Managing Security Issues in Software Containers: From Practitioners' Perspective

Maha Sroor[a], Rahul Mohanani[b], Ricardo Colomo-Palacios[c], Sandun Dasanayake[b], Tommi Mikkonen[a]

[a] *University of Jyväskylä, Mattilanniemi 2, Jyväskylä, 40100, Finland*
[b] *M3S, University of Oulu, Pentti Kaiterankatu 1, Oulu, 90570, Finland*
[c] *Technical University of Madrid, Calle Los Ciruelos, Boadilla del Monte, 28660, Madrid, Spain*

Abstract

Software development industries are increasingly adopting containers to enhance the scalability and flexibility of applications. Security in containerized projects is a critical challenge that can lead to data breaches and performance degradation, thereby directly affecting the reliability and operations of the container services. Despite the ongoing effort to manage the security issues in containerized projects in SE research, more investigations are needed to explore the human perspective of security management in containerized projects. This research aims to explore security management in containerized projects by exploring how SE practitioners manage the security issues in containerized projects. A clear understanding of security management in containerized projects will enable industries to develop robust security strategies that enhance software reliability and trust. To achieve this, we conducted two semi-structured interview studies to examine how practitioners approach security management. The first study focused on practitioners' perceptions of security challenges in containerized environments, where we interviewed 15 participants between December 2022 and October 2023. The second study explored how to address security issues, with 20 participants interviewed between October 2024 and December 2024. Data analysis reveals how SE practitioners address the various security challenges in containerized projects. Our analysis also identified the technical and non-technical enablers that can be utilized to enhance security in containerized projects. Overall, we propose a conceptual model that visualizes how practitioners manage security issues in containerized projects. We argue that our proposed model will guide practitioners in making informed decisions to plan, develop, and deploy secure container systems.

Keywords: Software engineering, software container, container security, security management, interviews

1. Introduction

In recent years, the reliance on digital services has significantly increased across multiple sectors, driving significant advancements in digital transformation. Software-intensive industries such as finance, healthcare, manufacturing, and retail have increasingly integrated digital technologies to enhance efficiency, security, and scalability [1], [2]. Similarly, many public and private organizations are increasingly adopting software applications to optimize operations and enhance customer experience [3]. This growing dependence on software applications and the incremental customer demand for new features has further added to the complexity of software applications, introducing new challenges in designing, developing, testing, and deploying safe and reliable software applications. To overcome this issue, containerization has emerged as a credible solution by improving flexibility, portability, and agility in software development and deployment. Containerization encapsulates applications and their dependencies into isolated environments, ensuring consistent service delivery across different infrastructures [4].

Despite many advantages over Virtual Machines (VMs), [5], software containers (mentioned as only 'containers' henceforth) introduce significant security concerns, such as faulty image, vulnerable configurations, unauthorized access, and data leakage [6]. These security concerns are considered the main barriers to a wider container adoption [7]. A thorough investigation of security issues in containers and their implications is critical for organizations because it would help them improve their security strategies and ensure the continuous delivery of software services [8]. Furthermore, investigating security practices in containerized environments can enhance trust and encourage broader adoption [8].

A review of existing software engineering (SE) literature on security concerns in container systems reveals a lack of a holistic view of security management in containerized projects. Many studies discuss security challenges in containers [9], [10], but do not detail the practical challenges that obstruct security in containerized projects. Although some studies propose frameworks to manage and improve security in container systems [11, 12, 13], the frameworks often remain impractical for real-world projects as they do not align with domain-specific requirements. Moreover, studies do not consider the non-technical factors that affect container deployments, such as human administration, communication, budget constraints, and customer demand [14]. These security frameworks do not incorporate the human perspective in the management of container systems, neglecting the significant role of humans in software container administration.

Software containers are not fully automated, and they need humans to administrate the technical aspects, such as configurations, monitoring, and version control. Usually, security management in container systems focuses on configuration and testing practices and often does not consider the human perspective [15, 16, 17, 18]. Including investigations into the human aspect in security management is crucial because it adds new dimensions to understanding containerized system security management strategies, such as planning, strategic decision-making, policy enforcement, and continuous adaptation to new threats, making container security management more effective in practice. A comprehensive understanding of security management in containerized projects will enable industries to develop strategies that enhance software reliability.

Building on our earlier work [14], which explored the risks, vulnerabilities, and their causes in software containers from the perspective of SE practitioners, this study provides a deeper investigation into how these practitioners understand and manage security in containerized projects. Specifically, it extends our previous findings by investigating both the strengths and limitations of current security management practices of containerized systems as perceived by practitioners. Our new findings complement the previous study by highlighting how SE practitioners actively address and mitigate critical security and vulnerability concerns in real-world containerized environments.

The main research question (RQ) guiding this study was developed by following the guidelines recommended by [19] and [20], and is as follows: **RQ:** ***How do software engineering practitioners manage security challenges in containerized projects?***

To answer our main research question, two critical aspects needed to be investigated. The first is to explore how practitioners perceive security issues in containerized systems in terms of their causes and implications. The second aspect is investigating how SE practitioners address these security issues in containerized projects. Since the focus of this study is on understanding the perspectives of SE practitioners on security management in containerized projects, we address the main RQ by conducting a semi-structured interview research approach. This research method facilitates the collection of rich and in-depth qualitative data [21], which is essential to explore how security challenges are addressed in ongoing containerized projects from the perspective of the human actors involved. In addition, a semi-structured interview-based study is a flexible approach that allows customisation of the questionnaire to align with the participants' expertise. Unlike other research methods that rely on the analysis of published research, such as secondary literature reviews (SMS and SLR), semi-structured interviews offer the flexibility to tailor follow-up questions based on the active responses of the participants. This flexibility makes this approach widely adopted in similar research contexts within the SE domain (e.g., [22], [23], [24], [25]).

This article makes the following novel contributions to the scientific body of knowledge on managing container security in containerized projects.

1. The research presents a comprehensive analysis of the perception of how security is managed in a containerized environment from the perspective of software practitioners;
2. We identify the key strengths and weaknesses of security management practices, highlighting real-world challenges encountered in containerized projects;
3. The findings also explore the fundamental (or *key*) enablers that can enhance security in containerized environments by combining the technical and non-technical factors influencing security management.
4. In addition to identifying how practitioners manage container security challenges, our study revealed notable differences in how security is perceived and approached across professional roles. By analyzing the perspectives of practitioners, managers, and academic researchers, we emphasize the need for context-sensitive security strategies that consider both technical implementation and broader organizational factors.
5. This article also contributes a conceptual model to guide SE practitioners in developing robust strategies for managing security in containerized projects.

The remainder of this paper is organized as follows. Section 2 presents a comprehensive overview of software containers and container security. Section 3 outlines the research design and the methodology used for empirical data collection. Section 4 details the procedures we followed to analyze and synthesize the data. Section 5 summarizes the key findings of the study, while Section 6 discusses their implications for software engineering practice and directions for future research. Finally, Section 7 concludes the paper by highlighting its main contributions.

2. Background

This section introduces the background of this research, giving an overview of software containers, security issues in container systems, and managing security in container systems.

2.1. Software Containers

Containers are lightweight executable software packages that encapsulate an application along with its dependencies, including libraries, configuration files, and binaries, ensuring consistent execution across different environments [26]. Unlike virtual machines (VMs) that run their operating system on top of the host system, containers share the kernel of the host operating system to improve resource utilization [27]. In addition, containers operate in isolated and controlled environments, which enhances the security and stability of software applications [28].

The development of software containers begins with creating a container image. Container images are built in-house or pulled from private registries managed by organizations or public registries [10] such as Docker Hub [1] or Amazon ECR Public Gallery [2]. After pulling the image, the hosting machine is configured to allocate system resources, including memory, CPU(s), and file access, to ensure optimal performance [29]. Additionally, network segmentation and configuration are required to enable secure communication between containers and external systems [30].

Deploying the software containers requires a hosting machine to execute the application in the container image. The container creates an instance of its image and operates in an isolated environment to [30]. Container orchestration tools like Kubernetes [3] manage the deployment, scaling, and

[1]https://hub.docker.com/
[2]https://gallery.ecr.aws
[3]https://kubernetes.io/

operation of containers in the cluster to provide the service [31]. The deployment of containers involves executing the application encapsulated within the container image on a hosting machine. The container instantiates an image and operates within an isolated environment to prevent conflicts with other applications [30]. Then, the orchestration tools such as Kubernetes automate deployment, scaling, and operational management, ensuring high availability and efficient resource allocation within a cluster [31].

Containers offer numerous advantages to software deployment [32]. One of the main advantages is their ability to provide portable environments across different stages of development, testing, and production [5]. Containers are lightweight and consume fewer resources compared to VMs. Containers are also scalable; they can be scaled up or down according to resource demand [33]. Additionally, containers support microservices architecture models, allowing developers to break down applications into smaller and manageable components [34].

There are many containerization technologies, such as LXC [4], Docker [5], Podman [6] and others. LXC is one of the early container technologies that are used to run multiple isolated Linux systems [28]. Docker is one of the most popular container technologies. It provides a simplified creation, deployment, and running of containers [35]. Podman is also a leading container technology. It is used in high-performance computing (HPC) environments as it offers a daemon-less container engine [36].

2.2. Security Issues in Software Containers

Security management in containers is considered one of the biggest challenges for this technology [6]. Considering container security challenges is crucial to improving container adoption and encouraging container migration [8], [9]. Consequently, there is a need to comprehend the container security issues and management as they affect usability, performance and service availability [10].

Delivering secure service by containerized applications requires embedding security as an element in the development and deployment of containers. This happens in multiple phases. The first phase is the container image, a lightweight and executable software package essential for running containers

[4]https://linuxcontainers.org/lxc/
[5]https://www.docker.com/
[6]https://podman.io/

[37]. The second phase is the preparation of the container host, including the infrastructure settings to provide the container with an isolated environment [29]. The third phase is intra-containers, which is a running form of the image and lightweight virtualization of the software [38]. The fourth phase is networking, which facilitates communication with external entities and establishes internal communication channels [39]. The fifth and last phase is Runtime, which is a tool used to manage and execute containers to deliver the service in production [40]. Security decisions within these phases are critical, as they frame the techniques and methods by which security should be implemented and configured.

Security issues often arise due to configuration flaws and inadequate security practices during container deployment [41]. In the container image phase, security threats primarily stem from malicious image sources and insufficient vulnerability scanning. Pulling container images from untrusted registries allows malicious codes to cause harm to the container system. Ignoring or insufficient image vulnerability scanning exposes sensitive credentials [42].

Security issues in the container host affect isolation and data protection in container systems [9]. It basically happens because of insecure configurations, inefficient resource isolation, and host-escalated access permissions. An insecure container host exposes the container system to buffer overflow, host exhaustion, unauthorized access, and data breaches [43], [44].

The main causes of security issues within containers are unauthorized access, misconfiguration, and weak isolation mechanisms [8]. These causes may lead to Denial-of-Service (DoS) attacks or complete system failures [45] [46]. Some containerized applications rely on dynamic architectures to manage workload scaling. If scaling is not properly managed, it may lead to host exhaustion [47].

Security issues in container networks and orchestration usually occur because of insecure external communication with other systems or insecure internal communication between the clusters. Misconfiguring the network or the orchestration settings leads to network and orchestration security issues. Misconfigurations can lead to a privileged network, allowing unrestricted capabilities to the network. Consequently, unauthorized access to the network means controlling the nodes (servers, routers, or devices) [48]. Another reason is the poor segregation of the container network, which might expose sensitive data or make it vulnerable to intercepted network traffic [48].

Security issues during runtime occur because of the development and production implementation in the same physical environment. It is risky behavior as it increases surface attacks [9]. Runtime misconfiguration also exposes the container's potential performance issues that affect stability and service delivery [49].

2.3. Addressing Security Concerns in Container Systems

Security management in container systems is crucial to protect container applications. One of the security management approaches is testing. Container testing is essential for ensuring the security, efficiency, and reliability of containerized applications. Container testing is the process of detecting anomalies that could disrupt the progress of containerized software development. Container testing encompasses analyzing container images, configurations, container communication, and pipelines [15], [50]. Implementing comprehensive testing protocols is essential to ensure the robustness and continuation of container functionality as well as maintain the integrity and reliability of applications deployed in containers [16].

Another approach to managing container security is implementing security practices. Security practices refer to enhancing the security of container systems through collective processes and techniques [17]. One of the primary best practices is to use trusted base images and regularly scan them for vulnerabilities [51], [52]. This helps in minimizing the risk of introducing security flaws into the container environment. Additionally, employing role-based access control (RBAC) ensures that only authorized users can have access [18]. Network segmentation and restricted network capabilities are crucial practices to limit communication between containers and avoid surface attacks [48].

Maintaining container systems involves continuous monitoring of container behavior for anomalies that could indicate a security breach. Regular checking of logs can protect container runtime against known vulnerabilities [53]. Effective secrets management is also critical to secure sensitive information. Secrets should not be accessible to all containers, and it is recommended to be stored in external volumes [54], [14]. Regular security audits and compliance checks help ensure container development and deployment follow the internal and legal policies to ensure users' data privacy [18, 55, 56].

Security in container systems faces significant issues in the development and deployment life-cycle. These challenges can potentially compromise the integrity and functionality of containerized applications. Security issues arise from faulty images, misconfigurations in the host machine, network settings, or container pipelines. Additionally, unauthorized access during runtime can further increase security risks, potentially leading to breaches or service disruptions. Therefore, effective security management is essential to maintain the reliability, availability, and integrity of containerized applications and their services. A review of SE literature on security management in container systems shows that container security primarily relies on security practices and rigorous testing. However, human administration can significantly influence container security, there is a lack of consideration of the human role in planning, decision-making, and strategy development of security management in container systems. Thus, there is a critical need to explore the human perspective in security management to enhance the effectiveness and adoption of security practices within containerized environments.

3. Study Design

This section outlines the study methodology, detailing the planning, data collection, data transcription process, and data analysis approach. In the following subsections, we describe the research questions and methodology employed for both studies in substantial detail, wherever necessary.

3.1. Research Questions

Improving security in containerized environments requires a thorough understanding of how security issues are conceptualized, managed, and implemented in software container projects. Exploring security management practices in containerized projects will also help to strengthen security strategies and inform decision-making in container security. Accordingly, this study aims to develop a clear and comprehensive understanding of security management in containerized projects.

To comprehensively understand and explore how practitioners manage security in containerized projects, it is necessary to explore how practitioners perceive security issues, their causes, and implications in container systems, and how they address these issues in containerized projects. Therefore, the main research question (RQ) is addressed by dividing it into two subresearch questions—RQ1 and RQ2, as follows:

RQ1*: How do practitioners perceive security issues in software containers?*
RQ2: *How do practitioners address security issues in containerized projects?*

Accordingly, we conducted two separate interview-based studies to address our main RQ. While *Study 1* was designed to answer RQ1, *Study 2* focused on addressing RQ2.

3.2. Research Approach

We conducted two separate semi-structured interview-based studies, adhering to the established guidelines for qualitative research outlined in [57]. The interview questionnaires for both studies were developed based on the best practices recommended in [58], [21], and [59], which provided a foundation to ensure integrity, flexibility, and consistency throughout the interview process. To improve the reliability of our methodology and ensure the rigor of our interview protocols, we followed the interview guidelines proposed by [58] and the Empirical Standards for interview studies published by ACM SIGSOFT [60]. For data analysis, we used the coding techniques described in [61] and conducted a *thematic analysis* by following the process recommended in [62].

3.3. Research Planning

For both studies, participants were recruited through the authors' LinkedIn [7] connections and research groups from other partnering universities. We also recruited participants from a consortium for an ongoing project on software containers—*Quantum Leap in Software Development (QLeap)* and through industries collaborating with the core research team. The interviews were conducted in English using the Microsoft Teams platform [8]. Before starting the interviews, participants were informed of the main objective of the research. The participants were then asked for their consent to record the interviews. Each interview lasted an average of 60 minutes. The replication package with the interview questionnaire for both studies—*Study 1* and *Study 2*, can be accessed at: `https://zenodo.org/records/15630352`.

***Study 1*:** The interviews were conducted between December 2022 and October 2023. The interview guide consisted of two separate sections. The first section of questions was intended to collect demographic data from the participants, such as the country of employment, role/position in the organization, and related responsibilities, years of experience working with software containers, and the domains in which the participants developed container applications. The second section comprised open-ended interview questions to explore container security issues, their causes, and relevant solutions in containerized projects.

[7]https://www.linkedin.com

[8]https://www.microsoft.com/en-us/microsoft-teams

***Study 2*:** The interviews were conducted between October 2024 and December 2024. For this study, demographic data were collected using an online survey to optimize the duration of the interview. It included questions about the participants' country of employment, job title, number of software container projects they had participated in, and the work domain. Whereas, the actual interview instrument consisted of open-ended questions on security practices, testing, logging, monitoring, and human communication. The interview guide was shared with the participants several days before the interviews, allowing them ample time to review the questions and reflect on their experiences. This approach was intended to support a more informed and meaningful discussion during the interview sessions.

3.4. Piloting the Interview Instrument

Two pilot interviews were conducted before both studies to clarify, refine, and improve the interview questionnaire. The primary objective here was to assess the clarity and relevance of the questions and to determine whether participants could understand the core of the questions. The pilot data was also analyzed to assess the quality and reliability of the data for the final analysis. However, data from the pilot interviews were not included in the final data analysis.

The feedback from the participants on *Study 1* interview guide was positive, thus validating the clarity of the questions. While the feedback from the participants on *Study 2* interview guide recommends improving the wording of two questions for clarity and precision. The first question—*How can logging and monitoring help container security?* was revised to—*In your opinion, how can logging and monitoring help manage container system security?* to encourage a more nuanced response regarding security management practices. Whereas the second question—*How can AI play a role in security practices to support container security?* was refined to—*How can AI be embedded in current practices to improve container security?* to emphasize the integration of AI within existing security workflows.

3.5. Participants Sampling

***Study 1*:** We recruited a total of 15 participants using convenience sampling [63], ensuring a diverse range of expertise in containerized software development and security. Participants held various roles in the software industry, including CEOs, security specialists, and software engineers, providing information from both technical and strategic perspectives. To ensure diversity in organizational culture, security practices, and regulatory environments, we recruited participants from Finland, India, Sri Lanka, and the Netherlands. The participants had SE industry experience ranging from three to twenty-eight years, with an average of 11 years, representing varying levels of seniority and expertise. Their experience working with containerized applications ranged from one to eight years, averaging approximately four years, ensuring a well-rounded perspective on container security challenges and best practices. The demographic diversity of the sample (as summarized in Table 1) is intended to improve the generalizability of the findings in different roles, domains of work, and organizational contexts within the software sector.

ID	**Country**	**Role**	**Container exp. (yrs)**	**Domain**
P1	Finland	Developer	5	Higher Education
P2	Finland	Senior SW Engineer	6	Gaming
P3	Finland	CTO	5	Web Applications
P4	Finland	Security Delivery Specialist	2	IoT
P5	Finland	Lead Architect	5	Healthcare
P6	Finland	Security Engineer	3	Elevators
P7	Finland	Team supervisor	6	Telecommunications
P8	Finland	Senior SW Engineer	4	E-commerce
P9	Sri Lanka	DevOps Engineer	1	Fintech
P10	Sri Lanka	CEO	6	Logistics
P11	India	CTO	8	Electronic Medical Record
P12	India	Cloud Architect	7	Telecommunications
P13	Finland	DevOps Engineer	3	Web Applications
P14	Netherlands	Testing Engineer	4	Healthcare
P15	Finland	Senior SW Architect	8	Healthcare

Table 1: Participants' Demographic Data (Study 1)

***Study 2*:** We sampled a total of 20 interviewees for the second study, aiming to capture a wide range of experiences and perspectives on the development and deployment of software containers. Participants held various roles, including project coordinator, software designer, software engineer, developer, tech lead, architecture engineer, team manager, researcher, postdoctoral researcher, and university professor. This mix ensured the inclusion of voices from both software engineering practice and research, allowing us to explore the topic from academic as well as industrial viewpoints. Participants were recruited from Finland, Spain, Sri Lanka, India, Colombia, Poland, and the Czech Republic, reflecting various organizational cultures and geographical contexts. Their experience with containers ranged from 1 to 10 years, and they had worked on between 1 and 20 projects, with an average of 3.5 years of experience and involvement in approximately 4 projects per participant. The diversity in roles, level of experience, and background (as detailed in Table 2) was intended to improve the generalizability and relevance of the findings in different domains within the SE domain.

ID	Country	Role	Container exp. (yrs)	Domain
I1	Finland	Researcher & Developer	2	Edge computing
I2	Finland	Project coordinator	2	Education
I3	Finland	Researcher	3	Software ecosystems
I4	Finland	PhD & Architecture Engg.	3	Cloud computing platform
I5	Finland	Project researcher	2	Academia
I6	Finland	Software designer	1	Web service
I7	Spain	Software Engg.	2	Order management
I8	Poland	PhD & Team Manager	5	Web service
I9	Spain	Backend Software Engg.	1	Healthcare
I10	Portugal	Application Security Consultant	4	Telecommunications
I11	Estonia	Developer	3	Logistics
I12	Czech Republic	Web Developer	2	Web Development
I13	Colombia	IT Project Manager & Professor	10	E-commerce
I14	Finland	Postdoctoral Researcher	8	E-commerce
I15	Spain	Associate Professor	5	Digital literacy
I16	Spain	Researcher & Developer	2	Bioinformatics
I17	India	Software Engg.	3	Web service
I18	Sri Lanka	Software Engg.	2	Manufacturing
I19	Sri Lanka	Tech Lead	3	Web service
I20	Sri Lanka	Software Engg.	1	Manufacturing

Table 2: Participants' Demographic Data (Study 2)

3.6. Data Transcription and Management

For both studies, the audio recordings were transcribed into text using the automated feature provided in *Microsoft Teams*. To maintain the accuracy and efficiency of the transcription process, the first author manually compared the audio files to the automated transcripts produced and addressed all mistranscripted words caused by accent variations and automated errors in interviews. Subsequently, the second author randomly chose five transcripts and compared them with the original recordings. We found no major discrepancies in the transcribed interviews, thus validating the accuracy of the transcription process. The author team then reviewed and validated the transcription process by randomly selecting five transcripts. No changes or revisions were made.

The transcribed files were completely anonymized for both studies to ensure that the transcripts could not be traced back to reveal the identities of the participants. For *Study 1*, transcripts were labeled using participant codes P1 to P15, while for *Study 2*, they were labeled I1 to I20.

4. Data Analysis

This section outlines the process we followed to analyze the qualitative data from both studies. We used the thematic analysis approach recommended by [62], which provided a systematic method to identify, analyze, and report patterns (themes) within the data.

4.1. Data Familiarization

The transcribed files from both studies were uploaded to "Atlas.ti' [9], which is known for its advanced coding capabilities that facilitate organizing, analyzing, and visualizing qualitative data. For both studies, the first author carefully read all interview transcripts, ensuring familiarity with the participants' responses and gaining a comprehensive understanding of the content. This process facilitated the initial recognition of the main ideas and potential themes of the data. The research team discussed the potential coding approaches/schema to maintain rigor and consistency in the coding process. The research team confirmed that the most relevant aspects of container security were adequately captured. However, in *Study 2*, the first and second authors did a second round of reading due to the large volume of data, to better understand the data and improve the initial coding ideas before discussing the coding schema with the research team.

4.2. Generating Codes

To maintain the rigor and scientific accuracy of the qualitative analysis, we followed a similar coding process for both studies. The first author systematically coded all transcripts using "Atlas.ti". The interviews were analyzed line by line using an inductive coding approach [61], ensuring the rigor and reproducibility of the data analysis. The first author assigned descriptive codes to each quote (henceforth known as 'data segment'). To ensure the accuracy of the coding process, the second author independently reviewed and validated the assigned codes, ensuring that the codes reflect the content of the data segments. Consequently, the entire author team reviewed random subsets of the data segments and related codes and validated

[9]https://atlasti.com

the coding process. No major refinements or changes were made. At the end of the coding process, *Study 1* resulted in 227 data segments. For *Study 2*, data saturation was reached after the sixteenth interview, indicating that no new codes emerged thereafter, with a total of 310 data segments.

4.3. Forming Themes and Categories

To increase the level of abstraction and to better understand the phenomena under investigation, the identified codes were grouped into *themes* in both studies. These themes were developed as a high-level conceptualization of multiple codes grouped to describe the significant aspects of practitioners' experiences with container security issues. The first author grouped the relevant codes into themes, after which the second author reviewed and refined the themes. The entire author team then confirmed the themes.

To develop an in-depth understanding of the findings, we grouped the themes into higher-level categories in both studies. Categories in this research context are higher-order themes that classify multifaceted themes based on commonalities to understand how container security issues are perceived and managed. To form these categories, the first and second authors collaboratively analyzed the themes to identify patterns of similarity and difference. The analysis resulted in categories that capture the broader dimensions of container security management.

4.4. Developing the Model

Building on the identified codes and themes, we developed an initial conceptual model (see Fig. 1) to represent the interconnections between key container security concerns from the perspective of practitioners. This model illustrates how various aspects of container security, including configuration practices, tool usage, organizational influences, and perceived risks, interact and influence one another in practice. The development process involved multiple iterations and collaborative refinement between the author team to ensure the coherence and representativeness of the empirical findings.

Whereas, Fig. 2 presents the refined and extended version of the model, which integrates themes from both *Study 1* and *Study 2*. This consolidated model offers a more comprehensive view of the interrelationships among critical aspects of addressing container security issues, as well as the key enablers for improving security management in containerized environments. Overall, the model provides a synthesized understanding of how practitioners approach, adapt to, and manage container security challenges in real-world projects. The two models are explained in detail in Section 6.

5. Findings

In this section, we report the findings of the analysis of the data collected from interviewing practitioners on the management of container security in containerized projects. Each theme, in the following subsections, is supported by providing at least one sample quote from the data to manage the space and maintain readability. Furthermore, a consolidated spreadsheet containing all categories, themes, codes, and supporting quotes for both studies is available as part of the replication package, which can be accessed here: `https://zenodo.org/records/15630352`.

5.1. Study 1—Practitioners' Perspective on Container Security Issues

To manage security issues in containerized projects, we must first understand and explore how practitioners perceive security issues in containers. This subsection focuses on the security patterns of container security issues and implications from practitioners in containerized projects. The identified data segments and themes are further categorized into patterns that support container security and patterns that require improvements. Table 3 provides a consolidated view of categories, themes, and relevant codes.

5.1.1. Strengths

Strengths in this research context refer to the approaches and practices that practitioners use to improve the security of containers throughout the system life cycle. Our analysis identified five *strengths* employed in containerized projects—*Container security is a chain of dependencies, Preferring automation, Common understanding of the security issues, Considering non-technical causes, and Reliance on tools.*

Container Security as a Chain of Dependencies

Practitioners comprehend container security as a life-cycle process, where each phase influences the security of subsequent stages. Consequently, security measures implemented at one stage directly affect the overall security posture of containerized applications. As one practitioner explained *"If one part of the container goes down, it might cause problems with the others*

Categories	**Themes**	**Codes**
Strengths	Container security is a chain of dependencies	Containers are integrated pieces
		Delayed security implications
	Preferring automation	Automation preference
		Human mistakes
	Common understanding of the security issues	Image issues
		Host issues
		Intra-Container issues
		Network issues
		Runtime
		Recurring security issues
	Considering non-technical causes	Technical causes
		Managerial causes
	Reliance on tools	Configuration management
		Code quality
		Monitoring
Weaknesses	(Only) Experience-based knowledge	Classic issues
		Well-known issues
		Unknown issues
	Uncertainty about improving security practices	Volume conflict
		Layered approach conflict
	Lack of standardisation and guidelines	Lack of standardisation
		Lack of guidelines
	Unclear resilience time	Undefined resilience time
		Average resilience time
	Container security is conditional	Conditional security
		Unconditional security

Table 3: Summary of Thematic Analysis (Study 1)

as well" (P14), underscoring the interconnected nature of container components. Securing containers requires diverse expertise in coding, cloud maintenance, and network security to protect the entire life-cycle. Practitioners emphasize expertise collaboration to clarify and plan interdependent configurations in container deployment. Many vulnerabilities remain undetected until deployment, often revealing issues through irregular system behavior in production, as one of the participants noted *"Whenever we pull images, we don't know if it is actually secure, and we will never know until something bad happens" (P2).* Therefore, collaboration among experts during the development and deployment phase plays a crucial role in ensuring a secure and stable production environment.

Preferring Automation

Automation in container systems aims to eliminate human involvement in managing, monitoring, and orchestrating containers. According to our findings, security issues often arise from poor configurations in the container development life-cycle. To mitigate risks, practitioners advocate automated solutions over manual configuration to maintain container security, as one practitioner pointed out *"If there are any security automated tools, they will be better than humans" (P13)*. While tasks like base image selection may require manual input, automation can enhance orchestration setup, CI/CD pipeline building, system monitoring, and testing, reducing human errors, as one of the participants explained *"Usually developers want to do tasks as fast and easy as possible, meaning insecure shortcuts in most cases" (P3)*.

Common Understanding of the Security Issues

Container security issues involve risks and vulnerabilities that can arise at any life-cycle phase. Risks can be attacks that exploit system weaknesses, while vulnerabilities arise from design flaws or misconfiguration. We noticed that practitioners have shared knowledge and a deep understanding of the major categories of risks and vulnerabilities in container systems, including image, host, intra-container, network, and runtime. Some of the example quotes that reflect the major categories of risks and vulnerabilities are *"Yeah, if you're using the outdated base image, there will be vulnerabilities that need to be fixed" (P1)* and *"The runtime security issues mostly happen near the dependencies, as I previously said" (P8)*. This knowledge helps reduce the likelihood of risk and potential vulnerability exploits. Additionally, practitioners had similar opinions about most recurring security issues. They agreed that misconfiguration issues are the most recurring and pose a significant threat to container security, as one of the participants said *"Ohh, the most recurring issues are definitely misconfigured containers" (P6)*.

Considering Non-Technical Causes

Practitioners were aware of technical triggers for container security issues, emphasizing how misconfiguration can lead to security breaches, as anticipated. One of the participants explained *"I think the main causes are lack of knowledge and tooling to scan containers, applications, and codes " (P4)*, explaining an example of the technical triggers for security issues. Surprisingly, some practitioners have pointed out other non-technical factors, such as inadequate team communication and poor organizational and project management, as one of the participants noted *"Of course, issues should be fixed as fast as possible and communicated. But always, there are issues with company policies and project management on how they tackle the issue" (P14)*. Practitioners believe that non-technical factors can contribute to security vulnerabilities. They believe that management challenges pose risks as well as technical challenges. While development and deployment issues can be addressed once identified, problems such as team miscommunication or balancing the technology stack with project requirements within the constraints of the customer's budget are more complex, as one of the participants explained *"Clients don't care about official images. They need the minimum resources and the best output and security" (P8).*

Reliance on Tools

Practitioners employ many tools across various phases of the container life cycle. Practitioners utilize various open-source, licensed, and proprietary inhouse tools. However, practitioners have a heavy reliance on tools, many of them are not satisfied with the current tool plans. For instance, one participant mentioned that *"We are planning to use Red Hat Advanced Cluster Security for Kubernetes, it was a matter of discussion in our last meeting." (P10)*, presenting the future plans on security tools. Some practitioners also presented their companies' future plans to improve security tooling strategies to elevate security levels. Moreover, practitioners emphasized the caution of human administration in tool management. While tools perform their designated function, the results depend on the understanding of the tools' capabilities and their proper implementation and administration, as one practitioner cautioned *"Tools are not smart enough to detect new issues, we need to manage them" (P9).*

The tools employed are used for purposes including code quality, identifying vulnerabilities in container images, and managing dynamic and static scanning of container systems. Tools serve to mitigate vulnerabilities during both the building and deployment phases. Additionally, practitioners utilize tools to manage infrastructure configuration and define infrastructure through declarative configuration files. Monitoring tools also track system metrics and behavior and visualize system data on the front end—e.g., *"Grafana and Prometheus are used for monitoring system metrics" (P9).*

5.1.2. Weaknesses

Weaknesses in this research context refer to existing approaches used to manage container security and expose the system to potential risks. Our analysis identified five *weaknesses* employed in containerized projects: *(Only) Experience-based knowledge, Uncertainty about improving security practices,* *Lack of standardisation and guidelines, Unclear resilience time, and Container security is conditional.*

Only Experience-Based Knowledge

Experience-based knowledge in this study context refers to practical knowledge gained through exposure to the complexities and challenges in containerized projects. The analysis of interview data highlights a significant variation in practitioners' comprehension of container security issues. Practitioners' knowledge is shaped by individual experiences and the specific demands of their respective fields, not by academic or educational background.

Interestingly, professional discussions about the security of containerized applications were quite different, even within the same domain. Some interviewees assumed that containers were secure by default, and their evidence was that they had not personally encountered a security issue—e.g., *"Normally we don't have any issues, I can't recall any serious security issues" (P15).* Other respondents assumed container security was challenging based on the multiple incidents they faced, which sometimes remain unresolved. Quotes such as, *"We did know how this problem happened, our team deployed the containers as usual, but it just happened" (P10)*, and *"For example, I have gone through a problem about how to pass secrets inside the container. If you do it incorrectly, you can expose them via environment variables" (P4)*, which supports this idea. Most discussions about container security discuss the technical aspect, and faulty configuration was frequently mentioned as a key security risk, similar to this opinion *"There are lots of places where information can be exposed and you can get access to a system or find out about how it's running" (P7)*, while few prioritize security concerns specific to their domain.

Uncertainty about Improving Security Practices

Security-improving practices aim to enhance overall system security without addressing specific issues, unlike mitigation techniques that focus on resolving particular problems. Common practices include selecting secure images, controlling authentication, monitoring network traffic, and managing container development and deployment.

Upon deep analysis of the application of security practices in the container development life cycle, we noticed a conflict in understanding some security practice outcomes in container systems. An example is using container volumes, which are external storage, to save sensitive files. Practitioners supporting this practice claim it is important to store sensitive services away from containers to avoid the implications of unauthorized access, as one practitioner explained *"I would keep the container stateless and use volumes to reduce risks" (P12).* In contrast, practitioners against it claim that using volumes increases the risk exposure on the threat tree, as another practitioner explained *"Let's say you mount the root partition of a Linux machine, you mount it to the container. The volume mount creates a direct link between the host's file system and the container, so whatever is mounted becomes accessible inside the container" (P2).*

Lack of Standardization and Guidelines

Practitioners complained about the lack of documents describing the best practices and systematic protocols for container deployment. They explained their complaint that the available security guidelines are general and do not consider the container infrastructure in terms of sharing resources and the dynamic nature of containers, as one participant explained *"We do not apply container security guidelines in our company, but we have a kind of generic guidelines" (P12).* Moreover, the security tools and orchestration platforms' best practices and tools are rarely available. Practitioners also emphasized the need for inter-organisational standards for implementation and deployment processes, automating CICD pipelines, disaster recovery plans, and security policies, as one participant noted *"Standardization is not really used for now, but like it's an ongoing process" (P5).*

Unclear Resilience Time

Resilience time refers to the duration required to address container security issues. Many practitioners noted that it is difficult to specify a fixed time frame for resolving such issues. It depends on the nature of the security issue and the required experience, as one of our participants noted *"Resilience time depends on the product's nature and the customer. It could be anything from one or two days to one or two years" (P7).* However, some practitioners estimated the acceptable time frame 4 hours to one day. The inability to determine a precise resilience time impacts the security and stability of the system.

Container Security is Conditional

Practitioners deeply believe that containers can be secure enough to support software deployment, as one participant explained *"Containers have software delivery mechanisms, they are secure enough" (P7).* At the same time, they put conditions in place to ensure security, like embedding security as an initial element of the development and maintaining good human administration for the container system, as one of the participants noted *"think if all the considerations and risk points are taken containers can be secure" (P2).*

To address RQ1—*How do practitioners perceive security issues in software containers?*—we analyzed the responses of practitioners based on how they perceive security issues in containerized projects. Our findings indicate that SE practitioners perceive container security as a dependent decision throughout the container life cycle. Performing effective security requires properly implemented configurations, automation, security practices, and system integration. SE practitioners emphasize that both technical and organizational factors are essential, as understanding security issues is based on direct exposure to real-world security challenges. While SE practitioners demonstrate strong awareness of common categories of vulnerability, they also highlight concerns about poor team communication, the absence of clear guidelines, limited standardization, and uncertain resilience expectations. SE practitioners do not consider containers inherently insecure; rather, they emphasize that effective security depends on proper planning and the use of suitable tools, automation, and consistent communication throughout the container life-cycle.

5.2. Study 2—Addressing the Concerns in Container Security

Effective container security management necessitates the identification of key enablers that enable practitioners to address security issues. Recognizing these enablers provides a foundation for strengthening security measures and ensuring ongoing enhancements in containerized environments. This section reports the critical enablers for addressing container security issues, categorizing them into technical and non-technical factors to provide a comprehensive perspective on security advancements in containerized projects. All the categories, themes, and codes are summarized in Table 4.

Categories	**Theme**	**Codes**
Technical Enablers	Risk Identification	Ways to identify risks in container systems
		Main challenges in risk identification
		Security practices support risk identification
	Container Testing	Testing types
		Challenges in container testing
		Testing can improve security practices
	Security Practices	Strategic security practices
		Proactive security practices
		Security practices support security
	Logging and Monitoring	Role of logging and monitoring
		Logging and monitoring affect container security
		Logs are not fully reliable
		Logging and monitoring guide improvements in container security
	Artificial Intelligence	AI helps in knowledge sharing
		AI assists humans to achieve security
		AI helps in automation
		AI helps in testing and analysis
		Limitations and concerns of AI
Non-Technical Enablers	Sharing Knowledge	Improving the knowledge about automation in containers
		Improving the knowledge about tools and best practices
		Improving the standards and guidelines
		Common shared knowledge about container issues
	Human Collaboration and Communications	Importance of human collaboration
		Maintaining human collaboration

Table 4: Summary of Thematic Analysis (Study 2)

5.2.1. Technical Enablers

Technical enablers are technology-related factors that support and address security issues in containerized projects. The analysis identified five technical enablers: risk identification, testing, logging and monitoring, security practices, and AI.

Risk Identification

Risk identification refers to recognizing and addressing potential vulnerabilities and threats that affect container systems, as a practitioner explained *"Risk identification involves looking for updated packages and base images, then referring to public CVEs" (I17).* Risk identification can be achieved by detecting abnormal system behavior through tools or human monitoring, or by combining both. Effective risk identification in software containers involves continuous updates, anomaly detection strategies, and tooling plans, according to another participant *"We use scanners to detect the risks in the* *images and dependencies, so we can make quick updates" (I14).*

Risk identification in container systems faces many challenges. These challenges arise from the complexity of container systems design and the combination of static and dynamic elements operating together. One of the main challenges in risk identification is the continuous need for updated security tools and the need for effective security tool administration to address both known and unknown anomalies. Container system complexity is also another challenge in container systems. One participant highlighted this issue *"One of the biggest challenges I've faced is the complexity of the dynamic nature of containers. Containers are designed to spin up and down quickly, which makes it hard to keep track of what's running and whether it's vulnerable" (I19).* The integration between containers, hosting machine, orchestration platforms, and user inputs makes it hard to trace the security issues in the threat tree.

Container Testing

Container testing is essential for ensuring containers' security and performance. Although testing increases the workload on development teams, it is crucial to identify and mitigate security risks before they can affect the entire system. Container systems require different types of testing to ensure a secure performance for container systems, such as unit testing, integration testing, stress testing, and end-to-end testing, as one of the participants explained *"Stress testing is crucial. If you create an API, you should ensure it works as expected and doesn't give out unwanted information" (I4).* Unit testing involves testing that individual components are functioning properly on their own. Integration testing focuses on verifying the integration of container components. Stress testing checks the performance stability under a heavy workload. End-to-end testing ensures that the application will perform as expected in a production environment.

Container testing faces many challenges in container systems. One of the challenges that most industries are facing is that developers are taking responsibility for testing container applications they are developing, instead of a separate testing team, due to budget constraints. This places additional load on the developers, as they must test the functionality in addition to security without a clear knowledge of the security metrics. Another challenge is the difficulty of managing testing within a large number of containers on the same host, making it hard to test and audit the container dependencies effectively. One participant described testing as *"Testing itself grows very complex in this dynamic environment, especially the testing setup" (I6).*

Testing results of containerized applications offer valuable insights for enhancing security. Testing results help to provide statistics on recurring security threats in container systems. The identified threats from the testing process should be thoroughly examined to determine effective mitigation and recovery strategies. These results from testing processes should be considered and translated into practical security improvements in container systems, as noted by one of the previous participants *"Testing results give you confidence that everything works as intended if combined with security practices" (I6).*

Security Practices

Security practices in container systems are a comprehensive set of planned actions and strategies that aim to protect applications, infrastructure, and data in a container environment. Security practices are supposed to be proactive and continuous to mitigate threats in their early phases. Proactive security practices must cover various levels, including image, code, application, infrastructure, ports, nodes, and user inputs, as one practitioner described *"Security practices involve maintaining and managing systems to avoid vulnerabilities and exploits" (I17).*

In addition to proactive security practices, strategic security practices are essential for ensuring the protection and integrity of container systems. It must include an integrated set of procedures, tools, and strategies to protect the container system. Strategic security practices ensure that security is embedded into the development process and aligns with DevSecOps principles for maximum protection. Strategic security practices should be tailored to the specific needs of each project to ensure security is effectively integrated into the development life-cycle, as explained by one participant *"Security practices are strategies, tools, processes, and various things like that" (I6).*

Following proactive and strategic security practices provides a structured approach to managing security in container systems. It provides a plan for implementing security in container systems that can be tailored according to the customer's needs and available budget. Moreover, it schedules defined time for regular auditing and vulnerability assessments that minimize threat exposure, as one participant emphasized *"Security practices help manage container security by applying measures such as image scanning, enforcing least privilege access, controlling network traffic, and ensuring proper config-*

uration of container orchestration platforms" (I16).

Logging and Monitoring

Logging and monitoring are continuous processes of collecting and analyzing data about the system's behavior, including errors, activities, resource consumption and performance. Logging and monitoring support container security in many ways. It helps to identify security issues and track their origin source, whether it is a user, system element, or container application. It provides a real-time overview of the running system to evaluate threats before they extend to other system elements. Moreover, it alerts the security team about users' failed logging attempts and the credentials that are exposed in log files, as one participant noted *"Practices like centralized logging, log persistence, policy enforcement, and monitoring ensure a certain level of security in containers" (I3).*

Unauthorized access is one of the major risks that affect the reliability, integrity and confidentiality of the container logs. Therefore, logging and monitoring security practices such as access control and encryption are essential to mitigate unauthorized access. Ensuring the reliability of the logs' data requires continuous updates to the container systems' dependencies, security practices, and security tools, as another participant informed *"Logging and monitoring system is really important to prevent unauthorized access" (I10).*

Artificial Intelligence

Artificial intelligence (AI) plays an important role in addressing security issues in container systems. It can be embedded in various security aspects of container systems. One of the main aspects of AI being a key player in container security is testing. Tools like Docker Scout [10] and Trivy [11] are using AI for unit testing in addition to their original function as vulnerability scanners. AI helps runtime security by detecting abnormal behavior in container logs, as one participant described, *"AI helps to provide the project status, understanding dependencies, and identifying main issues that need to be resolved" (I4).*

AI can strongly support security management in container systems. AI

[10]https://docs.docker.com/scout/
[11]https://trivy.dev/latest/

can automate security practices to avoid human mistakes, for example, AI can automate YAML files— a human-readable data serialization format used for configuration files— and image code modification according to the security guidelines. AI can also be used to monitor, assess risk, and check logs to detect patterns and anomalies. Some AI tools like CrowdStrike [12] help support anomaly decisions by prioritizing vulnerabilities in container systems. AI is also used to enforce security policies; for example, Aqua Security [13] is used to enforce container runtime security policies and block unauthorized processes in container systems, as informed by one of the participants *"AI enhances container security by automating tasks such as vulnerability scanning, anomaly detection, threat intelligence" (I16).*

AI helps improve communication and knowledge sharing among the development team. It can improve project collaboration and communication by summarizing meetings, tracking progress, transcribing discussions and creating project documentation. AI can also help at the foundational level for new employees' onboarding by mimicking a trial-and-error environment and giving guidance when needed to enhance learning and implementation, as one participant explained *"There could be some kind of tool that gathers discussions and forms a list of requirements of what is needed and what has been discussed" (I7).*

Despite the significant benefits AI can introduce to container security, it is associated with serious concerns. One of the main concerns is that developers use AI heavily in code generation, which introduces the potential to generate malicious code or expose sensitive data. Another concern is that the effectiveness of AI security solutions depends on the expertise of their users. If the user lacks experience, the outputs of the AI security solutions can be misinterpreted or improperly applied, leading to security vulnerabilities, as warned by one of the participants *"There could be numerous risks that AI poses that our current security measures are not able to address or identify" (I5).* Hence, AI should be used as a tool under the supervision of experienced professionals to ensure the reliability of security solutions.

[12]https://www.crowdstrike.com/platform/cloud-security/

[13]https://www.aquasec.com/

5.2.2. Non-technical Enablers

The non-technical enablers are the human-based enablers that help to address security issues in containerized projects. The data analysis provided two main non-technical enablers for container security: knowledge sharing and human collaboration and communication.

Sharing Knowledge about Container Security

Enhancing knowledge sharing in container security requires a deeper understanding of the challenges that practitioners face in container security. One of the challenges that requires more knowledge sharing in container systems is tools and their best practices. Improving knowledge about tools and their best practices needs trusted and comprehensive resources. While courses are usually recommended to learn more about tools and their best practices, they can help with foundational knowledge. For more advanced knowledge, practitioners recommend reading project documentation that provides valuable knowledge on tools and their best practices, as one participant informed *"It would be useful to know the characteristics of tools, such as the effort required to integrate them and their benefits" (I1).*

Another concern that requires increased knowledge sharing is the secure implementation of automation in container systems. Practitioners recommended a balanced approach to applying secure automation in container systems, where routine tasks can be automated, combined with human administration and monitoring. Furthermore, practitioners emphasize the importance of sharing knowledge about automation, such as downloading libraries, managing caches, and setting up initial infrastructure to reduce manual verification, as one participant noted *"Better knowledge on container security management automation is needed. Utilizing AI and predictive scaling could be future improvements" (I6).*

An alternative approach to enhancing shared knowledge about vulnerabilities in container systems is utilizing open-source vulnerability databases. Vulnerability databases such as CVE [14] and Snyk [15] help to identify, track, and mitigate vulnerabilities within container systems. In addition to vulnerability databases, there are other sources of knowledge about vulnerabilities, such as workshops, webinars, committee meetups, and recorded videos, as

[14]https://www.cvedetails.com/
[15]https://security.snyk.io/

one participant explained *"There are many databases for container vulnerabilities, but not everyone uses them" (I1).*

Another effective way to share knowledge about container security is through training programs, where professionals share personal experiences, lessons learned, and insights into how these experiences have influenced their approaches to implementing security in containers. Practitioners believe that these valuable insights should not be confined to training programs only. Instead, they should be shared more broadly through collaborative platforms, blogs, and documented use cases to make knowledge accessible to a wider audience, as one of the participants suggested *"We have lots of training about network security and general security practices, but not at the container security level. It would be nice to have such training as well" (I14).*

Human Collaboration and Communication

Effective human communication and collaboration are essential to successfully implementing and managing container security issues. Collaboration among teams, including developers, network engineers, cloud experts, and hardware specialists, ensures that all elements of the container system work cohesively. Clear communication regarding the implementation and configuration of each phase in the container life-cycle helps teams avoid potential configuration inconsistencies and mitigate potential security risks, as explained by one participant *"Good communication is essential to know the project progress, so when working in a team, we do not have to see every line of code in the pull request" (I18).*

Maintaining human communication and collaboration requires regular team meetings as well as accessible communication channels. Regular meetings, whether daily scrums or weekly sync-ups, help keep everyone informed about ongoing tasks and issues. Accessible communication channels are also essential for individual discussions, as one of the participants emphasized *"All practices involve human capital, whether it's monitoring, logging, deploying, or fixing attacks. That is why it has to be maintained" (I5).* Industries use various communication channels, such as Slack [16], Microsoft Teams, and WhatsApp groups [17], to ensure that all teams are updated on urgent matters. Overall. human collaboration should be actively encouraged to ensure that

[16]https://slack.com

[17]https://www.whatsapp.com

all team members are involved and contribute towards a secure container system implementation.

To answer RQ2—*How do practitioners address security issues in containerized projects?*—we analyzed the responses from participants on how to address security issues in containerized projects. Our findings reveal that practitioners utilize both technical and non-technical enablers to address security challenges in containerized systems. From a technical perspective, they highlight the critical role of security practices, comprehensive testing, continuous logging and monitoring, and the integration of AI to support automation, anomaly detection, and risk analysis. These enablers are fundamental for identifying vulnerabilities and ensuring the security of container system components. On the nontechnical side, enablers such as knowledge sharing, effective team communication, and collaboration are viewed as essential complements to technical enablers. Practitioners consider these nontechnical enablers vital for maintaining security throughout the container life cycle. Overall, the findings suggest that addressing security issues in container systems depends on the integration of both technical and non-technical enablers in the implementation life-cycle of containers.

5.3. Diverse Stakeholder Perspectives on Container Security Management

Although the primary objective of this study was to investigate how practitioners manage security challenges in containerized projects, we also observed notable variations in how different respondents perceived and approached container security management. To better understand these differences, we classified participants according to their professional background. Consequently, we present the perspectives of three distinct groups—*technical practitioners*, *managers*, and *academic researchers*, each offering unique insights into the security management in containerized projects.

The *technical practitioners* in this study are those who are directly involved in the design, implementation, and deployment of containerized applications. Their roles include developer, senior software engineer, and security delivery specialist. Building on the previous discussion, this group emphasized a hands-on, implementation-focused perspective on container security management. They view security management primarily as a matter of addressing practical challenges, such as configuration errors, weak isolation mechanisms, and network access controls, that can impact the availability and stability of containerized systems. *technical practitioners* generally expressed confidence that container security is achievable with the right practices in place. This includes proper configuration, careful integration of components, and the use of specialized security tools to detect potential vulnerabilities. Overall, they demonstrated a high level of trust in containers as a secure and reliable technology for software deployment, as one participant noted—*"It's all about how you use containers. If you have all the security practices in place, there are no issues with container security in general" (P12).* Similarly, another participant noted *"Containers are secure, without much effort. You need to set the right configuration, then you need tools for monitoring containers and expertise to understand what is and isn't supposed to be happening between containers" (I1).*

The *managers* group in the context of this analysis comprises participants responsible for the planning, coordination, and strategic decision-making in containerized software projects. Their roles include CTO, team supervisor, and CEO. In contrast to *Technical Practitioners*, *managers* perceive container security as a multifaceted and inherently challenging task. While they recognize the importance of technical safeguards, they place equal—if not greater—emphasis on non-technical factors that influence security outcomes. These non-technical considerations include budget-driven trade-offs, customer demands, and organizational priorities that may lead to security being given lower priority in favor of speed or cost-efficiency. For example, *managers* highlighted how pressures for rapid deployment often compromise secure development practices. From their perspective, effective security management in containerized environments depends not only on technical controls but also on adequate resource allocation, policy enforcement, and alignment with broader organizational goals.

Overall, *managers* conveyed a more cautious and sometimes skeptical view of container security management. Despite implementing best practices and using security tools, they acknowledged persistent limitations and residual risks. As one participant reflected: *"Security is not perfect actually because we encountered issues even with all those security tools"(P7)*, and another participant shared a similar opinion *"The security landscape is evolving a lot, and people don't spend time looking for new risks. In most scenarios, we don't know about the threat until it happens" (I2).*

Whereas, the *academic researchers* group in this study consists of participants engaged in the systematic study, analysis, and exploration of container security. Their roles include researcher, postdoctoral researcher, and project researcher. Compared to *technical practitioners* and *managers*, *academic researchers* adopt a broader and more forward—looking perspective on container security management—one that integrates technical innovation with organizational and socio-technical considerations.

They emphasized the importance of developing robust security practices alongside fostering organizational collaboration, knowledge sharing, and the creation of practical, evidence-based guidelines. In particular, *academic researchers* highlighted the need to investigate underexplored aspects of container security, including the emerging and increasingly prevalent use of AI in the development and operation of containerized systems. Although not dismissive of AI, *academic researchers* expressed a cautious position. They viewed AI as a double-edged sword that is capable of accelerating development and assisting practitioners in identifying vulnerabilities but also posing new risks if adopted uncritically or without sufficient human oversight. Their concerns centered on the potential for unknown vulnerabilities and over-reliance on AI-driven tools, which may not yet be mature or fully transparent.

Overall, *academic researchers* expressed a degree of skepticism about the future trajectory of container security in the age of AI, underscoring the need for continued research, critical reflection, and cross-disciplinary collaboration to ensure trustworthy and resilient containerized systems. This is evident from the following quotes—*"I'm not sure. There could be numerous risks that AI poses that our current security measures are not able to address or identify" (I14)* and *"AI can produce new issues like data leakage. This wasn't previously an issue, but AI can produce metadata out of your system, which can be used to reverse engineer your system in the worst instances" (I3).*

In summary, we investigated various perspectives on security management in software containers from three key stakeholder groups: *Technical Practitioners*, *Managers*, and *Academic Researchers*. The findings reveal that all groups shared a common understanding of the importance of technical practices–such as applying security controls, using appropriate tools, ensuring proper configuration, and conducting thorough testing – as an essential aspect for managing container security. However, the scope of concern varied across groups. *Technical Practitioners* primarily focused on addressing technical challenges within the container life-cycle. Whereas, the *Managers* emphasized broader organizational concerns, including resource allocation, policy enforcement, and achieving organizational goals. Finally, *Academic Researchers* were concerned about the unknown vulnerabilities associated with integrating AI into container systems.

6. Discussion

This research explores security management in containerized projects, examining two primary aspects: practitioners' perceptions of container security issues and the key enablers for addressing security issues in containerized projects. The findings emphasize technical and non-technical patterns in addition to technical and non-technical enablers. Integrating technical patterns and enablers with non-technical factors provides a holistic approach to managing container security. This integration ensures a more comprehensive and sustainable security strategy, fostering proactive threat mitigation and continuous improvement in security practices.

A thorough analysis of our findings reveals that *technical practitioners*, *managers*, and *academic researchers* largely agree on the expected security challenges in containerized projects. All groups recognize and confirm the importance of technical enablers such as anomaly detection, AI, security practices, testing, logging, and monitoring in managing container security. Furthermore, there is agreement on the importance of non-technical enablers, such as improving container security knowledge and fostering human collaboration within containerized projects. However, notable differences were observed in the primary concerns of each group regarding the management of container security in containerized projects. While *technical practitioners* prioritized the practical implementation of security, *managers* emphasized strategic organizational alignment involving resource allocation, budgeting, and policy enforcement, and *academic researchers* adopted a more cautious stance toward the integration of AI in container development and the application of security practices.

The data analysis reveals that better container security management requires increased collaboration from leading industries. These industries must maintain transparency by sharing internal standards and guidelines for managing containerized projects. There is also a need for more collaboration across industries to standardize processes for employee training and develop security guidelines suitable for general applications. Additionally, industries must engage more actively with regulatory authorities to ensure that future container security advancements align with data protection requirements.

Although improving the knowledge about container security is a personal responsibility for the practitioners in the first place, we think that it is also the responsibility of industries interested in containers. Individuals should proactively seek to attend relevant courses and webinars, and read research articles and blogs on container security. Industries also have responsibilities towards improving employees' knowledge about container security by providing fundamental and advanced training, proper onboarding, and supporting resources and tools. Moreover, organizations interested in containers establish alliances between industries and research institutes to deliver improved processes, standards, workshops, platforms, and YouTube channels to provide free knowledge about container security.

Fig. 1 illustrates our initial model that describes the interconnections among the themes on how practitioners perceive the issues and challenges in container security. Container security faces several challenges, including uncertainties in enhancing security practices, time limitations for resolving vulnerabilities, and the lack of standardized guidelines. Automating security processes and implementing security tools are crucial in strengthening container security by reducing human errors. Additionally, establishing a shared understanding of security issues is essential, as it provides deeper insights into their root causes and impact on container systems, thereby facilitating effective risk management. Practitioners also acknowledge the role of non-technical factors, such as efficient project management, in aligning security measures with the existing technology stack. Moreover, the model also emphasizes the influence of project-based experience on developing security expertise, reflected in variations in the time required to address security issues and differing perspectives on enhancing security practices.

By establishing the relationships between these themes, we can focus on the strengths and weaknesses of container security practices. Strengths include a comprehensive understanding of security issues, reliance on tools and automation, awareness of security dependencies, and consideration of non-technical factors. Conversely, weaknesses encompass the lack of systematic knowledge, guidelines, and standards, uncertainties regarding practice improvements, and resilience time. Integrating the key enablers for container

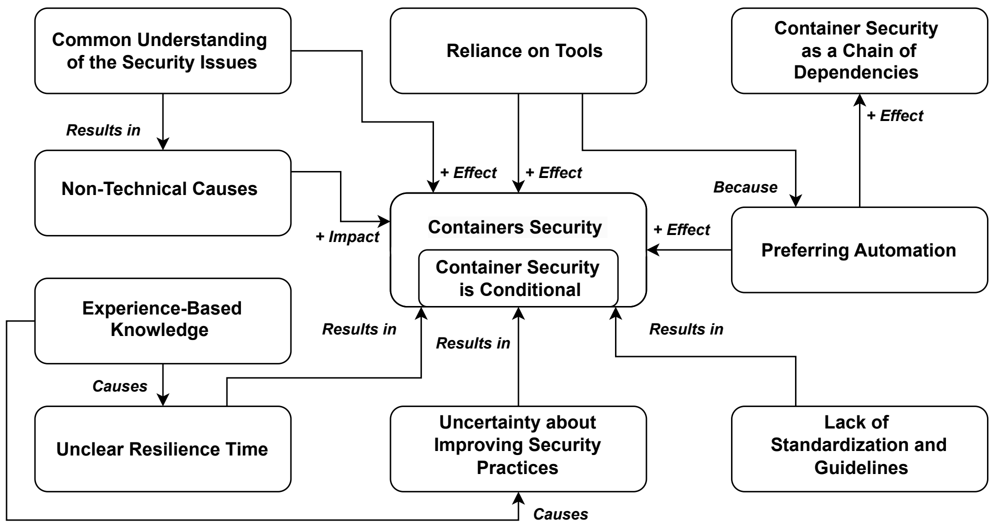
Common Understanding of the Security Issues
Reliance on Tools
Container Security as a Chain of Dependencies
Results in
+ Effect
Non-Technical Causes
+ Effect
+ Effect
Because
Containers Security
+ Effect
+ Impact
Preferring Automation
Container Security is Conditional
Experience-Based Knowledge
Results in
Results in
Results in
Causes
Unclear Resilience Time
Uncertainty about Improving Security Practices
Lack of Standardization and Guidelines
Causes

Figure 1: Model 1—Container Security Pattern Interrelation Model

security with the security patterns that emerged from the first study will provide a more comprehensive understanding of how security can be managed in containerized projects.

Thus, we developed our revised model (as shown in Fig. 2), which provides an overview of how practitioners manage security in containerized projects. The model further addresses the essential aspects and improvements, and relevant patterns (as seen in circular boxes) as follows:

Risk Identification supports container security management by identifying potential risks in container systems during the early phases of deployment. Analyzing risk identification techniques and their impact on the system will help estimate the time required to resolve security issues.

Container Testing reduces uncertainty in security practices by providing empirical evidence of their effectiveness when a system successfully passes various security tests, such as unit testing, integration testing, stress testing, and end-to-end testing. Additionally, testing helps determine resilience time by accurately evaluating the time required to detect, analyze, and mitigate vulnerabilities in the testing process.

Security Practices supports the container supply chain in container systems by safeguarding the chain of dependencies throughout the container development and deployment phases. Moreover, applying security practices

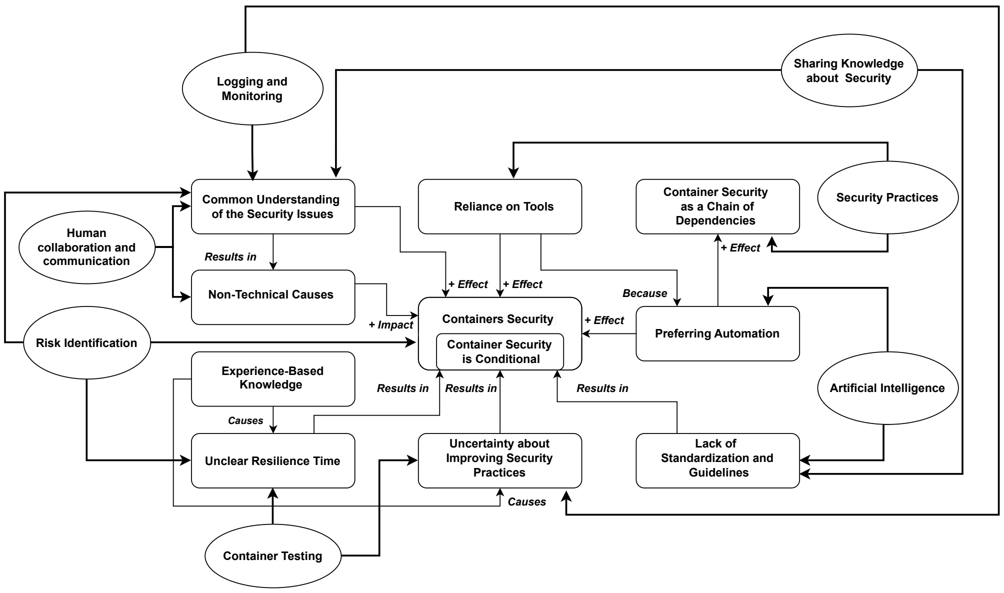
Logging and Monitoring
Sharing Knowledge about Security
Common Understanding of the Security Issues
Reliance on Tools
Container Security as a Chain of Dependencies
Security Practices
Human collaboration and communication
Results in
+ Effect
Non-Technical Causes
+ Effect
+ Effect
Because
+ Impact
Containers Security
Container Security is Conditional
+ Effect
Preferring Automation
Risk Identification
Experience-Based Knowledge
Results in
Results in
Results in
Artificial Intelligence
Causes
Unclear Resilience Time
Uncertainty about Improving Security Practices
Lack of Standardization and Guidelines
Causes
Container Testing

Figure 2: Model 2—Managing Security in Containerized Projects
Legend: Rectangles denote security patterns, while ellipses denote the key enablers

can significantly improve the performance of security tools, as it will substantially reduce false positive alerts in security reports.

Logging and Monitoring provides a detailed record of security events. Sharing these records among team members fosters a common understanding of security issues and the necessary precautions. Furthermore, continuous logging and monitoring help mitigate uncertainties regarding security practices by offering real-time insights into potential vulnerabilities.

Integrating *Artificial Intelligence* into container system development and deployment enhances security by automating vulnerability scanning and ensuring configuration compliance. AI can also compensate for the lack of general standardization for the development and deployment processes by analyzing the effect of security practices on overall container security and prioritizing effective practices to be implemented.

Sharing Knowledge about Container Security encourages the exchange of best practices and experiences, fostering a shared understanding of security issues among developers about container security. Additionally, sharing internal security standards among industries and organizations interested in container security will contribute to a collective repository of data that can inform future guidelines for securing container systems.

Human Collaboration and Communication is one of the primary non-technical factors influencing container security. Effective communication within teams enhances their understanding of security challenges and promotes collaborative efforts to address them during the development phase.

6.1. Implications on SE Practice

Our findings may assist SE practitioners by providing insights on improving and managing container security issues in the following ways:

1. Industries should focus on structured guidelines for project documentation to ensure that all relevant security measures, configurations, and challenges are recorded. Documentation should be regularly updated to reflect the current status of the project and any emerging security concerns. Maintaining structured and updated documentation facilitates knowledge retention within teams and streamlines the onboarding process for new developers.
2. Industries can highly benefit from AI security solutions in containerized projects, such as automated threat detection and self-healing systems. AI security solutions can significantly enhance system resilience and detect security breaches before they escalate. Moreover, it enables the security teams to focus on more complex threats and strategic security planning in practice.
3. Industries need to continue investing in advanced security tools and ensure tools are visible to the team to maintain security in containerised projects. Advanced security tools need to be implemented alongside practical security training. Security training must be tailored to developers for effective tool utilization. Industries need to incorporate workshops, interactive simulations, and real-world attack scenarios to help employees develop security skills.
4. The incorporation of gamification in security training can enhance engagement and knowledge retention, particularly for early-career developers. Security training programs should include interactive techniques such as real-world threat simulations with multilevel security challenges to develop a proactive security mindset. These techniques will help developers master security practices and apply them to real projects.

6.2. Future SE Research Avenues

Building upon the findings of this study, several avenues for future research can further advance the understanding of container security issues. The following research directions are proposed:

1. Establishing standardized security metrics is essential to evaluate security in containerized environments. Future SE research in containers should focus on defining measurable indicators that facilitate effective risk assessment, resource allocation, and mitigation strategies. An empirical approach can provide insights into the most critical security concerns that require prioritization.
2. Future studies should also assess the implications of security measures in different containerized environments. Employing exploratory research will help identify domain-specific security priorities and best practices to improve container security.
3. The responsibility of DevOps teams in securing container systems needs more exploration in the container context. Investigating ownership and accountability of security issues in security management can provide insights into how policies influence security outcomes. A mixed-methods approach —combining qualitative research and quantitative — can offer a comprehensive understanding and validate ownership and accountability in containers.
4. Future research should examine the ethical implications of AI security tools in container systems, particularly regarding the exposure of personal and sensitive data. Multidisciplinary research, including legal analysis, ethical frameworks, and technical evaluations, can provide a balanced perspective on the ethical implementation and implications of AI in container security.

6.3. Threats to Validity

To ensure the rigor and trustworthiness of our study, we refer to the ACM SIGSOFT Empirical Standards [60] in addressing the quality criteria of our research.

- *Credibility*: We maintain the credibility of the results by including supporting quotes for each identified theme. It also supports the reproducibility of the themes. A consolidated document with all the direct quotes, codes, and themes is available in the replication package.

- *Usefulness*: The findings of this research benefit practitioners by offering a visual model of container security management. The model highlights the patterns in containerized projects and the enablers to improve these patterns.

- *Transferability*: The model describing the security patterns and their relationship summarizes the experiences of practitioners working across various domains and roles. Additionally, it connects each pattern to the specific improving enabler. This makes the results comprehensive and applicable to a wide range of projects and domains.

- *Resonance*: We explain the strengths and weaknesses in the security patterns of container systems and provide an enabler to deepen the understanding of security management in container systems. Software practitioners could directly use these data to enhance, maintain, and manage container security.

7. Conclusion

Software containers have become a widely adopted approach for efficiently deploying software-intensive applications. However, existing SE research literature on security management predominantly focuses on technical security practices and testing methodologies, neglecting the significant role of human administration in planning, decision-making, and strategy development container systems. Consequently, this research contributes to the knowledge of security management in container systems by highlighting how SE practitioners perceive the various security challenges and their approaches to managing these security issues in software container systems.

We conducted two semi-structured interview studies to examine how practitioners manage security issues. While the first study explored how practitioners perceive security issues in containerized systems regarding their causes and implications, the second study investigated how SE practitioners manage these security issues in containerized projects.

The following are the main findings from our research:

1. Our findings provide insights into how practitioners perceive security issues, their causes, and the mitigation techniques and provide an overview of the security patterns in containerized projects.

2. The findings also present the advances of containers as a solution for deploying software applications in terms of clarity of security issues, integrating tools that help improve security and automation, and consideration of non-technical factors during developing and deploying containerized systems.
3. The findings also explore the weaknesses of containerized software systems, including the lack of systematic knowledge about security issues, guidelines uncertainty regarding practice improvements, and undefined resilience time.
4. Furthermore, we identified key enablers for addressing security issues in containers, categorizing them into technical and non-technical factors. Technical enablers include risk identification, security testing, security practices, logging and monitoring, and AI solutions. Non-technical enablers encompass knowledge sharing, effective communication, and collaboration among team members. A combination of technical and non-technical enablers ensures comprehensive addressing of container security issues on the technical and strategic levels.
5. We present diverse stakeholder perspectives on container security management. Technical practitioners expressed confidence in achieving container security through security practices and tools. In contrast, managers adopted a more cautious view, acknowledging that even with best practices and tools, persistent security issues may remain. Academic researchers offered a more skeptical perspective, particularly regarding the increasing reliance on AI that might introduce future risks.
6. We propose a conceptual model that describes how SE practitioners perceive and manage security in containerized projects. The model presents the security patterns, illustrates their interconnections, and highlights key enablers that support effective security management. The model will guide practitioners in developing robust strategies for planning and deploying highly secure container systems.

Acknowledgements

This research is supported by *Containers as the Quantum Leap in Software Development (QLeap)* project funded by Business Finland (BF) grant number 3215/31/2022.

During the preparation of this work, the author(s) used Copilot in order to enhance the readability and clarity of the text. After using Copilot,

the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] G. Liva, C. Codagnone, G. Misuraca, V. Gineikyte, E. Barcevicius, Exploring digital government transformation: a literature review, in: Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance, 2020, pp. 502–509.

[2] V. Maltese, Digital transformation challenges for universities: Ensuring information consistency across digital services, Cataloging & Classification Quarterly 56 (2018) 592–606.

[3] F. Almeida, J. D. Santos, J. A. Monteiro, The challenges and opportunities in the digitalization of companies in a post-covid-19 world, IEEE Engineering Management Review 48 (2020) 97–103.

[4] M. Koskinen, T. Mikkonen, P. Abrahamsson, Containers in software development: A systematic mapping study, in: Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20, Springer, 2019, pp. 176–191.

[5] G. Benguria, J. Alonso, I. Etxaniz, L. Orue-Echevarria, M. Escalante, Agile development and operation of complex systems in multi-technology and multi-company environments: Following a DevOps approach, in: Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings 25, Springer, 2018, pp. 15–27.

[6] D. P. VS, S. C. Sethuraman, M. K. Khan, Container security: precaution levels, mitigation strategies, and research perspectives, Computers & Security (2023) 103490.

[7] T. Combe, A. Martin, R. Di Pietro, To Docker or not to Docker: A security perspective, IEEE Cloud Computing 3 (2016) 54–62.

[8] S. Sultan, I. Ahmad, T. Dimitriou, Container security: Issues, challenges, and the road ahead, IEEE Access 7 (2019) 20.

[9] A. Martin, S. Raponi, T. Combe, R. Di Pietro, Docker ecosystem–vulnerability analysis, Computer Communications 122 (2018) 30–43.

[10] B. Kaur, M. Dugré, A. Hanna, T. Glatard, An analysis of security vulnerabilities in container images for scientific data analysis, GigaScience 10 (2021) giab025.

[11] V. Mahavaishnavi, R. Saminathan, R. Prithviraj, Secure container orchestration: A framework for detecting and mitigating orchestrator-level vulnerabilities, Multimedia Tools and Applications (2024) 1–21.

[12] L. Chen, Y. Xia, Z. Ma, R. Zhao, Y. Wang, Y. Liu, W. Sun, Z. Xue, Seaf: A scalable, efficient, and application-independent framework for container security detection, Journal of Information Security and Applications 71 (2022) 103351.

[13] R. Jolak, T. Rosenstatter, M. Mohamad, K. Strandberg, B. Sangchoolie, N. Nowdehi, R. Scandariato, Conserve: A framework for the selection of techniques for monitoring containers security, Journal of Systems and Software 186 (2022) 111158.

[14] M. Sroor, R. Mohanani, T. Das, T. Mikkonen, S. Dasanayake, Practitioners' perceptions of security issues in software containers: A qualitative study, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2024, pp. 423–430.

[15] T. Siddiqui, S. A. Siddiqui, N. A. Khan, Comprehensive analysis of container technology, in: 2019 4th international conference on information systems and computer networks (ISCON), IEEE, 2019, pp. 218–223.

[16] M. Souppaya, J. Morello, K. Scarfone, Application container security guide, Technical Report, National Institute of Standards and Technology, 2017.

[17] T. Balzacq, T. Basaran, D. Bigo, E.-P. Guittet, C. Olsson, Security practices, in: Oxford Research Encyclopedia of International Studies, 2010.

[18] M. S. I. Shamim, F. A. Bhuiyan, A. Rahman, XI commandments of kubernetes security: A systematization of knowledge related to Kubernetes

security practices, 2020 IEEE Secure Development (SecDev) (2020) 58–64.

[19] M. Liu, X. Peng, A. Marcus, C. Treude, J. Xie, H. Xu, Y. Yang, How to formulate specific how-to questions in software development?, in: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, pp. 306–318.

[20] K.-J. Stol, B. Fitzgerald, The abc of software engineering research, ACM Transactions on Software Engineering and Methodology (TOSEM) 27 (2018) 1–51.

[21] S. E. Hove, B. Anda, Experiences from conducting semi-structured interviews in empirical software engineering research, in: 11th IEEE International Software Metrics Symposium (METRICS'05), IEEE, 2005, pp. 10–pp.

[22] I. G. Ndukwe, S. A. Licorish, A. Tahir, S. G. MacDonell, How have views on software quality differed over time? research and practice viewpoints, Journal of Systems and Software 195 (2023) 111524.

[23] L. Leite, G. Pinto, F. Kon, P. Meirelles, The organization of software teams in the quest for continuous delivery: A grounded theory approach, Information and Software Technology 139 (2021) 106672.

[24] Á. González-Prieto, J. Perez, J. Diaz, D. López-Fernández, Reliability in software engineering qualitative research through inter-coder agreement, Journal of Systems and Software 202 (2023) 111707.

[25] R. Mohanani, P. Ram, A. Lasisi, P. Ralph, B. Turhan, Perceptions of creativity in software engineering research and practice, in: 2017 43rd euromicro conference on software engineering and advanced applications (seaa), IEEE, 2017, pp. 210–217.

[26] C.Anderson, Docker [software engineering], IEEE Software 32 (2015) 102–c3.

[27] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An updated performance comparison of virtual machines and linux containers, in: 2015 IEEE international symposium on performance analysis of systems and software (ISPASS), IEEE, 2015, pp. 171–172.

[28] O. Bentaleb, A. S. Belloum, A. Sebaa, A. El-Maouhab, Containerization technologies: Taxonomies, applications and challenges, The Journal of Supercomputing 78 (2022) 1144–1181.

[29] D. Bernstein, Containers and cloud: From LXC to Docker to Kubernetes, IEEE cloud computing 1 (2014) 81–84.

[30] P. Hoenisch, I. Weber, S. Schulte, L. Zhu, A. Fekete, Four-fold autoscaling on a contemporary deployment platform using Docker containers, in: Service-Oriented Computing: 13th International Conference, ICSOC 2015, Goa, India, November 16-19, 2015, Proceedings 13, Springer, 2015, pp. 316–323.

[31] A. Rahman, S. I. Shamim, D. B. Bose, R. Pandita, Security misconfigurations in open source Kubernetes manifests: An empirical study, ACM Transactions on Software Engineering and Methodology 32 (2023) 1–36.

[32] M. Sroor, Leverage software containers adoption by decreasing cyber risks and systemizing refactoring of monolithic applications, in: Product-Focused Software Process Improvement: 23rd International Conference, PROFES 2022, Jyväskylä, Finland, November 21–23, 2022, Proceedings, Springer, 2022, pp. 675–680.

[33] L. Benedicic, F. A. Cruz, A. Madonna, K. Mariotti, Sarus: Highly scalable docker containers for hpc systems, in: High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers 34, Springer, 2019, pp. 46–60.

[34] D. J. Reifer, How good are agile methods?, IEEE Software 19 (2002) 16–18.

[35] W. Kithulwatta, W. U. Wickramaarachchi, K. Jayasena, B. Kumara, R. Rathnayaka, Adoption of docker containers as an infrastructure for deploying software applications: A review, Advances on Smart and Soft Computing: Proceedings of ICACIn 2021 (2021) 247–259.

[36] H. Gantikow, S. Walter, C. Reich, Rootless containers with podman for hpc, in: International Conference on High Performance Computing, Springer, 2020, pp. 343–354.

[37] T. Xu, D. Marinov, Mining container image repositories for software configuration and beyond, in: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, 2018, pp. 49–52.

[38] F. Paraiso, S. Challita, Y. Al-Dhuraibi, P. Merle, Model-driven management of Docker containers, in: 2016 IEEE 9th International Conference on cloud Computing (CLOUD), IEEE, 2016, pp. 718–725.

[39] E. Casalicchio, Container orchestration: A survey, Systems Modeling: Methodologies and Tools (2019) 221–235.

[40] A. Ibrahim, S. Bozhinoski, A. Pretschner, Attack graph generation for microservice architecture, in: Proceedings of the 34th ACM/SIGAPP symposium on Applied Computing, 2019, pp. 1235–1242.

[41] A. Zerouali, V. Cosentino, T. Mens, G. Robles, J. M. Gonzalez-Barahona, On the impact of outdated and vulnerable javascript packages in Docker images, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2019, pp. 619–623.

[42] R. Shu, X. Gu, W. Enck, A study of security vulnerabilities on Docker hub, in: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017, pp. 269–280.

[43] H. Gantikow, C. Reich, M. Knahl, N. Clarke, Providing security in container-based hpc runtime environments, in: High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P^ 3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19–23, 2016, Revised Selected Papers 31, Springer, 2016, pp. 685–695.

[44] A. M. Dissanayaka, S. Mengel, L. Gittner, H. Khan, Vulnerability prioritization, root cause analysis, and mitigation of secure data analytic framework implemented with mongodb on singularity linux containers, in: Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis, 2020, pp. 58–66.

[45] Z. Jian, L. Chen, A defense method against Docker escape attack, in: Proceedings of the 2017 International Conference on Cryptography, Security and Privacy, 2017, pp. 142–146.

[46] A. R. MP, A. Kumar, S. J. Pai, A. Gopal, Enhancing security of Docker using Linux hardening techniques, in: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), IEEE, 2016, pp. 94–99.

[47] J. Xu, Y. Wu, Z. Lu, T. Wang, Dockerfile tf smell detection based on dynamic and static analysis methods, in: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), volume 1, IEEE, 2019, pp. 185–190.

[48] G. Budigiri, C. Baumann, J. T. Mühlberg, E. Truyen, W. Joosen, Network policies in Kubernetes: Performance evaluation and security analysis, in: 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, 2021, pp. 407–412.

[49] S. Gholami, H. Khazaei, C.-P. Bezemer, Should you upgrade official Docker hub images in production environments?, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), IEEE, 2021, pp. 101–105.

[50] S. Timonen, M. Sroor, R. Mohanani, T. Mikkonen, Anomaly detection through container testing: A survey of company practices, in: International Conference on Product-Focused Software Process Improvement, Springer, 2023, pp. 363–378.

[51] M. U. Haque, M. A. Babar, Well begun is half done: An empirical study of exploitability & impact of base-image vulnerabilities, in: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2022, pp. 1066–1077.

[52] T.-P. Doan, S. Jung, DAVS: Dockerfile analysis for container image vulnerability scanning, Computers, Materials & Continua 72 (2022).

[53] J. Cándido, M. Aniche, A. Van Deursen, Log-based software monitoring: a systematic mapping study, PeerJ Computer Science 7 (2021) e489.

[54] S. K. Mondal, R. Pan, H. D. Kabir, T. Tian, H.-N. Dai, Kubernetes in it administration and serverless computing: An empirical study and research challenges, The Journal of Supercomputing (2022) 1–51.

[55] M. Belair, S. Laniepce, J.-M. Menaud, Snappy: programmable kernel-level policies for containers, in: Proceedings of the 36th Annual ACM Symposium on Applied Computing, 2021, pp. 1636–1645.

[56] G. P. Fernandez, A. Brito, Secure container orchestration in the cloud: Policies and implementation, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019, pp. 138–145.

[57] C. B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Transactions on software engineering 25 (1999) 557–572.

[58] P. E. Strandberg, Ethical interviews in software engineering, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–11.

[59] B. DiCicco-Bloom, B. F. Crabtree, The qualitative research interview, Medical Education 40 (2006) 314–321.

[60] P. Ralph, S. Baltes, D. Bianculli, Y. Dittrich, M. Felderer, R. Feldt, A. Filieri, C. A. Furia, D. Graziotin, P. He, et al., ACM SIGSOFT Empirical Standards (2020).

[61] J. Saldaña, The coding manual for qualitative researchers, SAGE publications Ltd, 2021.

[62] D. S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: International Symposium on Empirical Software Engineering and Measurement, IEEE, 2011, pp. 275–284.

[63] S. Baltes, P. Ralph, Sampling in software engineering research: A critical review and guidelines, Empirical Software Engineering 27 (2022) 94.