ARCADA

Splunk Dashboard Development for Elisa Navitas

Jonathan Nyman

Degree Thesis

Information Technology

2025

Lärdomsprov

Jonathan Nyman

Splunk dashboard development for Elisa Navitas

Yrkeshögskolan Arcada: Informationsteknik 2025

Uppdragsgivare:

Elisa Navitas

Sammandrag:

Detta lärdomsprov har genomförts i samarbete med Elisas Navitas-team och fokuserar på utvecklingen av en centraliserad dashboard för logghantering och övervakning med hjälp av Splunk. Navitas är ett system som hanterar överföring av känsliga patientdata mellan sjukvårdsdistrikt. Loggövervakningen har tidigare skötts manuellt, vilket är en tidskrävande och felbenägen process som involverat flera servrar. Syftet med projektet var att öka systemets transparens och förbättra felupptäckten genom automatiserad insamling och visualisering av loggar. Studien följde en produktutvecklingsmodell och omfattade systemanalys, installation av ”forwarders”, utveckling av SPL-frågor, övervakningstavlans design samt validering. Splunk valdes för sina avancerade funktioner, realtidsövervakning och för att plattformen redan användes inom Elisa, vilket möjliggjorde integration med befintlig infrastruktur. Under implementeringen uppstod tekniska utmaningar kopplade till hantering av DEBUG-loggar som innehåller konfidentiella data. Dessa loggar exkluderades från Splunk genom att omdirigeras till lokala filer. På den organisatoriska sidan uppstod förseningar på grund av begränsad tillgång till nyckelpersoner och behörighetsbegränsningar i Splunk-miljön. Den slutliga lösningen innefattar en fungerande dashboard med interaktiva visualiseringar av adapterfel, inloggningshändelser och systemloggar, vilket möjliggör snabbare felsökning och upptäckt av mönster. Resultaten visar på ökad effektivitet, förbättrad säkerhetsövervakning och en väldigt bra grund för fortsatt utveckling. Rekommendationer för framtida arbete inkluderar att utöka övervakningen till produktionsmiljöer, använda maskininlärning för avvikelsedetektering samt att införa automatiserade varningar. Arbetet visar hur centraliserad logghantering kan stödja en proaktiv och säker IT-drift.

Nyckelord:

Splunk, log monitoring, system integration, cybersecurity, automation, dashboard, IT, Elisa Navitas

Degree Thesis

Jonathan Nyman

Splunk dashboard development for Elisa Navitas

Arcada University of Applied Sciences: Information Technology 2025

Commissioned by:

Elisa Navitas

Abstract:

This thesis project was conducted in collaboration with Elisa's Navitas team and focused on the development of a centralized log monitoring dashboard using Splunk. Navitas is a critical system responsible for transferring sensitive healthcare data between regions. Its log monitoring was previously handled manually, involving time-consuming and error-prone processes across multiple servers. The aim of the project was to improve system visibility and error detection through automated log collection and visualization. The study followed a product development approach and included system analysis, forwarder installation, SPL query development, dashboard design, and validation. Splunk was chosen for its advanced features, real-time monitoring capabilities, and existing use within Elisa, which allowed integration with the company's infrastructure. During implementation, technical challenges included handling DEBUG-level logs containing confidential data, which were excluded from Splunk by rerouting them to local files. Organizational challenges involved delays due to limited availability of team members and access restrictions within the Splunk environment. The final solution included a functioning dashboard with interactive visualizations for adapter errors, login events, and system-level logs, enabling faster troubleshooting and trend detection. The results showed improved efficiency in identifying issues, enhanced security monitoring, and a scalable foundation for future development. Recommendations included expanding the monitoring scope to production systems, applying machine learning for anomaly detection, and implementing alerting features. This work demonstrates how centralized log management can support proactive IT operations in a secure and efficient way.

Keywords:

Splunk, log monitoring, system integration, cybersecurity, automation, dashboard, IT operations, Elisa Navitas

Contents

1 Introduction

Ensuring the reliability and performance of IT infrastructure is crucial for any modern organization, especially when handling sensitive data transfers. Navitas, a service at Elisa, plays a key role in facilitating data exchange between different healthcare districts. Given its importance, effective system monitoring and error tracking are essential to maintaining seamless operations and preventing disruptions.

Currently, log monitoring at Navitas is performed manually, requiring administrators to check multiple servers individually and run scripts to analyze the logs. This manual approach is time-consuming, inefficient, and lacks the possibility of a historical view of the health of the service. Additionally, the absence of a centralized monitoring system makes it difficult to detect error patterns over time, leading to delayed troubleshooting and a reactive approach to system failures rather than a proactive one.

To address these challenges, this project focuses on developing and implementing a Splunk-based monitoring dashboard that provides the Navitas team with an automated, centralized solution for tracking system logs, identifying recurring errors, and analyzing historical trends. By leveraging Splunk's powerful search, visualization, and alerting capabilities, the system aims to enhance operational efficiency, reduce troubleshooting time, and provide data-driven insights for improving system reliability and decision-making.

1.1 Project Objective

The objective of this project is to develop and implement a Splunk-based monitoring dashboard for Navitas, aimed at improving log management and system monitoring. The project is structured into two main phases: planning and execution.

The planning phase focuses on assessing the feasibility of implementing a Splunk dashboard by evaluating technical constraints, infrastructure requirements, and business needs. This includes identifying the necessary deployment steps, estimating potential

costs such as licensing and hardware, and analyzing challenges related to security, scalability, and inter-team coordination.

The execution phase involves setting up the dashboard and configuring it to monitor selected logs, with a focus on capturing critical application errors, system performance metrics, and security-related events. These logs were identified through collaboration with the Navitas development team and an analysis of past incidents that required manual troubleshooting.

By automating and centralizing log monitoring, this project reduces the manual workload for system administrators, enables faster error detection and resolution, and provides real-time system insights. Additionally, it strengthens security by improving visibility into potential intrusions and anomalies in system access logs.

In the long term, this solution serves as a foundation for further advancements in system monitoring and data analysis at Elisa, supporting a proactive approach to IT operations and system performance.

To achieve these objectives, the project will:

- Analyze the current state of Navitas' log management to identify system needs and technical requirements.
- Plan and execute the Splunk integration for efficient log collection, storage, and analysis.
- Design a user-friendly dashboard that provides clear visibility into critical system parameters and error messages.
- Test and validate system functionality in both test and production environments.
- Assess the impact of the new monitoring solution on troubleshooting time, security, and operational efficiency.

These objectives helped the Navitas team transition from a manual, fragmented monitoring system to an automated, scalable, and intelligent solution, ultimately enhancing system stability, reducing downtime, and improving service quality for all stakeholders.

1.2 Problem Statement

Currently, log monitoring at Navitas is performed manually, requiring administrators to check multiple servers individually and run scripts to diagnose issues. This manual process is both time-consuming and prone to human error, leading to inefficiencies in troubleshooting and an increased risk of system failures. Logs are distributed across multiple servers, and while some are available in a centralized log server, many still require manual intervention to access and analyze. The lack of a unified monitoring solution makes it difficult to correlate errors, identify trends, and respond to incidents efficiently.

Furthermore, the absence of historical log data analysis prevents the team from proactively identifying recurring issues or predicting potential failures before they impact system performance. Without a centralized dashboard, system administrators rely on reactive troubleshooting rather than data-driven decision-making.

Another challenge is the feasibility assessment of implementing a Splunk-based solution. Several factors need to be evaluated, including technical constraints, infrastructure requirements, and cost considerations. Additionally, integrating a new monitoring system must align with security and compliance regulations, ensuring that sensitive data is properly masked and that access control policies are in place.

Based on this background, the project aims to address the following **research questions**:

- How much faster can errors be detected with Splunk compared to the manual method?
- How does the response time to issues change when using automated log analysis?
- In what ways does access to historical log data improve system reliability?
- To what extent can a centralized dashboard reduce the workload for system administrators?

This project addresses these challenges by developing a centralized, automated, and intelligent Splunk-based monitoring system that enhances error tracking, historical analysis, and real-time system monitoring. By implementing this solution, the Navitas team was able to transition from a fragmented, manual approach to a proactive, data-driven

system, significantly improving troubleshooting efficiency, system reliability, and security monitoring.

1.3 Scope of the Project

This project focuses on the design, implementation, and evaluation of a Splunk-based monitoring dashboard tailored for the Navitas service at Elisa. The primary goal is to create a centralized monitoring system that enables automated log tracking, real-time error detection, and historical trend analysis to improve system performance and troubleshooting efficiency.

The dashboard monitors critical logs, including application logs, system logs, and security-related logs, providing administrators with a clear overview of system health. Additionally, it incorporates historical data analysis to help identify recurring issues and patterns, allowing for proactive problem resolution. As a future development suggestion, the system could also include real-time alerting, enabling automated notifications for critical failures and anomalies, ensuring that the Navitas team can quickly respond to system incidents.

Security and compliance are an integral part of the implementation. The project ensures that log data is properly masked before ingestion into Splunk, preventing exposure of sensitive information, and restricting access based on predefined user roles and permissions. However, the scope of this project is limited to Navitas servers and will not extend to other Elisa systems. Additionally, while the dashboard generates automated alerts for anomalies, it does not include full automation of incident response. The project also does not cover long-term log storage, as retention policies are defined but compliance-related archival remains outside the project's scope.

By maintaining a well-defined scope, this project ensures that it remains focused on solving Navitas' log monitoring challenges while allowing for future scalability and enhancements based on evolving operational needs.

2 Background and Research

2.1 What is Splunk

Splunk (Splunk, n.d.) is a powerful log management and analysis platform designed to collect, index, and analyze machine-generated data from various systems and services. It offers real-time monitoring, advanced search capabilities, and customizable dashboards, which help organizations detect, understand, and act on operational data more efficiently. Unlike traditional log aggregation tools, Splunk enables users to filter, correlate, and visualize data without complex configurations.

Compared to competing solutions such as the ELK Stack (Elastic, n.d.) or Datadog (Datadog, n.d.), Splunk stands out due to its intuitive interface, robust automation features, and deep customization options. The ELK Stack, while open source and highly flexible, often requires more manual setup and configuration. Datadog, on the other hand, offers strong monitoring capabilities but is more limited when it comes to custom log parsing and complex search queries.

Splunk's ability to support both real-time and historical log analysis makes it a particularly effective solution for organizations with complex IT environments. For example, a financial institution might use Splunk to detect fraudulent transactions, while a healthcare provider could monitor system logs to ensure the integrity of sensitive data processing. These capabilities are achieved through Splunk's indexing engine and Search Processing Language (SPL), which allows powerful queries to be built for a wide variety of use cases

2.2 Existing Log Monitoring at Navitas

Currently, the monitoring of Navitas logs is done manually, with approximately 15 different log sources being checked individually by administrators. This process can take several hours each day, as administrators must log into multiple servers, retrieve logs, and manually analyze them for anomalies. Due to this being such a time-consuming task, logs are rarely checked without a cause which means that a lot of problems might go undetected for a long time.

The current lack of a centralized monitoring system also means that there is limited ability to analyze historical data. The absence of automated correlation significantly slows down root cause analysis. These challenges illustrate the need for a solution like Splunk, which can automate and centralize log data for more effective system monitoring and troubleshooting, making it easier and quicker to detect problems before they become big enough to impact the end users.

2.3 Challenges in Implementing Log Management

One of the primary challenges in implementing a centralized log management system for Navitas is access control and security compliance. The Navitas team consists of members who have all signed NDAs, allowing them to handle sensitive patient data and granting them full access to logs for operational monitoring, troubleshooting, and performance analysis.

However, since Monttu is Elisa's shared Splunk instance, multiple teams use the same platform for log monitoring and dashboarding. This means that access is not limited only to the security team but also includes other departments who utilize Splunk for their own systems. As a result, sensitive log data from Navitas could potentially be visible to people outside the Navitas team, raising concerns about data confidentiality and compliance.

This shared environment presents challenges in deciding what types of log data can be safely forwarded to Splunk, how sensitive information should be masked, and how access rights should be managed. To ensure proper handling of log data, Role-Based Access Control (RBAC) is applied. This ensures that only authorized users from relevant teams have access to specific dashboards and log sources, reducing the risk of unauthorized viewing while still allowing efficient operational monitoring.

Another challenge is scalability, which in this project primarily refers to the deployment and configuration of Splunk forwarders on every relevant server, both in test and production environments. Each server requires a proper setup to ensure that logs are forwarded correctly while maintaining security and compliance. This includes defining log sources,

implementing filters to exclude unnecessary data, and ensuring that sensitive information is masked before ingestion into Splunk.

To address scalability concerns, it would be ideal to pre-install Splunk forwarders on all servers that might need monitoring in the future. This proactive approach ensures that forwarders are ready for use whenever new monitoring requirements arise, eliminating the need for last-minute configurations. Additionally, creating clear and well-structured installation guides for future deployments would simplify the process for administrators, reducing setup time and ensuring consistency across all servers.

2.4 Research Context and Relevance

Log management and real-time monitoring have become key components of modern IT operations. Research (Mäkelä, 2020) has shown that automated log analysis tools like Splunk can significantly reduce the time spent on incident resolution and help detect anomalies before they escalate. Real-time data processing enables organizations to move from a reactive to a proactive mode of operation, where potential issues can be identified and addressed before they impact service availability or security.

Splunk also plays a crucial role in enhancing cybersecurity. With its ability to generate alerts and visualize unusual behavior, it supports rapid detection of suspicious activity such as failed login attempts or access from unknown locations. According to the referenced thesis, organizations using Splunk-like systems are often better equipped to meet security compliance requirements and prevent data breaches.

A focused review of existing material was conducted using Theseus and Google, with search terms such as “Splunk log management,” “IT security log analysis,” and “real-time log analytics.” The earlier mentioned selected source (Mäkelä, 2020) discusses the implementation and benefits of Splunk in log management and IT monitoring. It highlights Splunk’s effectiveness in handling large volumes of operational data and emphasizes best practices such as structured indexing strategies, data filtering, and the integration of machine learning to improve log analysis outcomes.

3 Research Approach

To make sure the Splunk-based monitoring dashboard works well for Navitas, a combination of product development, testing, data collection, and analysis methods is used. The goal is to create a useful and easy-to-use monitoring solution that helps the Navitas team track system logs, detect errors, and analyze historical data.

The development follows an iterative process, meaning that requirement specification, design, implementation, and testing are done in cycles. Using an agile approach, the system is improved step by step based on testing results and feedback from users. This way, the dashboard can be adjusted to meet the needs of the team.

3.1 Research Methods

The project is based on a product development approach, where a functional monitoring system is designed, implemented, tested, and evaluated in real-world conditions. The research is practical and application-oriented, focusing on how well Splunk can be integrated into the Navitas environment and how effectively it can improve log management, error detection, and troubleshooting.

The main steps in this process are:

- Requirement Specification: Deciding which log sources should be monitored, what kind of errors and system parameters should be tracked, and how the dashboard should look.
- System Implementation: Setting up Splunk Universal Forwarders on test and production servers to make sure logs are collected properly.
- Dashboard Design: Creating visual views that make it easy to understand system performance, errors, and security risks.
- Testing & Evaluation: Simulating system failures and incidents to check how well the dashboard detects problems and alerts the administrators.

By using this step-by-step approach, the system is continuously improved so that it works in a real production environment.

3.2 Data Collection Methods

To collect data, Splunk Universal Forwarders are installed on Navitas servers. These forwarders send logs to Splunk, where they are indexed and stored. The logs are then analyzed to find patterns, detect errors, and identify security risks.

The dashboard is tested in both a test environment and in the real production environment. Different system errors and failures are simulated to measure how well the dashboard reacts. Splunk's built-in Search Processing Language (SPL) is used to analyze logs and see how system performance changes before and after using Splunk.

To make sure the results are trustworthy, tests are done over a longer period and include logs from different types of system events. By comparing past incidents with real-time monitoring data, the accuracy and efficiency of the system are evaluated.

3.3 Justification for Research Approach

Splunk was chosen for this project because it is a powerful and scalable tool for log management and monitoring, offering both real-time tracking and historical analysis of system data. After evaluating alternative log management solutions such as ELK Stack and Datadog, Splunk was found to be the most suitable option for the Navitas team due to several key factors:

- Existing Use at Elisa: One major reason for selecting Splunk is that Elisa already uses Splunk in another team, meaning that the infrastructure is already in place. This significantly reduced setup time, implementation complexity, and costs, as a Splunk server was already available for integration.
- User-Friendly Interface & Customization: Splunk provides an intuitive dashboard, allowing administrators to search logs, filter data, and create visual reports without requiring extensive programming knowledge.
- Powerful Log Searching & Filtering: Splunk's Search Processing Language (SPL) enables efficient querying of large log datasets, making it easier to identify trends, detect recurring errors, and analyze system performance.

- Real-Time Alerts & Automation: Unlike manual log-checking methods, Splunk can automatically notify administrators when critical system errors or security threats occur, significantly improving response times and reducing downtime.
- Seamless Integration with Elisa's IT Infrastructure: Splunk fits well within Elisa's existing IT ecosystem, allowing for direct log collection from Navitas servers without requiring extensive modifications to current systems.
- Security & Compliance Features: Because Navitas handles sensitive healthcare data, security is a top priority. Splunk offers Role-Based Access Control (RBAC), data masking, and encryption, ensuring that log data is handled securely while complying with data protection regulations.

By choosing Splunk, the project ensures that log collection, monitoring, and troubleshooting are automated, making the process faster, more accurate, and more scalable than traditional manual methods. Additionally, Splunk's flexibility allows for future expansion, meaning it can be enhanced to support new monitoring needs as the Navitas service continues to evolve.

4 Project Planning

Proper planning is essential to ensure the successful implementation of the Splunk-based monitoring dashboard for Navitas. This section outlines the key requirements, timeline, and team responsibilities that are necessary for the project's execution.

4.1 Defining Key Requirements

The implementation of the Splunk dashboard relies on Elisa's existing Splunk infrastructure, specifically the Monttu Splunk server, which is managed by Elisa's security team. Within Monttu, different teams have their own instances, and the Navitas team has been assigned a separate instance for log monitoring. This setup removes the need to set up a new Splunk server from scratch and allows the team to integrate into an already operational system.

One of the key requirements of this project is the deployment of Splunk Universal Forwarders on all servers that currently generate logs. These forwarders ensure that relevant logs are collected and sent to Monttu for indexing and analysis. However, access control has been a challenge due to the sensitive nature of the data handled by Navitas. Since the service processes patient data and social security numbers, it is critical that logs do not expose unmasked confidential information. Initially, some ERROR logs contained unmasked patient data, which required proper masking before being forwarded to Splunk. Another issue was that DEBUG logs contained sensitive data that was needed for troubleshooting but could not be sent to Monttu, requiring an alternative solution. The final decision was to store DEBUG logs separately, ensuring that only properly masked and necessary logs were sent to Splunk while maintaining the ability to troubleshoot when needed.

The functional requirements of the dashboard focus on providing the Navitas team with a historical view of system health and errors. At present, logs are checked only when an issue arises, meaning that there is no proactive system health monitoring. The dashboard is intended to give the team a real-time overview of system health, error occurrences, and warning trends, allowing for quicker troubleshooting and better long-term system stability. To achieve this, the dashboard must include a clear visualization of critical system parameters, error tracking, filtering options, and historical data analysis to help identify recurring issues before they escalate into larger problems.

Security and compliance are also important factors in this project. It is essential to ensure that log data is properly masked before being ingested into Splunk to prevent the exposure of sensitive information. Access to logs must also be restricted based on predefined user roles and permissions, ensuring that only authorized personnel can view and process specific logs. The project is limited to Navitas servers only and does not extend to other Elisa systems. While the dashboard provides automated alerts for system anomalies, it does not include full automation of incident responses. Additionally, the project does not cover long-term log storage, as retention policies are set, but archival compliance remains outside the project scope.

By defining clear technical and functional requirements, the project ensures that the monitoring system remains focused on solving Navitas' log management challenges while allowing for future scalability and enhancements based on operational needs.

4.2 Project Timeline & Milestones

The project was originally scheduled to be completed in December, but due to delays caused by dependencies on multiple teams, the timeline has been extended. One of the biggest challenges has been that this project is not the primary focus of any of the teams involved, meaning that coordinating efforts between different teams has taken longer than expected. Despite this, significant progress has been made, and several major milestones have been achieved.

One of the most important milestones was discovering that Elisa already had a Splunk server, Monttu, which meant that setting up a new Splunk instance was not necessary. This significantly reduced the complexity of the project and allowed the team to focus on integrating Navitas logs into an existing system rather than building one from scratch. Another important step was installing and configuring the first Splunk Universal Forwarders, which made it possible to send logs from Navitas servers to Monttu and verify that communication between the systems was working as expected.

Security issues were another major milestone in the project. During testing, it became clear that some ERROR logs contained sensitive patient data that needed to be masked before being sent to Splunk. Additionally, it was discovered that DEBUG logs contained unmasked confidential data, which made it impossible to send them to Monttu without violating security policies. After discussions between teams, a solution was implemented where DEBUG logs were stored separately, ensuring that they were available for troubleshooting while keeping them out of Splunk to maintain compliance.

The next major milestone in the project was the development of the actual Splunk dashboard. This involved configuring the visualization of key metrics, setting up filtering options, and defining alert conditions based on the needs of the Navitas team. The dashboard

was built in collaboration with the Navitas team, who defined what data they needed and how it should be presented.

Although the timeline had been affected by delays and dependencies, the project was now moving towards its final phase, where the dashboard would be developed and tested before being fully deployed.

4.3 Resources & Team Responsibilities

The project involved multiple teams and individuals, each with specific responsibilities. The main participants in the project have been the Navitas team, the Splunk manager from the security team, and other contributors from Elisa's IT department. Each team has played a crucial role in ensuring that the system is properly implemented while meeting security and operational requirements.

The Navitas team was responsible for defining what logs should be monitored, which errors are critical, and what data needs to be visualized in the dashboard. Their role was to ensure that the final system meets their operational needs and provides useful insights into system health and error occurrences.

My role in this project has been to act as the main contact person and driving force behind the project. I have been responsible for coordinating between different teams, researching solutions, and pushing the project forward despite the delays caused by dependencies on others. Additionally, I have been in charge of writing this report and documenting the implementation process. The next phase of my responsibility was to create the Splunk dashboard based on input from the Navitas team, ensuring that it meets their expectations and provides them with the data they need for effective monitoring.

The Splunk manager from the security team has played a critical role in managing the Monttu Splunk instance, ensuring that logs are properly configured, and access is restricted according to security policies. One of the major tasks for this team has been

implementing RBAC (Role-Based Access Control) to ensure that only authorized users from Navitas have access to their specific logs in Splunk.

Other contributors, including members of Elisa's IT department, have provided technical support and expertise, particularly in areas like log filtering, data masking, and system integration. Their involvement has been important in resolving technical issues related to log transmission, storage, and security.

By defining clear roles and responsibilities, the project has been able to progress despite delays and challenges. With the final phase approaching, the focus is now on building the Splunk dashboard, testing its functionality, and ensuring that it meets the needs of the Navitas team.

5 Project Execution

This section outlines the steps taken to implement the Splunk-based monitoring system for the Navitas service. The execution phase involved setting up Splunk, deploying forwarders, developing queries, designing the dashboard, configuring alerts, and performing validation and testing to ensure the system works as expected.

5.1 Setting Up Splunk for Navitas

The Navitas Splunk instance was set up within Elisa's Monttu Splunk environment, which is managed by the ITBU team. Monttu provides a centralized Splunk infrastructure where different teams have their own isolated instances for log monitoring. This setup eliminated the need to configure a new Splunk server from scratch, allowing Navitas to integrate directly into an existing, managed environment.

Monttu consists of a clustered indexing system where two indexer nodes receive data from Splunk Universal Forwarders. The Search Head is the interface that users, including the Navitas team, use to query and analyze logs. Additionally, a Deployment Server is used to manage forwarder configurations by deploying predefined settings and policies to ensure consistency across the environment.

The access to Navitas' Splunk instance was set up through Active Directory (AD) authentication using a group called ACL-NAVITAS_Test. This ensures that only authorized users have access to the system while maintaining compliance with security policies.

5.2 Deploying Splunk Forwarders on Test & Production Servers

To enable automated log collection and monitoring for the Navitas service, Splunk Universal Forwarders were installed on both test and production servers. These forwarders act as agents that collect log data from Navitas servers and transmit it to the Monttu Splunk instance, where logs are indexed and analyzed.

The forwarder deployment followed Elisa’s internal Splunk guidelines, ensuring compatibility with Monttu’s existing infrastructure. For security reasons, all IP addresses, file paths, and system details have been replaced with generic placeholders.

5.2.1 Network Configuration & Connectivity

The most critical logs from Navitas are collected in Monttu, a shared Splunk platform managed by ITBU. A dedicated instance has been set up for Navitas within Monttu to facilitate service monitoring and troubleshooting within the Navitas environment. The goal is to build various views for system tracking and improve error detection.

The following Navitas networks have been configured with a connection to Monttu:

Table 1. Network segmentation and description of the Navitas infrastructure

Network (CIDR)	Description
10.xxx.xx.xxx/29	AD Network (Navitas authentication servers)
217.xxx.xxx.xxx/28	External Network (Jump servers and development servers)
217.xxx.xxx.xxx/27	Test Network (Test environment servers)

10.xxx.xxx.xxx/24	Trust Network (Elisa-maintained servers)
194.xxx.xxx.xxx/27	Production Network (Production servers)

If a network is unable to establish a connection with Monttu, the issue is likely caused by missing firewall rules or incorrect routing configurations.
Steps to Resolve Connectivity Issues:

Once network access to Monttu is confirmed (ping 10.X.X.X → OK), Splunk forwarders can be installed on the source servers.

Table 2. Firewall and routing tasks enabling Splunk integration with Navitas systems

Task	Action	Assigned to
Firewall rule update	Add source network to "navitas-to-monttu" rule to allow communication with Monttu.	Cloud Service or Security Team
Routing update	Add a new route `vpnn 300835` to direct traffic towards the Navitas firewall.	Cloud Service or Network Team
Production firewall update	Add a route for the network (x.x.x.x/x next-hop 10.xxx.xxx.xxx) and allow traffic to Splunk.	Security Team (INC reference)

5.2.2 Installing Splunk Forwarders on Linux Servers

1. Transfer the Splunk Forwarder package to the target server using WinSCP or a similar file transfer tool
2. Gain root access:
   ```
   sudo su
   ```
3. Install the RPM package:
   ```
   rpm -i splunkforwarder-8.2.2.x86_64.rpm
   ```
4. Accept the license:
   ```
   /opt/splunkforwarder/bin/splunk start –accept-license
   ```

5. Set up Splunk user credentials:

   Use password for personal splunk instance

6. Define the Splunk indexer address:

   /opt/splunkforwarder/bin/splunk add forwarder-server <SPLUNK_INDEXER_IP>_9997

7. Configure deployment settings:

   /opt/splunkforwarder/bin/splunk set deploy-poll <DEPLOYMENT_SERVER_IP>:8089

8. Restart and enable boot-time execution:

   /opt/splunkforwarder/bin/splunk stop
   /opt/splunkforwarder/bin/splunk enable boot-start

9. Verify the server's name in the configuration file:

   cat /opt/splunkforwarder/etc/system/local/server.conf

10. Modify the outputs configuration:

    nano /opt/splunkforwarder/etc/system/local/outputs.conf
    sql:
    Update it to include:
    ```
    [tcpout]
    defaultGroup = default-autolb-group
    [tcpout:default-autolb-group]
    sercer = <SPLUNK_INDEXER_IP>:9997
    disable =  false
    ```

11. Start the Splunk forwarder:

    /opt/splunkforwarder/bin/splunk start

5.2.3 Forwarder Installation on Windows Servers

1. For forwarder installation Administrator level credentials are needed:

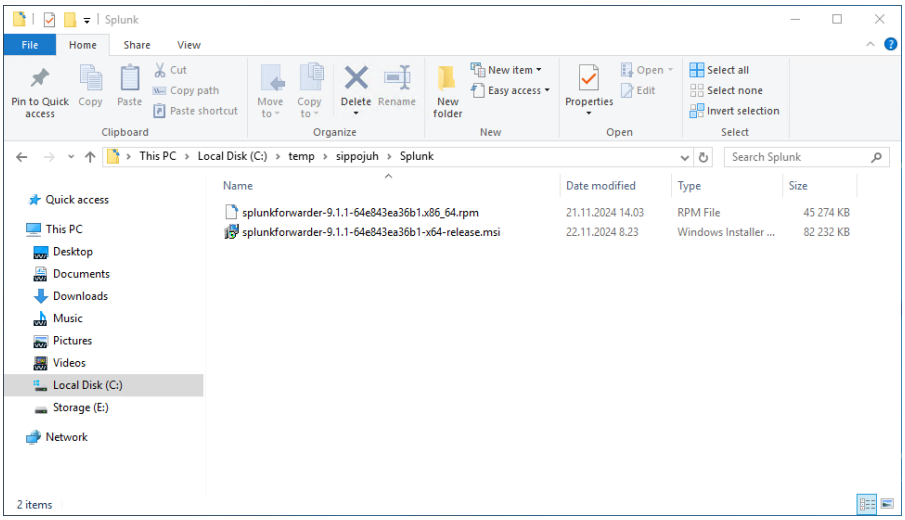

Figure 11, Windows installation files

2. Click the “Splunkforwarder*.msi” installation package and choose next:

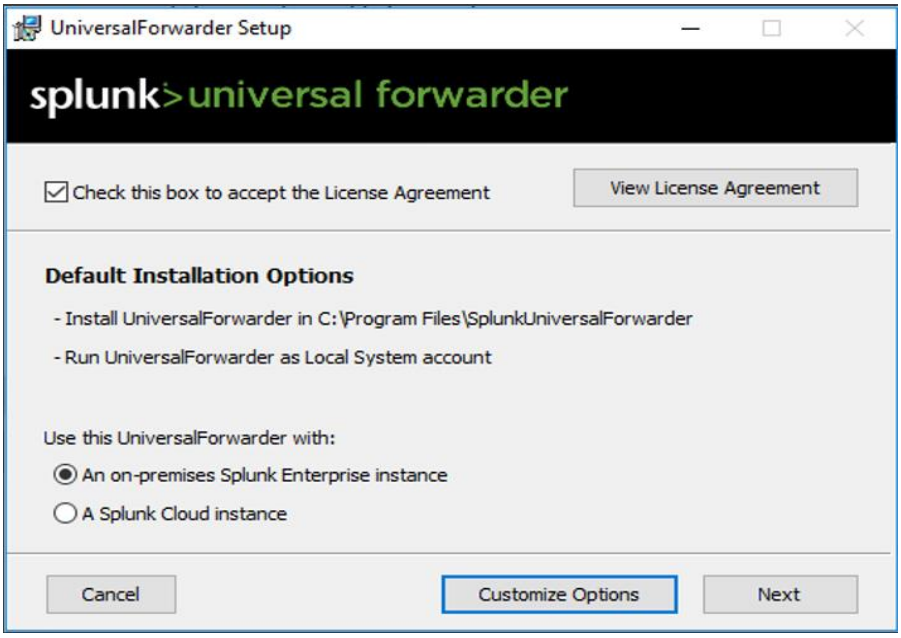

Figure 12, Splunk installation wizard for windows

3. Using your Splunk credentials fill in the fields and click next:

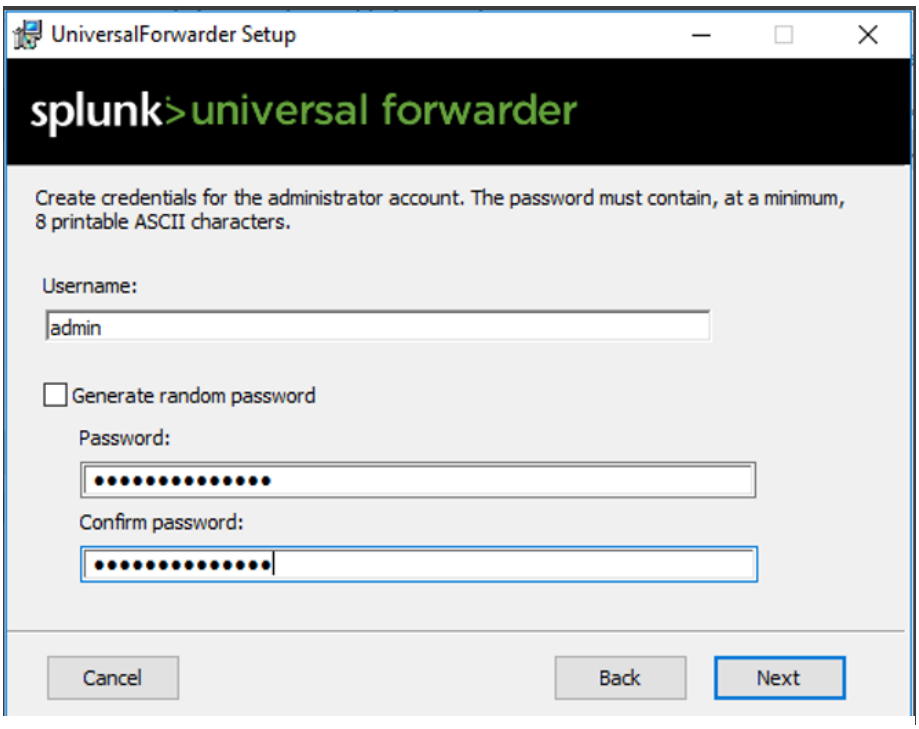

Figure 13, Splunk installation wizard for windows

4. Enter the Monttu deployment-server address and port and click next:

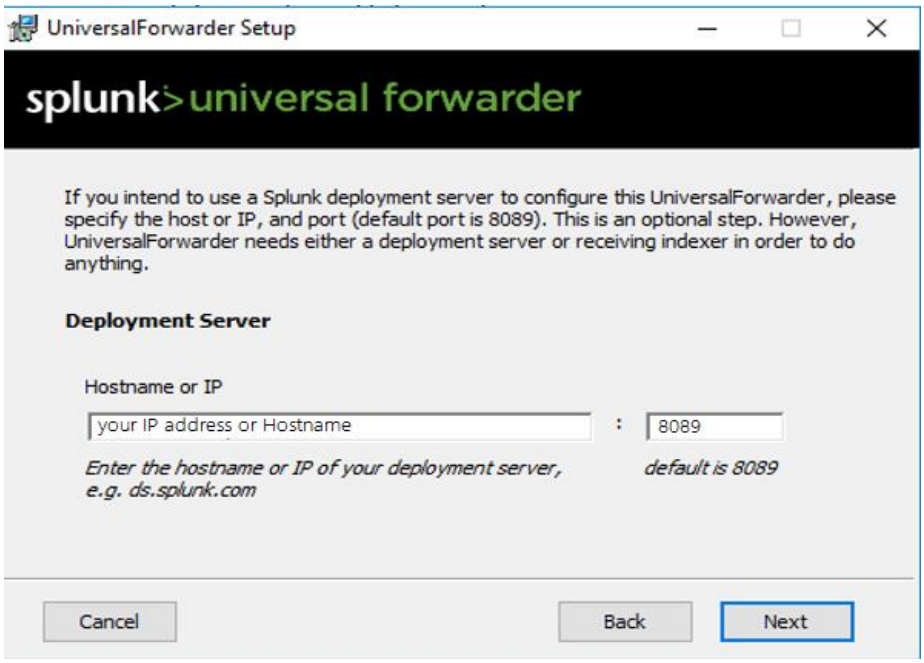

Figure 14, Splunk installation wizard for windows

5. Enter the Monttu Indexer-server address and port and click next:

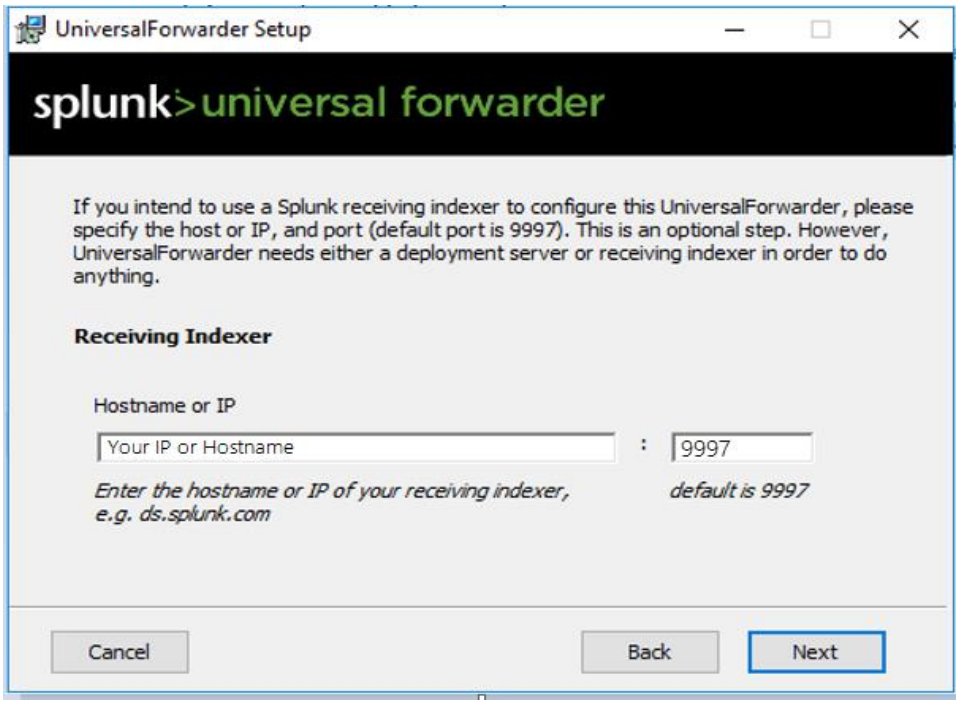

Figure 15, Splunk installation wizard for windows

6. Start the installation by clicking Install:

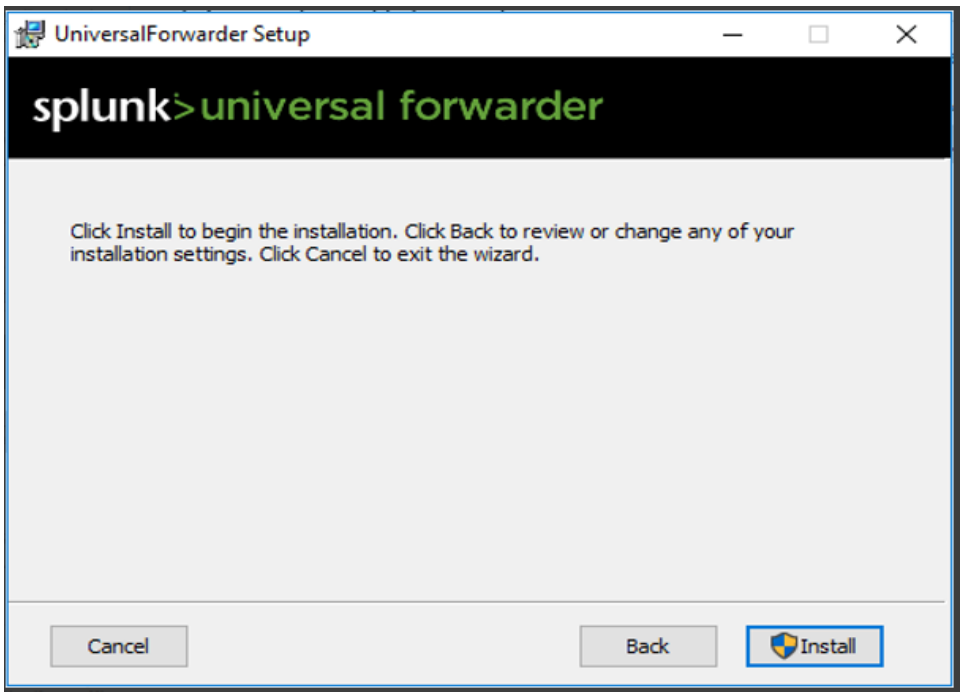

Figure 16, Splunk installation wizard for windows

7. Wait for the installation (takes a long time):

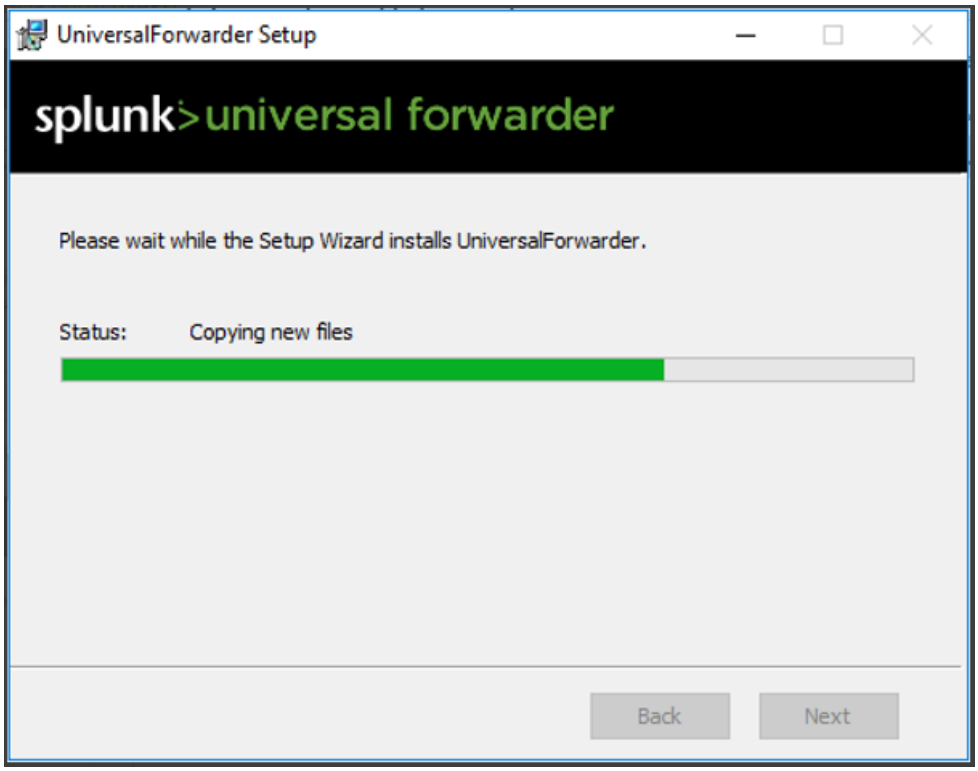

Figure 17, Splunk installation wizard for windows

8. when the installation is done click Finish:

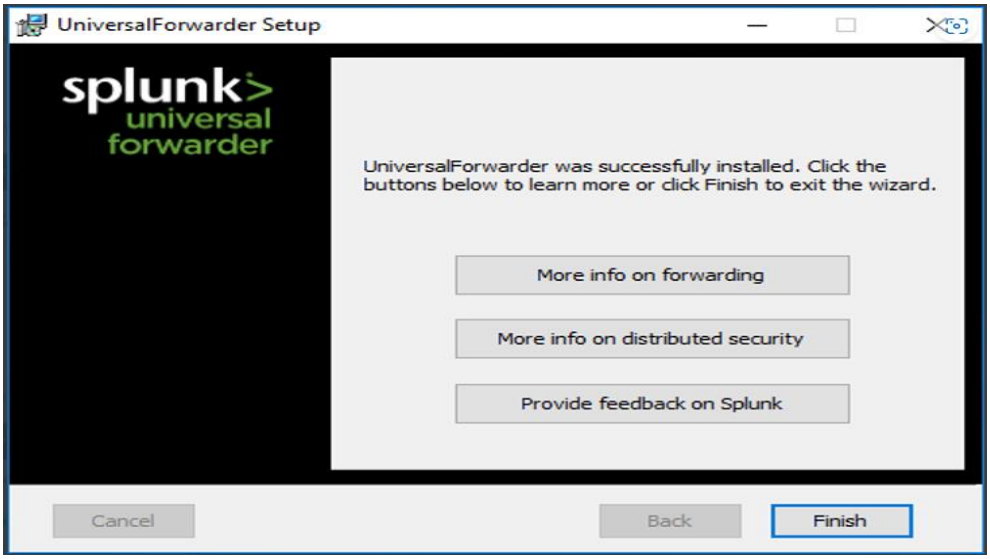

Figure 18, Splunk installation wizard for windows

9. Go to the path: C:\Program Files\SplunkUniversalForwarder\etc\system\local

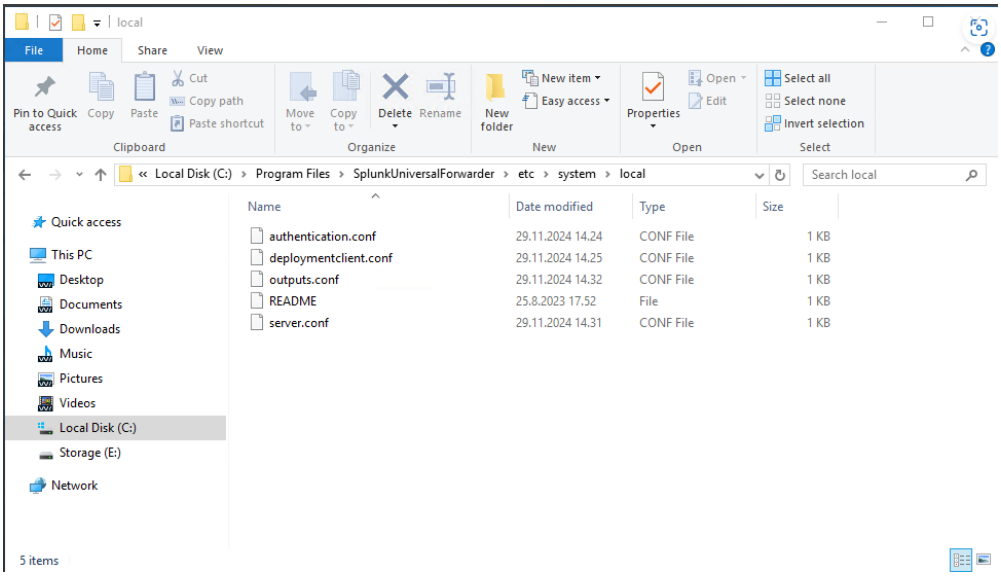

Figure 19, Forwarder deployment settings file location

10. Edit the server.conf file – fill in the serverName and click save:

```
[general]
serverName = Yourmgmtserver.servername.com
pass4SymKey = $yourgeneratedpasswordfortheforwarder?=
[sslConfig]
sslPassword = $yourgeneratedsslpassword?=
[1mpool:auto_generated_pool_forwarder]
description = auto_generated_pool_forwarder
peers = *
quora = MAX
stack_id = forwarder
[1mpool:auto_generated_pool_free]
description = auto_generated_pool_free
peers = *
quora = MAX
stack_id = free
```

11. Edit the outputs.conf file – fill in the rows 6 and 7 and click save:

```
[tcpout]
defaultgroup = default-autolb-group
server = your.server.name.or.ip, your.backup.server.name.or.ip
disabled=false
```

12. The changes come into effect after restarting the Splunk process:

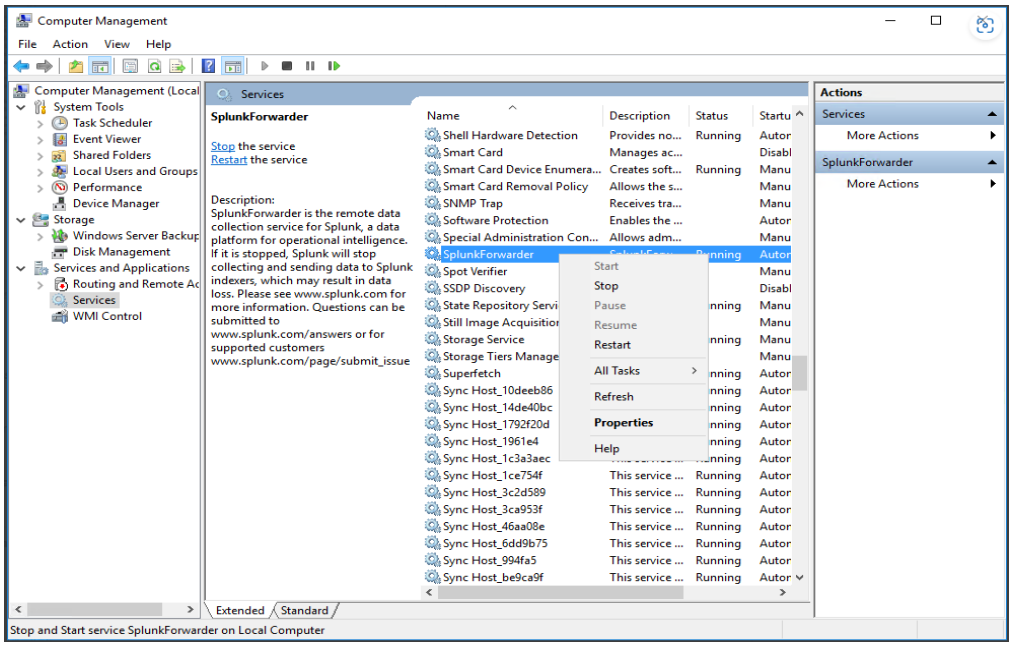

Figure 20, Splunk forwarder service

5.2.4 Configure the Server Side Monttu Instance with Log Paths

1. Find the paths for the logs you want to be able to see in Splunk
2. Verify that the “splunkfwd” user has the necessary permissions to access and read the logs
3. Log in to the Monttu management server and choose: “New Server Class”
4. Select “Add Clients”
5. Select the previously created Navitas-server (testapp1)
6. Through the SSH-client command line open the inputs.conf file: nano/opt/splunk/deployment-apps/Monttu_navitas_test_app/local/inputs.conf
7. In this file you can see the servers log paths, add the paths, file type and duration for the logs
8. After this finish the “testapp1” settings after which you are ready to start querying logs

5.3 Query Development for Error Tracking

After successfully installing the Splunk forwarders and ensuring that logs were being ingested into the Monttu Splunk instance, the next step was to develop queries that would allow the Navitas team to track critical system events efficiently. To start, the Navitas team participated in a meeting with two Splunk professionals who introduced us to Splunk Search Processing Language (SPL). This session covered the basics of querying logs in Splunk, best practices for writing efficient queries, and some of the most commonly used commands. During the session, we also performed hands-on testing by running simple queries to gain a better understanding of how Splunk retrieves data.

Following this introductory session, the Navitas team held an internal meeting to define the key queries they wanted to include in the monitoring dashboard. The team identified several essential data points for tracking errors and system activity. They wanted the dashboard to initially include the following views:

- Default View: Display logs for the past week with the possibility of dynamically adjusting graph resolution.
- Failed Data Retrieval from Adapter:
    - Adapter name/identifier (Adapter).
    - Reference identifier (ReferenceKey).
    - Line graph showing counts per adapter with drill-down capability.
- ERROR-Level Errors:
    - Line graph showing counts per adapter with drill-down capability
- ERROR-Level Errors by Error Type:
    - Table (default view: past 24 hours) displaying error type and count.
- Failed Logins:
    - Categorized by login type: username/password, VRK card, desktop integration, and count.
    - Table (default view: past 24 hours) displaying type and count.
- Successful Logins:
    - Same as failed logins but for successful login attempts.

When starting the query development, it was important to follow best practices to ensure efficient searches that would not overload the shared Monttu Splunk servers. Since multiple teams use this Splunk instance, excessive queries could cause performance issues. The best approach for writing efficient SPL queries involves defining the time window first, specifying the correct index upfront, and applying filters as early as possible to reduce the dataset before further processing. Splunk provides comprehensive documentation on how to optimize search performance, which was a valuable resource during this stage.

To begin developing the queries, I started with the Monttu_navitas_test index, which contains all logs being forwarded from the Navitas servers. For analyzing adapter data retrieval errors, the first step was to define an appropriate time window. Since the team wanted a default view covering one week, I initially set the time window to 24 hours to check for recent logs. If necessary, the timeframe could be expanded incrementally. The time window could be manually set in the Splunk UI or specified in the search query using the earliest=-3d command.

Filtering out unnecessary log levels was another important step. Since debug-level logs contain sensitive data and should not be included in production queries, I applied a filter to exclude them. This was done using level!=DEBUG, which removes DEBUG-level logs from the results. Alternatively, an explicit selection of required log levels could be used instead, such as level=INFO OR level=ERROR OR level=WARN. This approach is often more efficient because excluding logs with != still requires searching through all logs before filtering, while selecting specific levels ensures that only relevant logs are retrieved from the beginning.

Since Splunk automatically assigns fields to logs, I used the sourcetype field to narrow the search further. Most Navitas adapter logs fall under sourcetype="navitasvthlogs", so this was included in the query. Additionally, logs were being generated from both the primary and backup hosts. Since the Navitas team wanted to focus on the primary host, I excluded logs from the secondary host using host!="NAVITASTESTAPP2.ELISANAVITAS.COM".

To focus only on adapter-related errors, I searched for logs containing the "Adapter=" string within the log message. However, Splunk did not automatically extract the adapter name as a separate field, requiring additional processing using Regular Expression Extraction (rex). Since both adapter names and reference keys were embedded within the raw log message, I used rex to extract these values into separate fields.

For extracting the Reference Key, I applied the following rex command:

```
rex field=_raw "ReferenceKey=\s*(?<reference_key>[^\s|]+)"
```

Figure 21, Splunk query for extracting reference key

This command operates on the _raw log data, capturing the ReferenceKey value and assigning it to a new field called reference_key. The \s* pattern accounts for any spaces after "ReferenceKey=", while (?<reference_key>[^\s|]+) captures everything except spaces or delimiters.
Similarly, to extract the Adapter Name, I used:

```
rex field=_raw "Adapter=\s*(?<adapter>[^\s]+)"
```

Figure 22, Splunk query for extracting adapters

This ensures that the extracted adapter name is assigned to a new field called adapter.
At this point, the full query appeared as follows:

Splunk Query

```
1  index=monttu_navitas_test sourcetype="navitasvthlogs" level
2  host!="NAVITASTESTAPP2.ELISANAVITAS.COM"
3  | rex field=_raw "ReferenceKey=\s*(?<reference_key>[^\s|]+)
4  | rex field=_raw "Adapter=\s*(?<adapter>[^\s]+)"
5
```

Figure 23, Complete Splunk query for adapter and reference key extraction

This query formed the foundation for tracking adapter failures and served as a basis for additional error tracking queries that would be used in the Splunk dashboard.

After successfully retrieving and analyzing adapter-related errors, the next query to develop was for tracking failed login attempts. Failed logins are an important metric for system monitoring, as they can indicate both user issues and potential security threats, such as unauthorized access attempts or system misconfigurations.

To start, I used the same approach as before. First identifying relevant logs within Splunk. I began by setting the index to “Monttu_navitas_test” and filtering logs from the “navitasvthlogs” source type. Since the system logs events from multiple servers, I needed to ensure that results were only retrieved from the primary host (NAVITASTESTAPP1.ELISANAVITAS.COM), excluding the backup server. This was done using the filter host!="NAVITASTESTAPP2.ELISANAVITAS.COM". This again gave me a bunch of events from which I started looking for login related events. To help me find a failed login event I asked a member of the Navitas team to do a failed login, after which I set the time span in Splunk to 15 minutes.

By doing this I found two events related to the failed login. The events both had the same string “Kirjautuminen epaonnistui kayttajalla:” which is Finnish for “Login failed for user:” This string is consistently present in authentication failure events, making it a reliable filter. Since the failed attempts are so few I expanded the search time to 30 days. In the 30 days span I found 22 events. My query now looked like this:

```
index=monttu_navitas_test sourcetype="navitasvthlogs" host!="NAVITASTESTAPP2.ELISANAVITAS.COM"
"Kirjautuminen epaonnistui kayttajalla"
```

Figure 24, Splunk query for failed logins

Once I had a basic search query that displayed failed login logs, I structured the data to visualize login failures over time. For this, I used the timechart command, which allows me to create a time-series representation of failed logins. The count by host argument ensures that the data is grouped by the host where the login failure occurred. To avoid

performance issues and excessive data points, I limited the number of hosts displayed in the chart to 200.

Since the team wanted to be able to easily change the timeframe of the failed logins I added the parameter: earliest=$dd_timespan$ which I was used as a variable to store a date from a dropdown menu later.

```
index=monttu_navitas_test sourcetype="navitasvthlogs"
host!="NAVITASTESTAPP2.ELISANAVITAS.COM"
"Kirjautuminen epaonnistui kayttajalla:" earliest=$dd_timespan$
```

Figure 25, Final query for failed login attempts

To visualize the failed logins over time, a timechart function was used. The timechart command groups events into time intervals and counts the occurrences of failed logins per host. The limit=200 parameter ensures that a reasonable number of hosts are displayed in the graph.

With these modifications, the final query was:

```
index=monttu_navitas_test sourcetype="navitasvthlogs"
host!="NAVITASTESTAPP2.ELISANAVITAS.COM"
"Kirjautuminen epaonnistui kayttajalla:" earliest=$dd_timespan$
| timechart count by host limit=200
```

Figure 26, Query for failed logins with graphics

This query provides a clear visual representation of failed login attempts across different hosts. It allows the team to quickly detect unusual patterns, such as sudden spikes in failed login attempts, which may indicate security threats or authentication system failures.

This is the result of this Query:

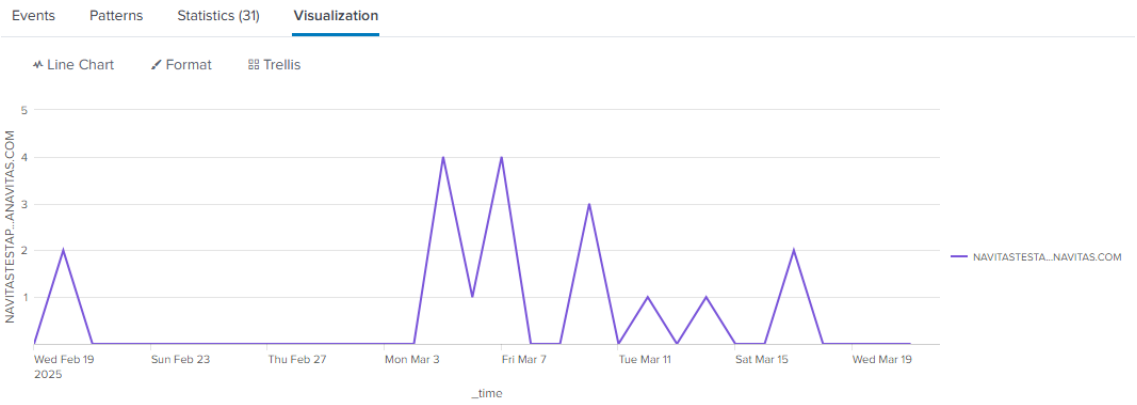

Figure 27, Failed logins graphic results

The final query to develop is the tracking of ERROR events. This is a quite simple query since it includes all ERROR events. Since the team wanted to see the ERROR events for the last 24 hours, we can use the earliest=-24h query to always see the last 24 hours. The query for all ERROR events within the last 24 hours would be:

```
index=monttu_navitas_test level=ERROR earliest=-24h
```

Figure 28, Error events under 24 hours query

Since they also wanted it as a chart, we can add the function for creating a time chart. The final SPL query for these events would then be:

```
index=monttu_navitas_test level=ERROR earliest=-24h | timechart count by host
```

Figure 29, Fina query for error events with visualization

5.4 Designing the Splunk Dashboard for Log Monitoring

Once the necessary queries for log monitoring were developed and tested, the next step was to design a Splunk dashboard that would visualize the data in a clear and accessible way for the Navitas team. The dashboard serves as a central place where the team can observe the system's current status, identify trends, and react quickly to unusual behavior or recurring issues. The design work began by returning to the list of requirements that had been defined in collaboration with the Navitas team earlier in the project. These included five key areas that the team wanted to visualize from the start:

- Failed Data Retrieval from Adapter: Show adapter name (Adapter) and reference key (ReferenceKey) with a line graph displaying counts per adapter.
- ERROR-Level events: Display a line graph showing error counts per adapter.
- ERROR-Level events by Error Type: Show a table listing error types and their count (default view: past 24 hours).
- Failed Logins: Categorized by login type (username/password, VRK card, desktop integration), shown in a table with event counts.
- Successful Logins: A table structured similarly to the failed logins view, showing login type and count.

To begin building the dashboard, I used Splunk's new Dashboard Studio, which provides a modern and more user-friendly interface for creating dashboards - especially helpful for users who are new to Splunk visualizations. I named the dashboard "Navitas Valvontataulu Test", as it was created for the test environment. Before adding the first visualizations, I spent some time reviewing Splunk's official documentation for Dashboard Studio. I discovered that while Dashboard Studio is more intuitive, it currently lacks support for some advanced functionality found in Classic Dashboards. However, for the initial monitoring needs of the Navitas team, these limitations were not an issue.

I began the actual implementation by creating the first graph for failed login attempts. Using the SPL query I had already developed for this purpose, I created a time-based line chart. To make the dashboard more interactive, I added a dropdown menu that allows users to select a time range for the data shown. This dropdown sets a token named

“$dd_timespan$”, with options to view data from either the past 7 days or the past 30 days.

Once the failed login visualization was completed, I moved on to the next one—successful logins. The process was very similar: I added a new graph and configured a new data source named "Onnistuneet kirjautumiset" (Finnish for “Successful Logins”). I applied the corresponding SPL query, then customized the appearance by setting the X-axis title to "Päivämäärä" (date) and the Y-axis title to "kirjautumiset" (logins), ensuring that the visual was both functional and localized for its users.

The next feature I worked on was the failed data retrievals from adapters, which required both a graphical overview and a list/table view. Again, I added two new visualizations—one graph and one table—and created new data sources for each by inserting the relevant SPL queries. I configured both views to display data by adapter and reference key, giving the team the ability to drill down into which adapter experienced the error and the associated reference.

As more panels were added to the dashboard, I took care to organize the layout so that it remained easy to navigate. For example, graphs and tables related to the same data (e.g., adapter errors) were placed side by side in two-column rows. This design choice helps users easily connect different views of the same dataset and quickly switch between high-level trends and detailed lists.

Throughout the design process, care was taken to ensure consistency and usability. The time range filters, queries, and host exclusions were applied uniformly across all visualizations. Additionally, dashboard performance was kept in mind by optimizing the queries and limiting the number of events returned when necessary.

At this stage, the dashboard contains all the essential visualizations defined in the project’s initial scope. It provides a solid foundation for monitoring, and it is already being used by the Navitas team in the test environment. The dashboard will likely evolve based on future feedback, and potential improvements may include enhanced visual indicators,

more advanced filtering options, or the addition of alerting functionality as the system transitions into production use.

After all the requested graphs and lists were added to the dashboard, I implemented a drilldown function to improve usability. This feature allows users to click on a specific data point in a graph or table and be redirected to the underlying SPL search. This makes it easier to inspect the raw events and, if necessary, further refine the query - for example, to investigate one particular adapter or narrow down the search to a specific time window or error type. This interactive element significantly improves the user experience and supports in-depth analysis directly from the dashboard.

Once the dashboard structure and functionalities were completed, I encountered a permission-related limitation: the dashboard was saved as private by default, and I did not have the necessary rights to make it publicly accessible to the whole Navitas team. To solve this, I contacted a member of the security team. Together, we agreed that he would create a new public dashboard instance. I then exported the full source code of my dashboard - Splunk provides an option to view the entire dashboard as a single code file—and sent it to him. He replaced the default content in the new dashboard with this code, effectively cloning my version into the public space. This process was quick and efficient, thanks to Splunk's flexibility in managing dashboard configurations through code.

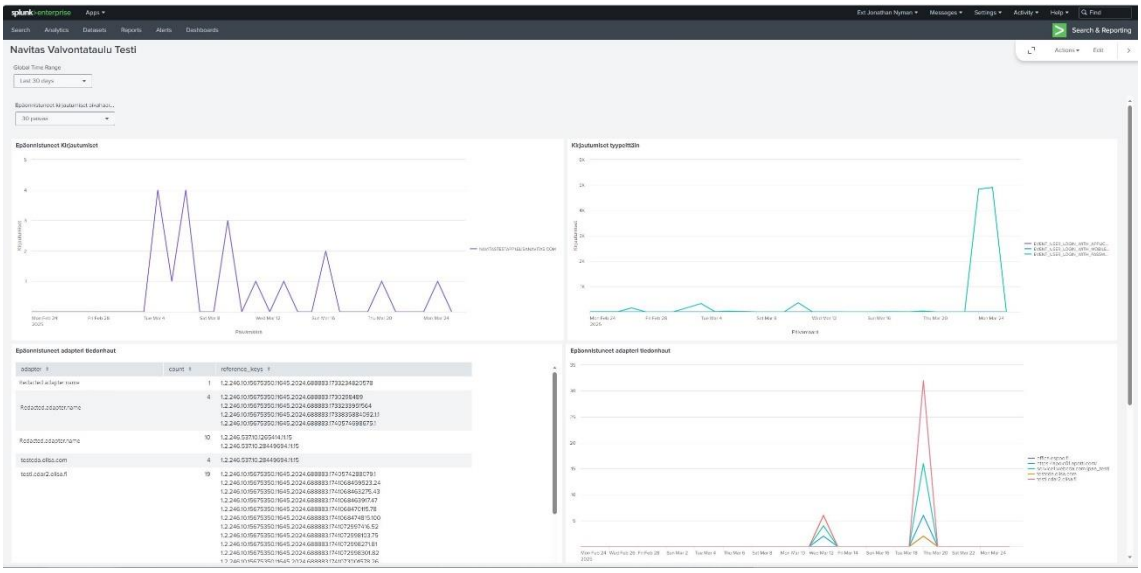

Figure 30, Completed dashboard 1

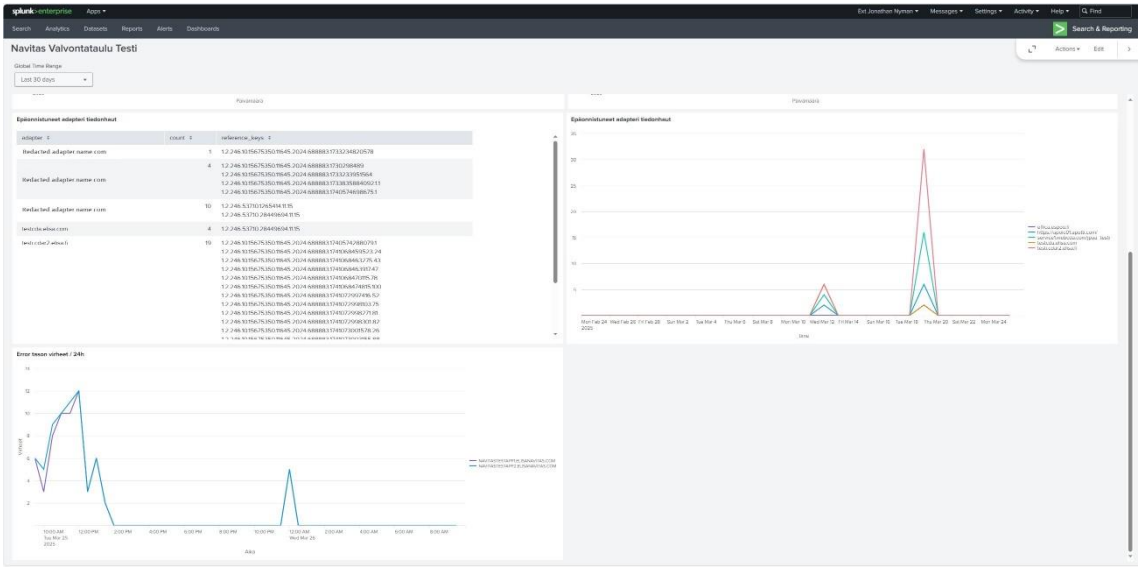

Figure 31, Completed dashboard 1

5.5 Validation & Testing

5.5.1 Unit Testing: Checking Data Ingestion Accuracy

The first step in validation was to check that logs were being correctly ingested from the Navitas servers into the Splunk index. This involved confirming that forwarders were installed and running on all necessary servers, and that log data appeared in the expected index (Monttu_navitas_test). I manually verified that the logs were being updated in real time and that the correct log sources were active. I also checked that field extractions using rex commands (such as Adapter and ReferenceKey) were producing consistent results and values were appearing as expected in queries.

5.5.2 Performance Testing: Measuring Query Execution Speed

To avoid performance issues, especially given that Monttu is a shared Splunk environment, the queries used in the dashboard were reviewed to ensure that they executed efficiently. I followed Splunk best practices, such as setting the time range early, using specific indexes and sourcetypes, and applying filters like host!=NAVITASTESTAPP2 and date ranges to minimize the data being scanned. I also tested how fast the queries returned results and confirmed that even with a longer timespan (e.g., 30 days), the queries performed well and did not overload the system if the correct query parameters were used.

If for example searching 30 days back for all events the query took a lot longer to complete. This just proves how important it is to use the correct query parameters.

5.5.3 User Testing: Collecting feedback from the Navitas Team

After the first version of the dashboard was completed, I invited the Navitas team to try it out and give feedback. During the session, we reviewed the different graphs and tables, and the team was able to click through the views and run the dropdown time filters. Based on this feedback, I made minor adjustments, such as updating axis labels to be in Finnish (for example, renaming labels to *päivämäärä* and *kirjautumiset*) and reordering the layout to better group related visualizations. Overall, the feedback was positive, and the dashboard was considered intuitive and useful for day-to-day monitoring.

6 Challenges, Solutions & Lessons Learned

Throughout the course of the project, several challenges were encountered that affected both the timeline and the overall execution. These challenges can be grouped into two main categories: technical and organizational. Overcoming them required collaboration, persistence, and a flexible mindset. This section describes the most significant issues faced, how they were handled, and what lessons were learned.

6.1 Technical Challenges

One of the most complex technical challenges arose from how confidential data was handled in Navitas log files, especially in DEBUG-level logs. While most of the log data forwarded to Splunk could be masked and filtered in advance, some errors and debugging details required access to unmasked data for proper troubleshooting. The original idea was to use Splunk's masking capabilities to filter out sensitive fields before they became visible to users. However, we discovered that in Splunk, masking occurs after the data has already been forwarded and indexed, meaning that sensitive data would still technically exist in the system even if it wasn't shown to users.

This was a critical problem, particularly because the Navitas service handles healthcare-related information and social security numbers, which must not be processed outside the

permitted scope. The final solution was to reroute DEBUG-level logs to a separate log file, which is not sent to Splunk at all. This approach made sure that essential DEBUG data was still available locally when needed, while ensuring that only masked and compliant log levels (INFO, ERROR, WARN) were forwarded to Monttu.

Another technical limitation was related to permissions and administrative access. The Splunk instance Monttu is centrally managed, and only designated administrators, mostly from the security team, can modify deployment server configurations or create new inputs. As a project contributor, I didn't have admin rights in Monttu, which meant that tasks like configuring forwarders or setting up new indexes required support from security personnel. While the Splunk UI is powerful, its complexity and access restrictions meant that some seemingly simple steps took longer due to the required handoffs between teams.

6.2 Organizational Challenges

From an organizational perspective, the biggest challenge was how time-consuming the project turned out to be. The project started with a very basic question: "Does Elisa already have a Splunk instance that can be used?" It took considerable time and coordination just to confirm that Monttu could be used for this project, and to get the necessary approvals. After that, the work continued in small increments, as the project was not a top priority for any of the teams involved. This meant that progress often depended on short weekly meetings, with only a few people available at one time.

The issue with DEBUG-level log handling alone required multiple cross-team meetings, with input from developers, the security team, and infrastructure specialists. Aligning everyone's schedules and decisions extended the timeline significantly. Additionally, the need to balance strict data protection policies with practical debugging requirements made the process more complex than expected.

6.3 Lessons Learned During Implementation

One of the key lessons from this project was the importance of clearly defining access rights and data policies before setting up any monitoring infrastructure. The sensitivity of the Navitas logs meant that even technically simple tasks, like sending logs to Splunk, required thorough security reviews and sometimes workarounds.

Another important takeaway was the value of having documentation and administrative support readily available. Since the Splunk environment was managed by another team, a lot of time could have been saved if more documentation had been available on how the forwarder setup and Monttu configuration process worked in practice.

It also became clear that organizational alignment is just as important as technical execution. Even though the dashboard and queries themselves were not overly complex, the coordination effort - getting everyone's input, ensuring compliance, waiting for approvals ended up being one of the most resource-intensive parts of the project.

Despite these challenges, the project succeeded in delivering a working and secure Splunk-based monitoring solution for the Navitas team. The experience gained will make future iterations of the system smoother and better aligned with both technical and regulatory requirements.

7 Results & Impact

Despite the various technical and organizational challenges encountered along the way, the project achieved its main goal: to build a working Splunk-based monitoring dashboard for the Navitas service that enables centralized visibility into log data and system behavior.

One of the most immediate and practical outcomes is that the manual log-checking process previously used by administrators has now been significantly improved. Instead of logging into multiple servers and manually running scripts to troubleshoot issues, the Navitas team can now view system activity, adapter-related errors, and login statistics

directly in the dashboard. This has reduced the need for ad hoc investigations and made the process of identifying issues faster and more efficient.

In addition to this, the team now has access to historical log data visualized in an accessible format. This makes it easier to detect recurring patterns, such as frequent login failures or adapter-specific problems, which might have previously gone unnoticed. The dashboard supports time-based filtering and includes drill-down functionality, so team members can move from a high-level overview to raw log details within a few clicks.

From a security perspective, the implementation also had a positive impact. One of the key concerns in the past was the lack of visibility into authentication issues. Now, failed and successful login attempts are tracked and categorized, making it easier to spot abnormal behavior or potential misuse of credentials.

The project also helped clarify and improve how sensitive data is handled within the Navitas environment. The decision to exclude DEBUG-level logs from Splunk and route them to a separate local file helped maintain security compliance without compromising the team's ability to perform deeper analysis when needed.

Although the project was originally limited to the test environment, it has laid a solid foundation for expanding the dashboard into production in the future. The queries and visualizations can easily be reused or adapted, and the lessons learned during the implementation will help make future deployments more efficient. Furthermore, now that the dashboard is in use, the Navitas team has started thinking about new use cases, such as alerting, anomaly detection, or monitoring additional subsystems indicating that the system has real long-term value.

In summary, the Splunk dashboard has improved operational visibility, reduced manual work, enhanced security monitoring, and established a platform that can continue to grow with the needs of the Navitas service.

8 Future Work

8.1 Expanding Monitoring Scope

One of the most natural next steps is to expand the monitoring scope beyond the initial test environment. As the dashboard has proven to be useful and effective, it could be extended to cover production servers and additional services related to Navitas. This would allow the team to track a broader set of metrics and gain a more complete overview of the system's real-time and historical behavior.

There is also potential to include new log sources, such as logs from supporting systems, integrations, or security appliances. Expanding the types of logs being ingested into Splunk would provide a more comprehensive view of system health and make it easier to correlate issues that span across multiple systems. This would also improve the team's ability to respond to incidents and monitor end-to-end service delivery.

8.2 Machine Learning for Anomaly Detection

While the current dashboard is based on predefined queries and visualizations, Splunk also supports machine learning (ML) capabilities that could be used to enhance monitoring. By leveraging tools such as the Splunk Machine Learning Toolkit (MLTK), the system could learn from past data and automatically detect patterns that indicate unusual or potentially problematic behavior.

For example, ML models could be trained to identify anomalies in login patterns, unexpected spikes in error messages, or usage outside of normal working hours. These models could then trigger alerts or provide suggestions for further investigation. Implementing machine learning would move the system from a reactive tool to a more proactive one, helping the team detect issues before they escalate.

8.3 Enhancing Automation & Log Retention

Another area for future development is enhancing automation and log lifecycle management. While Splunk can already automate alerts and actions based on search results, these

features have not yet been implemented in the current version of the dashboard. Introducing automated alerts for sudden increases in failed logins or repeated adapter errors would improve the team's ability to respond quickly without needing to constantly monitor the dashboard manually.

In addition, it would be useful to review and implement a clear log retention strategy. Since log data can grow rapidly and storage costs may become a concern, defining how long different types of logs are retained and whether they are archived or deleted is an important next step. This would ensure compliance with data governance policies and help maintain performance in the long term.

9 Conclusion

This project set out to improve the monitoring and troubleshooting capabilities of the Navitas service by designing and implementing a centralized log monitoring dashboard using Splunk. Through a combination of planning, technical implementation, and cross-team collaboration, the goal was successfully achieved. The new dashboard now provides the Navitas team with real-time visibility into key system events, including adapter failures, ERROR-level logs, and login activity.

Before this implementation, log monitoring was done manually across multiple servers, which was both time-consuming and error prone. The new Splunk-based solution has significantly improved the team's ability to detect problems, analyze historical trends, and act proactively. Thanks to structured SPL queries, dynamic visualizations, and user-defined filters, the dashboard has become an accessible and practical tool for everyday operations.

Along the way, the project encountered several technical and organizational challenges—ranging from handling sensitive DEBUG-level log data to coordinating with different teams under strict access limitations. These challenges required thoughtful solutions and highlighted the importance of secure data management, documentation, and communication between departments. Despite the delays caused by these issues, the end result was

a robust system that respects both performance requirements and data security constraints.

The project has also created a strong foundation for future development. There are clear opportunities to expand monitoring into production environments, incorporate machine learning for anomaly detection, and introduce alerting and automation features that can make the system even more proactive. The lessons learned during this implementation will be invaluable for future monitoring projects within Elisa and beyond.

In conclusion, this work has demonstrated how centralized log management using Splunk can transform reactive troubleshooting into proactive system oversight. The result is not only a technical improvement but also a shift in how system health is understood, tracked, and managed within the organization.

10 Sammanfattning på Svenska

10.1 Syfte med projektet

Syftet med detta slutarbete är att designa och implementera en Övervakningstavla i Splunk för Navitas-tjänsten på Elisa. Projektet har som mål att förbättra övervakningen av systemets loggar genom att möjliggöra realtidsinsyn i fel, historisk analys och visualisering av nyckeldata. Det nuvarande arbetssättet bygger på manuell logghantering, vilket är tidskrävande och ineffektivt. Genom att automatisera och centralisera loggövervakningen med hjälp av Splunk skapas en lösning som stödjer snabbare felsökning, förbättrad säkerhet och en mer proaktiv IT-drift.

10.1.1 Navitas-tjänsten och behovet av förbättrad loggövervakning

Navitas är en kritisk tjänst inom Elisa som ansvarar för överföring av känsliga patientdata mellan olika hälsovårdsdistrikt. Systemets stabilitet och säkerhet är därför avgörande. I dagsläget övervakas loggar manuellt från flera servrar, vilket försvårar upptäckten av felmönster och försenar insatser vid incidenter. Bristen på centraliserad övervakning innebär också att historisk analys av systemets hälsa inte är möjlig, vilket leder till att felsökning ofta sker reaktivt. Genom att införa en Splunk-dashboard kan Navitas-teamet få en tydlig överblick över systemets tillstånd och snabbt identifiera problemområden.

10.2 Bakgrund och forskningsöversikt

I takt med att moderna IT-system blir alltmer komplexa ökar också kraven på effektiv övervakning och logghantering. För organisationer som hanterar känslig information, såsom personuppgifter inom hälsovården, är behovet av robusta och säkra övervakningslösningar särskilt stort. Navitas är en viktig tjänst inom Elisa som ansvarar för dataöverföring mellan olika sjukvårdsdistrikt. Eftersom tjänsten är kritisk för flera hälso- och sjukvårdssystem är det av yttersta vikt att eventuella fel upptäcks snabbt och kan analyseras effektivt. Tidigare har detta arbete utförts manuellt, vilket medför stora risker för att fel inte upptäcks i tid eller att historiska mönster går obemärkta förbi.

Det här examensarbetet undersöker hur en automatiserad övervakningslösning baserad på plattformen Splunk kan implementeras i Navitas-tjänsten. Splunk är ett av marknadens mest använda verktyg för logghantering och realtidsanalys och används idag inom både privata och offentliga sektorer för att förbättra systemövervakning, felspårning och säkerhet.

Litteraturstudier och tidigare forskning inom området visar att automatiserad loggövervakning inte bara förbättrar felsökningskapacitet, utan också minskar risken för systemavbrott och förbättrar cybersäkerheten. Ett viktigt koncept i modern logghantering är realtidsanalys, där insamlad data bearbetas omedelbart och gör det möjligt att agera proaktivt på incidenter. Splunk möjliggör detta genom sin kraftfulla sökmotor och stöd för användardefinierade övervakningstavlor. En annan viktig aspekt är säkerhets- och åtkomstkontroll. I miljöer där personuppgifter behandlas måste lösningen uppfylla strikta regulatoriska krav. Splunk har inbyggda funktioner som stödjer rollbaserad åtkomstkontroll (RBAC) och maskering av känslig information.

För att stärka förståelsen för ämnet har flera vetenskapliga källor och tidigare lärdomsprov studerats, med fokus på nyckelord som "automatiserad logghantering", "realtidsövervakning", och "Splunk systemintegration". Bland de mest relevanta källorna finns ett examensarbete från Arcada (Mäkelä, 2020) samt dokumentation från Splunk (Splunk, n.d.). Dessa källor har gett värdefull vägledning kring tekniska lösningar, indexeringsstrategier och användning av maskininlärning inom loggövervakning.

10.2.1 Metodansats

Detta examensarbete är ett praktiskt utvecklingsprojekt som kombinerar teknisk implementering med ett undersökande arbetssätt. Arbetet syftar till att ta fram en fungerande Splunk-baserad övervakningstavla för Navitas-systemet och samtidigt analysera hur en sådan lösning påverkar systemets övervakningskapacitet och effektivitet. Projektet bygger på en iterativ och agil arbetsmetodik, där kravspecifikation, utveckling, testning och förbättringar genomförs i flera cykler. Målet har varit att stegvis bygga upp en lösning som är tekniskt hållbar och samtidigt användbar för Navitas-teamet.

10.2.2 Datainsamling och logganalys

Loggdata har samlats in via Splunk Universal Forwarders som installerats på Navitas olika servrar. Dessa forwarders skickar loggar till Elisas centrala Splunk-server, kallad Monttu, där loggarna indexeras och analyseras. Valet av vilka loggfiler som skulle övervakas fattades i samråd med Navitas-teamet, baserat på tidigare incidenter och behov av insyn i specifika systemdelar.

Med hjälp av Splunks inbyggda sökspråk, SPL (Search Processing Language), analyserades loggarna för att identifiera fel, säkerhetsincidenter och återkommande problem. För att säkerställa resultatens tillförlitlighet användes flera olika testmiljöer och loggdata från både test- och produktionssystem. Resultaten jämfördes med tidigare manuella arbetsprocesser för att kunna utvärdera förbättringar i felsökningstid och upptäcktsgrad.

10.2.3 Etiska och säkerhetsrelaterade överväganden

Eftersom Navitas hanterar känslig patientinformation har säkerhetsaspekten varit central under hela projektets gång. Det har varit nödvändigt att noggrant definiera vilka loggar som får skickas till Splunk, samt se till att dessa är korrekt maskerade så att inga personuppgifter kan ses i det centraliserade systemet. Rollbaserad åtkomst (RBAC) har implementerats för att säkerställa att endast behöriga användare kan se och hantera viss typ av information. Alla tester har genomförts i enlighet med Elisas interna säkerhetspolicyer och gällande lagstiftning för dataskydd.

10.3 Planering och Implementering

10.3.1 Planering

Projektet inleddes med att kartlägga Elisas befintliga Splunk-infrastruktur. Det visade sig att företaget redan hade en Splunk-server kallad Monttu, vilket innebar att det inte behövdes byggas en ny miljö från grunden. En egen instans för Navitas skapades i Monttu, vilket gjorde det möjligt att isolera loggdata från andra team och system.

Ett stort fokus i uppstartsfasen låg på att definiera tekniska krav, diskutera säkerhetsaspekter och identifiera de mest relevanta loggarna. Samtidigt upprättades kontakt med ansvariga för Splunk-servern samt Navitas utvecklingsteam för att samordna rättigheter och resurser. Projektet krävde mycket samarbete mellan olika team, vilket ibland fördröjde arbetet då det inte var någon parts huvudsakliga uppgift.

10.3.2 Installation av Splunk Forwarders

För att samla in loggar installerades Splunk Universal Forwarders på samtliga relevanta servrar i både test- och produktionsmiljö. Installationen baserades på en detaljerad instruktion framtagen av Navitas-teamet, där IP-adresser och loggsökvägar specificerades. På Linux-servrar användes .rpm-paket medan Windows-servrar installerades med .msi-paket. Installationen innefattade konfiguration av forwarders att skicka data till rätt index i Monttu samt definiering av vilka loggfiler som skulle inkluderas.

Särskild vikt lades vid att säkerställa att känsliga DEBUG-loggar inte skickades till Splunk, då dessa ibland innehåller personuppgifter. Lösningen blev att skicka DEBUG-nivåloggar till en separat loggfil som inte skickas till Splunk, vilket löste problemet utan att kompromissa med möjligheten att felsöka.

10.3.3 Utveckling av sökfrågor i Splunk

Efter att ”Splunk-forwarders” installerats och loggarna börjat strömma in till Monttu-indexet monttu_navitas_test, påbörjades arbetet med att utveckla sökfrågor i Splunk för att identifiera relevanta händelser och fel. Detta gjordes i nära dialog med Navitas-teamet, som specificerade vilka loggtyper och fel som var viktiga att visualisera i övervakningstavlan.

En introduktion till Splunks sökspråk (SPL – Search Processing Language) hölls med stöd från två Splunk-experter. Under denna workshop gick man igenom grunderna i hur SPL fungerar, hur man skriver effektiva sökningar och vilka funktioner som lämpar sig

för olika typer av analyser. Deltagarna fick även testa grundläggande sökningar för att själva bekanta sig med verktyget.

Arbetet med att bygga SPL-frågor följde Splunks rekommenderade best practices:

1. Först specificeras tidsintervall – till exempel de senaste 24 timmarna eller 7 dagarna.
2. Därefter anges vilket index som ska genomsökas – i detta fall index=monttu_navitas_test.
3. Slutligen läggs ytterligare filter, fält och visualiseringskommandon till – exempelvis rex, stats, timechart, eller table.

10.3.4 Design av övervakningstavlan i ”Splunk Dashboard Studio”

Övervakningstavlan byggdes i Splunk Dashboard Studio, som erbjuder ett mer användarvänligt gränssnitt jämfört med det tidigare ”Dashboard” - verktyget. Övervakningstavlan fick namnet - Navitas Valvontataulu Test, eftersom det är övervakningstavlan för testmiljön och designades i nära samarbete med Navitas-teamet.

Visualiseringar inkluderade linjediagram, stapeldiagram och tabeller som visar bland annat adapterfel, inloggningsstatistik och felfrekvens över tid. En ”dropdown”-meny lades till för att kunna växla mellan 7 och 30 dagars data. En ”drilldown” -funktion implementerades också, så att användaren kan klicka på en graf och direkt föras till den bakomliggande SPL-sökningen för djupare analys.

Eftersom vanliga användare inte hade behörighet att publicera övervakningstavlan, skickades källkoden från övervakningstavlan till en administratör som lade in den i en officiell, offentlig dashboard.

10.4 Utmaningar och lärdomar

10.4.1 Tekniska utmaningar

Ett av de största tekniska hindren i projektet var hanteringen av känsliga loggdata. Navitas-tjänsten loggar mycket detaljerad information, och vissa loggar – särskilt på DEBUG-

nivå – innehåller personuppgifter som inte får lämna systemet. Eftersom Monttu används av flera team inom Elisa, var det kritiskt att säkerställa att inga omaskerade känsliga data skickades till Splunk-instansen.

Initialt övervägdes att maskera känslig information direkt i Splunk Forwarder-konfigurationen. Det visade sig dock att denna maskning sker först efter att loggarna skickats och indexerats i Monttu, vilket alltså inte löste problemet. Den slutgiltiga lösningen blev därför att konfigurera systemet så att DEBUG-nivåns loggar skrivs till en separat loggfil som inte skickas till Splunk. Denna lösning krävde samarbete med både utvecklingsteamet och säkerhetsteamet, samt anpassning av loggstrukturen i applikationen.

Även Splunks komplexitet i sig utgjorde en teknisk tröskel. Eftersom Splunk kräver en viss kompetens för installation och hantering, krävdes det att personer med administratörsbehörighet (från säkerhetsteamet) bistod med konfigurering av forwarders, index och dashboard-behörigheter. Som extern projektresurs hade jag inte rättigheter att själv utföra dessa åtgärder.

10.4.2 Organisatoriska utmaningar

Projektet var beroende av samarbete mellan flera olika team: Navitas-utvecklingsteamet, säkerhetsteamet (ansvariga för Splunk Monttu) och andra stödfunktioner inom Elisa. En betydande utmaning var att detta projekt inte var någon parts huvudansvar, vilket ledde till begränsad tillgänglighet och långsam beslutsprocess. Oftast kunde endast ett möte per vecka hållas, vilket påverkade tempot i arbetet.

Det krävdes också tid för att kartlägga vem som ansvarade för olika delar – till exempel nätverksåtkomst till Monttu, AD-behörigheter för övervakningstavlan och godkännande för loggöverföring. Kommunikation mellan olika team var ibland svår på grund av skilda prioriteringar och tekniska förståelser.

10.4.3 Lärdomar

En viktig lärdom från detta arbete är vikten av tydlig initial planering, särskilt i projekt som involverar flera team och säkerhetskritiska data. Att tidigt identifiera rätt kontaktpersoner, definiera ansvarsområden och sätta realistiska tidsramar är avgörande.

Projektet gav även praktisk erfarenhet i Splunk och dess SPL-språk – både vad gäller logganalys, prestandaoptimering av sökningar och övervakningstavlans design. Jag lärde mig också hur Splunk hanterar dataflöden från ”forwarder” till ”indexer” och hur man felsöker problem med datainhämtning.

Sammanfattningsvis krävde projektet både teknisk kompetens och god samarbetsförmåga. Trots de utmaningar som uppstod kunde projektets mål uppnås tack vare ett strukturerat angreppssätt och kontinuerlig dialog med de involverade parterna.

10.5 Resultat och framtida arbete

10.5.1 Projektets resultat och påverkan

Det huvudsakliga resultatet av projektet var att en fungerande Splunk-dashboard togs fram för Navitas testmiljö. Övervakningstavlan visar loggdata i realtid och gör det möjligt för teamet att enkelt övervaka tjänstens status och identifiera fel, utan att manuellt behöva logga in på flera olika servrar. Detta markerar ett stort steg från det tidigare manuella logghanteringssystemet, som var både tidskrävande och bristfälligt i fråga om överblick.

Övervakningstavlan innehåller visualiseringar och tabeller som visar:

- Misslyckade datahämtningar från adapterkomponenter
- Felmeddelanden på ERROR-nivå, med klassificering efter typ
- Lyckade och misslyckade inloggningar, uppdelade på autentiseringstyp

Alla dessa komponenter är byggda utifrån teamets önskemål och validerades i nära samarbete med dem. Funktioner som drill-down (för att kunna klicka på ett diagram och gräva djupare i loggdata) samt en valbar tidsperiod (t.ex. senaste 7 eller 30 dagar) bidrar till att göra övervakningstavlan flexibel och användbar. Projektet har också resulterat i en tydlig

dokumentation över hur Splunk Forwarders installeras och konfigureras i Navitas-miljön, vilket underlättar framtida implementationer.

10.5.2 Möjligt framtida arbete

Eftersom detta projekt primärt fokuserade på testmiljön, är nästa naturliga steg att rulla ut övervakningstavlan i produktionsmiljön. Detta kräver att Splunk Forwarders installeras och konfigureras på produktionsservrarna, samt att säkerhetskraven uppfylls för denna miljö.

Andra framtida förbättringsområden innefattar:

- Utökad övervakning: Övervakninstavlan kan kompletteras med ytterligare loggtyper och applikationer.
- Maskininlärning: Splunks stöd för ”Machine Learning Toolkits” kan användas för att automatiskt upptäcka avvikelser i loggmönster och förutse fel innan de inträffar.
- Incidenthantering: Genom att aktivera Splunks inbyggda notifieringssystem kan systemet skicka varningar via e-post eller andra kanaler vid kritiska händelser.
- Loggarkivering: Fastän detta projekt inte omfattade långsiktig loggförvaring, kan framtida arbete inkludera strategi för retention och compliance-arkivering.

Projektet har lagt en solid grund för ett mer proaktivt och datadrivet sätt att hantera driftövervakning inom Navitas-tjänsten. Erfarenheterna och de verktyg som tagits fram kan återanvändas i andra delar av organisationen som har liknande behov.

11 References

Datadog. (n.d.). *Cloud Monitoring as a Service*. https://www.datadoghq.com/

Elastic. (n.d.). *Elastic Stack: Elasticsearch, Kibana, Beats, and Logstash*. https://www.elastic.co/elastic-stack

Korhonen, M. (2020). *Implementing log management and monitoring system with Splunk* [Bachelor's thesis, Haaga-Helia University of Applied Sciences]. Theseus. https://www.theseus.fi/handle/10024/333929

Splunk Inc. (n.d.). *What is Splunk?*. Splunk Docs. https://www.splunk.com/en_us/blog/learn/what-splunk-does.html