from authenticating user root 200.122.249.203 port 41232 [preaut
Feb 02 11:48:47 debian-s-1vcpu-1gb-fra1-01 sshd[426228]: pam_unix(sshd:auth): authentication failure: logname= uid=0 euid=0 tty=ssh ru
Feb 02 11:48:49 debian-s-1vcpu-1gb-fra1-01 sshd[426228]: Failed password for root from 200.122.249.203 port 52360 ssh2
Feb 02 11:48:51 debian-s-1vcpu-1gb-fra1-01 sshd[426228]: Received disconnect from 200.122.249.203 port 52360:11: Bye Bye [preauth]
Feb 02 11:48:51 debian-s-1vcpu-1gb-fra1-01 sshd[426228]: Disconnected from authenticating user root 200.122.249.203 port 52360 [preaut
Feb 02 11:50:15 debian-s-1vcpu-1gb-fra1-01 sshd[426230]: pam_unix(sshd:auth): authentication failure: logname= uid=0 euid=0 tty=ssh ru
Feb 02 11:50:16 debian-s-1vcpu-1gb-fra1-01 sshd[426230]: Failed password for root from 200.122.249.203 port 35255 ssh2
Feb 02 11:50:17 debian-s-1vcpu-1gb-fra1-01 sshd[426230]: Received disconnect from 200.122.249.203 port 35255:11: Bye Bye [preauth]
```

Fig. 8: JournalCTL depicting failed SSH attempt from a specific sample IP address.



```

ThreatDetection > Anomaly_Detection > JournalCTL > ! potentialThreat.yaml
78889 ---
78890 description: Frequent unauthorized access attempts detected. High risk of security
78891 ⚠breach.
78892 source: 200.122.249.203
78893 count: 30
78894 first_timestamp: "2024-04-03 11:58:43"
78895 last_timestamp: "2024-04-03 12:38:44"
78896 status: Should investigate
78897 recommendation: Should Review activities from this IP
78898 severity: High
78899 ---

```

Fig. 9: Results from the JournalCTL from the same IP address.

Standard Log

The standard log Go modules are structured with LogEntry, AttackDetail, and SecurityReport. The analysis attained by this structure is as follows:

- The LogEntry creates a single log entry from the web server. The fields in the log entry include the inquirer's IP address, the time of the inquiry, and other details of the HTTP.
- The AttackDetail reveals information about any detected security threats and the description.
- The SecurityReport collects and presents all detected attack details. These details include the timestamp.

The Go module further provides several functions for parsing log files, identifying attack patterns based on extensive failed login attempts, and providing a detailed report of the detected threats in YAML format.

```
request_body":""}
{"time_local":"05/May/2024:09:32:10 +0000","remote_addr":"57.129.23.166","remote_user":"","request":"GET /.env HTTP/1.1","status":
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36","request_body":""}
ENV GET /.env HTTP/1.1
{"time_local":"05/May/2024:09:32:10 +0000","remote_addr":"57.129.23.166","remote_user":"","request":"POST / HTTP/1.1","status":
la/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36","request_body":"0x%5B%5D=an
{"time_local":"05/May/2024:09:37:32 +0000","remote_addr":"152.32.157.92","remote_user":"","request":"GET / HTTP/1.1","status": "2
lla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.0",
{"time_local":"05/May/2024:09:37:50 +0000","remote_addr":"152.32.157.92","remote_user":"","request":"GET / HTTP/1.1","status": "2
la/5.0 (Macintosh; Intel Mac OS X 10_7_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11","request_body"
{"time_local":"05/May/2024:10:06:12 +0000","remote_addr":"57.129.23.166","remote_user":"","request":"GET /.env HTTP/1.1","status"
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36","request_body":""}
ENV GET /.env HTTP/1.1
```

Fig. 10: Standardlog.log original file structure.

```
! payloads.yaml M X
ThreatDetection > Anomaly_Detection > Standard_log > ! payloads.yaml
---
865 description: Brute force login detected
866 source: 57.129.23.166
867 count: 533
868 severity: OH MY GOD, GOD DAMN, I THINK WE SHOULD BLOCK THIS GUY!
869 threshold: 533 failed attempts, 1 months, 12 days, 23 hours, 46 minutes, 8 seconds
870 first_timestamp: "2024-04-04 13:21:02"
871 last_timestamp: "2024-05-17 13:07:10"
872 status: Should Investigate
873 recommendation: Investigate the source IP for further suspicious activities. Consider
874 permanent account lockout or password reset protocols if necessary.
875 request_path: POST / HTTP/1.1
876 user_agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
877 Chrome/81.0.4044.129 Safari/537.36
878 payload: 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
879 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
880 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
881 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
882 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
883 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
884 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
885 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
886 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
887 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
888 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
889 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st ||| 0x%5B%5D=androXgh0st |||
```

Fig. 11: Payload detection from the specific IP address saved in a YAML file for easy reading.

Log Analysis and Brute Force Detection Script linking up to Standard Log

The script's core functions begin with parsing log files into two segregated *payload.yaml* and *non-payload.yaml* files, precisely extracting the client's IP address, timestamp, and HTTP request path (i.e., the attempted login endpoint) from each log entry. Using regular expressions ensures reliable identification of the crucial data. Next, the script identifies brute force attempts by monitoring repeated failed logins originating from the same IP within short timeframes. It classifies the severity of attacks ('Low', 'Middle', 'High', 'Extreme') based on thresholds related to the attempt count and duration.

Threat Analysis

The threat analysis Go module functions “*extractIPsFromjson*” and “*extractIPsFromyaml*” enable the user to extract the required data from the JSON and YAML files. The modules can thereby find common IP addresses from different data sources. The function “*intersect*” enables this process.

There is also a function, “*outputRecord*,” that lists the sets of common IP addresses in YAML format. The list contains details of description, source, timestamp, status, severity, and file evaluation.

(See figure below). To generate updates of this file, the user may use a ticker named: “*time.NewTicker*”, enabling the renewal of the analysis process at regular intervals.

```

() non_threat_ips.json +D, M X
ThreatDetection > ThreatAPIVendor > {} non_threat_ips.json > {} 177 >
1240 },
1241 {
1242   "ip": "57.129.23.166",
1243   "abuseipdb_potentialThreat": false,
1244   "alien_vault_potentialThreat": false,
1245   "threat_fox_potentialThreat": false,
1246   "timestamp": "2024-05-06 20:24:05"
1247 },
1248

! payloads.yaml M X
ThreatDetection > Anomaly_Detection > Standard_log > ! payloads.yaml
2259 ---
2260 description: Brute force login detected
2261 source: 57.129.23.166
2262 count: 533
2263 severity: OH MY GOD, GOD DAMN, I THINK WE SHOULD BLOCK THIS GUY!
2264 threshold: 533 failed attempts, 1 months, 12 days, 23 hours, 46
2265 first_timestamp: "2024-04-04 13:21:02"
2266

! matches_as_threat.yaml +R, M X
ThreatDetection > Anomaly_Detection > ThreatAnalysis_non-threat > ! matches_as_threat.yaml
19 ---
20 description: 'Common IP between JSON and payloads.yaml: 57.129.23.166'
21 source: 57.129.23.166
22 timestamp: "2024-05-18T18:35:58+02:00"
23 status: Review Needed
24 severity: High
25 file: non_threat_ips.json, payloads.yaml
26
27
  
```

Fig. 12: The Threat API Vendor file shows a non-threat that is detected in the standard log using payload

Threat API Vendor^(ED. 9, 10, 11, 12)

The threat API vendor Go module sends API requests to each ETI database service (AbuseIPDB, ThreatFox, and AlienVault) to detect whether there is a potential threat or a non-threat in our IP address file. We will receive data as a boolean indicating whether the IP address exists as a threat in their database. An additional API also detects Geolocation from the specific IP address.

The script utilizes several key components for interacting with multiple APIs:

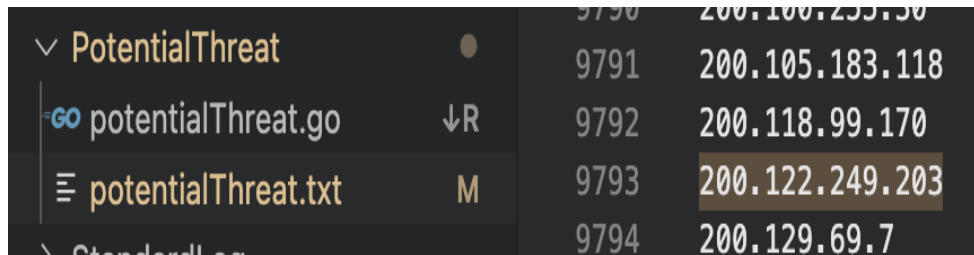
Key Component	Description
Data Structures:	IPReputation and GeoLocation store results retrieved from the APIs.
Communication:	HTTP requests are sent to specific API endpoints. Each response is parsed and stored in the corresponding data structures.
Caching:	The ipCache function caches IP reputation and geolocation results for efficiency.

Tbl. 3: Key Components

These functions work together in a defined sequence:

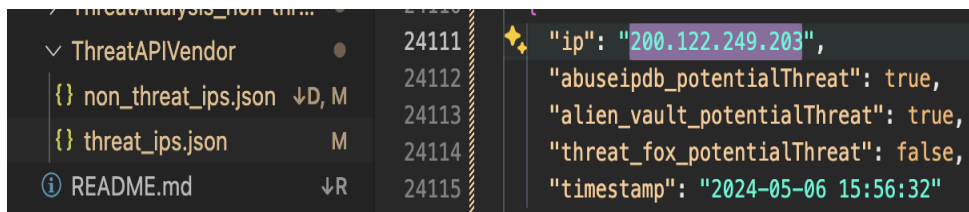
Key Component	Description
Data Structures:	Initialization: Loads any cached data.
Monitoring:	Continuously reads a list of potential threats.
API Calls:	Simultaneously fetches reputation data from all three APIs.
Result Processing:	Merges, updates, and sorts results into malicious and non-malicious categories. Stores the results on disk
Geolocation Updates:	Continuously updates geolocation data for retrieved IPs.

Tbl. 4: Functions



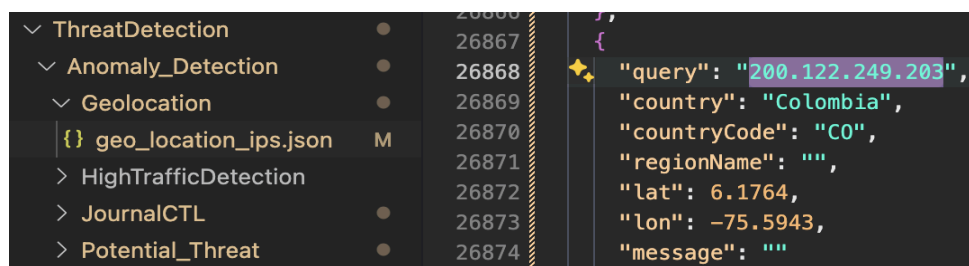
File Name	IP Address
potentialThreat.go	9790 200.100.255.50
potentialThreat.txt	9791 200.105.183.118
potentialThreat.txt	9792 200.118.99.170
potentialThreat.txt	9793 200.122.249.203
potentialThreat.txt	9794 200.129.69.7

Fig. 13: Potential Threats list of unique IP addresses from our database.



File Name	IP Address	Threat Info
ThreatAPIVendor	24111	"ip": "200.122.249.203",
non_threat_ips.json	24112	"abuseipdb_potentialThreat": true,
threat_ips.json	24113	"alien_vault_potentialThreat": true,
README.md	24114	"threat_fox_potentialThreat": false,
README.md	24115	"timestamp": "2024-05-06 15:56:32"

Fig. 14: Threat Vendor Intelligence evaluated threat info from (ETI Service).



File Name	IP Address	Geolocation Info
ThreatDetection	26866	"query": "200.122.249.203",
Anomaly_Detection	26867	"country": "Colombia",
Geolocation	26868	"countryCode": "CO",
geo_location_ips.json	26869	"regionName": "",
HighTrafficDetection	26870	"lat": 6.1764,
JournalCTL	26871	"lon": -75.5943,
Potential_Threat	26872	"message": ""

Fig. 15: Geolocation info from (IP Geolocation API).

5.3.2 Indicators of Compromise

Glitnir's approach to managing IoCs leverages real-time log analysis with custom scripting to identify unusual activities that could indicate potential breaches. It analyzes various log data, such as access error logs and system event logs, to identify suspicious patterns. By integrating with open-source, crowd-sourced threat intelligence feeds constantly updated by a network of security researchers, Glitnir gains valuable context about known malicious entities. These feeds, often built on tactics like global honeynets (large networks of honeypots that act as 'bait' for threat actors), empower Glitnir to enrich its analysis, improve detection accuracy, and reduce false positives.

In addition to the threat intelligence feeds, Glitnir leverages custom-developed scripts as detailed in subsection 5.3.1 on Go Modules, (page. 33) to actively monitor system activity for abnormalities that might signal the use of Tactics, Techniques, and Procedures (TTPs) documented within the MITRE ATT&CK framework. These TTPs could be recently discovered zero-day exploits or established attack methods employed by malicious actors. Glitnir's scripts play a critical role in identifying and logging IoCs by continuously monitoring for these indicators of potentially malicious behavior. These IoCs are then categorized based on their severity and the potential threat they pose, enabling security teams to prioritize and investigate potential security incidents.

While the use of Glitnir's IoC management is evaluated through a controlled testing environment, Glitnir has shown a significant reduction in time spent in compromise detection, focusing on metrics such as detection speed, accuracy, and system response time. While these results are promising, it is important to acknowledge that controlled environments may not capture the full spectrum of real-world threats. Future investigations could involve testing Glitnir in a more dynamic setting to assess its effectiveness further.

6. Results and Discussion

6.1 Key Findings Summary

6.1.1 Accomplishments

Key findings emerged throughout the development and testing phase. Among the several surfaced findings, one was discovered during the development of the threat vendor script. The threat vendor script successfully identified malicious IP addresses in a Linux-based Nginx web server, serving as proof of concept for anomaly detection. This approach considerably improved security, accurately flagging potential threat actors. It highlighted the aim of the script by providing a “*True* or *False*” boolean statement of IP addresses as malicious or not, emphasizing the importance and role of the Minimum Viable Product.

The standard log Go module, discussed in 4.3.2 Anomaly Detection (Page 29), offered numerous benefits for log monitoring. It automated the tedious task of log file analysis, freeing up additional resources. It also provided early detection of potential security breaches and generated automated reports, enabling users to take appropriate actions and select the best response procedures

Glitnir has displayed high accuracy in its IoC detections, effectively minimizing false positives, which was a high-priority goal mentioned in section 1.1, (page. 7). The enhancement of security in web servers and the importance of log analysis. Glitnir is developed in a modular fashion, allowing it to easily receive updates and expansions, making it a tool that can be used and developed further in the future. As a security tool, it was important to accommodate new threat intelligence sources and detection scripts, which are vital for maintaining relevance against evolving cyber threats.

6.1.2 Functionalities

The key functionalities of the product are listed below.

1. Real-time Log Analysis:
Continuous monitoring and processing of log data from web servers to detect and respond to security threats in real-time.
2. High Traffic Detection:
Monitors network traffic levels to identify potential Distributed Denial of Service (DDoS) attacks or other abnormal traffic patterns. The system uses predefined thresholds to trigger alerts when traffic exceeds normal operating levels.
3. Deny List Firewall Rule High Traffic Detection:
Glitnir automatically updates firewall rules based on detected threats. When a malicious IP is identified, the system adds it to a deny list, effectively blocking any further attempts from that source to access the network and preventing potential breaches.

4. **SSH Authentication Monitoring:**
Glitnir tracks SSH login attempts to find and stop unauthorized access. It looks at failed and successful logins, flags suspicious behavior like repeated failures, and alerts supervisors about potential brute-force attacks.
5. **Standard Log Monitoring:**
This feature collects and analyzes various log types, such as access, error, and system event logs, and ensures ongoing visibility into server operations and security events.
6. **Threat Analysis:**
Glitnir's threat analysis automatically utilizes predefined rules to examine log data for signs of compromise and respond to security threats.
7. **Threat API Integration:**
Glitnir connects with external threat intelligence platforms via APIs to stay updated with the latest threat data.
8. **Visualization and Alerting:**
Generates various graphics and graphs based on a predefined ruleset, giving stakeholders and supervisors easy access to important information through a user-friendly dashboard.
9. **Bridge:**
The bridge, "Bivrost", in Glitnir standardizes log data, ensuring seamless integration and communication and enhancing modularity and scalability.

6.1.3 Implications

As part of the custom-made SIEM, the anomaly detection system enabled us to detect potential threats, listing IP addresses for testing through open-source APIs that revealed Geolocation, Threats, and Non-Threats.

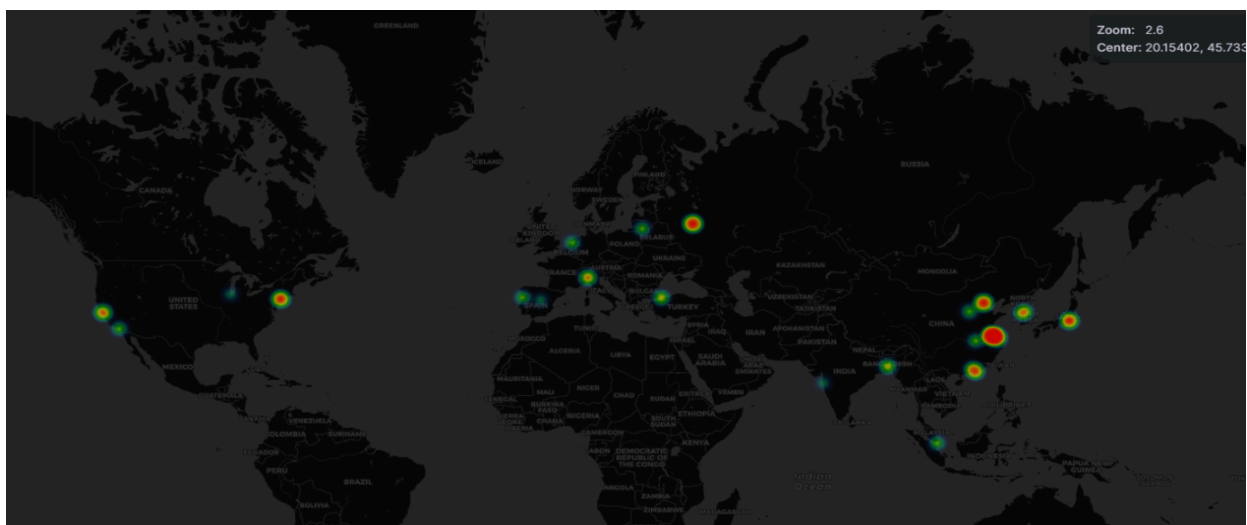


Fig. 16 Image: Grafana dashboard. Showcases Geolocation

The web browser Standard Log revealed Payloads and Non-Payloads, indicating potential threats. The tshark command line enabled monitoring and analysis of High Network trafficking, mitigating SYN flood attacks. In addition, an extension of the system includes a script that sets up a firewall rule, denying further system access by the attacker.

Overall, the team delivered a comprehensive Cybersecurity system to the Product Owner, Kristiania University College. Capable of defending against potential threats through early detection and Anomaly Detection through Log Analysis, the team has provided the Product Owner with an enhanced SIEM system that is fully modular and can be extended, future-proofing the final product.

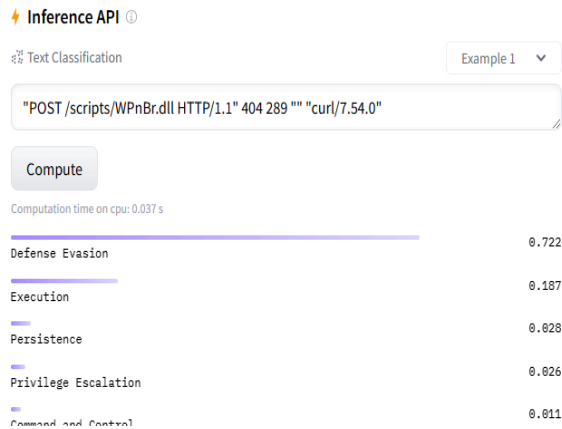
6.1.4 Limitations and Issues

The analysis of specific log data worked well, and the results met our criteria. However, we had issues with the number of the API daily rate limit permitted. The standard version is limited to 1000 IP requests per day. Above this limit, we would receive “HTTP 429 Too many requests,” resulting in all excess requests being “false”. Other versions would permit a larger amount of API requests, yet that is chargeable and outside our budget. Further, the Geolocation API requests are limited to 45 per minute. The team, however, registered several user accounts on the websites, facilitating a few more API keys. If more than 250 requests every 5 minutes were made, the API would block our IP as the ETI database service (AbuseIPDB, ThreatFox, and AlienVault) would consider the action for web scraping using bots. The issues described were solved by setting specific limits and having not more than five API user keys.

The team had coding issues, such as only part of some IP addresses appearing during High-Traffic detection. Fortunately, the team's cooperation secured the required coding assistance, solving this and other problems.

Although “Machine Learning”, as described in Chapter 2, Doc. 2 (page 10), was employed during the testing phase; these techniques did not become part of our project due to limitations in anomaly detection executions. The models perform well in classifying attacks and anomalous logs in terms of speed. However, the number of alerts generated implies that more data sources are needed to lower the false positives, as the models cannot verify the success of the attacks.

We explored the use of the MITRE BERT Base Cased model to enhance the analysis and classification of security logs. Despite its advanced capabilities in natural language processing, the model produced too many classifications, leading to an overwhelming number of alerts. This high volume of alerts increased false positives, making it difficult to accurately identify genuine threats, leading to consequential counterproductivity.



Based on these findings, we've come to believe that analyzing a log by itself gives little to no value, as there is no context surrounding the log. In terms of a correctly classified SQL injection attempt, there is no way to determine if it was successful without further context. For instance, while the log might indicate that an SQL injection attempt was made, without additional data such as database responses or subsequent user actions, ascertaining outcomes of the attack becomes challenging.

Fig 17. Interference API

To address this limitation, integrating multiple data sources is necessary. By combining logs from various parts of the system, such as database access logs, network traffic logs, and application server logs, we can create a more comprehensive picture of the events. This would allow for cross-verification, where a detected anomaly in one log source can be validated through information from another, reducing the likelihood of false positives.

In reference to Fig 17, The Inference API is being used to classify the log entry, and based on the provided data and computed scores, it suggests that the request is highly likely to be part of a discovery phase of an attack, potentially linked to a known exploit pattern (Log4Shell). This classification helps in identifying and prioritizing potential security incidents for further investigation.

6.1.5 Project Execution

The project was executed expeditiously, with sound and effective strategies, on time, and without much controversy. The team made the bold decision to expand the project from the initial task, making it more comprehensive by building a custom SIEM from scratch. The document stands as a testament to this accomplishment.

6.1.6 General Data Protection Regulation (GDPR)

The team took care to make the collected data anonymous, thereby not infringing on any GDPR, and avoided registration in the SIKT registry. All collected data was from open sources and was already listed by other APIs.

6.1.7 Risk Management

The team devised a successful Risk Management Structure to handle and mitigate risks that could occur and did occur during the project. The risk exposures in the form of delays due to sickness, technical issues, and coding problems to make the systems work flawlessly were mitigated and handled successfully. During the process, all team members got sick at various times, making some sprints longer than planned, yet the team managed to catch up and fulfill the respective assignments with some delay.

As soon as one team member was out of action due to flu or stress, the tasks got delegated to reduce further delays, and the team stepped in cooperatively and complemented the tasks by helping out. The cooperative spirit of the team staved off the need for redelegation of the assigned tasks.

6.1.8 The coding to attain successful Anomaly Detection

After four months, the team members could finally work out the final adjustments to make it all work as intended and deliver the desired outcome to the Project Owner, appropriately visualized with design features documented, which adhere to the Project owner's requirements and expectations.

7. Conclusion and Recommendations

7.1 Conclusions

The findings of this thesis have successfully demonstrated the design, development, and implementation of a custom SIEM system named Glitnir. The system has been crafted to enhance anomaly detection capabilities within web server environments, showing high integration with real-time threat intelligence. Integrating these feeds has improved the detection accuracy of potential anomalies, making Glitnir a successful tool in anomaly detection and against cyber attacks.

One of the most notable achievements of this thesis is the demonstration of Glitnir's proactive threat detection capabilities. Through custom software and scripts written in Go and Bash, Glitnir can effectively handle high network traffic and detect unauthorized IP addresses, thereby identifying and mitigating potential attacks such as DDoS and brute force attacks. The proactive approach upholds the integrity and security of web server environments. The architecture of Glitnir is designed to be both modular and scalable, ensuring its ability to integrate new functionalities and handle increasing data volumes.

Another crucial aspect of this project has been the effective management and visualization of log data. Tools like Loki and Grafana enable efficient and actionable insights from log data.

We can conclude that the team has successfully enhanced anomaly detection techniques by applying specially crafted parsing solutions to the system and service logs of a web server in a Linux environment. This, in turn, improved the identification and mitigation of cybersecurity threats, all while building and utilizing a functional SIEM solution to better execute this task.

7.2 Recommendations

In light of the findings demonstrated by the Glitnir system, several recommendations can be proposed to enhance its functionality in anomaly detection. First, there is a significant opportunity for continuous improvement in threat intelligence integration. Future work should focus on expanding the integration of threat intelligence sources to include more diverse and potentially underrepresented threat vectors. The system can ensure broader coverage and access to up-to-date information by incorporating newer databases and real-time threat feeds.

The increasing emphasis on data privacy and protection regulations globally warrants enhancing the system's capabilities to handle log data in a privacy-compliant manner. Techniques such as data anonymization could be further explored and implemented to ensure that Glitnir respects privacy laws and regulations, enhancing its use in different regulatory environments.

Regular audits and system updates should be conducted to maintain Glitnir's effectiveness and potential. These audits are necessary to adapt the system to new cybersecurity threats and comply with emerging security standards and regulations.

Additionally, incorporating machine learning (ML) models to analyze patterns and predict potential threats based on historical data significantly advances the system's predictive capabilities. Added context such as metrics from the database at the time of the log containing the potential attack. A read of equal amount of rows in the database as the performed query in the detected log would greatly increase the likelihood of an actual successful attack where an alert is warranted.

Research into methods suitable for log data analysis and anomaly detection enhanced by context, should be prioritized to enhance Glitnir's analytical power. Such advancements could lead to earlier detection of potential threats, substantially reducing the risk and impact of cyber attacks. In the initial exploration phase, ML was contemplated in identifying unusual traffic patterns. While this might have been a fruitful endeavor, resources were deemed needed elsewhere. Nevertheless, ML may successfully be employed in anomaly detection, and further exploration is warranted. There may be promising results using relevant algorithms like One-Class SVM (Support Vector Machine).

Engaging with the cybersecurity community and industry to share knowledge, challenges, and solutions can foster improvements and innovations in SIEM technologies and practices. This collaboration can lead to shared knowledge that benefits the wider community and can also help refine Glitnir based on real-world feedback and evolving security challenges.

Bibliography (Bbl.)

1. IBM. "Data Breach Report." Updated 2024. Accessed February 19, 2024.
<https://www.ibm.com/topics/data-breach>
2. Dyrek, Gerri."Radware 2024 Report: Malicious Web Application and API Transactions Rise 171% Driven by Layer 7 Web DDoS Attacks." Yahoo Finance, 2024. Accessed February 19, 2024.
<https://finance.yahoo.com/news/radware-2024-report-malicious-application-110000416.html>
3. Cheng-Yuan, Ho, Y.R. Lin, Yuan-Cheng Lai, I-Wei Chen, Fu-Yu Wang, and Wei-Hsuan Tai. "False Positives and Negatives from Real Traffic with Intrusion Detection/Prevention Systems." ResearchGate, August 2012. Accessed February 19, 2024,
<https://people.cs.nycu.edu.tw/~tommyho/publication/FPN.pdf>
4. The Institute of Electrical and Electronics Engineers, Inc.(IEEE), The Open Group, "IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7.", Published January 31 2018, The Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue New York, NY 10016-5997 USA, ISBN 978-1-5044-4542-9, Accessed May 18, 2024.
<https://ieeexplore-ieee-org.egms.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=8277153>
5. Nengwen Zhao, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Xidao Wen, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. "An Empirical Investigation of Practical Log Anomaly Detection for Online Service Systems". In Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece. ACM, New York, NY, USA, Accessed February 10, 2024,
<https://netman.aiops.org/wp-content/uploads/2021/09/logstudy.pdf>
(ACM Reference format)
6. Mandagondi, L., G., "Anomaly Detection in Log Files Using Machine Learning Techniques", Faculty of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, February 2021, Accessed February 11, 2024,
<http://www.diva-portal.org/smash/get/diva2:1534187/FULLTEXT02.pdf>
7. Ma, Rongjun, "Anomaly detection for Linux System log", University of Twente, August 2020., Accessed February 11, 2024,
http://essay.utwente.nl/83142/1/Ma_MA_FacultyOfEEMathematicsAndCS.pdf
8. Petai, S., "Detecting Anomalies in System Logs", Tallinn University of Technology, Faculty of Information Technology, Department of Computer Science, Chair of Network Software, 2014, Accessed February 10, 2024,
<https://digikogu.taltech.ee/en/Item/ae0bfe34-e3e3-4756-b17a-22b0b4112843>

9. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. "Experience Report: System Log Analysis for Anomaly Detection". Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China, IEEE 27th. International Symposium on Software Reliability Engineering, 2016, Accessed January 30, 2024, https://jiemingzhu.github.io/pub/slhe_issre2016.pdf (ACM Reference format)
10. Simplilearn. "Agile vs Scrum: Know the Main Differences and Similarities". Last modified March 18, 2024, Accessed April 27, 2024 <https://www.simplilearn.com/agile-vs-scrum-article>
11. West, Dave. "Sprint Planning". Published April 1st, 2019. Atlassian.com. View-source, Accessed April 27, 2024. <https://www.atlassian.com/agile/scrum/sprint-planning>
12. LeanWisdom. "Choosing the Right Sprint Length: A practical Guide in Agile Methodology". Updated January 18, 2024. Accessed April 27, 2024. <https://www.leanwisdom.com/blog/choosing-the-right-sprint-length-a-practical-guide-in-agile-methodology>
13. Brown, Lucy. "What is Project Resource Management?" Last modified January 11, 2024. Invensis Learning Blog. Accessed April 27, 2024. <https://www.invensislearning.com/blog/what-is-project-resource-management/>
14. Ellingwood, Justin. "Understanding IP Addresses, Subnets, and CIDR Notation for Networking." Updated on December 15, 2021. DigitalOcean Community Tutorials. Accessed March, 14, 2024, <https://www.digitalocean.com/community/tutorials/understanding-ip-addresses-subnets-and-cidr-notation-for-networking>
15. Aleroud, Ahmed, Zhiyuan Chen, and George Karabatis. "Network Trace Anonymization Using a Prefix-Preserving Condensation-Based Technique (Short paper)." Department of Computer Information Systems, Yarmouk University, Irbid 21163, Jordan, and Department of Information Systems, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA. October 2016, Accessed March 15, 2024. Available from: https://www.researchgate.net/publication/309218082_Network_Trace_Anonymization_Using_a_Prefix-Preserving_Condensation-Based_Technique_Short_paper
16. Anant, V., Lisa Donchak, James Kaplan, and Henning Soller, "The Consumer Data Opportunity and the Privacy Imperative." McKinsey & Company, April 27, 2020 Accessed March 15, 2024, https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-consumer-data-opportunity-and-the-privacy-imperative#
17. Sharif, Arfan. "What is Log Aggregation: A Guide to Observability in Cybersecurity." Published on December 21, 2022. CrowdStrike Cybersecurity 101. Accessed March 15, 2024. <https://www.crowdstrike.com/cybersecurity-101/observability/log-aggregation/>

18. Ozkok, Ozgur. "Grafana Loki: OSS Log Aggregation System." October 21, 2023, Accessed March 15, 2024, <https://ozgurozkok.com/grafana-loki-oss-log-aggregation-system>
19. Cloudflare. "What is a SYN flood attack?". Published and updated on multiple unspecified dates. Accessed April 11, 2024, <https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/>
20. Medium: Written by Digest Academy "What is a SYN Flood Attack (DDoS)? How to detect them?" Published October 12, 2019. Accessed April 11, 2024, <https://medium.com/@digestacademy/what-is-a-syn-flood-attack-ddos-how-to-detect-them-515ad4d7e1ee>
21. Github: username Rmfatemi. Name of the author Fatemi, Rasool "synflood-attack-detection" Updated April 5, 2020, Accessed April 15, 2024, <https://github.com/rmfatemi/synflood-attack-detection>
22. Ellingwood, Justin "How To Use Journalctl to View and Manipulate Systemd Logs" Updated July 9, 2021, Accessed March 4, 2024, <https://www.digitalocean.com/community/tutorials/how-to-use-journalctl-to-view-and-manipulate-systemd-logs>
23. CloudFlare. "What is SSH? | Secure Shell (SSH) protocol" Updated January 4, 2024. Accessed March 4, 2024, <https://www.cloudflare.com/learning/access-management/what-is-ssh/>

External Documentation (ED.)

1. Miro:
<https://miro.com/>
2. Nginx web:
<https://nginx.org/en/docs/>
3. Nginx. "ngx_http_upstream_module - Nginx Documentation." Accessed March 22, 2024.
https://nginx.org/en/docs/http/ngx_http_upstream_module.html.
4. The Heartbleed Bug, Open Source, Accessed February 14th. 2024.
<https://heartbleed.com/>
5. Grafana. Open Source, Accessed April 15, 2024. <https://grafana.com/> .
6. Grafana. "Promtail - Grafana Documentation." Open Source, Accessed April 15, 2024.
<https://grafana.com/> / .
7. Grafana. "Loki Documentation.", Open Source, Accessed April 15, 2024
<https://grafana.com/docs/loki/latest/> .
8. Grafana Labs. "Promtail - Grafana Documentation." Open Source, Accessed April 15, 2024
<https://grafana.com/docs/loki/latest/send-data/promtail/>.
9. Sqlite3, Open Source, Accessed February 5, 2024
<https://www.sqlite.org/docs.html>
10. AbuseIPDB "AbuseIPDB API", Open Source, Accessed February 14th. 2024.
<https://www.abuseipdb.com/>
11. Alienvault "Alienvault API", Open Source, Accessed February 14th. 2024.
<https://otx.alienvault.com/>
12. ThreatFox "ThreatFox API", Open Source, Accessed February 14th. 2024.
<https://threatfox.abuse.ch/>
13. IP Geolocation "IP Geolocation API", Open Source, Accessed February 20th. 2024.
<https://ip-api.com/>
14. tshark "network protocol analyzer", Open Source, Accessed April 15, 2024.
<https://www.wireshark.org/docs/man-pages/tshark.html>

Abbreviations:

1. SYN (Synchronize)
2. Nmap (Network Mapper)
3. SSH (Secure Shell Protocol)
4. SIEM (Security Information and Event Management)
5. SSL (Secure Sockets Layer)
6. DDoS (Distributed Denial of Service)
7. VM (Virtual Machine)
8. PCA (Principal Component Analysis)
9. HTTP (Hypertext Transfer Protocol)
10. API (Application Programming Interface)
11. GDPR (General Data Protection Regulation)
12. IoC (Indicators of Compromise)
13. VPC (Virtual Private Cloud)
14. TTPs (Tactics, Techniques, and Procedures)
15. TFIDF (Term Frequency Inverse Document Frequency)
16. BI-LSTM (Bidirectional Long Short-Term Memory)
17. IoT (Internet of things)
18. MitM (Man-in-the-Middle)
19. CIDR (Classless Inter-Domain Routing)
20. Go (The Go Programming Language)
21. txt (Text File Format)
22. IP (Internet Protocol)
23. SSHFS (Secure Shell Protocol File System)
24. LogQL (Log Query Language)
25. VPS (Virtual Private Server)
26. YAML (Yet Another Markup Language)
27. JSON (JavaScript Object Notation)
28. POSIX (The Portable Operating System Interface)
29. GNU (GNU's Not UNIX)
30. ETI (External Threat Intelligence)
31. ML (Machine Learning)

Appendix:

Appendix A

Risk Plan

Risk	Consequence	Likelihood	Risk point	preventive measures	Measures if the problem occurs
Short term Sickness	5,00	0,40	2,00	Take care of health	Inform immediately when sickness develop.
Long term sickness	8,00	0,15	1,2	Maintain a continuous status report on work in progress. Implementation of secondary taker to be able to easily transfer any work tasks to another person.	Discuss scope. Is it still achievable? Redistribute work assignments or reduce the amount of work/tasks
Delays	6,00	0,25	1,50	Make sure all plans and time estimates are reviewed in advance	Choose what to prioritize and do these tasks first
Bad/poor group dynamic	7,50	0,10	0,75	Respect your colleague and make sure you show up in a good mood. Leave personal situations at the door	Have a meeting where it is discussed what can make social and work-related things better.
Loss of Data	8,00	0,40	3,20	Make sure all data is available and backed up in the cloud. The group will use github for this purpose.	A meeting will be held if the scope of the data loss is large enough that everyone has to go through their files to find previous versions
Lack of attendance	6,00	0,30	1,80	Coordinate well in advance to know who is able to attend the mandatory meetings.	Those who show an inability to attend will be taken into dialogue where a solution will be found.
Technical challenges	3,00	0,75	2,25	Make sure all equipment is in place before the project starts	Technical problems will be solved as soon as possible, and other members will cover if work becomes difficult for the person with the problems
Plagiarism	9,00	0,20	1,80	Only use code you own, or are allowed to copy.	Rewrite code.
Lack of participation	8,00	0,1	0,80	Ensure that each member of the group has a concise task that they are working on at all time	If the member does not actively ask for new tasks, which results in 'laziness', the group must sit down with the member and discuss opportunities and the way forward
Little to no planning	9,00	0,2	1,80	Ensure that each phase is planned carefully, with opportunities for expansion or limitations if needed	The group must sit down together and draw up a new battle plan where the way forward is discussed.
Professional problems/lack of competence	5,00	0,5	2,50	Ensure that the solutions being worked on are within the limits of competence	Time must be set aside and it must be decided whether there is something that needs to be changed, or whether the competence is achievable
One individual dominates a field or subject	5,00	0,15	0,75	It must be planned in advance who does what. It is in everyone's common interest that everyone contributes equally.	If one person dominates a field, it is up to the group to address the individual and find a solution
Unable to meet deadline/desired scope	10,00	0,2	2,00	Have a realistic view of what is possible. Present concern in each phase, and iterate the scope if something seems threatening to get the delivery.	Have a dialogue with the councillor, inform about the status, the way forward and what remains.
Internal disagreement	8,00	0,25	2,00	Follow instructions in the group contract	Follow instruction in the group contract
Iteration/change (demand) from councillor	5,00	0,4	2,00	Take into account that a project in progress can change quickly. Avoid situations where too much code/work is deposited on an uncertain solution.	Get hold of the person responsible for what is to be changed and move/distribute tasks.
Lack of commitment/motivation	8,00	0,2	1,60	Ensure that the team works as a unified unit where everyone has a shared vision and everyone feels respected and seen.	The team leader engages in a dialogue with those concerned and together find a goal and a solution for how commitment/motivation can be found.
Having too much ambition	7,00	0,6	4,20	Ensure in advance that the tasks are feasible and that the group has the competence to carry out the tasks. Seek help from the Supervisor.	Reconsider scope
	117,50	5,15	32,15		

Appendix B

Explanation of the APIs used by ThreatAPIVendor:

- AbuseIPDB API: This API provides access to a comprehensive database of IP addresses reported for involvement in various cyber attacks. By querying the AbuseIPDB API (endpoint: <https://api.abuseipdb.com/api/v2/check>), the script can identify if an IP has been linked to malicious activities in the past.
- AlienVault Open Threat Exchange (OTX) API: The OTX API offers a rich dataset of indicators of compromise (IoCs) which is a piece of evidence that suggests a system has been breached or compromised by a cyberattack. Our indicator being the IP addresses so we utilizes the AlienVault OTX API (endpoint: <https://otx.alienvault.com/api/v1/indicators/IPv4/%s/general>) where '%s' is replaced by an IP to query for information associated with the given IP address. This allows the checker to identify potential threats based on known IoCs.
- ThreatFox API: The ThreatFox API (endpoint: <https://threatfox-api.abuse.ch/api/v1/>) provides threat intelligence data on IoCs, including IP addresses. By integrating this API, the checker expands its scope of investigation and increases the likelihood of detecting potential threats.
- IP Geolocation API: The IP Geolocation API (endpoint: <http://ip-api.com/json/{ip}>)

———— //————