



UNIVERSITETET I AGDER

IS-304: 2024

Tittel:

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen & Geir Inge Hausvik
Veileder	Tim A. Majchrzak
Oppdragsgiver:	Netsecurity

Studenter:

Etternavn	Fornavn
Dushantha	Siddharth
Hannestad	Eirik Bakke
Hansen	Tobias Bo
Nymo	Magnus Eugene
Rolf	Victor Johan

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI ___
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI ___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI ___

SafeQR

An anti-phishing tool for automatically searching and parsing QR codes hidden in email attachments.

Siddharth Dushantha
Eirik Bakke Hannestad
Tobias Bo Hansen
Magnus Eugene Nymo
Victor Johan Rolf

SUPERVISOR

Tim A. Majchrzak

University of Agder, 2024

Faculty of Social Sciences

Department of Information Systems

Table of Contents

TABLE OF CONTENTS.....	2
ABSTRACT	6
1 INTRODUCTION.....	8
1.1 CLIENT OVERVIEW AND PROJECT SIGNIFICANCE	8
1.2 PROJECT CRITERIA GIVEN BY NETSECURITY.....	8
1.3 THE TEAM	10
1.4 OVERVIEW OF THE REPORT.....	11
2 BACKGROUND.....	11
2.1 CENTRAL CONCEPTS	11
2.2 PHISHING	12
2.3 HOW QR CODES ARE USED IN PHISHING ATTEMPTS.....	13
2.4 WHY ARE QR CODES USED IN PHISHING ATTACKS?.....	14
3 PROJECT MANAGEMENT	15
3.1 METHODOLOGY.....	15
3.2 OUR USE OF SCRUM.....	16
3.2.1 <i>Task Estimation</i>	16
3.2.2 <i>Prioritization of Tasks</i>	17
3.2.3 <i>Tools for Project Management</i>	17
3.3 QUALITY ASSURANCE	18
3.3.1 <i>Quality Specification</i>	18
3.3.2 <i>QA Activities</i>	19
3.3.3 <i>Risk Analysis</i>	20
3.3.4 <i>Benchmark Test</i>	21
3.4 PLATFORM AND TECHNOLOGY	22
3.4.1 <i>Python</i>	22
3.4.2 <i>API (Application Programming Interface)</i>	22

3.4.3	<i>Cortex XSOAR</i>	23
3.4.4	<i>Microsoft 365</i>	23
4	TECHNICAL CONSIDERATIONS	24
4.1	QR CODES.....	24
4.2	EML.....	25
4.3	XSOAR AUTOMATIONS, INTEGRATIONS, AND PLAYBOOKS	26
4.4	SECURITY	27
4.4.1	<i>Supply-Chain Risks</i>	27
4.4.2	<i>Docker Hardening</i>	28
5	OVERVIEW OF SPRINTS	28
5.1	PRE-SPRINT	28
5.2	SPRINT 1 (22.01.24 – 11.02.24).....	29
5.2.1	<i>Planning Meeting</i>	29
5.2.2	<i>Sprint 1 Retrospective</i>	29
5.3	SPRINT 2 (12.02.24 – 03.03.24).....	30
5.3.1	<i>Planning Meeting</i>	30
5.3.2	<i>Sprint 2 Retrospective</i>	30
5.4	SPRINT 3 (04.03.24 – 24.03.24).....	31
5.4.1	<i>Planning Meeting</i>	31
5.4.2	<i>Sprint 3 Retrospective</i>	31
5.5	SPRINT 4 (25.03.24 – 15.04.24).....	31
5.5.1	<i>Planning Meeting</i>	32
5.5.2	<i>Sprint 4 Retrospective</i>	32
5.6	TIME AFTER SPRINTS	32
6	THE PRODUCT – SAFEQR.....	32
7	LESSONS LEARNED.....	35
7.1	CHALLENGES ENCOUNTERED	35
7.2	IMPACT OF CHALLENGES	35
7.3	MITIGATION OF CHALLENGES	36
7.4	COOPERATION WITH CLIENT.....	36

7.5	COOPERATION WITH ADVISOR.....	37
8	DISCUSSION.....	37
8.1	CHANGES IN TECHNOLOGY.....	37
8.2	GROUP DYNAMIC AND DEVELOPMENT STYLE.....	39
8.2.1	<i>Methodology</i>	40
8.2.2	<i>Integration into XSOAR</i>	40
8.3	SECURITY ASPECTS.....	41
8.4	FURTHER DEVELOPMENT.....	42
8.5	LEARNING OUTCOME.....	43
8.6	VALUE CREATED FOR NETSECURITY.....	44
9	CONCLUSION.....	45
10	BIBLIOGRAPHY.....	46
11	APPENDIX.....	50
11.1	BACKLOG MoSCoW – ANALYSIS.....	50
11.2	BACKLOG – SPRINT OVERVIEW.....	51
11.3	STATEMENT FROM NETSECURITY.....	52
11.4	GROUP CONTRACT.....	53
11.5	RISK ANALYSIS.....	55

Figure List

Figure 1 - Workflow of SafeQR	9
Figure 2 - Quishing email where the QR code is an inline image	14
Figure 3 - Example email where QR code is in attachment	14
Figure 4 - Kanban board	18
Figure 5 - Risk Matrix.....	20
Figure 6 - Classification of risk, probability, severity, its rating, and measures taken	21
Figure 7 - Visualization of timed tests of SafeQR	21
Figure 8 - Typical QR code formatting.....	25
Figure 9 - Example of EML/IMF file, inspired by RFC 5322 (Resnick, 2008, p. 43)	26
Figure 10 - XSOAR dashboard with SafeQR Integration	27
Figure 11 - Finished SafeQR in the XSOAR dashboard	33
Figure 12 - Detailed flowchart of SafeQR	34

Abstract

For the spring semester of 2024, our team collaborated with Netsecurity for our bachelor project. Our goal was to develop a script capable of detecting and decoding QR codes in emails reported by Netsecurity's customers and display the URLs in XSOAR, the platform used by their security analysts to monitor security incidents. Implementing this script allowed us to streamline the handling of phishing with the help of QR codes (*quishing*).

Utilizing Python to develop our script enabled us to integrate it into the XSOAR platform. We looked at various other technologies during our development. Technologies such as Docker, a dedicated server, and concepts such as API were once considered but were ultimately not included in the final product. Partially due to security concerns, but later revealed to be a misunderstanding in the scope of our assignment. Regardless, the team gained valuable experience and knowledge by using these technologies.

The group completed our project within the specified timeframe and fulfilled the system requirements provided by Netsecurity. Simultaneously, we proposed and implemented additional features that, upon agreement with Netsecurity, further enhanced the project. Certain limitations were identified and demonstrated to our contact person, such as SVG files in PDFs and QR code ASCII art. Incidents with these properties were rarely seen or encountered, so these limitations were deemed acceptable.

Our team gained significant experience and knowledge during this project. Through our experience, we gained a deeper understanding of development, security, and project management. Working at a company and solving a real-world problem was rewarding and exciting. Netsecurity needs to conduct a security analysis of our script to ensure they can safely integrate it into their production version of XSOAR. However, by implementing it into their development environment, we have demonstrated that it works as intended.

Preface

As the world increasingly adopts digital processes and more people use digital devices than ever, digital fraud is becoming more prevalent (International, 2023). Accounting for significant drops in revenue for companies affected by attacks (Help Net Security, 2023). Falling victim to such an attack does not only cost companies money in the form of damage control and potential replacements of hardware or software. It can also cost them revenue, caused by customers losing trust and taking their business elsewhere.

The victims of these attacks also generally feel less safe in digital spaces and will hesitate to use digital tools in their day-to-day lives, which can decrease quality of life as digital processes permeate many aspects of life.

Netsecurity, a cybersecurity company from Kristiansand, has reported a rise in *quishing* incidents. This form of phishing attack involves the use of QR codes, prompting Netsecurity to optimize its approach to addressing these threats.

We want to thank Netsecurity for taking us in and trusting us with this project and for the invaluable help we received throughout the process. We would also like to thank our contact person, Odd-Egil Solheim Egeland, who continuously followed up with us, answered questions efficiently, and arranged meetings with Netsecurity employees when we needed expertise. Which also extends our gratitude to Dag-Philip Lango Thorbjørnsen, Erlend Halsnes, and Martin Langballe.

We also want to thank our course coordinators, Hallgeir Nilsen and Geir Inge Hausvik, for their informative lectures and valuable feedback sessions throughout the semester. Our supervisor, Tim A. Majchrzak, also helped throughout the project with technical assistance and consultation on writing our bachelor report.

1 Introduction

1.1 Client Overview and Project Significance

Netsecurity, the client for our bachelor project, specializes in cybersecurity solutions for private and public organizations. Employing approximately 150 individuals in four cities in Norway, the company offers various cybersecurity services, including operating a Security Operations Center (SOC), conducting penetration testing, and providing specialized consulting.

A notable rise in digital fraud, specifically using QR codes in phishing attacks, has recently been identified. Having observed this trend directly, Netsecurity enlisted our bachelor student group to develop a tool for detecting and decoding QR codes in phishing emails reported by their customers. Their current method, which involves importing each QR code into a virtual environment to check the destination URL, is time-consuming and resource-demanding.

Our project's significance lies in the potential to boost Netsecurity's capabilities in effectively identifying and mitigating phishing attempts that leverage QR codes. By automating the process of decoding QR codes, we aim to provide Netsecurity with an effective and reliable way of extracting data from QR codes, enabling the company to protect their clients from QR code phishing attempts. This collaboration provides a practical, real-world application of the skills we developed during our bachelor's program, benefiting both our educational goals and Netsecurity's operational effectiveness.

1.2 Project Criteria Given by Netsecurity

Netsecurity gave us the objective of developing a tool capable of identifying QR codes in files attached to emails reported as phishing by Netsecurity's customers. The script's primary function was to automate the detection of QR codes, with the desired capability of decoding information contained within them.

QR codes are often embedded within the body of an email or attached as files, serving as vectors in phishing attempts. Given the variety of formats in which these QR codes can be presented, for example, in PDF files, Office files, and images, the script needed to be created with the capability to support a broad selection of file types. However, extending this capability beyond the initial requirements was considered desirable rather than mandatory.

The specification document outlined the following requirements:

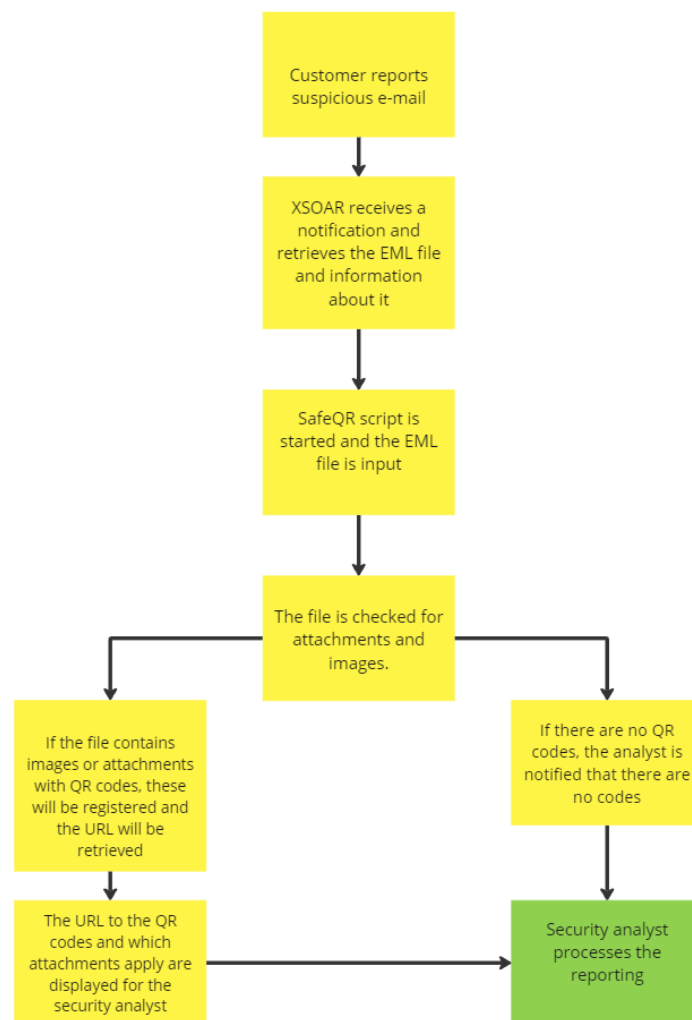
Must-have requirements

- Can identify QR codes in one or multiple files of the following formats: .pdf, .jpg, and .png
- Must report any findings.

Desired requirements

- Support for a wider variety of file formats.
- Written in Python for integration with their SOAR (Security Orchestration, Automation, and Response) platform.
- Extract the information contained within the QR codes.

Figure 1 visually depicts the functions of the script.



miro

Figure 1 - Workflow of SafeQR

- 1) **Incident Report Initiation:** A Customer reports a suspicious email, which triggers the security protocol.

- 2) **Alert and Retrieval:** XSOAR receives a notification and retrieves the relevant EML file containing the email content.
- 3) **Script activation:** Upon receipt of the EML file, our script is initiated to process the contents.
- 4) **Content analysis:** The script checks the email for any attachments or images containing QR codes.
- 5) **QR Code Detection and Decoding:**
 - a. If QR codes are found within the attachments or images, the program extracts them, and the embedded URLs are returned.
 - b. If there are no QR codes, the analyst is informed that there are no QR codes.
- 6) **Results After Decoding:** The URLs and relevant attachment details of detected QR codes are presented to the security analyst.
- 7) **Threat Assessment and Response:** The security analyst evaluates the decoded information to determine the appropriate action.

1.3 The Team

Group 17 comprises five students with a history of successful collaboration on different projects within the IT and Information Systems bachelor's program. Our teamwork over these three years has refined our problem-solving approach, enabling us to work effectively and tackle various challenges.

Victor: Able to work both with technical and administrative tasks.

Magnus: Experienced in software development and information systems security. Took responsibility for deploying and administering a realistic test environment for the project.

Siddharth: Use previous programming and cybersecurity experience to complete the project's administrative and technical work.

Eirik: Proficient in both the technical and administrative work involved in projects. Earlier courses included Web programming, universal design, and information systems security.

Tobias: Many years of experience from working with IT infrastructure.

Our group's various skills and collective dedication have made us a cohesive team and enabled us to tackle various development and project management challenges. All group members contributed to the coding of SafeQR. We split up the group regularly to ensure that we employed our strengths where needed and to have some variety in our assignments. Siddharth

was appointed group leader, but we maintained a flat hierarchy so every group member could make suggestions and actively participate in decision-making. Siddharth works at Netsecurity and was a natural choice with his knowledge of the company and their employees.

1.4 Overview of The Report

The remainder of this report is structured as follows: First, in Chapter 2, we will discuss the project's background, explain some central concepts, and lay out essential information regarding our project. Secondly, we explain in Chapter 3 how we managed our project through our use of SCRUM, quality assurances, and platform and technology aspects of the management. We then move on to the technical considerations of our project in Chapter 4. The work sprints we planned for our project are next in Chapter 5, where each sprint and their respective contributions to the project are explained. After the sprints, in Chapter 6 we present the final product. In Chapter 7, we list some of the lessons learned during our project, their consequences, our mitigation of them, and how we cooperated with our client and advisor. Before our conclusion, we discuss our project in Chapter 8 and the themes and aspects it brought with it. Lastly, in Chapter 9, we conclude our project with our final thoughts.

2 Background

We have summarized and defined the central concepts that pertained significantly to our project. We deemed other central concepts better presented with more context and will elaborate further in the report.

2.1 Central Concepts

The following section introduces key concepts relevant to our bachelor's project, including Security Operation Center (SOC), Cortex XSOAR, automation, playbooks, and QR codes. This information ensures that readers are familiar with essential elements involved in the project. More detailed explanations of these concepts, along with their roles and significance within our project's scope, are provided in Chapters 4 and 5.

Security Operations Center (SOC) is a unit that deals with cyber threats. It comprises different security analyst teams that oversee incidents and manage security operations. The primary function of a SOC is to monitor, assess, and defend against cybersecurity threats and incidents (Palo Alto Networks, n.d.).

Cortex XSOAR, often abbreviated to XSOAR, is a Security Orchestration, Automation, and Response (SOAR) platform by Palo Alto Networks (2021).

Automation transforms manual tasks into automatic processes, reducing the need for human intervention (IBM, n.d.). By automating specific tasks, companies can reallocate their human resources to more strategic activities, allowing routine operations to be managed by automation.

Playbooks, in cybersecurity terminology, are a predetermined plan for what measures need to be taken when specific threat incidents happen. It can range from small tasks such as deleting a suspicious email to vast sweeping operations to decrease the damage taken from an attack (Cofense, n.d.).

EML is a file format used by email clients to store email messages for archival and later use. EML files contain the entire email message, including the body as plain text or HTML, and headers such as the sender, recipient, subject, and timestamp. EML files can also include attachments, such as files and images sent with the email (Indeed Editorial Team, 2022).

QR codes are two-dimensional matrix codes representing up to 7089 characters, depending on the data type and the QR code version used. The data is stored by arranging squares within a square grid, encoded according to a specific pattern representing numeric, alphanumeric, kanji, or binary data.

2.2 Phishing

Phishing is one of the earliest known cyber-attacks observed over the Internet. In 1995, Koceilah Rekouche (2011) created a program called AOHell, which provided an automatic mechanism for stealing passwords and credit card information through AOL services, an all-in-one online service tool. Since then, phishing attacks have been distributed through a variety of means.

One of the latest developments is called *quishing*, where the malicious phishing payload is encoded into a QR code. When the user scans this QR code, it is the equivalent of clicking a URL, taking the user to a malicious website created to steal the victim's information (Cloudflare, n.d.).

In recent years, the landscape of cyber threats has evolved dramatically, with phishing emails increasingly using QR codes as a tool for malicious activities(Griffiths, 2024). This increase is making our project a vital tool to face these challenges.

The adoption of QR codes has surged for legitimate applications, as has their exploitation by cybercriminals. The use of QR codes in phishing emails has risen significantly in recent years. A study by Hoxhunt (2023) revealed that in October 2023, 22% of phishing attempts employed QR codes to distribute malicious payloads. The research included 38 organizations from nine distinct industries, with participants from 125 countries. This benchmark included close to 600,000 participants. Statistics now show that QR codes are potent for cyber-attacks. As phishing attacks become increasingly sophisticated, it raises an urgency to develop efficient methods for decoding and analyzing QR codes within emails and their attachments. This need is essential for companies like Netsecurity, whose job is to respond to cyber threats.

2.3 How QR Codes are Used in Phishing Attempts

Provided below are some examples of *quishing* that we used to test SafeQR. Common among *quishing* attempts identified by Netsecurity are QR codes embedded in the email body as inline images and as attachments. Attackers often evoke a sense of trust by emulating emails from known organizations and companies (Irwin, 2022).

Embedded directly into the email body as inline images

Figure 2 shows a *quishing* email with the QR code as an inline image. Attackers use this method to compromise users easily.

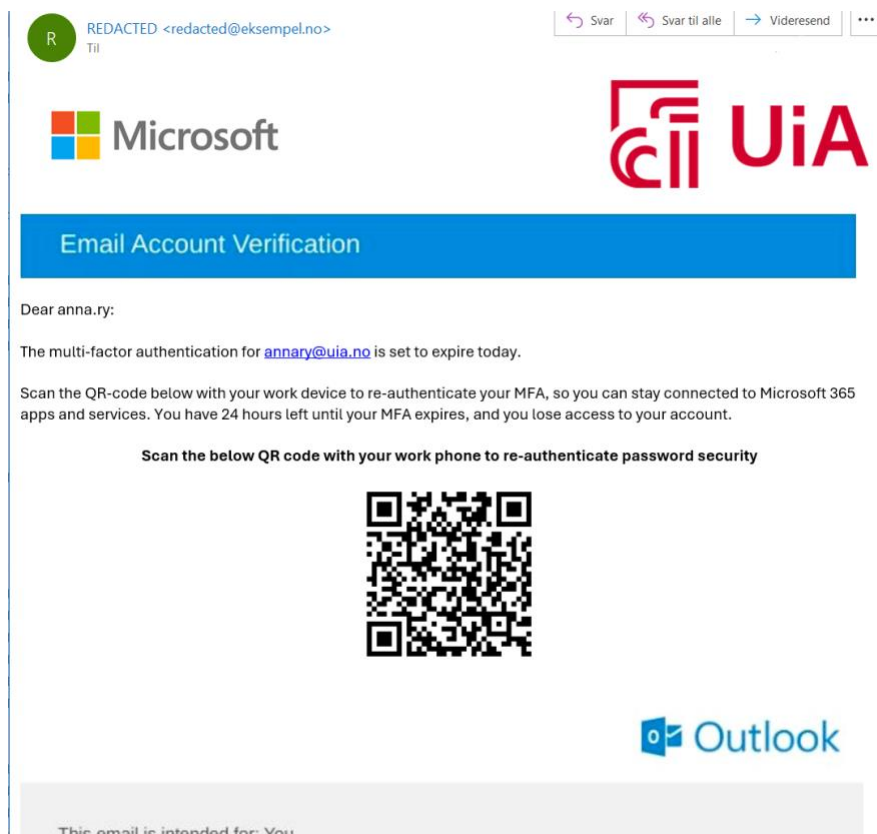


Figure 2 - Quishing email where the QR code is an inline image

Having the QR code inline in the message, as shown in Figure 2 gives the victim a perceived ease of use. Scanning them to see what they contain quickly is easy, leveraging natural curiosity that contributes to this method often used.

In attachments

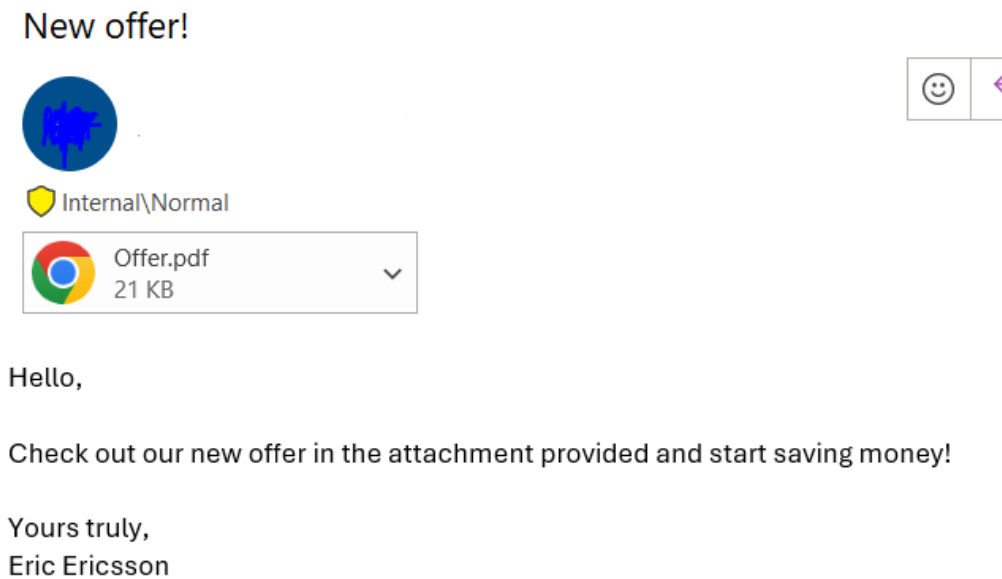


Figure 3 - Example email where QR code is in attachment

The ability to include attachments in the email is something that attackers exploit. An attacker can state in the body of the email that the attachments contain essential information and that the user needs to open and learn this information. There has not been a dedicated tool to detect these QR codes as attackers “hide” them in the attachments, which scanners do not detect, but users can still find and scan them. After researching similar tools that scan emails for QR codes, the group saw that only a few mention QR codes in attachments but do not specify if they scan them (Gandhi, 2023). Therefore, our tool needed to accommodate this need.

2.4 Why are QR Codes Used in Phishing Attacks?

There are several reasons why malicious actors use QR codes instead of regular URLs in their phishing attacks. QR codes give the malicious actor advantages, such as ease of use. Numerous websites can generate QR codes from any URL, allowing easy export into multiple file types. Another advantage is their perceived convenience from users, as QR codes are used in many benign settings. Users often accept scanning the QR codes without much consideration of what

they might contain or lead to, as the codes must be rendered before the information inside them can be read.

Although QR codes are not inherently dangerous, not all devices allow for previewing the content before accessing the linked site, making verifying the destination's safety challenging before proceeding. This obscurity is a significant advantage to malicious actors, enabling them to disguise harmful links.

Most users have smartphones that have built-in functionality to scan QR codes. Therefore, upon receiving a phishing email containing a QR code, the easiest solution is to scan the code with one's personal device. Which is not as hardened as company-issued devices, providing a gateway for attackers. Working from home became normalized after the COVID-19 pandemic, which puts the potential victims in an unprotected environment when on their home network (Silver, 2023).

3 Project Management

3.1 Methodology

Our project's methodology is based on agile principles, leading us to implement a hybrid approach that combines SCRUM and Kanban, commonly called Scrumban (Kanban Tool, n.d.). This decision was taken to combine structured planning and the review elements of SCRUM with the flow and flexibility rooted in Kanban. The group has experience and good results derived from using this approach from earlier projects, which also contributed to adapting this methodology into our project.

Agile methodology emphasizes iterative development, which involves dividing projects into sprints. During each sprint, the workload is divided into defined tasks and timed for completion within the sprint or a specific timeframe (Association for Project Management, n.d.). Once the sprint is complete, the team gathers to review the results of the work done and consider what went well and what could have been done better. These meetings are known as sprint reviews and retrospectives. This methodology suits digital development projects as each iteration's philosophy is continuous improvement and a defined product.

Scrum is a framework within the Agile methodology that enables teams to work on projects through iterative and incremental practices (Scrum.org, n.d.). We used this framework to ensure our development was experiencing progress. There is a plethora of ways to implement this framework. Our past experiences with it have shaped our scrum approach to better suit this

project. In Chapter 9, we will reflect on our choices and discuss our experiences with these decisions.

3.2 Our Use of SCRUM

In the project, SCRUM is being used to structure the overall development process. We organized the work needed in the project into sprints, where each sprint focuses on a specific set of required features in the project. The different SCRUM meetings have guided the planning, how work has been performed, and the retrospective analysis, ensuring that we remained aligned with the project goals and responded to any challenges that were raised.

We conducted “daily scrum” meetings each Monday and Friday. Sprint retrospectives were performed at the end of each sprint, which is three weeks long. The sprint review was conducted on the same day as the sprint retrospectives.

We decided to do “daily scrum” two times a week as the group agreed that too much planning and meetings might stagnate the progress and workflows undertaken in the given week. The week’s first meeting was to plan the rest of the week and consider how the previous week went, and the second meeting was to review, summarize, and tie up loose ends. This balance gave the group time to plan and consider different aspects of the project and the work being done while giving time to do the job, self-study, and other elements such as meetings with the company.

3.2.1 Task Estimation

During the creation of tasks in sprint planning, all tasks were given an estimated size for better planning. Many of the tasks required knowledge that the group members did not have at the time, which made it more challenging to estimate the size of the tasks.

By delegating the tasks before estimating their size, we considered the different skill sets of the group members when estimating the time needed. Many of the functions also required some degree of self-study before being able to start working on the tasks. This time was taken into consideration when estimating their size. Tasks were therefore split into small, medium, or large. Small for functions estimated to take half a day of work, medium for a full day of work, and large for tasks that would take 1-2 days. The group considered tasks that required longer timeframes for completion to be too large and, therefore, split them into small, medium, or large tasks.

3.2.2 Prioritization of Tasks

The MoSCoW framework was combined with a Kanban board to prioritize the tasks in order of importance. The MoSCoW method is a prioritization method where each task is prioritized with either “Must Have,” “Should Have,” “Could Have,” or “Won’t have” (Agile Business, 2014). Using the MoSCoW method helped identify what tasks to prioritize, especially in the early phases of the project, which increased the work’s efficiency and effectiveness.

3.2.3 Tools for Project Management

Discord was used as our primary platform for communication among the group members, allowing us to discuss different topics and share resources that seemed helpful to the project. In addition, it was used to communicate with our advisor.

Miro served as our collaborative whiteboard for brainstorming and visualizing the code’s functionality before development. Choosing a digital over a physical whiteboard offered significant advantages: enhanced remote collaboration, creating and modifying copies, and saving the content for later use.

Git was utilized as our distributed version control system, allowing us to track changes in our source. This allowed multiple developers to collaborate on the code efficiently, forming our development’s basis.

GitHub served as our code storage platform, facilitating collaboration during development. In addition, it helped us with task management, allowing us to track changes and address issues efficiently.

Kanban provided a continuous visual representation of the workflow, complementing SCRUM. We chose to use GitHub Projects, where we created a Kanban board that suited our development process. Using GitHub for code sharing, collaboration, and Kanban, we could track our progress all in one place.

All tasks listed on the board were converted into GitHub issues, facilitating discussions around each task. These issues are centralized points for discussing and tracking progress on specific tasks. Furthermore, these issues were linked to pull requests, where group members could propose and discuss new changes related to the tasks. This integration automated the Kanban board, directly reflecting the pull request status on the overview and enhancing collaboration and workflow.

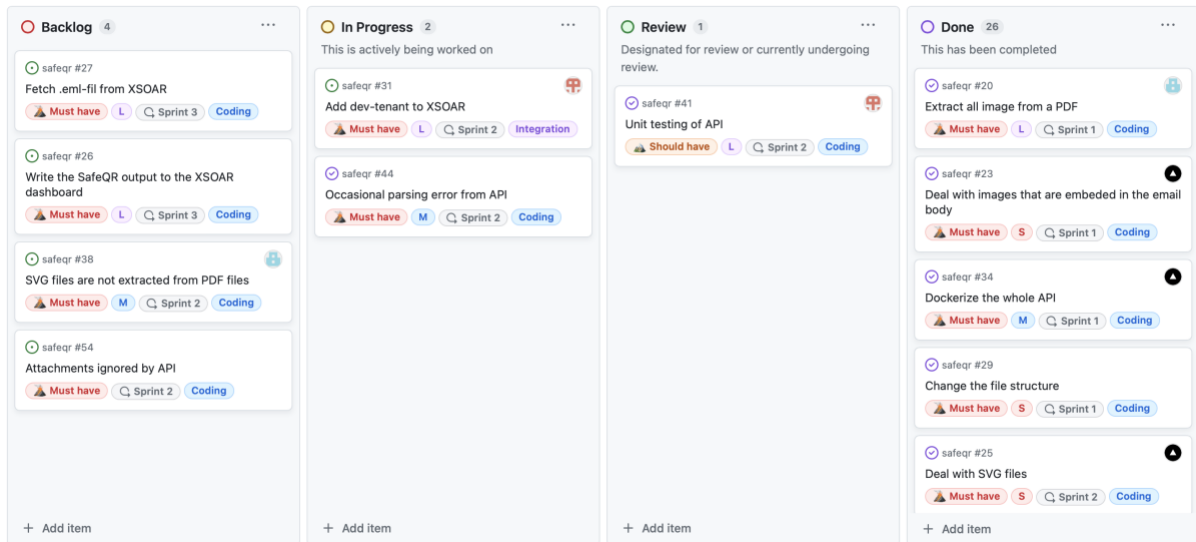


Figure 4 - Kanban board

Tasks were also assigned with tags that defined the sprint they belonged to, their estimated size, the task's relation, the timeframe it was completed, and the assignees responsible. This also allowed sorting the board to create a more uncluttered visualization.

3.3 Quality Assurance

Quality Assurance played a significant role in ensuring the success and reliability of the project. We used different metrics to maintain standards and the quality of the work throughout the project. The use of quality assurance practices has not only made us meet the requirements given by the client but also ensured a good work balance in the group, which has made us work effectively and maintain quality throughout the project.

3.3.1 Quality Specification

We have chosen to define quality in this project by working through the specifications provided by the product owner. Based on this, we presented a flow chart to ensure common ground between the group and Netsecurity. We maintained this common ground in expectations and functionality by continually reviewing new features with Netsecurity.

We did not deem it necessary to organize interviews and user testing because the project was a background process almost invisible to the users. The client gave us fixed requirements for the solution. As such, interviews and research were not a demand made by the client either. This solidified our stance that the quality of our product needed to be derived from the specifications provided and verified by the product owner. As our product owner works closely

with the SOC at Netsecurity, his verification was deemed valuable. To maintain quality, we conducted some QA activities.

3.3.2 QA Activities

The QA activities we have used in the project include:

Code reviews: Code reviews were conducted every time the group merged code into the GitHub repository. This was done to ensure that the codebase in GitHub remained clean and efficient. By making other members review the code before merging, we ensured that the code met our standards and worked on different devices.

Meetings with the product owner: The group scheduled weekly meetings to discuss our progress and get feedback on our development and plans for further development. Using these meetings as feedback sessions and having our product owner give confirmation or feedback on our features gave us confidence that our work met their expectations and was of the quality they wanted.

Unit testing: When testing the functionality of SafeQR, we conducted unit testing of its main features to ensure that our automation was providing the expected results. Unit testing is testing parts of the code to validate expected behavior (Huizinga & Kolawa, 2007, p. 75). In our case, the results were being tested to determine what data was derived from the QR codes.

We worked with the client to use realistic test scenarios with copies of real phishing emails. As many phishing emails now contain QR codes, it has been easy to find a lot of unique campaigns to work on. Our product owner within Netsecurity gave us these emails as he works as a security analyst.

Due to the risks with real phishing emails, we created synthetic test files that the group could use to test the code for bugs. These were used before pushing new code to our GitHub repository and testing with real phishing emails at significant updates. The synthetic EML files contained QR codes pointing to our website. The website is a link tracker that will notify one of the group members when the URL in a QR code is opened.

We can precisely track scan quality by giving each QR code in every file a unique link. Each link contains the name of the file it was found in and, optionally, a description of where it was inserted. E.g., *malicious_xlsx_in_cell* would be a QR code inside a cell in an Excel file.

We introduced unit testing during Sprint 2 to simplify quality assurance. This would involve uploading one of our EML files to the script and comparing the links found to a list of known

QR data. This QA procedure identified several issues in our project that would otherwise be overseen during manual checks. Therefore, this process was repeated in consequent sprints to ensure that the code performed as intended.

3.3.3 Risk Analysis

A risk analysis was conducted in the early stages of the project to identify and evaluate potential challenges. This analysis was revised throughout the project. To pinpoint what could negatively affect the project. The analysis used a risk matrix, as shown in Figure 5, to calculate the probability and severity of each risk, resulting in an overall risk rating.

Understanding the different risk factors of the project early on allowed us to create different mitigation strategies to manage and avoid potential problems. We also needed to start a discussion and reflect on what might hinder our progress.

	4 Highly Likely				
Probability	3 Likely				
	2 Unlikely				
	1 Highly Unlikely				
		1 Negligible	2 Minor	3 Moderate	4 Severe
		Severity			

Figure 5 - Risk Matrix

Two examples from our risk analysis included insufficient experience with Python and internal problems within the group. As seen in Figure 6, the concern regarding Python experience was mitigated with a two-week self-study of the coding language before the project started. In the final example regarding issues within the group, the measure of clear role responsibility, at least two weekly meetings, and finally, a feedback culture was established to mitigate the risk. The complete risk analysis can be found in Appendix 11.5.

1	Identified Risk	Probability	Severity	Risk Rating	Measures
2	Not enough experience with Python	2	2	Minor	Allocate two weeks of self-study before project-start.
3	Internal problems within the group	2	3	Moderate	Clear role and responsibility distribution. At least two weekly meetings with status updates. Encourage a feedback culture.

Figure 6 - Classification of risk, probability, severity, its rating, and measures taken

As stated, performing this analysis proved beneficial to the group from both a quality assurance perspective and a platform for reflection and discussion.

3.3.4 Benchmark Test

To ensure that our product improved or shortened the workflow of Netsecurity’s security analyst, we performed a benchmark test in the last stages of our project. The test was to handle a set incident in XSOAR with and without SafeQR. We performed three tests with three security analysts working at Netsecurity. They were available in the office, and we could observe their work process as the test was undertaken. The test was timed to check the time used to handle the incident, and feedback from the analysts was gathered after the test. The time started when the analysts opened the incident in XSOAR and stopped after identifying the QR code and its URL in the reported email.

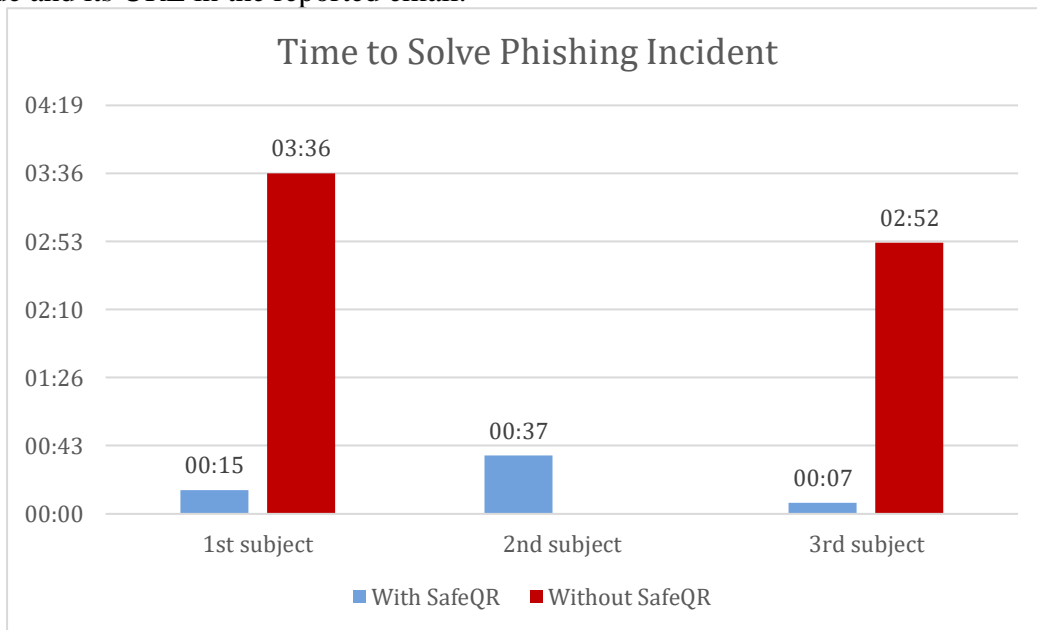


Figure 7 - Visualization of timed tests of SafeQR

The test proved to the group that our work had the potential to improve the analyst’s workflow significantly. The time used to handle the incident without SafeQR was considerably higher than with our script, as the analysts had to manually open the email in the incident and find an

online QR scanner to find the URL. With SafeQR, they only had to find the integration in the XSOAR dashboard. We will discuss this test further in the discussion chapter.

3.4 Platform and Technology

This section outlines the technological foundation of our project, detailing the diverse platforms and technologies utilized. Below is an overview of the technology stack, presenting their roles, reasons for selection, and configurations.

3.4.1 Python

Due to the integration requirements with Cortex XSOAR, we had to choose a programming language that met the platform's requirements. "While automations support JavaScript and PowerShell, the most comprehensive support is for Python-based development" (Palo Alto Networks, 2023a). Given that the client had also developed their scripts in Python, this became a natural choice. Additionally, Python's extensive selection of frameworks and libraries made it a good option for this project.

3.4.2 API (Application Programming Interface)

SafeQR utilizes third-party Python libraries to detect and decode QR codes. The libraries allow code written by other developers to be easily integrated into the project, adhering to the common software principle of not "reinventing the wheel" (Kansas State University Computer Science, n.d.). This approach avoids the inefficiency of replicating advanced, well-established functionalities. Attempting to replicate these ready-made functions would be time-consuming and counterproductive, so our applications incorporate these functions through libraries.

While these libraries are effective and necessary for our project, they also introduce potential security liabilities, such as supply-chain attacks. The considerable size of the code makes these libraries challenging to thoroughly review, increasing the risk of overlooking hidden functions. Netsecurity concluded that they could not accept this risk, which required us to reconsider the architecture of our application. Our solution involved dividing the application into two separate components. The functions for detecting and decoding QR codes were restructured to be hosted on a more isolated server environment. At the same time, the remaining components continued to run on the XSOAR platform as initially planned.

This change in architecture required the development of an API to enable data transfer between the two environments. An API allows applications to exchange data, functionality, and features (Goodwin, 2024).

We used FastAPI, a web API framework, which was a good choice. Its functionality met the project criteria, being fast and easy to implement. This change in architecture moved the security liabilities from the sensitive XSOAR platform to a more isolated environment.

The need to host our application in a separate environment was revoked after further discussions with Netsecurity revealed that the decision was based on a misunderstanding between our team and the client. Although we discarded this solution, we retained the code of this alternative application structure due to its enhanced porting capabilities, which could cater to various customers.

3.4.3 Cortex XSOAR

XSOAR is a SOAR platform developed by Palo Alto Networks (2023b, p. 1). Given its capability to integrate various technologies and tools, SafeQR has been implemented into XSOAR. This implementation was accomplished by creating a custom Docker image via the XSOAR command line interface, ensuring that all SafeQR prerequisites were installed. This also resulted in SafeQR being isolated from the XSOAR container.

Netsecurity manages two different XSOAR instances. One is a multi-tenant configuration designed for customer use, where each customer is provided with an individual tenant. This allows customers to manage their environments independently if they choose to do so without impacting other customers. Additionally, this setup enables security analysts to consolidate all cases related to a particular customer in one centralized location. The other instance is single-tenant, dedicated to development and testing purposes. Our group was granted access to this instance during our project to facilitate testing without impacting the production environment or customer data.

3.4.4 Microsoft 365

Netsecurity's phishing response service is specifically designed for Microsoft Outlook. It utilizes users' ability to report suspicious emails in Outlook, flagging them as potential phishing attacks. This reporting functionality in Outlook is only available for Microsoft 365 organizations with mailboxes in Exchange Online (chrisda et al., 2023).

The XSOAR tenant utilizes two integrations with Microsoft 365:

- Microsoft Graph, a RESTful web API (MSGraphDocsTeam et al., 2023), to retrieve the reported email messages in EML format.
- Content Search, a Microsoft security and compliance feature, to find and delete phishing emails.

After an email is reported as a potential phishing attempt, the EML file is fetched to the customer's XSOAR tenant, and the SafeQR script will be activated as part of a sequence of response actions. Simultaneously, the email is moved to the user's Deleted Items folder in Outlook, which it remains temporarily. A security analyst reviews the email, assisted by the script results, and decides whether to flag it as phishing or restore it to the user's inbox. If the security analyst flags it as a phishing attempt, the Content Search integration detects and deletes emails with similar attributes across all customer mailboxes.

The group set up a Microsoft 365 tenant to test our script and its integration with the XSOAR platform. By linking Netsecurity's development instance of XSOAR to this tenant, we could simulate the entire phishing detection and response process. This configuration provided high-quality testing opportunities and allowed us to thoroughly evaluate the whole process, from receiving a phishing email to the finished incident response.

4 Technical Considerations

4.1 QR Codes

Quick Response (QR) codes were designed in 1994 by Denso Wave, a Japanese automotive parts company, to streamline the tracking of automotive parts in the supply chain. This technology saw a significant development in 2002 when Sharp introduced a mobile phone capable of scanning QR codes, significantly expanding its application beyond industrial uses. (Sugiyama, 2021) QR codes are scanned through digital devices as they can be used to access websites, applications, and more (Unicode, n.d.). Although QR codes are not inherently interactive, users primarily derive their utility from the content they access through them.



Figure 8 - Typical QR code formatting

Since then, QR codes have seen widespread adoption in various sectors. Tesco's 2012 advertising campaign in South Korea allowed subway riders to conduct their grocery shopping by scanning QR codes on the station walls (Lee, 2012). The latest major use case for this was the EU digital COVID certificates, which were used to securely validate someone's vaccination status across country borders (European Council, n.d.). They are also used in day-to-day tasks, such as tickets for public transport and two-factor authentication. Authentication is based upon something you are, something you know, or something you have (Grassi et al., 2017, pp. 12–13). A personal QR code could configure an authenticator as something you have.

4.2 EML

EML, a file format closely tied to the Internet Message Format (Resnick, 2008), has become a widely accepted de-facto format for storing raw email data. The data in an EML file is structured according to RFC 5322 (IMF), the ASCII-based syntax required by RFC 5321 (SMTP) to transfer email messages (Klensin, 2008). By storing email data using the Internet Message Format, EML files allow for seamlessly structuring both headers and content and ensuring close compatibility with SMTP.

IMF looks a lot like the traffic sent through an HTTP response. The headers come first as a colon-separated key-value list. The main headers describe the sender, recipient, subject, date, and message ID. The receiving SMTP server may also add some headers for their reference, most commonly a spam score, sender IP address, and security-related information. The body

is formatted by creating boundaries, making it possible to separate different blocks in the email from each other. These blocks commonly contain plaintext, HTML, or attachments.

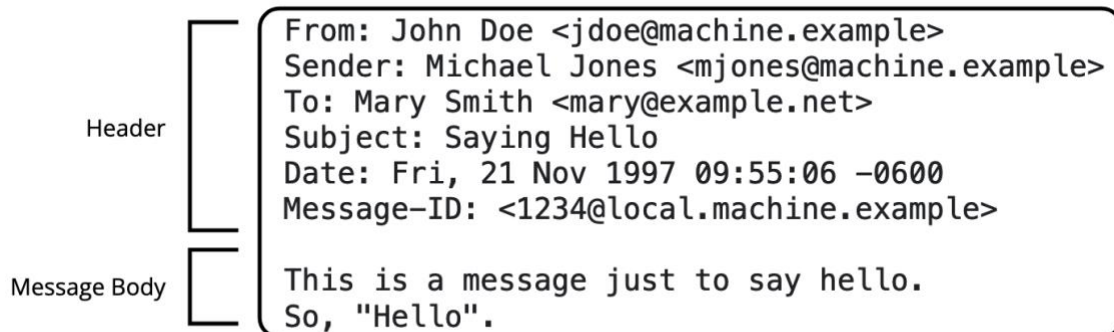


Figure 9 - Example of EML/IMF file, inspired by RFC 5322 (Resnick, 2008, p. 43)

4.3 XSOAR Automations, Integrations, and Playbooks

The security analysts' tools on Cortex XSOAR are generally split into three categories: Automations, Integrations, and Playbooks.

SafeQR is an example of automation. It is a script that will fetch data from an incident, process it, and execute tasks without user interaction. Automations work well for scanning files, querying IP reputation scores, and otherwise parsing data from the incident.

Integrations connect the XSOAR instance to other services. That is why XSOAR can automatically fetch newly reported spam emails from Microsoft 365. Integrations can also be downloaded from Palo Alto's marketplace, allowing the user to install tools further to orchestrate the collection of relevant data for the analysis.

Playbooks are batches of instructions that can be run manually or automatically on an incident. In the case of phishing emails, an integration will first fetch the email from Microsoft servers. When the incident has been created, a playbook will use an Outlook integration to email the customer about the receipt of their report. While that happens, the playbook will launch the SafeQR automation to prepare the data before the security analyst opens the incident dashboard. The security analyst will view the data collected by SafeQR, as shown in Figure 10. If the QR code contains a URL, the URL will be set as an indicator in XSOAR so that if there are other incidents with the same URL, the security analyst will be able to see that there has already been an incident and can use that to gain more knowledge to assess the new incident.

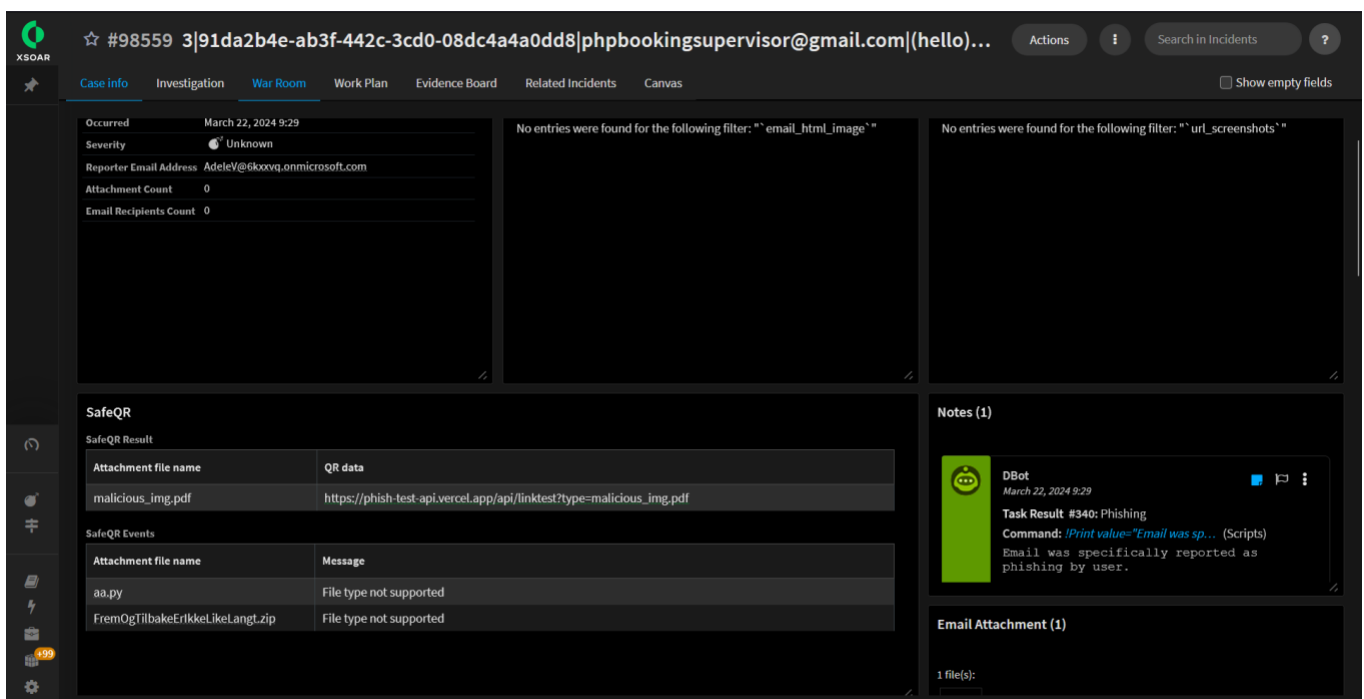


Figure 10 - XSOAR dashboard with SafeQR Integration

4.4 Security

4.4.1 Supply-Chain Risks

Naturally, Netsecurity is skeptical of third-party software and source code. Using someone else's product carries a risk of exploitation. Netsecurity cannot control another party's software and must analyze the product's safety.

A recent example of intentional exploitation through software was XZ Utils, a data compression library often used on Linux machines (Tukaani Project, n.d.). An open-source contributor, Jia Tan, added a backdoor to the tool, making it possible to run unauthorized commands remotely through SSH (Collin, n.d.; Freund, 2024).

Python is significantly exposed to supply chain attacks through typosquatting. This involves republishing packages under names similar to the originals but with common spelling errors. These counterfeit packages have malware attached (Taylor et al., 2020, pp. 112–113). When a programmer misspells the name of a package, a malicious package will be downloaded without their knowledge.

4.4.2 Docker Hardening

Docker was part of the initial scope of our project, as the plan was to create an API that would run on its own server. Docker hardening became part of the scope as we saw the potential for exploitation. This plan has since fallen out of our scope as our plans changed to direct deployment.

Cortex XSOAR handles all Docker administration under the hood, and automation developers cannot deploy their own Dockerfiles or, in other ways, manipulate the configuration. Docker hardening will need to happen on a deeper level, and it is not feasible for us to do it.

This does not mean that Docker hardening is unimportant. The server administrators need to ensure proper configuration directly on the server so that all Docker containers running XSOAR automations are isolated unless otherwise required.

5 Overview of Sprints

In total, we planned four sprints for the project. Each sprint was set up to tackle a different part of the SafeQR project, from pre-sprint planning to development, testing, and implementing the script into XSOAR. This way, we took things step by step, making it easier to handle challenges and give feedback to each other under the project. The following paragraphs will include information and insights into each of the completed sprints in the project.

5.1 Pre-Sprint

Before the first sprint, we had a phase called pre-sprint. This sprint was all about some groundwork to ensure a good start towards the project and the coming sprints. During the pre-sprint, we did the following:

Setting up the project: We ensured everything was in place for the project to start well, set up the development environments, and got the necessary tools ready.

Python Learning: Given that the script was to be written in Python, we dedicated some time for the team to learn the Python syntax and get familiar with the language. About two weeks

of the pre-sprint went towards this. This was important to ensure everyone could contribute to developing the Python script.

Introduction meetings with Netsecurity: We established a good relationship with Netsecurity in the pre-sprint. Here, we held introductory meetings to ensure we interpreted the task correctly. To ensure we had done that, we presented a flowchart of the script's functionality to our product owner at Netsecurity to confirm that we understood the task ahead of us. This flowchart is shown in Chapter 6, Figure 12. We also created a clear communication strategy during these meetings with the client. Here, we agreed upon meeting every Friday.

Creation of the Project backlog

The project backlog was also a working point in the pre-sprint. We gave the tasks a time estimate and prioritized them by their importance with the MoSCoW method. This gave the group a good planning phase for which tasks would be prioritized in the first sprint and how long they would take.

5.2 Sprint 1 (22.01.24 – 11.02.24)

Sprint 1 marked the start of the coding in the project, where we had goals of completing essential coding tasks. Sprint 1 was a critical phase for building a foundation for the project.

5.2.1 Planning Meeting

The sprint started with a planning meeting where the team gathered to review the project backlog created in the pre-sprint. In this meeting, we discussed the goals and objectives the group wanted to achieve in the sprint. These tasks were assigned to each group member based on their importance using the MoSCoW method. When selecting the coding assignments for this sprint, the group focused on using our resources on the tasks with the tag “Must have.” The assigned tasks from the backlog for Sprint 1 can be found in Appendix 11.2.

These tasks were then allocated to the team, forming two groups: one consisting of three members and another of two. The reason for working together in groups was to promote efficient collaboration in doing the tasks, sharing knowledge, and helping each other do the task.

5.2.2 Sprint 1 Retrospective

In this sprint, the team encountered several learning points and challenges that contributed to the project's development and the group's growth. One of the primary challenges we faced was miscommunication issues with Netsecurity. Initially, we had the assumption that we would be

allowed to use third-party libraries for the project. However, after two meetings with other individuals in Netsecurity, it became clear that we needed to reconsider our approach to using these libraries for security reasons. To tackle this obstacle, we decided to integrate an API on which these libraries would run, ensuring that the libraries were not directly downloaded on the XSOAR server.

On the positive side, the coding phase progressed much in this sprint. By the end of the sprint, we had successfully developed a Minimum Viable Product (MVP) for our script. This MVP processed an EML file containing QR codes located at a fixed path as a temporary solution, as XSOAR would later give us the exact path for the EML file.

In this sprint, the group also grew our understanding and usage of Python. The hands-on experience we gained from the project's initial coding phase made us comfortable with the programming language.

In this sprint, we had weekly meetings with the client. After the first sprint, the group agreed that it played a good role in the start of the project. The meetings with the client were a time when we could seek guidance and clarify any problems.

5.3 Sprint 2 (12.02.24 – 03.03.24)

In Sprint 2, the goal was to continue the coding phase of the project as well as tasks related to writing the bachelor report.

5.3.1 Planning Meeting

The planning meeting for Sprint 2 also started with looking into which backlog tasks would be done. Most tasks were coding, debugging, and adding missing functionality after creating the MVP in Sprint 1. The group also had some tasks regarding the report and presentations of the status of the project in this sprint. The planned backlog of Sprint 2 can be found in Appendix 11.2.

5.3.2 Sprint 2 Retrospective

During Sprint 2, we made substantial progress in developing SafeQR and implementing it within the XSOAR platform.

The group collaborated effectively during the sprint, actively taking on tasks and working efficiently. That said, this sprint also experienced delays due to waiting on the client and some bugs in the code. Some key learnings from the sprint included a better understanding of

containers, the functionalities of XSOAR, debugging, and integrating scripts into the XSOAR playbooks and displaying content onto the dashboard.

For future improvement in Sprint 3, the group sought to enhance follow-up processes by implementing more structured documentation of meeting outcomes. There was also a suggestion for in-person work sessions in smaller groups at UiA or the Netsecurity office to facilitate more effective collaboration.

5.4 Sprint 3 (04.03.24 – 24.03.24)

Sprint 3 focused on continuing development on SafeQR while slowly ramping up our focus on the bachelor report.

5.4.1 Planning Meeting

The planning meeting for Sprint 3, consistent with the previous sprints, began with a review of the backlog. The group concluded that the sprint would focus on fixing bugs within SafeQR and implementing the project into the development instance of XSOAR to test how the program was running inside the XSOAR dashboard. The group also agreed to start with the bachelor report. So, we separated the group into two teams. Three members would mainly focus on the code, and the other two would concentrate on the report and other small assignments.

5.4.2 Sprint 3 Retrospective

In Sprint 3, we finished the implementation of SafeQR into XSOAR. However, this proved more time-consuming than anticipated, as we had to remove FastAPI from the code after further meetings with Netsecurity. Therefore, we could not implement it as we had thought of it initially. This was combined with focusing on debugging different bugs in SafeQR that had come up within the development process. We also made significant progress on the bachelor report. While discussing the difficulties of the sprint, the group agreed that the lack of documentation on working with containers in XSOAR was an issue, as it made it harder to implement SafeQR into XSOAR. This also meant we had to rely more on the client, which took time as they were busy. Furthermore, another difficulty was that the group did not assign specific writing tasks to everyone, meaning that writing efficiency was not the best. We agreed to prioritize this in the next sprint.

5.5 Sprint 4 (25.03.24 – 15.04.24)

Sprint 4 focused mainly on the report and debugging of SafeQR.

5.5.1 Planning Meeting

When discussing this sprint, the group agreed that writing the report was the focus while debugging minor bugs in SafeQR. As the application now meets the client's criteria, there would not be any significant changes to SafeQR. However, the group had set a code freeze to April 19, so if there were any additional requests from the client, we could work on it in this sprint. As stated in the retrospective in Sprint 3, the group wanted to assign writing tasks. So, at the start of the sprint, this was done.

5.5.2 Sprint 4 Retrospective

In the fourth sprint of the project, the time mostly went to working on the bachelor report. As mentioned in the planning meeting, we were better at assigning specific writing tasks to the group members. This measure helped us be more effective as it was easier to understand what we would work on in the iteration.

5.6 Time after Sprints

After completing four sprints, our team stopped working in sprints. This decision was motivated by the need to dedicate our efforts to the bachelor's report. Despite moving away from sprints, we maintained our commitment to productivity and organization by using the Kanban board on GitHub. The Kanban board was good for assigning and managing writing tasks, leading to a more efficient workflow. Each team member was assigned writing tasks every week, with a deadline of one week for each task. This approach ensured that everyone was aware of their responsibilities in the report.

6 The Product – SafeQR

After our sprints, we demonstrated the finished product to Netsecurity, who expressed satisfaction with our solution. We will lay out the product further in this chapter.

Most of SafeQR's functionality is a background process, as we wanted its use to be easy and unobtrusive. What the analysts will see when utilizing SafeQR is a box in their XSOAR dashboard titled "SafeQR." The relevant information from the reported email for the incident will be displayed in this box. If the QR code is inline in the email, it will display the name of the picture and its URL. If the QR code is within a file in the attachments, SafeQR will display the name of the attachment, the name of the picture in the attachment, and the URL for the code. If the email contains multiple codes, all relevant information will be displayed. The

events table also gives data about the files that do not contain QR codes, such as “No QR codes found. Manual analysis is needed”.

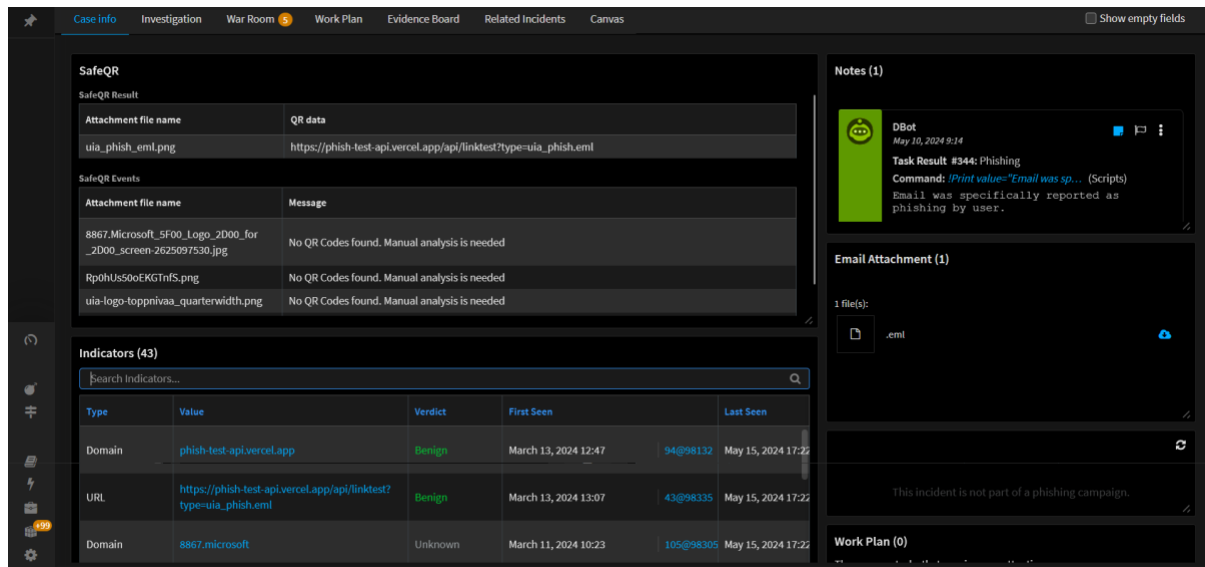


Figure 11 - Finished SafeQR in the XSOAR dashboard

The flowchart in Figure 12 illustrates how *quishing* incidents are managed when SafeQR is integrated into Cortex XSOAR, along with other playbooks. The background processes of the SafeQR script are detailed inside the grey dashed lines. Outside these lines, the flowchart depicts the workflows of end-users and security analysts, along with the roles of other

playbooks and processes in the broader incident response process, including email management and potential incident escalation.

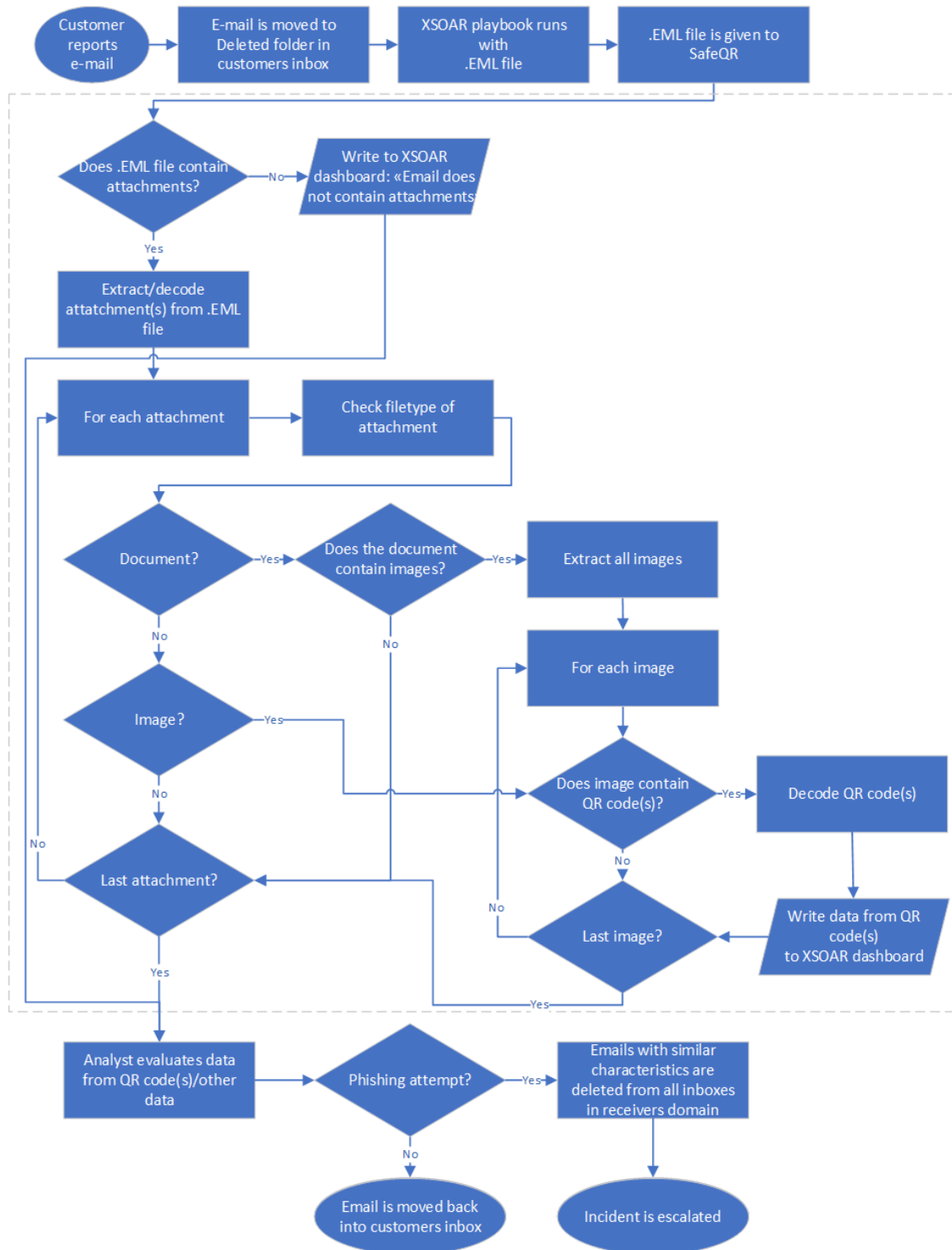


Figure 12 - Detailed flowchart of SafeQR

SafeQR runs as a playbook in XSOAR in conjunction with other playbooks related to phishing incidents. Netsecurity develops other playbooks used in the incident response processes and cannot be disclosed due to concerns about information privacy. Our playbook checks each

phishing incident created and runs SafeQR to extract the relevant information for the security analyst. Our product differentiates itself from similar scanners with its broad support of different file types. Our scanner accurately scans codes from all Microsoft Office files, PDFs, and image files. It also scans any codes in PDFs that contain JavaScript, which runs code to show the QR codes.

7 Lessons Learned

The group experienced many different aspects of development throughout the project. This chapter outlines the lessons learned from encountering these aspects.

7.1 Challenges Encountered

Challenges occurred as early as the pre-sprint of our project. The group had to learn Python to code our project. We all have coding experience and knowledge of other languages, but most of the group did not have experience with Python. Owing to our already existing knowledge, we expected we could learn the basics needed for the start of our project in the pre-sprint. The goal was to have enough knowledge to start work on the code for our script in Sprint 1 and then learn the language further along in the project.

Later in the project, we had to restructure our script to accommodate the security concerns Netsecurity had given us. As our application used third-party libraries, there was a particular risk in not knowing exactly how these libraries were structured. The group's focus from the start of the project was to develop a secure application that Netsecurity could use. After meetings with other employees, we were permitted to use third-party libraries, and using an API and server was considered redundant.

The second restructure lasted into our integration into XSOAR. The integration also proved to cause trouble for the group. Only one of our group members had any experience with XSOAR, so the group had to ask for guidance from Netsecurity. By having meetings with the employees of Netsecurity and trial and error in XSOAR, we managed to integrate our script.

7.2 Impact of Challenges

The consequences of the challenges we faced did not stagnate development in a way that affected the quality of our product. We experienced downtime developing some features as we had to wait for clarification from Netsecurity. Instead of halting the entire process of our project, we focused on other features while waiting for clarification so that we maintained our

progress. It was still a considerable effort to restructure our script twice and learn concepts that, in the end, did not factor into our product.

We spent the restructuring learning about the concepts we needed to implement, and as such, we did have a positive learning outcome when we successfully implemented the technologies needed.

7.3 Mitigation of Challenges

Mitigation efforts were conducted throughout our project to lessen the consequences of challenges.

Risk analysis: At the start of the project, we investigated potential problems and the risks they included that could occur during the project. We conducted the risk analysis and formulated some countermeasures for each risk (see Appendix 11.5).

Effective communication with Netsecurity: Effective communication with Netsecurity is also essential in handling problems. We were very active in keeping Netsecurity up to date with the issues we encountered. By doing this, Netsecurity could give us their thoughts, help us stay on track with the project, and lead us in taking the right approach to different problems. Examples are the meetings we had to ensure that we could use third-party libraries and the follow-up meetings conducted to reevaluate the use of API and a server.

Communication within the group: Effective communication within the group was also how we tackled challenges. During challenges, everyone could develop ideas to solve the problem, which would be discussed within the group. Where we then moved forward with the agreed-upon method. An example of this was the weekly meetings on Friday to summarize and plan further.

Help from our supervisor: We also received support from our supervisor, Tim, when tackling challenges. Tim was particularly helpful in giving us advice from a more academic and theoretical angle. He also gave us insights into communicating effectively with Netsecurity.

7.4 Cooperation with Client

A big part of the success of our project was working closely with Netsecurity. This partnership included them helping us use XSOAR and explaining different things to us at every step of the project. The group also worked hard to make sure the client was involved and understood what we were doing.

Right from the start of the project, we made sure to talk often with the client. We had meetings every Friday to share updates and present challenges. Even though the daily communication was good, misunderstandings occurred. This experience highlighted the importance of thoroughly understanding the discussed topic to ensure clear communication and avoid misunderstandings, especially when discussing complex cybersecurity topics.

7.5 Cooperation with Advisor

Having a good advisor was also crucial in our project. Our advisor, Tim A. Majchrzak, helped us both with academic guidance and was the person we could turn to when we faced any problems. Our relationship was a key to our project's success.

Our advisor provided us with academic advice throughout the project. We had biweekly meetings with our advisor, which allowed us time to make substantial progress on the report. This ensured that each meeting was beneficial and productive. Tim has a deep understanding of the academic field and guided us through complex parts of the work. Whenever we were stuck or unsure about something, Tim helped us in the right direction. Tim was also accommodating in writing the bachelor's project report, helping us with grammar and what to include.

8 Discussion

In this chapter, we will discuss and reflect on the project, the changes in technology, our group dynamic and development style, security aspects, further development, and our learning outcome.

8.1 Changes in Technology

Early in the project, when the group was determining the technologies required for the project, we focused on the technology required for development, such as IDEs, as these were the most essential to ensure we could deliver on the requirements given by Netsecurity. The group also agreed that our application would need to utilize third-party libraries in Python to make the application match the criteria from Netsecurity, as image processing and QR code reading are high-level code concepts that would take considerable time to implement manually. A standard software engineering practice is not to reinvent the wheel. The group agreed that we could not confidently promise Netsecurity that a manually coded solution could be finished within the timeframe given. The quality of such an application was also not guaranteed. In our experience, the knowledge required to write code that accurately and effectively scans images is acquired

from extensive research and proficiency. Third-party libraries that have been coded by experienced developers and are maintained regularly can accommodate this need. We checked the libraries we would use thoroughly to ensure they provided our needed function and were maintained regularly.

A meeting was arranged for us to discuss our use of the libraries, and we were then asked to remove or modify the structure to accommodate the security of the application and the potential implementation into XSOAR. The group decided that it could not 100% guarantee that the third-party libraries did not include code that communicated with some outside source or that an update to the library would not introduce this.

We scrambled to find a solution to how we could continue development using third-party libraries while still meeting the security standard given by the meeting.

The new structure of the application was that of an API and a dedicated server. After changing the application's structure and asking our supervisor if a server could be provided, we continued to finetune the application. The rework required some time, and we were waiting for acceptance and clarification on whether our new solution and structure were what they needed, which meant we were approaching the halfway point of our development time. We were also requested to “*containerize*” our application, which could put all of our application's dependencies, libraries, and configurations in a docker container (*Docker Overview, 2022*). We were told that containerizing our application would ease our integration into XSOAR. After this implementation, the group believed we had aligned with the requirements given in the meeting earlier on the project. The rework proved beneficial as we learned how our application would function and how it would be implemented into XSOAR.

We then moved on to work on the implementation in XSOAR, where we arranged meetings with other employees in Netsecurity who had more hands-on experience with implementation. At this meeting, we were asked why we had implemented our solution the way we did. They expressed that an API and dedicated server were not necessary. As XSOAR isolates its integrations, the resources needed for hosting an API would be unnecessarily high for our project and its intended use. This feedback surprised the group as considerable time and effort had been made to accommodate what we believed was the most optimal solution. We discussed handling this information in the group and with our product owner. We agreed to arrange a new meeting with the employees who have experience with integration and the ones in our first meeting on our third-party libraries. This meeting also proved integral to our development as

the API and dedicated server were reconsidered, and we did not have to include these technologies.

This process proved time-consuming, as we had to wait to implement central functions to accommodate new features not initially included in the requirements. The wait consisted of waiting for confirmation on what we could develop, awaiting answers from meeting invites, confirming which employees were available when scheduling meetings, and doing the extra meetings we did with our product owner.

After we removed the API and realized that XSOAR containerizes all its automation scripts, we could resume normal development and integrate it into XSOAR. After these exchanges, the group agreed that our communication at the project's start possibly convoluted the product's understanding. At that time, we did not know precisely how our application would work, how it would be integrated into XSOAR, and how XSOAR worked. The first meetings were arranged for us to ask questions about implementation and our development. However, the meetings soon evolved into requirement restructures and clarifications for all involved on how our script would function.

We learned a lot from the interactions with the different employees of Netsecurity, which showed that expectations and perceptions of an application can vary wildly for a group of people. It also proved that communication and clarification at the start of the project are vital to ensure stability throughout the development process and that meetings early on to get and give clarification can benefit projects greatly (*Project Communication--Foundation for Project Success*, n.d.). We developed the application successfully within the given time frame, and the group will take this experience into further projects. Each meeting we had with the employees at Netsecurity was insightful, and the wait for it to be arranged and held was never too long to hinder our project's development significantly.

8.2 Group Dynamic and Development Style

We strived to maintain a good group dynamic throughout the project. By working together physically at Netsecurity's offices often, we met each other every week. We could cooperate and discuss other matters to nurture a healthy group dynamic. Social gatherings were also arranged for the group to team-build and do different activities besides working together.

8.2.1 Methodology

The group has successfully developed other projects using Scrumban and considered this approach a fitting choice for this project.

Our earlier experience with Scrumban compelled us to make some adjustments to suit our group. We agreed that some aspects of the methodology produced considerable work documentation, like daily standups. Daily standups proved more of a hassle than productive meetings for the group, which we decided would be performed each Monday and Friday. The group agreed that the need to meet and plan daily could hinder our progression because we would focus more on the meeting and planning than on performing our tasks. We usually divided the group into smaller teams for different tasks on Mondays, and these groups would coordinate when to meet and how to do their tasks. When we met each Friday again, we would show our completed tasks and progress on other tasks undertaken between Monday and Friday.

As explained in our project management section, we divided tasks into categories based on estimating the time needed to complete them. These categories were small, medium, and large. Previously, in other projects, we estimated how many hours a task would take and documented how many hours it took to perform it when done. The group opted for a more categorical estimation, as keeping hours is time-consuming, and if one forgets or does not remember how many hours a task took, it quickly turns into guesswork.

The group decided early in the project that we would try this approach for three weeks to test if it suited the group's work ethic and schedule. The group felt increased productivity by focusing on planning for the week on Mondays and summarizing on Fridays. Work and self-study could be coordinated for the other days of the week, focusing on task aspects instead of meetings, planning, and documentation of redundant work processes. The group decided early in the project that we would try this approach for three weeks to test if it suited us. We discovered this approach complemented us well and continued it throughout the project.

8.2.2 Integration into XSOAR

Integration into XSOAR was a high-priority feature from the start of our project. Most of the group had no experience with XSOAR; similarly, we had no experience with integration into it. One of our group members worked at Netsecurity, had experience with the system, and was knowledgeable of its integrations, but a comprehensive understanding was required for our project. Our product owner assured us we would receive assistance and documentation for the integration. We also received laptops from Netsecurity to access the development instance of

XSOAR and test our application. Our product owner would gather our questions each week and forward these to developers at Netsecurity to provide us with the answers as quickly as possible. Unfortunately, the documentation for XSOAR from Palo Alto was lacking, and it often did not provide answers to basic questions or concepts. Again, we had to turn to the developers, who had extensive experience using the system and could give us walkthroughs and help with specific problems.

We felt a great sense of learning during the integration process. At the same time, our progress halted each time we encountered a new feature we wanted to use or did not understand how it worked. Pauses or slowing of progress are not uncommon when encountering unexpected hindrances. However, these were especially troublesome for the group because we perceived the documentation as a reliable source of help when it lacked answers. We did not want to be a nuisance to Netsecurity, our product owner, and the developers who graciously helped us. Still, they shared our frustration with the documentation and assured us their expertise was derived from extensive trial and error and experience using the system.

Implementation was a significant learning experience for the whole group as it happened late in the project and introduced new concepts and challenges. The group's experience was positive as our product owner, and the developers who helped us made themselves available and helped us when we got stuck. The implementation was also the subject of significant security aspects of our entire project, and some of these aspects made it so our project was only integrated into the development environment of XSOAR, which we will discuss next.

8.3 Security Aspects

Security is bound to be involved when collaborating with a cybersecurity company for a project. We encountered problems when implementing third-party libraries. The security concern was whether the code in these libraries communicated data from their usage to an outside source. This could include what systems they run on, where their servers are located, the networks associated with the system, and many more potential data leaks. We were upfront with Netsecurity that we would align with their security policies for this project so that we could achieve the best product that they could utilize in their daily operations. This is why we changed the structure of our application to meet these demands. However, when we learned more about the functions of our application and the implementation into XSOAR and spoke with the other employees, we realized these security demands would be met regardless.

The security aspects of projects within a cybersecurity company also touch on ethical aspects. Introducing new systems that positively promote better and faster functionality can often be a considerable risk for the company and its customers. If these new systems introduce new weaknesses in existing systems, the company can no longer guarantee that their systems are foolproof. For cybersecurity companies, it is critical that their systems are robust and that the trust they gain from their customers is not squandered. A single leak or system breach can spell the company's downfall and present its customers with many problems if sensitive operational data is accessed, leaked, or modified. So, the ethical aspect considers if it is worth the risk of introducing new systems if it can lead to the security company shutting down and a group of people losing their jobs because of a potential weakness in the new system.

As such, during our project development, it was deployed onto the development version of XSOAR so that we could test its functionality and keep their systems safe. Our product owner has expressed a desire for SafeQR to be implemented in the production version of XSOAR, but this will most likely require Netsecurity to overtake the development of SafeQR and perform a security analysis. In the next section, we will discuss the potential for further development.

8.4 Further Development

Our project, SafeQR, is considered a feature complete by the group and Netsecurity. Nonetheless, some specific functionalities remain to ensure its robustness in different usage scenarios.

One of the key features yet to be implemented is the support for decoding QR codes embedded in SVG format within PDF files. This functionality is not crucial for the project but would enhance SafeQR's versatility. The group decided with our product owner not to move forward with development on this functionality, as there was poor documentation on how to realize it, and after looking into real examples, embedding SVG files in PDF files is rarely seen.

In addition to minor functional enhancements, Netsecurity must examine some security aspects before SafeQR is put into production. Here, Netsecurity will have to undergo a security analysis to check whether there are some risks to SafeQR and if they are manageable. This type of analysis takes time and will not be completed within the project's timeframe.

The group performed a benchmark test of SafeQR by asking security analysts at Netsecurity to handle a set incident with and without SafeQR. As shown in the quality assurance chapter, the time saved by using SafeQR was considerable. We showed these results to our product owner,

who was surprised by the decreased time used. The analysts we tested also expressed that our solution would be a welcome addition to their workflow. As all the subjects had different approaches to handling the incident, we theorized if this was the case for all analysts. We asked the analysts if their handling of the incident followed some official guide from Netsecurity. Everyone answered “No”. As each analyst's process differed, the time used to handle incidents without SafeQr also fluctuated. *Quishing* is a somewhat new development in phishing methods. So, official guides formulated for or by cybersecurity firms are not yet commonplace.

The group believes that if SafeQr is implemented, the time to handle *quishing* incidents would be reduced and more uniform for all the analysts.

8.5 Learning Outcome

Throughout our bachelor’s project, we have achieved various learning outcomes that have significantly contributed to our professional and personal development.

Technical Proficiency: We deepened our understanding of various technical domains throughout our project. We learned much about QR codes, including their history and development, inherent risk, and how they can be detected and decoded using Python. Furthermore, our project expanded our knowledge of EML file formats and structural design using API coding in Python. Additionally, we gained experience with XSOAR, learning how to develop and integrate our tools into this platform to automate security workflows. These technical skills have contributed to our project’s success and prepared us for advanced problem-solving and development tasks in our future careers.

Project Management: Our bachelor project provided crucial insights into fundamental project management. We started the project by defining our goals and planning the necessary steps. This structured approach was essential in ensuring that every phase was well-coordinated and efficiently executed. By changing the previously used method to suit our group better, we felt a great sense of progress and interconnection with the project as we set the terms for our workflow. Throughout the project, we sharpened our time management skills, learning how to set realistic deadlines and prioritize tasks effectively.

Additionally, we developed our abilities in resource allocation. We learned to assign tasks among the team members, effectively utilizing everyone’s competence. This ensured the optimal use of resources and promoted a collaborative environment. Lastly, getting hands-on experience with these project management abilities has prepared us to undertake projects confidently in the future.

Working experience in a company: During our project at Netsecurity, we used the academic knowledge we had learned during the last few years in a professional setting. This experience gave us a deeper understanding of how a development process works within a security company. This, combined with learning about SOC and tools like XSOAR, not only enhanced our technical skills but also gave us valuable exposure to how it is to work within a company. Therefore, this was an excellent experience to bring into our work lives.

Cybersecurity: Throughout our bachelor's degree in Netsecurity, we also gained first-hand experience with actual cybersecurity incidents, deepening our knowledge of threats like phishing and *quishing*. We learned how they respond to these threats while at the company. Our exposure to actual security breaches and preventative measures were essential skills to learn to create our tool SafeQR. Furthermore, the cyber security skills learned at Netsecurity have prepared us for future roles in cybersecurity.

During our project, we gained a deep understanding of QR codes. Before the project, we viewed the codes as handy shortcuts for different tasks, but our views changed after learning about their malicious use. As *quishing* is a growing concern, our group also views the widespread use of QR codes as a similar concern. The main concern is that everyday use of QR codes normalizes scanning something without checking where the link leads. This could cause less awareness of the inherent danger of *quishing*. We formed some guidelines we will use forward to guard ourselves and hopefully others if they decide to follow them as well:

- Only scan codes from trusted sources
- If you can preview the link the code leads to and it is suspicious, do not interact with it.
- Be aware of offers with “too good to be true” deals.
- Do not let curiosity get the best of you. If a QR code appears to be the only way to access information, seek other ways.

8.6 Value Created for Netsecurity

In our work for this project, we have generated value for Netsecurity by streamlining their handling of *quishing* incidents. The time security analysts use to process these incidents has been reduced, allowing them to focus on more complex issues. Our project has also furthered the handling of email *quishing* incidents as it handles QR codes in attachments. The functionality of our script has the potential to save time and resources. As far as we know, no

products available have SafeQR's capabilities. This could give Netsecurity a competitive edge over its competitors by enabling them to offer a unique service.

9 Conclusion

In this report, we introduced our project client and explained the background of our project. We then detailed how we managed our project and the technical aspects we had to consider. We structured our work in sprints, so each sprint has been laid out and explained. After this, we described our finished product and the lessons we had learned from the process. Lastly, we discussed the whole project and its aspects.

During our project, we faced many different challenges. We adapted to a new programming language, acquired domain knowledge to facilitate our development, and faced challenges in communicating the functionality of our product and its implementation in XSOAR. All these aspects of our project gave the group immense learning outcomes in technological development, collaboration, and communication skills. Our exciting project kept the group vigilant in our work, learning, and collaboration with Netsecurity.

In retrospect, our communication with Netsecurity was initially confusing and lacked proper knowledge. A desire to appease the client quickly may have contributed to this, as we wanted to show the desired results quickly. The same desire drove us to accommodate their requests to change our project twice. We continually communicated and verified that we were on the right track, and we believe this was a significant factor in the success of our project.

To conclude, we have had an immensely positive experience working on our project with Netsecurity. It is natural for projects to encounter problems and obstacles. It is how you manage these aspects that make a difference. The product was completed within our given timeframe and met and even, in some respects, exceeded the requirements given by Netsecurity. Per the statement from Netsecurity in the Appendix 11.3, we achieved a level of satisfaction for our client and their expectations. We are proud to have developed this product together, including Netsecurity as our client. The entire project has been a positive learning experience for the whole group, and we will apply the skills we acquired and the knowledge we have gained in future projects.

10 Bibliography

Agile Business. (2014, January). *Chapter 10: MoSCoW Prioritisation*.

<https://www.agilebusiness.org/dsdm-project-framework/moscow-prioritisation.html>

Association for Project Management. (n.d.). *What is agile project management? What Is Agile Project Management? | APM Methodology & Definition*. Retrieved March 13, 2024, from <https://www.apm.org.uk/resources/find-a-resource/agile-project-management/>

chrisda, Dansimp, MithunRathinam, v-ratulach, v-stsavell, v-mathavale, dhairyya, beccarobins, & Ratulch. (2023, April 24). *Report phishing and suspicious emails in Outlook for admins*. <https://learn.microsoft.com/en-us/microsoft-365/security/office-365-security/submissions-outlook-report-messages?view=o365-worldwide>

Cloudflare. (n.d.). *What is quishing*. Retrieved March 13, 2024, from <https://www.cloudflare.com/learning/security/what-is-quishing/>

Cofense. (n.d.). *What is a Cybersecurity Response Playbook? What Is Cybersecurity Playbook? | Learn How Cofense Helps*. Retrieved April 8, 2024, from <https://cofense.com/knowledge-center/what-is-a-cyber-response-playbook/>

Collin, L. (n.d.). *XZ Utils backdoor*. Retrieved April 4, 2024, from <https://tukaani.org/xz-backdoor/>

Docker overview. (2022, November 17). Docker Documentation. <https://docs.docker.com/get-started/overview/>

European Council. (n.d.). *EU digital COVID certificate: How it works*. Retrieved March 13, 2024, from <https://web.archive.org/web/20240207080814/https://www.consilium.europa.eu/en/policies/coronavirus/eu-digital-covid-certificate/>

Freund, A. (2024, March 29). *Backdoor in upstream xz/liblzma leading to ssh server compromise* (oss-security). Openwall. <https://www.openwall.com/lists/oss-security/2024/03/29/4>




























- Gandhi, U. (2023, December 12). *Protect your organizations against QR code phishing with Defender for Office 365*. TECHCOMMUNITY.MICROSOFT.COM. <https://techcommunity.microsoft.com/t5/microsoft-defender-for-office/protect-your-organizations-against-qr-code-phishing-with/ba-p/4007041>
- Goodwin, M. (2024, April 9). *What is an API? What Is an Application Programming Interface (API)?* <https://www.ibm.com/topics/api>
- Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). Digital Identity Guidelines. In *Digital identity guidelines: Revision 3* (3rd ed.). U.S. Dept. of Commerce, National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-63-3>
- Griffiths, C. (2024, 05). *The Latest Phishing Statistics*. The Latest Phishing Statistics (Updated May 2024) | AAG IT Support. <https://aag-it.com/the-latest-phishing-statistics/>
- Help Net Security. (2023, October 30). Cyberattacks cause revenue losses in 42% of small businesses. *Help Net Security*. <https://www.helpnetsecurity.com/2023/10/30/small-business-data-security/>
- Hoxhunt. (2023). *Hoxhunt Challenge: Results*. <https://pages.hoxhunt.com/hubfs/Hoxhunt-challenge-2023-results.pdf>
- Huizinga, D., & Kolawa, A. (2007). *Automated defect prevention: Best practices in software management*. Wiley-Interscience ; IEEE Computer Society.
- IBM. (n.d.). *What is automation? What Is Automation?* | IBM. Retrieved April 8, 2024, from <https://www.ibm.com/topics/automation>
- Indeed Editorial Team. (2022, October 6). *EML File Extension: What Are .eml Files and How To Open Them*. Indeed Career Guide. <https://www.indeed.com/career-advice/career-development/eml>
- International, F. (2023, April 5). *5 reasons behind the increase in digital banking fraud*. Fraud.Com. <https://www.fraud.com/post/increase-in-digital-banking-fraud>

- Irwin, L. (2022, March 22). *5 Ways to Detect a Phishing Email: With Examples*. IT Governance UK Blog. <https://www.itgovernance.co.uk/blog/5-ways-to-detect-a-phishing-email>
- Kanban Tool. (n.d.). *What is Scrumban? What Is Scrumban?* | Kanban Tool. Retrieved April 9, 2024, from <https://kanbantool.com/kanban-guide/what-is-scrumban>
- Kansas State University Computer Science. (n.d.). *Don't Reinvent the Wheel: CC 410 Textbook*. Retrieved March 14, 2024, from <https://textbooks.cs.ksu.edu/cc410/ii-gui/14-libraries/07-dont-reinvent-wheel/index.html>
- Klensin, Dr. J. C. (2008). *Simple Mail Transfer Protocol (RFC5321)*. RFC Editor. <https://doi.org/10.17487/rfc5321>
- Lee, J. (2012, January 4). Tesco's cool QR code advertising campaign. *Vancouver Sun*. <https://vancouver.sun.com/news/staff-blogs/tescos-cool-qr-code-advertising-campaign>
- MSGraphDocsTeam, dkershaw10, alexbuckgit, FaithOmbongi, angelgolfer-ms, Linda-Editor, jasonjoh, adrianwells, DCtheGeek, rbaemmert, Lauragra, davidmu1, nokafor, rwiki77, OfficeGSX, & jthake. (2023, July 26). *Use the Microsoft Graph API*. Use the Microsoft Graph API - Microsoft Graph. <https://learn.microsoft.com/en-us/graph/use-the-api>
- Palo Alto Networks. (n.d.). *What Is a Security Operations Center (SOC)?* Retrieved April 25, 2024, from <https://www.paloaltonetworks.com/cyberpedia/what-is-a-soc>
- Palo Alto Networks. (2021). *Cortex XSOAR Datasheet*. <https://www.exclusive-networks.com/uk/wp-content/uploads/sites/28/2021/07/cortex-xsoar-DATASHEET.pdf>
- Palo Alto Networks. (2023a, August 31). *Cortex XSOAR Python Development Quick Start Guide* [Palo Alto Networks documentation portal]. <https://docs-cortex.paloaltonetworks.com/r/Cortex-XSOAR/6.x/Cortex-XSOAR-Python-Development-Quick-Start-Guide/Cortex-XSOAR-Platform-Overview>

- Palo Alto Networks. (2023b, October 10). *Cortex XSOAR*. Palo Alto Networks.
https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/datasheets/cortex-xsoar
- Project communication—Foundation for project success*. (n.d.). Retrieved May 15, 2024, from <https://www.pmi.org/learning/library/project-communication-foundation-project-success-7796>
- Rekouche, K. (2011). *Early Phishing*. <https://doi.org/10.48550/ARXIV.1106.4692>
- Resnick, P. (2008). *Internet Message Format (RFC5322)*. RFC Editor.
<https://doi.org/10.17487/rfc5322>
- Scrum.org. (n.d.). *What is Scrum?* Scrum.Org. Retrieved October 5, 2022, from <https://www.scrum.org/resources/what-is-scrum>
- Silver, H. (2023). Working from Home: Before and After the Pandemic. *Contexts (Berkeley, Calif.)*, 22(1), 66–70. <https://doi.org/10.1177/15365042221142839>
- Sugiyama, Y. (2021, November 10). From Japanese auto parts to ubiquity: A look at the history of QR codes. *The Mainichi*.
<https://mainichi.jp/english/articles/20211109/p2a/00m/0bu/024000c>
- Taylor, M., Vaidya, R., Davidson, D., De Carli, L., & Rastogi, V. (2020). Defending Against Package Typosquatting. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12570, 112–131.
- Tukaani Project. (n.d.). *tukaani-project/xz: XZ Utils* [GitHub Repository]. Retrieved March 28, 2024, from <https://web.archive.org/web/20240328130125/https://github.com/tukaani-project/xz>
- Uniqode. (n.d.). *What is a QR Code? Learn How do they Work and other Basics*. What Are QR Codes and How Do They Work? Retrieved April 8, 2024, from <https://www.uniqode.com/what-is-qr-code>

11 Appendix

11.1 Backlog MoSCoW – Analysis

MoSCoW - analysis	Priority
Extract all images from office files	 Must have
Return the MIME type of a file	 Must have
Create emails with QR codes for testing	 Must have
Delete attachments and images after returning the results	 Must have
Attachments with matching names	 Must have
Extract all attachments from the EML file	 Must have
Unnecessary folder creation	 Must have
Invalid response from API	 Must have
Attachments ignored by API	 Must have
Internal server error when trying to delete non-existent images	 Must have
Function for decoding QR codes	 Must have
FastAPI implementation	 Must have
Occasional parsing error from API	 Must have
Deal with images that are embedded in the email body	 Must have
Write the SafeQR output to the XSOAR dashboard	 Must have
Extract all images from a PDF	 Must have
Fetch .eml-file from XSOAR	 Must have
Change the file structure	 Must have
Deal with SVG files	 Must have
Dockerize the whole API	 Must have
SVG files are not extracted from PDF files	 Must have
Add dev-tenant to XSOAR	 Must have
More event messages	 Should have
Create README.md	 Should have
Unit testing of API	 Should have
Create realistic phishing EML for the demo	 Could have
Confusing file names from API	 Could have

11.2 Backlog – Sprint Overview

Tasks	Sprint
Attachments with matching names	Sprint 1
Create README.md	Sprint 1
Change the file structure	Sprint 1
Extract all attachments from the EML file	Sprint 1
Extract all images from office files	Sprint 1
Return the MIME type of a file	Sprint 1
Create emails with QR codes for testing	Sprint 1
Dockerize the whole API	Sprint 1
Unnecessary folder creation	Sprint 1
Extract all images from a PDF	Sprint 1
Create realistic phishing EML for the demo	Sprint 1
Deal with images that are embedded in the email body	Sprint 1
Function for decoding QR codes	Sprint 1
FastAPI implementation	Sprint 1
Internal server error when trying to delete non-existent images	Sprint 2
Confusing file names from API	Sprint 2
Invalid response from API	Sprint 2
Attachments ignored by API	Sprint 2
Occasional parsing error from API	Sprint 2
Unit testing of API	Sprint 2
Add dev-tenant to XSOAR	Sprint 2
SVG files are not extracted from PDF files	Sprint 2
Delete attachments and images after returning the results	Sprint 2
Deal with SVG files	Sprint 2
More event messages	Sprint 3
Write the SafeQR output to the XSOAR dashboard	Sprint 3
Fetch the EML file from XSOAR	Sprint 3

Netsecurity statement

In november 2023, Netsecurity and the group agreed to collaborate on the SafeQR project. Netsecurity held a selection process to figure out what group would be the best fit for this assignment. The assignment was initially set to include some bare minimum requirements (must-haves), along with some “nice-to-haves” features, and some tasks if the students would have needed more work. The group that was chosen for this project immediately suggested one or more creative ideas on how the different tasks could be solved. The group showed great interest in the project, and it was prominent that they had thought the assignment through, and even asked us some great critical questions about the project. This made them stand out from the rest of the groups.

Netsecurity asked the group to develop a tool that could identify QR-codes in one or more files of different formats, while also reporting back about the findings. This was the base requirement, but the group was also given more tasks if needed.

A weekly meeting with Netsecurity’s representative and the group was held each Friday in the office building. In these meetings, the students presented their work from the past week, and asked questions related to the project, which Netsecurity’s representative then would delegate to the viable resources within the organization.

Netsecurity has been nothing else but impressed with the group’s professionalism, ability to take responsibility of the project, and problem-solving abilities. Communication and questions between the group and Netsecurity’s representative worked well, while the communication within Netsecurity was a bit more challenging at times. However, the students worked around those difficulties, and progressed on other tasks while waiting for answers. This goes to show how well organized the group is, and the effectiveness of their work is.

The bachelor group finished a product that Netsecurity asked for, but they also included more functionality beyond what was requested. They were also able to integrate it with our platform in dev, which was beyond the requirement, since we knew it could be difficult for someone not working with this platform before. For now, Netsecurity has not decided whether, or when, it will be implemented in our production environment yet. This is because of the strict policies that needs to be fulfilled when implementing anything new to prod. We hope that we can utilize this project soon.

Best regards

Odd-Egil Solheim Egeland

11.4 Group Contract

Vi, gruppe nummer 17 (IS-304 / IS-305)

Som består av følgende medlemmer: Magnus Eugene Nymo, Siddharth Dushantha, Tobias Bo Hansen, Victor Johan Rolf og Eirik Bakke Hannestad

Skal jobbe sammen i emnene IS-304 og IS-305

Bakgrunn og motivasjon

Vi ønsker å lære om, og mestre fagenes læremål, slik de er oppgitt på Canvas og i studiehåndboka.

Samtidig som vi får erfaring i det å utvikle et prosjekt med en faktisk bedrift.

Bestemmelser om fremmøte og samarbeid

Jeg forplikter meg overfor gruppen min, til at jeg vil:

- Følge forelesninger, og delta i gruppearbeid, samlinger, møter og øvelser, be om/ta imot veiledning. Jeg forplikter meg også til å gjøre selvstudier nødvendig for prosjektene.
- Respektere kravet om minst 80% fremmøte og også oppgi grunn hvis jeg er forhindret. 30 min eller mer for seint regnes som ikke møtt/fravær.
- Overholde frister og levere oppgaver som publiseres på Canvas, innen fristene.
- Sørge for at arbeid fordeles jevnt mellom gruppemedlemmer, men samtidig utnytte hver enkelt students spesielle ferdigheter og bakgrunn.

Evt. konflikter rundt samarbeid og innsats søkes løst gjennom diskusjon i gruppen. Fører ikke dette frem, kontaktes lærer/studierådgiver/veileder, så snart som mulig, og senest før innleveringsfrister.

Andre bestemmelser

Gruppeleder er: Siddharth Dushantha. Til viseleder, som også har ansvar for eventuelle planleggingsverktøy er Magnus Eugene Nymo valgt.

Gruppeleder leder gruppemøtene og sørger for at alle er med i arbeidet. Er det noe som må diskuteres, skal gruppeleder styre diskusjonen, og sørger for at alle blir hørt. Etter en diskusjon av en lengde som passer til temaets viktighet, avveid mot behovet for å ta en beslutning og komme videre i prosjektet, tas det en omforent beslutning. Er det uenighet, foretas det en avstemming, der gruppeleder har dobbeltstemme ved stemmelikhet. Etter at beslutninger er fattet skal alle forholde seg lojalt til disse.

Gruppeleder er gruppens kontaktperson utad, og hovedansvarlig for fremdrift og for at frister for innleveringer og lignende overholdes. Gruppeleder er også hovedansvarlig for å holde kontakt med og arrangere møter med eksterne partnere.

Sekretær fører logg/kort møtereferat, om hva vi ble enige om, hva gjorde vi og hva gjør vi neste gang. Av referatet skal fremgå: fremmøte, forfall (meldt på forhånd, med grunn), fravær (ikke møtt/uten oppgitt grunn). Tobias velges som sekretær.

Gruppeleder eller viseleder har ansvar for å innkalle til møter.

Sted, dato: Kristiansand, 12.01.2024

Underskrifter

Viktor ROLF

Eirik Hannestad

Siddharth

Tobias Bo Hansen

Meg Mmo

Netsec IS-304 kontrakt

11.5 Risk Analysis

Identified Risk	Probability	Severity	Risk Rating	Measures
Not enough experience with Python	2	2	Minor	Allocate two weeks of self-study before the project starts.
Internal problems within the group	2	3	Moderate	Clear role and responsibility distribution. At least two weekly meetings with status updates. Encourage a feedback culture.
Poor communication with mentor or advisor	2	3	Moderate	Establish regular meetings. Adaptability in case meetings cannot be conducted as planned.
Tasks take longer time than planned	3	3	Moderate	Precise definitions of tasks. Break tasks into subtasks. Be realistic when estimating task size.
Tasks are not being performed with sufficient quality	3	3	Moderate	Two or more people on any work task. SCRUM meetings. Code reviews.
The code does not meet clients' requirements	3	3	Moderate	Good communication with the advisor and other stakeholders. Requirement specification
Problems with integrating our solution into the customer's framework	3	3	Moderate	Testing application in development environment. Good communication with the advisor.