ROBERT GORDON UNIVERSITY ABERDEEN

SCHOOL OF COMPUTING

MSc Project Report

Student Name: Israel Blankson Effiong	Matriculation Number:
Supervisor: Dr. Christopher McDermott	Second Marker: Dr. Niccolo Capanni
Course: MSc. Cyber Security	
Project Title: Integrating Security into DevOps Methodology using the 'shift left' approach via threat modelling.	
Start Date: 06/02/2023	Submission Date: 01/05/2023

CONSENT

I agree

That the University shall be entitled to use any results, materials or other outcomes arising from my project work for the purposes of non-commercial teaching and research, including collaboration.

DECLARATION

I confirm:

- **That the work contained in this document has been composed solely by myself and that I have not made use of any unauthorised assistance.**
- **That the work has not been accepted in any previous application for a degree.**
- **All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.**

Student Signature: ISRAEL BLANKSON EFFIONG	Date Signed: 01/05/2023

Acknowledgements

I want to thank my friends and especially my family for the opportunities they opened for me and for the unwavering support throughout the years. Without my family, I would not have a support system. Their love and understanding have made me get through the most difficult of obstacles.

I also want to thank Stephen Olatunji, my team lead at work for serving as a second set of eyes and being ever present when I needed his help with volunteers, corrections and proofreading of my reports. He was never wary of my constant discussions about the project and always encouraged me in the best way he could.

Finally, I want to thank my project supervisor, Dr. Christopher McDermott, for his guidance and help during the course of conducting this entire project, from the get-go to the implementation and to writing of this report. He has helped point me in the right directions and offer advice when needed. He has also been very understanding of my situation.

Thanks once again.

Abstract

As software development practices evolve towards faster and more agile methodologies, such as DevOps, security has become a critical concern. Traditional security measures, implemented late in the development cycle, often lead to vulnerabilities being identified late in the process, resulting in increased costs and risks. The 'Shift Left' approach, which emphasizes integrating security earlier in the development process, has gained traction as a solution to address this challenge.

The integration of security into the DevOps methodology using the 'Shift Left' approach via threat modelling was explored in this research. Threat modelling is a proactive technique that involves identifying potential security threats and vulnerabilities at the design and development stages of the software development lifecycle (SDLC). By identifying and mitigating security risks early in the SDLC, the 'Shift Left' approach aims to prevent security issues from becoming critical risks in production.

Starting off with an in-depth review of the DevOps methodology, the project also explored its principles, practices, and benefits. Next, the concept of 'Shift Left' is examined, highlighting its key principles and how it aligns with DevOps practices. The project then delves into the theory and practical implementation of threat modelling, including different threat modelling techniques and their applicability to DevOps environments.

The study also includes a case study, where a practical implementation of the 'Shift Left' approach via threat modelling is conducted in a real-world DevOps environment. The case study involves conducting threat modelling exercises at various stages of the SDLC, integrating security controls and processes into the DevOps pipeline, and measuring the impact of the 'Shift Left' approach on the overall security posture of the software product.

The findings of this research project are expected to contribute to the body of knowledge on integrating security into DevOps methodologies using the 'Shift Left' approach via threat modelling. The results of the case study will provide insights into the practical challenges and benefits of implementing this approach, and recommendations for organizations seeking to enhance their security posture in DevOps environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

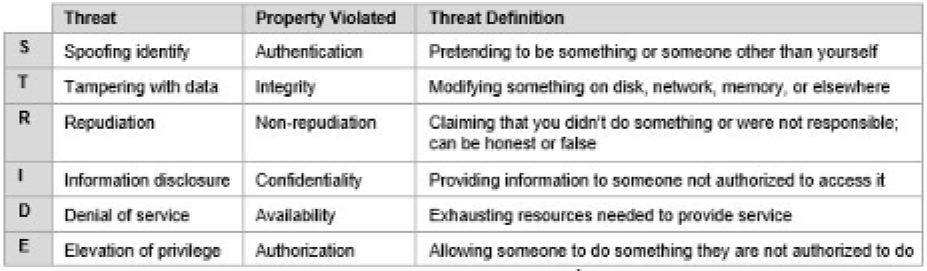
Several businesses are understandably worried about the vulnerability of their software systems to outside attacks. According to a poll conducted by Ernst & Young (2019), over half (53%) of businesses have raised their cybersecurity expenditure in the past year, and another 65% plan to do so in the following year. As a result of new data privacy laws (GDPR) and forthcoming security standards, expenditures on security have increased (e.g., ISO 21434). Regulatory conformity has been demonstrated to be an important driver of security spending (Cavusoglu et al. 2015; Moore and Dynes 2016). For instance, Cavusoglu et al. (2015) looked at how institutional pressures affect the purchase of securities. The authors conclude that coercive pressure (evidenced by perceived pressure from business partners and pressure from legislation) is one of the primary drivers affecting investments in security resources, alongside normative pressure (evidenced by the state-of-practice) (Tuma et al. 2021).

The practice of threat modelling is very significant in software security, as a lot of security vulnerabilities are generated by architectural design defects (Viega and McGraw 2006). In addition, resolving such flaws after deployment can be extremely expensive, necessitating workarounds that occasionally increase the attack surface. An explicit threat model aids in the identification of threats to a system's various assets by making well-grounded assumptions about the abilities of any attacker with an intent in exploiting the system. It also allows development teams to pinpoint crucial designs that must be safeguarded, as well as mitigation techniques. Yet, threat modelling can be difficult for developers to accomplish, especially in agile software development (Cruzes et al. 2021).

Adoption of security practices in agile software development presents unique problems, most commonly due to a lack of priority given to security activities or a failure on the part of practitioners to recognize the significance of such activities in bolstering project security (Whiting and Datta 2021). The same can be said of threat modelling: it is not widely used, and agile developers have limited resources to turn to for advice on incorporating it into their workflow (Viega and McGraw 2006; Tuma, Calikli and Scandariato 2018; Cruzes et al. 2021). Research on how to integrate threat modelling into software development methods that do not depend on having up-front architecture for their solutions is still lacking, despite the efforts of researchers. When it comes to software security research, most articles cover broad topics like policies and procedures, but there are hardly any empirical studies that zero in on specific practices in agile software development.

This study's overarching goal is to learn how to effectively use the STRIDE methodology and threat modelling to incorporate application security into agile software development operations. A threat modelling tool known as CAIRIS will also be used for documentation.

STRIDE threat modelling is the pinnacle of the shift-left strategy (Hewko 2021). It may be used to discover and eliminate possible vulnerabilities prior to writing any code. Using threat modelling approaches should be the initial step in developing secure-by-design networks, systems, and applications. STRIDE is a threat model that may be used as a framework to ensure application security (Hewko 2021). CAIRIS is an abbreviation for Computer Aided Integration of Requirements and Information Security. It is a free and open-source software environment for gathering requirements for functional and trustworthy software. Usability, needs, and risk analysis are all integrated into its structure (CAIRIS 2023).



	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

***Figure 1.1**: STRIDE Approach to Threat Modelling (Hewko 2021)*

1.2 PROBLEM STATEMENT

With the advent of new software development methodologies (such Agile and DevOps), the development lifecycle has shrunk, leaving less time for thorough security research of programs' architecture (Khan 2017). Namely, architectural models are (usually manually) examined for vulnerabilities. Risk-last threat analysis recommends cataloguing potential dangers before ranking them in severity (Tuma et al. 2021). Instead of prioritizing threats, the risk-first approach recommends detecting potential dangers first. The implications here appear positive for businesses where rapid innovation is a priority. Yet, there is a lack of data on how threat analysis is affected when systematicity is compromised in favor of high-priority threats.

1.3 MOTIVATION

Software and computer systems that aren't well-protected might cause a company financial hardship for many reasons. IBM found that 83% of 550 organizations across 17 countries and 17 industries had multiple data breaches between March 2021 and March 2022 (Cruzes et al. 2021). According to (Cruzes et al. 2021), over the aforementioned time frame, the average cost of a data breach was $4.35 million. There is a monetary penalty to data breaches, but also uncovered attack routes, such as third-party software flaws (Cruzes et al. 2021). The findings described in are only one illustration of why software development and deployment processes need to be more secure (Khan 2017).

Relatedly, the difficulties in choosing and employing security technologies in DevOps have been elucidated by the work of Rajapakse et al. (2022). This study hypothesizes that the responses to the posed research questions will shed light on how best to choose and implement security tools for use with the DevOps methodology. The findings of this study will hopefully be put to use by academics, industry professionals, and security tool creators in the context of DevOps and maybe related approaches to further enhance the safety of software deployed in the real world.

PERSONAL MOTIVATION

As a Cyber security student and a professional DevOps Engineer, I am highly interested in the intersection between software development and security. DevOps methodology is becoming increasingly popular in software development, and I believe that integrating security into this methodology is crucial for ensuring that software is secure and resilient to cyber-attacks. By focusing on STRIDE threat modelling, I can delve deep into the specific threats that can arise during the software development process and identify ways to mitigate them. This project also allows me to gain practical experience with conducting interviews and data analysis, which are valuable skills for a career in cyber security. Ultimately, my goal is to contribute to the development of more secure software and help protect organizations and their users from cyber threats.

1.4 REPORT STRUCTURE

There are six sections in this report:

The first part of the study examines threat modelling as a means of integrating security with the DevOps technique in software applications. Bearing in mind the problem-solving technique, it outlines why the work is being conducted, and the project's primary purpose.

In the second chapter, a complete literature overview of the project topic is provided, as well as a critical analysis of threat modelling and its many tools and methodologies.

The project specifications which include the aims, objectives, system requirements, LESP issues, and many others are discussed in chapter three.

The fourth chapter explores how STRIDE threat modelling may be integrated into the DevOps process of the case study with enough assessment carried out.

The evaluation of results will be explored in the fifth chapter.

In its sixth and final chapter, the report is summarized and concluded, as well as future areas for research.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

This section presents a complete literature overview of the project topic is provided, as well as a critical analysis of threat modelling, its tools, and methodologies.

2.2 DEVOPS

DevOps involves a collection of integrated activities or procedures used to automate and connect the processes of software development with IT developers in order to produce, test, and release deliverables rapidly and reliably (Yarlagadda 2021). DevOps is an integrated phrase that relates to development and operations and reflects a cultural interconnection between developers and operators, whose activities were formerly based on silos (Whittle 2014).

Many factors facilitated the merging of the development and operations teams. While the groups' responsibilities are extensively dispersed, making their operations more difficult, the notion was inspired by long-standing obstacles and problems (Yarlagadda 2021). Individual problem-solving capacity diminished as the complexity chain grew more complex. Providing virtual machines in a broad network region, configuring extended network devices and servers intricately, and deploying diverse applications are some of the crucial interventions that led to the consolidation (Yarlagadda 2021). Moreover, log gathering and aggregation, monitoring services, monitoring network performance, and monitoring application performance had become complicated issues. In addition, the developers, operators, and engineers were unable to comprehend the events that led to these difficulties, preventing them from issuing alarms and seeking solutions.

2.3 THREAT MODELLING

The practice of creating an adversarial threat model for each cyber-physical system is known as threat modelling (Dobaj et al. 2021). Many aspects, including applications, sub-systems, devices, mission functions, cyber-physical systems, and networks, executing these functions, might be the target of or affected by such threats. Cyber-attacks can also have an impact on whole enterprises, areas, or crucial infrastructure sectors (Dobaj et al. 2021).

In other words, cyber dangers might appear at different system levels and influence a range of system attributes and operations. To identify possible attacks, weaknesses, and gaps in the system under consideration, the threat modelling approach may give information on cybersecurity elements in several different ways (Bodeau, McCollum and Fox 2018):

1. Management of Risk: Risk management includes several processes, some of which include threat modelling, such as risk conceptualization, assessment, and analysis of risk responses

(Bodeau, McCollum and Fox 2018). Most of the time, these tasks are included into broader frameworks for managing business and project-level risks. While this study primarily focuses on threat modelling, risk management does consider non-adversarial risks (such as dangers to devices from an operational safety perspective) (Dobaj et al. 2021).

2. Cyber wargaming: Modelling threats facilitates the identification and development of threat scenarios and possible attacks for cyber wargaming and vulnerability scanning (Bodeau, McCollum, and Fox 2018; Dobaj et al. 2021).
3. System Security Engineering**:** Threat modelling may be utilized at any stage of the system development lifecycle, including requirement analysis, systems design, implementation, validation, and maintenance and operation (Dobaj et al. 2021). When it comes to design evaluation and testing, threat modelling is crucial since the detected attack events and threat scenarios are used to direct system design, the choice of mitigation strategies, and the evaluating of these measures. Threat modelling may be adjusted to represent the structure, functions, and layers of an individual system at granularity levels that are useful for addressing specific security concerns (Dobaj et al. 2021).

In summary, threat modelling is a general technique that may be used for a wide range of cybersecurity-related tasks throughout any stage of the system/product lifecycle, from design to development to analysis to operations and maintenance (Dobaj et al. 2021). Yet, in a given lifespan stage, only certain details are usually of concern. For this reason, it is essential that the method of threat modelling be adapted to the requirements at each stage of the lifecycle.

2.3.1 Threat Modelling Techniques

As shown on the left side of figure 2.1, there are three ways to approach assessing cybersecurity threats and risks (Ebert and Jones 2009):

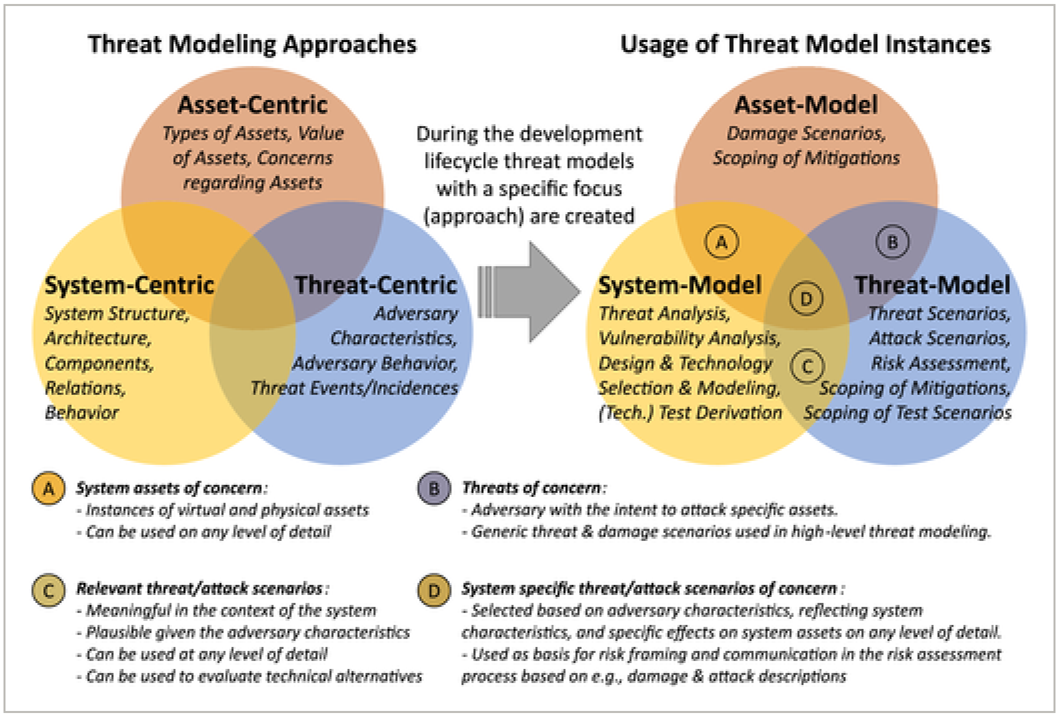

Threat Modeling Approaches
Usage of Threat Model Instances
Asset-Centric
Types of Assets, Value of Assets, Concerns regarding Assets
System-Centric
System Structure, Architecture, Components, Relations, Behavior
Threat-Centric
Adversary Characteristics, Adversary Behavior, Threat Events/Incidences
During the development lifecycle threat models with a specific focus (approach) are created
Asset-Model
Damage Scenarios, Scoping of Mitigations
System-Model
Threat Analysis, Vulnerability Analysis, Design & Technology Selection & Modeling, (Tech.) Test Derivation
Threat-Model
Threat Scenarios, Attack Scenarios, Risk Assessment, Scoping of Mitigations, Scoping of Test Scenarios
A
B
C
D
A System assets of concern:
- Instances of virtual and physical assets
- Can be used on any level of detail
B Threats of concern:
- Adversary with the intent to attack specific assets.
- Generic threat & damage scenarios used in high-level threat modeling.
C Relevant threat/attack scenarios:
- Meaningful in the context of the system
- Plausible given the adversary characteristics
- Can be used at any level of detail
- Can be used to evaluate technical alternatives
D System specific threat/attack scenarios of concern:
- Selected based on adversary characteristics, reflecting system characteristics, and specific effects on system assets on any level of detail.
- Used as basis for risk framing and communication in the risk assessment process based on e.g., damage & attack descriptions

Figure 2.1*: Threat modelling methodologies and their utilization for modelling system security elements (Dobaj et al. 2021)*

1. **Asset-Centric**: To define possible risks to the identified assets, threat modelling first recognizes assets of the system under evaluation that might be affected by the threats (Dobaj et al. 2021).
2. **System-Centric**: To establish what dangers are pertinent to the modelled context, threat modelling first models the examined system, including its architecture, data, scope, and environment (Dobaj et al. 2021).
3. **Threat-Centric**: Modelling threats (such as adversary attributes, behavior, threat occurrences, and assault patterns) is the initial step in incorporating them into the evaluation process (Dobaj et al. 2021).

Threat modelling is a unidirectional method for doing analyses such as risk evaluation and identifying weaknesses and vulnerabilities (Dobaj et al. 2021). Nevertheless, systems engineers advise using a multifaceted approach to threat modelling to gain a more complete picture of the system under evaluation. Figure 1's right-hand Venn diagram shows how combining threat modelling instances that focus on diverse elements of cybersecurity may be valuable because of the overlap between them (areas A, B, C, and D).

2.3.2 Methods of Threat Modelling

The threat discovery step is the main emphasis of most threat modelling approaches. By offering a high-level classification of threat families, STRIDE seeks to achieve this (Hussain et al. 2014; von der Assen et al. 2022). Some methods concentrate on identifying pertinent dangers from a list of actual threats. A list of typical attack trajectories that are detailed from an adversarial standpoint is made available to the public by CAPEC (2019a) These descriptions frequently cross-reference a second technique known as the Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) to offer operational context from an adversary perspective (CAPEC 2019b). Several threat enumeration libraries also concentrate on certain technology.

There are several methodologies available for threat modelling in the industry, and the most popular ones include STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege), Trike (Threat, Risk, Impact, Knowledge, and Estimate), PASTA (Process for Attack Simulation and Threat Analysis)< DREAD (Damage, Reproducibility, Exploitability, Affected Users, and Discoverability), VAST (Visual, Agile, and Simple Threat modelling), and PIA (Privacy Impact Assessment).

For this study, the STRIDE threat modelling approach is used as a way of integrating security with DevOps methodology in software application development. Deeper dive into each methodology and reason for selecting STRIDE among others are discussed in detail in chapter three.

2.4 THREAT MODELLING TOOLS

Threat modelling tools may be grouped along a wide range of criteria. Applications that are used often and lack a particular technique are known as general purpose tools. Nonetheless, due to their versatility, threat modelling is possible. While asynchronous and remote settings are not supported, whiteboards are nonetheless widely used for quick modelling (Shostack 2014). Office suites offer a similar adaptable method for asynchronously sharing threat models. Lastly, threat modelling extensions are available for the online diagramming application diagrams.net (Tuma, Calikli and Scandariato 2018). Several distant users can collaborate on the same diagram simultaneously thanks to differential synchronization (Shostack 2014). The synchronization interval for the models, however, makes this tool susceptible to mistakes. Moreover, none of these systems offer any instruction to novice users.

Manual modelling applications that include a diagram drawing component include the Microsoft Threat Modelling Tool, the Open Web Application Security Project (OWASP) Threat Dragon (Tøndel et al. 2019; Cruzes et al. 2021), the TRIKE, the SeaMonster, and the CAIRIS (Rajapakse et al. 2022). Nonetheless, they all use the same approach to threat modelling. Web-based apps make it feasible to employ these technologies in an asynchronous and distant context. Nevertheless, simultaneous model creation is not possible, as evidenced by surveys of relevant knowledge (Cruzes et al. 2021). Because of their inseparable connection to a single approach, they may be used in only the most appropriate collaboration contexts.

Automated Threat Modelling and software-based threat modelling tools, such as Tutamantic, IriusRisk, MAL, securiCAD, Threagile, ThreatSpec, and Raindance, have been developed to automate the detection of threats and their responses using pre-existing artifacts such as source code or architectural diagrams. Although collaborative use cases like workshops may be challenging due to the reliance on current input or the editor's inability to manage multiple users at the same time, automated threat discovery systems provide a knowledge base, allowing consumers with less expertise to detect threats without the assistance of a security professional. ThreatModeler, SDElements, and pytm offer hybrid methods that combine these techniques with the software development life cycle.(von der Assen et al. 2022).

2.4.1 CAIRIS Threat Modelling Tool

Computer Aided Requirements and Information Security, or CAIRIS, is the system's official name. It is a free and open-source tool for gathering requirements for functional and trustworthy software (CAIRIS 2023). It has been developed from the bottom up to include all the features required for usability testing, requirements gathering, and risk assessment (CAIRIS 2023).

Several programs are designed specifically for the task of defining requirements. While some researchers center their efforts on hazard simulations, others take a different approach. The focus of still others is on UX data management. Only CAIRIS can do all these tasks (and more). When it comes to modelling different use scenarios, CAIRIS is the only security design tool that considers the concept of environments (CAIRIS 2023).

Software is only as good as its users, so even if security is "incorporated in," users may still be at risk. CAIRIS was developed to facilitate the processes essential to include security and usability from the outset of software development. Furthermore, CAIRIS' automatic analysis, validation, and visualization capabilities will aid the study as the repository for data acquired in this research (CAIRIS 2023) .

CAIRIS was used by Faily and Iacob (2017) as a means for usability and security engineers to work together. Following an overview of CAIRIS's primary features, they provided several examples of how the tool could be put to use by security and usability engineers. This demonstrated the potential for usability and security engineers to work together using automated procedures facilitated by tools, such as Design as Code, which involves the management and provisioning of machine-readable models of the early design stages. Through these examples, they were able to demonstrate how CAIRIS streamlines common security and usability engineering tasks and improves communication between interaction designers and security engineers.

Altaf et al. (2019) demonstrated the creation of attacker personas using CAIRIS as a tool of support, using the rail sector as an example. It was proven that the interdependence of safety, security, and human factors engineering could be modelled using attacker personas and the Polish Tram Incident as a case study. The Polish Tram Incident was used as a model. The hidden system vulnerabilities that could cause harm to the larger environment were discovered utilizing attacker personas. These flaws prompted the identification of threats, which in turn influenced risk analysis, and finally, the associated safety dangers, such as service interruptions, accidental collisions involving two or more trains, and personnel or passenger deaths, were identified based on this research. Human factors issues like roles, responsibilities, and goal-obstacle modelling were also identified with the help of the personas' story. The verification of this character, however, was left for further study.

Following news of the Stuxnet worm, Faily (2018) presented a case study in which IRIS and CAIRIS were used to examine and develop an information security strategy for a UK water firm.

2.4.2 Justification of Tools' Choice

CAIRIS (Computer-Aided Integration of Requirements and Information Security) is an open-source tool used for modelling and analyzing security requirements. It is designed to assist developers in designing secure software systems through an automated threat modelling process.

There are several alternative solutions available for threat modelling, such as Microsoft's Threat Modelling Tool, IriusRisk, and PyTM. However, the decision to focus on CAIRIS appears to be not only for its wide usage in the literature, but also because it is known to be a reliable and user-friendly tool.

With CAIRIS being an open-source tool, it becomes more accessible to students and small businesses. Additionally, CAIRIS has been designed to integrate with other software engineering tools, such as JIRA, making it easier to incorporate security into the development process.

Moreover, CAIRIS uses a data-driven approach to threat modelling, which makes it easier to create and manage large and complex models. It also supports a wide range of threat modelling methodologies, including STRIDE, which is the methodology chosen to be used for this research.

CAIRIS compatibility makes it easy to implement on desktop or laptop and it also possesses a broad variety of documentation/manual that can be easily accessed and followed when being implemented.

Overall, the combination of CAIRIS's reliability, user-friendliness, open-source nature, integration capabilities, and support for various threat modelling methodologies makes it a suitable tool for my project.

2.5 INTEGRATING THREAT MODELLING WITH DEVOPS

When it comes to developing safe software, threat modelling is a common practice. Threat modelling allows one to examine a system, pinpoint potential entry points for attackers, and devise countermeasures to reduce vulnerability to attack. If done correctly, threat modelling can be a valuable part of any strategy for mitigating risk. Because it allows for the detection and correction of design flaws at an early stage, it can also aid in the reduction of associated expenses. According to a previous NIST study, the price tag for addressing a design flaw in live code is roughly 40 times greater than if it had been caught and fixed in the design process. It prevents future design problems from adding extra expenses due to security breaches. Threat modelling is the initial step in securing a software solution since it functions on the solution's architecture. This quality makes it the most effective security approach applicable to the Software Development Lifecycle (SDLC).

2.6 RELATED WORKS

There are several academic works that investigate threat modelling in diverse settings [10–17]. The methods currently in use for threat modelling were analyzed in research by Tuma, Calikli and Scandariato (2018). There was a total of 26 approaches that were assessed for their applicability, features of the needed input for analysis, characteristics of the analysis approach, attributes of the analysis findings, and ease of adoption across 38 main studies that were analyzed in the study. Misuse instances, attack trees, issue contexts, and numerous software-centric methods, like STRIDE (Shostack 2014; Cruzes et al. 2021), CORAS (Lund, Solhaug and Stølen 2011), and P.A.S.T.A (Ucedavélez and Morana 2015), were among the most frequently employed techniques in the offered body of knowledge. The STRIDE (Mead, Shull and Vemuru 2018) method is the industry standard. They also examined the present state of acceptance of the existing methodologies and provided insight into the barriers to adoption in software engineering. (e.g., Agile, DevOps, etc.). The conclusions can be summed up as follows: the analysis technique was not clearly specified; quality assurance of analysis results was lacking; and there was a lack of tool support and validation.

In the past, threat modelling has been linked to the various steps of the waterfall methodology, beginning with an overarching, front-end design of the final product. Agile development, on the other hand, a gradual and service-oriented approach to software creation do not rely on a single, overarching authority figure. Throughout the years, several suggestions have emerged for incorporating security engineering techniques into agile approaches like Scrum and XP (Türpe and Poller 2017), founded on the widely held belief that agile teams fail to adequately address security since it is not a required element of most agile frameworks.

Although Poller et al. (2017) mentioned that SCRUM is effective as a management strategy, and security advancement requires repetitions just like in agile development, SCRUM teams could fail to address security needs because of their limited visibility, competing objectives, and the division of labor inherent in SCRUM.

Security is seldom significantly improved by ad hoc repairs (Cruzes et al. 2021). Security is a difficult-to-verify quality criteria because of the intricate link it has with the development process. While prioritizing security features in the pipeline would be an appropriate solution to many security problems, doing so might be challenging due to competition from other needs and the potential need for specialized knowledge to be designed and implemented successfully (Cruzes et al. 2021). Some of the difficulties they discovered in their exploratory study overlap with those they discovered in the literature, and the additional difficulties they discovered stemmed mostly from the activities of asset identification as well as their observations of what transpired following the threat modelling sessions. For businesses wishing to do threat modelling utilizing the Microsoft Threat Modelling technique, they also identified several lessons learnt.

The study revealed that there are still many issues to be resolved before threat modelling is properly used in agile development projects. Therefore, the objective of future work is to more fully comprehend the difficulties raised here and to validate findings in many scenarios.

CHAPTER 3: PROJECT SPECIFICATION

3.1 INTRODUCTION

This chapter discusses the project's aims and objectives, system design and architecture. It will cover the methodology used to develop the framework and how it is adjusted to the research objectives while taking into consideration project constraints.

3.2 AIMS AND OBJECTIVES

3.2.1 Aim

This project aims to enhance application security by integrating threat modelling into the continuous integration pipeline during software automation.

3.2.2 Objectives

The overall objective of this project is to discover and mitigate threats in the continuous integration toolchain using the four-step framework which will be:

1. To Investigate an existing system, conduct interviews and collect necessary data towards designing of threat model to improve the system security.
2. To build a solution that models the entirety of the case study system – including all assets and processes.
3. To find and address threats in the solution.
4. To explore the challenges of modelling threats and potential strategies for addressing those challenges.

3.3 SYSTEMS REQUIREMENTS

This section describes the elicited basic functionality required for daily operations of modelling solution from the system case study. These requirements will be summarized and applied in CAIRIS for the studied system being modeled. These requirements can be fed into CAIRIS using its requirements function of which "requirements" will be chosen from among its group specification concepts. Entry information will include the identified requirements from studied system, its specification, requirement type whether it is functional, privacy, security, and legal and so on.

These requirements must be met by the model under development for the research objective to be attained. Both system and user requirements need to be met for the final output to be regarded accurate and exhaustive.

This study will employ the MoSCoW Prioritization Technique. This method is employed to increase stakeholder comprehension of the relative importance of meeting each criterion. The abbreviation MoSCoW is composed of the initial letters of the four key levels (MUST HAVE, SHOULD HAVE, COULD HAVE, and WON'T HAVE) (Product plan 2022; Volkerdon 2022).

3.3.1 Functional Requirements

These conditions must be met by both the system and its users before the study can be carried out successfully. These functional criteria must be met for the proposed system to be considered comprehensive and accurate.

Req. ID	Functional Requirements	Prioritization
R1	**Support for multiple threat modelling methodologies** Model should support a variety of threat modelling methodologies, such as STRIDE, DREAD, or PASTA, to accommodate different types of systems and users' preferences**. In this case the STRIDE approach is considered**	**Should**
R2	**User Authentication** All users (admins, KPO and van salesman) of the DMS **must** be authenticated.	**Must**
R3	**Server Availability** The solution **must** be available to enable an easy threat modelling process.	**Must**
R4	**Security** The solution **must** ensure that no two users share the same login credentials.	**Must**
R5	**Diagramming Capabilities** The tool **must** provide the ability to create and edit diagrams to represent the system and its components, data flows, and other relevant information.	**Must**
R6	**Threat Model** Solution **must** be able to build a threat model of the DMS.	**Must**
R7	**Automated Threat Identification** The tool must have a capability to automatically detect potential threats in the system based on the model created.	**Must**
R8	**Risk Management and Assessment** CAIRIS **should** allow users to assess the severity of threats and prioritize them based on the likelihood of occurrence and potential impacts	**Should**
R9	**Integration with other tools** The Solution should have the capability to integrate with other tools in the	**Should**

	development and security ecosystem, such as bug tracking systems or security testing tools.	
R7	**Reporting** Solution **must** provide the ability to generate reports that summarize the identified threats, their likelihood and impact, and recommended mitigation strategies.	**Must**
R8	**User-friendly Interface** Model should have a user-friendly interface that allows for easy navigation and interaction with the different features and functionalities.	**Should**

Table 3.1*: Functional Requirements based on MoSCoW Technique*

3.3.2 Non-Functional Requirements

These requirements consider potential performance constraints as well as functional aspects of the system. Time and other conventional constraints on development are common.

Req. ID	**Non- Functional Requirements**	**Prioritization**
R1	**User Validation:** The solution **should** ensure that user login details are validated in less than 15 seconds.	**Should**
R2	**Response Time:** Solution **should** provide results within the shortest possible time under good network condition.	**Should**
R3	**Language:** To meet the comprehension necessities of its users, the system's user interfaces **must** be designed in English.	**Must**
R4	**Performance** System **should** be able to handle large and complex models without slowing down or crashing and should provide fast and responsive feedback to users.	**Should**
R5	**Usability:** Solution **should** be easy to use, intuitive, and provide clear instructions and guidance to users.	**Should**
R6	**Compatibility** Model **must** be compatible with different operating systems, platforms,	**Must**

	and other tools in the development and security ecosystem.	
R7	**Maintainability** Model **should** be easy to maintain and update, with clear documentation and support available to users.	**Should**
R8	**Scalability** When more resources are introduced to the system, the solution **should** be able to handle an increasing workload**.**	**Should**
R9	**Security:** The solution **must** be secure and protect sensitive information, such as system architecture diagrams and threat data, from unauthorized access or disclosure.	**Must**

Table 3.2*: Non-Functional Requirements based on MoSCoW Technique*

3.4 METHODOLOGY

3.4.1 Research Methodology

This study intends to investigate the use of threat modelling in DevOps. The case study for this research is a popular brewery company located in Nigeria. The purpose of the research is to understand their operations and challenges using the Distributor Management System which will be discussed in the software methodology. The study's findings will come from interviews with a group of employees at the company. The three stages of the study's methodology—study preparation, gathering of data, and data analysis—are depicted in the figure below. The descriptions of each step are also seen below:

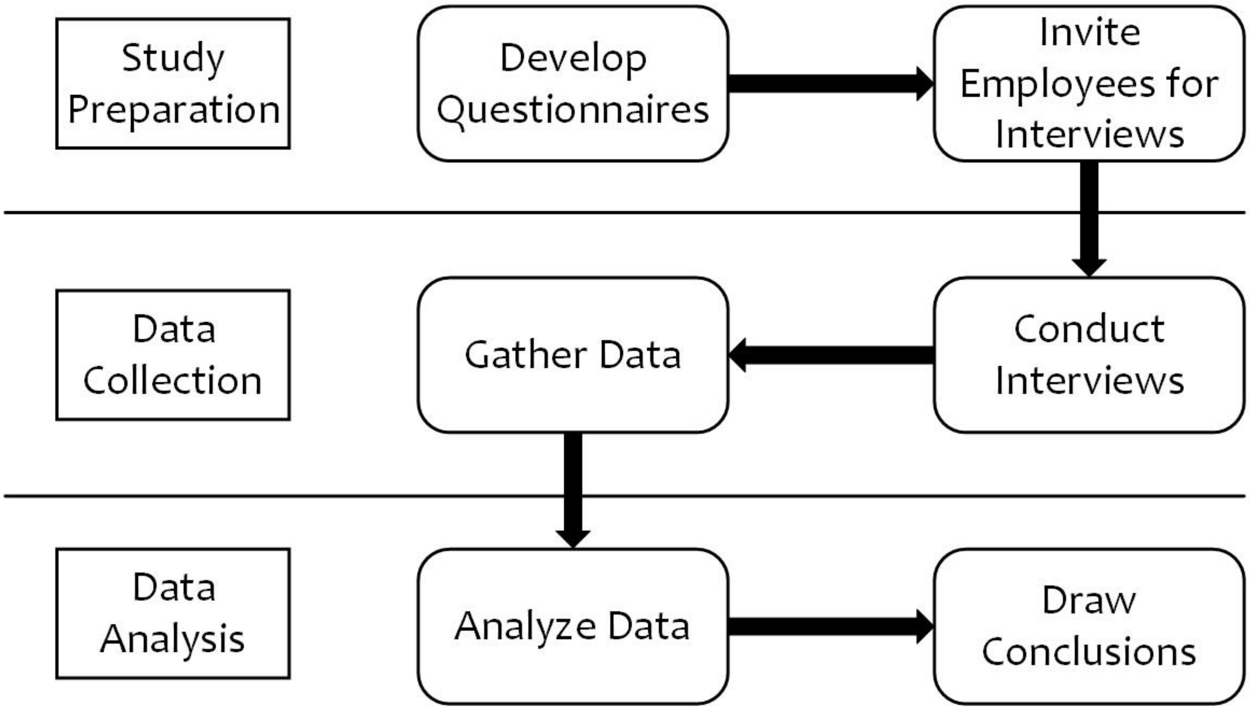


***Figure 3.1**: Basic Research Process*

1. **Study Preparation**: A set of interview questions with open-ended responses has been developed. This set of open-ended questions is designed to elicit extensive responses from the participants. Some employees from the organization will be invited. The aim of this interview is to understand their experiences using the Distributor Management System. This includes their daily operations on the system and the challenges they face. This will enable us to understand the different vulnerabilities and attacks the system is prone to and therefore develop a threat model for the system.
2. **Data collection**: The interview will be conducted with the selected participants from the company through Zoom and the use of questionnaires. A poll will be done to enable easy gathering of data.

3. **Data analysis**: the collected data will then be analyzed using a data analysis tool.

3.4.2 Software Methodology

There are several methodologies available for threat modelling in the industry, and the most popular ones are Trike, PASTA, STRIDE, DREAD, VAST, and Privacy Impact Assessment (PIA). Each methodology has its strengths and weaknesses, and the choice of methodology depends on the specific needs of the project. Here is a brief discussion of each methodology:

1. **PASTA (Process for Attack Simulation and Threat Analysis)**

   PASTA is a seven-step methodology for threat modelling that provides a comprehensive approach to evaluating risks. It is well-suited for complex systems and environments that have many interacting components. One of the strengths of PASTA is its flexibility, allowing organizations to tailor the methodology to their specific needs. However, PASTA can be time-consuming and resource-intensive due to its comprehensiveness.

2. **Trike**

   Trike is a three-stage methodology for threat modelling that emphasizes collaboration between security and development teams. It includes a visual modelling component that makes it easier to understand and communicate potential risks. Trike's primary advantage is its emphasis on collaboration, which can help ensure that security is integrated into the development process. However, Trike may not be as comprehensive as other methodologies, and it may not be suitable for organizations that have a complex threat landscape.

3. **VAST (Visual, Agile, and Simple Threat modelling)**

   VAST is a methodology for threat modelling that emphasizes simplicity and ease of use. It includes a visual component that makes it easier to understand and communicate potential risks. VAST is ideal for small projects and organizations that do not have a lot of resources or expertise in threat modelling. However, it may not be comprehensive enough for more complex systems or environments.

4. **DREAD (Damage, Reproducibility, Exploitability, Affected Users, and Discoverability)**

   DREAD is a methodology for evaluating and prioritizing potential threats based on five criteria: damage, reproducibility, exploitability, affected users, and discoverability. It is a straightforward methodology that can be used by non-experts to evaluate potential risks quickly. However, DREAD may not be as comprehensive as other methodologies and may not provide enough detail to develop effective mitigation strategies.

5. **PIA (Privacy Impact Assessment)**

   PIA is a methodology for evaluating the potential impact of software systems on user privacy. It includes a comprehensive checklist of privacy-related risks that can be used to evaluate potential threats. PIA is ideal for organizations that handle sensitive user data or are subject to privacy regulations. However, it may not be as comprehensive as other methodologies and may not be suitable for organizations that do not handle sensitive user data.

Each of these methodologies has its strengths and weaknesses. However, after careful consideration, we have decided to use the STRIDE methodology for our project.

3.4.2.1 STRIDE Methodology

Microsoft proposes the STRIDE technique, which is an acronym for six distinct categories of security threats (Khan et al. 2018):

1. Spoofing: the impersonation of a valid user, process, or system component.
2. Tampering: this is the alteration or altering of valid data.
3. Repudiation: Denial or disavowal of a particular system activity.
4. Information Disclosure: data breach or illegal access to confidential data.
5. Denial of Service (DoS): Service disruption for authorized users.
6. Elevation of privilege: granting a user with restricted authorization access to a system element with elevated privileges.

Utilizing the STRIDE technique, this study examines each system component's vulnerabilities that an attacker may exploit to compromise the entire system. The system in consideration for this study is the Distributor Management System.

As noted in table 3.3, STRIDE threats can be translated to an abstracted DFD system.

Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Interactors	X		X			
Processes	X	X	X	X	X	X
Data Stores		X		X	X	
Data Flows	X	X		X	X	

***Table 3.3**: STRIDE to DFD elements matrix (Mead, Shull and Vemuru 2018)*

It is important to note that generic elements are also mapped from STRIDE to DFD elements in the manner. This matrix (Table 3.3) could be fine-tuned (Shostack 2014 p. 5) when applied to more elements and communication scenarios; for example, a data store might not always be vulnerable to denial-of-service assaults, but to repudiation when installing a log service.

3.4.2.2 Justification for STRIDE Approach

The STRIDE methodology is an effective and widely adopted framework for threat modelling. It stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. STRIDE is a Microsoft-created methodology that is widely utilized in software development organizations.

One of the significant advantages of STRIDE is its comprehensiveness. The methodology covers all significant aspects of threat modelling and helps developers to identify and mitigate potential risks at an early stage of software development. It provides a structured approach for evaluating threats, and the threat library is frequently updated to incorporate new threats.

Another advantage of STRIDE is its simplicity. It's easy to learn and apply, making it an ideal choice for DevOps teams who may not have a lot of experience with threat modelling.

However, the STRIDE methodology does have some limitations. For instance, it can be too broad and fail to provide specific guidance for addressing specific threats. Additionally, it may not work for organizations that have unique threat models that differ significantly from those covered in the STRIDE library.

For this study, the STRIDE threat modelling approach is used as a way of integrating security with DevOps methodology in software application development. The following considerations led to the selection of STRIDE for this work:

1. It employs a methodical model-based approach to examine cybersecurity hazards for every asset (in an automated manner)(Dobaj et al. 2021) .
2. It is thorough and assesses each asset's availability, secrecy, authenticity, non-repudiation, and authorization in terms of security features (Dobaj et al. 2021).
3. It makes it possible to automate the model-based description of probable assault vectors (Dobaj et al. 2021).
4. It helps in identifying and outlining danger situations (Dobaj et al. 2021).
5. In keeping with the signal-flow analysis utilized for functional safety assessment, it leverages dataflow diagrams for system modelling and analysis (Dobaj et al. 2021).

6. It is hierarchical and modular in nature (Dobaj et al. 2021).

3.4.2.3 Tabular Summary and Comparison of All Methodologies

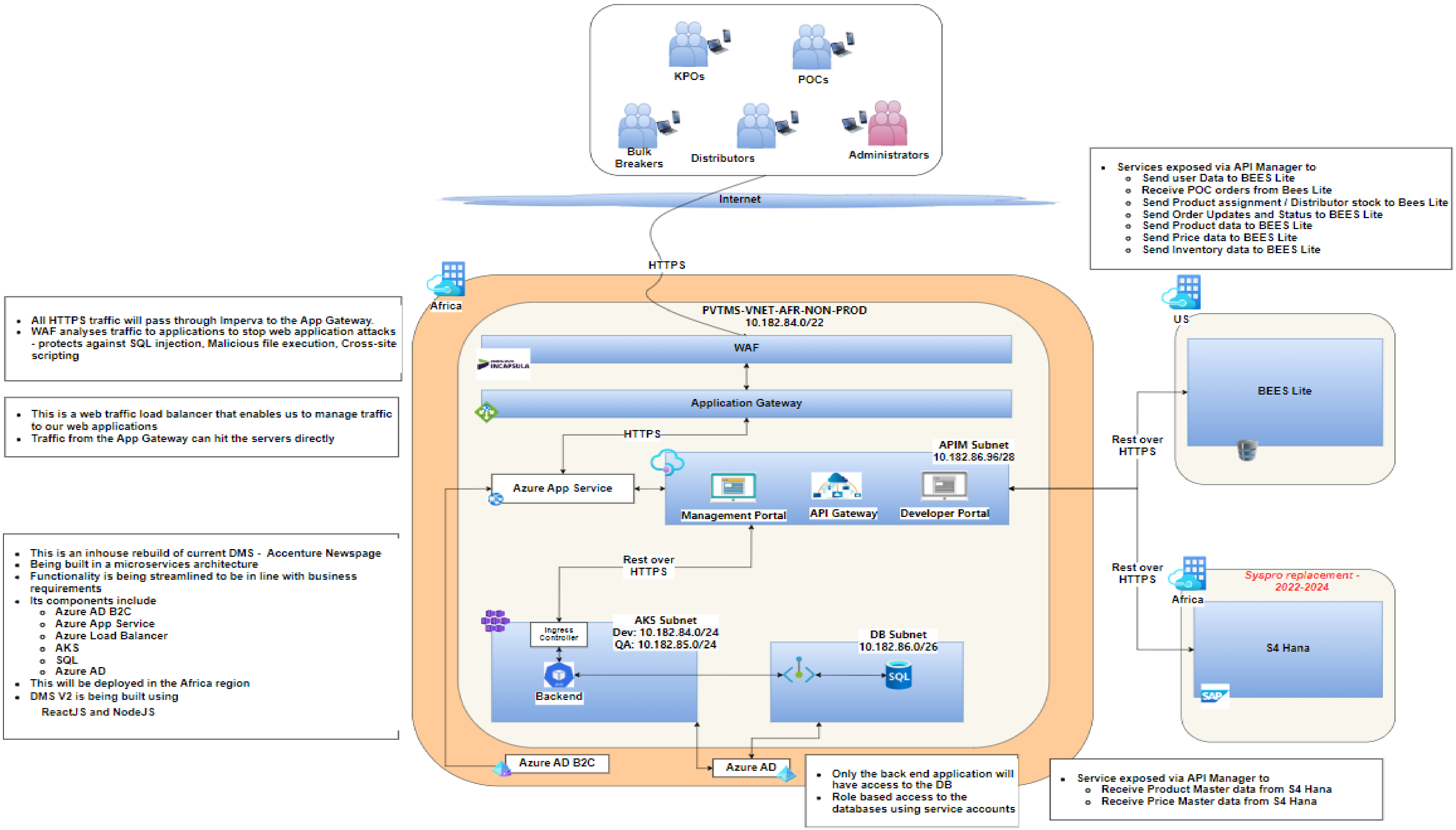
Methodology	**Widely Used**	**Identifies threats in design phase**	**Comprehensive**	**Structured**	**Collaboration**	**Visual modelling**	**Covers wide range of threats**	**Privacy focused**
STRIDE	✓	✓	✓	✓		✓	✓	
Trike		✓	✓		✓	✓		
PASTA			✓	✓				
DREAD				✓				
VAST			✓			✓		
PIA								✓

***Table 3.4**: Tabular Summary of all Methodologies*

3.4.3 Case Study: Distributor Management System (DMS)

A case study was conducted towards the collection of data for the designing of this project and the unit of evaluation was a distribution management system owned by a brewery company. The company has over 50 employees that work in different roles related to the system. Thus, during the case study, interviews were conducted with different volunteers from the organization to understand the main operations of the system which include their daily activities and the system's architecture. The interview covers the different roles and aspect of the system and how they work together to manage the operations of the distribution network from the placing of order up to when the order is delivered, and the transaction Is completed and closed. At the start of the interview, permission was sought, and volunteers agreed to be kept anonymous. Also, consent was given for the collection of information to be used for the purpose of the project work. The captured individual interview responses were developed into transcripts and vital information needed for the system design was extracted.

A Distributor Management System (DMS) is a software solution designed to manage and optimize the operations of a distribution network. It helps to streamline processes such as sales and inventory management, ordering and shipping, pricing, and customer relationship management. The goal of DMS is to improve efficiency, reduce costs and enhance the overall performance of the distribution network.



- Services exposed via API Manager to
  - Send user Data to BEES Lite
  - Receive POC orders from Bees Lite
  - Send Product assignment / Distributor stock to Bees Lite
  - Send Order Updates and Status to BEES Lite
  - Send Product data to BEES Lite
  - Send Price data to BEES Lite
  - Send Inventory data to BEES Lite

- All HTTPS traffic will pass through Imperva to the App Gateway.
- WAF analyses traffic to applications to stop web application attacks - protects against SQL injection, Malicious file execution, Cross-site scripting

- This is a web traffic load balancer that enables us to manage traffic to our web applications
- Traffic from the App Gateway can hit the servers directly

- This is an inhouse rebuild of current DMS - Accenture Newspage
- Being built in a microservices architecture
- Functionality is being streamlined to be in line with business requirements
- Its components include
  - Azure AD B2C
  - Azure App Service
  - Azure Load Balancer
  - AKS
  - SQL
  - Azure AD
- This will be deployed in the Africa region
- DMS V2 is being built using

  ReactJS and NodeJS

- Only the back end application will have access to the DB
- Role based access to the databases using service accounts

- Service exposed via API Manager to
  - Receive Product Master data from S4 Hana
  - Receive Price Master data from S4 Hana

***Figure 3.2**: Distribution Management System Architecture*

The DMS has three roles: an Administrator, a Key Punching Officer (KPO) and a Van Salesman (Driver).

Key Punching Officer (KPO)

These are the sitting cashiers of every distributor; they are the ones in the warehouses of these distributors. They take the stock count of what they have in the warehouse, then they update those values in the system. They also set up the inventory with each product that they have put in there, the prices and the quantity of product that they have for that distributor.

Generally, the application enables the KPO to:

1. Record in-person and telephone sales orders.
2. Setup new distributor’s product inventory.
3. Modify system stock to appropriately represent the situation of the warehouse’s physical stock.
4. Van replenishment and assign orders to drivers.
5. View and update the inventory where necessary.
6. Validate and record returned items.

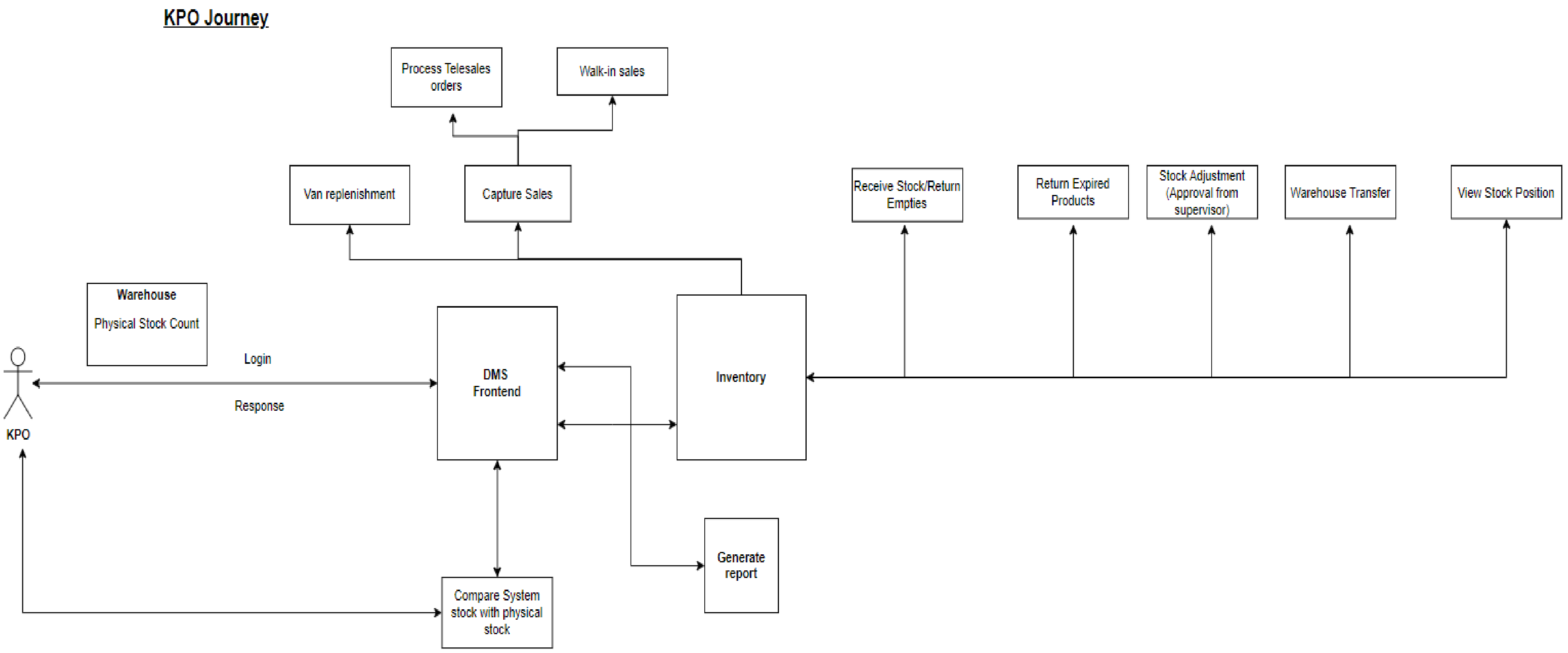


Figure 3.3*: KPO User Journey*

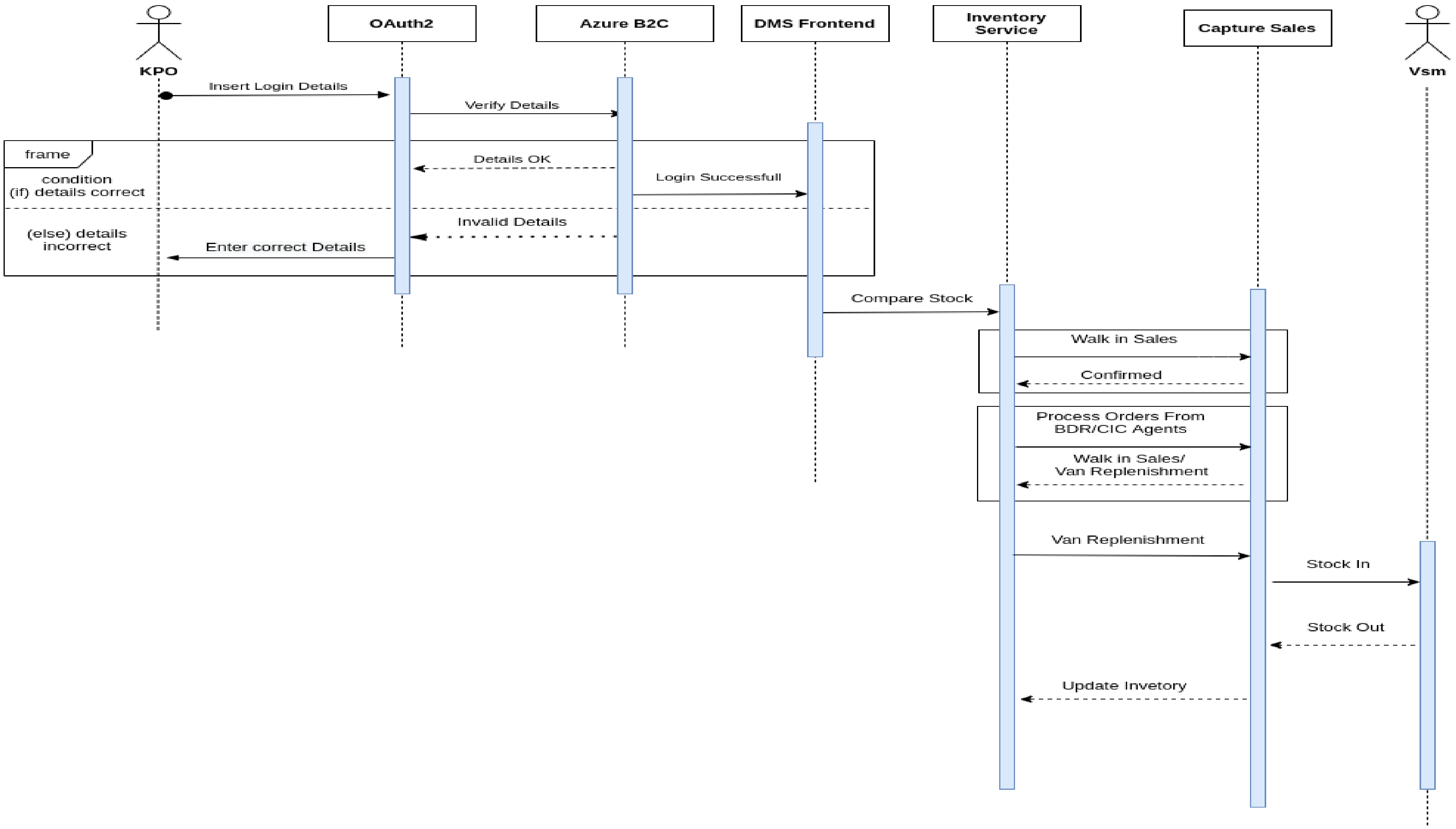


***Figure 3.4**: KPO Sequence Diagram*

Administrator

The administrators control the overall usage of the system and are divided into two:

1. Super admin
2. Mini admin

Mini Admin

1. They have read only access.
2. They can view what is happening on the system.
3. They can see the number of distributors assigned to them.
4. They can see all the information which is assigned to them.

Super Admins

They can:

1. Add new customers.
2. Register and assign new distributors.
3. Assign roles to drivers.
4. Do everything except changing the recommended access (or price) of product.

Rationale

Since they can change what happens in the system, they are held responsible for what goes on in the system.

Admin Journey

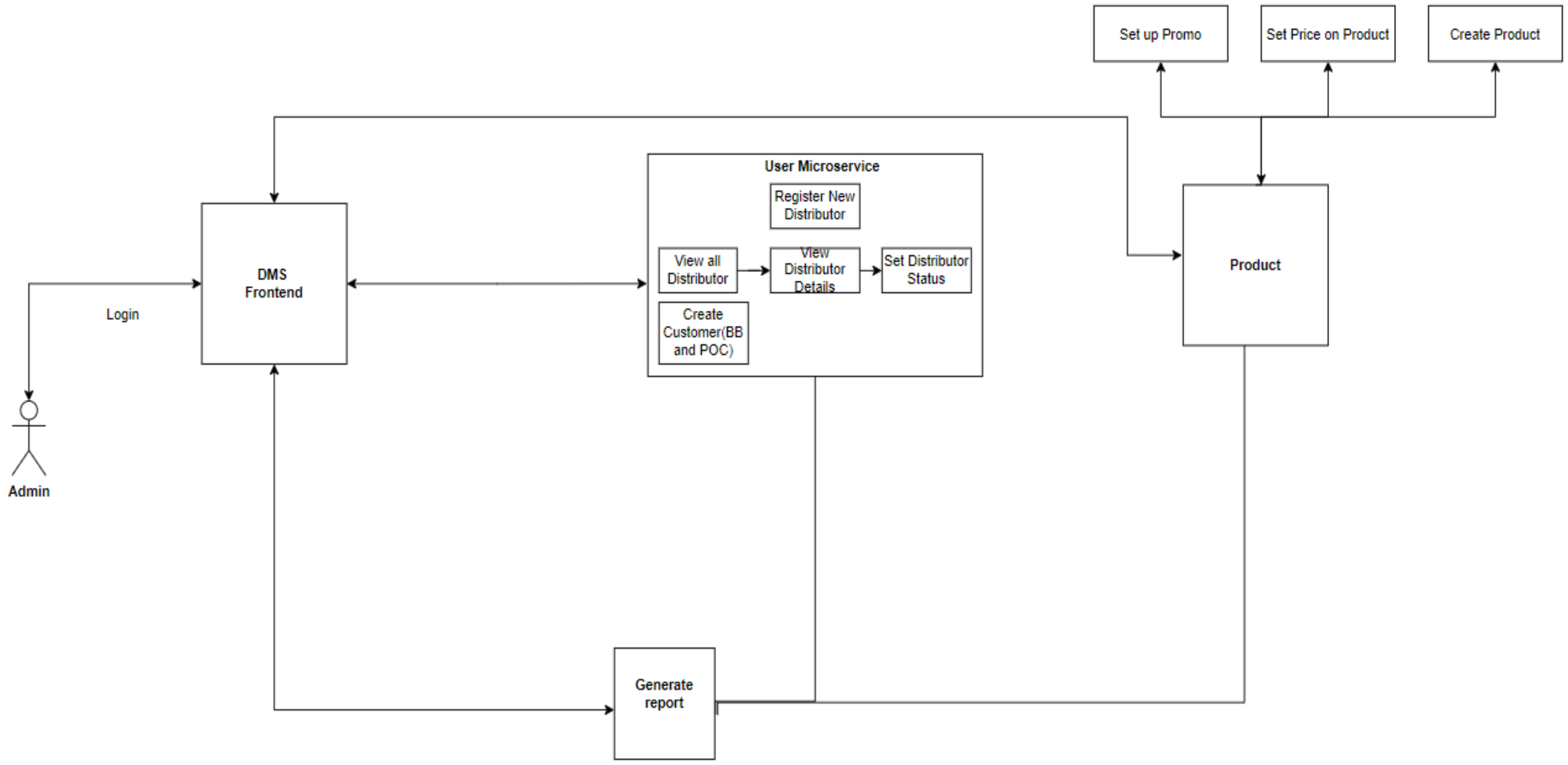


Figure 3.5*: Administrator User Journey*

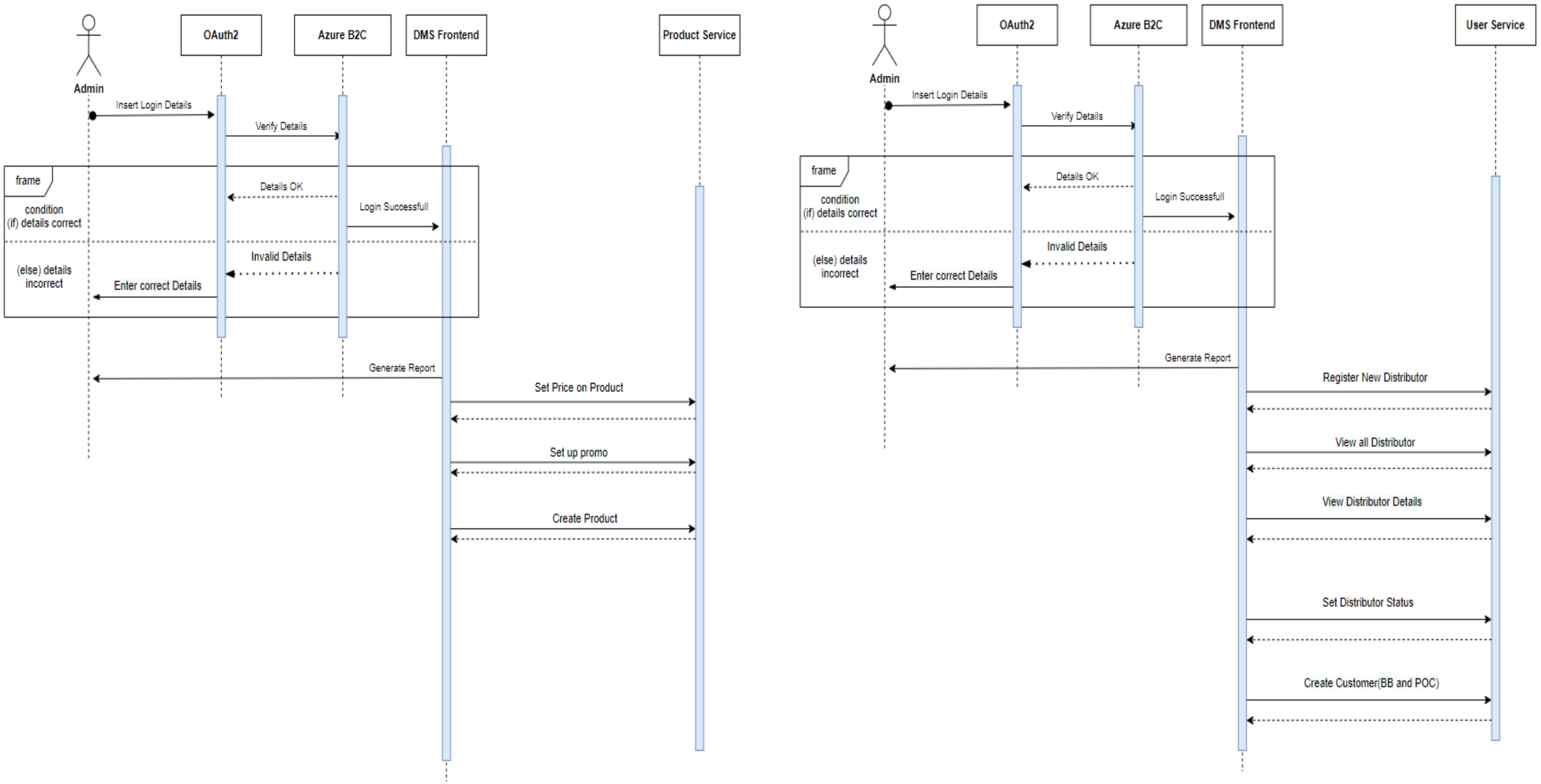


***Figure 3.6**: Administrator Sequence Diagram*

Van Salesmen/Drivers

They are responsible for taking orders from the distributor to the customers that have placed the orders. When the orders are assigned to them from the KPO, they take these orders and deliver them.

The application enables them to:

1) Create orders.
2) Retrieve orders.
3) Deliver orders.

Bulk Breakers (BB) and POCs.

These are the different types of customers. They do not use the system directly. It is just their orders that come to the system. BB are like mini-distributors and are not as big as distributors, so they raise orders from 3rd party applications, e.g., Shop-DC, Salesforce.

3.4.3.1 CAIRIS Threat Modelling Tool

This study uses CAIRIS as a threat modelling tool for the DMS. It is a free and open-source tool for gathering requirements for functional and trustworthy software (CAIRIS 2023) and has been developed from the bottom up to include all the features required for usability testing, requirements gathering, and risk assessment (CAIRIS 2023).

The following five steps were carried out as seen in Figure 3.7:

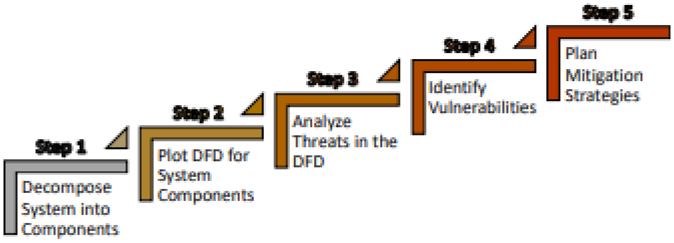

Step 1
Decompose System into Components
Step 2
Plot DFD for System Components
Step 3
Analyze Threats in the DFD
Step 4
Identify Vulnerabilities
Step 5
Plan Mitigation Strategies

Figure 3.7*: STRIDE Methodology Steps (Khan et al. 2018)*

1. **Decomposing the system**: The logical and structural parts of the DMS were broken down. Internal processes, parts communicating among themselves within the system, or external elements interacting with the system, are all examples of components.
2. **Developing the Data Flow Diagram (DFD)**: the DFD depicts the system's internal and external functionalities. The DFD employs four common symbols:
    1. Endpoints of the system, often known as external entities (EE).
    2. Process, also known as functional units.
    3. Data flow, or communication data, is item number.
    4. Data Storage (DS), sometimes known as a database, logs, or file.

    The DFD also has trust boundaries that separate trustworthy parts from untrustworthy ones. As illustrated in table 3.3, only a few of all STRIDE threats pose a risk to each DFD element type.

Before being able to model a DFD for DMS, the CAIRIS system along with all necessary components had to be built accordingly.

Building the CAIRIS system involved several steps. These include:

- Create a cloud account on a public cloud hosting service – Microsoft Azure
- Create and configure a Linux VM (Ubuntu 20.04, 4vcpus, 16GiB memory) to host the threat modelling tool – CAIRIS.
- Install and configure all the required software components, including the CAIRIS software itself, its dependencies, and the database system.
- Ensure the application is accessible over the internet for easy access.
- Set up the CAIRIS server and client applications, which involved creating user accounts and configuring access rights.

Below are the steps that will be followed to create a DFD for the DMS using CAIRIS:

I. Identify the main components of the DMS: This includes components such as the web server, database server, client applications, and various interfaces.

II. Define the data flows: Identify the types of data that are processed by the system, such as user data, transaction data, or system logs. The flow of data between the different components of the system will then be mapped out.

III. Identify the actors: These are the people or entities that interact with the system, such as administrators, KPOs, drivers, bulk breakers. The points where these actors enter or exit the system, and the types of data they access are identified afterwards.

IV. Create the DFD diagram: CAIRIS is then used to create a visual representation of the data flow and interactions within the system. This diagram will clearly show the different components, data flows, and actors involved in the system.

This DFD can be used as a starting point for detecting potential risks and weaknesses when modelling threats on CAIRIS. Using the STRIDE technique, one may determine the potential threats for each diagram component and data flow. This can aid in the creation of a complete threat model for the DMS and the development of mitigation methods.

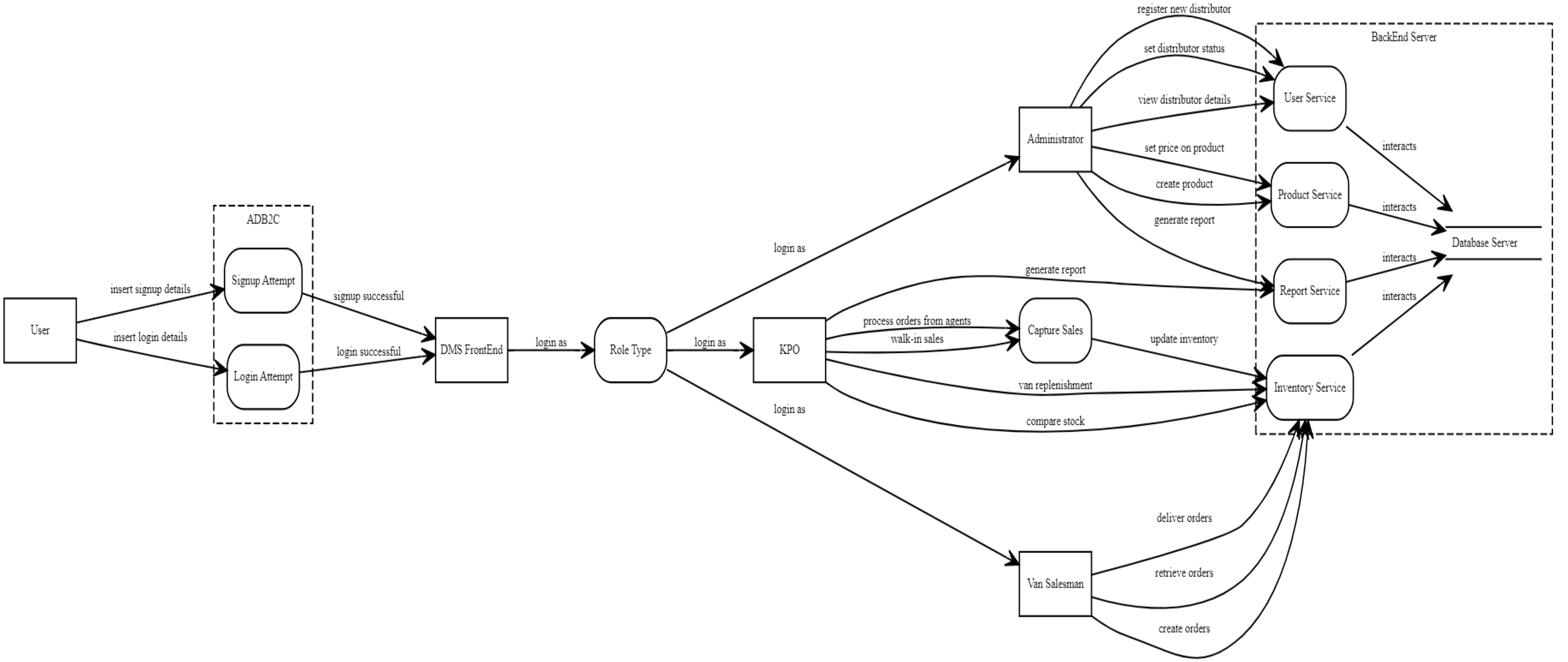


***Figure 3.8**: Data flow Diagram for DMS*

3. **Analyze STRIDE threats and identify vulnerabilities**: for this step, potential STRIDE threats in the DFD of each system subcomponent were identified and analyzed. As potential threats to each part of the system were identified, the underlying vulnerabilities that made them feasible were investigated.
4. **Prioritize the threats**: Once the potential threats are identified, they are prioritized based on their likelihood and impact. There is a need to focus on the threats that are most likely to occur and have the greatest impact on the security of the DMS.
5. **Develop mitigation strategies**: Finally, strategies to mitigate the identified threats will be developed. These may include implementing access controls, using encryption to protect data, and monitoring the DMS for suspicious activity.

3.5 SCOPE

This study evaluates the efficacy of introducing threat modelling into DevOps processes to improve the overall security of software development and deployment. A case study will be conducted on a representative organization to illustrate how threat modelling may be integrated into their DevOps process.

3.6 BRIEF EXPLANATION OF PROJECT PLAN

1. **Case Study**: A case study will be conducted on a sample organization to assess the impact of integrating threat modelling into their DevOps processes on their overall security posture.
2. **Interviews**: Key stakeholders within the sample organization will be interviewed to gather their perspectives on the benefits and challenges of integrating threat modelling into DevOps processes.
3. **Questionnaire**: Two separate questionnaires would be created to access possible loopholes of where an imminent attack could suffice in the case study.
4. **Data Analysis**: Qualitative data gathered from the case study, interviews and questionnaires will be analyzed to identify potential security risks and challenges of integrating threat modelling into DevOps processes.
5. **Threat modelling**: The STRIDE approach of threat modelling will be adopted, and a threat modelling tool known as CAIRIS will be used for modelling the case study.
6. **Report Writing**: A summarized report will be created detailing the project findings together with recommendations.

N.B: a well detailed project plan can be found in **Appendix A** showing the steps taken together with its duration that were carried out for the project.

3.7 REVIEW OF LEGAL, ETHICAL, SOCIAL AND PROFESSIONAL ISSUES

The legal, ethical, social and professional issues considered in this project work can be described as follows:

3.7.1 Ethical Issues

The Robert Gordon University research ethics policy was considered during the design and implementation of this project work.

Participant

(i) Informed Consent: Since the research is based on a live case study, consent will be taken from all the volunteers that participated in the case study interview.

(ii) Confidentiality and Anonymization: To protect the privacy of research participants or subjects, data may need to be anonymized prior to analysis to ensure that individuals or certain information cannot be identified from the data.

Data management

(i) Integrity and confidentiality: Collected data/information will be documented and stored in a secure storage to prevent unauthorised access.

(ii) Storage Limitation: Stored data will be kept only for duration of 3 months (project timeline).

(iii) Purpose Limitation: Data will be used solely for the project work.

3.7.2 Legal Issues

The legal issue that will be considered towards this project include:

Privacy and public right

This work will not be dealing with information which is private or confidential and therefore, will neither use nor store personal/public data. Thus, privacy right will not be misused.

Third party software License

Software to be used in this project are free and will not require licensing but, necessary acknowledgement will be given to software owners.

Compliance

Data/Information to be used in this project will be collected by conducting interviews and Information collected will be managed in compliance with data protection laws such as the General Data Protection Regulation (GDPR) and the Data Protection Act.

The Project will also comply with relevant cybersecurity laws such as the Computer Misuse Act and the Cybersecurity Information Sharing Act.

3.7.3 Social Issues

The project may have implications for privacy, security, and data protection, which may have an impact on society.

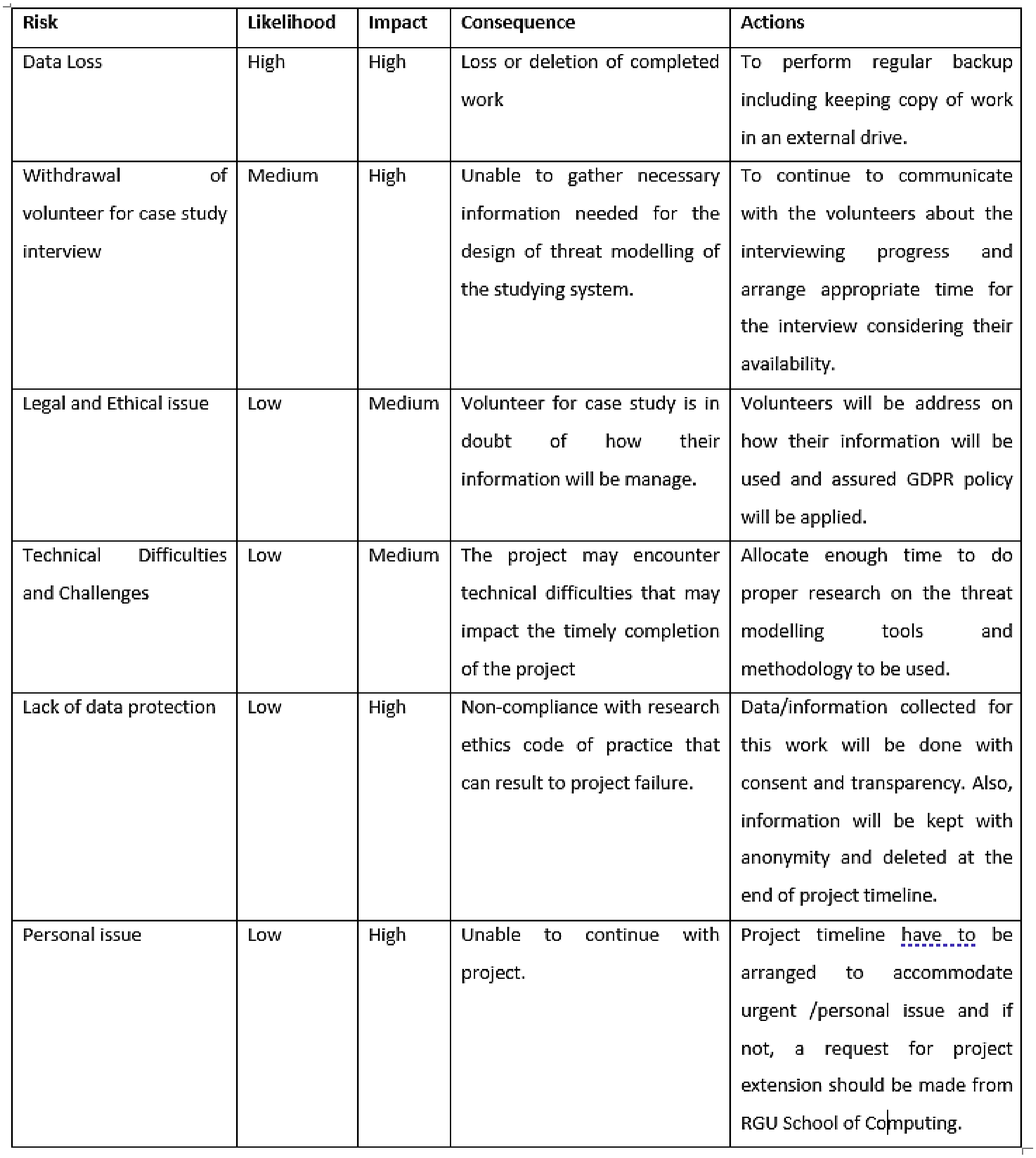
The project will consider the potential social implications of the use of threat modelling and ensure that the project does not violate the rights of individuals.

3.7.4 Professional Issues

The project will adhere to professional standards such as ISO/IEC 27001 and 27002.

The project will ensure that the use of threat modelling is in line with industry best practices and standards.

3.8 PROJECT RISK ANALYSIS



Risk	Likelihood	Impact	Consequence	Actions
Data Loss	High	High	Loss or deletion of completed work	To perform regular backup including keeping copy of work in an external drive.
Withdrawal of volunteer for case study interview	Medium	High	Unable to gather necessary information needed for the design of threat modelling of the studying system.	To continue to communicate with the volunteers about the interviewing progress and arrange appropriate time for the interview considering their availability.
Legal and Ethical issue	Low	Medium	Volunteer for case study is in doubt of how their information will be manage.	Volunteers will be address on how their information will be used and assured GDPR policy will be applied.
Technical Difficulties and Challenges	Low	Medium	The project may encounter technical difficulties that may impact the timely completion of the project	Allocate enough time to do proper research on the threat modelling tools and methodology to be used.
Lack of data protection	Low	High	Non-compliance with research ethics code of practice that can result to project failure.	Data/information collected for this work will be done with consent and transparency. Also, information will be kept with anonymity and deleted at the end of project timeline.
Personal issue	Low	High	Unable to continue with project.	Project timeline have to be arranged to accommodate urgent /personal issue and if not, a request for project extension should be made from RGU School of Computing.

***Figure 3.9**: Project Risk Analysis*

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

This chapter discusses the system's design and architecture. It will cover the methodology used to develop the framework and how it is adjusted to the research objectives while taking into consideration project constraints.

4.2 DEVELOPMENT TOOLS

This section describes the software tools and general applications utilized in the development attack model and vulnerability assessment of the ABinBev System using the STRIDE shift left approach:

1. CAIRIS (Computer-Aided Integration of Requirements and Information Security):

This is a web-based platform designed for the collaborative creation of secure and privacy-respecting software systems. It offers a range of tools for security risk assessments, including threat modelling, attack trees, and security requirements elicitation. With CAIRIS, users can create a model of a system and its attack tree, identify potential vulnerabilities, and determine the potential impact of an attack as it was utilized for this research. The tool's user-friendly interface and automation capabilities make it an effective way for software development teams to integrate security into their development lifecycle and build more secure systems.

N.B: Full Installation Guide of the CAIRIS software is described in detail with screenshots in **Appendix B** section.

2. Google forms:

This is an online survey tool that allows users to create custom surveys, questionnaires, and polls. It offers a variety of question types, including multiple-choice, short answer, and rating scales, as well as customizable themes and designs. Responses are automatically collected and organized into spreadsheets for easy analysis. Google Forms is widely used in research to gather data from participants, with its simplicity and user-friendly interface making it an ideal tool for both novice and experienced researchers. Additionally, its integration with other Google tools such as Google Sheets and Google Analytics allows for further analysis and visualization of survey data.

N.B: Screenshots of both Questionnaires created using Google form can be found in **Appendix D** section.

4.3 SYSTEM PROCEDURE

This section describes the steps required to model a system using CAIRIS software.

- The first step is to create a new environment based on a case study, such as the Distributor Management System.
- Next, assets are created based on their corresponding user roles and other system components such as Azure ADB2C, Database Server, Users, etc.
- The third step involves creating user roles and their dependencies, including the Administrator, KPO, VSM and BB.
- The fourth step is to create personas and attackers which are pseudo individuals assigned to the roles on the platform with unique attributes like activities, attitudes, skills, etc.
- Processes were then created in the form of use cases that manage each user activity based on their role on the platform.
- Next was Creating dataflows to link entities (assets) to the processes (use case) and Datastore (Database server).
- Finally, creating trust boundaries that categorize all processes and entities associated together.

N.B: Each of these steps is described in detail with examples and screenshots provided in the **Appendix C** section.

4.3.1 Create and modelled a standard Data Flow Diagram for the case study system.

The DFD to be used for the case study was created next. The data flow diagram is an especially important feature shared by STRIDE framework and CAIRIS for eliciting theat. STRIDE is well known for its supports for flow of information and threat modelling activities that is good to assessing security risks (Cole, Faily and Ki-Aries 2018). Hence, in a system when data flows are tainted, the safety of the data in transit become questionable thus, with human entities and processes a taint can occur due to both the human and the system flow error and when this occur, an endpoint can be impacted resulting in further enquiry (Faily et al 2020).

In CAIRIS, DFD process are analogous with use cases that has its actors being represented with human or system entities. In tasks context, DFDs can link to usability model and can also be indirectly linked when it is a role context that can be fulfilled by personas (Faily 2022). The Distributor Management system DFD model can be shown in figure 4.1 and more information on DMS data flow can be found in chapter 3 of the report.

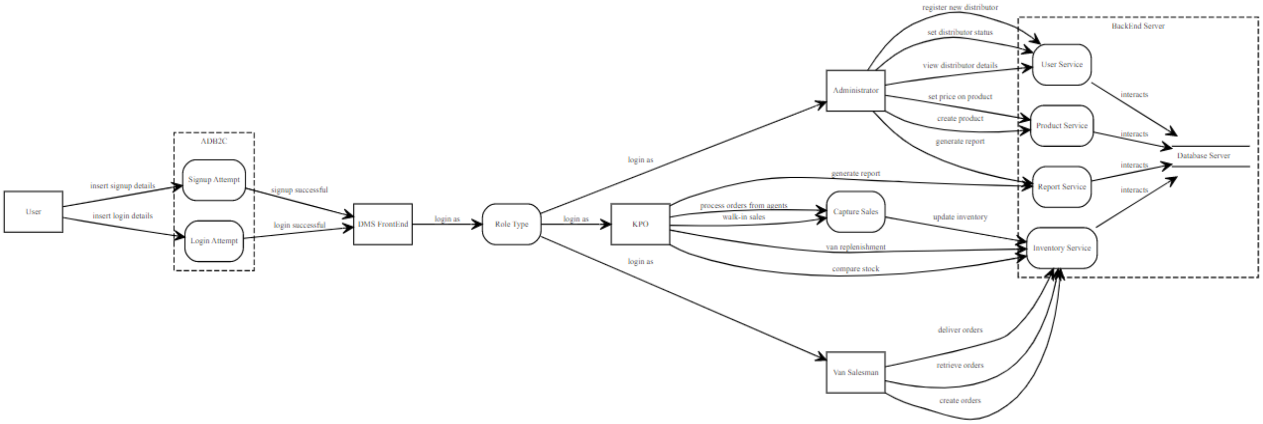


Figure 4.1: *DFD for Distributor Management System*

4.4. ATTACK TREE ASSESSMENT

This section describes how the modelled system is tested on CAIRIS. It depicts how the study objective is achieved through the modelling of the attack tree using the STRIDE approach. These attacks are mapped on the DMS system as modeled by the DFD model as described in section 4.3.1. To model the attack tree, several steps are taken as outlined and discussed below:

4.4.1 Create and Model Task Diagram

The tasks were created after the DFD which depicts the different functions conducted by the individual roles of the application. It gives a broad overview of how the functions shown in the DFD are linked to the separate roles and the processes. Figure 4.2 shows the modelled tasks defined for the DMS environment together with their individual personas.

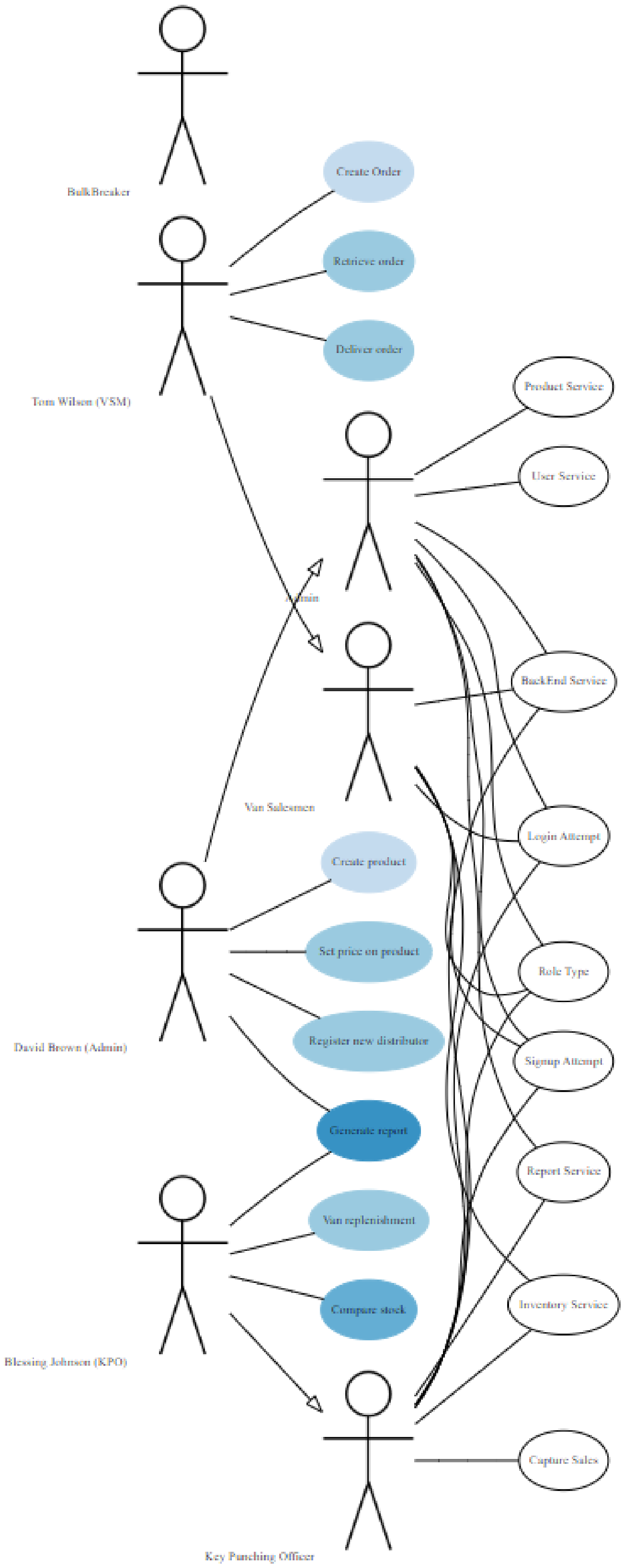
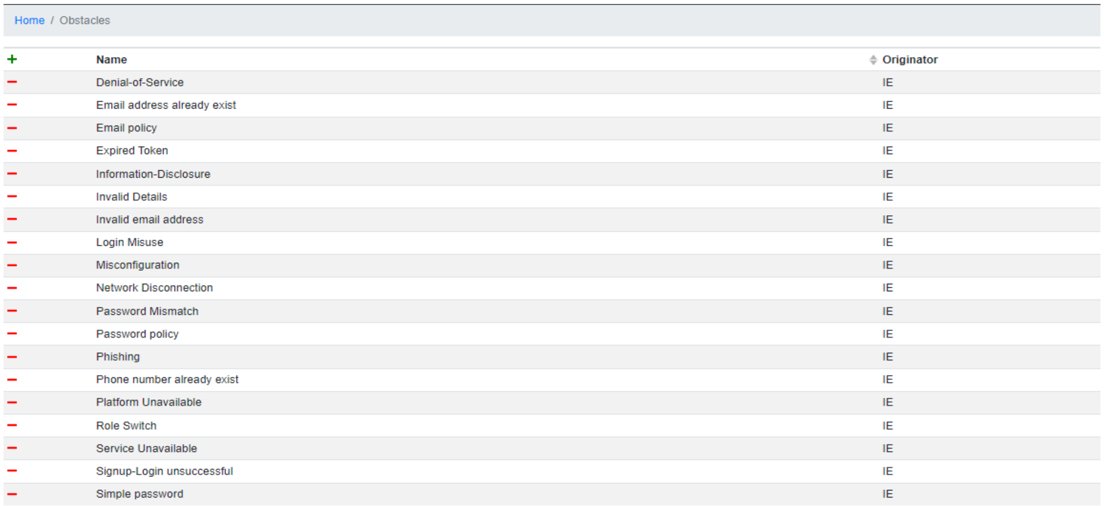


Figure 4.2: *Task model for Distributed Management System*

4.4.2 Create Obstacles

Obstacle can be referred to as a condition that represent undesired behaviour which when refined can reflect a vulnerability or a threat that help in risk analysis (Faily 2022). Thus, the information about the obstacle for this system can be found below in figure 4.3. As mentioned earlier, the obstacle and obstacle model generated by CAIRIS will be achieved using the selected threat trees of elicited threats from STRIDE framework.



Home / Obstacles

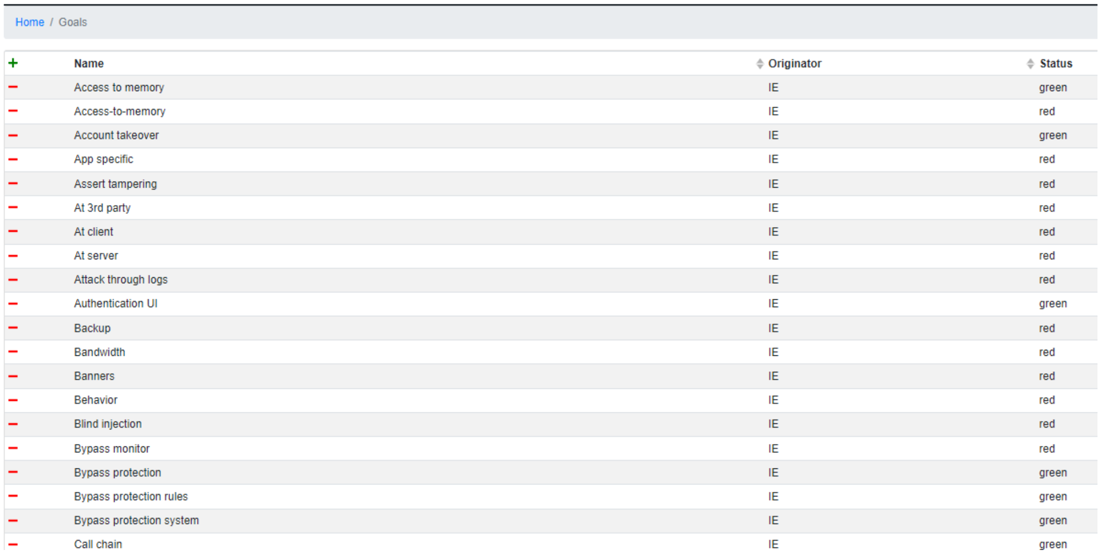
+	Name	Originator
−	Denial-of-Service	IE
−	Email address already exist	IE
−	Email policy	IE
−	Expired Token	IE
−	Information-Disclosure	IE
−	Invalid Details	IE
−	Invalid email address	IE
−	Login Misuse	IE
−	Misconfiguration	IE
−	Network Disconnection	IE
−	Password Mismatch	IE
−	Password policy	IE
−	Phishing	IE
−	Phone number already exist	IE
−	Platform Unavailable	IE
−	Role Switch	IE
−	Service Unavailable	IE
−	Signup-Login unsuccessful	IE
−	Simple password	IE

Figure 4.3: *Obstacles Page*

4.4.3 Create Attacks (Goals) and KAOS Associations that linked the attacks/obstacles to the parent tree.

In CAIRIS, goals can be used to understand how it can contribute to other goals. Thus, it can be used to identify new requirement, threat or vulnerability which can occur from satisfied or denied goals. In addition, it can be used to represent the different form of attacks and the associations of those attacks to the parent attack which in this case is the STRIDE methodology (Faily 2022).

Thus, the suggested and assumed list of possible attacks relating to the goals of the studied system adapted from CAIRIS (2022) exemplar and its model can be found in figure 4.4 below.



Home / Goals

+	Name	Originator	Status
−	Access to memory	IE	green
−	Access-to-memory	IE	red
−	Account takeover	IE	green
−	App specific	IE	red
−	Assert tampering	IE	red
−	At 3rd party	IE	red
−	At client	IE	red
−	At server	IE	red
−	Attack through logs	IE	red
−	Authentication UI	IE	green
−	Backup	IE	red
−	Bandwidth	IE	red
−	Banners	IE	red
−	Behavior	IE	red
−	Blind injection	IE	red
−	Bypass monitor	IE	red
−	Bypass protection	IE	green
−	Bypass protection rules	IE	green
−	Bypass protection system	IE	green
−	Call chain	IE	green

Figure 4.4: *Goals Page*

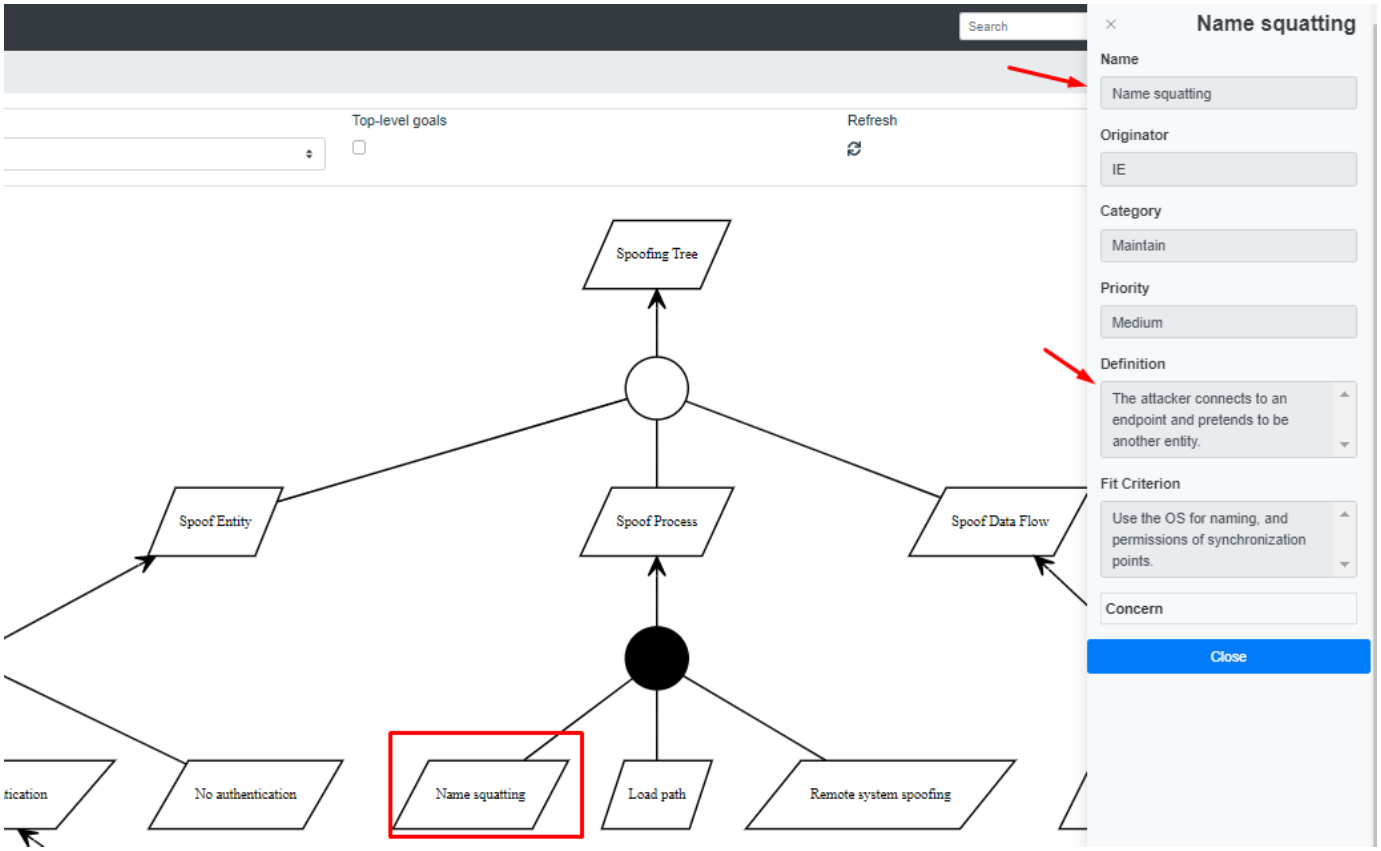
4.5 CREATION OF ATTACK TREE MODEL USING THE STRIDE METHODOLOGY

An attack tree is a visual representation of potential attacks against a system, organized in a tree-like structure.

This attack tree is modeled from these goals listed by selecting the **Models** dropdown menu, then select **Goals**. This will automatically generate the attack tree based on the specifications of each goal listed in figure 4.4 above. The attack generated in this research uses the shift- left STRIDE approach and it generates an extensive attack tree. This tree will be illustrated in the subsequent sections to highlight the various attacks modeled on the platform. Each attack corresponds to the attacks listed in table 3.3 which include spoofing, tampering, repudiation, information disclosure, denial of services, and elevation of privilege.

Each attack is modeled based on DFD object type which are entities, data flows, processes, and datastore. Table 3.3 shows which attack affects which object. The attack is also developed with some repetitions to avoid a complex intertwined model illustration. Therefore, certain attacks may be duplicated on the attack tree for the purpose of modelling simplicity e.g., Weak or no Channel Integrity. It should be noted that each attack (goal) on the attack tree can be managed as in the figure 4.5 below which shows the properties of insufficient authentication attack under the spoofing of entity types.

Below is the depicted model of all the possible attacks from the attack tree for a distributor management system application using the STRIDE methodology:



Name squatting

Name: Name squatting

Originator: IE

Category: Maintain

Priority: Medium

Definition: The attacker connects to an endpoint and pretends to be another entity.

Fit Criterion: Use the OS for naming, and permissions of synchronization points.

Concern

Figure 4.5: *Attack Details Module*

4.5.1 Spoofing Attack Model:

This attack tree models spoofing attacks that can be executed on the DMS as modeled in the DFD model in figure 3.8. This spoofing attack occurs at three out of the four DFD objects which are the entities, dataflows, and process. The figure below shows the accurate model of spoofing attacks on the DMS platform.

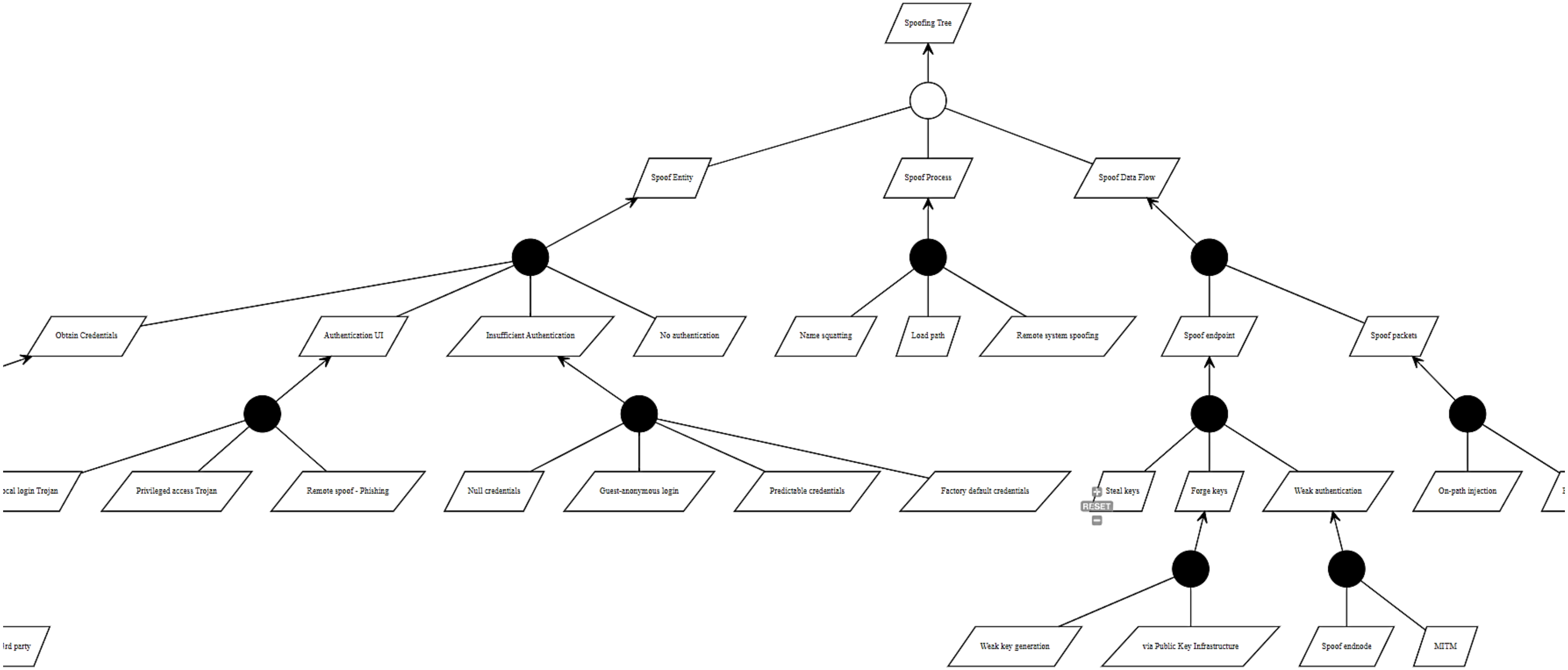


Figure 4.6: *Spoofing Attack Tree Page*

4.5.2 Tampering Attack Model:

The tampering attack affects three DFD objects which are the processes, data stores and the dataflows. This attack tree is quite extensive and requires two images to illustrate the full breath of the attack. Figure 4.7 illustrates the left side of the tampering attack and figure 4.8 illustrates the right side of the tampering attack tree. This attack tree models all sorts of tampering that may be used to exploit and gain unauthorized access to the system. The attack tree shows how the spoofing attack is likely to affect processes, datastore and dataflows.

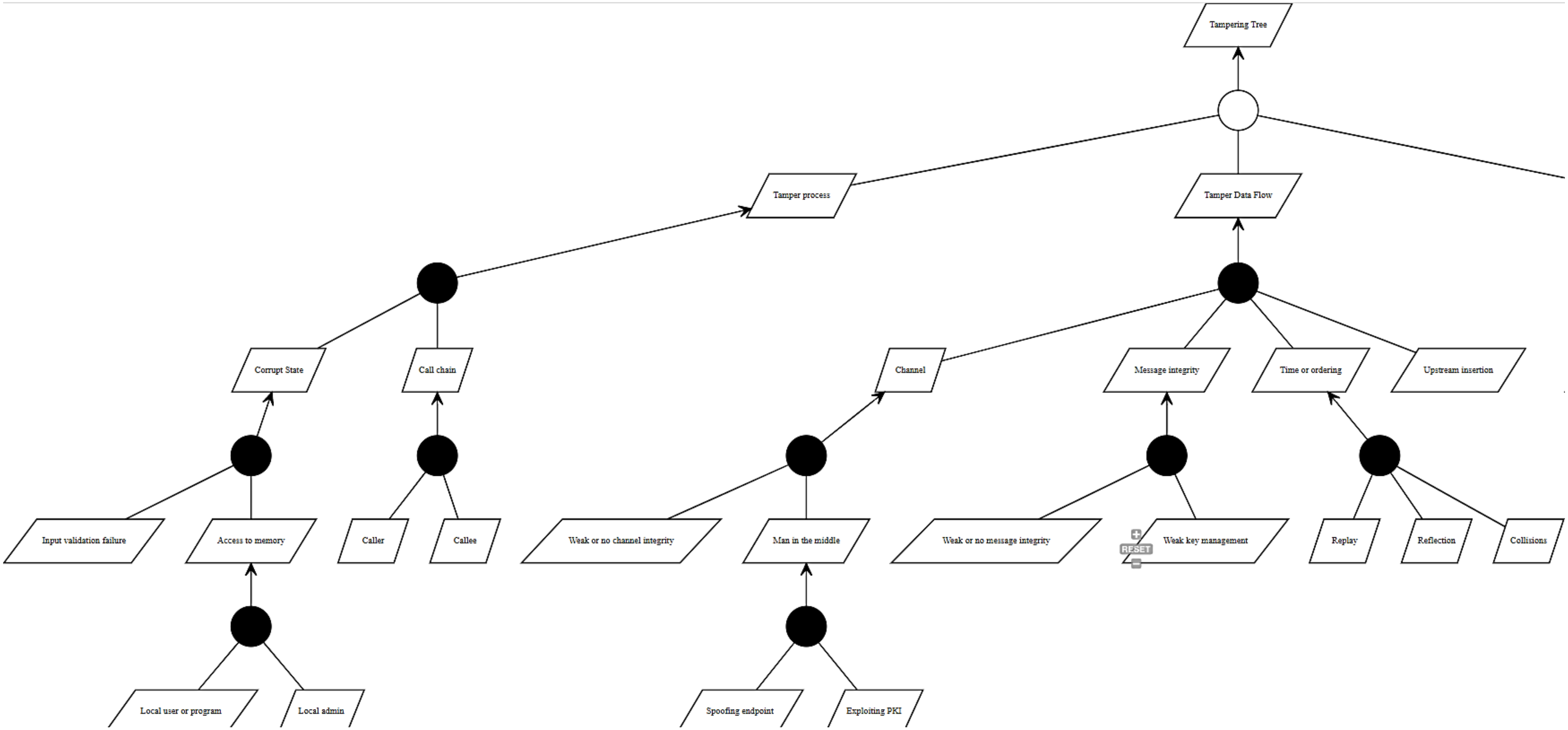


Figure 4.7: *Tampering Attack Tree - left.*

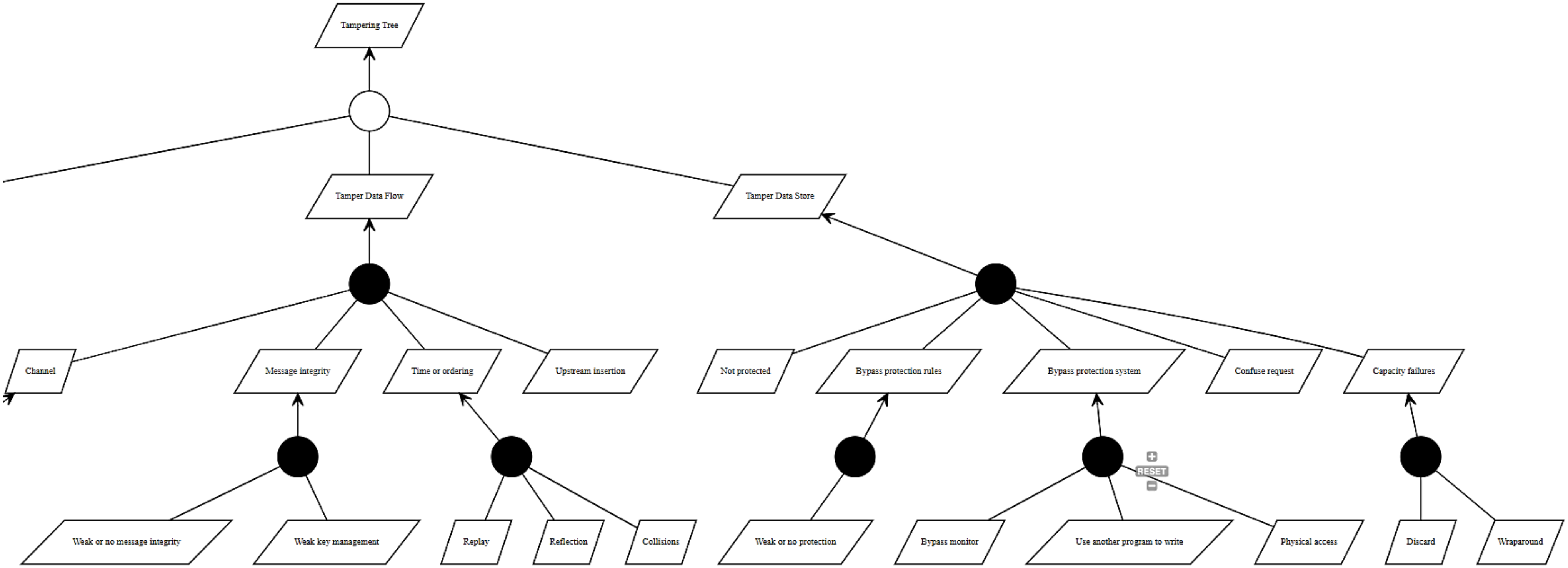


Figure 4.8: *Tampering Attack Tree - right.*

4.5.3 Repudiation Attack Model:

The repudiation attack models intrusions aimed at acquiring control of the system to manipulate transaction or system logs with the aim of damaging the data integrity of the system. The repudiation attack tree as shown in figure 4.9 shows that the attacks affect the datastore and processes. In this context, the effect of the repudiation attack is synonymous with attacks on system entities (services e.g., web app service).

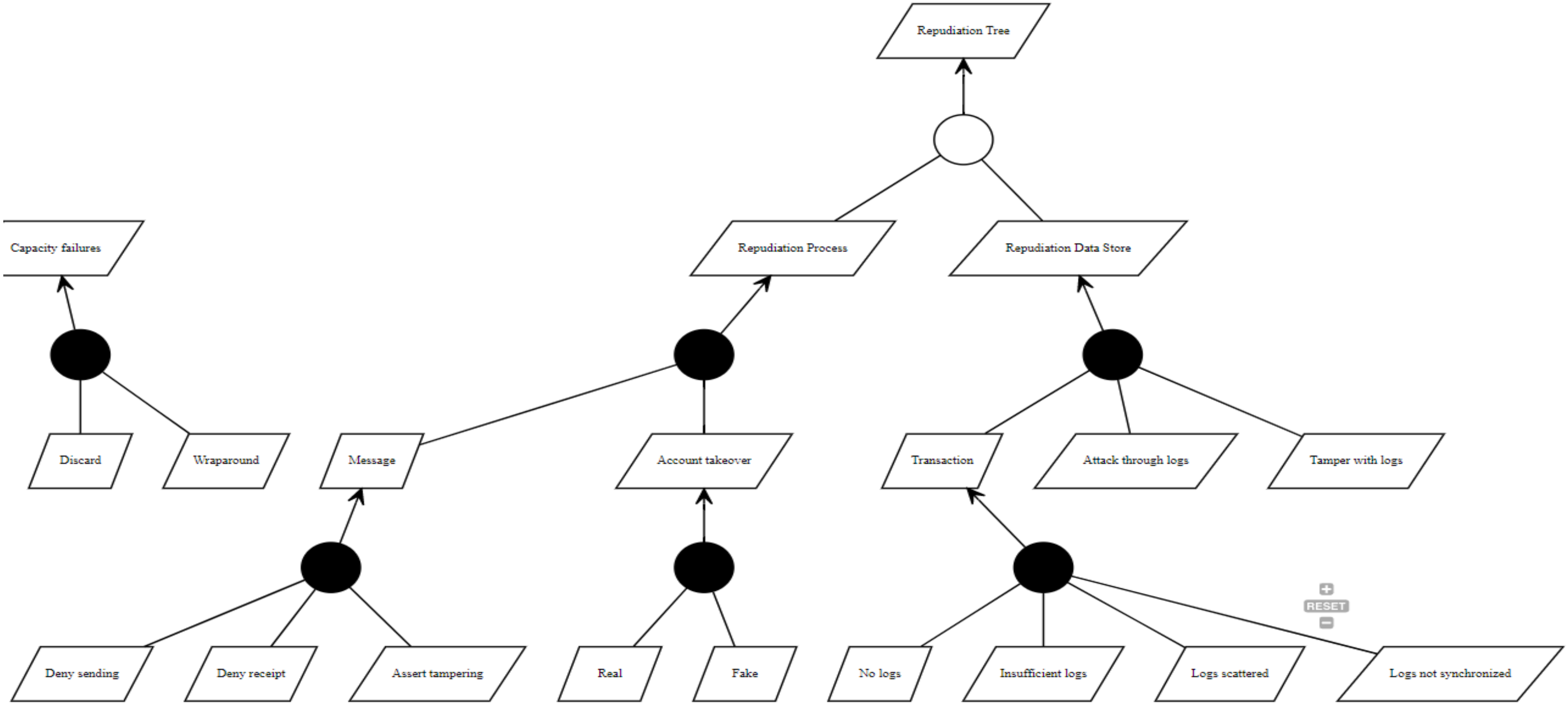


Figure 4.9: *Repudiation Attack Tree Page*

4.5.4 Information Disclosure Attack Model:

This attack model affects the datastore, processes and dataflow. This attack models the various way attacks may be utilized to access and alter sensitive data with the aim of exploiting the organization with said information. Figure 4.10 below illustrates information attack model of the DMS environment modeled in CAIRIS.

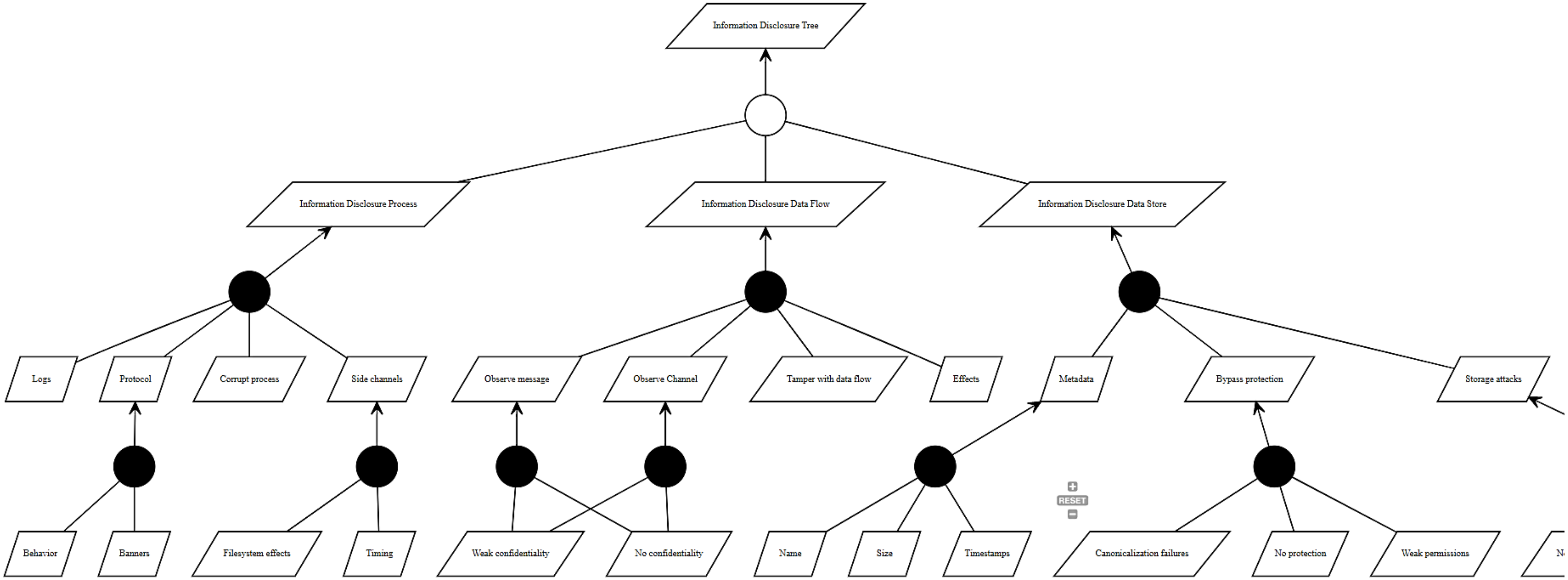


Figure 4.10: *Information Disclosure Attack Tree Page*

4.5.5 Denial of Service Attack Model:

This attack tree models the weak points of the system that can be exploited using the DoS attack. This attack affects three DFD objects which are the processes, the dataflows and the datastore. The implication of a successful attack is the attack will overload the system while accessing it for malicious intent. Figure 4.11 below illustrates the Denial-of-Service attack model for the DMS environment.

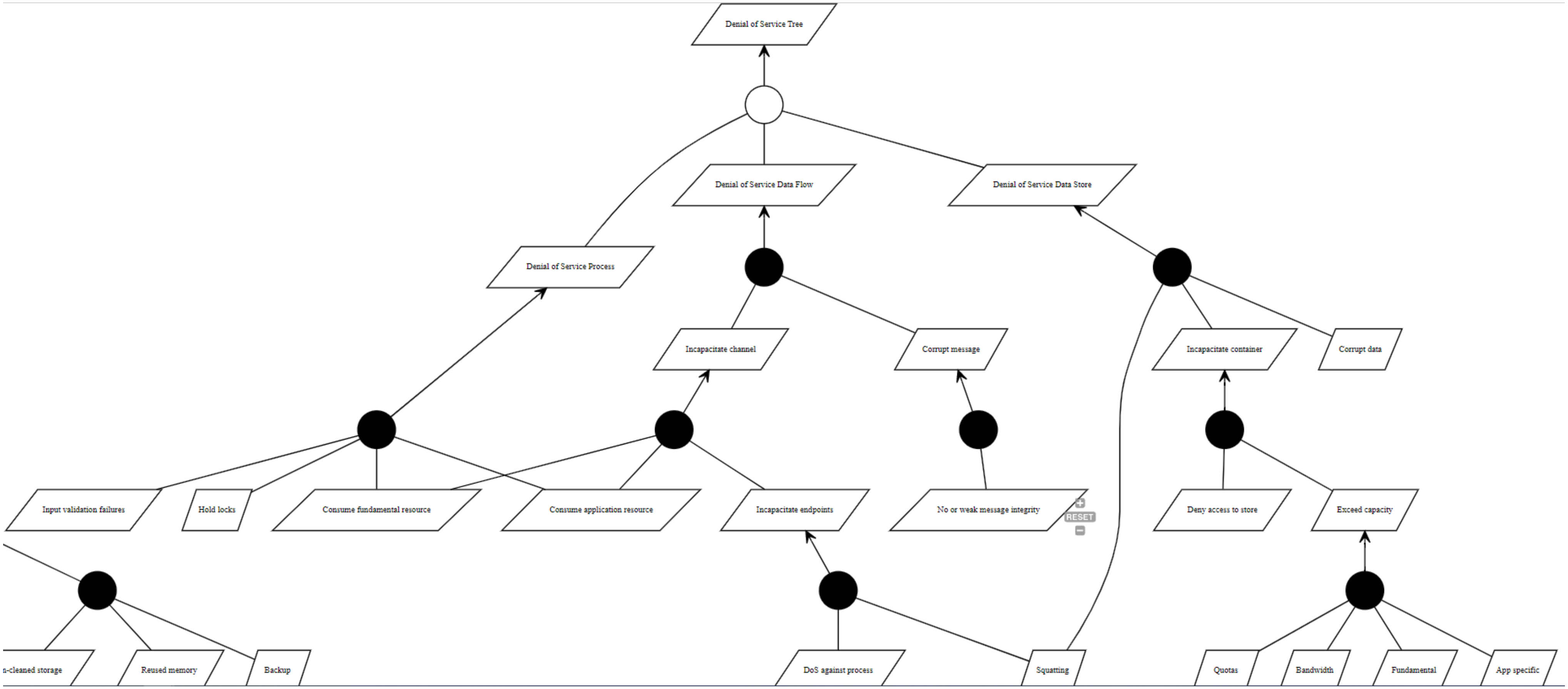


Figure 4.11: *Denial of Service Attack Tree Page*

4.5.6 Elevation of Privilege Attack Model:

A platform such as DMS which interacts with different actors provides access control to the information on the platform. Depending on the development architecture each actor type might be different on a microservice level, wherein, the system services that tend to a KPO may not be the same service that tend to an admin, even though these two services may have the same function. To overcome this challenge, the attacker may seek to elevate their privilege to gain more access and control of the system. This attack tree models the various aspects of the DMS platform that could be exploited for this attack as shown in figure 4.12 below.

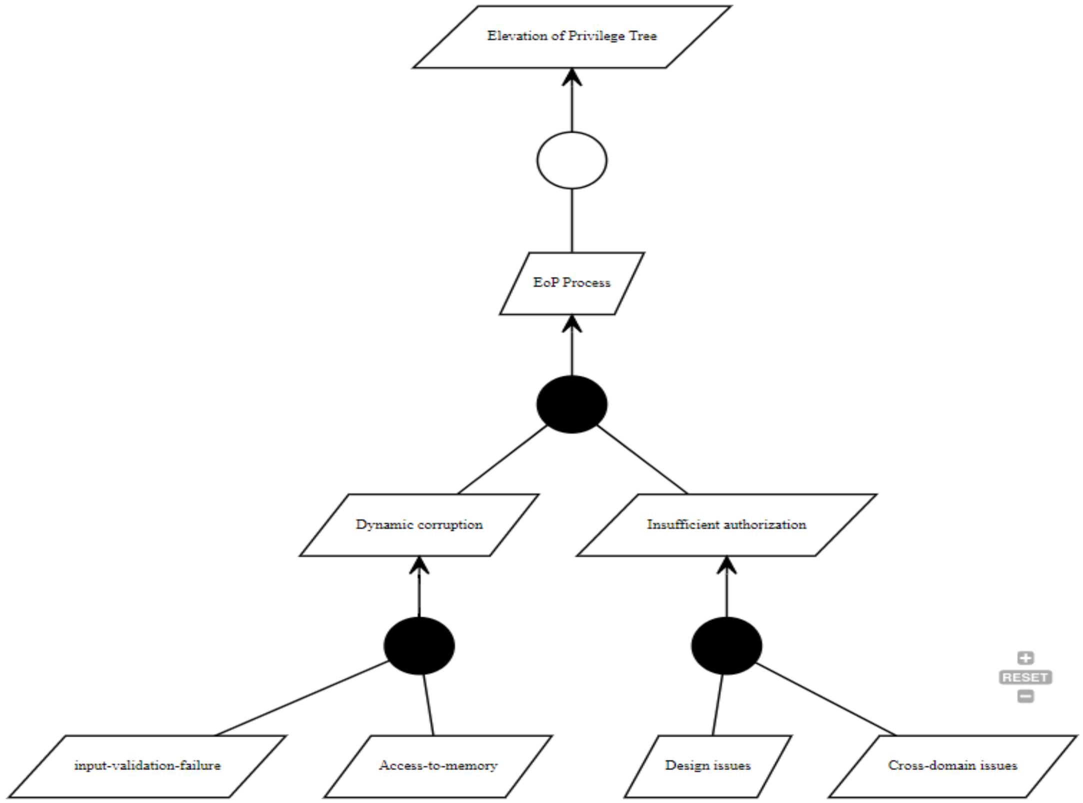


Figure 4.12: *Elevation of Privilege Attack Tree Page*

4.6 CHALLENGES ENCOUNTERED

The CAIRIS platform is a well-designed and developed web-based threat modelling platform. The challenge experienced in this research was understanding the CAIRIS functionality using the document. This was a learning curve, however, with practice shortcuts were discovered and the system became more familiar with usage.

This research also develops a questionnaire which evaluates the likelihood of attacks from vulnerabilities of the DMS based on responses from users of the platform and the systems maintenance manager who understand the inner workings of the application. Two questionnaires had to be developed, one for the technical manager and the other for users of the platform. Distributing the questionnaire was quite challenging because of the respondent target number for this research and time available to gather this research. Using the right media channel after several attempts proved optimal as 60% of the respondents were able to engage the user question in 20% of the time utilized for gathering responses.

4.7 QUESTIONNAIRE ASSESSMENT

The Google forms was used to develop a questionnaire for users of the DMS platform to ascertain highest points of vulnerability in the current system, to ascertain the likely areas of cyberattack based on the attack tree modelled in CAIRIS. This tool was also used to develop the interview questionnaire for the lead IT of the DMS platform which provides deeper insights into the architecture and operations of the platform.

The application was presented to a group of the organization's staff randomly with the goal of collecting some informal evaluation and usage of the application. The aim of this was to compare and analyze the likelihood of the attacks from the attack tree, implement necessary changes to the application if necessary and come up with an updated architecture for the system.

4.7.1 First Questionnaire - Vulnerability Assessment of DMS

The test was carried out with over 80 staff using the application whose names are withheld for ethical reasons. The test was done by providing the volunteers a google form link to a questionnaire fill out about their experience.

The first questionnaire was divided into 3 sections:

- **Identity Verification**: This section was needed to identify staff role and experience at the organization and with the DMS application, both web and mobile.

- **General Questions**: This section ask about challenges experienced from using the KUJA mobile and web application.
- **Role-Specific Questions**: This section pertains to the unique challenges to the users based on their role type.
    - Admin
    - Mini Admin
    - Van Salesman

4.7.2 Second Questionnaire - Technical Security Assessment of DMS

The second questionnaire was sent to the team lead of the DMS platform which provided deeper insights into the architecture and operations of the platform. The IT team lead understands the core technical operations of the web and mobile platform. He is farmilar with the architecture and process flow, and is able to answer any questions pertaining to the software and its host environment.

N.B: Screenshots of both Questionnaires created using Google form can be found in **Appendix D** section containing well detailed information about the questions asked and responses gotten.

4.8 QUESTIONNAIRE INSIGHTS

4.8.1 First Questionnaire Insight

Out of the 80 volunteers that filled out the survey,

- 20% are Admin (16)
- 21.2% are Mini-Admin (17)
- 33.8% are KPOs (27)
- 25% are Van Salesman (20)

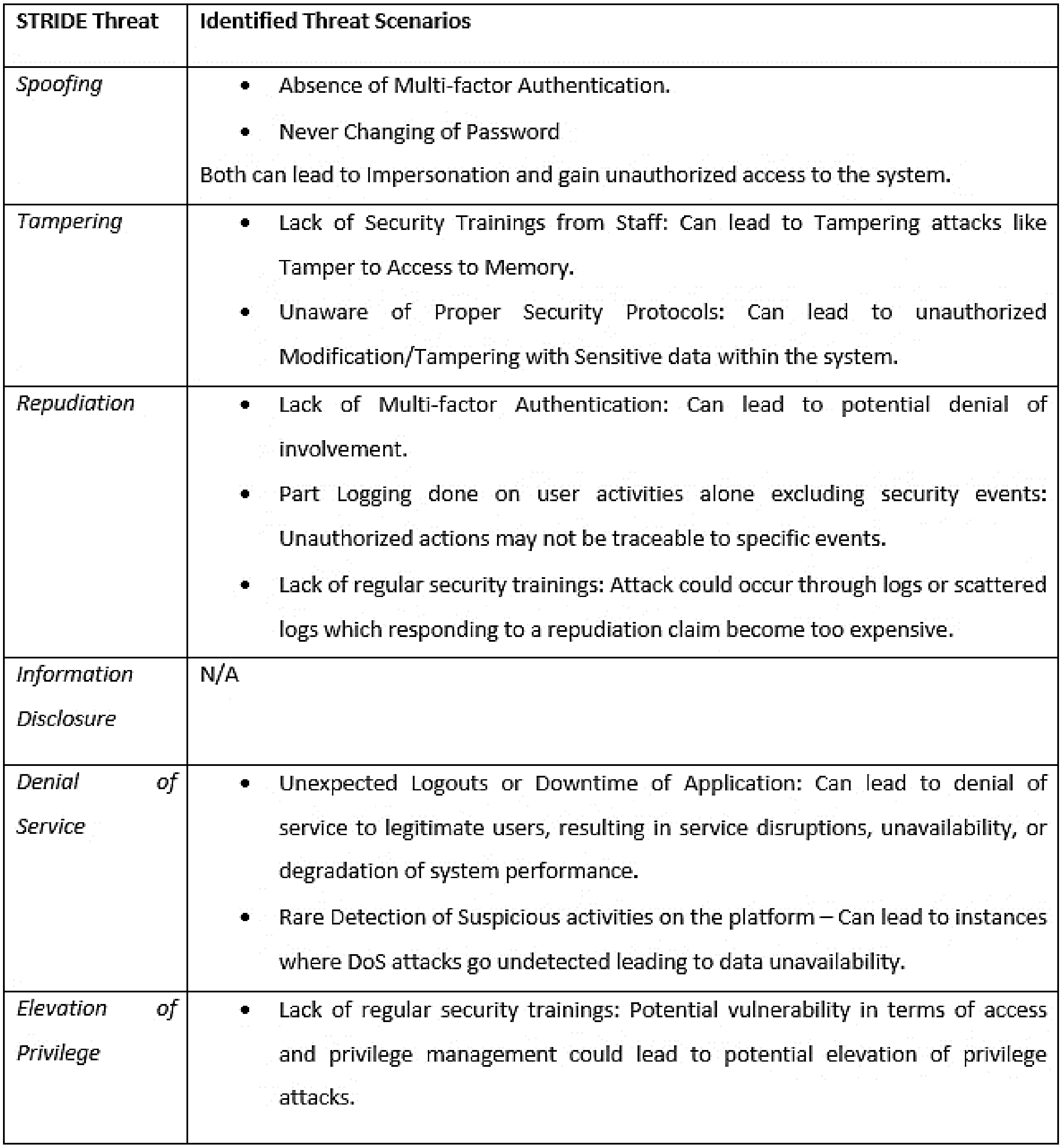
Based on the insights gotten from the questionnaire and responses obtained, potential areas of security vulnerabilities in the DevOps methodology using the "shift left" approach via threat modelling can include authentication and access controls, password security, system performance and stability, user awareness and training on security best practices, and role-specific vulnerabilities. These areas would be further evaluated and addressed using the STRIDE methodology to identify potential threats and vulnerabilities in each category and implement appropriate security measures to mitigate the risks. This would be done by comparing and analysing the technical questionnaire sent to the technical lead of the application.

4.8.2 Further Insight on Both Questionnaires Using STRIDE Approach

Based on the responses provided in the technical questionnaire together with the initial analysis gotten from the first questionnaire, A summarized insights that can be gathered using the STRIDE methodology:

The section evaluates the KUJA/DMS system's security using the STRIDE methodology based on responses provided in two questionnaires. The evaluation considers potential vulnerabilities associated with spoofing, tampering, repudiation, and information disclosure attacks. The absence of multi-factor authentication for accessing the web and mobile platforms creates a loophole for spoofing and repudiation attacks. Lack of regular security policy training increases the risk of tampering and repudiation attacks on data store. The evaluation also indicates that adequate monitoring and encryption of sensitive data during transmission reduces the risk of information disclosure attacks. The system's logging of user activities can stop attacks relating to spoofing of processes, while proper logging and constant training of security protocols can avoid attacks related to information disclosure process.

N.B: Both insights are described and explained in further and deeper detail with examples and screenshots provided in the **Appendix E** section.



STRIDE Threat	Identified Threat Scenarios
Spoofing	• Absence of Multi-factor Authentication. • Never Changing of Password Both can lead to Impersonation and gain unauthorized access to the system.
Tampering	• Lack of Security Trainings from Staff: Can lead to Tampering attacks like Tamper to Access to Memory. • Unaware of Proper Security Protocols: Can lead to unauthorized Modification/Tampering with Sensitive data within the system.
Repudiation	• Lack of Multi-factor Authentication: Can lead to potential denial of involvement. • Part Logging done on user activities alone excluding security events: Unauthorized actions may not be traceable to specific events. • Lack of regular security trainings: Attack could occur through logs or scattered logs which responding to a repudiation claim become too expensive.
Information Disclosure	N/A
Denial of Service	• Unexpected Logouts or Downtime of Application: Can lead to denial of service to legitimate users, resulting in service disruptions, unavailability, or degradation of system performance. • Rare Detection of Suspicious activities on the platform – Can lead to instances where DoS attacks go undetected leading to data unavailability.
Elevation of Privilege	• Lack of regular security trainings: Potential vulnerability in terms of access and privilege management could lead to potential elevation of privilege attacks.

Figure 4.13: *DMS Identified threat scenarios.*

4.8.3 Summary of Existing Attack Loopholes in DMS Using Stride Methodology

Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Entities	X		X			
Processes	✓	X	X	✓	X	X
Data Flows	✓	✓		✓	X	
Data Stores		X		✓	✓	

***Table 4.1**: Tabular Summary of Existing Attack Loopholes in DMS using STRIDE Methodology*

4.9 RECOMMENDATIONS

To address the potential security threats identified in the analysis, here are some recommendations:

1. **Spoofing**: To address the risk of spoofing attacks, it is important to implement strong authentication mechanisms, such as multi-factor authentication, to ensure that only authorized users can access the system. Additionally, measures like digital signatures and certificate authorities can help to prevent spoofing attacks.

2. **Tampering**: To prevent tampering attacks, it is important to implement proper access controls and monitoring mechanisms to ensure that only authorized users can make changes to the system or data. Encryption can also be used to protect data both at rest and in transit, and digital signatures can help to detect any tampering attempts.

3. **Repudiation**: To prevent repudiation attacks, it is important to implement proper logging and auditing mechanisms to record all user activities and security events. This can help to detect any suspicious activity and provide evidence in case of any disputes or legal issues.

4. **Information Disclosure**: To prevent information disclosure attacks, it is important to implement proper access controls and encryption mechanisms to protect sensitive data both at rest and in transit. Additionally, regular security reviews and penetration testing can help to identify and address any vulnerabilities in the system.

5. **Denial of Service**: To prevent denial of service attacks, it is important to implement measures such as firewalls, load balancers, and intrusion detection systems to monitor and block any suspicious traffic. Additionally, implementing proper rate limiting and throttling mechanisms can help to prevent overload of system resources.

6. **Elevation of Privilege**: To prevent elevation of privilege attacks, it is important to implement proper access controls, privilege management, and role-based access mechanisms to ensure that only authorized users have the necessary permissions to perform specific actions. Additionally, regular security reviews and penetration testing can help to identify and address any vulnerabilities in the system.

Overall, it is important to implement a multi-layered security approach that includes both preventive and detective measures, as well as continuous monitoring and improvement of security mechanisms. This can help to minimize the risk of security threats and protect the confidentiality, integrity, and availability of the system and data.

CHAPTER 5: EVALUATION

5.1 INTRODUCTION

This chapter covers the discussion on the results of the project implementation, analysis of the results, comparing its achievement with the objective declared in chapter one, feedback from stakeholder, validation, and future work.

5.2 CRITICAL ANALYSIS OF DESIGN OUTCOMES

A case study was successfully carried out and the necessary data required to design security threat modelling for the studied system successfully implemented using STRIDE framework. Thus, the security threat modelling of STRIDE which are DFD, and attack threat tree were used in this design to elicit for possible threats in the system. Furthermore, to generate a robust documentation report that can be used by stakeholders in making decisions to the security architecture of the system to prevent occurrence of threat on the system, CAIRIS was adapted as a tool-support. With CAIRIS, the studied system prototype was able to be presented in visual models that include:

- ***Asset and asset modelling**:* The studied system asset was successfully modelled, and the visual presentation make it easy to see the relationship among the asset, to detect weak links and to identify assets that are vulnerable as they were identified with color code. The darker the color the greater the possibility of vulnerabilities or threat. However, assets modelled in this section were partly adapted from CAIRIS exemplar because of their similarity as IT equipment. Thus, the lesson learned in this area of project work indicates the need to have a thorough understanding of how to be able to identify the appropriate assets of the system being modelled which might be hard to be done as an individual without seeking the opinion of the stakeholder.

- ***Goals and goal model**:* This model was successfully used to visualize all possible obstacles/vulnerabilities in the way of the successful achievement of the system goal. It has the same function as an obstacle threat tree and was used to successfully implement elicited threat trees in STRIDE framework. With goal model, threats and vulnerabilities, relationship could be modelled; and the model can make it easier to visually identify any threat and vulnerability in the system. However, due to limited knowledge on threat modelling, the STRIDE threat trees that were used in this project work were similar threat trees adapted from the work of Shostack (2014). These adapted threat trees were pruned as needed to fit the required threat trees for the studied system.

- ***DFD model**:* DFD is one of the common features between STRIDE framework and CAIRIS. The DFD model was generated in CAIRIS by feeding the details from the system's architecture into its data flow function, the trust boundary was fed into the trust boundary section. The outcome of the entry generated a DFD model that can be visualized to easily identify the relationship between the DFD elements and any compromised event that could occur on the system can easily be elicited. This modelling was successfully implemented as well. Although the CAIRIS exemplar is very useful as a guide in this aspect, there is a need for more practice for better understanding on how to carry out the replicate of DFD on CAIRIS.
- ***Task and Task modelling**:* The system task was applied on CAIRIS with use case and a task model was successfully generated from which any existing vulnerabilities on the system with respect to a specific role can be identified. Thus, this modelling was successfully done.

5.3 STAKEHOLDER FEEDBACK

Feedback from the stakeholder in the aspect of the result on envisaged threats such as spoofing and possibility for denial-of-service attack that could occur due to the users' lack of training and awareness on threat issue like never changing of passwords was highly welcomed by the senior staff. Based on this, a notification on improvement towards regular security training as well as enforcing the logging of both user and system activities was shared on the group Microsoft Team. The notification include:

(i) The immediate implementation of strong authentication mechanisms, such as multi-factor authentications to ensure that only authorized users can access the system.

(ii) Weekly training courses and security newsletters will be held and sent to staff for regular upskill and create security awareness among them. Attendance is said to be mandatory.

(iii) Weekly reminders will be sent to staff reminding them to always change their passwords regularly and make use of strong passwords.

(iv) Implementation of multi-layered security approach that includes both preventive and detective measures as well as continuous monitoring to help combat and minimize the risk of security threats and protect the confidentiality, integrity and availability of the system and data.

(v) To further address these issues at the organization next "Your Say" which is a monthly forum for voicing concern of any issue that can impact the organization both negatively and positively for future improvement.

5.4 COMPARISON OF ACHIEVEMENT WITH PROJECT OBJECTIVES

This section can be used to reflect on the project achievement by comparing it with the project objectives stated earlier in the project. The comparison will be done by stating whether the objectives have been achieved or not as listed below:

1. *Investigate an existing system, conduct interviews and collect necessary data towards designing of threat modelling to improve the system security.*

   This objective has been achieved; DMS has been chosen, the case study was conducted, and information required for designing its security threat modelling has been collected and applied using STRIDE and CAIRIS.
2. *To build a solution that models the entirety of the case study system – including all assets and processes.*

   This objective has been achieved; CAIRIS was successfully utilized for the implementation of the solution modelling system. This was done through:

   - Analyze all qualitative data gathered from both case study and interviews to identify best practices and potential challenges of integrating threat modelling into DevOps processes.
   - Utilizing CAIRIS as the threat modelling tool to model the system.
3. *To find and address threats in the solution.*

   This objective has been achieved; Finding and Addressing Threats was achieved by:
   - The STRIDE attack tree was used to find all possible threats linked to the case study.
   - Creating two different questionnaires used to get likelihood of all possible threats and compare with the attack tree done on CAIRIS.
   - Analyze responses from both questionnaires and get the likelihood of all possible kinds of threats that exist in the system.
   - Compare threats with the STRIDE attack tree done on CAIRIS to eliminate non-consequential vulnerabilities and threats.
   - Provide recommendations on how to secure the existing system.
4. *To explore the challenges of modelling threats and potential strategies for addressing those challenges.*

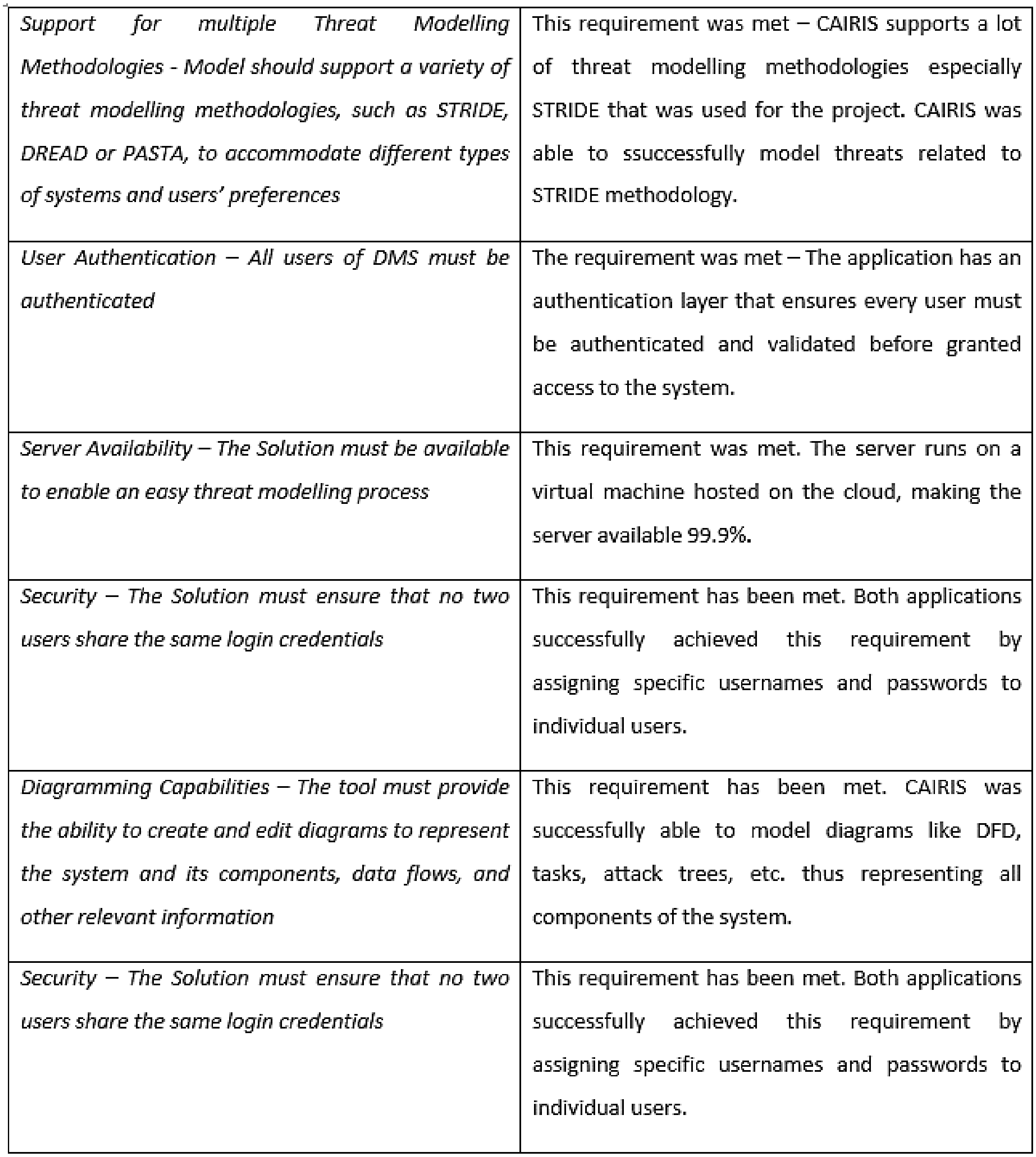
   This objective was achieved to some extent. Challenges surrounding the threat modelling on CAIRIS, and distribution of the questionnaires were discussed. However, the potential strategies to address those challenges were discussed in future works.

N.B: - Important to note that all LESP issues were strictly adhered to as per Section 3.7.

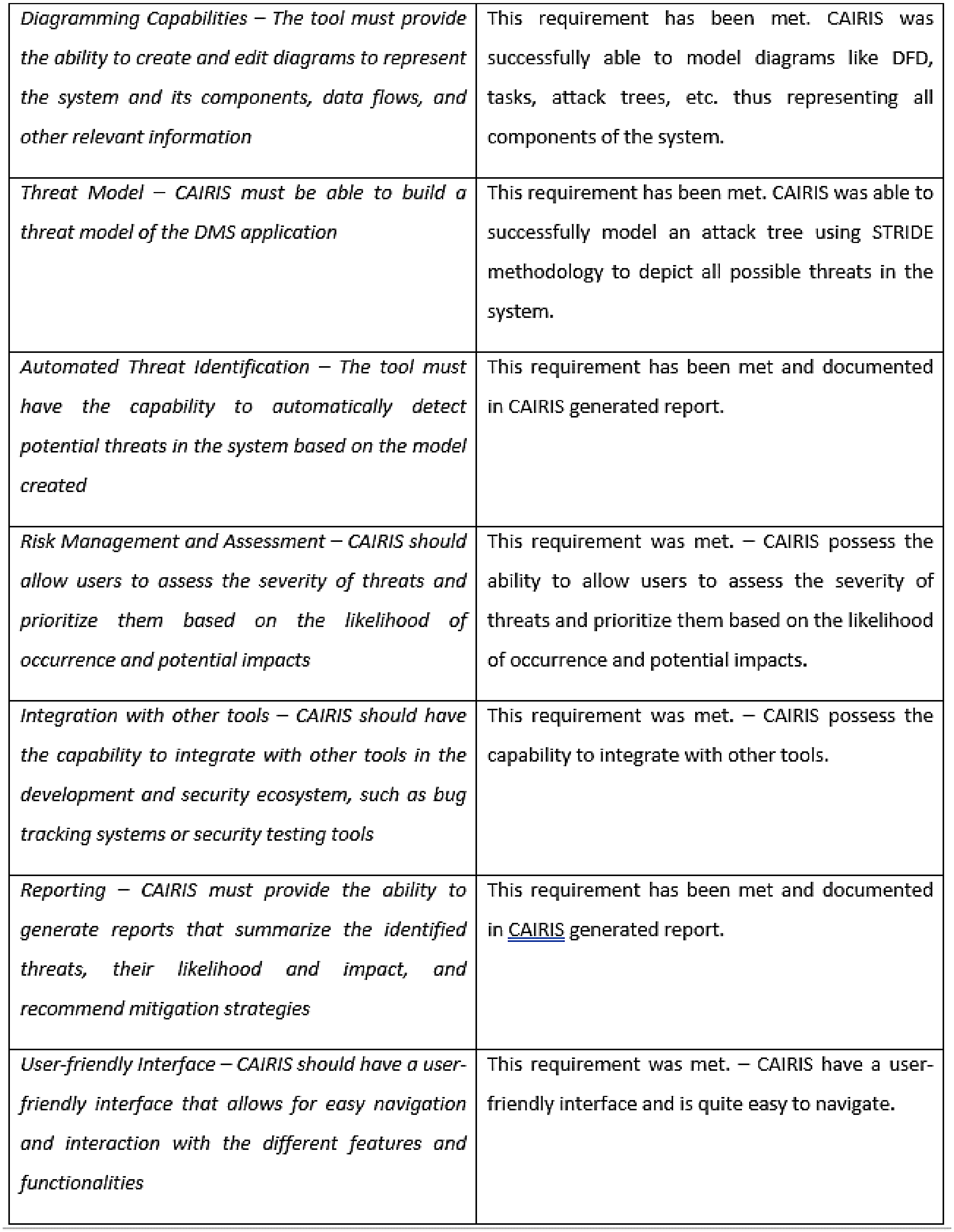
5.5 COMPARISON OF ACHIEVEMENT WITH PROJECT REQUIREMENTS

In this section, we will discuss the functional and non-functional requirements of the tool-support system used in this project. These can be described as follow:

5.5.1 Functional Requirements:



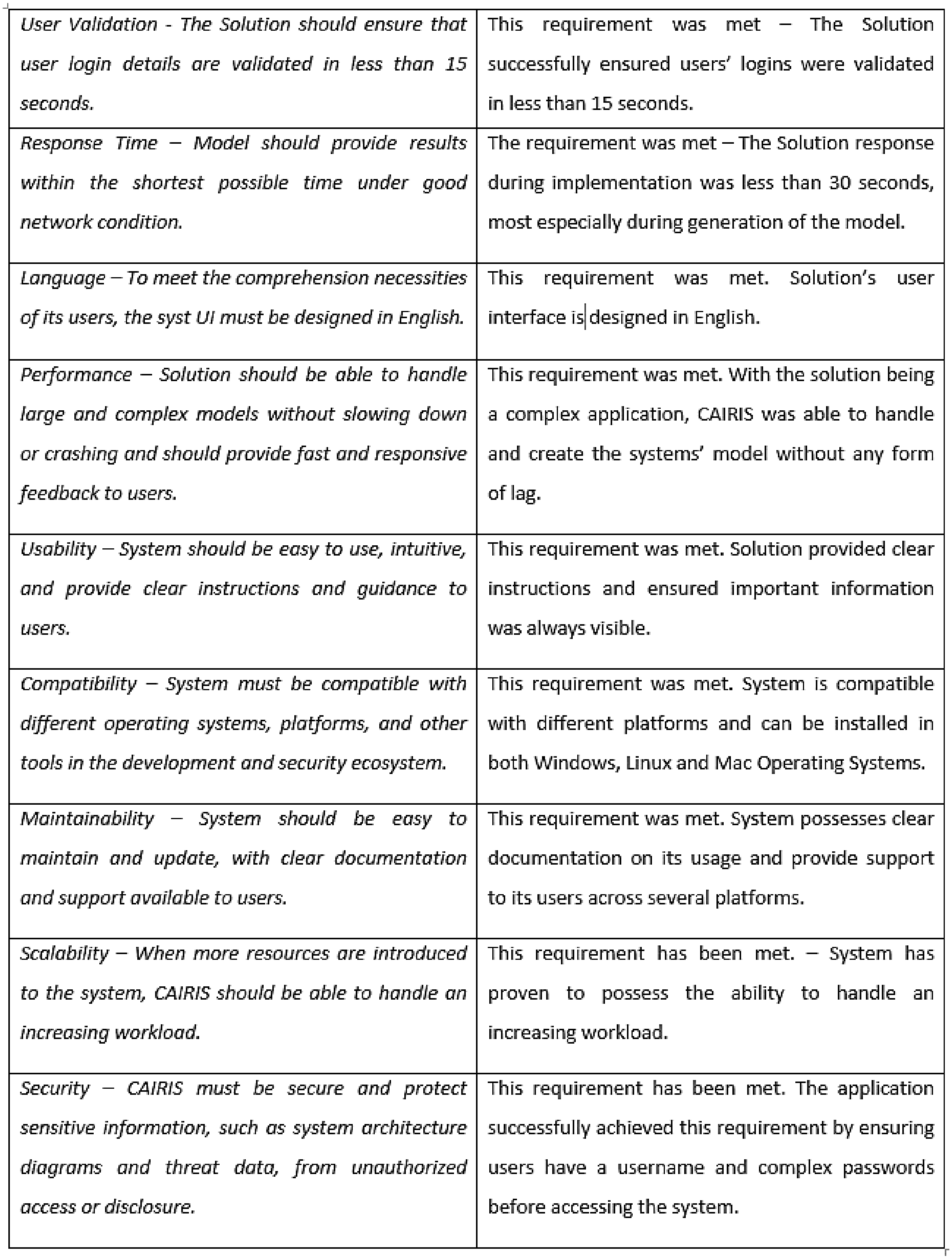
Support for multiple Threat Modelling Methodologies - Model should support a variety of threat modelling methodologies, such as STRIDE, DREAD or PASTA, to accommodate different types of systems and users' preferences	This requirement was met – CAIRIS supports a lot of threat modelling methodologies especially STRIDE that was used for the project. CAIRIS was able to ssuccessfully model threats related to STRIDE methodology.
User Authentication – All users of DMS must be authenticated	The requirement was met – The application has an authentication layer that ensures every user must be authenticated and validated before granted access to the system.
Server Availability – The Solution must be available to enable an easy threat modelling process	This requirement was met. The server runs on a virtual machine hosted on the cloud, making the server available 99.9%.
Security – The Solution must ensure that no two users share the same login credentials	This requirement has been met. Both applications successfully achieved this requirement by assigning specific usernames and passwords to individual users.
Diagramming Capabilities – The tool must provide the ability to create and edit diagrams to represent the system and its components, data flows, and other relevant information	This requirement has been met. CAIRIS was successfully able to model diagrams like DFD, tasks, attack trees, etc. thus representing all components of the system.
Security – The Solution must ensure that no two users share the same login credentials	This requirement has been met. Both applications successfully achieved this requirement by assigning specific usernames and passwords to individual users.



Diagramming Capabilities – The tool must provide the ability to create and edit diagrams to represent the system and its components, data flows, and other relevant information	This requirement has been met. CAIRIS was successfully able to model diagrams like DFD, tasks, attack trees, etc. thus representing all components of the system.
Threat Model – CAIRIS must be able to build a threat model of the DMS application	This requirement has been met. CAIRIS was able to successfully model an attack tree using STRIDE methodology to depict all possible threats in the system.
Automated Threat Identification – The tool must have the capability to automatically detect potential threats in the system based on the model created	This requirement has been met and documented in CAIRIS generated report.
Risk Management and Assessment – CAIRIS should allow users to assess the severity of threats and prioritize them based on the likelihood of occurrence and potential impacts	This requirement was met. – CAIRIS possess the ability to allow users to assess the severity of threats and prioritize them based on the likelihood of occurrence and potential impacts.
Integration with other tools – CAIRIS should have the capability to integrate with other tools in the development and security ecosystem, such as bug tracking systems or security testing tools	This requirement was met. – CAIRIS possess the capability to integrate with other tools.
Reporting – CAIRIS must provide the ability to generate reports that summarize the identified threats, their likelihood and impact, and recommend mitigation strategies	This requirement has been met and documented in CAIRIS generated report.
User-friendly Interface – CAIRIS should have a user-friendly interface that allows for easy navigation and interaction with the different features and functionalities	This requirement was met. – CAIRIS have a user-friendly interface and is quite easy to navigate.

Figure 5.1: *Comparison of Achievement with Functional Requirements*

5.5.2 Non-functional Requirements:



User Validation - The Solution should ensure that user login details are validated in less than 15 seconds.	This requirement was met – The Solution successfully ensured users' logins were validated in less than 15 seconds.
Response Time – Model should provide results within the shortest possible time under good network condition.	The requirement was met – The Solution response during implementation was less than 30 seconds, most especially during generation of the model.
Language – To meet the comprehension necessities of its users, the syst UI must be designed in English.	This requirement was met. Solution's user interface is designed in English.
Performance – Solution should be able to handle large and complex models without slowing down or crashing and should provide fast and responsive feedback to users.	This requirement was met. With the solution being a complex application, CAIRIS was able to handle and create the systems' model without any form of lag.
Usability – System should be easy to use, intuitive, and provide clear instructions and guidance to users.	This requirement was met. Solution provided clear instructions and ensured important information was always visible.
Compatibility – System must be compatible with different operating systems, platforms, and other tools in the development and security ecosystem.	This requirement was met. System is compatible with different platforms and can be installed in both Windows, Linux and Mac Operating Systems.
Maintainability – System should be easy to maintain and update, with clear documentation and support available to users.	This requirement was met. System possesses clear documentation on its usage and provide support to its users across several platforms.
Scalability – When more resources are introduced to the system, CAIRIS should be able to handle an increasing workload.	This requirement has been met. – System has proven to possess the ability to handle an increasing workload.
Security – CAIRIS must be secure and protect sensitive information, such as system architecture diagrams and threat data, from unauthorized access or disclosure.	This requirement has been met. The application successfully achieved this requirement by ensuring users have a username and complex passwords before accessing the system.

Figure 5.2*: Comparison of Achievement with Non-functional Requirements*

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The project involved a thorough review of the literature on DevOps, threat modelling, and different techniques, methods, and tools available for threat modelling. The analysis helped in choosing the most suitable techniques and tools, and STRIDE and CAIRIS were selected due to their advantages.

The aim of the project was to improve security using threat modelling, and it was successfully implemented using STRIDE and CAIRIS. The attack generated using the shift-left STRIDE approach resulted in an extensive attack tree meeting all functional and non-functional requirements.

The project's literature review emphasized the importance of the technology acceptance model in threat modelling and how it affects an organization's overall security posture. Key factors in determining the acceptance of the technology were perceived usefulness and ease of use, which are essential for the successful implementation of threat modelling in an organization.

Furthermore, the review highlighted the significance of using the right technologies for development and usability design principles to create threat trees that are easy to understand and navigate.

The project's outcome met the requirements of the project, and further research could be conducted to enhance the functionality of the CAIRIS tool. One potential area of future research could be to expand the tool's capabilities to address emerging security threats and incorporate the latest security best practices.

In conclusion, the project successfully demonstrated the integration of security into the DevOps process using the 'shift left' approach via threat modelling. It also highlighted the importance of choosing the right techniques and tools, considering the technology acceptance model, usability design principles, and the latest security best practices. The project's outcome serves as a valuable reference for organizations seeking to integrate security into their DevOps process effectively.

6.2 FUTURE WORK

In the work of this project, the main aim was to enhance application security by integrating threat modelling into the CI pipeline during software automation. CAIRIS was adopted as tool-support for modelling security threats. Despite the clear achievement of the project’s aim, some limitations exist in this work that can inform further study. The recommendations below are based on these limitations and room for improvement.

Output documentation format: One way to adapt CAIRIS as tool-support for security threat modelling is to improve the output documentation format. This can include creating new templates or improving existing ones to provide more detailed information about identified security risks, threats and vulnerabilities. For example, the output documentation format can include information about the severity of identified threats, the likelihood of exploitation, and suggested mitigation strategies. This can help stakeholders better understand the potential security risks and enable them to make informed decisions about remediation efforts.

New/improved validation checks: Another way to adapt CAIRIS as tool-support for security threat modelling is to introduce new or improved validation checks. Validation checks can be used to ensure that the identified threats and vulnerabilities are accurate, complete, and consistent. For example, validation checks can be used to verify that all possible threats have been identified and that the associated risks have been appropriately assessed. This can help improve the overall quality of the threat modelling process and reduce the likelihood of overlooking critical security threats.

In summary, adapting CAIRIS as tool-support for security threat modelling can involve improving the output documentation format and introducing new/improved validation checks. These improvements can help stakeholders better understand the potential security risks and ensure that the identified threats and vulnerabilities are accurate, complete, and consistent.

PERSONAL REFLECTION

I have gained several valuable lessons and personal reflections from this project, including:

Importance of Collaboration: One of the key takeaways from this project is the importance of collaboration between security professionals, developers, and other stakeholders in the software development process. By involving all stakeholders early in the development process, security can be integrated from the outset, rather than being bolted on as an afterthought.

Need for Continuous Learning: The fast-evolving nature of cyber threats and the constantly changing security landscape means that security professionals need to keep learning and adapting their skills and knowledge. This project has reinforced the importance of staying up to date with the latest trends, techniques, and technologies in the field of cyber security.

Importance of Documentation: Throughout the project, I realized the importance of documenting my work and findings, as this helps with knowledge retention and transfer, especially for future reference and collaboration.

Better Time Management: I also learned that better time management is critical when conducting research projects, especially when working with tools and methodologies that require careful attention to detail. Effective time management allowed me to balance my academic work with other commitments and complete the project on time.

Use of Different Threat Modelling Methodologies: While STRIDE was an effective threat modelling methodology for this project, I recognize that other methodologies like DREAD or PASTA could have also been used to identify and mitigate threats. In future projects, I will explore the use of different methodologies to gain a broader perspective on threat modeling.

Overall, this project has been a valuable learning experience that has helped me develop key skills and knowledge in the field of cyber security, and I believe that the lessons learned will be useful in my future career as both a security and DevOps professional.

REFERENCES

ALTAF, A. et al., 2019. Evaluating the Impact of Cyber Security and Safety with Human Factors in Rail Using Attacker Personas. [online]. Available from: https://cairis.org [Accessed 26 Mar 2023].

BODEAU, D.J., MCCOLLUM, C.D. and FOX, D.B., 2018. *Cyber Threat Modelling: Survey, Assessment, and Representative Framework*. [online]. Available from: https://apps.dtic.mil/sti/citations/AD1108051 [Accessed 28 Feb 2023].

CAIRIS, 2023. *CAIRIS*. [online]. Available from: https://cairis.org/ [Accessed 27 Feb 2023].

CAPEC, 2019a. *CAPEC - About CAPEC*. [online]. Available from: https://capec.mitre.org/about/index.html [Accessed 28 Feb 2023].

CAPEC, 2019b. *CAPEC - ATT&CK Comparison*. [online]. Available from: https://capec.mitre.org/about/attack_comparison.html [Accessed 28 Feb 2023].

CRUZES, D.S. et al., 2021. Challenges and Experiences with Applying Microsoft Threat Modelling in Agile Development Projects. [online]. Available from: https://www.bsimm.com [Accessed 27 Feb 2023].

CURZI, S. et al., 2023. *Integrating threat modelling with DevOps - Security documentation | Microsoft Learn*. [online]. Available from: https://learn.microsoft.com/en-us/security/engineering/threat-modelling-with-dev-ops [Accessed 9 Mar 2023].

DOBAJ, J. et al., 2021. Towards a security-driven automotive development lifecycle. *Journal of Software: Evolution and Process*, p. e2407. Available from: https://onlinelibrary.wiley.com/doi/full/10.1002/smr.2407 [Accessed 28 Feb 2023].

EBERT, C. and JONES, C., 2009. Embedded software: Facts, figures, and future. *Computer*, 42(4), pp. 42–52.

FAILY, S., 2018. Case Study: Defending Critical Infrastructure Against Stuxnet. *Designing Usable and Secure Software with IRIS and CAIRIS*, pp. 155–175. Available from: https://link.springer.com/chapter/10.1007/978-3-319-75493-2_8 [Accessed 26 Mar 2023].

FAILY, S. and IACOB, C., 2017. *Design as Code: Facilitating Collaboration between Usability and Security Engineers*. [online]. Available from: https://eprints.bournemouth.ac.uk/29464/1/faia17.pdf [Accessed 26 Mar 2023].

GÁLVEZ, R. and GÜRSES, S., 2018. The Odyssey: modelling privacy threats in a brave new world.

GRUVER, G., 2016. Starting and scaling DevOps in the enterprise, p. 93.

HOWARD, K., 2019. Security by Design. *Journal of Physical Security*. [online]. Available from: https://www.academia.edu/44134569/Security_by_Design [Accessed 9 Mar 2023].

HUSSAIN, S. et al., 2014. *(PDF) THREAT MODELLING METHODOLOGIES: A SURVEY*. [online]. Available from: https://www.researchgate.net/publication/307902746_THREAT_MODELLING_METHODOLOGIES_A_SURVEY [Accessed 28 Feb 2023].

KHAN, R. et al., 2018. STRIDE-based Threat Modelling for Cyber-Physical Systems. [online]. Available from: https://doi.org/10.1109/ISGTEurope.2017.8260283 [Accessed 17 Mar 2023].

KHAN, S.A., 2017. A STRIDE Model based Threat Modelling using Unified and-Or Fuzzy Operator for Computer Network Security. *International Journal of Computing and Network Technology*, 5(1), pp. 13–20. Available from: https://www.researchgate.net/publication/318611364_A_STRIDE_Model_based_Threat_Modelling_using_Unified_and-Or_Fuzzy_Operator_for_Computer_Network_Securit [Accessed 27 Feb 2023].

LOREN, K. and PRAERIT, G., 1999. *'The Threats to Our Products' - Microsoft Security Blog*. [online]. Available from: https://www.microsoft.com/en-us/security/blog/2009/08/27/the-threats-to-our-products/ [Accessed 28 Feb 2023].

LUND, M.S., SOLHAUG, B. and STØLEN, K., 2011. Model-driven risk analysis: The CORAS approach. *Model-Driven Risk Analysis: The CORAS Approach*, pp. 1–460.

MEAD, N.R., SHULL, F. and VEMURU, K., 2018. A Hybrid Threat Modelling Method. [online]. Available from: http://www.sei.cmu.edu [Accessed 28 Feb 2023].

OWASP, 2022. *OWASP Top Ten | OWASP Foundation*. [online]. Available from: https://owasp.org/www-project-top-ten/ [Accessed 28 Feb 2023].

POLLER, A. et al., 2017. Can security become a routine? A study of Organizational change in an agile software development group. *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pp. 2489–2503. Available from: https://research.aalto.fi/en/publications/can-security-become-a-routine-a-study-of-organizational-change-in [Accessed 28 Feb 2023].

PRODUCT PLAN, 2022. *What is MoSCoW Prioritization? | Overview of the MoSCoW Method*. [online]. Available from: https://www.productplan.com/glossary/moscow-prioritization/ [Accessed 10 Sep 2022].

RAJAPAKSE, R.N. et al., 2022. Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, 141, p. 106700.

SHOSTACK, A., 2014. *Threat Modelling: Designing for Security [Book]*. [online]. Available from: https://www.oreilly.com/library/view/threat-modelling-designing/9781118810057/ [Accessed 28 Feb 2023].

SOUPPAYA, M. and SCARFONE, K., 2016. Guide to Data-Centric System Threat Modelling. Available from: https://csrc.nist.gov/publications/detail/sp/800-154/draft [Accessed 28 Feb 2023].

TØNDEL, I.A. et al., 2019. Understanding challenges to adoption of the Protection Poker software security game. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11387 LNCS, pp. 153–172.

TUMA, K., CALIKLI, G. and SCANDARIATO, R., 2018. Threat analysis of software systems: A systematic literature review. *Journal of Systems and Software*, 144, pp. 275–294.

TÜRPE, S. and POLLER, A., 2017. Managing Security Work in Scrum: Tensions and Challenges.

UCEDAVÉLEZ, T. and MORANA, M.M., 2015. Risk Centric Threat Modelling: Process for Attack Simulation and Threat Analysis. *Risk Centric Threat Modelling: Process for Attack Simulation and Threat Analysis*, pp. 1–664. Available from: https://www.researchgate.net/publication/327549205_Risk_Centric_Threat_Modelling_Process_for_Attack_Simulation_and_Threat_Analysis [Accessed 28 Feb 2023].

VON DER ASSEN, J. et al., 2022. On Collaborative Threat Modelling.

VOLKERDON, 2022. *MoSCoW prioritization technique | Volkerdon*. [online]. Available from: https://www.volkerdon.com/pages/moscow-prioritisation [Accessed 23 Oct 2022].

WHITTLE, D., 2014. *An Introduction to DevOps - DevOps.com*. [online]. Available from: https://devops.com/introductiontodevops/ [Accessed 9 Mar 2023].

YARLAGADDA, R.T., 2021. *(PDF) DevOps and Its Practices*. [online]. Available from: https://www.researchgate.net/publication/350006289_DevOps_and_Its_Practices [Accessed 9 Mar 2023].

APPENDICES

Appendix A: Project Plan

Appendix B: Full Installation Guide

Appendix C: Detailed System Procedure

Appendix D: Questionnaires

Appendix D: Full Questionnaires Insights

APPENDIX A: PROJECT PLAN

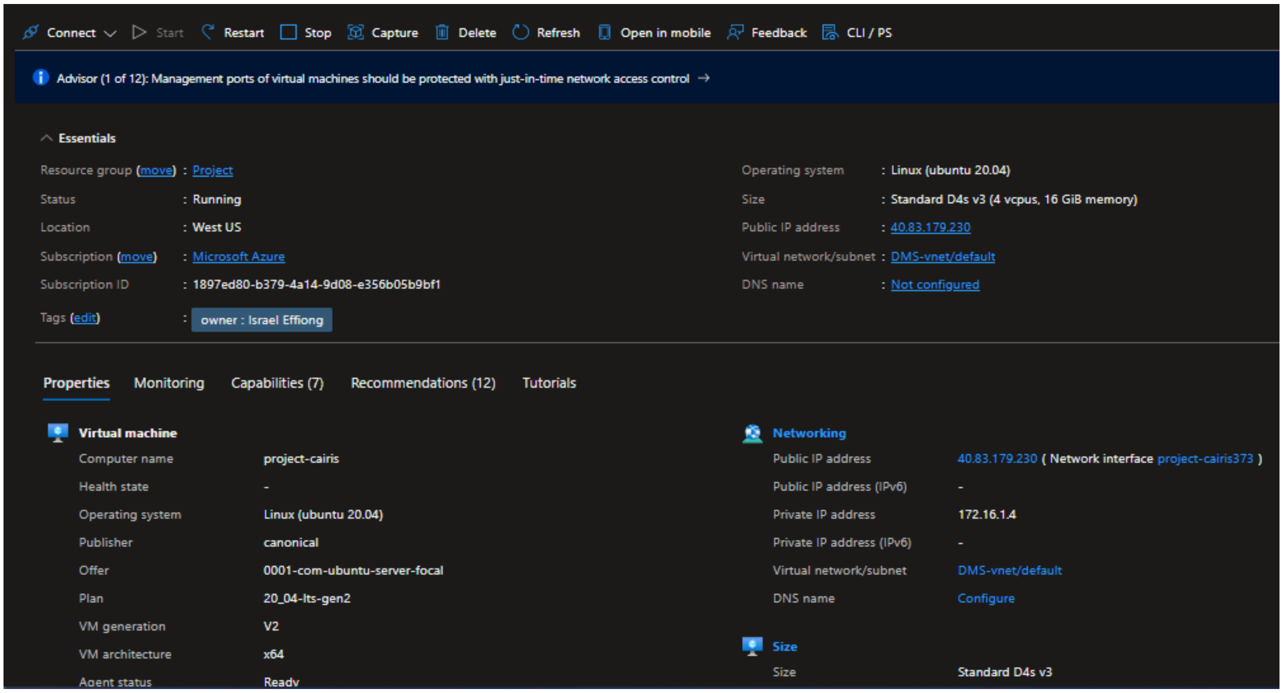
PROJECT PLAN				
Task ID	**Task Name**	**Duration**	**Start Date**	**End Date**
	Project Report	78 days	06-02-2023	25-04-2023
Phase 1	**Project Planning and Preparation**	5 days		
1.1	Define scope and objective of project.	2 days	06-02-2023	07-02-2023
1.2	Create a detailed project plan with timelines and milestones.	1 day	08-02-2023	08-02-2023
1.3	Set up necessary tools and infrastructure for the project	3 days	09-02-2023	11-02-2023
Phase 2	**Case Study Preparation**	10 days		
2.1	Identify a suitable case study.	2 days	12-02-2023	13-02-2023
2.2	Gather information and documentation related to case study.	3 days	14-02-2023	16-02-2023
2.3	Analyze existing system architecture and design.	2 days	17-02-2023	18-02-2023
2.4	Identify key stakeholders and their roles in the project	3 days	19-02-2023	21-02-2023
Phase 3	**Conduct Interview**	5 days		
3.1	Draft Interview questions	1 day	22-02-2023	22-02-2023
3.2	Schedule interview date/conduct interview with available volunteers	2 days	23-02-2023	24-02-2023
3.3	Complete case study interview	2 days	25-02-2023	26-02-2023
Phase 4	**Questionnaire Creation and Distribution**	5 days		
4.1	Create two separate questionnaires.	2 days	27-02-2023	28-03-2023
4.2	Share the questionnaires with the key stakeholders.	2 days	01-03-2023	02-03-2023
4.3	Collect responses from the stakeholders	1 day	03-03-2023	03-03-2023
Phase 5	**Data Analysis**	10 days		
5.1	Analyze the data gathered from the case study, interviews, and questionnaires	5 days	04-03-2023	08-03-2023
5.2	Identify potential security risks and vulnerabilities.	2 days	09-03-2023	10-03-2023
5.3	Prioritize the identified risks based on their impact and likelihood	3 days	11-03-2023	13-03-2023
Phase 6	**Threat Modelling using STRIDE Approach**	25 days		
6.1	Set up CAIRIS Application	2 days	14-03-2023	15-03-2023
6.2	Perform the STRIDE threat modelling for the case study.	18 days	16-03-2023	02-04-2023
6.3	Identify and document security controls to mitigate the identified risks	5 days	03-04-2023	07-04-2023
Phase 7	**Report Writing**	15 days		
7.1	Create a report summarizing the project findings and recommendations.	10 days	08-04-2023	17-04-2023
7.2	Share report with key stakeholders for feedback.	3 days	18-04-2023	20-04-2023
7.3	Incorporate feedback and finalize the report	2 days	21-04-2023	22-04-2023
Phase 8	**Project Presentation and Submission**	3 days		
8.1	Prepare presentation summarizing the project findings and recommendations.	2 days	23-04-2023	24-04-2023
8.2	Review final project documentation.	1 day	25-04-2023	25-04-2023
	Project submission			

APPENDIX B: INSTALLATION GUIDE

The system was developed using the CAIRIS software adopting the STRIDE shift left approach. To perform this analysis, several steps were taken to evaluate the system to result in an attack tree as guides provided at https://cairis.readthedocs.io/en/latest/install.html. The attack model developed from this process only shows the possible ways the system **could** be compromised, in the next chapter the result of the survey will show the possible areas the system **might** be compromised, an evaluation of what is possible versus what is probable: The step as follows is:

1. **Created a Cloud Account (Microsoft Azure) and configured a Linux VM (Ubuntu 20.04, 4vcpus, 16GiB memory) to host the Threat Modelling Tool - CAIRIS**

   First, a cloud Virtual Machine (VM) is created from Microsoft Azure. This platform provided a host machine on the cloud with system specifications as shown in the figure below.



Advisor (1 of 12): Management ports of virtual machines should be protected with just-in-time network access control →

Resource group (move) : Project
Status : Running
Location : West US
Subscription (move) : Microsoft Azure
Subscription ID : 1897ed80-b379-4a14-9d08-e356b05b9bf1
Tags (edit) : owner : Israel Effiong

Operating system : Linux (ubuntu 20.04)
Size : Standard D4s v3 (4 vcpus, 16 GiB memory)
Public IP address : 40.83.179.230
Virtual network/subnet : DMS-vnet/default
DNS name : Not configured

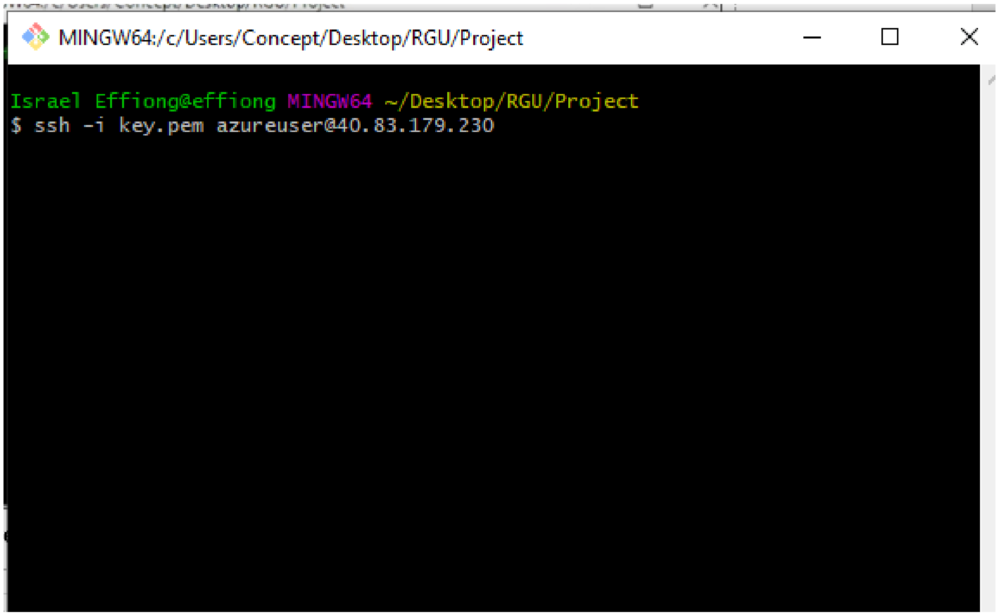
Virtual machine
Computer name project-cairis
Health state -
Operating system Linux (ubuntu 20.04)
Publisher canonical
Offer 0001-com-ubuntu-server-focal
Plan 20_04-lts-gen2
VM generation V2
VM architecture x64
Agent status Ready

Networking
Public IP address 40.83.179.230 (Network interface project-cairis373)
Public IP address (IPv6) -
Private IP address 172.16.1.4
Private IP address (IPv6) -
Virtual network/subnet DMS-vnet/default
DNS name Configure

Size
Size Standard D4s v3

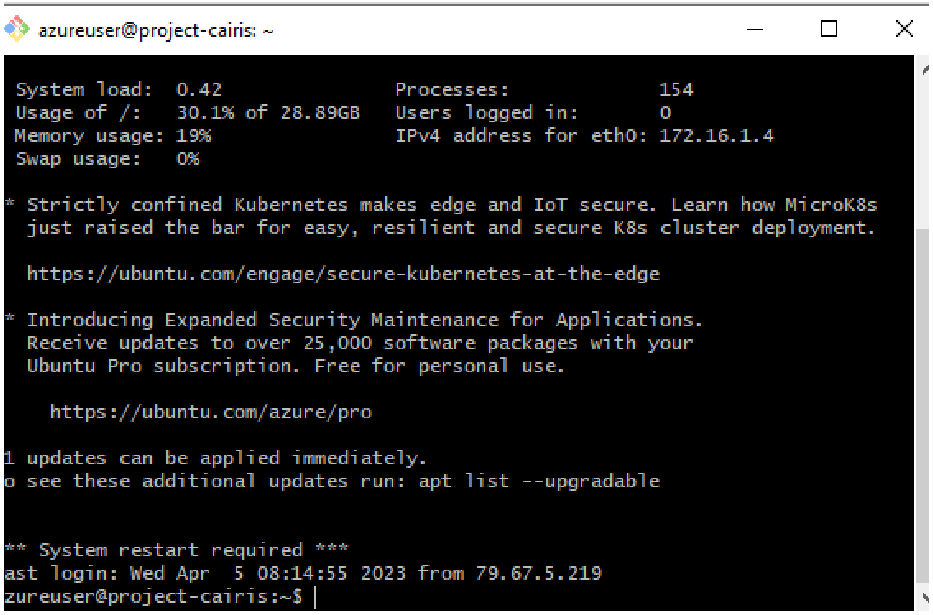
System Specification for CAIRIS platform hosted on Microsoft Azure

To access the Azure cloud account, a private hash key is used. This key is stored as a PEM file named key.pem. The Gitbash command is opened in the same directory of private PEM key and the command as seen in Figure below, the result of the command shown in Figure below. The Figure shows the status of the instantiated Linux Machine running CAIRIS.



```
Israel Effiong@effiong MINGW64 ~/Desktop/RGU/Project
$ ssh -i key.pem azureuser@40.83.179.230
```

GitBash Command Line Command to Access Virtual Machine

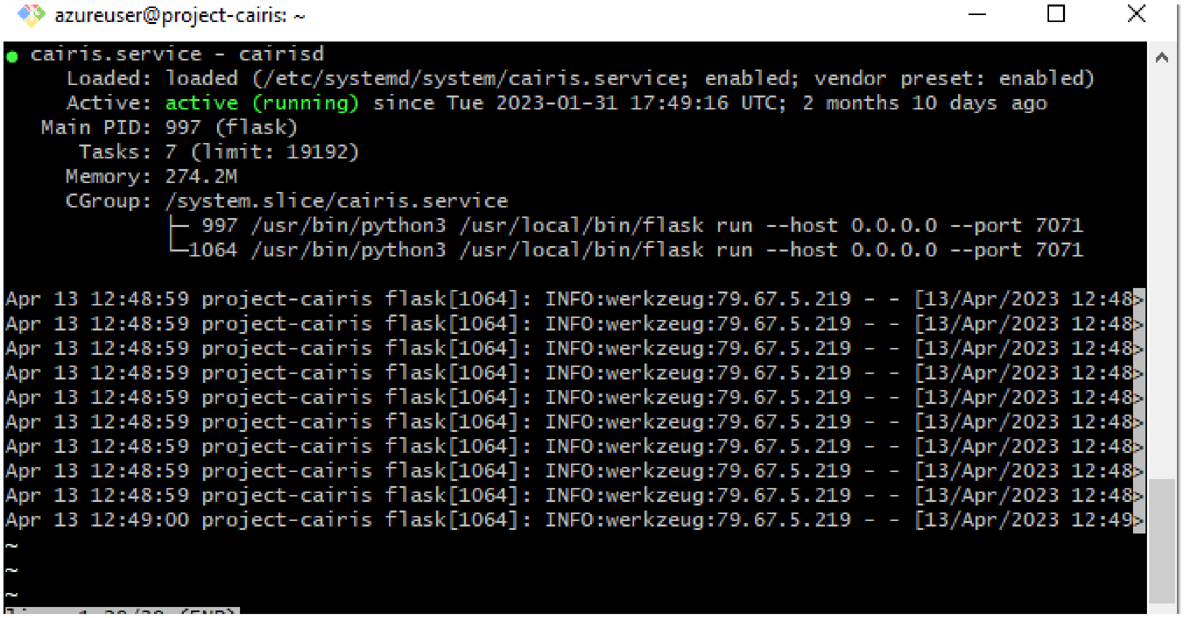


```
 System load:  0.42               Processes:             154
 Usage of /:   30.1% of 28.89GB   Users logged in:       0
 Memory usage: 19%                IPv4 address for eth0: 172.16.1.4
 Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

  https://ubuntu.com/engage/secure-kubernetes-at-the-edge

* Introducing Expanded Security Maintenance for Applications.
  Receive updates to over 25,000 software packages with your
  Ubuntu Pro subscription. Free for personal use.

    https://ubuntu.com/azure/pro

1 updates can be applied immediately.
o see these additional updates run: apt list --upgradable


** System restart required ***
ast login: Wed Apr  5 08:14:55 2023 from 79.67.5.219
zureuser@project-cairis:~$ |
```

GitBash Command Line Command to Access Virtual Machine

2. **Installed and configured the CAIRIS application to be accessible over the internet.**

   To install the CAIRIS application, the Gitbash command line interface is utilized. The command “cd ../../” is entered to change the current directory to a parent directory, the command “ls” is entered to list the folders and files in the new directory. To install CAIRIS, the following command is entered, “*sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get dist-upgrade -y && sudo apt install curl -y && sudo apt install net-tools -y && curl -s https://cairis.org/quickInstall.sh | bash -s my-secret-pw*”. After installation, the following command checks the status of the newly installed CIARIS web platform from command line, “systemctl status cairis” results as shown in figure below.

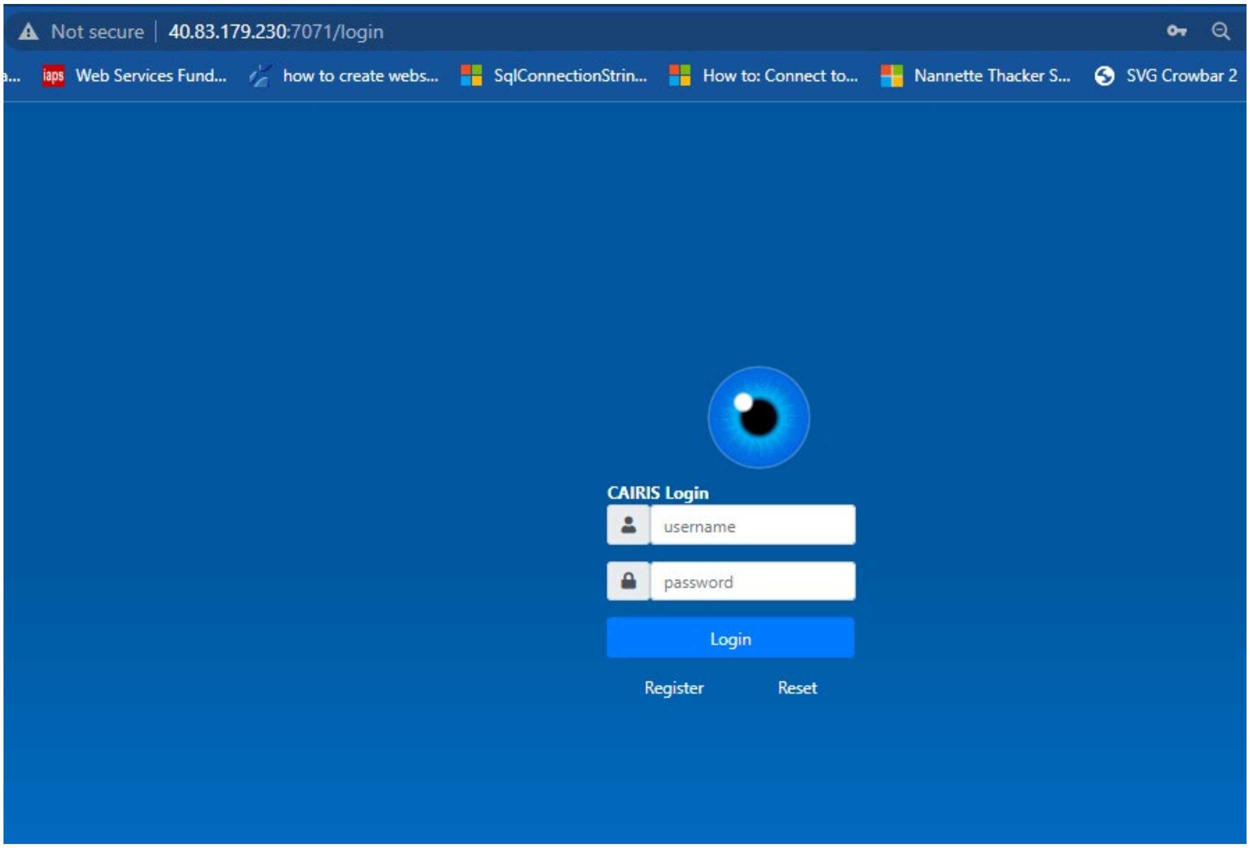


```
● cairis.service - cairisd
     Loaded: loaded (/etc/systemd/system/cairis.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-01-31 17:49:16 UTC; 2 months 10 days ago
   Main PID: 997 (flask)
      Tasks: 7 (limit: 19192)
     Memory: 274.2M
     CGroup: /system.slice/cairis.service
             ├─ 997 /usr/bin/python3 /usr/local/bin/flask run --host 0.0.0.0 --port 7071
             └─1064 /usr/bin/python3 /usr/local/bin/flask run --host 0.0.0.0 --port 7071

Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:48:59 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:48>
Apr 13 12:49:00 project-cairis flask[1064]: INFO:werkzeug:79.67.5.219 - - [13/Apr/2023 12:49>
~
~
~
lines 1-20/20 (END)
```

View CAIRIS status on Azure Host VM

3. Application became accessible via http://40.83.179.230:7071/

To access the CAIRIS platform from the Linux machine hosted on Azure as specified in the figure above, the public IP address needs to be entered into the browser URL, with a colon and port number as seen in Figure above. On load, the URL directs the operator to a login page, where authorized users may gain access, else the need to register an account on the CAIRIS platform.

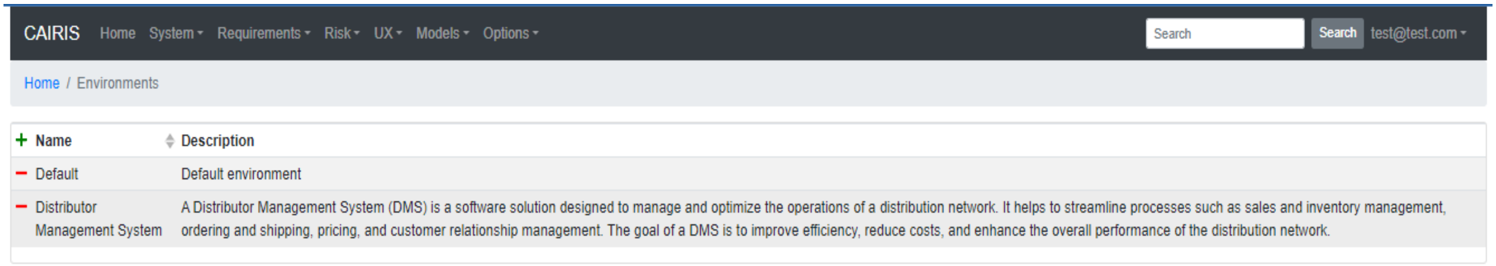


CAIRIS Home Page

APPENDIX C: DETAILED SYSTEM PROCEDURE

- **Create New Environment Based on Case Study - Distributor Management System.**

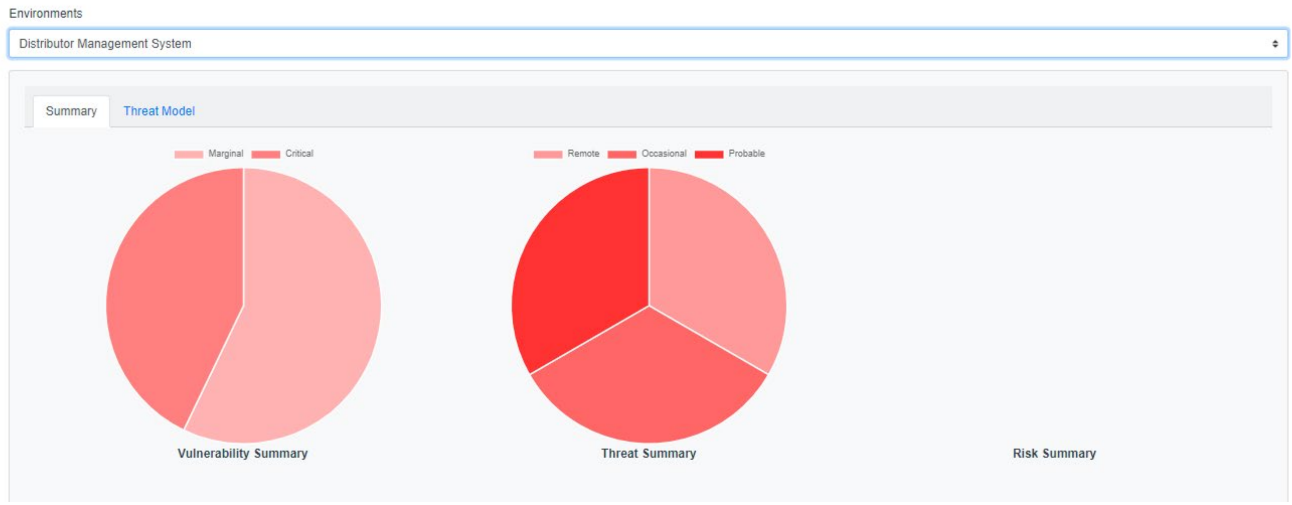
To model a system and its threat on CAIRIS, the environment needs to be modelled first. To do this, select the **UX** dropdown menu and Select **Environment**. This process provides the module to assign a name to the environment and describe it. The CAIRIS platform is preloaded with a default environment.



+ Name	Description
− Default	Default environment
− Distributor Management System	A Distributor Management System (DMS) is a software solution designed to manage and optimize the operations of a distribution network. It helps to streamline processes such as sales and inventory management, ordering and shipping, pricing, and customer relationship management. The goal of a DMS is to improve efficiency, reduce costs, and enhance the overall performance of the distribution network.

Create Environment

This new environment is also configured to the specification appropriate for modelling the assets, roles, personas, processes, dataflows, obstacles, attacks, and attack tree. This environment provides a summary page to view the status of vulnerabilities, threats, and risks as pie charts on the environment as updated in real-time as seen in the figure below.



Environments: Distributor Management System

Environment Summary

- **Create Assets with Their Corresponding Type -**

This section describes the steps taken to create assets based on their corresponding user roles and other system components as shown below.

The system component functionalities are listed below:

i. Azure ADB2C: Utilized for the authentication layer of the application.

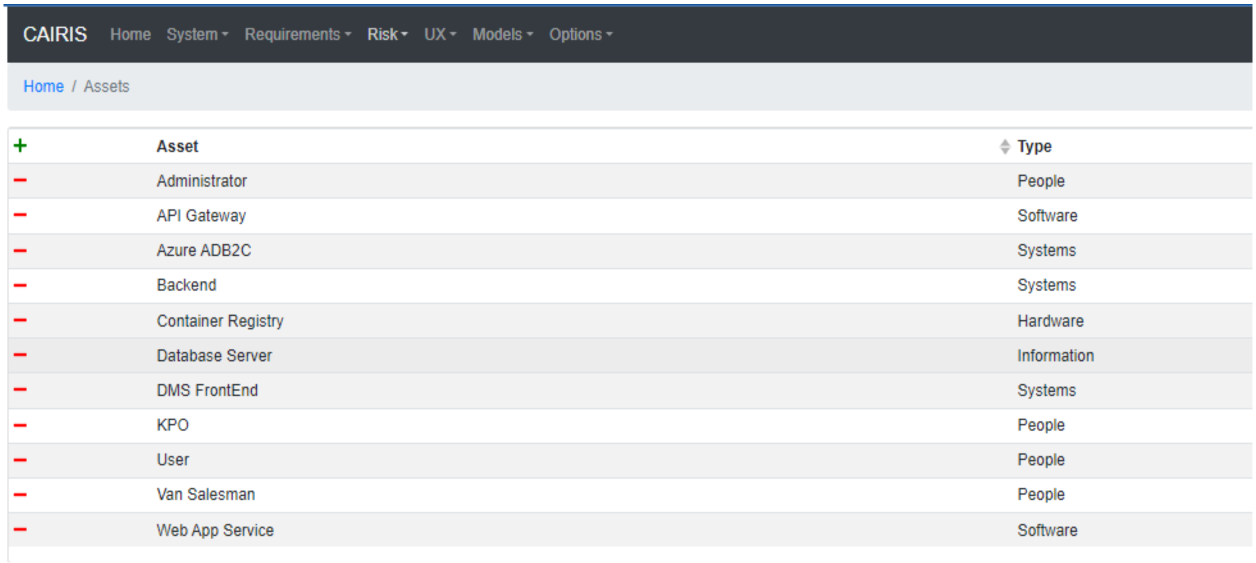
ii. API Gateway: Bridge between frontend and web application services such as microservices etc.

iii. Database Server: Stores data for the application.

iv. DMS Frontend: web and mobile interface of the DMS.

v. Container Registry: Image storage for the microservice application.

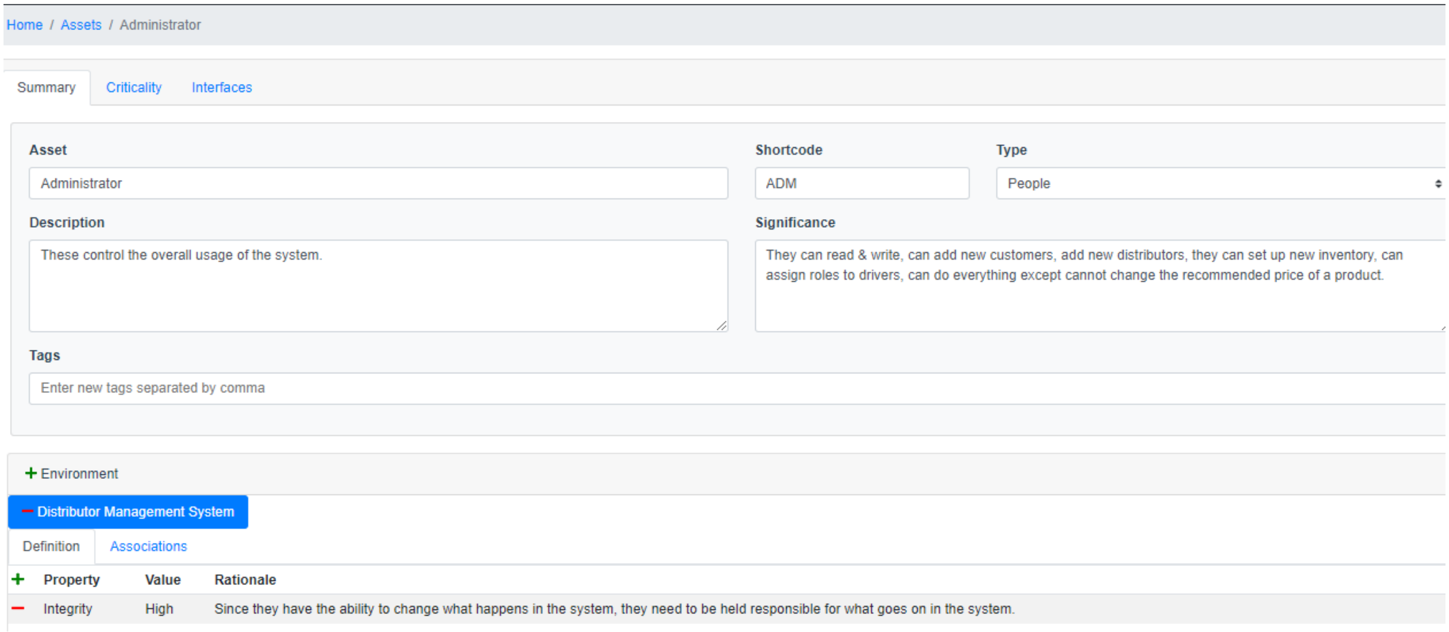
vi. Web App Service: Host the front-end application.



+	Asset	Type
−	Administrator	People
−	API Gateway	Software
−	Azure ADB2C	Systems
−	Backend	Systems
−	Container Registry	Hardware
−	Database Server	Information
−	DMS FrontEnd	Systems
−	KPO	People
−	User	People
−	Van Salesman	People
−	Web App Service	Software

Create Asset Module

The asset management page which enables the creation and update of assets details can be seen in the figure below. The figure below shows that Administrator details, which show the role significance, short-code, description, tags, and their property based on the assigned integrity level. The figure below shows the administrator details as an asset, which describes the administrator as a people in the Type attribute. The type of all user roles in people which include the Van Salesman, KPO and User. However, none-person type assets such as API Gateway are defined as Software type, Azure ADB2C is defined as type Systems, Backend defined as type Systems, DMS Frontend defined as type Systems, Database Server defined as type Information and Container Registry is defined as type hardware.



Home / Assets / Administrator

Summary | Criticality | Interfaces

Asset	**Shortcode**	**Type**
Administrator	ADM	People
Description	**Significance**	
These control the overall usage of the system.	They can read & write, can add new customers, add new distributors, they can set up new inventory, can assign roles to drivers, can do everything except cannot change the recommended price of a product.	

Tags

Enter new tags separated by comma

+ Environment

− Distributor Management System

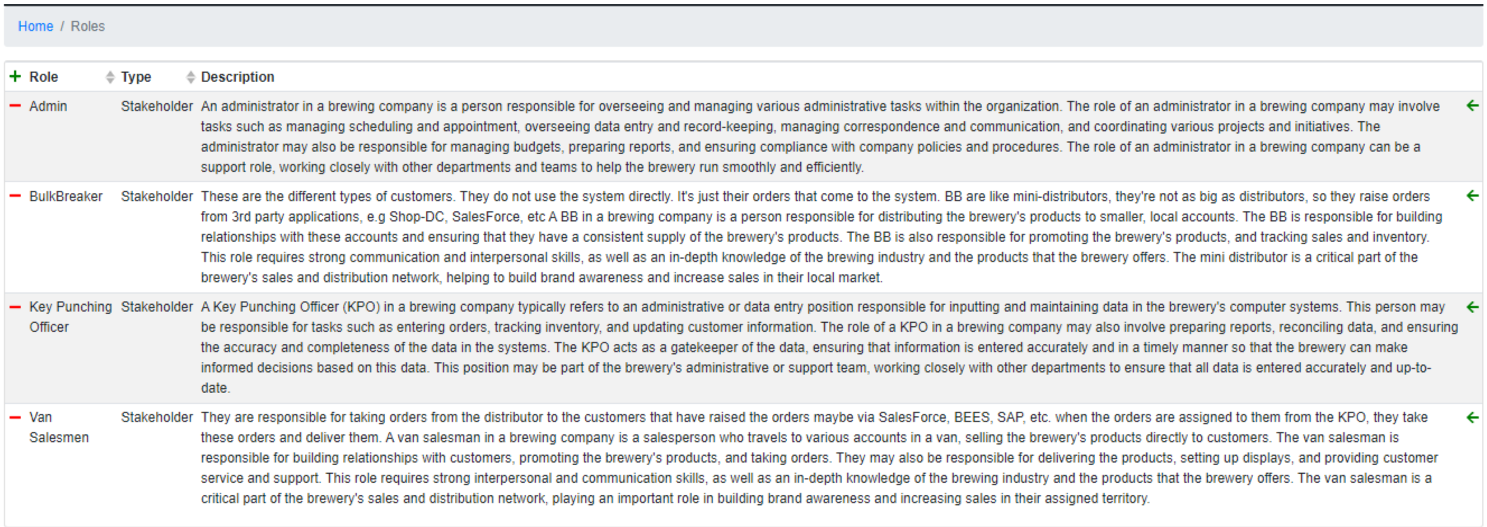
Definition | Associations

+	Property	Value	Rationale
−	Integrity	High	Since they have the ability to change what happens in the system, they need to be held responsible for what goes on in the system.

Asset - Administrator Details Page

- **Create roles and dependency (Admin, KPO, VSM and BB):**

The next step is to create the user roles and their dependencies, and to achieve this select the **Risk** menu dropdown and select **Roles**. This opens the page where one can define each row and provide a description detailing their functionalities within boundaries. To create a role, one can click the "**+**" at the top left corner of the table as seen in the figure below. This provides a form to file in the title and details of the user role.



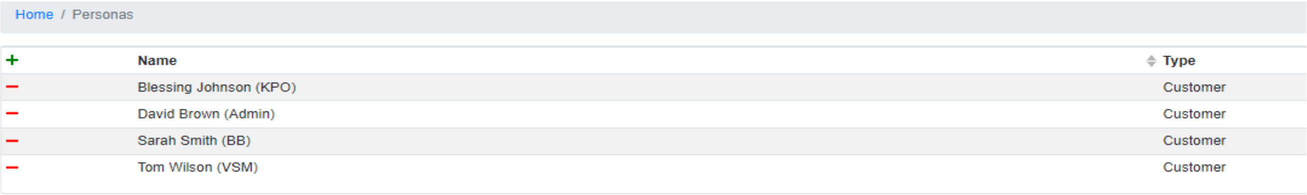
Home / Roles

+ Role	Type	Description	
− Admin	Stakeholder	An administrator in a brewing company is a person responsible for overseeing and managing various administrative tasks within the organization. The role of an administrator in a brewing company may involve tasks such as managing scheduling and appointment, overseeing data entry and record-keeping, managing correspondence and communication, and coordinating various projects and initiatives. The administrator may also be responsible for managing budgets, preparing reports, and ensuring compliance with company policies and procedures. The role of an administrator in a brewing company can be a support role, working closely with other departments and teams to help the brewery run smoothly and efficiently.	←
− BulkBreaker	Stakeholder	These are the different types of customers. They do not use the system directly. It's just their orders that come to the system. BB are like mini-distributors, they're not as big as distributors, so they raise orders from 3rd party applications, e.g Shop-DC, SalesForce, etc A BB in a brewing company is a person responsible for distributing the brewery's products to smaller, local accounts. The BB is responsible for building relationships with these accounts and ensuring that they have a consistent supply of the brewery's products. The BB is also responsible for promoting the brewery's products, and tracking sales and inventory. This role requires strong communication and interpersonal skills, as well as an in-depth knowledge of the brewing industry and the products that the brewery offers. The mini distributor is a critical part of the brewery's sales and distribution network, helping to build brand awareness and increase sales in their local market.	←
− Key Punching Officer	Stakeholder	A Key Punching Officer (KPO) in a brewing company typically refers to an administrative or data entry position responsible for inputting and maintaining data in the brewery's computer systems. This person may be responsible for tasks such as entering orders, tracking inventory, and updating customer information. The role of a KPO in a brewing company may also involve preparing reports, reconciling data, and ensuring the accuracy and completeness of the data in the systems. The KPO acts as a gatekeeper of the data, ensuring that information is entered accurately and in a timely manner so that the brewery can make informed decisions based on this data. This position may be part of the brewery's administrative or support team, working closely with other departments to ensure that all data is entered accurately and up-to-date.	←
− Van Salesmen	Stakeholder	They are responsible for taking orders from the distributor to the customers that have raised the orders maybe via SalesForce, BEES, SAP, etc. when the orders are assigned to them from the KPO, they take these orders and deliver them. A van salesman in a brewing company is a salesperson who travels to various accounts in a van, selling the brewery's products directly to customers. The van salesman is responsible for building relationships with customers, promoting the brewery's products, and taking orders. They may also be responsible for delivering the products, setting up displays, and providing customer service and support. This role requires strong interpersonal and communication skills, as well as an in-depth knowledge of the brewing industry and the products that the brewery offers. The van salesman is a critical part of the brewery's sales and distribution network, playing an important role in building brand awareness and increasing sales in their assigned territory.	←

Create User Roles

- **Create Personas and Attackers**

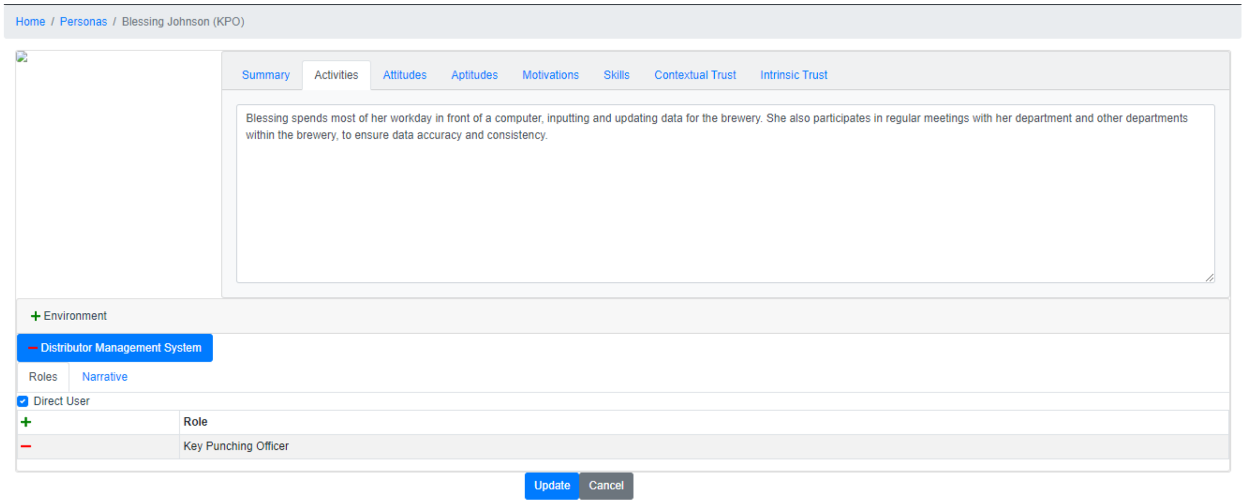
The next step is to create personas, which requires one to select the **UX** menu dropdown, and select **Personas**. These personas can be pseudo individuals who are assigned a role as defined in the DMS environment. The personas created can be seen in the figure below. To create a Persona, the "**+**" at the top left corner of the table can be clicked, as seen in the figure below.



+	Name	Type
−	Blessing Johnson (KPO)	Customer
−	David Brown (Admin)	Customer
−	Sarah Smith (BB)	Customer
−	Tom Wilson (VSM)	Customer

Create Personas

The personas have several attributes which define their uniqueness on the platform. These attributes can be seen as tabs in the figure below. The activities indicate the personas daily activities on the platform, which is a breakdown of their assigned role responsibilities into streamlined actions. The attitude describes the personas behaviour. The aptitude describes their work-ethic, skills outline the personas relevant skills applicable to organization in relation to their role on the DMS platform. The contextual trust indicates the core responsibility of the personas which the DMS requires of that persona to meet its functional requirement and maintain system integrity e.g., Blessing Johnson has a Contextual trust of being trusted to maintain the accuracy and confidentiality of brewery's data.



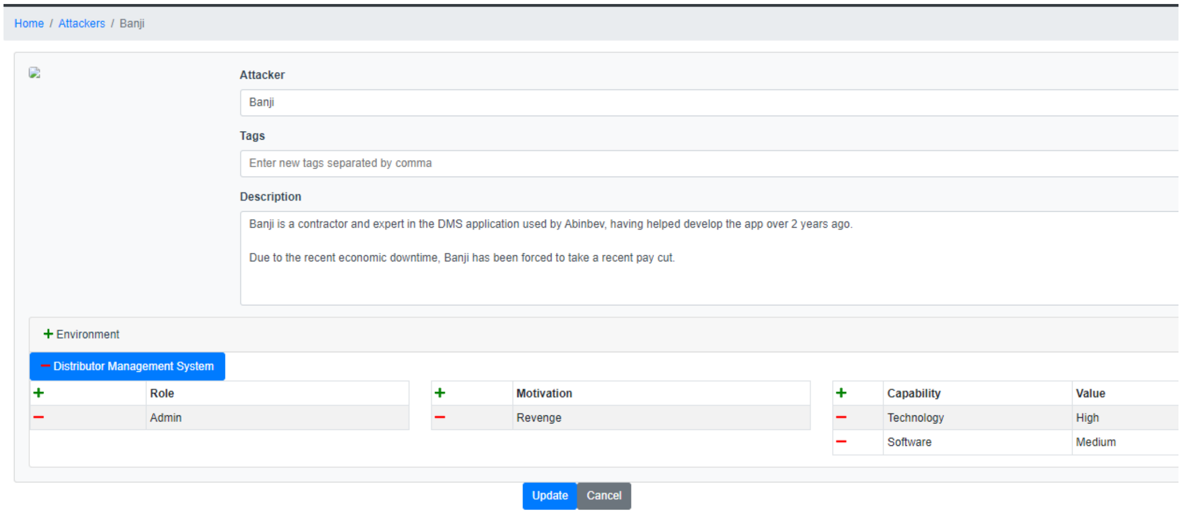
Blessing spends most of her workday in front of a computer, inputting and updating data for the brewery. She also participates in regular meetings with her department and other departments within the brewery, to ensure data accuracy and consistency.

☑ Direct User

+	Role
−	Key Punching Officer

Sample Persona Management Page - Blessing Attacker

This section also creates an attacker with the name **Banji**. This attacker description is provided in the attacker table on attackers' page. To navigate to attackers' page, select **Risk** menu dropdown and select attackers. More details about the attacker are managed from the Attacker Management Page as shown in the figure below which manages the attacker information. This information includes Tags (identifiers), Environment the attacker is exploits, motivation behind the attacker's actions, and capabilities which indicates the attacker's proficiency.



Attacker

Banji

Tags

Enter new tags separated by comma

Description

Banji is a contractor and expert in the DMS application used by Abinbev, having helped develop the app over 2 years ago.

Due to the recent economic downtime, Banji has been forced to take a recent pay cut.

+	Role
−	Admin

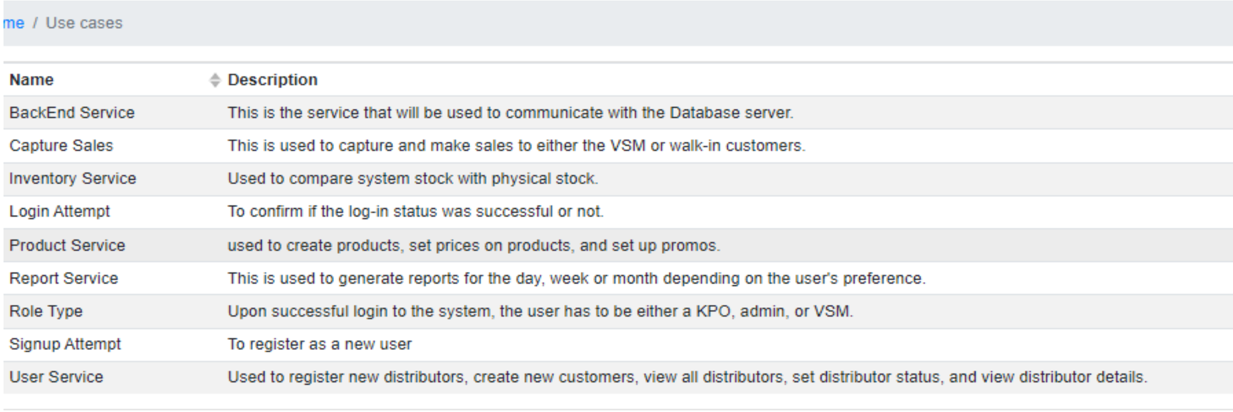
+	Motivation
−	Revenge

+	Capability	Value
−	Technology	High
−	Software	Medium

Attacker Management Page

- **Create Processes in the form of Use Cases**

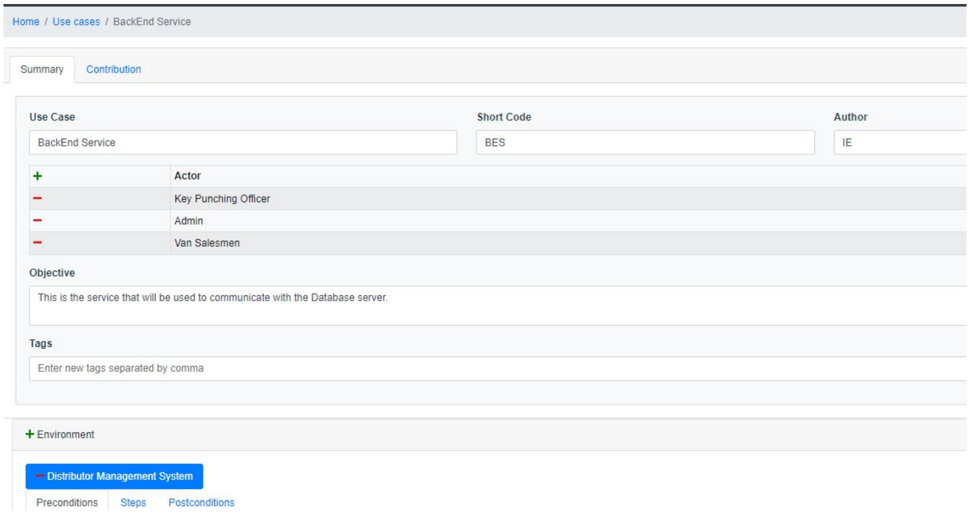
This section creates processes which manages each activity of users based on their roles on the platform. These processes are best illustrated in the Data Flow Diagram (DFD) as services (rounded corner rectangles) as seen in figure 3.8. A listing of these services (processes) or use cases are seen in the figure below. Each process uniquely satisfies their functional requirement of the DMS based on their intrinsic functionality of the collaborative functionality of multiple processes.



Name	Description
BackEnd Service	This is the service that will be used to communicate with the Database server.
Capture Sales	This is used to capture and make sales to either the VSM or walk-in customers.
Inventory Service	Used to compare system stock with physical stock.
Login Attempt	To confirm if the log-in status was successful or not.
Product Service	used to create products, set prices on products, and set up promos.
Report Service	This is used to generate reports for the day, week or month depending on the user's preference.
Role Type	Upon successful login to the system, the user has to be either a KPO, admin, or VSM.
Signup Attempt	To register as a new user
User Service	Used to register new distributors, create new customers, view all distributors, set distributor status, and view distributor details.

Processes (Use Cases) for each DMS Service

The figure above shows the listings of processes on the DMS platform. Each process can be managed to update its properties which include its Objective, Tags, Actors, Short code, Environment, Preconditions, Steps and Post conditions as shown in the figure below.


Home / Use cases / BackEnd Service
Summary
Contribution
+ Environment
− Distributor Management System
Preconditions
Steps
Postconditions

Use Case: BackEnd Service

Short Code: BES

Author: IE

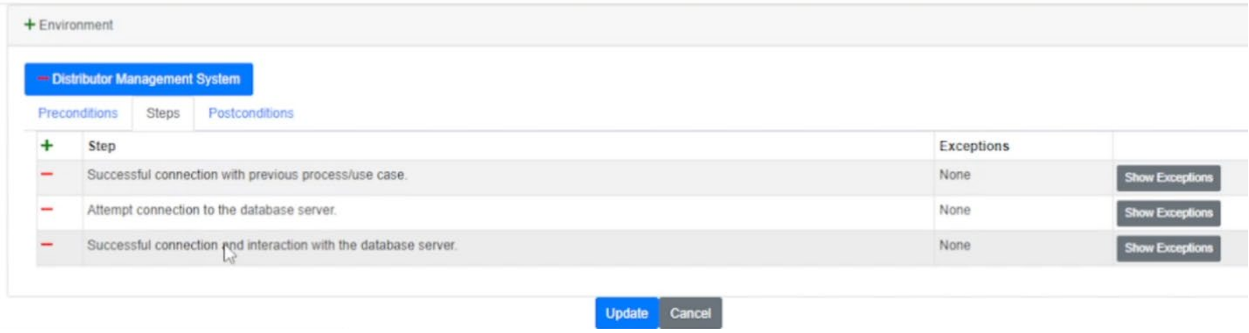
+	Actor
−	Key Punching Officer
−	Admin
−	Van Salesmen

Objective: This is the service that will be used to communicate with the Database server.

Tags: Enter new tags separated by comma

Backend Service Details & Management Page

The **actors** are the user roles that engage the process, **objective** describes the core function of the process, **preconditions** explain the system operations requirements that be met to engage the service i.e., based on the DFD illustration in figure 3.8 the signup attempt and login attempt services, role type service and capture sales services need to be successfully accessed before the user access the Inventory service. The **steps** indicate the sequence of activities that must be complete before access is granted to a particular service e.g., to access the Backend Service the steps to undertake can be shown in the figure below. The post-condition is the service functional requirement that must be met to satisfy the system and data integrity during and after operations.

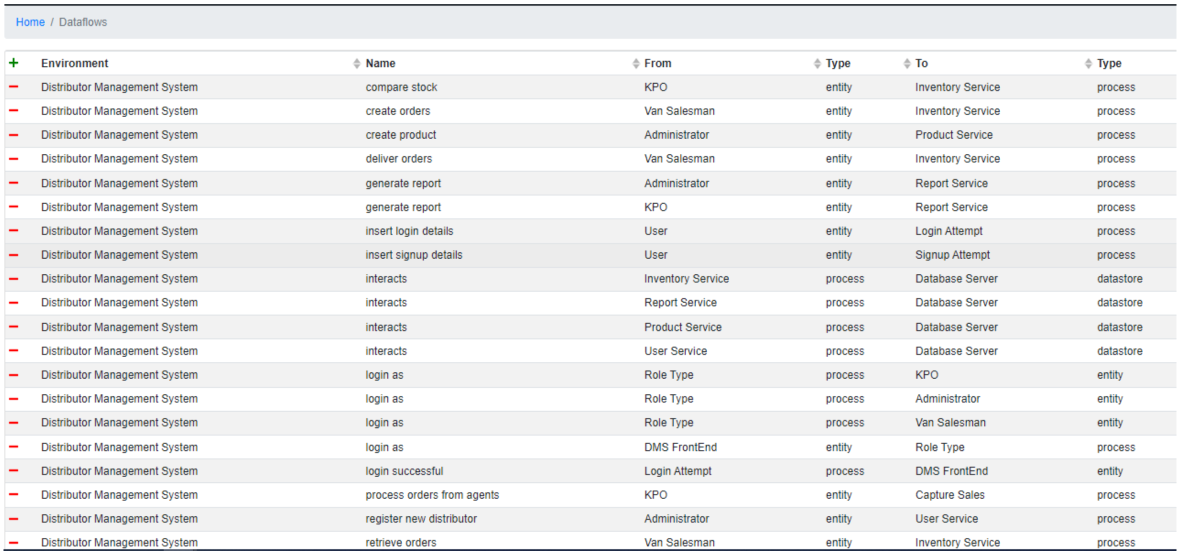


+	Step	Exceptions	
−	Successful connection with previous process/use case.	None	Show Exceptions
−	Attempt connection to the database server.	None	Show Exceptions
−	Successful connection and interaction with the database server.	None	Show Exceptions

Backend Service Steps

- **Create Data flows to link the Entities (Assets) to the Processes (Use Case) and Datastore (Database server)**

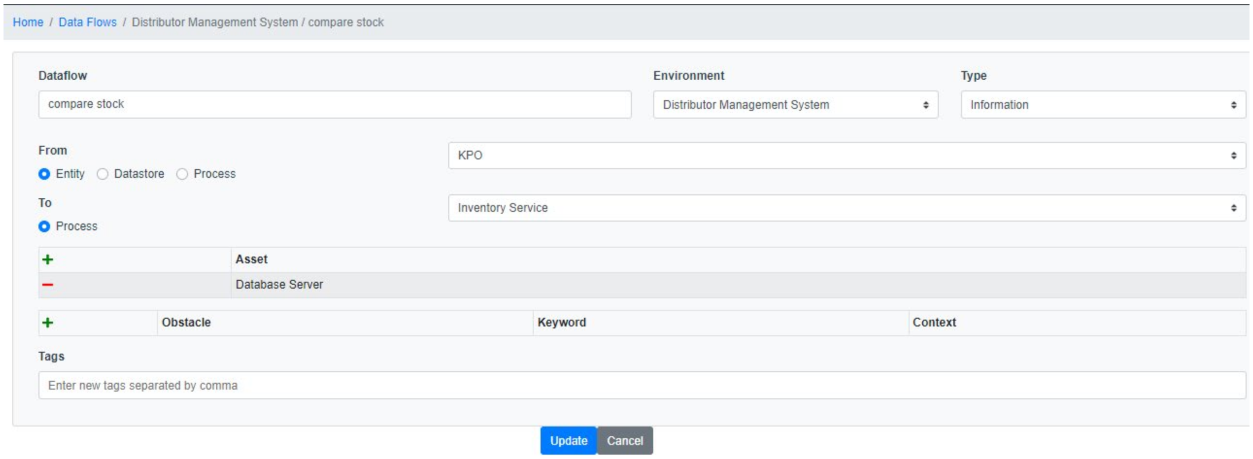
This section discusses the creation and management of entities on the DMS. These entities can be seen on figure 3.8 as rectangles, which are the assets as shown in the figure above. The connecting arrows are managed on the data flow page as shown in the figure below, is accessed from the **UX** menu dropdown, select **Data Flows**. This page shows the list of flows connecting each entity and process on the dataflow model in figure 3.8. The figure below does not show the entire dataflow to the Dataflow model in figure 3.8 which contains a list equal to the number of arrowed lines in the DFD. To create a data flow, click the green plus sign at the top left corner of the table. To manage a dataflow, select the dataflow on the list which proceeds to the dataflow management page as show in the figure below. The figure shows the details of each dataflow, the name, the entity or process it connects from, entity or process it connects to, type of the dataflow.



+	Environment	Name	From	Type	To	Type
−	Distributor Management System	compare stock	KPO	entity	Inventory Service	process
−	Distributor Management System	create orders	Van Salesman	entity	Inventory Service	process
−	Distributor Management System	create product	Administrator	entity	Product Service	process
−	Distributor Management System	deliver orders	Van Salesman	entity	Inventory Service	process
−	Distributor Management System	generate report	Administrator	entity	Report Service	process
−	Distributor Management System	generate report	KPO	entity	Report Service	process
−	Distributor Management System	insert login details	User	entity	Login Attempt	process
−	Distributor Management System	insert signup details	User	entity	Signup Attempt	process
−	Distributor Management System	interacts	Inventory Service	process	Database Server	datastore
−	Distributor Management System	interacts	Report Service	process	Database Server	datastore
−	Distributor Management System	interacts	Product Service	process	Database Server	datastore
−	Distributor Management System	interacts	User Service	process	Database Server	datastore
−	Distributor Management System	login as	Role Type	process	KPO	entity
−	Distributor Management System	login as	Role Type	process	Administrator	entity
−	Distributor Management System	login as	Role Type	process	Van Salesman	entity
−	Distributor Management System	login as	DMS FrontEnd	entity	Role Type	process
−	Distributor Management System	login successful	Login Attempt	process	DMS FrontEnd	entity
−	Distributor Management System	process orders from agents	KPO	entity	Capture Sales	process
−	Distributor Management System	register new distributor	Administrator	entity	User Service	process
−	Distributor Management System	retrieve orders	Van Salesman	entity	Inventory Service	process

Data Flows Page

The figure below shows the details of the compare stock dataflow. It shows the different values that can be assigned each property of the dataflow. The page provides the input to select the environment, type, from and to. The from enables the selection of these types of objects on the DFD which are the entities, datastores and processes. Whichever option is selected in this section provides the list of that object in the select menu which holds the value **KPO** as seen in the figure below. A rule of thumb in CAIRIS for dataflow modelling is that **Entities** and **Datastore** (From) can only connect to **Processes**. However, **Processes** can connect to **Entities**, **Processes** and other **Datastores**.



Home / Data Flows / Distributor Management System / compare stock

Dataflow: compare stock

Environment: Distributor Management System

Type: Information

From
● Entity ○ Datastore ○ Process

KPO

To
● Process

Inventory Service

+	Asset
−	Database Server

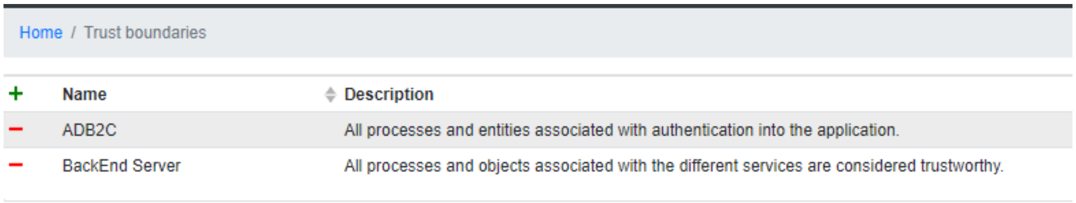
+	Obstacle	Keyword	Context

Tags: Enter new tags separated by comma

Data Flow Details Page

- **Create Trust Boundaries that categorizes all processes and entities associated together.**

This is the next step in the development of the DFD which create and categorize processes and datastores using trust boundaries. To select the Trust Boundaries, select the **UX** dropdown menu, and select trust boundaries. The figure below shows the trust boundaries defined for the DMS environment. The figure below shows two trust boundaries, their description below.

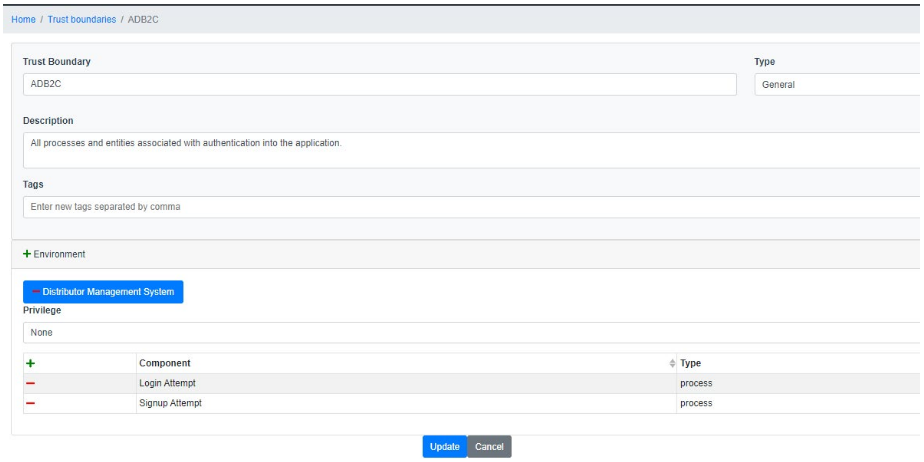


Home / Trust boundaries

+	Name	Description
−	ADB2C	All processes and entities associated with authentication into the application.
−	BackEnd Server	All processes and objects associated with the different services are considered trustworthy.

Trust Boundaries Page

Each trust boundary can be managed by updating its property as seen in the figure below. This shows that each trust boundary has tags, types, and privileges. The figure below shows the properties of the ADB2C trust boundary. This trust boundary has two privileges, which groups two processes, login attempt and signup attempt.



Home / Trust boundaries / ADB2C

Trust Boundary

ADB2C

Type

General

Description

All processes and entities associated with authentication into the application.

Tags

Enter new tags separated by comma

+ Environment

− Distributor Management System

Privilege

None

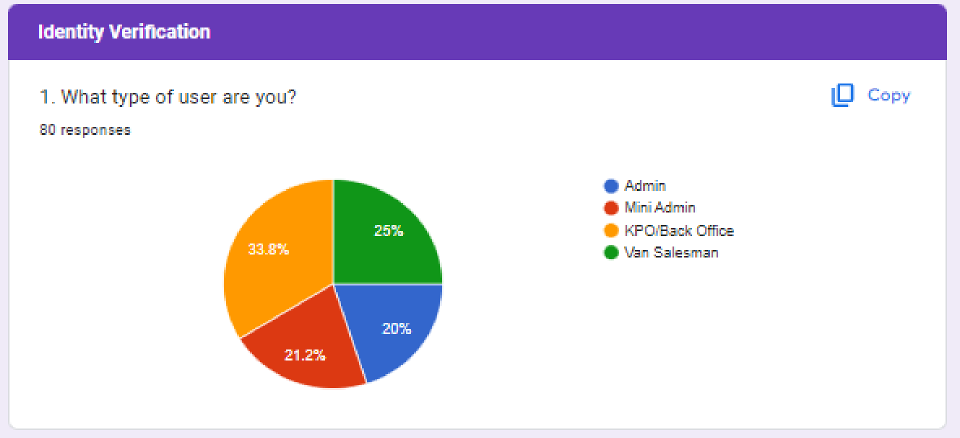
+	Component	Type
−	Login Attempt	process
−	Signup Attempt	process

Trust Boundary Details Page

APPENDIX D: QUESTIONNAIRES

The first questionnaire was divided into 3 sections:

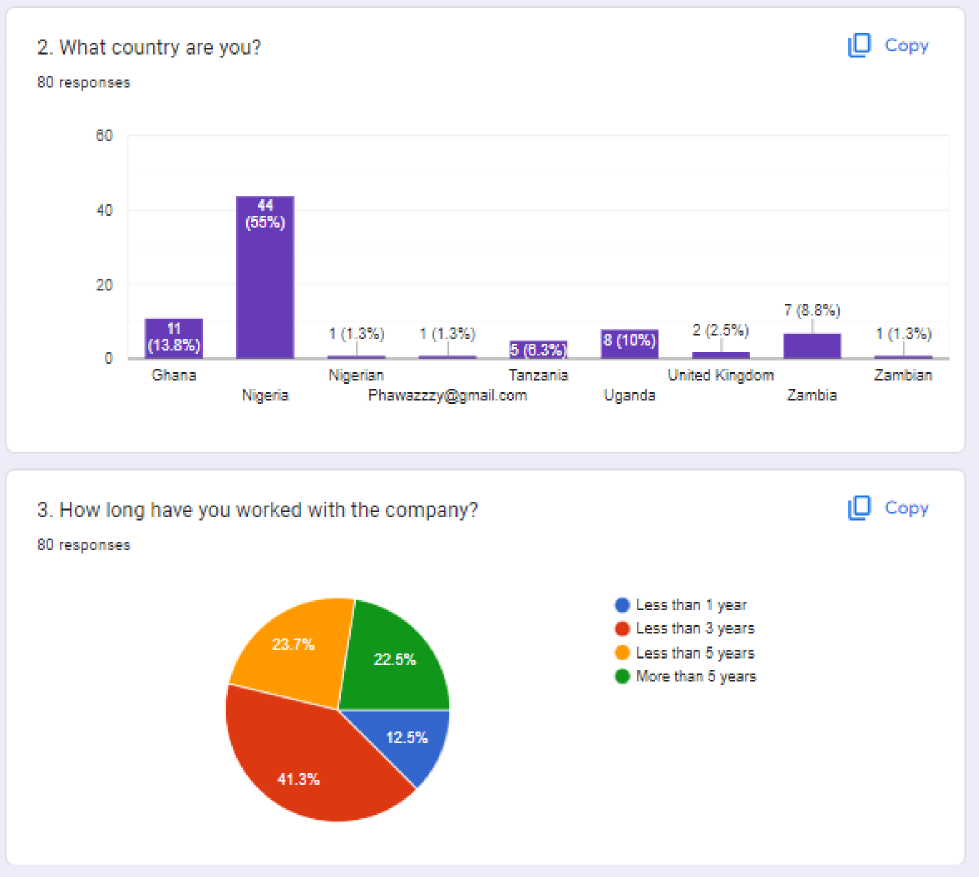
- **Identity Verification**: This section was required to identify staff role and experience at the organization and with the DMS application, both web and mobile.



Identity Verification

1. What type of user are you?

80 responses

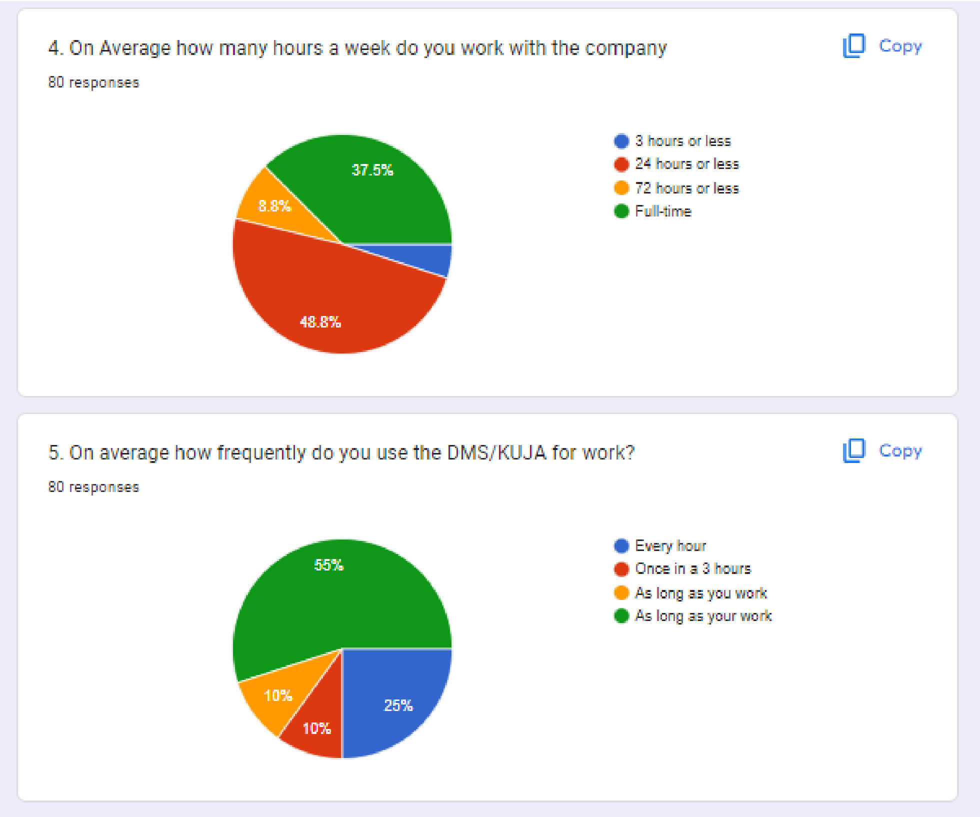


2. What country are you?

80 responses

3. How long have you worked with the company?

80 responses



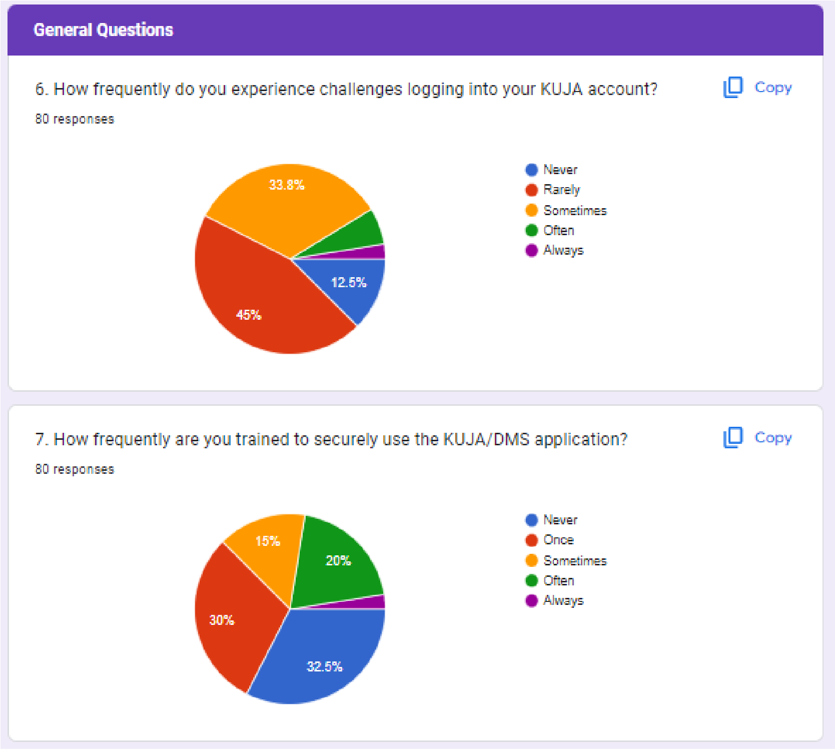
4. On Average how many hours a week do you work with the company

80 responses

5. On average how frequently do you use the DMS/KUJA for work?

80 responses

- **General Questions**: This section ask about challenges experienced from using the KUJA mobile and web application.



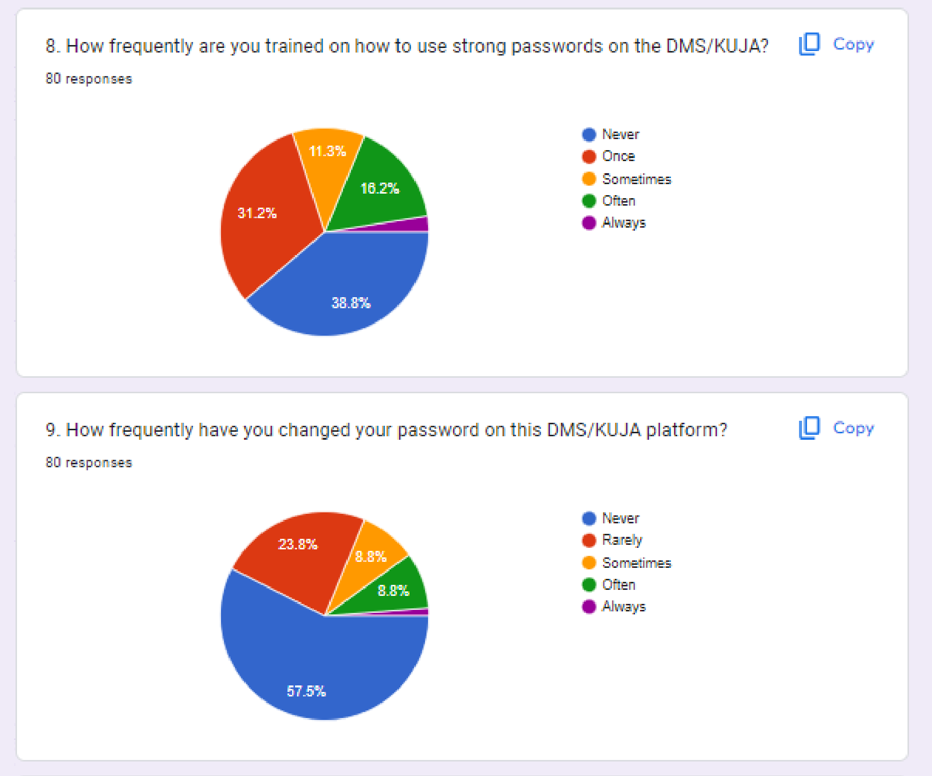
General Questions

6. How frequently do you experience challenges logging into your KUJA account?

80 responses

7. How frequently are you trained to securely use the KUJA/DMS application?

80 responses

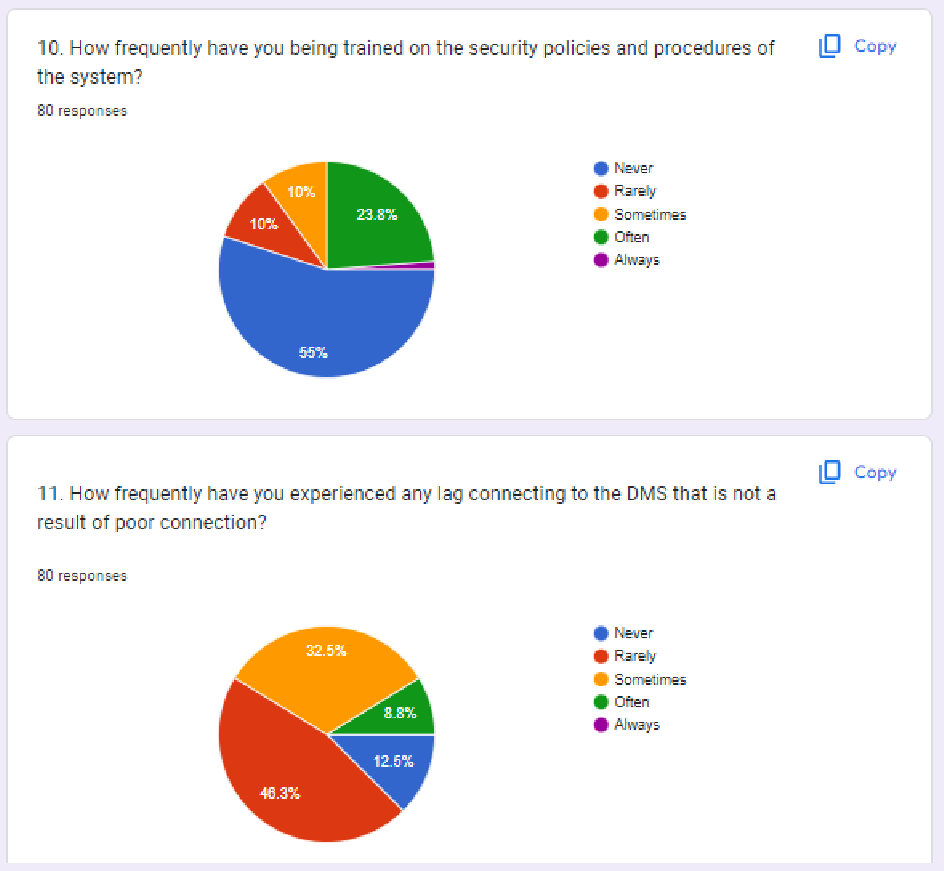


8. How frequently are you trained on how to use strong passwords on the DMS/KUJA?

80 responses

9. How frequently have you changed your password on this DMS/KUJA platform?

80 responses

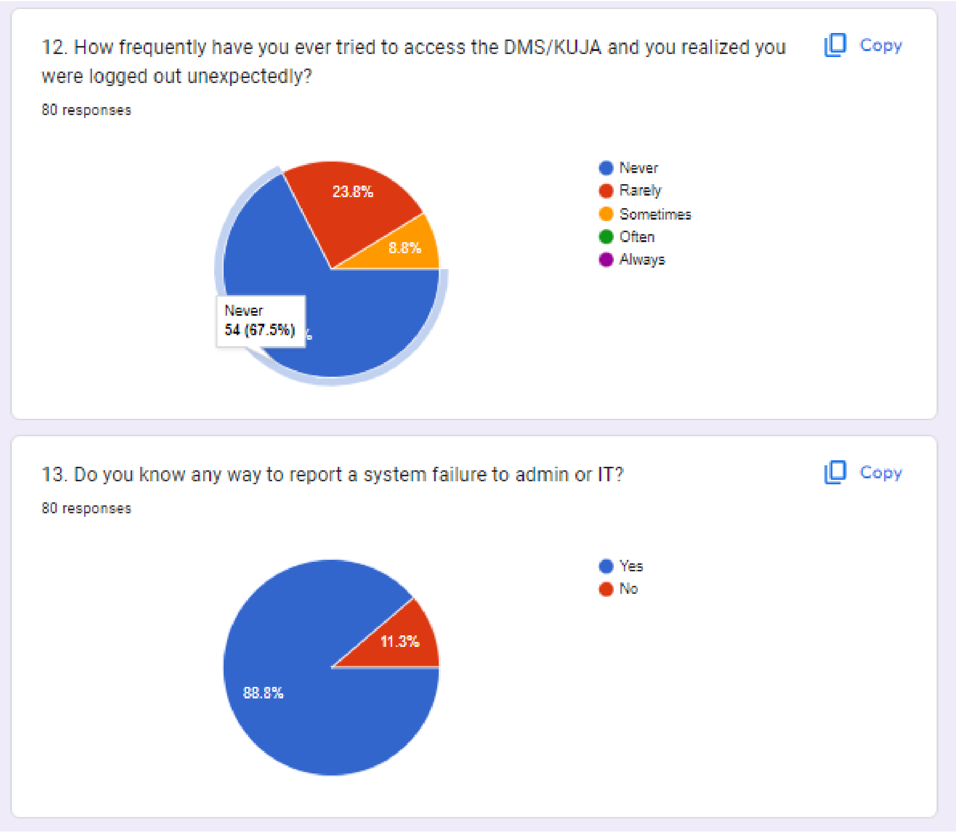


10. How frequently have you being trained on the security policies and procedures of the system?

80 responses

11. How frequently have you experienced any lag connecting to the DMS that is not a result of poor connection?

80 responses

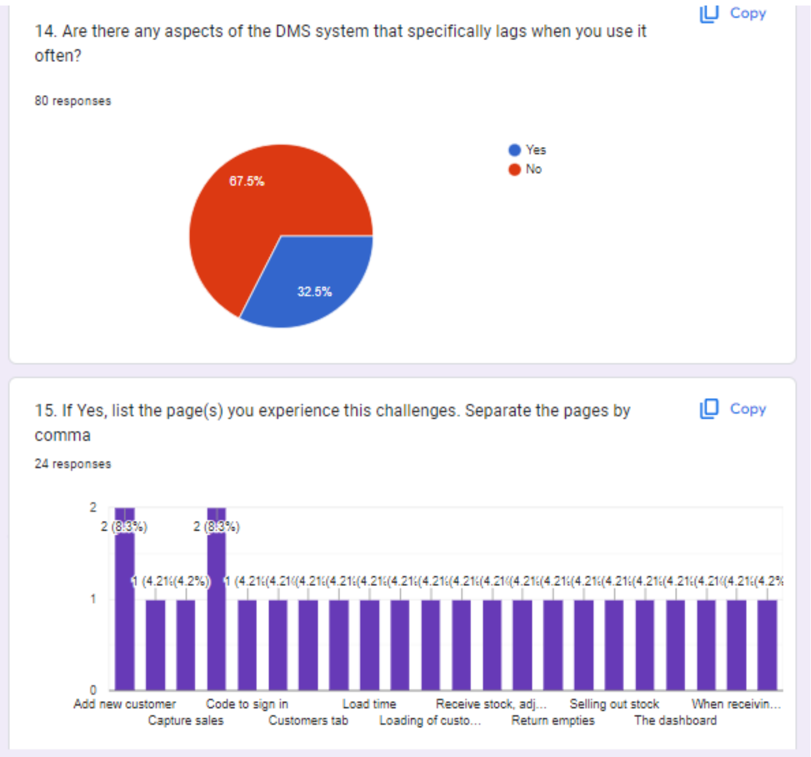


12. How frequently have you ever tried to access the DMS/KUJA and you realized you were logged out unexpectedly?

80 responses

13. Do you know any way to report a system failure to admin or IT?

80 responses



14. Are there any aspects of the DMS system that specifically lags when you use it often?

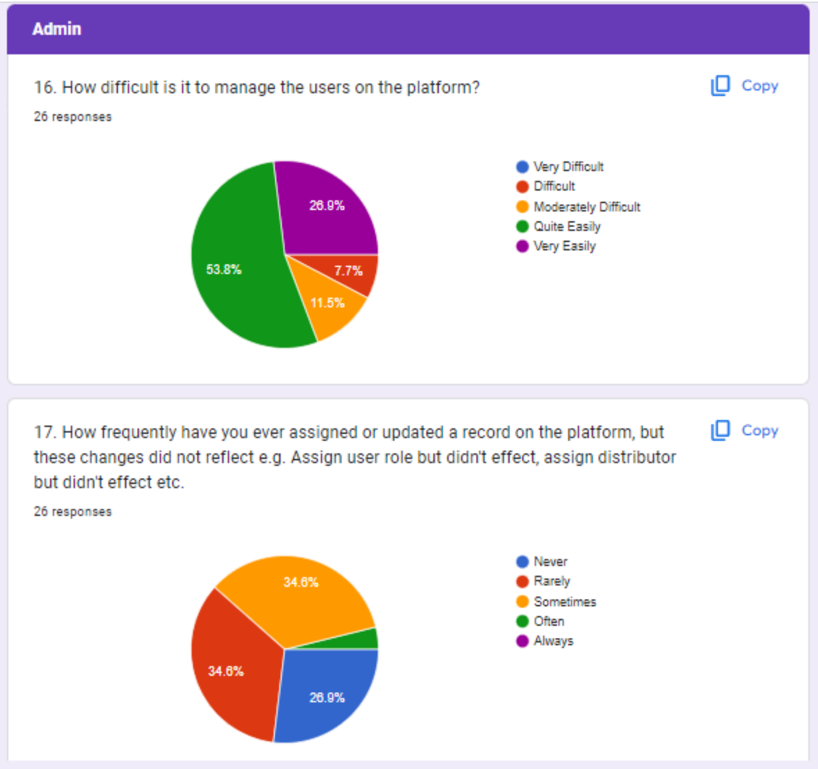
80 responses

15. If Yes, list the page(s) you experience this challenges. Separate the pages by comma

24 responses

- **Role-Specific Questions**: This section pertains to the unique challenges to the users based on their role type.

- Admin



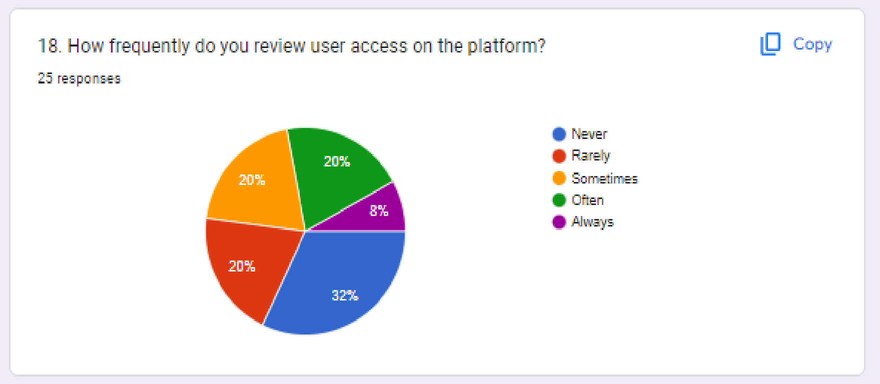
Admin

16. How difficult is it to manage the users on the platform?

26 responses

17. How frequently have you ever assigned or updated a record on the platform, but these changes did not reflect e.g. Assign user role but didn't effect, assign distributor but didn't effect etc.

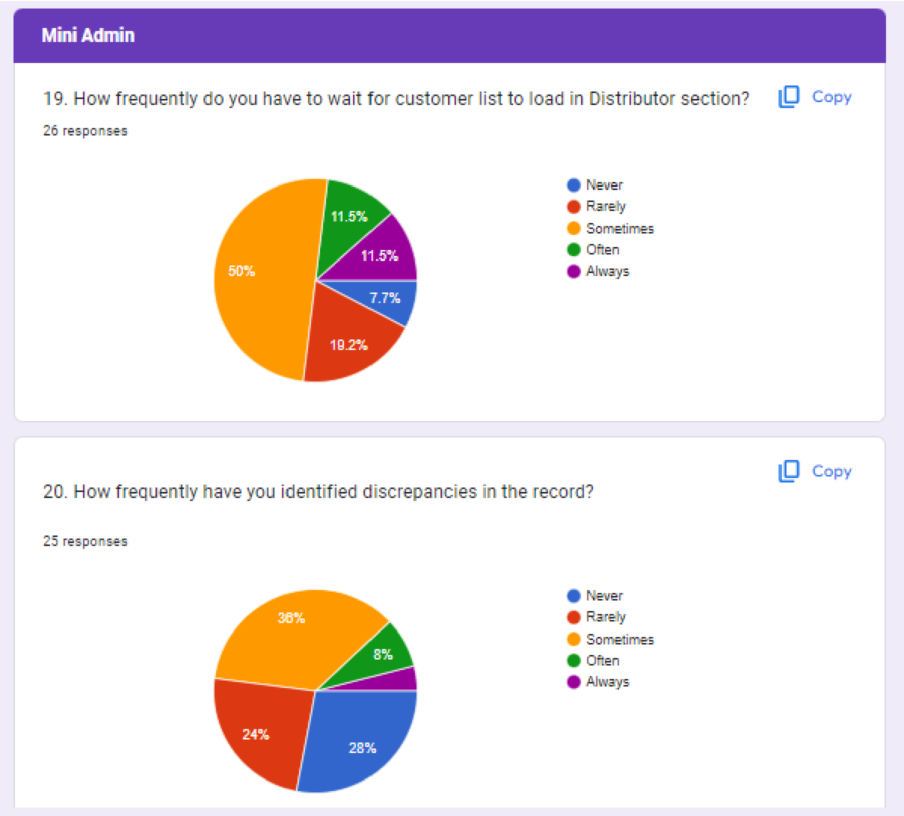
26 responses



18. How frequently do you review user access on the platform?

25 responses

- Mini Admin



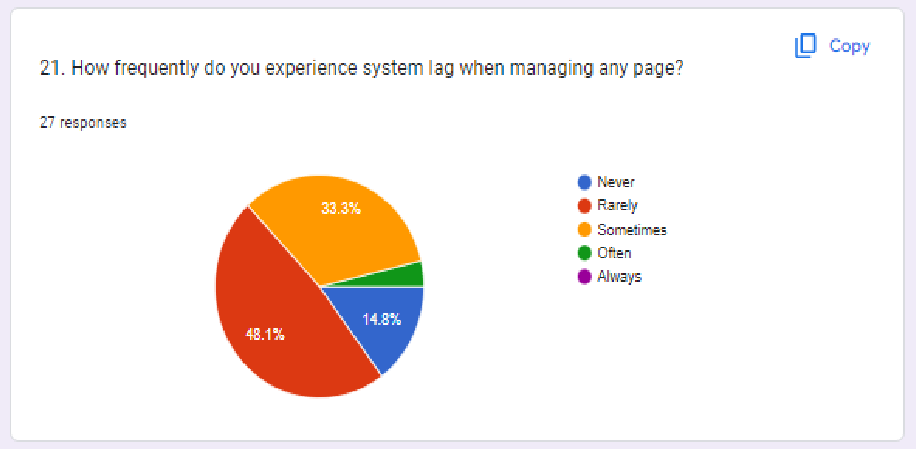
Mini Admin

19. How frequently do you have to wait for customer list to load in Distributor section?

26 responses

20. How frequently have you identified discrepancies in the record?

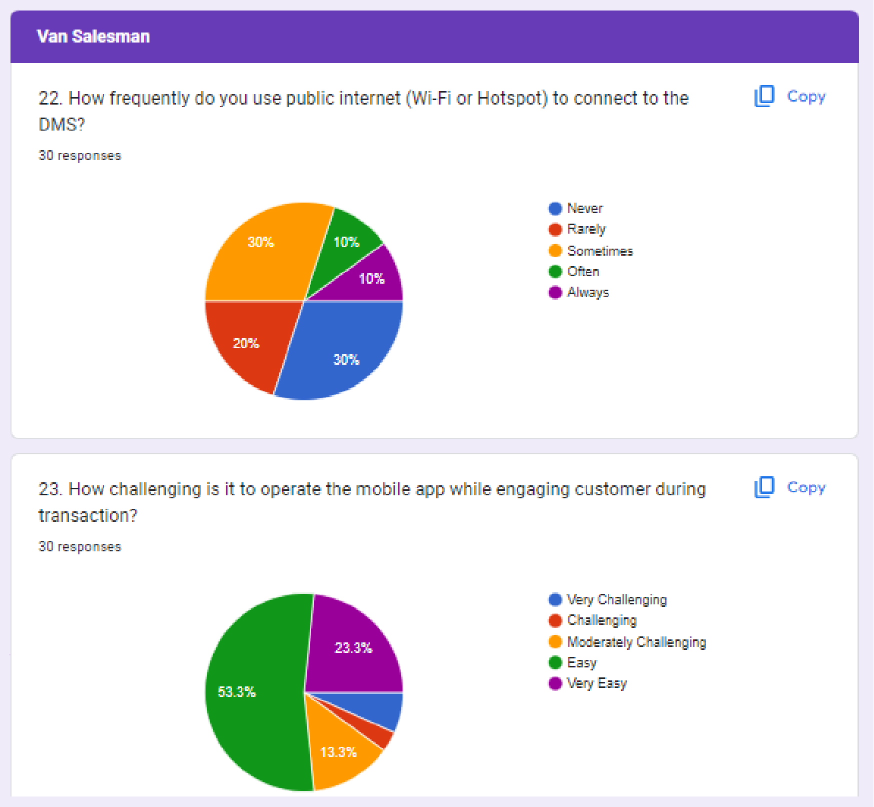
25 responses



21. How frequently do you experience system lag when managing any page?

27 responses

- Van Salesman



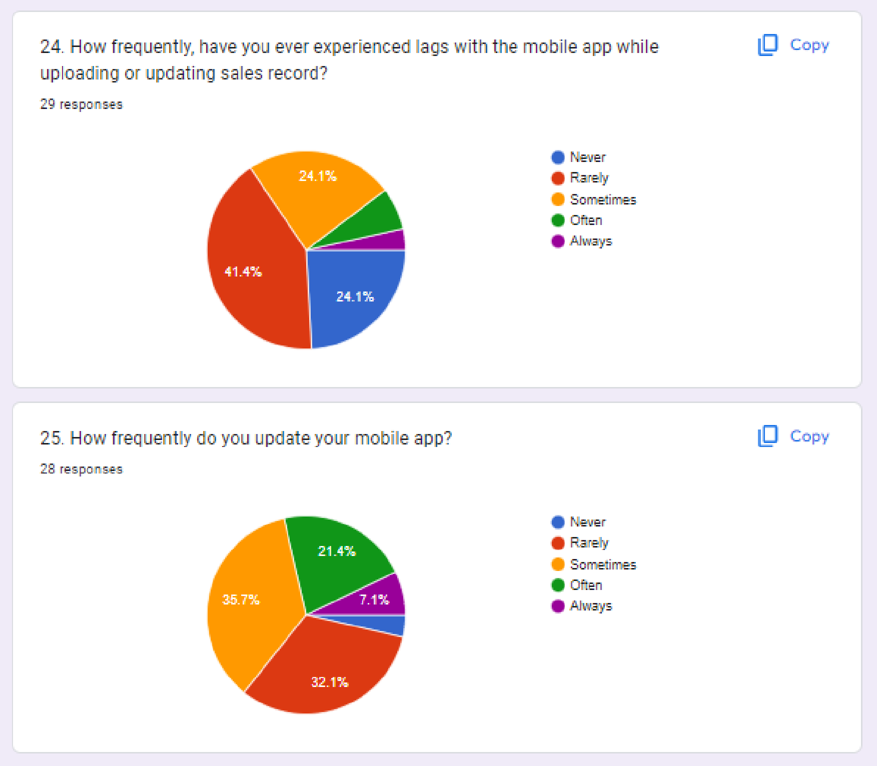
Van Salesman

22. How frequently do you use public internet (Wi-Fi or Hotspot) to connect to the DMS?

30 responses

23. How challenging is it to operate the mobile app while engaging customer during transaction?

30 responses

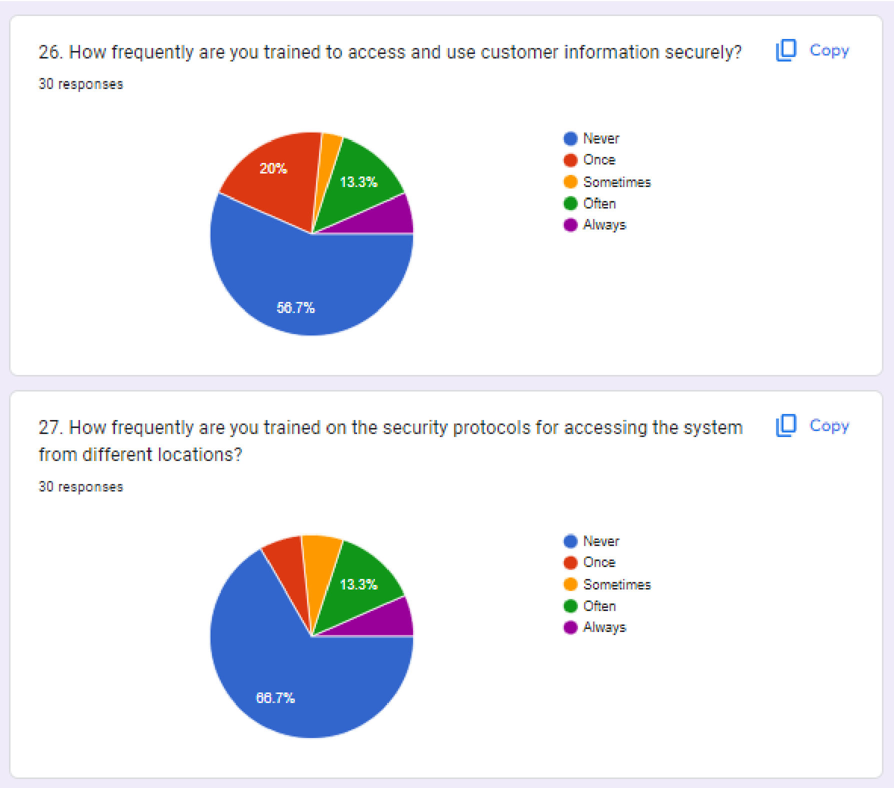


24. How frequently, have you ever experienced lags with the mobile app while uploading or updating sales record?

29 responses

25. How frequently do you update your mobile app?

28 responses



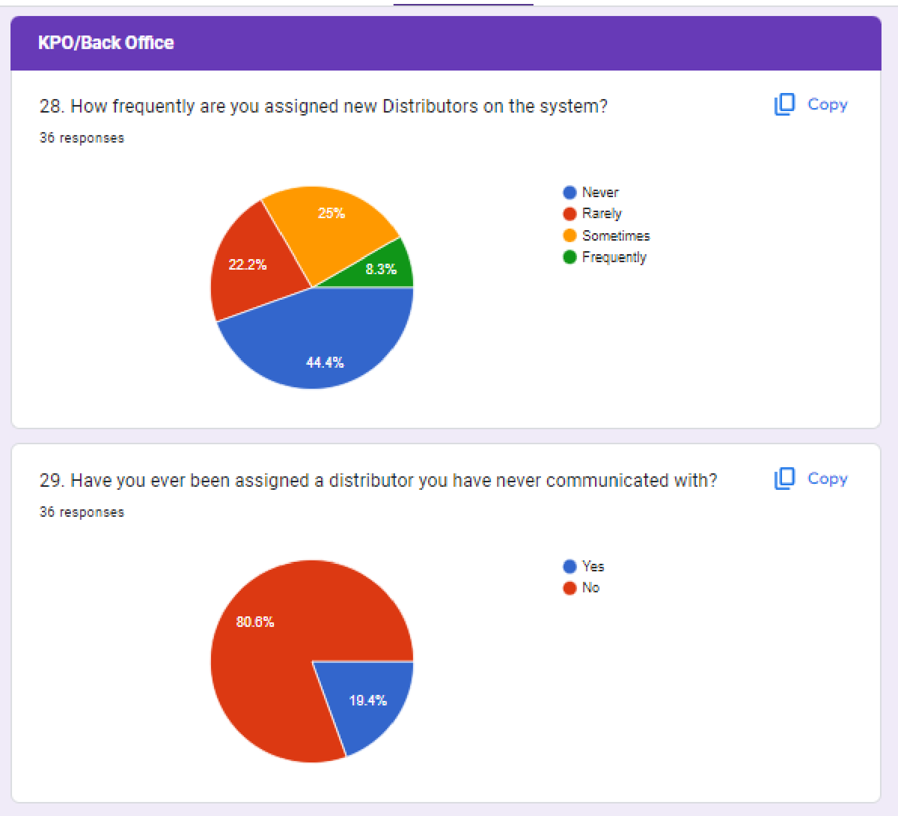
26. How frequently are you trained to access and use customer information securely?

30 responses

27. How frequently are you trained on the security protocols for accessing the system from different locations?

30 responses

- KPO/Back Office



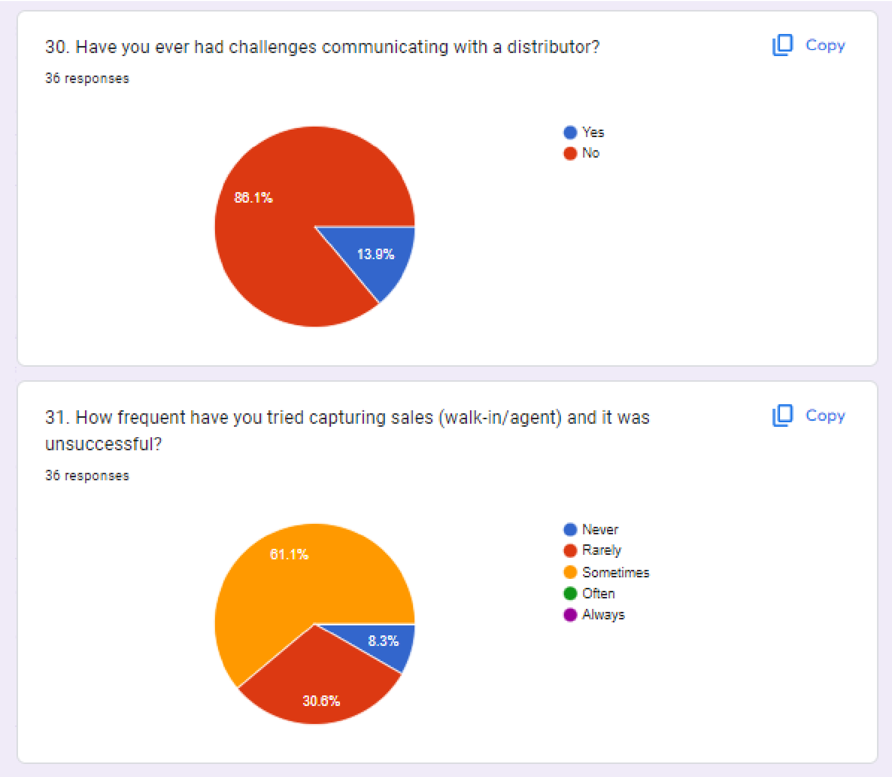
KPO/Back Office

28. How frequently are you assigned new Distributors on the system?

36 responses

29. Have you ever been assigned a distributor you have never communicated with?

36 responses

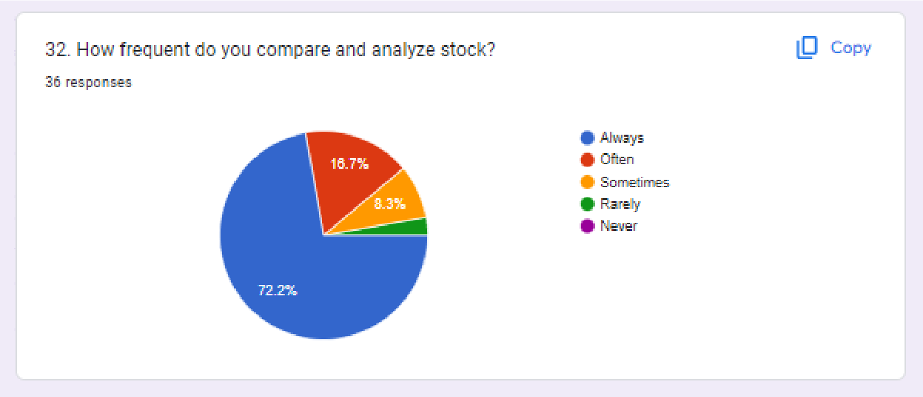


30. Have you ever had challenges communicating with a distributor?

36 responses

31. How frequent have you tried capturing sales (walk-in/agent) and it was unsuccessful?

36 responses

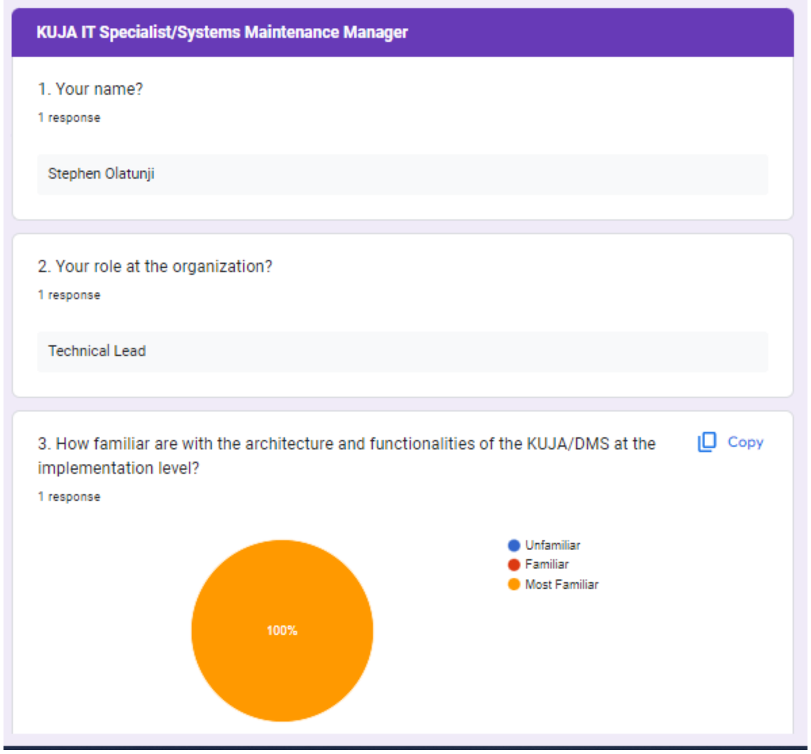


32. How frequent do you compare and analyze stock?

36 responses

SECOND QUESTIONNAIRE

The second questionnaire was sent to the team lead of the DMS platform which provided deeper insights into the architecture and operations of the platform.



KUJA IT Specialist/Systems Maintenance Manager

1. Your name?

1 response

Stephen Olatunji

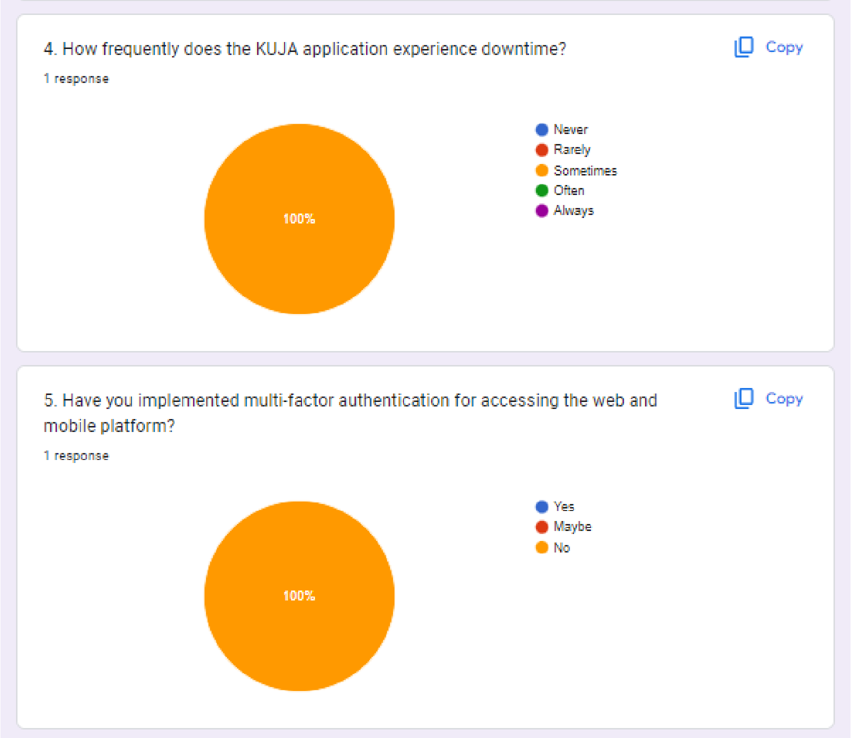
2. Your role at the organization?

1 response

Technical Lead

3. How familiar are with the architecture and functionalities of the KUJA/DMS at the implementation level?

1 response

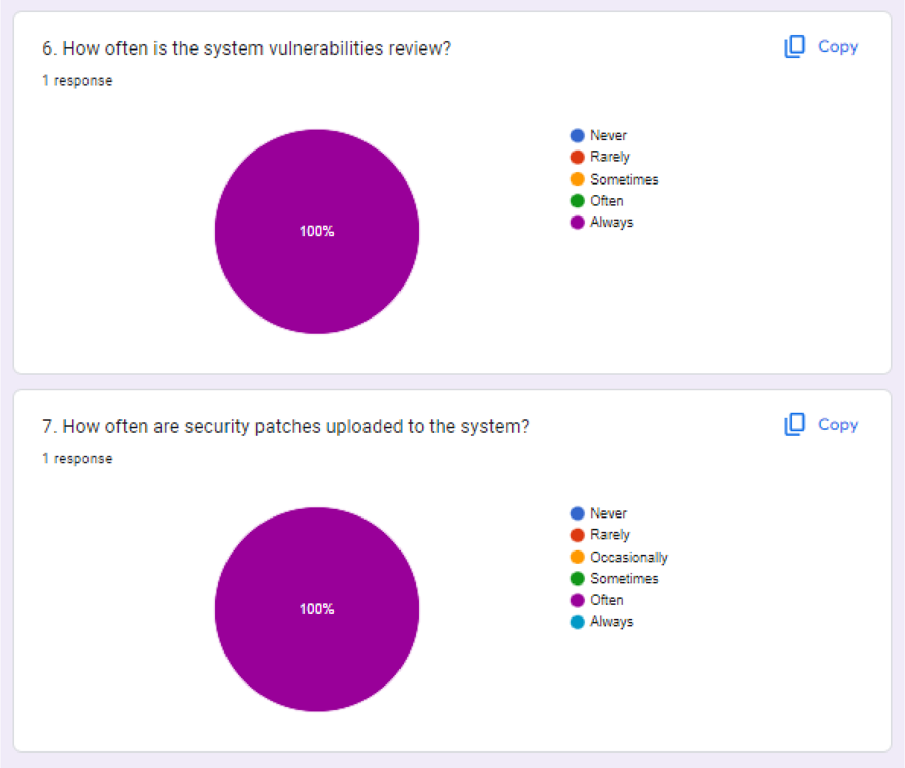


4. How frequently does the KUJA application experience downtime?

1 response

5. Have you implemented multi-factor authentication for accessing the web and mobile platform?

1 response

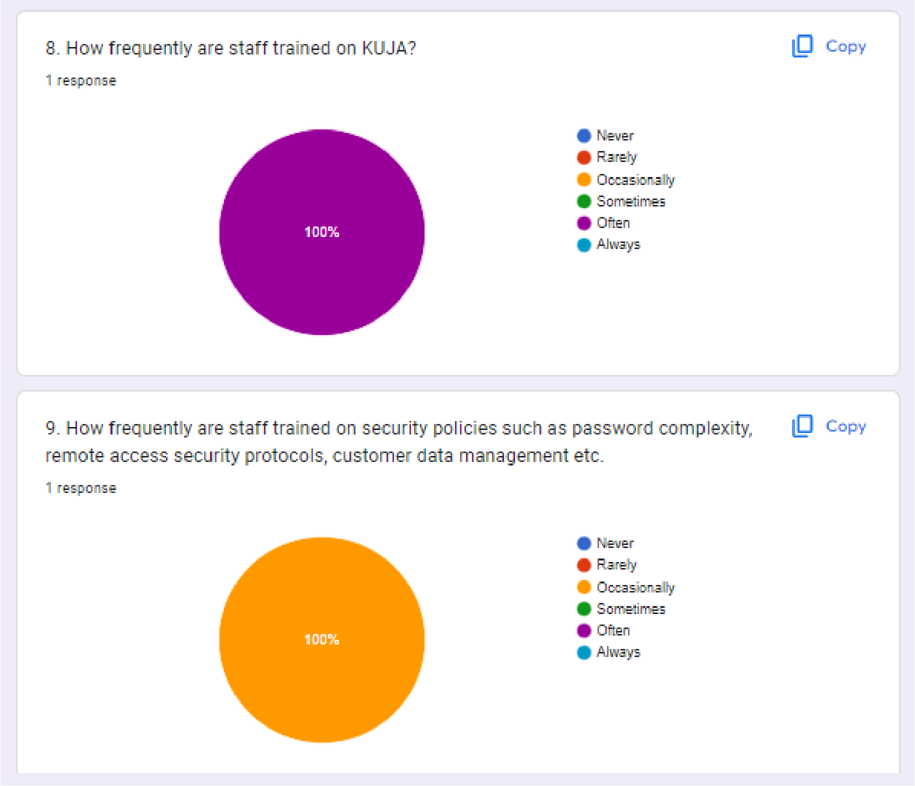


6. How often is the system vulnerabilities review?

1 response

7. How often are security patches uploaded to the system?

1 response

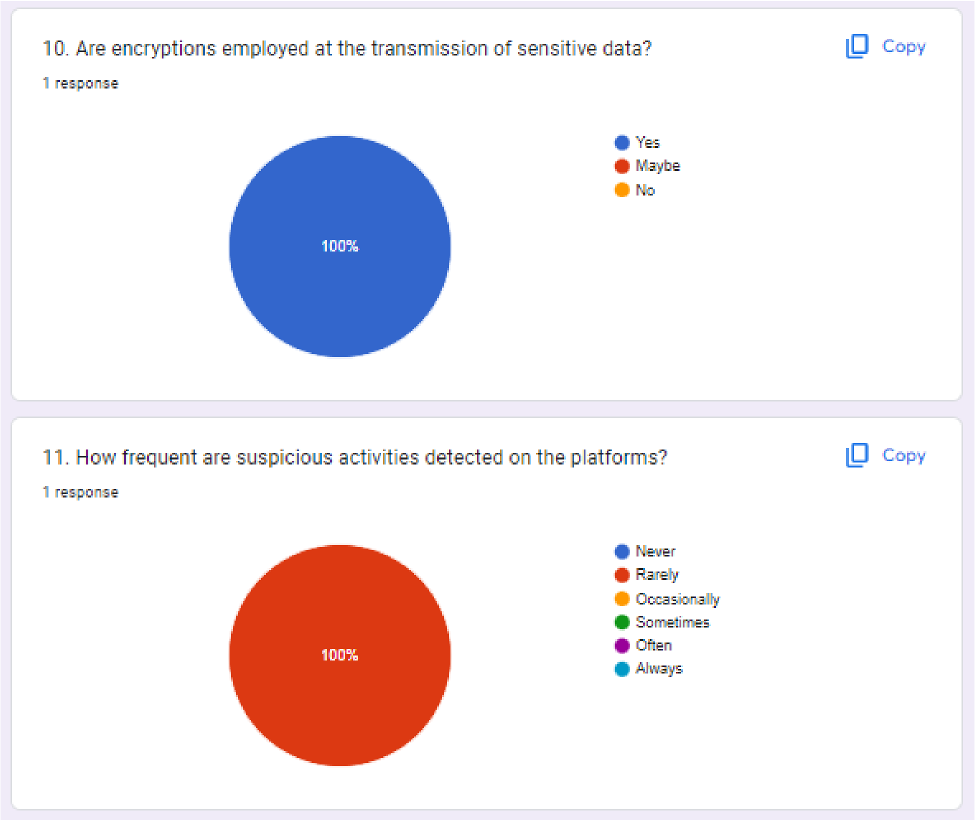


8. How frequently are staff trained on KUJA?

1 response

9. How frequently are staff trained on security policies such as password complexity, remote access security protocols, customer data management etc.

1 response

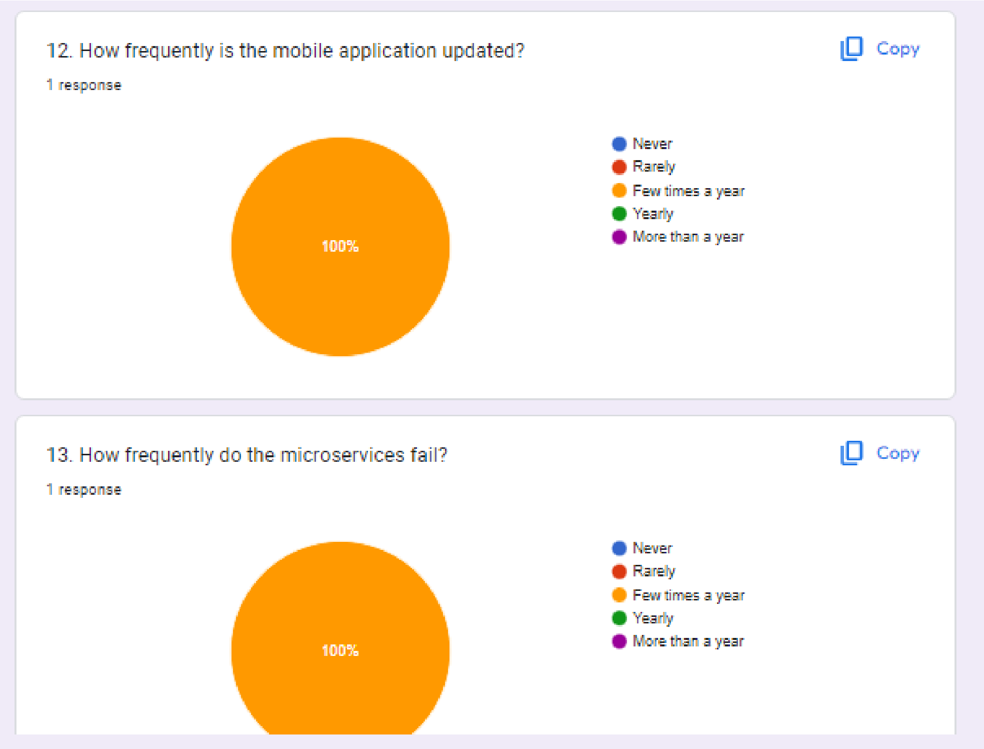


10. Are encryptions employed at the transmission of sensitive data?

1 response

11. How frequent are suspicious activities detected on the platforms?

1 response

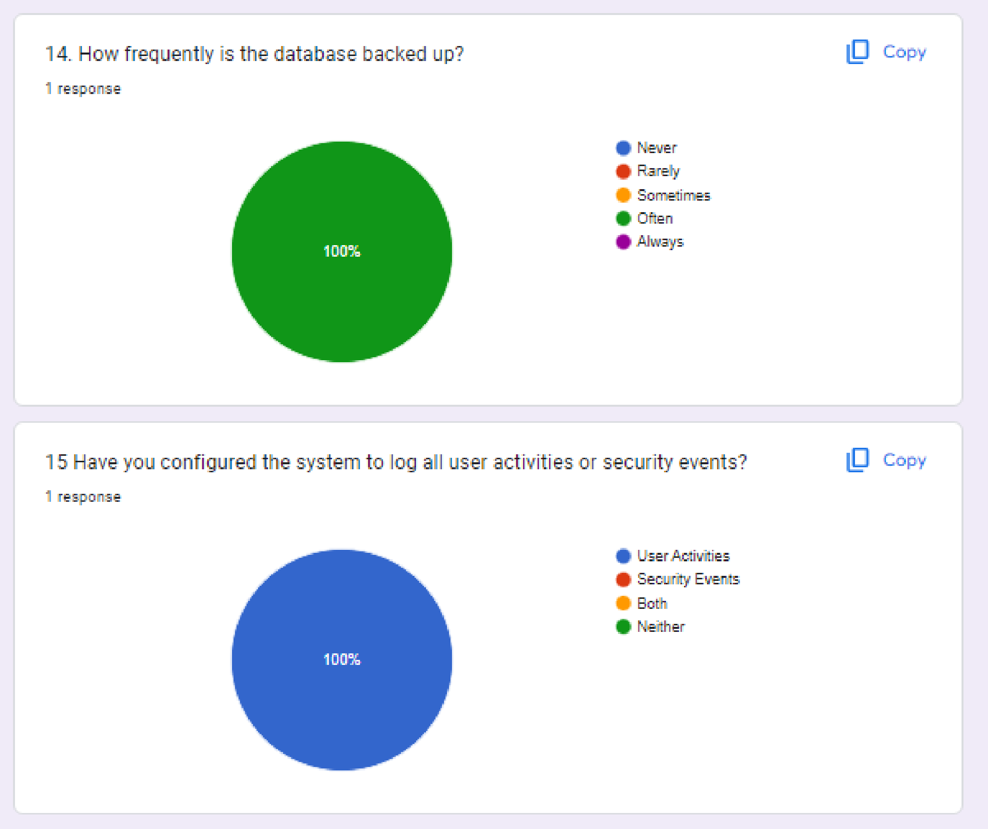


12. How frequently is the mobile application updated?

1 response

13. How frequently do the microservices fail?

1 response

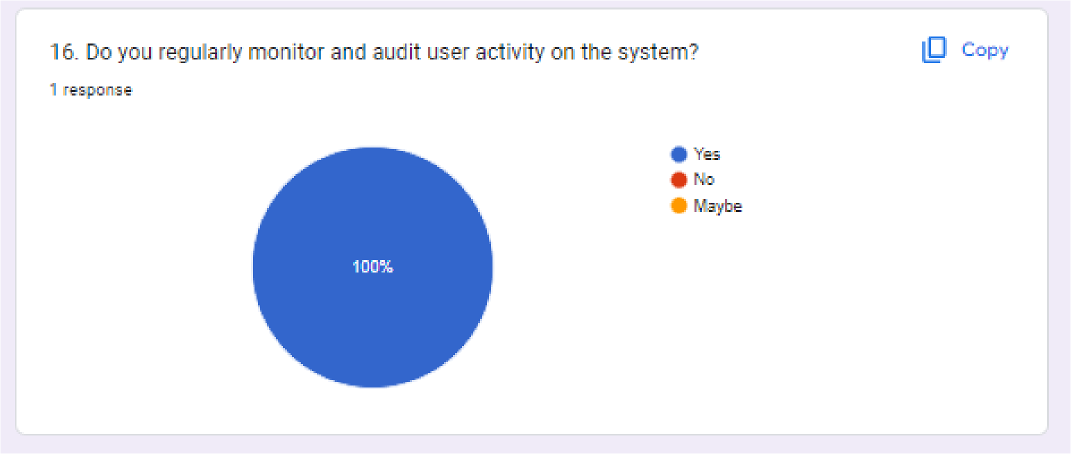


14. How frequently is the database backed up?

1 response

15 Have you configured the system to log all user activities or security events?

1 response



16. Do you regularly monitor and audit user activity on the system?

1 response

APPENDIX E: FULL QUESTIONNAIRE INSIGHTS

Evaluation of First Test Results:

Out of the 80 volunteers that filled the survey,

- 20% are Admin (16)
- 21.2% are Mini-Admin (17)
- 33.8% are KPOs (27)
- 25% are Van Salesman (20)

Based on the questionnaire and responses obtained, the following insights was gotten to ascertain and evaluate the areas where security vulnerabilities are likely to be utilized in an attack using the STRIDE methodology:

User Types: Majority of users are KPOs (33.8%) and Mini Admins (21.2%), indicating that these roles may be at higher risk due to their increased access and responsibilities within the system. Admins and Van Salesmen also form a significant portion of the user base.

Length of Employment: Users who have worked with the company for less than 3 years (33%) and between 3-5 years (19%) may have less familiarity with the system and its security protocols compared to those who have worked for more than 5 years (18%).

Average Weekly Work Hours: Users who work full-time (37.5%) or 24 hours or less (48.8%) may have higher exposure to potential security vulnerabilities due to their frequent use of the system.

Frequency of System Usage: Majority of users (65%) reported using the DMS/KUJA system if they work, indicating high frequency of system usage which may increase the likelihood of encountering security vulnerabilities.

Challenges with Logging In: A significant percentage of users (45%) reported experiencing challenges with logging into their KUJA account, indicating potential security vulnerabilities related to authentication and access controls.

Training on Secure Usage: A notable percentage of users (32.5%) reported never receiving training on securely using the KUJA/DMS application, which may indicate a lack of awareness and knowledge about security best practices.

Password Security: A significant portion of users (57.5%) reported never changing their password on the DMS/KUJA platform, indicating potential vulnerabilities related to weak or outdated passwords.

Training on Security Policies: More than half of the users (55%) reported never receiving training on the security policies and procedures of the system, indicating potential gaps in understanding and awareness of security protocols.

Lag and Unexpected Logouts: Users reported experiencing system lag (46.3%) and unexpected logouts (67.5%), indicating potential vulnerabilities related to system performance, stability, and session management.

Reporting System Failure: A significant percentage of users (88.8%) reported knowing how to report a system failure to admin or IT, which indicates a positive security practice in place.

Role-Specific Vulnerabilities: Admins reported finding it easy to manage users on the platform (53.8%), but Mini Admins reported frequent discrepancies in record updates (34.6%) and system lag (50%). Van Salesmen reported challenges with loading customer lists (50%) and identifying discrepancies in records (36%), indicating potential vulnerabilities specific to their roles.

Based on these insights, potential areas of security vulnerabilities in the DevOps methodology using the "shift left" approach via threat modelling can include authentication and access controls, password security, system performance and stability, user awareness and training on security best practices, and role-specific vulnerabilities. These areas would be further evaluated and addressed using the STRIDE methodology to identify potential threats and vulnerabilities in each category and implement appropriate security measures to mitigate the risks. This would be done by comparing and analysing the technical questionnaire sent to the technical lead of the application.

Further Evaluation of Both Test Results using STRIDE Approach:

Based on the responses provided in the technical questionnaire together with the initial analysis gotten from the first questionnaire, here are some insights that can be gathered using the STRIDE methodology:

Spoofing Evaluation:

The technical team lead, Stephen Olatunji, mentioned being "most familiar" with the architecture and functionalities of the KUJA/DMS system. However, there is no mention of multi-factor authentication being implemented for accessing the web and mobile platforms. This indicates a potential vulnerability in terms of spoofing attacks, where attackers could potentially impersonate legitimate users and gain unauthorized access to the system.

I. Spoof Entity: The questionnaire indicates absence of the implementation of multi-factor authentication for accessing the web and mobile platforms. Also, as a significant portion of users reported never changing their password could potentially allow for spoofing of entities, such as users or system components, where attackers could impersonate legitimate entities and gain unauthorized access to the KUJA/DMS system. Lack of multi-factor authentication increases the risk of spoofing attacks, where attackers could potentially bypass authentication mechanisms and gain unauthorized access.

II. Spoof Process: The response from the questionnaire does indicate that the system has proper logging of user activities or security events. This could potentially stop attacks relating to spoofing of processes, where attackers could manipulate or tamper with data or system processes to gain unauthorized privileges or perform malicious actions without being detected.

III. Spoof Data Flow: The response from the questionnaire does indicate presence of encryption of sensitive data during transmission. This could potentially stop spoofing of data flow, where attackers could intercept or eavesdrop on data transmitted between users and the DMS system.

Tampering Evaluation:

There is no specific response related to data integrity or tampering in the questionnaire, which could indicate a potential oversight in considering this security aspect. Without proper measures in place to ensure data integrity, there is a risk of unauthorized modification or tampering with sensitive data within the KUJA/DMS system.

Tamper Process: The response from the questionnaire does indicate presence of monitoring and regular review of system vulnerabilities. However, it also indicates that staff receive training on security policies occasionally with a notable percentage of users reporting to have never receive security training. This could potentially allow for some attacks associated with tampering of processes like Access to Memory, where a local user, program, or admin with authorized write access to memory can unknowingly tamper with a process leading to vulnerabilities in the system. Lack of regular security policy training increases the risk of tampering attacks on processes, as vulnerabilities may go unnoticed, or staff may not be aware of proper security protocols.

Tamper Data Flow: Just like Spoof data flow, analysis into the responses from both questionnaires does indicate presence of encryption of sensitive data during transmission and adequate monitoring in place for detection of suspicious activities. This reduces the risk of tampering attacks on data flow.

Tamper Data Store: Evaluation into the responses gotten from both questionnaires indicates data is backed up on a regular basis, However, both Mini-admins and Van Salesman reported discrepancies in records which are indications of vulnerabilities linked to tampering of data store. This risk of tampering attacks on data store occurs where attackers could manipulate or modify data in the database, leading to potential data discrepancies or confidentiality breaches. Lack of proper logging increases the risk of tampering attacks on data store, as unauthorized modifications or deletions may go unnoticed or untraceable.

Repudiation Evaluation:

Repudiation Process: Analysis into the responses from the questionnaire indicates lack of multi-factor authentication for accessing the web and mobile platform. Additionally, logging is done on user activities alone and not for security events. This creates a loophole which could potentially allow for repudiation attacks on processes, where attackers could gain unauthorized access to the system or perform actions without proper authentication, leading to potential denial of involvement or repudiation of actions. Lack of multi-factor authentication and logging of security events increases the risk of repudiation attacks on processes, as unauthorized actions may not be traceable to specific events.

Repudiation Data Store: The response from the questionnaire indicates data is backed up on a regular basis but some staff receive training on security policies occasionally while others have never been trained. This could potentially allow for repudiation attacks on data store, where attacks could occur through logs or scattered logs which responding to a repudiation claim become too expensive. As a result, attackers could manipulate or modify data in the database without proper traceability, leading to potential denial of involvement or repudiation of actions. Lack of regular security policy training increases the risk of repudiation attacks on data store.

Information Disclosure Evaluation:

Information Disclosure Process: Proper logging and constant training of security protocols could avoid attacks related to information disclosure process. The response from the questionnaire does indicate this occurs, hence reducing such forms of attack.

Information Disclosure Data Flow: The response from the questionnaire does indicate presence of encryption of sensitive data during transmission and adequate monitoring in place for detection of suspicious activities. Lack of encryption and detection of suspicious activities increases the risk of information disclosure attacks on data flow, as sensitive data may be exposed or intercepted during transmission.

Information Disclosure Data Store: Metadata like name, size and timestamps are properties about the data that can be vulnerable to information disclosure attacks. This can be avoided with regular system vulnerabilities review and security patches added to the system. The response from the questionnaire indicates regular security patches which reduces information disclosure attacks on data store.

Denial of Service (DoS) Evaluation:

The questionnaire mentions that the KUJA application sometimes experiences downtime, and the microservices and mobile application also fail a few times a year. Additionally, Users reported experiencing system lag and unexpected logouts. This indicates a potential vulnerability in terms of availability, where the system may be prone to Denial of Service (DoS) attacks that can disrupt the normal functioning of the system.

Denial of Service Process: The response "Sometimes" to the question about system downtime could indicate that there may be instances where the KUJA application experiences disruptions or downtime, potentially due to DoS attacks on processes. This could lead to denial of service to legitimate users, resulting in service disruptions, unavailability, or degradation of system performance. Proper measures such as regular system vulnerability reviews and staff training on security policies can help prevent DoS attacks on processes and minimize system downtime.

Denial of Service Data Store: The response "Often" to the question about database backups could indicate that database backups are performed regularly, which could help mitigate the risk of data loss due to DoS attacks on data store. Proper data backup and database management can help ensure availability and integrity of data, even in the event of DoS attacks.

Denial of Service Data Flow: The response "Rarely" to the question about detection of suspicious activities on the platforms could indicate that there may be instances where malicious activities or DoS attacks on data flow go undetected. This could potentially result in disruption or denial of service to legitimate data flows, leading to data unavailability or degradation of data integrity. Proper detection mechanisms and monitoring of data flow can help detect and mitigate DoS attacks on data flow.

Elevation of Privilege Evaluation:

The questionnaire mentions that system vulnerabilities are reviewed "always", and security patches are uploaded "often." However, it does not mention the frequency of staff training on security policies or remote access security protocols. We can only deduce that more than half of the users reported never receiving training on the security policies and procedures of the system. This indicates a potential vulnerability in terms of privilege management, where staff may not be adequately trained on security policies, which could lead to potential elevation of privilege attacks by insiders or attackers gaining unauthorized access.

Elevation of Privilege Process: The response "Most Familiar" to the question about familiarity with the architecture and functionalities of the KUJA/DMS at the implementation level could indicate that the technical team lead may have high levels of access or privileges within the system. This could potentially pose a risk of elevation of privilege, where unauthorized access or privilege escalation could occur. It is important to implement proper access controls, privilege management, and role-based access mechanisms to minimize the risk of elevation of privilege attacks.

SUMMARY OF EXISTING ATTACK LOOPHOLES IN DMS USING STRIDE METHODOLOGY

Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Entities	X		X			
Processes	✓	X	X	✓	X	X
Data Flows	✓	✓		✓	X	
Data Stores		X		✓	✓	