Understanding Socio-Technical Aspects of Software Security Patch Management

Nishadi Nesara Munasinghe
Centre for Research on Engineering Software Technologies (CREST)
Faculty of Science, Engineering and Technology
The University of Adelaide

Principal Supervisor: Professor Muhammad Ali Babar
Co-Supervisors: Dr Asangi Jayatilaka and Dr Mansooreh Zahedi

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

March 30, 2023

Contents

List of Figures

List of Tables

The University of Adelaide

Abstract

Understanding Socio-Technical Aspects of Software Security Patch Management

by Nishadi Nesara Munasinghe

Several security attacks that resulted in catastrophic outcomes including system downtime, data breaches, financial losses, reputational damage, and in some cases, loss of life, can be traced back to a delay in applying a security patch. The most effective remediation of this problem is to apply security patches on time to the identified vulnerabilities through a process called software security patch management. Despite the criticality of timely software security patch management, it is one of the most challenging endeavours due to the inherent technical and socio-technical interdependencies involved in the process. While there have been significant research efforts on the technical aspects of security patch management, little is known about the socio-technical aspects of patch management that may cause delays in applying security patches. It is an important limitation as the software security patch management process is inherently a socio-technical endeavour where human, organisational and technological interactions are tightly coupled.

This thesis aims to fill this gap by contributing to the body of knowledge providing an in-depth evidence-based understanding of the socio-technical aspects of software security patch management. We first systematise the current state of research on socio-technical aspects of software security patch management to identify the challenges, solutions, best practices, and open research challenges. Based on a longitudinal field study involving patch meeting observations, artefacts analysis, semi-structured interviews and discussions with practitioners from 10 teams between three organisations in the healthcare domain, we then conduct in-depth empirical investigations to identify, understand and address the role and impact of socio-technical aspects on software security patch management delays in practice. The empirical findings contribute to (1) providing an evidence-based understanding of the reasons and mitigation strategies for delays in software security patch management; (2) presenting a grounded theory of the role of coordination in software security patch management explaining how (in)effective coordination contributes to a majority of the delays in the process; and (3) providing an understanding of the role of automation in software security patch management detailing insights into the as-is state of automation in practice, the limitations of current automation, how automation support can be enhanced to effectively meet practitioners' needs and the role of the human in an automated process, and proposing a set of recommendations to guide future tool development to address the identified limitations and needs, and reduce patching delays. The evidence-based knowledge and insights reported in this thesis will provide a useful resource and guideline for practitioners and researchers to identify, understand and address the socio-technical concerns leading to delays in software security patch management.

Declaration of Authorship

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Nishadi Nesara Munasinghe

March 2023

Acknowledgements

"If I have seen further, it is by standing on the shoulders of giants"

Sir Isaac Newton

First, I express my sincere gratitude to my supervisors Professor M. Ali Babar, Dr Asangi Jayatilaka and Dr Mansooreh Zahedi for their amazing support, guidance and encouragement throughout my PhD. My heartfelt thanks to Ali, my principal supervisor, for taking me under his wing to conduct my PhD research and guiding me to be an independent researcher. I am extremely indebted to him for constantly guiding me, providing me with excellent collaboration and networking opportunities, challenging me to reach heights that I didn't imagine possible, encouraging me, and most of all, for having continued faith in me.

I am deeply grateful to my co-supervisors Asangi and Mansooreh for being excellent mentors to me, always being by my side through ups and downs, motivating me to be my best version and appreciating my improvements, even the slightest ones. I owe many thanks to their constructive feedback, insightful advice and endless support throughout, without which the completion of this thesis would not have been possible.

I am immensely thankful to our industry collaborators, without whose help, this research would not have been accomplished. A special thanks to Eddy and Neil for their incredible support throughout our collaboration. Thank you for allowing us a window into your organisation, and providing us with an opportunity to participate in the patch meetings. My sincere thanks to all participants and organisations involved in this research who generously shared their time, knowledge and experiences with us.

My warmest thanks to my colleagues, the current and former members of the Centre for Research on Engineering Software Technologies (CREST) at the University of Adelaide for their encouragement and support. Special thanks to Chadni Islam, Triet Le, Roshan Rajapakse, Faheem Ullah and Bushra Sabir for their helpful feedback and thoughtful commentary on the early drafts of my papers and presentations.

I also acknowledge the CREST research lab, the University of Adelaide and the Government of Australia for the financial support provided to pursue my doctoral research and attend international conferences in the USA and Singapore to present my research.

Finally, I extend my deepest gratitude to my family for their endless support and encouragement throughout this journey. My heartfelt thanks to my best friend and partner, Kanchana Munasinghe, for always being my pillar of strength and stress reliever. Thank you for all your sacrifices, patience, advice, feedback and constant support throughout these years, I do not know how I would have done it without you. My sincere thanks to my parents, my brother and my in-laws for their constant encouragement and support to navigate through the challenges and come this far.

This thesis would not have been possible without the overwhelming support of all of you.

Dedicated to my dear parents, thank you for all your efforts in supporting me throughout my education.
Also, I am grateful to the people of Sri Lanka for providing the gift of free education, which paved the path to this thesis.

Chapter 1

Introduction

Cyberattacks exploiting software vulnerabilities remain one of the most critical risks facing modern corporate networks. Applying timely security patches to the identified vulnerabilities continues to be the most effective and widely recognised strategy for protecting software systems against such cyberattacks [1, 2]. However, despite the rapid releases of security patches addressing newly discovered vulnerabilities in software products, a majority of cyberattacks in the wild have been a result of an exploitation of a known vulnerability for which a patch had already existed [3, 4, 5, 6]. More disturbing is that such failures in the prompt application of available security patches have resulted in cyberattacks that have caused devastating consequences of huge financial and reputation losses from breaches of confidentiality and integrity of company data. For instance, the infamous Equifax case [7, 8] impacted 143 million people. In some cases, cyberattacks have even led to human fatalities, caused by the unavailability of critical software systems during an attack [9]. As such, cyberattacks that particularly target software systems in mission-critical domains such as healthcare can be catastrophic [10, 11]. For example, a failure in the timely installation of an available patch exploited by the WannaCry ransomware cryptoworm brought down several critical systems of the National Health Services (NHS) in the UK in 2017 [12]. A systematic analysis of Hospital Episodes Statistics (HES) data revealed that the cyberattack resulted in significant disruption across thousands of NHS hospitals, including the need to revert to manual processes (e.g., reporting blood results and paper notes), disruption to radiology services, 19,000 cancelled outpatient appointments, elective admissions, and day case procedures. Additionally, for five infected acute trusts, emergency ambulances were diverted to other hospitals. The financial impact was estimated to total £5.9 million for the NHS. [13, 14]. These outcomes can be largely attributed to the inherent complex issues faced when applying software security patches in organisational IT environments.

Software security patch management, hereafter called security patch management, refers to the process of applying security patches to the identified vulnerabilities in the software products and systems deployed in an organisation's IT environment. A security patch is an additional piece of code developed to address security vulnerabilities identified in software [1]. Following the discovery of a new vulnerability, a candidate security patch is developed and released by third-party vendors to prevent exploitation by malicious entities. For example, the Meltdown [15] and Spectre [16] patches released in 2018 by vendors such as Microsoft, Google, IBM and Apple were aimed at fixing two critical vulnerabilities in modern processors allowing malicious programs to gain unauthorised access to the software system. In security contexts, patch management represents a critical concern in achieving and maintaining the security of the managed software systems. This is because applying a security patch is considered the most effective mechanism to mitigate the identified vulnerabilities [17]. Similarly, applying security patches with minimum delays is instrumental in significantly reducing the risks of cyberattacks that exploit software vulnerabilities [17]. To guide the security patch management process, several guidelines such as the National Institute of Standards and Technology (NIST)'s Special Publication (SP) 800-40 [17, 18, 19] have been published over the years.

Despite the importance of timely security patch management, it remains one of the most challenging processes facing modern organisations. The process consists of identifying existing vulnerabilities in managed software systems, acquiring security patches from software vendors, testing the patches for accuracy, installing patches, and verifying installed software security patches [2, 17, 20, 21]. Performing these activities involves managing interdependencies between multiple stakeholders, and several technical and socio-technical tasks and decisions that make security patch management one of the most complex endeavours [2, 20, 22]. The need to apply a software security patch as early as possible is further challenged by the rise and distribution of an organisation's attack surface, often leaving a large number of unpatched systems vulnerable to attacks [23]. Recent industry statistics [23] reveal that an organisation's patching time for a critical vulnerability continues to increase with an average of 16 days, and medium and low severity vulnerabilities increase even further from 125 to 151 days. Such evidence reveals that modern organisations are struggling to meet the requirements of “patch early and often" indicating serious concerns and the increased importance of efforts to reduce delays in security patch management in practice.

1.1 Research Objectives and Questions

Despite the criticality of security patch management in the industry, this is still an emerging area of rising interest in research that needs further attention. While many studies have aimed at providing technical advancements to improve security patch management tasks (e.g., an algorithm for optimising patching policy selection [24], an automated approach for integrated patch testing, deployment and verification [25]), the socio-technical aspects of security patch management have received relatively limited attention. Socio-technical aspects relate to the organisational process, policies, skill and resource management and technical systems, and involve a complex interaction between humans, machines and the environmental aspects of the context [26, 27]. The lack of attention to socio-technical aspects is an important limitation because the security patch management process is inherently a socio-technical endeavour, where human, environmental and technological interactions are tightly coupled, such that the success of security patch management significantly depends on the effective collaboration of humans with the technical and organisational systems. Hence, an understanding of the socio-technical aspects is essential for identifying the prevailing issues around patching delays and improving the effectiveness of the security patch management process [2, 20].

Motivated by the increased awareness that there is a critical gap in the knowledge of the socio-technical aspects of security patch management, the research presented in this thesis aims to fill that gap by providing an evidence-based understanding of the socio-technical aspects of security patch management. Specifically, this PhD thesis focuses on empirically investigating the role and impact of socio-technical aspects on security patch management delays. Based on the findings from the empirical investigations, the thesis presents a taxonomy of root causes for patching delays, practical strategies for reducing the delays in security patch management, a framework for research and practice aimed at supporting the identification and remediation of the key reason for delays in security patching, and suggests recommendations for improving automation support for reducing delays. The problem statement of this thesis is presented as follows.

Problem statement: As security patch management is inherently a socio-technical endeavour involving a complex interaction between humans, machines and environmental aspects, a deep knowledge of the socio-technical aspects of security patch management is needed to gain a complete understanding of how human, organisational and technological factors in security patch management lead to delays. An evidence-based understanding is important to identify and understand the role and impact of socio-technical aspects on the delays in security patch management, thereby addressing the root causes of delays with solutions of high industrial relevance and practical utility.

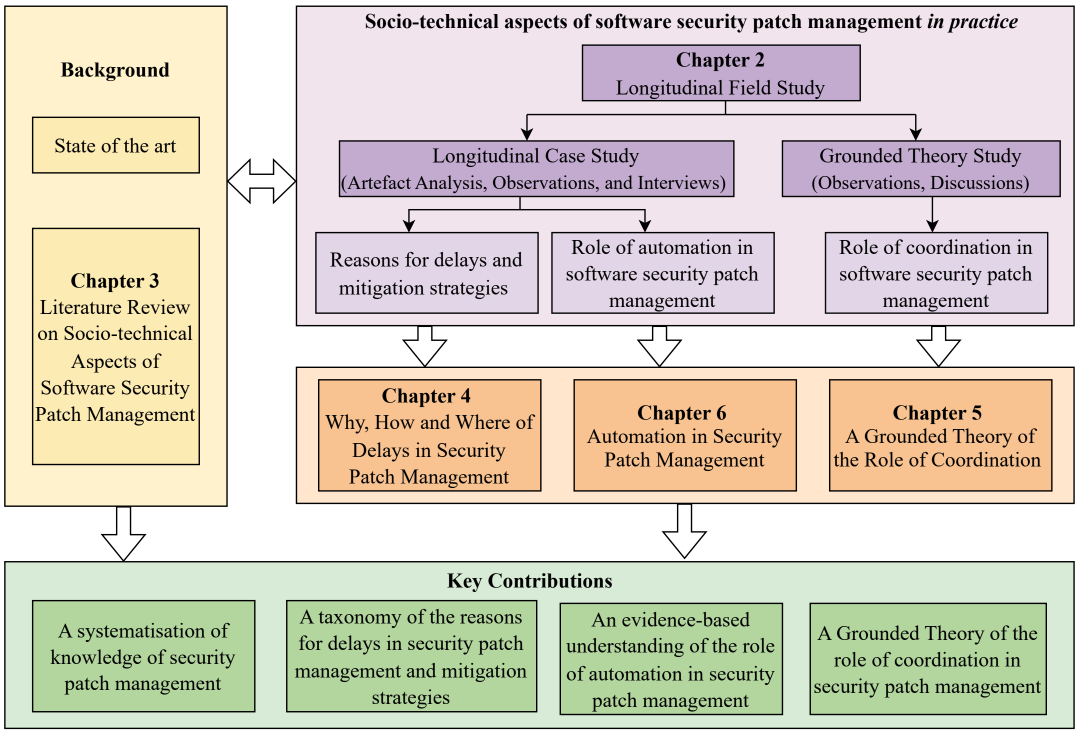

Background
State of the art
Chapter 3
Literature Review on Socio-technical Aspects of Software Security Patch Management
Socio-technical aspects of software security patch management in practice
Chapter 2
Longitudinal Field Study
Longitudinal Case Study
(Artefact Analysis, Observations, and Interviews)
Grounded Theory Study
(Observations, Discussions)
Reasons for delays and mitigation strategies
Role of automation in software security patch management
Role of coordination in software security patch management
Chapter 4
Why, How and Where of Delays in Security Patch Management
Chapter 6
Automation in Security Patch Management
Chapter 5
A Grounded Theory of the Role of Coordination
Key Contributions
A systematisation of knowledge of security patch management
A taxonomy of the reasons for delays in security patch management and mitigation strategies
An evidence-based understanding of the role of automation in security patch management
A Grounded Theory of the role of coordination in security patch management

FIGURE 1.1. Overview and scope of the thesis.

To achieve this overarching goal, we define three research questions (RQs) as illustrated in Table 1.1. Further, Table 1.1 presents the sub-research questions for each high-level RQs along with the research methods used to answer them and the corresponding thesis chapter.

1.2 Thesis Overview and Contributions

This section provides a summary of how the research objectives and questions reported in Section 1.1 and Table 1.1 are addressed across four chapters. An overview of the thesis is presented in Figure 1.1. While I, the author of this thesis, am solely responsible for the research done in this thesis, most of the work was conducted in collaboration with my PhD supervisors. Hence, the pronoun "we" is used in this thesis to reflect the collaborative research efforts. For places that need explicit distinctions

Table 1.1. An overview of the research questions and the research methods

High-level Research Question	**Sub-Research Questions**	**Research Method**	**Chapter**
RQ1: What is the state of the art in software security patch management research?	**RQ1.1:** What socio-technical challenges have been reported in software security patch management?	Systematic Literature Review	Chapter 3
	RQ1.2: What types of solutions have been proposed? **RQ1.2.1:** What approaches and tools have been proposed to facilitate software security patch management? **RQ1.2.2:** What practices have been reported to successfully implement the software security patch management process?		
	RQ1.3: How have the solutions been assessed? **RQ1.3.1:** What types of evaluation have been used to assess the proposed solutions? **RQ1.3.2:** What is the level of rigour and industrial relevance of the reported solutions?		
RQ2: What are the reasons for delays in software security patch management and how can they be mitigated?	**RQ2.1:** Why, how, and where do delays occur in software security patch management?	Longitudinal Case Study	Chapter 4
	RQ2.2: How can the delays be mitigated?		
RQ3: What is the role of coordination in software security patch management?		Grounded Theory	Chapter 5
RQ4: What is the role of automation in software security patch management?	**RQ4.1:** What is the as-is state of automation in software security patch management?	Longitudinal Case Study	Chapter 6
	RQ4.2: What are the limitations of current automation?		
	RQ4.3: How automation in software security patch management can be enhanced to support practitioners?		
	RQ4.4: What is the role of the human in software security patch management automation?		

of the roles of the researchers involved, for example, when describing the tasks of each researcher in the data analysis process, **the present author** refers to the author of the thesis and **other researchers** are used to refer to others in the research team.

Chapter 3: Systematic Literature Review

Chapter 3 provides a background on the topic of socio-technical aspects of security patch management. In response to RQ1, we conduct a comprehensive review of the state of the art of security patch management to devise a taxonomy of the existing socio-technical challenges, and solution approaches, tools, and practices in security patch management. For the solutions, we analyse their level of rigour and industrial relevance, paving the way to identify the gaps for future research. Specifically, the review provides an organised evidential body of knowledge on the topic and provides a foundation to support the main focus of the thesis and the findings reported in Chapters 4, 5 and 6. Additionally, the review serves as a guideline for researchers and practitioners to gain a thorough knowledge of the domain and identify gaps and opportunities for improvement.

Chapter 4: Why, How and Where of Delays in Security Patch Management

While several disastrous security attacks over the years can be attributed to delays in applying an available security patch, there is not much empirically known about the reasons for such delays. To address this critical gap (RQ2), Chapter 4 investigates why, how and where delays happen in security patch management in practice, and how the delays can be mitigated. Based on a longitudinal case study involving analysis of artefacts collected from 132 delayed patching tasks and observations of patch meetings, this chapter answers RQ2 by identifying and categorising the reasons that cause delays in security patch management. Further, we quantitatively analyse the identified set of reasons to uncover the most prominent causes of delays and identify where the majority of delays occur in the patch management process. Additionally, we present a set of practical strategies to mitigate the delays. To the best of our knowledge, we are the first to delve deeper into understanding the reasons for delays in security patching and mitigation strategies in practice. Such findings provide useful insights for researchers and practitioners to identify the root causes of patching delays, and mitigation points and strategies for reducing such delays.

Chapter 5: A Grounded Theory of the Role of Coordination in Security Patch Management

Chapter 4 discovers that the most prominent cause of delays is associated with coordination delays in the security patch management process. Chapter 5 addresses this problem detailed in RQ3 by providing a Grounded Theory, i.e., a framework of the role of coordination in security patch management. Based on a longitudinal Grounded Theory study of 51 observations of patch meetings in the same context as the longitudinal case study conducted in Chapter 4, the theory explains the causes that define the need for coordination among interdependencies in the security patch management process, constraints that can negatively impact effective coordination of such interdependencies, breakdowns resulting from ineffective coordination of causes and constraints, and mechanisms for ensuring effective coordination in the process. Practitioners can adapt the theory to avoid coordination delays and failures and increase confidence in their security patching decisions. Additionally, the provided understanding based on empirical evidence provides useful insights for researchers to develop solutions to address the practical concerns in coordination in the process.

Chapter 6: Automation in Security Patch Management

Other notable causes of delays in security patch management revealed by Chapter 4 are attributed to the limitations of current patch management tools and the need for human intervention in the process. While researchers and practitioners have paid significant attention to devising and integrating automation support in different activities of the security patch management process, there has been relatively little effort committed to gaining an in-depth understanding of how automation is used in practice and its constraints in meeting practitioners' needs and wants. This chapter finds answers to these questions reported in RQ4 by providing a holistic understanding of the role of automation in security patch management. The findings provide insights grounded in empirical evidence into the current status of automation in practice, the limitations of existing automation, how automation support can be improved to effectively meet practitioners' needs and the role of the human in security patch management automation. We also propose a set of recommendations that can guide future efforts aimed at developing automation support for effectively meeting the current gaps and needs, and reducing delays in security patch management.

The key contributions of this thesis from these four chapters are summarised as follows.

1. **A systematised evidential body of knowledge of the socio-technical aspects of security patch management establishing a solid background knowledge on the topic (Chapter 3)**:
   (*i*) A taxonomy of the socio-technical challenges in security patch management.
   (*ii*) A classification of the existing solution approaches, tools, and practices to address the socio-technical challenges and a detailed analysis of the mapping of the challenges to the available solutions.
   (*iii*) A detailed analysis of the solutions' rigour and industrial relevance.

2. **An evidence-based understanding of why, how, and where of delays in practical security patch management and mitigation strategies (Chapter 4)**:
   (*i*) Identification of the reasons for delays when applying security patches in practice and their distribution over the security patch management process.
   (*ii*) An understanding of the most important causes for delays with rationales.
   (*iii*) A set of practical strategies for reducing the delays in security patch management.

3. **A grounded theory of the role of coordination in security patch management (Chapter 5)**:
   (*i*) A theoretical model explaining how (in)effective coordination (which is the main cause of delay according to Chapter 4) impacts the security patch management process across four interrelated dimensions: Causes, Constraints, Breakdowns, and Mechanisms.

4. **An evidence-based understanding of the role of automation in security patch management (Chapter 6)**:
   (*i*) Insights into the as-is state of automation in real-world security patch management.
   (*ii*) Identification of the limitations of the current automation in practice.
   (*iii*) An analysis of how automation support can be enhanced to effectively meet practitioners' needs.

(*iv*) Understanding the role of the human in process automation.
(*v*) A set of recommendations for guiding future efforts aimed at developing human-automation collaborative solutions for addressing the current limitations in timely security patch management.

1.3 Publications

All core chapters of this thesis have been published during my PhD candidature. The list of publications in correspondence to each chapter is presented below. It is important to highlight that my surname recorded in the subsequent publications refers to my name before marriage.

❶ **Nesara Dissanayake**, Asangi Jayatilaka, Mansooreh Zahedi and M. Ali Babar. 2021. *Software security patch management - A systematic literature review of challenges, approaches, tools and practices.* Information and Software Technology, 144 (2021), 106771. [CORE ranking: **rank A**, Impact factor (2022): 3.862, SJR rating: Q1] (Chapter 3)

❷ **Nesara Dissanayake**, Mansooreh Zahedi, Asangi Jayatilaka, and Muhammad Ali Babar. 2022. *Why, How and Where of Delays in Software Security Patch Management: An Empirical Investigation in the Healthcare Sector.* In Proceedings of the ACM on Human-Computer Interaction, 6(CSCW2), Article 362 (November 2022), 29 pages. [CORE ranking: **rank A**, Impact factor (2022): 4.57] (Chapter 4)

❸ **Nesara Dissanayake**, Mansooreh Zahedi, Asangi Jayatilaka, and Muhammad Ali Babar. 2021. *A Grounded Theory of the Role of Coordination in Software Security Patch Management.* In Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece. ACM, New York, NY, USA, 13 pages. [CORE ranking: **rank A***, Acceptance rate: 24.5%] (Chapter 5)

❹ **Nesara Dissanayake**, Asangi Jayatilaka, Mansooreh Zahedi, and Muhammad Ali Babar. 2022. *An Empirical Study of Automation in Software Security Patch Management.* In 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA. ACM, NewYork, NY, USA, 13 pages. [CORE ranking: **rank A***, Acceptance rate: 22%] (Chapter 6)

1.4 Thesis Organisation

The remainder of the thesis is organised as follows. Chapter 2 presents the research methodologies used in this thesis. It provides a detailed description of the research methods used for each RQ with the rationale behind the choices. Chapter 3 presents a comprehensive literature review of socio-technical aspects of software security patch management. Chapter 4 investigates why, how and where the delays occur when applying security patches in practice, and how the delays can be mitigated. Chapter 5 presents a Grounded Theory based framework of the role of coordination in security patch management to address the most prominent cause of the delay identified in Chapter 4. Chapter 6 reports an empirical study of automation in security patch management, another key factor contributing to delays in patching. Chapter 7 concludes the thesis by providing suggestions for future work.

Chapter 2

Research Design

This chapter presents the research methodologies used in this thesis. It starts with a brief description of the systematic literature review, followed by the longitudinal field study involving the case study research method adopted for Chapters 4 and 6, and the Grounded Theory research method used for Chapter 5.

2.1 Systematic Literature Review

Despite the critical importance and growing industrial interest in security patch management, we found a lack of resources for establishing a solid background on the topic. Responding to this evident lack of attention in the industry and academia, we used Systematic Literature Review (SLR) method to systematically review and rigorously analyse the literature on security patch management. An SLR, one of the most commonly used research methods in Evidence-based Software Engineering, aims at consolidating the body of knowledge of a particular research question or topic by identifying, evaluating and interpreting all available research through a well-structured systematic process [28].

An SLR would largely benefit both researchers and practitioners to gain an in-depth holistic overview of the state-of-the-art of security patch management as well as support in transferring the research outcomes to industrial practice [28]. Further, the results provide useful insights to identify the limitations of the existing solutions, and gaps that need the attention of the research community. An SLR research methodology involves three main phases: planning, conducting and reporting the review [29], which we describe in detail in Chapter 3.

2.2 Longitudinal Field Study

As socio-technical aspects of security patch management are relatively less explored and given the exploratory nature of research questions (RQs), we conducted a longitudinal field study to find answers to RQs 2, 3 and 4. We chose the field study approach as it allows us to understand what practitioners do in which context and why. With access to the field over 2.7 years, we conducted a longitudinal study enabling us to study the topic in depth. The field study was conducted following the guidelines of the University of Adelaide's Human Research Ethics Committee approval (Appendix G). Based on the suitability of the RQ, we adopted two research methods in the context of the field study, Case Study and Grounded Theory, which we describe in detail in Subsections 2.2.2 and 2.2.3.

2.2.1 Field Study Context

Our field study included three case organisations (Org A, B and C) with 10 teams in Australia. In selecting the case organisations, we used a combination of purposive [30] and convenience [31] sampling to ensure that our data are representative of the substantive area through which the findings emerge. We initially held an informal meeting with the senior manager of security services in Org A to understand the company's security patch management process and the organisational setup with Org B and C, the security patch management service providing companies. It was confirmed to us during this meeting that this case presented an excellent opportunity to study security patch management in a large-scale organisation with a highly complex environment and a large set of diverse stakeholders. Further, the meeting enabled us to understand available data collection sources for our field study.

Org A is a large public health services agency responsible for maintaining the IT systems in an Australian state government's healthcare sector. In addition to maintaining third-party vendors' software systems, they also maintained some medical software applications developed in-house. However, security patch management of operating systems (OS) and associated third-party software applications were outsourced to Org B and Org C, which are IT services and consulting organisations. Org B and Org C are sister companies of a large American multinational corporation, and they were responsible for patching Org A's 1500 servers representing the entire state's healthcare system. The non-security patches were handled by different teams in Org A, which is not included in our scope. The field study revolves around the domain of healthcare as the case organisations focus on security patch management in the healthcare domain in the studied context.

The demographics of the studied teams in the case organisations are presented in Table 2.1. Org A consisted of several teams each managing different modules, for example, teams T1-3 represented the main modules in Org A while teams T5-6 were central to all other in-house teams overseeing their respective modules. The security patch management process was coordinated through bi-weekly patch meetings between the three organisations, attended by key stakeholders representing each team Table 2.1. Abiding by the human ethics guidelines, the details of the companies, teams and participants have been kept confidential.

2.2.2 Longitudinal Case Study

Given the exploratory nature of RQs 2 and 4, we adopted a case study approach to obtain an in-depth understanding of the reasons and mitigation strategies for real-world security patching delays and the role of automation in security patch management. The case study research method allows "*the study of a contemporary phenomenon (the "case") in depth and within its real-world context, especially when the boundaries between phenomenon and context are unclear*" [32], such as the complex socio-technical aspects involved in real-world security patch management. Our longitudinal case study follows an exploratory, holistic and multiple-case study approach in which we applied an ethnographic approach to the data collection and a Grounded Theory approach to the data analysis. We followed the guidelines for case study research by Yin [32] and its alteration to software engineering research by Runeson and Höst [33] in the case study design, conducting and reporting.

TABLE 2.1. Demographics of case organisations

Company	Team	Team's Domain	Team Size *	Roles
Org A	T1	Electronic Medical Records (EMR)	5	Application Owner, System Administrator, Server Engineer, Server Manager
	T2	Digital Health Windows (Win)	3	Server Engineer, System Administrator, Application Services Manager
	T3	Digital Health Non-Windows (Non-Win)	2	Unix Specialist, Server Engineer, System Administrator
	T4	Clinical and Pathology Services	1	Pathology Server Engineer
	T5	Security	1	Security Advisor
	T6	Change Management	1	Change Manager
Org B	T1	Server (Technical)	7	Server Engineer, Senior Server Engineer, Unix Engineer, Server Manager, Client Delivery Manager
	T2	Finance and Audit (Non-technical)	1	Accounts Manager
Org C	T1	Operations (Technical)	1	Account Run Lead
	T2	Server (Technical)	1	Senior Server Engineer

* The team size and roles refer to the number and roles of team participants in the patch meeting.

Data Collection

In our longitudinal case study approach, we collected data using various methods including artefact analysis, interviews, and observations as shown in Figure 1.1. Our data consist of artefacts of patch meeting minutes spanning over four years from October 2016 to May 2021, observations of 66 patch meetings, and 17 in-depth interviews. Based on the need and suitability, we selected the appropriate data collection technique for each research question. We first started the data collection with two organisations (Org A and B) and included a third organisation (Org C) based on the need as the research progressed.

Artefact Analysis: To understand why and how delays occur in practice (RQ2), we collected longitudinal data from patch meeting minutes maintained by the case organisations as the main artefact from patch meetings. This study included two organisations (Org A and Org B) involving 21 participants from 8 teams, and the demographics of the studied teams are illustrated in Table 2.1. Figure 2.1 shows the organisational setup in the studied context.

As the main source of data collection, we gathered patch meeting minutes referred to as the *"patching tracker"* by the studied teams. The patching tracker, a detailed Excel spreadsheet, was used as a tracking tool between the collaborative parties to document the status of the tasks, similar to centralised version control and issue tracking system. The three main teams of Org A (i.e., T1, T2, and T3) each maintained separate patching trackers to document their patch management tasks (i.e., activities) with details of the task number, subject, raised date, action required or taken, raised by, owner, assigned to and the status (including *Closed*, *In-progress*, *New*, *On-hold* and *Monitor*), as shown in Figure 2.2. Each tracker was updated regularly with the date

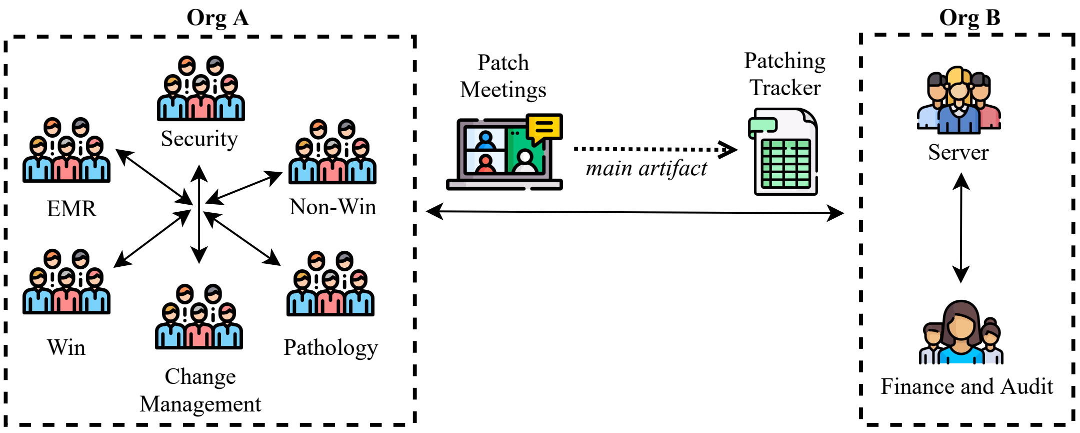
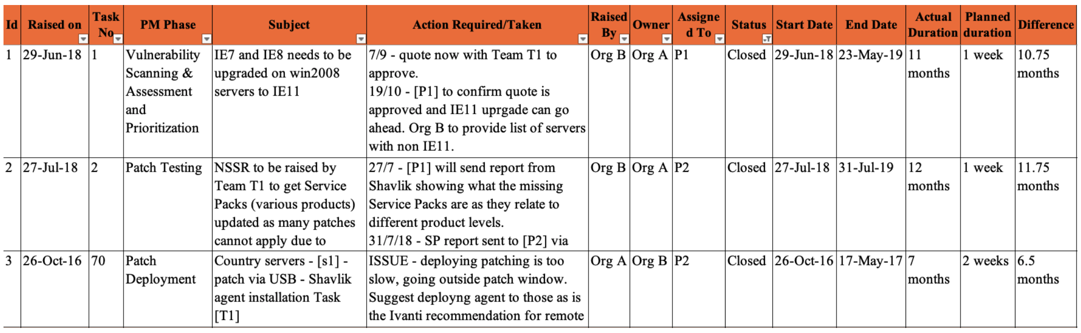


FIGURE 2.1. The organisational setup present in the studied context.

and action or decision taken when the task was discussed in detail at patch meetings.



Id	Raised on	Task No	PM Phase	Subject	Action Required/Taken	Raised By	Owner	Assigned To	Status	Start Date	End Date	Actual Duration	Planned duration	Difference
1	29-Jun-18	1	Vulnerability Scanning & Assessment and Prioritization	IE7 and IE8 needs to be upgraded on win2008 servers to IE11	7/9 - quote now with Team T1 to approve. 19/10 - [P1] to confirm quote is approved and IE11 uprgade can go ahead. Org B to provide list of servers with non IE11.	Org B	Org A	P1	Closed	29-Jun-18	23-May-19	11 months	1 week	10.75 months
2	27-Jul-18	2	Patch Testing	NSSR to be raised by Team T1 to get Service Packs (various products) updated as many patches cannot apply due to	27/7 - [P1] will send report from Shavlik showing what the missing Service Packs are as they relate to different product levels. 31/7/18 - SP report sent to [P2] via	Org B	Org A	P2	Closed	27-Jul-18	31-Jul-19	12 months	1 week	11.75 months
3	26-Oct-16	70	Patch Deployment	Country servers - [s1] - patch via USB - Shavlik agent installation Task [T1]	ISSUE - deploying patching is too slow, going outside patch window. Suggest deployng agent to those as is the Ivanti recommendation for remote	Org A	Org B	P2	Closed	26-Oct-16	17-May-17	7 months	2 weeks	6.5 months

FIGURE 2.2. A screenshot of an extract from the Patching Tracker - 19.05.2021.

Observations: Additionally, to supplement our understanding of the security patch management process and activities, and to obtain a better understanding of the documented tasks in the patching tracker, we observed 66 patch meetings from March 2020 - May 2021. The patch meetings provided a collaborative platform for the participants to discuss and refine the patching process, plan monthly patch schedules, assess the vulnerability remediation progress, resolve problems, and make decisions about patch exemptions. The meetings were held every fortnight lasting approximately an hour and a half. Due to COVID-19, the meetings were held online through Microsoft Teams. In addition, the observations increased analytical validity and ensured triangulation in our findings [32].

Interviews: To understand the role of automation in security patch management in practice (RQ4), we extended the longitudinal case study by adding Org C to the data collection using semi-structured interviews. We chose interviews as the data collection method to gain a better understanding of real-world practices through practitioners' perspectives. Further, we chose semi-structured interviews over fully structured interviews as they allow freedom to participants to voice their real concerns rather than forcing them to a topic. We collected data through semi-structured interviews with 17 practitioners from three organisations, Org A, B and C. In comparison to previous studies aimed at addressing RQs 2 and 3, the addition of Org C as an additional data source in this study helped verify our existing findings in a broader sample in the case.

For participant recruitment, we requested the manager of Org A's security team to suggest potential interviewees engaged in security patch management in the case organisations. Then, we sent email invitations to the nominated candidates. The participant selection covered a diverse set of job roles in security patch management to achieve a rounded perspective. A pre-interview questionnaire was emailed to the participants to collect demographic information about them and their teams. The demographics of the participants are presented in Table 2.2. The participants were distributed across 10 teams in three organisations, and on average had 22 years of experience in security patch management. The participant (PID) and organisation identities have been omitted to preserve participant anonymity as per the human ethics guidelines. The last three columns refer to the number, types of machines or devices, and types of software components managed by the participants.

The interviews held in November-December 2021, lasted between 30 and 60 minutes and were conducted via Microsoft Teams because of Covid-19 restrictions. The present author led all the interviews, while the other researchers participated in seven interviews (researcher2 = 4 interviews and researcher3 = 3 interviews) along with the present author for helping in asking the follow-up questions and increasing the reliability of the interviewing process.

The interviews were focused on security patch management of operating systems (OS) and third-party software applications in the context of healthcare. The open-ended interview questions covered the following areas: (a) role and responsibilities in security patch management; (b) the current status of automation tools used in the process including features of the specific tools used and how they supported security patch management tasks, and the activities not supported by automation; (c) the challenges and limitations of the current automation tools; (d) role of humans in security patch management automation and the reasons for human involvement, and (e) the need for better automation support (i.e., the envisioned solutions and their features).

All researchers jointly prepared the interview guide which was modified with mutual agreement when appropriate during data collection to reduce the potential bias. All interviews were recorded and transcribed for analysis by the present author. The interviews were conducted in two phases in parallel with the data analysis. We continued with the data collection until the data analysis confirmed *theoretical saturation* [34], for example, the last three interviews provided more examples for the emerged findings but no new categories or insights emerged.

Data Analysis

In search of answers for RQ2: why and how delays happen in security patch management, and RQ4: what's the role of automation in security patch management in practice, we qualitatively analysed the data employing the Grounded Theory (GT) [34, 35] data analysis procedures. Additionally, to identify how the delays and the causes are distributed throughout the security patch management process in RQ2, we quantitatively analysed the data using frequency analysis, a widely used technique for producing descriptive statistics derived from the data.

For qualitative data analysis, we used Grounded Theory (GT) [34, 35] data analysis technique as it aligned well with our case study goals of investigating a phenomenon in a real-world context. We adopted the data analysis procedure in Strauss and Corbin's GT version (Straussian GT) [34] as it provides a well-structured and rigorous data analysis approach guided by open-ended and practice-based RQs [34, 36]. The data analysis procedure followed different types of coding: ***Open coding***, ***Axial coding***

TABLE 2.2. Participant demographics

PID	Role	Yrs of Exp	Org	Domain	Team	Team Size	No of Machines	Machine Type	Software Component Type
P1	Lead Solution Analyst	20	Org A	Healthcare	T1	65	250	Server	Applications [*], EOL [+]
P2	Team Lead	25	Org A	Healthcare	T2	17	3000	Server	OS
P3	Infrastructure Specialist	23	Org A	Healthcare	T2	17	3000	Server	OS, Applications, Drivers and Firmware
P4	Server Administrator	23	Org A	Healthcare	T2	17	3000	Server	OS, Applications, EOL
P5	Infrastructure Server Engineer	17	Org A	Healthcare	T2	17	3000	Server	OS
P6	Manager (Infrastructure Server)	25	Org A	Healthcare	T2	17	3000	Server	OS
P7	Service Transition Manager (EUC)	35	Org A	Healthcare	T3	3	35000	Client	OS, Applications
P8	ICT Change Manager	22	Org A	Healthcare	T4	3	38000	Server, Client, Network devices	All Types
P9	Senior Manager (Security Services)	20	Org A	Healthcare	T5	6	38000	Server, Client	OS, Applications [*], EOL, Drivers and Firmware
P10	Senior Security Advisor	40	Org A	Healthcare	T5	6	38000	Server, Client	OS, Applications [*], EOL, Drivers and Firmware
P11	Manager (EMR)	19	Org A	Healthcare	T6	250	160	Server	Applications
P12	Change Coordinator (EMR)	19	Org A	Healthcare	T6	250	160	Server	OS, Applications, EOL
P13	Patching Lead	3.5	Org B	IT	T7	5	1800	Server	OS, Applications
P14	Client Delivery Manager	21	Org B	IT	T8	50	160	Server	OS
P15	Account Delivery Lead	20	Org B	IT	T8	50	2500	Server	OS, Applications, EOL
P16	Account Run Lead (Operations)	14	Org C	IT	T9	130	38000	Client	OS, Applications
P17	Senior Server Engineer	25	Org C	IT	T10	30	1500	Server	OS

[*] Includes both third-party applications and custom in-house applications. [+] EOL = End-Of-Life (Legacy) Software

and ***Selective coding*** [34]. The present author conducted the data analysis and created a codebook containing all the codes and memos. The codebook and raw data (i.e., artefacts and observation and interview transcripts) were shared with all researchers. The other researchers cross-checked all the codes, concepts, categories, and core categories in the codebook against the raw data throughout the data analysis process to reduce bias and increase the reliability of the findings [37]. The categories and their relationships were thoroughly discussed in weekly meetings and finalised after several rounds of revisions among all researchers. Any disagreements were resolved in weekly detailed discussions between all researchers throughout the process. The use of analytical tools such as diagramming and memoing facilitated the data analysis process. The raw data, codes, and memos were stored in NVivo [38], a qualitative analysis tool.

For the qualitative data analysis in RQ2, i.e., to understand *why* and *how* delays happen in security patch management, we analysed the data from artefacts and supplementary observations at the task level as previous studies [39, 40] have shown that most project delays are caused by delays in the smallest unit of work (i.e., task-level delays). A task in this study refers to a single row recorded in the patching tracker. Out of 268 tasks available in total, 232 tasks were closed. We only analysed closed tasks since we needed the end dates to calculate delays. Figure 2.2 shows a screenshot of an extract from the patching tracker. To define a delay according to the studied context, the present author held a discussion with Org A's Senior Security Advisor about their policies to understand the defined time frames for any given task during the monthly patch cycle practised in the studied context. Table 2.3 presents a summary of the standard time frames as expected in the organisation. Correspondingly, we mapped the tasks to their relevant phase of the patch management process based on two existing studies [2, 20].

TABLE 2.3. Definition of standard time frames in the studied case

Phase ID	Patch management process phase	Standard time frame	Note
P1	Patch Information Retrieval	2 days	Needs to be completed within two days of patch release
P2	Vulnerability Scanning, Assessment and Prioritisation	1 week	Needs to be completed within the first week of patch release
P3	Patch Testing	1 week	Needs to be completed within the second week of patch release
P4	Patch Deployment	2 weeks	Needs to be completed within the fourth week of patch release
P5	Post-Deployment Patch Verification	1 month	Any post-deployment issues must be resolved by the next patch cycle

The preliminary analysis revealed 132 delayed tasks from a total of 232 closed tasks that we analysed (56.9%). While there were 57 tasks (24.6%) not delayed, the remaining tasks were excluded for several reasons such as duplicate tasks, lack of information (e.g., no end date), and not being related to patch management specifically. To understand the causes of delays and remediation mechanisms, we analysed in-depth the delays identified through preliminary analysis following ***Open***, ***Axial*** and ***Selective*** coding procedures [34], as shown in Figure 2.3.

We started with **Open coding**, whereby we analysed all columns row by row in

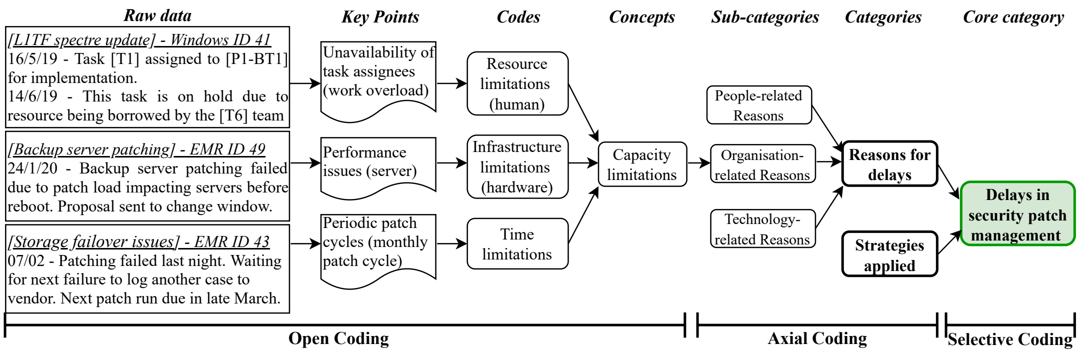

Raw data
Key Points
Codes
Concepts
Sub-categories
Categories
Core category
[L1TF spectre update] - Windows ID 41
16/5/19 - Task [T1] assigned to [P1-BT1] for implementation.
14/6/19 - This task is on hold due to resource being borrowed by the [T6] team
[Backup server patching] - EMR ID 49
24/1/20 - Backup server patching failed due to patch load impacting servers before reboot. Proposal sent to change window.
[Storage failover issues] - EMR ID 43
07/02 - Patching failed last night. Waiting for next failure to log another case to vendor. Next patch run due in late March.
Unavailability of task assignees (work overload)
Performance issues (server)
Periodic patch cycles (monthly patch cycle)
Resource limitations (human)
Infrastructure limitations (hardware)
Time limitations
Capacity limitations
People-related Reasons
Organisation-related Reasons
Technology-related Reasons
Reasons for delays
Strategies applied
Delays in security patch management
Open Coding
Axial Coding
Selective Coding

FIGURE 2.3. Emergence of the category *Reasons for delays* from the underlying concept of *Capacity limitations* and codes in RQ2.

the spreadsheet to identify *key points* summarising the content. It was further encapsulated into *codes* containing short phrases. Constant comparison of emerged codes between each team's patching tracker, different teams' patching trackers of a single meeting, and different meetings resulted in *concepts* [34], a higher level of abstraction of the *codes*. Similarly, we grouped *sub-categories*, and continuously comparing sub-categories gave rise to *categories*, the next level of abstraction. Next, we performed **Axial coding** in which we systematically linked the emerged categories to their sub-categories based on the relationships between categories relating to their *properties* (i.e., "characteristics of a category") and *dimensions* (i.e., "variations within properties") [34]. This process of axial coding was guided through activities involving referring back and forth to *memos*, i.e., notes written during Open coding explaining the codes and their relationships, drawing the inter-relationships in diagrams, and refining them through frequent team meetings. The process continued until no new properties, dimensions, or relationships emerged for each category, which indicated *theoretical saturation* [34]. Finally, we applied ***Selective coding*** where we finalised the categories and integrated them with the *central* or *core* category which represents the most recurrent and central problem in the studied phenomenon, or simply which explains *"what this research is all about"* [34], in this case, delays in security patch management.

The same process continued for the data analysis of interviews in search of answers for RQ4: the role of automation in security patch management, as shown in Figure 2.4. The raw data obtained in this case study cannot be shared because of confidentiality agreements with the industry collaborators, which happen to be in the health sector where data sensitivity has extra layers of governance. However, we made our codebook containing the codes, descriptions and examples of raw data for RQ2 available ((see Appendix A and Appendix B). We have also made our interview guide containing the pre-interview questionnaire and interview questions available in Appendix C).

Member Checking

We conducted a member checking [41] session to ensure the credibility, accuracy, validity, and transferability of our findings for RQ2. Member checking, a technique of *"taking ideas back to research participants for the confirmation"* [42], provides an opportunity to validate the findings with participants and resonance with their experiences [43]. We presented the findings at a session held at Org A. The present author and two other researchers attended the session in person while nine patch meeting participants (six from Org A and three from Org B) and an executive director of Org

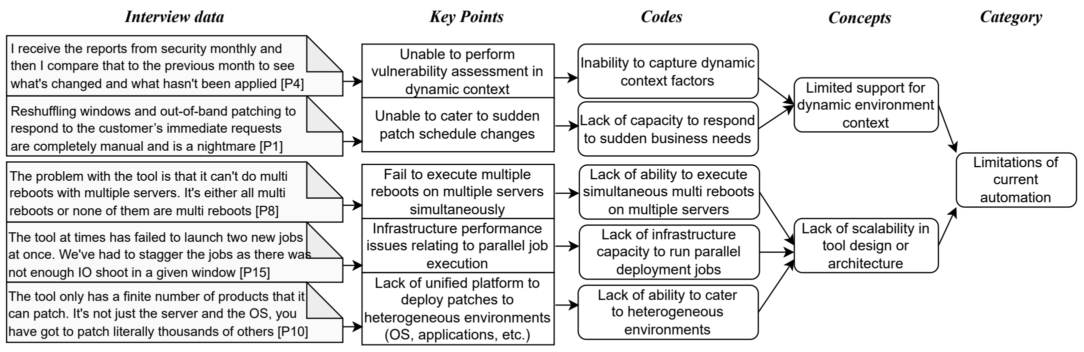

Interview data
Key Points
Codes
Concepts
Category
I receive the reports from security monthly and then I compare that to the previous month to see what's changed and what hasn't been applied [P4]
Reshuffling windows and out-of-band patching to respond to the customer's immediate requests are completely manual and is a nightmare [P1]
The problem with the tool is that it can't do multi reboots with multiple servers. It's either all multi reboots or none of them are multi reboots [P8]
The tool at times has failed to launch two new jobs at once. We've had to stagger the jobs as there was not enough IO shoot in a given window [P15]
The tool only has a finite number of products that it can patch. It's not just the server and the OS, you have got to patch literally thousands of others [P10]
Unable to perform vulnerability assessment in dynamic context
Unable to cater to sudden patch schedule changes
Fail to execute multiple reboots on multiple servers simultaneously
Infrastructure performance issues relating to parallel job execution
Lack of unified platform to deploy patches to heterogeneous environments (OS, applications, etc.)
Inability to capture dynamic context factors
Lack of capacity to respond to sudden business needs
Lack of ability to execute simultaneous multi reboots on multiple servers
Lack of infrastructure capacity to run parallel deployment jobs
Lack of ability to cater to heterogeneous environments
Limited support for dynamic environment context
Lack of scalability in tool design or architecture
Limitations of current automation

FIGURE 2.4. An example of the data analysis steps in RQ4 leading to a category from raw data as evidence.

A were present physically. In addition, seven patch meeting participants (four from Org A and three from Org B) attended the session virtually. The present author presented the findings for 20 minutes followed by a detailed feedback discussion lasting for 40 minutes. For the member checking, we revisited findings for each sub-RQ of RQ2 and asked questions including if they agree with the findings, which reasons for delays they have encountered the most in their experience, any other reasons or strategies they use that are not captured in the findings, and if they can relate the findings with their experiences. The session was audio-recorded with permission and transcribed for analysis by the present author. The feedback and comments from member checking are presented in Section 4.4.3.

2.2.3 Grounded Theory

We used Grounded Theory (GT) [35, 44] research method for understanding the role of coordination in security patch management (RQ3). Grounded Theory (GT) is a rigorous qualitative research method that enables the systematic generation of theory from data, relating to social interactions and behaviour in real-world settings [35, 45]. GT is a complete research method that includes data collection, analysis, theory development, and reporting. However, given its rigour and well-structured approach, GT can also be used as a data analysis approach rather than adopting the complete research method leading to theory development, as we have used in Section 2.2.2 for RQs 2 and 4. The choice of GT as our research method for RQ3 was based on two reasons:

(a) The aim of our research, to understand the socio-technical aspects causing delays in security patch management in practice, e.g., the role of coordination in security patch management, suited well with GT as it allows the investigation of people and interactions in a real-world phenomenon [45].

(b) GT is considered most relevant to research areas that have not been deeply explored before [46], and research on the socio-technical aspects of security patch management is limited in the literature.

We followed the Glaserian version of GT [44] since it offers more flexibility to uncover the underlying concerns from the emergent data rather than limiting the research angle with a defined research hypothesis upfront like in the Straussian GT version. Following the guidelines, we started with an *"area of interest"* - socio-technical concerns in security patch management delays. The guidelines by Stol et al. [36] were followed for reporting the GT findings. [1]

Data Collection

We collected data through observations and discussions with practitioners as shown in Figure 1.1. We selected an ethnographic data collection method such as observation as it aligned well with our goals of understanding how the role of coordination unfolds in a real-world setting of security patch management [47]. Additionally, ethnographic data collection approaches allow the researcher to gain a deep understanding of the studied topic by engaging with and in the context of the participants for a long time [47, 48].

We observed 51 patch meetings between Org A and Org B between March 2020 - January 2021. It should be noted that Org A and Org B refer to the same case organisations described in Section 2.2.1 and patch meetings refer to the same meetings reported in Section 2.2.2 (under Observations). The fortnightly patch meetings were attended by 21 key stakeholders from 8 teams. These stakeholders represented diverse roles centred on decision-making, planning, and executing security patching. Table 2.1 presents the investigated teams' demographics. Regarding the teams' distribution, all teams of Org A (T1-6) were co-located while Org B teams (T1-2) were distributed across two locations in Australia.

We held discussions immediately after the meetings with the senior security advisor of Org A to clarify any doubts that emerged during the observations and gather additional information. Table 2.4 presents a summary of the data collection. We also gathered additional data by analysing artefacts such as meeting minutes and patch mailing threads to supplement our understanding of the process, practices, and used terminology.

Table 2.4. Summary of the data collection

No. of meetings	Duration	No. of discussions	No. of hours	
			Meetings *	Discussions +
51	9 months	11	30 hours	7 hours

* The average time of a patch meeting=30 minutes.

+ The average time of a post-patch meeting discussion=30-45 minutes.

The present author attended all 51 meetings that were held over the course of 9 months and conducted all 11 post-meeting discussions. All the meetings and discussions were audio-recorded with permission and shared with all researchers. We adopted the protocol proposed by Spradley [49] to guide the data collection during the meetings, which is available in Appendix D. The present author briefed other researchers about the key aspects of the fortnightly meetings and the post-meeting discussions regularly. The data collection and analysis were performed in iterative and intertwined stages throughout. We continued with the data collection until the data analysis confirmed *theoretical saturation*. The last few observations (M46-M51) provided more examples and evidence for the emerged findings during the analysis, but no new concepts, categories, or insights emerged. All researchers mutually agreed that this was a clear indication of the *theoretical saturation* [34] and that any additional data collection would not add value to the findings.

[1]Grounded Theory (GT) is used to refer to the research method while grounded theory is used to refer to the outcome/product of the research.

Data Analysis

We followed Glaser's data analysis procedure starting from ***Open coding*** through ***Selective coding*** to ***Theoretical coding*** [36, 44, 50]. The data analysis was led by the present author and supported by other researchers who took the role of validators at each stage throughout the iterative and intertwined rounds of data collection and analysis. All data including transcripts, observation and discussion notes, other artefacts (meeting minutes and patch mailing thread notes), codes, and memos were saved in the NVivo [38] data analysis tool and shared with all researchers. The other researchers cross-validated all the emergent codes, concepts, categories, and core categories. Any conflicts in the coding and coding procedures were resolved in weekly detailed discussions between all researchers throughout the analysis phase involving several rounds of revisions. Additionally, the emerged findings were further cross-checked with one of the senior members of Org A's security team.

Open coding started with thoroughly reviewing the transcripts and recording *key points* containing summarised phrases [51]. It was further summarised into *codes* of three-five words each, and any specific *properties* of the code were captured in brackets, as shown in the example below.

Transcript: *"The patching is delayed because this .NET security vulnerability was reported in August after patching happened for the month. But, we also have another problem as this is a different version of .NET from what is standing across the fleet. This is .NET core, not .NET version 4.801."*
Key Point: *Need to identify and match Framework version dependencies*
Code: *Software application interdependencies (version)*

Applying constant comparison on the codes that emerged within each observation, between different observations, and post-meeting discussions, we grouped them to a higher level of abstraction, i.e., *concepts* [35, 44]. Similarly, continuously comparing concepts produced *categories*, a third-level of abstraction, and categories generated *core categories* [44] at the end of the first round of coding. The core category represents the main problem or concern (core) in the studied phenomenon, which presents the research question in the Glaserian GT version [45, 52]. Correspondingly, three potential *core categories* emerged - "Legacy software systems", "Role of coordination", and "Role of patch meetings". The percentage split of codes between the three core categories was 18%, 51.3%, and 30.7% respectively. We selected "Role of coordination in security patch management" as the core category because it met all the criteria defined by Glaser [45] for selecting a category as the core. For example, the selected core category was central to other categories and frequently occurred in the data; meaningfully related to both other categories easily and took the longest to saturate. Our decision to focus on the "role of coordination" from the initial general focus (i.e., *area of interest*) of the "role of the socio-technical aspects in security patch management delays" was informed by Glaser's Grounded Theory guidelines [45].

After establishing the core category, we continued ***Selective coding*** [45] limited to only those codes that were related to the selected core category. For example, Figure 2.5 illustrates the emergence of the category of *Technical dependencies* that relate to the *Role of coordination* in the second round of coding - Selective coding. We continued to selectively code until no new insights or aspects emerged for each category, which indicated *theoretical saturation* [35, 44].

As the final step of the analysis, we applied ***Theoretical coding*** [36, 44, 45, 53] to establish conceptual relationships between categories, resulting in the development of a theory. We used *memos* (*memo sorting*) [44, 45] to guide us to uncover the

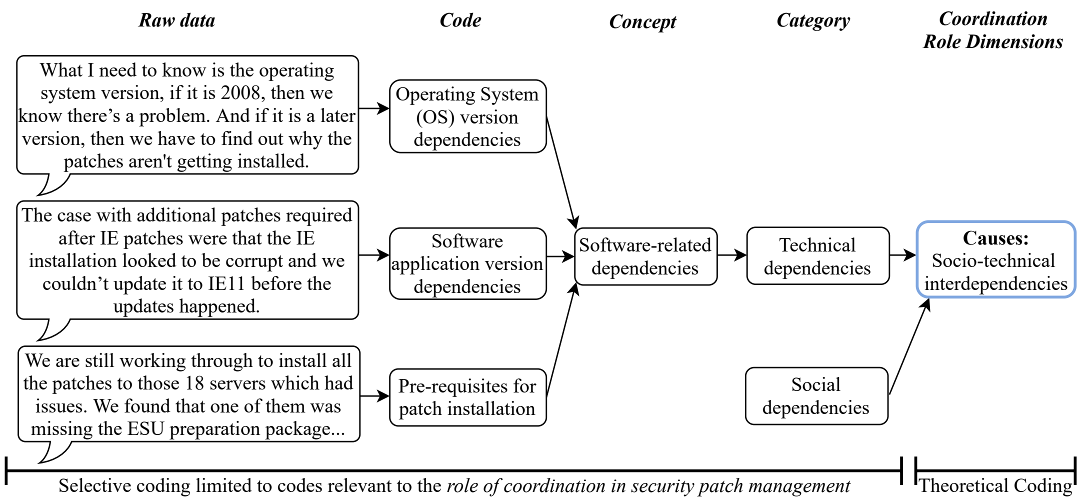

Raw data
Code
Concept
Category
Coordination Role Dimensions
What I need to know is the operating system version, if it is 2008, then we know there's a problem. And if it is a later version, then we have to find out why the patches aren't getting installed.
The case with additional patches required after IE patches were that the IE installation looked to be corrupt and we couldn't update it to IE11 before the updates happened.
We are still working through to install all the patches to those 18 servers which had issues. We found that one of them was missing the ESU preparation package...
Operating System (OS) version dependencies
Software application version dependencies
Pre-requisites for patch installation
Software-related dependencies
Technical dependencies
Social dependencies
Causes: Socio-technical interdependencies
Selective coding limited to codes relevant to the role of coordination in security patch management
Theoretical Coding

FIGURE 2.5. The emergence of category *Socio-technical interdependencies* from the underlying codes and concepts in RQ3.

links between the categories when developing the theory. At this point, we consulted the literature, particularly Glaser's *'Theoretical Coding Families'* [45] to find if any existing theoretical structure fits our current findings to visualise the theory. Glaser argues using a coding family to present a grounded theory would help increase the completeness and relevance of the emerging theory [45, 53]. As the findings (i.e., the causes, constraints, breakdowns, and mechanisms) emerged from the data, we found that the theoretical coding model *"Dimension family"* is the best fit to visualise the relationships between the categories. The *Dimension family* is a theoretical structure that enables the findings to be presented as dimensions or elements of a phenomenon [45, 53], in this case, dimensions of the role of coordination in security patch management. Thus, the theory of the role of coordination in security patch management, depicted in Figure 5.1, is described using: (a) ***Causes:*** socio-technical dependencies that define the need for coordination; (b) ***Constraints:*** factors that hinder coordination; (c) ***Breakdowns:*** scenarios of patching failures resulting from ineffective coordination of the causes and constraints; and (d) ***Mechanisms:*** strategies devised for supporting the coordination in security patch management.

Chapter 3

A Systematic Literature Review on Software Security Patch Management

Related publication: This chapter is based on our paper titled *"Software security patch management - A systematic literature review of challenges, approaches, tools and practices", published in the Information and Software Technology journal (CORE Ranking A) [54].*

Security patch management purports to support the process of patching known software security vulnerabilities. As mentioned in Chapter 1, patching security vulnerabilities in large and complex systems is a hugely challenging process that involves multiple stakeholders making several interdependent technological and socio-technical decisions. Given the increasing recognition of the importance of security patch management, it is important and timely to systematically review and synthesise the relevant literature on this topic. This chapter aims to systematically review the state of the art of security patch management to identify the socio-technical challenges in this regard, reported solutions (i.e., approaches, tools, and practices), the rigour of the evaluation and the industrial relevance of the reported solutions, and to identify the gaps for future research. To achieve this goal, we conducted a systematic literature review (SLR) of 72 studies published from 2002 to March 2020, with extended coverage until September 2020 through forward snowballing. Based on the findings that highlight the important concerns in security patch management and the lack of solutions, we discuss a list of future research directions at the end of the chapter that opens the doors for the main contributions of this thesis in Chapters 4, 5 and 6.

3.1 Introduction

As mentioned in Chapter 1, cyber attacks targeting unpatched software vulnerabilities continue to pose a critical risk to modern organisations. Applying security patches to identified software vulnerabilities on time is regarded as the most effective and widely recognised strategy for protecting software systems against such cyberattacks [17]. Despite the critical importance of timely security patch management, cyber attacks resulting from patching delays continue to rise with increased concerns about the state of security patch management in practice and suggest the need for more attention on the topic with evidence-based research [55].

While this is an emerging area of rising interest in research, the focus has been mainly on the technical aspects of security patch management. Despite the importance of understanding the socio-technical aspects of security patch management as described in Chapter 1, this area of focus has been largely neglected in research. To the best of our knowledge, there has been no review or survey aimed at organising the body of knowledge on the socio-technical aspects of security patch management covering the existing challenges and solutions in this area.

Responding to this evident lack of attention to a highly critical and timely topic with growing interest in the industry and academia, we aimed to systematically analyse the literature on security patch management. A systematic review would largely benefit both researchers and practitioners to gain an in-depth holistic overview of the state-of-the-art of security patch management as well as support in transferring the research outcomes to industrial practice [28]. Further, the results provide useful insights to identify the limitations of the existing solutions, and gaps that need the attention of the research community. We focus on the less-explored socio-technical aspects of security patch management investigating the challenges and existing solutions to address those challenges reported in 72 primary studies. The key contributions of this novel systematic literature review (SLR) are as follows:

1. A consolidated body of knowledge of research on socio-technical aspects of security patch management, providing guidance for practitioners and researchers who want to better understand the process.

2. A comprehensive understanding of the socio-technical challenges faced in the security patching process.

3. A classification of the current solutions in terms of approaches, tools, and practices to address the challenges and mapping of the challenges to the proposed solutions.

4. An analysis of the proposed solutions' rigour of evaluation and industrial relevance to inform and support the transferability of research outcomes to an industrial environment.

5. Identification of the potential gaps for future research highlighting important and practical concerns in security patch management that require further attention.

Chapter Organisation: Section 3.2 describes the background and other reviews related to the topic. Section 3.3 describes the research methodology used for this SLR. Sections 3.4, 3.5, 3.6 and 3.7 present the findings of the research questions. In Section 3.8, we discuss some key future research opportunities and threats to validity are presented in Section 3.9. Finally, Section 3.10 concludes the review.

3.2 Background and Related Work

In this section, we present an overview of the software security patch management process. Then, we provide a comparison of our study with the existing related reviews.

3.2.1 Overview of the Software Security Patch Management Process

Given there is no commonly known/accepted definition of software security patch management, we decided to devise an operational definition for our research based on our evidence-based understanding from the longitudinal field study reported in Section 2.2 [56] and the existing related literature [1, 17, 24, 57, 58, 59, 60, 61]. Hence, our operational definition of software security patch management is below:

Software security patch management is a multifaceted process of identifying, acquiring, testing, installing, and verifying security patches for software products and systems.

Software security patch management, commonly referred to as security patch management, is a security practice designed to proactively prevent the exploitation of security vulnerabilities that exist within an organisation's deployed software products and systems [1, 59]. Software security patches are *"pieces of code developed to address security problems identified in software"* [1]. In general, software security patches are always prioritised over non-security patches by industry practitioners and researchers as they are aimed at mitigating software vulnerabilities (or security bugs) that present exploitable opportunities for malicious entities to gain access to systems [1, 17]. In addition, software security patches are acknowledged as the most effective strategy to mitigate software vulnerabilities [17, 62]. A successful security patch management process is thus essential and critical to sustaining the confidentiality, integrity, and availability of IT systems [1]. Figure 3.1(a) shows the focus of security patch management from a typical software vulnerability life cycle perspective. For example, the focus here relates to the process of a company (company A) applying security patches to its deployed third-party software after the patches are released by the corresponding third-party software vendors (company B).

Despite the importance, security patch management remains one of the most challenging efforts facing IT practitioners. Figure 3.1(b) illustrates the five main phases of the security patch management process [2, 17, 20]. Firstly, in the *patch information retrieval* phase, practitioners learn about new patches and acquire them from third-party software vendors like Microsoft. In the next phase of *vulnerability scanning, assessment and prioritisation*, the practitioners scan the managed software systems for newly disclosed vulnerabilities to identify the applicability of patches in their organisational context, assess the risk, and correspondingly prioritise the patching decisions. Followed by the *patch testing* phase whereby the patches are tested for accuracy and stability and prepared for installation by changing machine configurations, resolving patch interdependencies and making backups. Then the patches are installed at their target machines in the *patch deployment* phase. Finally, the patch deployments are verified through monitoring (for unexpected service interruptions) and post-deployment issues are handled in the *post-deployment patch verification* phase.

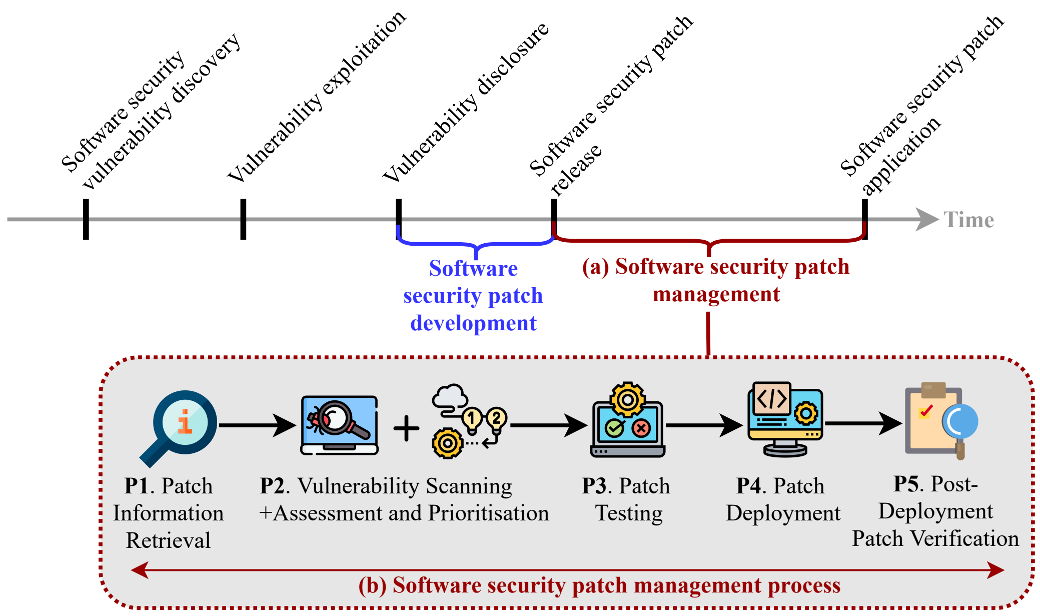

Software security vulnerability discovery
Vulnerability exploitation
Vulnerability disclosure
Software security patch release
Software security patch application
Time
Software security patch development
(a) Software security patch management
P1. Patch Information Retrieval
P2. Vulnerability Scanning +Assessment and Prioritisation
P3. Patch Testing
P4. Patch Deployment
P5. Post-Deployment Patch Verification
(b) Software security patch management process

FIGURE 3.1. (a) The focus of software security patch management in the software vulnerability life cycle. (b) An overview of the software security patch management process.

3.2.2 Other Reviews Related to Security Patch Management

Despite the increasing demand and growing body of literature on the topic of security patch management, we did not find any existing systematic literature review or systematic mapping study focused on the security patch management process. However, there have been several existing reviews/surveys on security patch management-related topics (Table 3.1), for example, international standards on patch management and dynamic software updating (DSU) [63], i.e., a method that allows for runtime patching without restarts or downtime. A comparison between these existing reviews/surveys on related topics and our SLR is presented below.

Recently, Gentile and Serio [21] reviewed a set of existing international standards on patch management and current industry best practices, assessing their relevance to the context of complex and critical infrastructures, particularly the industrial control systems (ICSs). Based on the survey results, they defined a general-purpose workflow to support the patch management process in the ICSs. While our study provides a set of practices for successful security patch management similar to this survey study, we also include a set of recommendations, guidelines, lessons learned and shared experiences of researchers and industry practitioners providing more coverage. However, the main difference between the study conducted by Gentile and Serio [21] and our study is the research focus, i.e., our SLR focuses on the existing challenges and solutions in security patch management which is not covered in the study by Gentile and Serio [21] that is limited to ICS context.

The other set of reviews and mapping studies exclusively focuses on dynamic software updating (DSU) [63]. DSU aims at live patching to avoid restarts or downtime that cause service interruptions. As such, the contributions of these studies focus on facilitating one phase in the security patch management process, namely the patch deployment phase (Figure 3.1(b)). For example, Miedes et al. [64] in their technical report surveyed and classified the common dynamic update mechanisms providing

Table 3.1. Comparison of contributions between our study and the existing related reviews/surveys.

Study	**Study contribution**	**Focus on security patch management**	**Challenges**	**Solutions**	**Evaluation of solutions**
Gentile and Serio 2019 [21]	International standards and best practices for patch management of complex industrial control systems	Overall software patch management process	–	✓(ICS specific best practices)	–
Miedes and Munoz-Escoi 2012 [64]	A classification of the dynamic software update (DSU) mechanisms	Patch deployment phase	–	✓(DSU types)	–
Seifzadeh et al. 2013 [65]	A framework for the evaluation of dynamic updating features	Patch deployment phase	–	–	✓(DSU features)
Gregersen et al. 2013 [66]	A systematic mapping of DSU approaches, tools, models, and techniques	Patch deployment phase	–	–	✓(DSU features)
Mugarza et al. 2018 [67]	An analysis of existing DSU techniques for industrial control systems	Patch deployment phase	–	✓(safety-compliant)	–
Ahmed et al. 2020 [68]	A framework for the evaluation of dynamic updating features	Patch deployment phase	–	✓(DSU approaches, tools)	–
Islam et al. 2023 [69]	A review of state-of-the-art literature on runtime patching approaches	Patch deployment phase	–	✓(DSU approaches)	–
This study	A SLR on the socio-technical aspects of software security patch management	Overall software security patch management process	✓	✓	✓

an overview of the concepts and techniques of DSU in the literature. Subsequently, several studies [65, 66, 67, 68, 69] followed the trend of reviewing the state-of-the-art of DSU techniques such as the review by Seifzadeh et al. [65] in which they provided a framework for evaluating the DSU features. Moreover, they highlight the need for future research investigating the challenges of adopting runtime patching in organisations that have been investigated in our study. In summary, our review differs from the existing studies by contributing to the gap area of a lack of a systematic review that identifies and analyses the challenges and solutions in security patch management.

3.3 Research Methodology

As described in Chapter 2, we used Systematic Literature Review (SLR) to gain solid background knowledge on socio-technical aspects of security patch management. We conducted this SLR by following the guidelines proposed by Kitchenham and Charters [29]. This section provides the details of the process illustrated in Figure 3.2.

To summarise the execution process of this SLR, after all the researchers discussed and agreed on the research questions, the present author developed a review protocol. All researchers were involved in the search string construction following several pilot searches and multiple rounds of discussions. The study selection was jointly done

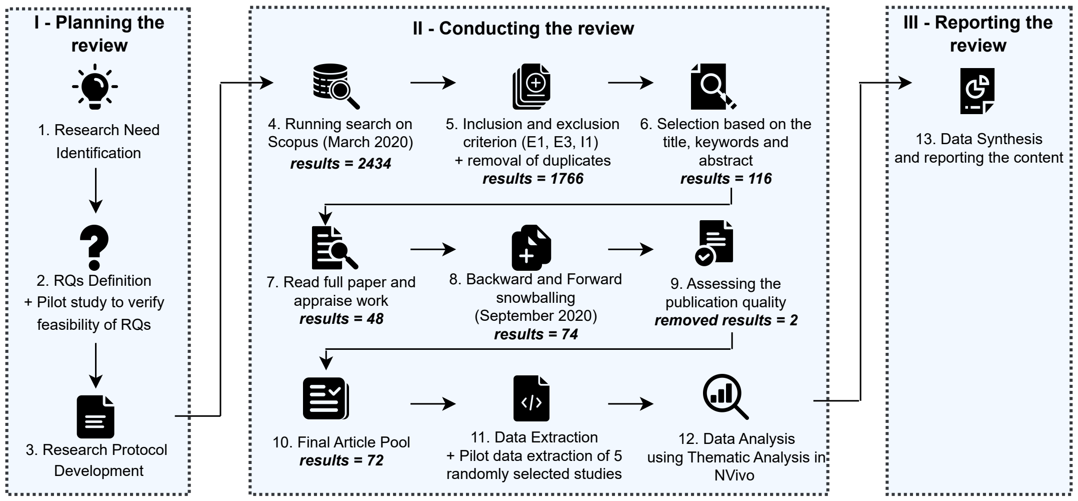

I - Planning the review
1. Research Need Identification
2. RQs Definition + Pilot study to verify feasibility of RQs
3. Research Protocol Development
II - Conducting the review
4. Running search on Scopus (March 2020) results = 2434
5. Inclusion and exclusion criterion (E1, E3, I1) + removal of duplicates results = 1766
6. Selection based on the title, keywords and abstract results = 116
7. Read full paper and appraise work results = 48
8. Backward and Forward snowballing (September 2020) results = 74
9. Assessing the publication quality removed results = 2
10. Final Article Pool results = 72
11. Data Extraction + Pilot data extraction of 5 randomly selected studies
12. Data Analysis using Thematic Analysis in NVivo
III - Reporting the review
13. Data Synthesis and reporting the content

FIGURE 3.2. An overview of the research methodology.

by the present author and other researchers. The present author conducted a pilot data extraction (DE) where the DE form and quality assessment (QA) form were reviewed by other researchers. Then, the present author performed data extraction, quality assessment and data synthesis under the close supervision of other researchers who are experienced in conducting SLRs in SE. The data extraction and synthesis of results were regularly discussed and verified by all in weekly meetings throughout the process of 8 months.

3.3.1 Research Questions

This SLR is aimed at providing an overview of state-of-the-art of security patch management. We formulated three research questions (RQs) to guide this SLR. Table 3.2 presents the RQs, along with their motivations.

The answers to these RQs will provide an in-depth understanding of the socio-technical challenges in security patch management (RQ1.1), available solutions (RQ1.2) and how the solutions have been evaluated (RQ1.3). The findings will enable researchers to identify gaps in this domain and potential future directions. It should be noted that we present solutions (RQ1.2) in two categories namely *approaches and tools*, and *practices*. We followed a similar strategy to Shahin et al. [70] in distinguishing between approaches and tools, and practices. The Cambridge dictionary defines an approach, method, and technique as *"a particular way of doing something or handling a problem"*; a tool as *"something that helps in a particular activity"*; and practice as *"the act of doing something regularly or repeatedly"* [71]. In this review, we define an approach, along with framework, method, technique, and tool as *a technical approach for addressing problems in security patch management*, and classify them in the category of ***"approaches and tools"***, for ease of reference. It should be noted that we categorised the studies that provided a comparison overview of the existing tools also under *"tools"*. Some studies reported more than one type of solution, hence those studies were included in more than one category. Practices, on the other hand, are defined as social practices and shared standards that can be supported by an approach or tool to facilitate a process [72, 73]. We classified the recommendations, guidelines, best practices, lessons learned and shared experiences as ***"practices"*** in this review study.

Table 3.2. Research questions of the SLR and their motivations

Research Question (RQ)	Motivation
RQ1.1: What socio-technical challenges have been reported in security patch management?	This RQ aims to understand the socio-technical challenges faced by practitioners in the security patch management process.
RQ1.2: What types of solutions have been proposed? RQ1.2.1. What approaches and tools have been proposed to facilitate security patch management? RQ1.2.2. What practices have been reported to successfully implement the security patch management process?	The motive of this question is to obtain a detailed understanding of the reported solutions in terms of approaches and tools (RQ1.2.1); and practices including industry experts' recommendations, guidelines, best practices, lessons learned and shared experiences for a successful security patch management process (RQ1.2.2).
RQ1.3: How have the solutions been assessed? RQ1.3.1. What types of evaluation have been used to assess the proposed solutions? RQ1.3.2. What is the level of rigour and industrial relevance of the reported solutions?	This RQ is aimed at analysing how the proposed solutions have been assessed. Since security patch management is highly industry-centric, identifying the types of evaluation used to assess the proposed solutions (RQ1.3.1); and understanding how well the solutions have been evaluated aligned with industrial relevance (RQ1.3.2) would help practitioners to adopt the solutions, and researchers to understand the gaps in the current evaluation approaches.

3.3.2 Search Strategy

We decided to use only the Scopus search engine to identify the relevant primary studies. The decision was based on the experiences reported by several other studies [70, 74, 75, 76] justifying that Scopus indexes a large majority of the journals and conference papers in software engineering indexed by many other search engines, including ACM Digital Library, IEEE Xplore, Science Direct, Wiley Online Library and SpringerLink. Furthermore, there are several restrictions placed by the other digital libraries (e.g. SpringerLink, Wiley Online Library, IEEE Xplore) on large-scale searches on the meta-data of the published studies. Additionally, the search string needs to be modified for every single digital library which can result in errors being introduced. Therefore, running the search string on Scopus enabled us to use one search string while retrieving mostly relevant hits. We performed the search using the studies' titles, abstracts, and keywords.

Initially, the search string was developed by selecting keywords based on the related literature and the reference lists from those relevant primary studies. Then, we systematically modified it by adding a set of alternative search terms obtained through synonyms and subject headings used in the existing related research papers. The identified search terms were merged using Boolean AND and OR to construct various combinations of search strings. Based on these search string combinations, we conducted several pilot searches to find the best search string and verify the inclusion of well-known primary studies. However, given that security patch management is still a new and emerging topic in research, we observed inconsistent use of different terminology in the literature. For example, one of the initial search terms was "patch management" but we decided to exclude it as a keyword since the inclusion of the term "management" resulted in returning a large number of irrelevant studies due to the inconsistent use of the term in the literature. A similar decision was followed for the keywords: *"update"* and *"socio-technical"*. Although these keywords were not included, the structure of the search string was capable of finding relevant papers, but we had to identify these papers through the study inclusion/exclusion phases. Furthermore, to mitigate the risk of missing potential primary studies from these decisions, we kept the search string as generic as possible and did not limit the search to any particular time range. Following the above strategies, we finalised the search string presented below:

TITLE-ABS-KEY *((‘software’ OR ‘system’) AND (‘patch*’) AND (‘security’ OR ‘vulnerabilit*’))*

3.3.3 Study Selection

We retrieved 2434 studies from the execution of the search string on Scopus on 31st of March 2020. We did not restrict our search based on publication year. We applied snowballing [77] to scan the references of the selected studies to find more potential studies. A backward and forward search ensured the extensiveness of our snowballing results and extended the coverage of included studies until September 2020.

We filtered the retrieved studies based on the inclusion and exclusion criteria presented in Table 3.3, initially defined when the review protocol was developed. Specifically, we included the studies related to the process of application of software security patches, that were in line with our SLR objectives and RQs. We refined the criteria during several iterations of search and study selection to ensure we achieve accurate classification of papers. For example, we did not include the short papers (E2) because they presented only concepts or ideas instead of well-defined, concrete approaches and did not provide sufficient and relevant evidence to answer the RQs. A similar strategy has been followed by several other SLR studies, for example, [70]. Similarly, we decided to introduce E3 during the pilot searches as we observed that those studies did not provide sufficient or useful data to extract based on the data extraction form. This decision was mutually agreed upon by all researchers after a careful review of the full text in several rounds of the study selection process. Correspondingly, the application of the inclusion and exclusion criteria and assessment of publication quality (Section 3.3.4) resulted in a final count of 72 primary studies to be included in the data extraction, as listed in Appendix E.

Table 3.3. Inclusion and exclusion criteria.

Inclusion Criteria
I1 Full text of peer-reviewed conference or journal article in English that is accessible.
I2 A study that relates to or addresses at least one phase of the security patch management process (i.e. the phases in Figure 3.1(b)).
Exclusion Criteria
E1 Workshop articles, books, and non-peer-reviewed papers such as editorials, position papers, keynotes, reviews, tutorials, and panel discussions.
E2 Short papers (i.e., less than 6 pages).
E3 A study that reports only numerical analysis, algorithms, mathematical techniques related to software security patch management
E4 A study that is only focused on hardware or firmware.
E5 A study that is not in the domain of security patch management (i.e. outside the focus area in Figure 3.1(a)).
E6 Full text is unavailable.

3.3.4 Assessing the Publication Quality

We assessed the quality of the reviewed primary studies with regard to their ability to help answer the RQs and the effect on the drawn conclusions [29]. We developed quality assessment criteria adopted and modified from a few published studies [78, 79, 80]. Table 3.4 provides a summary of the quality assessment. We graded the reviewed studies on each element of the quality assessment criteria using a three points ("Yes", "Partially" or "No") scale. We assigned the values: 2 to Yes, 1 to Partially, and 0 to No, to produce a quantifiable result. A paper was considered to be of acceptable quality and therefore included in the SLR if it received an average score ≥ 0.5. Two studies were excluded based on the quality assessment score. The present author performed the quality assessment while another researcher validated the results by independently performing the quality assessment of a smaller set of randomly selected studies. Any disagreements were sorted through discussions. The quality assessment was used to exclude studies with low quality and to indicate the credibility of the study's findings [78, 81].

Table 3.4. Assessment of the quality of publications.

Id	Quality Assessment Criteria	Yes	Partially	No
C1	Does the paper have clearly stated aims and objectives?	63(87.5%)	9(12.5%)	0(0.0%)
C2	Does the paper provide a clear context (e.g., industry or laboratory setting)?	54(75%)	13(18.1%)	5(6.9%)
C3	Does the paper have a research design that supports the aims?	51(70.8%)	21(29.2%)	0(0.0%)
C4	Does the paper explicitly discuss the limitations?	22(30.6%)	9(12.5%)	41(56.9%)
C5	Does the paper add value for research or practice of security patch management?	42(58.3%)	28(38.9%)	2(2.8%)
C6	Does the paper provide a clear statement of findings?	52(72.2%)	19(26.4%)	1(1.4%)

3.3.5 Data Extraction

We extracted data from the selected primary studies using a pre-defined data extraction (DE) form in an Excel spreadsheet as presented in Appendix F. The present author conducted a pilot DE on five randomly selected studies under the supervision of the other researchers and refined the DE form to capture all the required information in the best possible summarised version, through continuous discussions. We extracted some demographic information (e.g., authors name, venue published, and published year), and wrote critical summaries of the extracted data to be analysed and synthesised for answering the RQs.

3.3.6 Data Analysis and Synthesis

The demographic and contextual set of data items (D1 to D10 in Appendix F) were analysed using descriptive statistics while the other set of data items (D11 to D16 in F) was analysed using thematic analysis [82, 83], a widely used qualitative data analysis method. The decision to use thematic analysis was based on our effort to classify the reported socio-technical challenges and solutions in the domain. We used the following steps guided by Braun and Clarke's thematic analysis process [82] to synthesise the qualitatively gathered evidence.

Familiarising with data: First, we got familiarised with the extracted data by carefully reading each set of data. All data in the DE sheet were saved in the NVivo data analysis tool [38] and shared among all researchers.

Open coding: Open coding started with breaking the data into smaller components to generate the initial codes. A code (i.e., a phrase) of three-five words was assigned to summarise the key points of the data.

Building themes: Next, we assigned the codes to potential themes by iteratively revising and merging the codes based on their similarities using a multi-layered coding structure in NVivo.

Merging themes: Iteratively applying constant comparison on the codes and themes that emerged within one paper and between different papers, we grouped them to produce higher levels of themes. In the final step, we mapped the aggregated data to the security patch management process phases based on the literature [2, 20]. The synthesis of results for each RQ was carefully reviewed by all researchers in weekly meetings before finalising the answers to the RQs.

3.3.7 Overview of Selected Primary Studies

In this subsection, we report the findings of the descriptive analysis of the demographic and contextual set of data items extracted.

Demographic Data

Reporting demographic information in an SLR is considered useful for new researchers in that domain [70]. We present the demographic data of the distribution of the year and types of venues of the reviewed studies. Figure 3.3 presents the distribution of 72 primary studies over the years and the different types of venues. The selected studies were published from 2002 to 2020 as we did not find any relevant studies published before 2002. We found that 60% of the studies were published in conferences (43 of 72), while only 40% of the studies appeared in scientific journals.

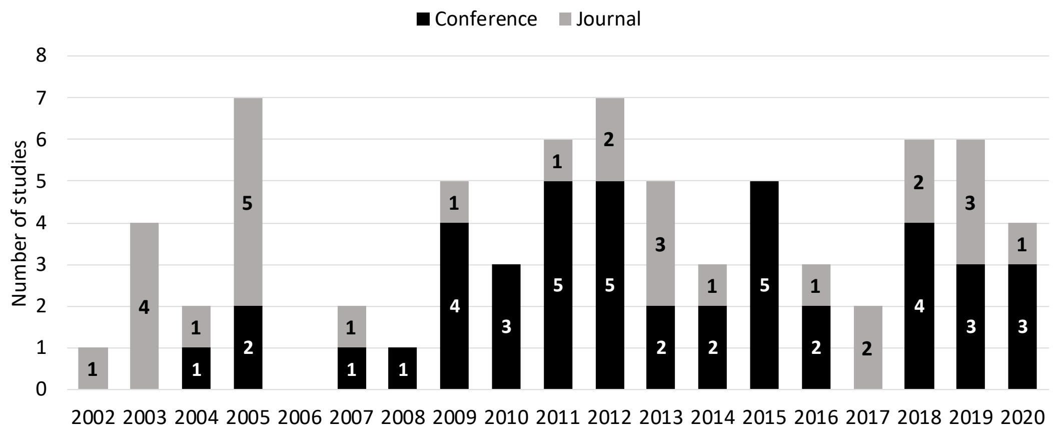


Figure 3.3. Distribution of studies over years and types of venues.

Studies Distribution in the Security Patch Management Process

We looked at the distribution of the reviewed studies mapped onto the security patch management process discussed in Section 3.2. Figure 3.4 reveals that a majority of the studies (38.9%) focus on vulnerability scanning, assessment and prioritisation. Patch information retrieval, patch testing and post-deployment patch verification have received the least attention from the reviewed studies, with only 5 of 72 studies (6.9%) focusing on those particular phases of the process. Twenty studies (27.8%) focus on more than one phase of the process, which we classified under the overall process.

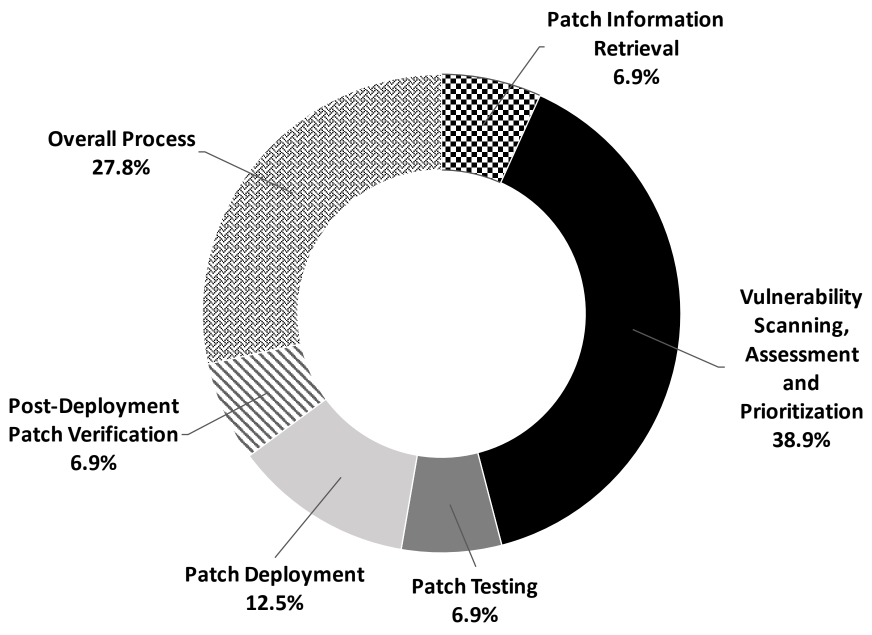


FIGURE 3.4. Distribution of studies over the security patch management process.

Research Type

We analysed the reported studies' research types and classified them into four categories, as illustrated in Figure 3.5, based on the classification proposed by Petersen et al. [84]. It should be noted that in Figure 3.5, the solution types are depicted in green colour, while the research types are shown in blue colour. Furthermore, the size and colour intensity of the circles refer to the number of solution types or research types per software security patch management process phase. A majority of the studies (44, 61.1%) reported *validation research*, in which the dominant research methods consisted of simulation, laboratory experiments, mathematical analysis and prototyping [85]. Thirteen studies (18.1%) reported *solution proposals* while only 10 studies (13.9%) reported *evaluation research* which consisted of strong empirical research methods such as industrial case studies, controlled experiments with practitioners, practitioner-targeted surveys and interviews [85]. The least reported were 9 *experience papers* (12.5%) that included industrial experience reports. We did not find any studies related to the *philosophical paper* and *opinion paper* categories. The lack of evaluation research and experience papers indicates a large need for research aligned with industrial relevance in security patch management.

3.4 Socio-technical Challenges in Security Patch Management

This section presents the findings for RQ1.1, the socio-technical challenges in security patch management. Our analysis resulted in the identification of 14 challenges as shown in Table 3.5. We have classified the challenges that are common across all phases of the security patch management process as "common challenges" and others as specific to each phase of the process.

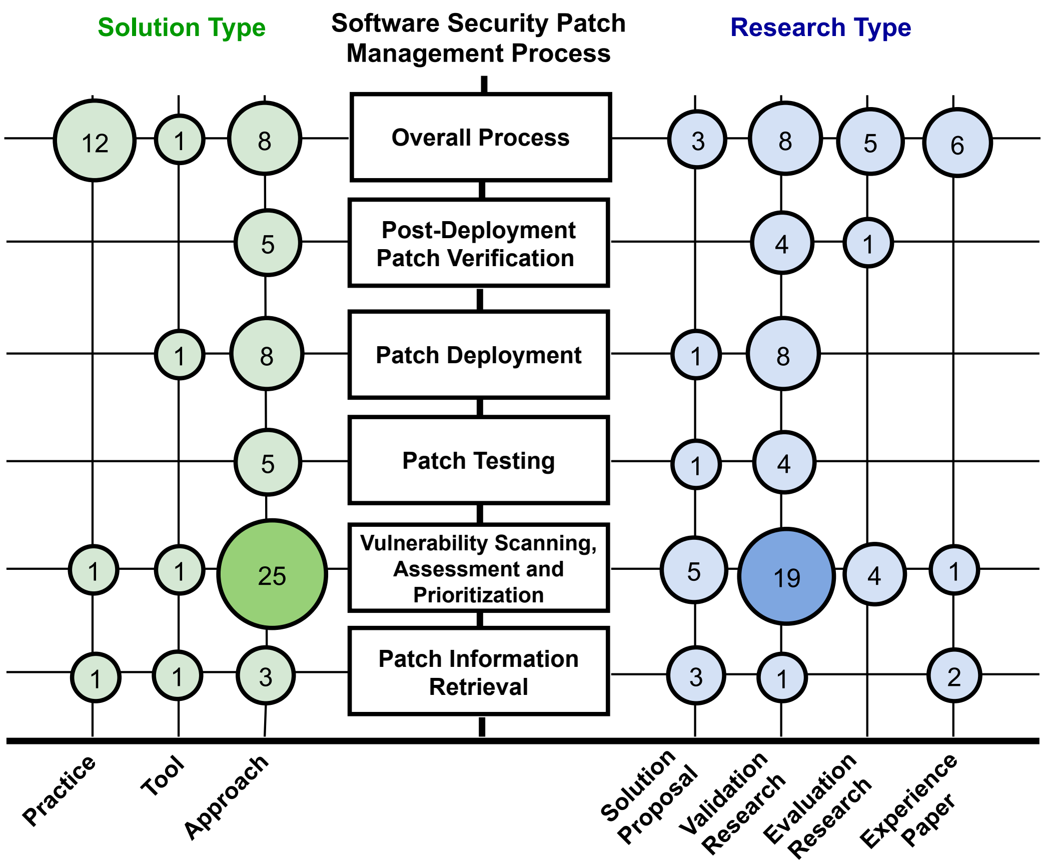


FIGURE 3.5. Mapping of the research types and solution types with the security patch management process.

TABLE 3.5. Socio-technical challenges in security patch management.

Relevant Patch Management Phase	**Challenges**	**Key Points (Included Papers)**	#
Common Challenges	**Ch1:** Collaboration, coordination and communication challenges	• Administrative overhead of coordinating with several stakeholders of conflicting interests [P4, P8, P11, P12, P30, P37, P63, P71] • Delegation issues due to lack of accountability and well-defined roles and responsibilities [P20, P29, P30, P40] • Communication challenges with multiple stakeholders with conflict of interests [P10, P15, P30, P34] • Lack of collaboration among several stakeholders [P20]	14
	Ch2: Impact of organisational policies/compliance	• Need to balance between complying with organisational policies and enforcing security [P8, P10, P20, P30, P71]	5

Relevant Patch Management Phase	Challenges	Key Points (Included Papers)	#
	Ch3: Complexity of patches	• Diversity of patches (heterogeneity) [P15, P46, P49] • Increasing rate of patch release [P1, P10, P29, P33, P35, P36, P40, P41, P44, P46, P55, P57, P62, P63, P66, P71] • Large attack surface (large and distributed organisation structure) [P38, P42, P43]	21
	Ch4: Limitations of existing tools	• Lack of standardisation in heterogeneous tools [P10, P11, P18, P19, P25, P53, P64, P66, P71] • Cost [P11, P19, P24, P25, P28, P36] • Time-consuming [P23, P29, P36, P43] • Lack of accuracy [P44, P45, P49, P51, P53, P67, P68, P69, P70] • Lack of security [P1, P24, P30, P46, P47, P55, P56, P72] • Lack of usability [P1, P25, P38, P45, P58, P70] • Lack of scalability [P40, P54, P56, P58, P66, P68, P72]	33
	Ch5: Need of human expertise	• Difficulty to achieve full automation in the process [P1, P9, P10, P11, P14, P16, P22, P25, P35, P48, P60, P62, P69, P70, P71]	15
	Ch6: Lack of resources	• Lack of skills and expertise [P1, P10, P15, P30, P45, P50] • Lack of process guidelines [P2, P10, P14, P15, P20, P40, P69, P70] • Lack of process automation solutions [P10, P19, P25]	14
Patch Information Retrieval	**Ch7:** Lack of a central platform for information retrieval and filtering	• Lack of a unified platform for information retrieval [P8, P10, P15, P18, P53, P56, P65, P71] • Lack of automatic validation, filtering and classification according to organisational needs [P8, P15, P18, P53, P56, P71]	8
Vulnerability Scanning, Assessment and Prioritisation	**Ch8:** Lack of a complete scanning solution	• Lack of understanding of the system [P22, P26, P29, P37, P42, P48, P64, P65, P70] • Lack of support for configuration management (detection) [P29, P32] • Lack of knowledge of system inventories [P20, P48, P65]	10
	Ch9: Lack of support for dynamic environment context	• Inability to capture dynamic context-specific factors [P7, P13, P14, P32, P34, P41, P42, P49, P51, P55, P62] • Lack of unified powerful metrics that capture the contextual factors [P6, P33, P51]	13
	Ch10: The gap of knowledge of technical and business context	• Lack of knowledge of organisational business risk posture and technical risk [P34, P50, P52, P71]	4
Patch Testing	**Ch11:** Lack of proper automated test strategy	• Need for fully automated patch testing [P3, P8, P24, P59, P60, P68]	6

Relevant Patch Management Phase	Challenges	Key Points (Included Papers)	#
	Ch12: Poor test quality in manual testing techniques	• Difficulty dealing with patch dependencies [P4, P9, P10, P15, P16, P20, P60, P67, P68, P71, P72] • Manual testing is slow and delays the patch deployment [P8, P12, P59, P71] • Error-prone due to difficulty in exact replication of production state [P15, P29, P59, P60, P67]	15
Patch Deployment	**Ch13:** Failures and side effects due to deployment of patches	• Need for managing the risk of problematic patches, missing configuration, and timely deployment [P3, P8, P9, P10, P12, P15, P16, P37, P45, P59, P60, P65, P71] • Difficulty dealing with organisation constraints (system downtime) [P8, P10, P12, P17, P22, P25, P26, P30, P31, P37, P54, P58, P62, P63, P65, P66, P67, P71, P72]	27
Post-Deployment Patch Verification	**Ch14:** Lack of efficient automated post-deployment patch verification strategy	• Lack of an overview of patch state of the system [P24, P37, P48] • Issues with manual patch deployment verification: difficult, error-prone, time-consuming task [P5, P8, P12, P21, P48, P57]	8

3.4.1 Common Challenges

The security patch management process is a collaborative effort between multiple stakeholders such as internal teams including security managers, engineers and administrators, third-party software vendors like Microsoft, Adobe, and customers/end users. The conflicting interests and interdependencies between these parties (e.g., delays of patch release from third-party software vendors) make security patch management a challenging undertaking [P4, P20, P30, P37]. The *lack of collaboration, coordination and communication* between the involved stakeholders thus represents one of the main barriers in maintaining the security of the managed software systems [P4, P8, P11, P12, P30, P37, P63, P71]. Moreover, the *impact of organisational policies*, i.e., the need to balance between complying with heterogeneous organisational policies and maintaining software security is acknowledged as a key challenge in security patch management [P8, P10, P20, P30, P71]. This is because the policies set by the higher management (e.g., the minimum service interruptions policy) sometimes contradict the timely application of emergency security patches [P20].

The rapid increase in the number and diversity of attacks has resulted in an increased rate of patch release causing a nightmare situation for practitioners to handle the increasing *complexity of patches* [P1, P33, P35, P40, P41, P44, P46, P55]. Another contributor to the increased complexity of patches is the large and distributed attack vectors (i.e., the use of diverse software systems and products) in organisational environments [P38, P42, P43]. In addition, the *limitations of the existing tools* have been noted as a major hindrance to the achievement of goals of security patch management. Of them, some prominent limitations include the lack of a standard platform to integrate the heterogeneous tools used for patch management [P10, P11, P18, P19, P25, P53, P64, P66, P71], the lack of accuracy (e.g., the current tools failing to take into consideration the dynamic organisation context resulting in erroneous output [P44, P45, P49, P51, P53, P67, P68, P69, P70]), the lack of security [P1, P24, P30, P46, P47, P55, P56, P72], and the lack of scalability in the design/architecture of tools that create difficulties in applying patches to multiple systems with different operating systems.

Due to the increased complexity and dynamic nature of security patch management and the limitations of the current technologies used in patching, the *need for human expertise* is inevitable throughout the patching process [P1, P9, P10, P11, P14, P16, P22, P25, P35, P48, P60, P62, P69, P70, P71]. However, as a result of human involvement in patching tasks and decisions, the time to patch has increased leaving several attack vectors open to exploits [P1]. The risk of delays is further increased due to a *lack of resources* in terms of skills and knowledge expertise, process guidelines and process automation support. An important point highlighted in the literature [P4, P29] regarding the lack of process automation support is that most of the existing solutions only focus on patch deployment, but do not provide solutions covering the entire process. Furthermore, several studies [P1, P19, P25, P30, P45, P50] have reported a significant gap in the required skills and knowledge expertise in security patch management particularly due to the increased complexity of patches.

3.4.2 Patch Information Retrieval Related Challenges

Practitioners are forced to spend hours monitoring multiple information sources due to the *lack of a central platform for patch information retrieval and filtering* [P18, P53, P56]. Li et al. [P8] reported that modern patch information sources range from security advisories (78%), official vendor notifications (71%), mailing lists (53%), online forums (52%), news (39%), blogs (38%) to social media (18%). Further, due to the rapid rate of patch releases, the lack of automated validation, filtering and classification of patch information according to organisational needs [P18, P53, P56] results in delayed patch application and increases the risk of a zero-day attack [P53].

3.4.3 Vulnerability Scanning, Assessment and Prioritisation Related Challenges

One of the prominent factors for the increased exposure to malicious attacks is the *lack of a complete scanning solution*. As a result, the practitioners fail to obtain a clear understanding of the system leading to missing the detection of software vulnerabilities [P22, P26, P29, P37, P42, P48, P64, P65, P70] and system misconfigurations [P29, P32]. Concerning the vulnerability assessment and prioritisation, the *lack of support for dynamic environment setting* presents a prominent challenge [P7, P13, P14, P32, P34, P41, P42, P49, P51, P55, P62]. The existing approaches are generally "one size fits all" that create difficulties in incorporating the needs of the organisational context and require a significant manual effort, particularly in virtual environment patching [P41, P42, P49, P51, P55]. A few studies [P6, P33, P51] have mentioned the need to have a common set of rigorous metrics with information such as exploit dates for accurate patch prioritisation since the existing vulnerability scanners depend on public vulnerability information which mostly includes only vulnerability disclosure dates. In addition, the *knowledge gap of technical and business context* (e.g., the need to apply security patches as soon as possible vs prioritisation of systems' availability) often results in patch prioritisation conflicts between different teams [P34, P50, P52, P71].

3.4.4 Patch Testing Related Challenges

One of the most pressing challenges in modern patch testing can be attributed to the *lack of a proper automated test strategy* [P3, P8, P24, P59, P60, P68]. A lack of automated testing may stem from different reasons such as the difficulty in dealing with issues of patch dependencies [P4, P9, P10, P15, P16, P20, P60, P67, P68, P71, P72] and the significant amount of human effort required for configuring a test environment to simulate a production-identical environment [P8, P59, P60]. However, interestingly, most of the current patch testing is done manually to avoid the risks of unexpected system breakdowns caused by faulty and malicious patches [P8, P59, P60, P68]. Nevertheless, the *poor test quality in manual patch testing* increases a system's vulnerability exposure as it often delays subsequent patch deployment [P8, P12, P59, P71]. Moreover, manual patch testing is largely error-prone due to the difficulty in the exact replication of a production state [P15, P29, P59, P60, P67].

3.4.5 Patch Deployment Related Challenges

One of the central challenges facing modern organisations in security patch management is the *failures and side effects from the deployment of patches.* This challenge emerges as a result of poor patch testing leading to faulty patches being deployed and missing requisites for deployment such as the configuration and dependency changes causing deployment errors [P3, P8, P9, P10, P12, P15, P16, P37, P45, P59, P60, P65, P71]. Such errors would subsequently lead to additional service downtime, hence, many practitioners often delay or refuse to install patches and keep using outdated software instead leaving known vulnerabilities readily exploitable [P3, P37, P59, P60]. The other major challenge relates to dealing with the organisational constraints of system downtime. The lack of a proper run-time patch deployment strategy coupled with the organisational policies to avoid system downtime presents a serious challenge for timely patch installation [P8, P10, P12, P17, P22, P25, P26, P30, P31, P37, P54, P58, P62, P63, P65, P66, P67, P71, P72]. This is particularly challenging in critical infrastructure system contexts such as healthcare for which downtime can create a significant adverse impact [P25, P30, P31, P37, P54, P58].

3.4.6 Post-deployment Patch Verification Related Challenges

Most of the existing security patch management solutions *lack an efficient automated post-deployment patch verification strategy* offering an overview of the system's patch state. This results in difficulties in detecting the problem location when an issue occurs following patch deployment [P24, P37, P48]. In addition, most current patch auditing methods are manual requiring practitioners to manually inspect the application for signs of an attack and repair the damage if an attack is found. This is a frustratingly difficult and time-consuming task with no guarantee of finding every intrusion and reverting all changes exploited by an attacker [P21, P48, P57]. The need for this verification to be done as quickly as the patch is deployed adds to the complex, effort-intensive and time-consuming manual verification highlighting the challenges of a lack of an efficient automated verification strategy [P5, P8, P12, P21, P48, P57].

3.5 Approaches and Tools Proposed to Facilitate Security Patch Management

This section presents the findings for RQ1.2.1, the solution approaches and tools proposed to facilitate the security patch management process. The analysis of the solution types has revealed that 75% of the proposed solutions are *approaches* while only 5.6% of the solutions are *tools*. Table 3.6 summarises the results for RQ1.2.1 presenting an overview of the key solution areas and the associated capabilities of the proposed approaches and tools, mapped onto the security patch management process.

TABLE 3.6. A classification of solution areas and the associated capabilities of the reported approaches and tools.

Relevant Patch Management Phase	**Solution Areas**	**Associated Capabilities (Included Papers)**	#
Patch Information Retrieval	**S1:** Patch Information Management	• Patch information retrieval from multiple sources (P18, P25, P28, P53, P56) • Information filtering based on organisational configuration needs (P18, P56) • Patch information validation (P18, P53, P56) • Patch download and distribution (P105, P56, P28)	5
Vulnerability Scanning, Assessment and Prioritisation	**S2:** Scanning for system vulnerabilities, potential attacks and ongoing attacks	• Central platform integrating the scan results from multiple sources (P22, P37, P41, P48, P51, P52, P64) • Detailed host-based analysis to identify assets resident on host (P48, P51, P42) • Detection of system misconfigurations (P32) • Guidance on scanning tool selection (P69, P70) • Identifying ongoing attacks (P22, P44) • Providing historical scanning analysis (P22, P34, P42)	27
	S3: Assessment and prioritisation of system vulnerabilities, potential attacks and ongoing attacks	• Providing a customisable, detailed and comprehensive analysis of vulnerability risks (P13, P26, P32, P34, P41, P43, P45, P49, P51, P52, P55) • Prediction of optimal fixing strategy for potential and ongoing attacks (P22, P26, P43, P55) • Measuring organisational vulnerability remediation effectiveness (P38, P43, P71) • Capturing the dynamic context for accurate assessment and prioritisation (P7, P14, P23, P32, P38, P41, P44, P49, P51, P52, P55, P62)	
Patch Testing	**S4:** Automated detection and recovery from faulty and malicious patches	• Automated detection of faulty patches (P9, P16, P59, P60, P68) • Automated detection of malicious patches (P24, P46, P47, P72) • Automated crash recovery of faulty patches (P3)	10

Relevant Patch Management Phase	**Solution Areas**	**Associated Capabilities (Included Papers)**	**#**
Patch Deployment	**S5:** Automated patch deployment	• Consideration of the dynamic context in patch deployment (P9, P11, P16, P27, P31, P36) • Reducing system downtime in reboots (P12, P17, P54, P58, P63, P67)	15
Post-deployment Patch Verification	**S6:** Automated patch monitoring and patch auditing	• Automated detection of exploits and patch deployment verification (P5, P9, P16, P24, P30, P37, P39, P48, P57, P72) • Automated repair of past exploits (P21, P36)	12

3.5.1 Patch Information Retrieval Related Solutions

A unified *patch information management* platform including the capabilities of patch information retrieval from multiple sources, information filtering, classification, validation, download and distribution of patches benefits practitioners with timely patch information retrieval to protect from zero-day vulnerability attacks [P53]. Such a platform reduces the administrative overhead of having to monitor multiple information sources for receiving up-to-date patch information while providing an easy way to obtain patch information with high accuracy [P18, P25, P28, P53, P56]. However, verifying the information is important because the information is obtained through various sources that may contain non-validated information, for example, Twitter [P18]. To achieve that, Trabelsi et al. [P53] report a trust and reputation system to verify patch information using the KPI trust model.

3.5.2 Vulnerability Scanning, Assessment and Prioritisation Related Solutions

One of the first and foremost steps in securing software systems is *scanning the systems to identify existing vulnerabilities and potential and ongoing attacks.* A set of studies have aimed at improving vulnerability scanning in different aspects. For example, a central platform that aggregates the scan results has been proposed to provide an overview of the systems' patch state [P22, P37, P41, P48, P51, P52, P64]. This would serve as a proactive environment facilitating the identification of vulnerabilities, and potential and ongoing attacks to assist practitioners with decision-making on the possible mitigation actions (e.g., applying patches, changing firewall rules, closing IP ports, etc.) [P22]. In an attempt to guide vulnerability scanning tool selection, Holm et al. [P69] find that there are significant differences in the accuracy of the scans of Windows and Linux hosts, through a comparative evaluation of seven tools used in the industry.

Following the identification of vulnerabilities, and potential and ongoing attacks in managed systems, *accurate risk assessment* is essential for prioritising critical patches to protect against attacks. Approaches of a customised and comprehensive analysis of vulnerability risks have been proposed in line with the industry standard, the Common Vulnerability Scoring System (CVSS) [86] using different vulnerability characteristics [P13, P26, P32, P34, P41, P43, P45, P49, P51, P52, P55]. In addition, new quantitative metrics have also been introduced to consider the context-specific risks [P51, P52], for example, patch time and patch discovery time to consider the risk window between patch deployment and vulnerability scanning [P51]. Measuring the patch impact and effectiveness of remediation actions are equally important as devising timely remediation strategies for strategic planning. For that, solutions with real-time feedback on the remediation delays and analysis of patch applicability have been proposed [P38, P43, P71]. Several attempts [P7, P14, P23, P32, P38, P41, P44, P49, P51, P52, P55, P62] have been made to address the challenge of a lack of support for dynamic environment settings in risk assessment, particularly in the cloud, as the standard CVSS algorithm does not take into account the cloud-specific settings [P23, P49]. For example, the solution by Lin et al. is to consider temporal and environmental metrics on top of the base score in the current CVSS algorithm.

3.5.3 Patch Testing Related Solutions

The necessity for rigorous patch testing emerges from the existence of faulty and malicious patches. To overcome the identified challenges of rapid patch release rates and poor test quality in manual patch testing techniques, approaches for *automated detection and recovery from such faulty and malicious patches* have been proposed [P9, P16, P59, P60, P68]. For example, Maurer and Brumley [P59] propose a tandem execution approach that immediately detects vulnerability exploits with no false positives. Few studies [P24, P72] have proposed using Blockchain to ensure the integrity of patches that are resilient to malicious attacks during patch distribution. Although several attempts have been made at detecting faulty and malicious patches, we found only one solution [P3] for surviving crashes that result from faulty patches. The proposed solution is based on multi-version execution and helps achieve minimal disruption to operations during a crash.

3.5.4 Patch Deployment Related Solutions

Several solutions have been proposed for *automating patch deployment* extending context-specific support for distributed and heterogeneous environments and reducing the patch deployment time, cost and overhead [P19, P25, P27, P31, P36, P40, P54, P58]. Additionally, reducing system downtime and reboots has been a priority of several studies [P12, P17, P54, P58, P63, P67] to address the critical challenge of minimising service interruptions when deploying patches. For example, Yamada et al. [P54] propose a virtual machine monitor (VMM)-based approach namely Shadow Reboot, to shorten the downtime and enable applications to run while rebooting. This approach can serve as a complementary solution to the existing dynamic software updating methods that usually require practitioners to have knowledge about the target kernels at the source code level [P54]. Alternatively, an approach that achieves minimal downtime through instant kernel updates without additional modifications to programs or state change tracking has been proposed by Kashyap et al. [P58]. They use an application checkpoint and restart (C/R) method to reduce the downtime to just three seconds.

3.5.5 Post-deployment Patch Verification Related Solutions

Several solutions have been suggested for *automating post-deployment patch verification tasks* [P5, P9, P16, P24, P30, P37, P39, P48, P57, P72] and *repairing past exploits* [P21, P36]. For example, “Pakiti" [P30] is a system that provides a central view of the patching status to help practitioners be informed and detect problems following patch deployment. Concerning the focus on automating the repair of past exploits, “Nuwa" [P36] is a tool that can automatically detect and repair patch deployment failures. It allows practitioners to retroactively patch vulnerabilities by automatically repairing the changes that have resulted from exploits while maintaining legitimate user changes [P21].

3.6 Practices Proposed to Successfully Implement Security Patch Management

This section presents the findings for RQ1.2.2, the practices to successfully implement the security patch management process. The classification of practices includes the reported recommendations, guidelines, best practices, lessons learned and shared experiences of researchers and industry practitioners. Similar to RQ1.1, we present the practices that are common to all phases of the security patch management process and those that are specific to each phase as shown in Table 3.7.

Table 3.7. Practices proposed for successful implementation of security patch management.

Relevant Patch Management Phase	**Practices**	**Key Points (Included Papers)**	**#**
Common Practices	**PR1:** Planning and documentation	• Document the life cycle of each vulnerability including reporting and tracking of remediation measures (P20, P66) • Review and update the process on a bi-yearly basis (P20, P61) • Time and dedication need to be given for proactive planning (P20)	3
	PR2: Establish formal policies and procedures into process activities	• Develop an appropriate mitigation strategy when no patch/workaround is supplied by the vendor (P20) • Have formal processes defined into the process covering all phases of the process (P2, P20, P50, P61, P65, P66, P72) • Measure the performance and effectiveness of the process (P2, P20) • Formalise procedures for dispute resolution (P2, P20)	8
	PR3: Define roles and responsibilities in the process	• Define the roles and responsibilities of groups and individuals involved in the process (P2, P20, P40, P61, P72) • Require stakeholders to take accountability (P20)	5
	PR4: Get management involvement and a clear understanding of the process	• Get senior management approval and involvement in process activities (P20, P40, P72) • Require clear understanding of the process for accurate decision-making (P8, P20, P61)	5
	PR5: Define procedures to facilitate efficient communication and collaboration	• Establish procedures to enable efficient communication and collaboration between stakeholders (P20, P61, P66) • Hold frequent patch meetings (P20) • Increase stakeholders' awareness of the process (P50) • Coordinate patch release schedules of different vendors (P4)	5

Relevant Patch Management Phase	**Practices**	**Key Points (Included Papers)**	**#**
Patch Information Retrieval	**PR6:** Establish policies and responsibilities for information retrieval, notification and dissemination	• Establish and maintain a list of the information resources (P2, P8, P19, P35, P65) • Maintain an upstream and downstream infrastructure for patch download and distribution to limit latency (P28) • Have proper patch information notification and dissemination policies in place (P35)	6
Vulnerability Scanning, Assessment and Prioritisation	**PR7:** Regularly monitor both active and inactive applications and security intelligence sources	• Regularly scan and monitor the network and vulnerability alerts (P2, P4, P20, P34, P40) • Establish a dedicated mailbox for vulnerability alerts that are sent via email (P2) • Close down unnecessary ports on network devices (P34) • Maintain historical scanning reports for future analysis (P34)	4
	PR8: Maintain up-to-date system inventory	• Create and maintain a system inventory including all the previous patches installed on every system (P20, P40, P61, P72) • Classify assets by platform hardware type, location and software application records, and develop risk potential for each asset (P2, P65, P66)	7
	PR9: Perform vulnerability assessment based on organisation needs and context	• Organisations should perform their own vulnerability assessment (P50, P61, P65) • Assess and respond to vulnerabilities on time (P2, P20, P50) • Consider historical scanning analysis in risk assessment (P34)	6
Patch Testing	**PR10:** Improve testing activity	• Prepare and store the test environment for manual system testing (P19) • Develop and test back-out procedure (P2, P20, P40)	4
Patch Deployment	**PR11:** Install patches on time balancing the security risks, resources and system availability	• Install timely patches balancing the need for security, resources, and time required to test a patch for system stability (P2, P20, P35, P40, P50, P65, P66, P72) • Facilitate automation as much as possible in the process (P35) • Investigate workarounds to reduce system reboots (P35, P50, P66) • Define a matrix for patch scheduling by patch severity and profile of managed systems (P19, P66)	9
Post-deployment Patch Verification	**PR12:** Keep track of the deployment status of every patch	• Regularly monitor the system's patch status to make sure every single patching job is executed successfully (P19, P40, P65, P66)	4

3.6.1 Common Practices

A *well-planned and structured process* is vital for successful security patch management. To define a solid patch management process, an organisation needs to give its time and dedication upfront [P20]. The process should *establish formal policies and procedures in the process activities* in all phases including documentation, communication, management reporting and auditing [P2, P20, P50, P61, P65, P66, P72]. For example, standard procedures should be in place for dispute resolution to handle conflicts and escalation paths for emergencies [P2, P20]. It is also important to measure the performance and effectiveness of the defined policies and procedures in the process and update them accordingly on a bi-yearly basis [P2, P20].

Since several internal and external stakeholders are involved in the process, having *well-defined roles and responsibilities* of individuals and groups helps reduce the administrative overhead of coordinating with multiple stakeholders and increases task accountability [P2, P20, P40, P61, P72]. According to Nicastro [P20], a local Computer Incident Response Team (CIRT) and Information Risk Managers (IRMs) are some of the roles that should be defined in every organisation. As such, adhering to a standard set of procedures, roles and responsibilities can help achieve *a clear understanding of the process among all stakeholders* and minimise conflicts [P8, P20, P61]. It is also important to have the *senior management actively involved and supporting the patch management decisions* to obtain organisational approvals without delays [P20, P40, P72]. Finally, *efficient communication and collaboration* between all stakeholders are vital for the smooth execution of the process [P20, P61, P66].

3.6.2 Patch Information Retrieval Related Practices

Developing formal policies and responsibilities for patch information retrieval, patch download and dissemination (e.g., creating and maintaining a list of information sources) has been reported as a useful practice to reduce the latency in information retrieval-related activities [P2, P8, P19, P28, P35, P65].

3.6.3 Vulnerability Scanning, Assessment and Prioritisation Related Practices

Concerning vulnerability scanning, *regularly monitoring both active and inactive applications and security intelligence sources* is important since there is a possibility of exploitation for applications that are not frequently used [P4]. Maintaining a history of the scanning reports [P34] is useful for analysing trends and making predictions of potential attack opportunities. Another best practice for vulnerability scanning is to *maintain an up-to-date system inventory* to increase the understanding and awareness of the system. With regards to the vulnerability assessment, *organisations should perform their own risk assessment based on the context* instead of solely relying on the vulnerability assessment scores from software vendors to get a more accurate vulnerability risk.

3.6.4 Patch Testing Related Practices

As identified in Section 3.4.4, most current patch testing is done manually to avoid the risks of unexpected system breakdowns from faulty and malicious patches. To *improve testing activities*, it is proposed to prepare the test environment including pre-configuration tasks and storage in advance to save time in testing [P19]. Although some practitioners avoid testing small patches due to the large overhead involved with patch testing, the authors in [P2, P20, P40] highlight the necessity for testing all security patches and developing and testing the back-out procedure to be deployed when required [P20].

3.6.5 Patch Deployment Related Practices

The *patches need to be installed on time while balancing the risks of time for proper patch testing and potential attacks while effectively managing the organisation's constraints* (e.g., service availability constraints). According to Marx et al. [P50], "a successful patch management process is capable of patching vulnerabilities in the shortest possible time frame while preventing the system downtime caused by an insufficiently tested patch". The path to achieving this balance is to have an appropriate risk-focused patch management process [P50] and proper patch scheduling (e.g., defining a matrix for scheduling patches based on the patch severity and its impact on the managed systems) [P19, P66].

3.6.6 Post-deployment Patch Verification Related Practices

Keeping track of the deployment status of every patch is useful to verify the deployment of patches, detect post-deployment issues early, and ensure the potential exploits during patch deployment are properly identified and repaired. To achieve this, it is suggested to regularly monitor a system's patch status and seek client feedback for any adverse impact on service continuity after every patching job [P19, P40, P65, P66].

3.7 Evaluation of the Reported Solutions in Security Patch Management

In this section, we report the results for RQ1.3, on how well the solutions have been assessed. We adopted the classification of evaluation approaches proposed by Chen et al. [87] presented in Table 3.8 to categorise the evaluation types used in the reviewed studies. We have slightly modified the adopted classification with two additions (i.e., *"SR - Simulation with real data"* and *"NE - No Evaluation"*) to make it more suitable for our review.

TABLE 3.8. The scheme for classification of the evaluation types.

Evaluation type	Definition
Field experiment	Controlled experiment performed in industry settings
Case study	An empirical study that investigates a contemporary phenomenon within its real-life context; i.e., studies involving industry practitioners [88]
Experience	The result has been used on real examples, but not in the form of case studies or controlled experiments, the evidence of its use is collected informally or formally. e.g., industrial experience reports
Simulation with artificial data	Execution of a system with artificial data, using a model of the real world
Simulation with real data	Execution of a system with real data, using a model of the real world performed in laboratory experiment
Laboratory experiment with software subjects	A laboratory experiment to compare the performance of the newly proposed system with other existing systems

Evaluation type	Definition
Laboratory experiment with human subjects	Identification of relationships between variables in a designed controlled environment using human subjects and quantitative techniques
Rigorous analysis	Rigorous derivation and proof, suited for formal model (i.e., statistical or mathematical verification)
Discussion	Provided some qualitative, textual, opinion-oriented evaluation. e.g., compare and contrast, oral discussion of advantages and disadvantages
Example application	Authors describing an application and provide an example to assist in the description for evaluation
No Evaluation	A study that reports no evaluation

3.7.1 Types of Evaluation Used to Assess the Proposed Solutions

The importance of rigorous evaluation to assess the appropriateness of the proposed solutions has been emphasised by the software engineering research community [87, 88, 89]. Accordingly, in this subsection, we present the results for RQ1.3.1, a distribution of the evaluation types used in the proposed solutions based on the solution type (i.e., approach/tool/practice) and the security patch management phase, as shown in Figure 3.6.

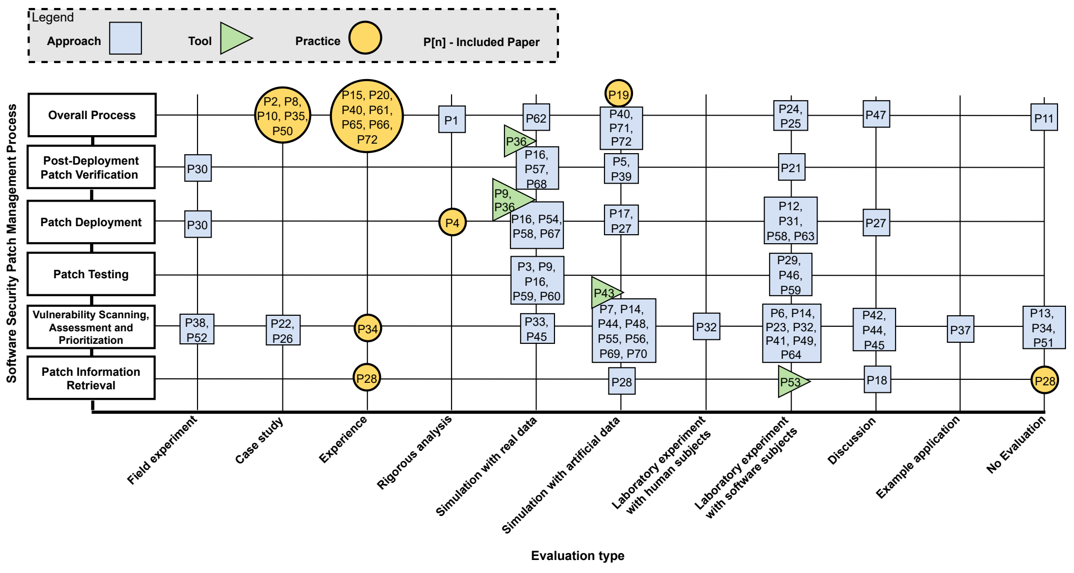


FIGURE 3.6. Mapping of the evaluation types based on the solution type and security patch management phase (symbol size based on the number of papers per solution type).

With regards to the evaluation types of the proposed *approaches*, *"Laboratory experiment with software subjects"* (30.36%) is the most frequently used type followed by *"Simulation with artificial data"* (28.57%). Interestingly, four studies [P11, P13, P34, P51] have reported *"No evaluation"* of their proposed approaches while evaluation types such as *"Case study"* (3.57%), *"Laboratory experiment with human subjects"* (1.79%) and *"Rigorous analysis"* (1.79%) have been rarely used. Concerning the *tools*, the majority (50%) have been evaluated using *"Simulation with real data"*. Regarding *practices*, *"Experience"* has been a widely used method for assessment (56.25%). Five studies (31.25%) have used *"Case study"* whereby the real-world insights have been captured through research methods such as industrial pilot projects [P2], practitioner-targeted surveys [P8, P10, P35, P50] and interviews [P8, P10]. One study by Nappa et al. [P4] proposes a set of recommendations for patch deployment based on a rigorous analysis of 1593 client-side vulnerabilities.

It should be noted that 11 of the reviewed studies (15.3%) [P14, P27, P28, P32, P34, P40, P44, P45, P58, P59, P72] have used two types of evaluation to assess their proposed solutions. For example, Xiao et al. [P44] used *"Simulation with artificial data"* to evaluate the robustness of the proposed approach against vulnerability exploits, and theoretical reasoning (i.e., *"Discussion"*) to demonstrate its practical utility for real-world monitoring of software vulnerabilities. Similarly, some studies have proposed an *approach* and a *tool* that address the challenges across multiple patch management phases (e.g., [P9] presents a method and a tool to analyse the patch impact and support patch deployment).

3.7.2 The Level of Rigour and Industrial Relevance of the Reported Solutions

The importance of providing practitioners with solutions to real problems and understanding how well the solutions have been evaluated cannot be overlooked in software engineering research [90]. Correspondingly, this subsection reports our attempts in assessing the level of rigour and industrial relevance of the evaluation types used for assessing the reported solutions as findings for RQ1.3.2. Of the evaluation types listed in Table 3.8, *"Field experiment"* is considered the most rigorous form of evaluation, followed by *"Case study"*. It is because both methods have the highest industrial relevance as the evaluation involves industry practitioners [85]. Similarly, evaluation based on *"Experience"* (e.g., industrial experience reports) also results in industry-relevant outcomes. By contrast, all other evaluation types are not acknowledged as rigorous forms of evaluation with proper industrial relevance. It is considered that *"Discussion"* and *"Example application"* evaluation types contain the least rigour and industrial relevance.

Since security patch management is a highly industry-oriented topic, employing evaluation types of *"Field experiment"* and *"Case study"* would produce solutions having higher industry adaptation and usefulness. However, a concerning finding is that only 15 solutions (20.8%) have used an evaluation type with industrial relevance. Of those 15 studies, seven solutions have been evaluated using *"Case study"*, three with *"Field experiment"* and the remaining five solutions using *"Experience"* (i.e., industrial experience reports). Another notable finding from this analysis is the lack of replication studies in the reviewed studies. According to Chen et al. [87], replication helps to provide solid and reliable evidence to support the adoption of a particular solution. We have found that 65 studies (90.3%) have evaluated their solutions in only one study indicating a general lack of replication. These findings reveal that the majority of the proposed solutions lack rigorous and industry-suitable evaluation, which is alarming given that the domain is highly industry-centric.

3.8 Discussion

In this section, we discuss the key findings from this SLR and the potential future research and development opportunities in this domain based on the key limitations and gaps identified through our study findings. We present a mapping of our findings in Figure 3.7 to enable the reader to quickly identify the relationships between the challenges (Section 3.4) and the proposed solutions (i.e., approaches and tools (Section 3.5), and practices (Section 3.6)) and the dependencies between them. An important observation is the dependencies exist only among the challenges and practices, and that they can be classified into two types, *dependencies among challenges that negatively affect or exacerbate another challenge* and *dependencies among practices that positively affect or support another practice*, as illustrated in Figure 3.7. For example, the *lack of a proper automated test strategy* exacerbates the issues with *poor test quality in manual testing techniques*. Consequently, it leads to faulty patches deployed causing *failures and side effects* during patch deployment. Alternatively, *establishing formal policies and procedures for process activities* helps *obtain a clear understanding of the process and obtain approval for security patch management decisions from the management without delays.*

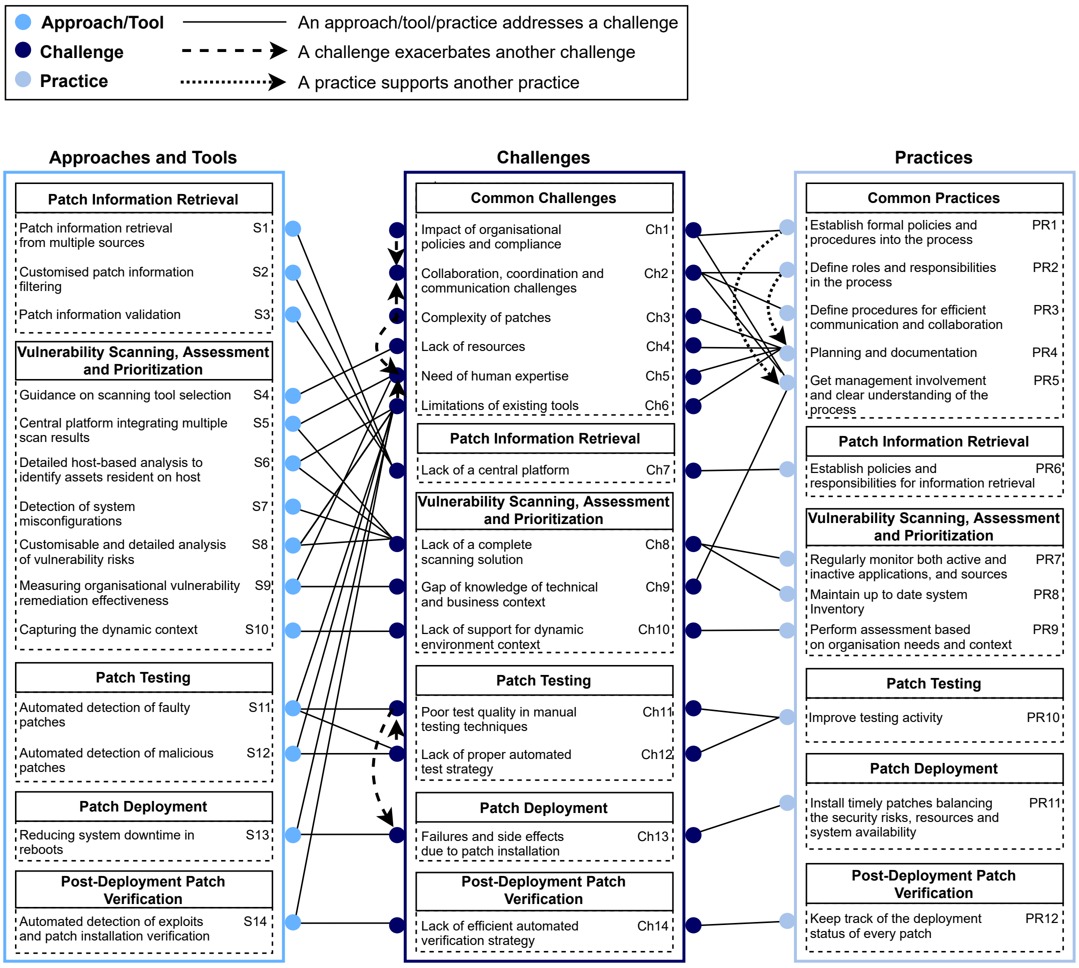

Approach/Tool
Challenge
Practice
An approach/tool/practice addresses a challenge
A challenge exacerbates another challenge
A practice supports another practice
Approaches and Tools
Patch Information Retrieval
Patch information retrieval from multiple sources S1
Customised patch information filtering S2
Patch information validation S3
Vulnerability Scanning, Assessment and Prioritization
Guidance on scanning tool selection S4
Central platform integrating multiple scan results S5
Detailed host-based analysis to identify assets resident on host S6
Detection of system misconfigurations S7
Customisable and detailed analysis of vulnerability risks S8
Measuring organisational vulnerability remediation effectiveness S9
Capturing the dynamic context S10
Patch Testing
Automated detection of faulty patches S11
Automated detection of malicious patches S12
Patch Deployment
Reducing system downtime in reboots S13
Post-Deployment Patch Verification
Automated detection of exploits and patch installation verification S14
Challenges
Common Challenges
Impact of organisational policies and compliance Ch1
Collaboration, coordination and communication challenges Ch2
Complexity of patches Ch3
Lack of resources Ch4
Need of human expertise Ch5
Limitations of existing tools Ch6
Patch Information Retrieval
Lack of a central platform Ch7
Vulnerability Scanning, Assessment and Prioritization
Lack of a complete scanning solution Ch8
Gap of knowledge of technical and business context Ch9
Lack of support for dynamic environment context Ch10
Patch Testing
Poor test quality in manual testing techniques Ch11
Lack of proper automated test strategy Ch12
Patch Deployment
Failures and side effects due to patch installation Ch13
Post-Deployment Patch Verification
Lack of efficient automated verification strategy Ch14
Practices
Common Practices
Establish formal policies and procedures into the process PR1
Define roles and responsibilities in the process PR2
Define procedures for efficient communication and collaboration PR3
Planning and documentation PR4
Get management involvement and clear understanding of the process PR5
Patch Information Retrieval
Establish policies and responsibilities for information retrieval PR6
Vulnerability Scanning, Assessment and Prioritization
Regularly monitor both active and inactive applications, and sources PR7
Maintain up to date system Inventory PR8
Perform assessment based on organisation needs and context PR9
Patch Testing
Improve testing activity PR10
Patch Deployment
Install timely patches balancing the security risks, resources and system availability PR11
Post-Deployment Patch Verification
Keep track of the deployment status of every patch PR12

FIGURE 3.7. A mapping of challenges onto solutions.

3.8.1 Need for More Investigation on the Less Explored Security Patch Management Phases

As shown in Figure 3.4, out of the primary studies selected, 28 (38.9%) have exclusively focused on proposing solutions to address the challenges in vulnerability scanning, assessment and prioritisation phase, 20 studies (27.8%) have focused on facilitating more than one security patch management process phase, and nine studies (12.5%) have proposed solutions to patch deployment phase. However, an important realisation from the analysis is the lack of attention paid to patch information retrieval, patch testing and post-deployment patch verification phases where only five studies (6.9%) have focused on each phase. The patch management process represents a tightly coupled sequence of phases where the output of one phase is input to the next phase. In addition, the dependencies among them exacerbate the challenges in the execution of tasks as identified in Figure 3.7 (e.g., [Ch12, Ch11] $\underrightarrow{\textit{exacerbate}}$ Ch13). Hence, there is a clear opportunity for valuable future research focusing on these less-explored phases of the patch management process.

3.8.2 Need for Evidence-Based Research on Socio-technical Aspects in Security Patch Management Delays

Security patch management is predominantly a socio-technical phenomenon where the human and technological issues/solutions are intertwined demanding exhaustive collaboration and coordination between human-human and human-technical systems interactions. The results in Section 3.4.1 have revealed that a lack of efficient collaboration, coordination and communication [Ch2] during the security patch management process can cause a significant negative influence on the timely vulnerability remediation. However, as shown in Figure 3.7, this important challenge has not received much attention in the reviewed studies. Based on our findings (see Figure 3.7), we argue that the patch interdependencies intrinsic to security patches resulting in increased complexity of patches [Ch3] can have a significant effect on other socio-technical factors like collaboration and coordination [Ch2]. Subsequently, this may result in additional struggles for organisations to apply timely patches. This apparent influence of socio-technical challenges on timely security patch management implies a need for more evidence-based research on the role and effects of socio-technical aspects on security patch management delays. Moreover, we believe such understanding would be useful for software developers to consider the socio-technical aspect integral to the patch application when developing patches. Additionally, future studies can invest effort into building tools and solution frameworks to address the socio-technical aspects causing patching delays, for example, a tool enabling better collaboration across patch management tasks and multiple stakeholders.

3.8.3 Human-AI Collaboration for Securing Software Systems

Over the years, several attempts have been made to integrate automation into security patch management tasks. However, an important realisation in the results presented in Section 3.4.1 is that there needs to be a delicate balance between human intervention and automation in security patch management. Automation enables practitioners to enjoy the benefits of less manual effort, while human expertise is required in the loop taking control of the decision-making and tasks that cannot be completely automated due to complexities of patches [Ch3] and limitations in the technology [Ch6]. Our review has unveiled that such decision-making points exist throughout the security patch management process. For example, according to P59, *"in an aim to automate the patch testing as much as possible, it is noted that human intervention is inevitable. As patches can change the semantics of a program, a human will likely always need to be in the loop to determine if the semantic changes are meaningful"*. As noted in Figure 3.7, only a few studies have addressed this challenge opening several possibilities for future research in *"Human-AI Collaboration"* [91], which is a new and emerging research paradigm aimed at exploring how human intelligence can be integrated with Artificial Intelligence (AI) systems to complement machine capabilities. Such systems aim at achieving complex tasks through hybrid intelligence to collectively achieve goals and progressively improve by learning from each other [92]. We believe that an intelligent system with a real-time, human-like, cognition-based framework can guide practitioners to decisions by providing timely information and asking logical questions, thereby assisting autonomous decision-making in security patch management [93].

3.8.4 Standardisation of Heterogeneous Tools

As identified in section 3.4.1, the diversity (heterogeneity) of patches increases patch complexity [Ch3]. This usually results in several challenges to practitioners from patch information retrieval to post-deployment patch verification. In addition, organisations use a multitude of software products (e.g., operating systems (OS), software applications, tools and platforms) increasing the challenges of patch heterogeneity. Several studies [P40, P54, P56, P58, P66, P68, P72] have reported the limitation of the lack of scalability in the existing tools. It was also observed that the majority of the reported solutions are only compatible with Linux, possibly due to the reasons it's being open source, easier to configure than other OSs and that patches applied to many Linux distributions result in only minor changes as opposed to patches of Windows [P69]. Hence, there is an increasing realisation of an orchestrated platform that caters to these heterogeneous tools. Future research could focus on designing and evaluating an architecture to support the standardisation of heterogeneous patch management tools which is dynamically adaptable to the organisational context and needs.

3.8.5 Real-world, Rigorous Evaluations

As revealed by the results in Section 3.7, a large number of studies lack rigorous evaluation using more mature forms of evaluation like field experiments and case studies. The low percentage of studies with industry-related evaluation highlights the need for researchers to work with practitioners to improve the state of the practice of rigorously evaluating research outcomes. We recommend that more attention be paid to rigorously evaluating the solutions using approaches with industrial relevance. The robust evaluation will improve the quality and transferability of the research outcomes to industrial adoption.

3.8.6 Contextual Factor

The importance of reporting contextual factors has been emphasised in the literature stating that software engineering research should investigate and understand their respective context [94, 95, 96]. In our review, we have tried to identify how the studies have reported the methodological and organisational context (i.e., research type, solution type). It should be noted that some studies had to be included in two categories since they have reported two research types. Based on the findings in Subsection 3.3.7 and Section 3.7, only 20.8% of the reviewed studies have provided industry-related evidence. The industry-related evidence enables researchers to understand the practical utility of the reported solutions and practitioners to adopt the proposed solutions in the literature. Given security patch management is a very industry-centric topic and complements the need highlighted by previous literature [e.g., P10], we recommend future research focus more attention on reporting the contextual factors as it helps to increase the credibility and quality of the research.

3.9 Threats to Validity

In this section, we report the validity threats of our study and the corresponding mitigation strategies following the guidelines proposed by [81, 97, 98].

3.9.1 Internal Validity

Bias in study selection (i.e., study filtering) and data extraction represent standard threats to all SLRs [98]. To address this, we defined a review protocol with explicit details about the search string construction, search process, study inclusion/exclusion and data extraction strategy [78, 97, 99]. Following a well-defined protocol helps achieve consistency in the study selection and data extraction, particularly, if multiple researchers are involved in the process [99]. We iteratively developed and improved the protocol, particularly the inclusion/exclusion criteria, after conducting a staged study selection process and pilot data extraction. Further, two researchers selected the studies while the other researchers cross-checked the outcomes and appropriateness of the selection criteria using randomly selected papers.

Concerning data extraction bias, we executed a pilot data extraction on a randomly selected sample of five studies to ensure the data extraction form captures all the required data to answer the RQs. We used a data extraction form (adapted from [29, 80]) which was reviewed by all researchers through the pilot data extraction. The present author extracted the data which was cross-checked by the other researchers for accuracy. Throughout the study selection and data extraction phases, weekly detailed discussions were held between all researchers to resolve any disagreements.

Additionally, publication bias is acknowledged as an internal validity threat which refers to the issue of the high likelihood of publishing positive results than negative ones [29]. However, we have reported the negative results captured in the primary studies (e.g., challenges in security patch management (RQ1.1)) and the challenges have been mapped against the reported solutions (RQ1.2), i.e., the positive results when identifying the gaps in Section 3.8 moderating the effect of unreported negative results. Further, using snowballing to increase the time and publication coverage has helped mitigate the publication bias of outcomes [97].

3.9.2 External Validity

Generalisability, referring to the likelihood of not being able to generalise the results, presents an important threat to overcome in SLRs. To address this, we conducted broad searches using one of the most well-known digital libraries (Scopus) to increase the identification of the related primary studies with broad time and publication coverage [97]. However, we acknowledge that our findings may not necessarily generalise to grey literature and studies outside the review period.

3.9.3 Construct Validity

We are unable to guarantee that we have captured all the relevant primary studies in our SLR. The possibility of missing primary studies is an inevitable limitation in an SLR due to limitations in the search string construction and selection of non-comprehensive digital libraries (DL) [78, 97]. However, to minimise the effects of this, we used several strategies which are described below.

We executed several pilot searches through which we systematically improved the search string to retrieve as many relevant papers as possible. An important point to note is that although the term "*software security patch management*" is widely used in the industry, this is still a new and emerging topic in research. Thus the use of inconsistent or different terminology in research papers, in particular, the term "*management*", resulted in a large number of irrelevant studies after its inclusion in the search string. Therefore, we have excluded it from the search string. Although this keyword was not included, the structure of the search string (i.e., broad and not time-bounding) was capable of finding patch management papers, but we had to identify these papers through the study inclusion/exclusion phases. In addition, we used snowballing (i.e., forward and backward search on references of the selected studies) to mitigate the threat of missing relevant primary studies from the exclusion of this term.

Regarding the selection of digital libraries, while using only Scopus to identify studies may present a limitation of this study, this decision has enabled to increase the coverage of the relevant studies since Scopus is considered the most comprehensive search engine among other digital libraries with the largest indexing system [70, 75]. We also did a pilot search on ACM Digital Library to compare and confirm the coverage of results from Scopus. To further mitigate this threat, we made our search string very broad by including the most common keywords to capture as many potentially relevant studies as possible.

3.9.4 Conclusion Validity

Researcher bias or the potential bias of researchers while interpreting or synthesising the data can impact the conclusions reached [97]. To reduce this impact, we adopted the recommended best practices for qualitative data analysis and research synthesis [97]. The present author led the data analysis and synthesis and the codebooks were shared with all researchers every week where the second and third researchers went through all the emergent codes, themes and synthesis results in detail. Disagreements between researchers were discussed in detail in weekly meetings until an agreement was reached between all researchers.

3.10 Chapter Summary

This chapter presents our research effort aimed at systematically reviewing and rigorously analysing the literature on security patch management. We have provided an organised evidential body of knowledge in the area by identifying and categorising the socio-technical challenges and available solutions, and analysing how well the reported solutions have been assessed with their level of rigour and industrial relevance. To the best of our knowledge, this SLR can be considered the first attempt toward systematically reviewing the literature on this topic. Based on a comprehensive analysis of 72 primary studies, we conclude:

- The review has enabled us to identify and classify 14 socio-technical challenges and available solutions including 6 themes of *approaches and tools*, and 12 *practices* as common ones affecting all phases of the security patch management process and those that are specific to each process phase. Moreover, the mapping of challenges onto solutions has revealed that 50% of the common challenges are not directly associated with solutions unveiling open research challenges for future work.

- The distribution of security patch management solutions is congregated around *vulnerability scanning, assessment and prioritisation*, with 37.5% of the reviewed studies focused on addressing the challenges related to this phase of the patch management process. In contrast, *patch information retrieval*, *patch testing* and *post-deployment patch verification* have received the least attention with only 5 studies (6.9%) each implying that there is a need for more studies investigating these under-researched phases.

- The findings have revealed that only 20.8% of the solutions have been rigorously evaluated in an industry setting using real-world representative evaluation approaches such as field experiments and case studies. These results indicate a large need for robust evaluation of solutions facilitating industrial relevance, and with more representative and diverse cases.
- With regard to the research type, a large majority of the reviewed studies (61.1%) have reported validation research. While only 10 studies (13.9%) have reported evaluation research, even fewer studies (12.5%) reported experience papers. The low numbers of evaluation research and experience reports reflect the scarcity of research with industrial relevance. Hence, there is the potential for future studies with active industry collaborations that will result in research outcomes having higher value addition and practical utility.
- Concerning the reported solution types, 75% are approaches, 19.4% are practices and 5.6% are tools. Further, a large number of limitations in the current tools (e.g., lack of accuracy, security and scalability) have been reported in the reviewed studies. Hence, research and development on new, advanced tools that address the limitations of current tools and support timely security patch management present a current need.
- Even though it has been reported that a significant number of challenges in the security patch management process emerge from socio-technical aspects such as coordination and collaboration, there is not much empirically known about the role (i.e., how and why) of such socio-technical aspects in the process leading to patching delays. Our findings have revealed that the socio-technical aspects have a wide-ranging effect across all phases of the process. Thus we recommend more focus on both evidence-based research and practice on socio-technical aspects of security patch management to explore their roles and impact on delays in applying security patches.
- Despite the widespread attempts to adopt full automation, we note that human-in-the-loop is inevitable in security patch management due to its inherent complexity and dynamic nature. Based on the findings, we recommend that the emerging research paradigm of *"Human-AI Collaboration"*, which explores how AI-based solutions can be developed to collaborate with human intelligence, presents an important future research opportunity in this topic.
- Finally, the mapping of challenges with solutions and the security patch management process will be beneficial for practitioners to easily understand what approaches, tools, and practices exist for facilitating each challenge. The classification of practices can serve as recommendations for guidance on the successful execution of the security patch management process. As a direct practical implication of the provided understanding, the security practitioners will be able to identify and assess the factors associated with timely security patch management and devise suitable decision-making to improve their organisational patching process.

Chapter 4

Why, How and Where of Delays in Security Patch Management

Related publication: This chapter is based on our paper titled *"Why, How and Where of Delays in Software Security Patch Management: An Empirical Investigation in the Healthcare Sector", published in the ACM on Human-Computer Interaction journal (CSCW), 2022 (CORE Ranking A) [100].*

Numerous security attacks that resulted in devastating consequences can be traced back to a delay in applying a security patch. However, as revealed in Chapter 3, despite the criticality of timely patch application, not much is known about why and how delays occur when applying security patches in practice, and how the delays can be mitigated. This chapter presents our attempts to address this important gap. Based on longitudinal data collected from 132 delayed patching tasks over a period of four years and observations of patch meetings involving eight teams from two organisations in the healthcare domain, and using quantitative and qualitative data analysis approaches, we identify a set of reasons relating to technology, people and organisation as key explanations that cause delays in patching. Our findings also reveal that the most prominent cause of delays is attributable to coordination delays in the security patch management process and a majority of delays occur during the patch deployment phase. Towards mitigating the delays, we describe a set of strategies employed by the studied practitioners. This research serves as the first step toward understanding the practical reasons for delays and possible mitigation strategies in vulnerability patch management. Our findings provide useful insights for practitioners to understand what and where improvement is needed in the patch management process and guide them towards taking timely actions against potential attacks. Also, our findings help researchers to invest effort into designing and developing computer-supported tools to better support a timely security patch management process.

4.1 Introduction

As described in Chapter 1, cyberattacks breaching corporate networks often result in catastrophic consequences ranging from exposure of sensitive and confidential data [101, 102] and betrayal of client trust to even human death [9]. Despite security patch management being the most effective remediation of this problem [17], it remains one of the most challenging endeavours due to the inherent technical and socio-technical interdependencies involved in the collaborative process of dealing with third-party vulnerabilities and vendor patches [2, 20]. As a result, organisations struggle to apply timely patches often leaving myriad vulnerabilities open to exploits. Consequently, it has resulted in most security attacks targeting known vulnerabilities for which a patch existed but delayed application. Despite the demonstrated criticality of timely patching, the recent statistics [23] reveal that the situation has still not improved indicating serious concerns and the increased importance of the efforts aimed at reducing delays in security patch management in practice.

While the previous studies that investigated the socio-technical aspects of security patch management have particularly focused on the process and its challenges [2, 20, 59], these studies have not exclusively focused on the delays in applying security patches. Further, another set of studies has attempted to optimise the patch management process by synchronising the organisational patch cycle with the vendor's patch release cycle [60, 61, 103]. An important observation was that although previous studies have focused on approaches to reduce delays in security patch management, to date, there has been no study that explores why and how delays continue to happen when applying security patches. It adds to the demonstrated critical need for investigating the delays in patching, grounded in evidence from practice as observed in Chapter 3. Motivated by the need and the background knowledge gained in Chapter 3, we decided to empirically investigate the delays in real-world security patch management from a socio-technical perspective. Figure 4.1 shows the focus of the study in the vulnerability timeline. Towards this goal, we conducted a longitudinal case study investigating the delays in the security patch management process. Our study was guided by the following key research questions (RQs):

RQ2.1. *Why, how, and where do delays occur in security patch management?*

RQ2.2. *How can the delays be mitigated?*

Based on qualitative and quantitative analysis of the longitudinal data gathered from patch meeting minutes spanning over four years from October 2016 to May 2021 between two organisations (Org A and B) as detailed in Section 2.2.2, we attempt to answer these crucial overarching questions of delays in security patch management. The findings explain the causes of delays with a taxonomy comprising technology, people and organisation-related reasons and describe which reasons are more prominent based on their frequency distribution, and where the delays occur in the security patch management process. This study also reports a classification of strategies applied in practice to mitigate the delays including when to apply them during the patch management process. To the best of our knowledge, this is the first study to provide an evidence-based understanding of the causes and strategies for delays in security patch management. Grounded in descriptive evidence from practice, our research contributes to the state-of-the-art understanding of research and practice in several ways:

1. Identifies a set of reasons for delays when applying security patches in practice.

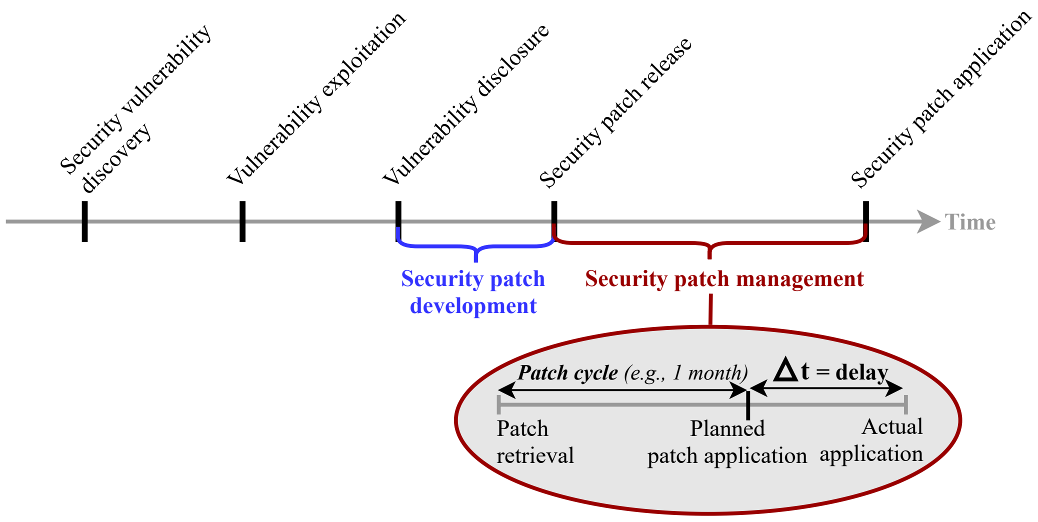

Security vulnerability discovery
Vulnerability exploitation
Vulnerability disclosure
Security patch release
Security patch application
Time
Security patch development
Security patch management
Patch cycle (e.g., 1 month)
Δt = delay
Patch retrieval
Planned patch application
Actual application

FIGURE 4.1. The focus of the study in the vulnerability timeline.

2. Describes the most prominent reasons for delays with rationales explaining their variations.
3. Reports where a majority of delays occur in the patch management process presenting their distribution over the process phases.
4. Presents a collection of strategies employed in practice to mitigate the delays including when to apply them in the security patch management process.
5. Structures the understanding of delays in vulnerability patch management, drawing attention to a critical yet less explored phenomenon in the software engineering community.
6. Grounded in practical evidence, the findings lay a foundation for future researchers and tool designers to design and develop computer-supported solutions to reduce delays in patch application.
7. Offers practical guidance for practitioners to identify what and where is improvement needed to mitigate patching delays and drive their decisions appropriately.

Chapter Organisation: Section 4.2 describes the related work. Section 4.3 reports the research methodology used. Section 4.4 presents the findings of the research questions. In Section 4.5, we discuss the findings and implications for research and practice, and threats to the validity are presented in Section 4.6. Finally, Section 4.7 summarises the chapter.

4.2 Related Work

Prior research has invested effort in improving the patch management process through both technical and socio-technical aspects. In the scope of technical enhancements, advancing automation in the security patch management process, for example, automated detection of faulty patches [25, 104, 105] and mechanisms for reducing system downtime in reboots [106, 107, 108], have been widely studied. However, the literature presents little empirical evidence of the socio-technical aspects relating to security patch management. Existing socio-technical studies have primarily focused on the workflows of system administrators but did not focus on other roles (e.g., change manager) and external stakeholders (e.g., customer) involved in the patch management process. Crameri et al. [25] were among the first to investigate system administrators' patch management practices. In a survey conducted with 50 system administrators, they reported that 70% of administrators avoided deploying patches due to issues caused by a lack of integration between patch testing, deployment and post-deployment issue reporting. Dietrich et al. [109] explored the system administrators' perspective on factors leading to security misconfigurations. Their findings confirmed that the situation has not changed even after a decade, reporting that delaying and avoiding security patches are among the most frequently reported security misconfigurations. However, these studies did not explain the reasons for such delays or missed patches.

Extending the study by Crameri et al., two recent studies [2, 20] have examined a larger sample of system administrators through a combination of surveys and interviews to perform a comprehensive investigation of the patch management process. Both studies explored system administrators' practices, behaviour, and experiences in the patch management process. According to them, administrators rely on various sources such as security advisories, direct vendor notifications, patch management tools, mailing lists and online forums to retrieve meaningful patch information. Further investigating system administrators' patch information retrieval-related needs and practices, Jenkins et al. [110] studied how the mailing list of the website PatchManagement.org extends support in patch management activities. They argue that the mailing list acts as an online community of practice extending support not only in the patch information retrieval phase but throughout the process in various aspects such as guidance for patch prioritisation, workarounds for post-deployment issues and tool selection.

Another set of studies [2, 20, 22, 54, 103, 106] has explored the challenges in the patch management process. For example, the impact of organisational policies and culture [2, 20, 59], collaboration and coordination challenges due to conflicts between stakeholders [2, 22, 103, 106], lack of resources in terms of skills and expertise required for handling complex patching tasks [20, 110, 111], and the increasing rate of patch release [20, 106, 111] are some of the most common challenges faced by practitioners. In addition, challenges relating to the lack of dedicated patch testing environments [2, 105, 112], post-deployment patch verification [2, 113], and system downtime during patch deployment [2, 20, 106, 107] have been widely discussed. Despite widespread attempts to the adoption of automation in different phases of the process, it is revealed that the need for human interaction still presents an inevitable challenge [2, 20, 54, 104, 111].

To address some of these pressing socio-technical challenges, several studies have proposed tools, frameworks and practices. For example, a set of studies [24, 60, 61] has proposed synchronising an organisation's patch cycle with the vendor's patch release cycle to optimise the process, minimise stakeholder conflicts and reduce costs. Although several approaches have been proposed to improve the patch management process to reduce delays, the reasons why such delays occur and how to mitigate them remain unexplored. Furthermore, given patch management is largely an industry-centric topic, relatively little has been done to understand the state of practice. For example, why do practitioners continue to delay applying the security patches leading to compromises that would have been easily prevented like the Equifax case [102]?

In contrast, delays have been widely studied in related fields like software development. In the majority of these studies [114, 115, 116, 117], the focus has been on delays in global software development (GSD). For example, they have explored the effect of distance on delays in a multi-site software development organisation and mechanisms to reduce delays. Closely related to our study but focused on software development projects is the empirical analysis conducted over a decade ago by Genuchten [39]. By analysing the planning data of six projects in one software development department of an organisation, he provided a classification of reasons for delays in software development activities. The findings report that capacity-related reasons cause the majority of the delays in the studied context. Further, the study highlights the importance of understanding the causes of delays for software developers to take necessary actions for improvement. Similarly, the recent expediting attacks targeting unpatched software security flaws exhibit a pressing need towards understanding the practical causes of delays in patch management and suitable strategies to reduce such delays, which is accomplished by this study.

4.3 Research Method

To understand why and how delays occur in practice, we conducted a longitudinal industrial case study with Org A and Org B involving 21 participants from 8 teams, which was described in Section 2.2.2 in Chapter 2. The findings of this chapter are based on the longitudinal data collected from patch meeting minutes maintained by the case organisations as the main artefact from patch meetings. In addition, we observed the patch meetings to supplement our understanding of the process, activities and strategies documented in the meeting minutes and verify the emerged findings from the artefact analysis.

4.4 Findings

In this section, we present the findings of our study. Figure 4.2 presents an overarching representation of the findings from the qualitative analysis. We provide examples from the patching tracker chosen based on their representativeness, as supporting evidence and to increase the verifiability of our findings [50]. In the examples, we include the subject of the task (see Figure 2.2) and evidence relating to the delay using unique identifiers for ease of reference, for example, *"P[n]-AT1"* refers to a participant from Org A's EMR team, and *"Win, Task ID 2"* refers to the 2nd task discussed in the Digital Health Windows meeting.

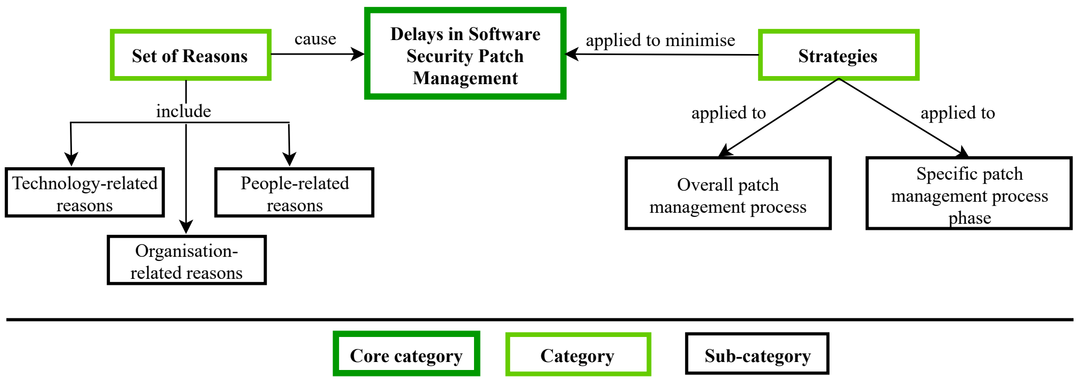

Set of Reasons
cause
Delays in Software Security Patch Management
applied to minimise
Strategies
include
Technology-related reasons
People-related reasons
Organisation-related reasons
applied to
applied to
Overall patch management process
Specific patch management process phase
Core category
Category
Sub-category

FIGURE 4.2. High-level overview of the findings from the qualitative data analysis.

4.4.1 Why, How, and Where do Delays Occur in Security Patch Management?

We identified a set of reasons that cause delays in security patch management, presented as a taxonomy in Figure 4.3. In summary, we found nine reasons, grouped into three main categories: technology-related reasons, people-related reasons and organisation-related reasons.

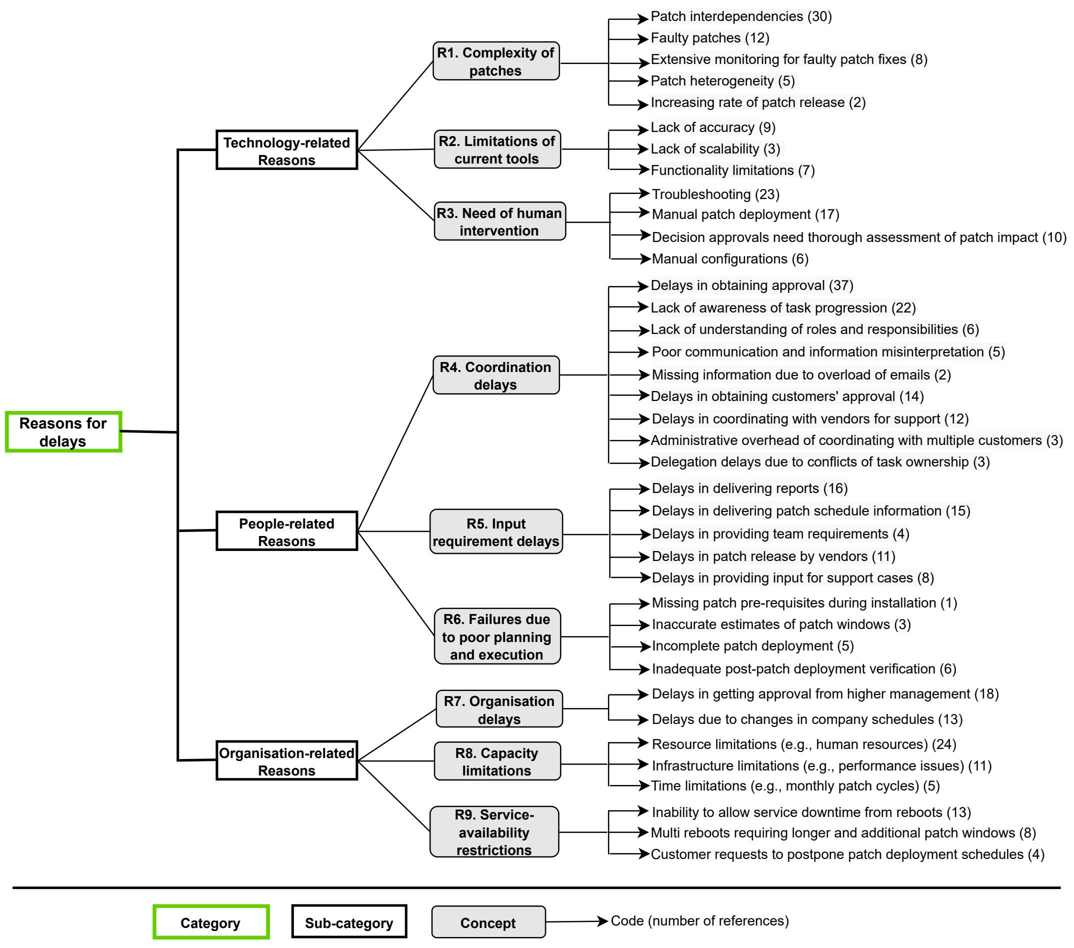

Reasons for delays
Technology-related Reasons
R1. Complexity of patches
Patch interdependencies (30)
Faulty patches (12)
Extensive monitoring for faulty patch fixes (8)
Patch heterogeneity (5)
Increasing rate of patch release (2)
R2. Limitations of current tools
Lack of accuracy (9)
Lack of scalability (3)
Functionality limitations (7)
R3. Need of human intervention
Troubleshooting (23)
Manual patch deployment (17)
Decision approvals need thorough assessment of patch impact (10)
Manual configurations (6)
People-related Reasons
R4. Coordination delays
Delays in obtaining approval (37)
Lack of awareness of task progression (22)
Lack of understanding of roles and responsibilities (6)
Poor communication and information misinterpretation (5)
Missing information due to overload of emails (2)
Delays in obtaining customers' approval (14)
Delays in coordinating with vendors for support (12)
Administrative overhead of coordinating with multiple customers (3)
Delegation delays due to conflicts of task ownership (3)
R5. Input requirement delays
Delays in delivering reports (16)
Delays in delivering patch schedule information (15)
Delays in providing team requirements (4)
Delays in patch release by vendors (11)
Delays in providing input for support cases (8)
R6. Failures due to poor planning and execution
Missing patch pre-requisites during installation (1)
Inaccurate estimates of patch windows (3)
Incomplete patch deployment (5)
Inadequate post-patch deployment verification (6)
Organisation-related Reasons
R7. Organisation delays
Delays in getting approval from higher management (18)
Delays due to changes in company schedules (13)
R8. Capacity limitations
Resource limitations (e.g., human resources) (24)
Infrastructure limitations (e.g., performance issues) (11)
Time limitations (e.g., monthly patch cycles) (5)
R9. Service-availability restrictions
Inability to allow service downtime from reboots (13)
Multi reboots requiring longer and additional patch windows (8)
Customer requests to postpone patch deployment schedules (4)
Category
Sub-category
Concept
Code (number of references)

FIGURE 4.3. Detailed overview of the causes of delays in security patch management.

Next, we quantitatively analysed the identified set of reasons to understand the *most prominent reasons* that need practitioners' and researchers' attention. To achieve this, we conducted a frequency analysis of the reasons for delays. An important observation was that in the majority of the delays, we found multiple reasons attributed to one delayed task. For example, a delay in applying a critical security patch was identified due to a combination of reasons such as delayed input by the vendor (R5), delays in coordination with the vendor (R4), and lack of expertise (R8). In total, we found 417 occurrences of the identified nine reasons ascribed to the 132 delayed tasks analysed. Figure 4.4 presents the frequency distribution of the reasons for delays. Accordingly, the most prominent causes for delays relate to people-related reasons, for example, delays in coordinating the patch management activities (24.9%) and providing input requirements (16.8%).

In determining *where the aforementioned delays occur in the patch management*

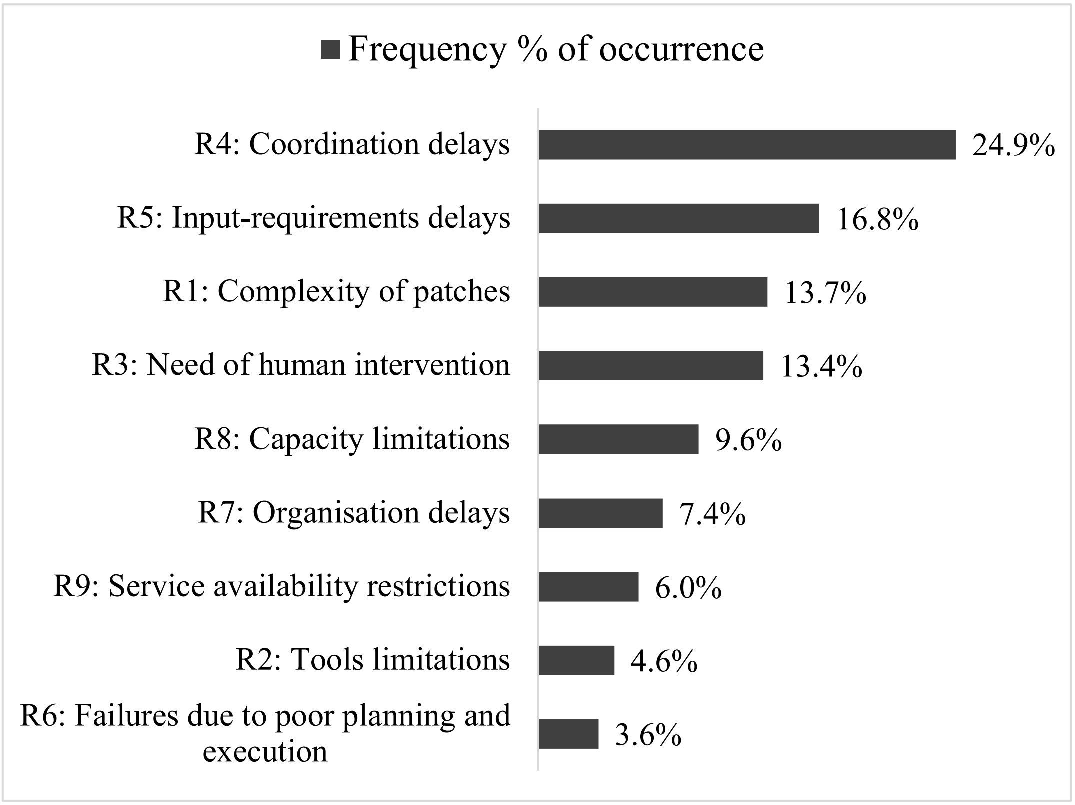


Figure 4.4. Frequency distribution of the reasons for delays in security patch management from a total of 417 occurrences of delayed reasons.

process, our quantitative data analysis revealed that the delays are distributed throughout the process with a majority of the delays, i.e., 54% occurring during the patch deployment (P4) phase as shown in Figure 4.5. We identify that it can be attributed to the inherent socio-technical complexities involved in the patch deployment tasks and decisions. The second-highest number of delays happen during patch testing (P3) and post-deployment patch verification (P5) phases where each account for 15% of the delays. Possible explanations of these numbers can be recognised by the evident challenges in the respective stages, for example, managing the delays occurring due to the poor quality of patches, which may result in unanticipated post-patching failures leading to disastrous consequences and inconvenience to users, e.g., unavailability of service. Additionally, we have reported the average delay duration in months in each process phase. As shown in Figure 4.5, the longest average delay is reported as 3.6 months belonging to patch testing and vulnerability scanning, assessment and prioritisation tasks. In the following, we describe the nine reasons for delays mentioned in *italic* under their corresponding main categories.

Technology-Related Reasons.

The technology-related reasons denote the compound characteristics intrinsic to software security patches, limitations of the tools used in patch management, and technological limitations resulting in the need for human intervention in the process.

Concerning the *complexity of patches*, the patch interdependencies consisting of software, hardware, and firmware presented a major reason for delays in patch testing

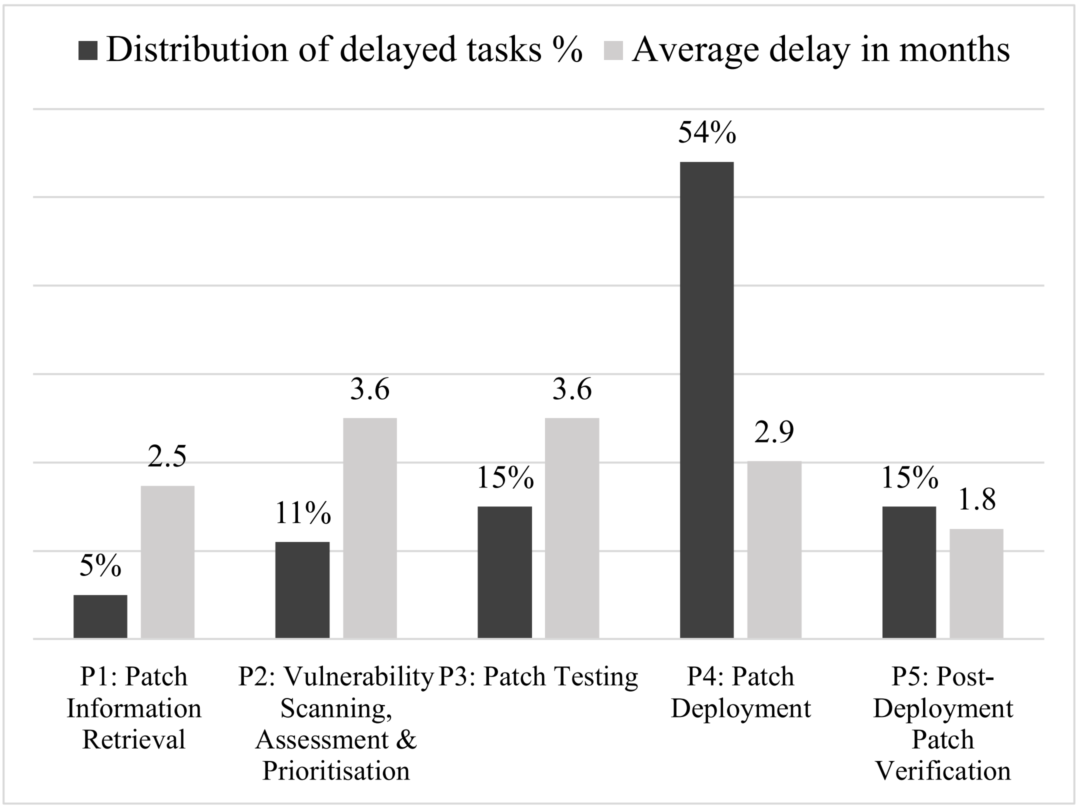


FIGURE 4.5. Distribution of delays over the security patch management process and average delay duration in months in each phase. Total number of delayed tasks = 132.

and deployment tasks. We identify that such complexities emerge from the existing dependencies in the source code, for example, function-level or library-level dependencies. Patching large and complex software systems involves a diverse set of operating systems, tools, and software applications with multiple versions. It introduces additional challenges to match the compatibility of several versions which often leads to delays during patch testing. Moreover, patch interdependencies with the legacy software were a recurrent cause of the delays in the studied context. This reason exacerbated problems with delays since the solutions, for example, upgrading or decommissioning the legacy system, or continuing to receive extended support (i.e., obtaining patches) from the vendors presented even further challenges. This is because, besides the large costs involved in these workaround solutions, the teams were faced with high risks as most of the legacy systems operated on critical medical services. In addition, due to the complex and business-critical nature of legacy systems, resolving legacy software dependencies often resulted in significant delays leading up to several months in some cases.

On the other hand, the unknown errors during patch testing, deployment, and post-deployment arising from faulty patches led to delays. In such instances, the practitioners spent a significant amount of time troubleshooting the error not knowing that it is caused by a faulty patch. Following the identification of the root cause as a faulty patch, the practitioners often pursued the vendor's support which further delayed the completion of the task. We also observed that some security patches required extensive monitoring to verify the fixes for post-deployment errors. For example, the task was kept under monitoring for several weeks until the results confirmed the applied fix poses no unanticipated adverse effects on the managed systems. Furthermore, the increasing rate of patch release coupled with the patch heterogeneity adds to the complexity of patches creating delays in patching. This is because as the number and diversity of patches increase, the number and complexity of the patch interdependencies that need to be managed also increase. Consequently, it leaves myriad attack vectors vulnerable to cyberattacks increasing the risk of exploits.

> [Subject - Patch deployment error at the [server s1]
> *"13/12/19 - Workaround applied and timings were all good. Keep open till January run for confirmation."* - EMR, Issue ID 35

The analysis unveiled that some delays can be attributed to the *limitations of tools*. In particular, the lack of accuracy in the output of current tools (e.g., missing some vulnerabilities during scanning, omitting patches during patch deployment) resulted in inaccurate vulnerability prioritisation and incomplete patch deployment respectively. Subsequently, the practitioners had to re-execute the tasks resulting in delays in task completion. Another limitation is associated with the lack of scalability to handle diverse types of patches and their features. In such cases, patches introduce complications to tool functionalities such as disabling some tool functions. Furthermore, we identified functionality limitations of existing tools like the inability to detect patch compatibility arising from the patch dependencies and the lack of capability to detect multi-reboot requirements that delayed the tasks.

> [Subject - Additional reboot required for .NET patching]
> *"7/2/20 - An investigation is needed around the number of required reboots for EMR patching and window requirements as a result if more reboots are required. A new process needs to be fleshed out when patching is postponed to accommodate the identification of the number of reboots required."* - EMR, Task ID 35

Another prominent cause of delays is ascribed to the *need of human expertise* throughout the security patch management process. The need for human intervention emerges because of the inability to achieve complete automation in the process owing to technological limitations. Troubleshooting the issues, mostly related to the unknown errors during and post-deployment, and faulty patches consumed a lot of the practitioner's time and effort delaying patch testing and deployment tasks. Similarly, manual configurations, for example, selecting the suitable Group Policy Object (GPO) configurations based on the needs and making decisions about the patch process, e.g., changes to the patch cycle and patch window, needed to be thoroughly assessed for the impact on multiple aspects to avoid breakdowns. Moreover, we noticed that the practitioners undertook manual patch deployment during complex, erroneous, or business-critical patch installations, for example, legacy systems patching. Manual intervention was also required for re-executing failed patch deployments and re-planning patch schedules due to requirement changes.

> [Subject - [Hospital h1] patching stage 3 on 27th November]
> *"18/10/19 - Patching needs to be moved to OOB due to the change freeze from 15th November to 3rd December.*
> *31/10/19 - [B-T1] team putting in significant amounts of work, like 15-20 hours per month, to redo the schedules on custom dates each time the deployments move off standard windows."* - EMR, Task ID 30

People-Related Reasons.

These refer to a group of reasons relating to the coordination of patch management, delivery of input requirements, and planning and execution of patch management tasks.

Delays occurring due to *lack of coordination* presented the most recurrent reason for delays. It refers to the delays in getting things done in the security patch management process. It is challenging because completing a single patching task (e.g., applying security patch X to server Y in Customer Z) involves multiple interdependent activities and several stakeholders. We found coordination delays stemming from both internal and external stakeholders.

Internal stakeholder coordination delays, in the studied context, relate to the delays from lack of coordination of dependencies deriving from the interactions between stakeholders of Org A and Org B. As several interdependent teams between the two organisations collaboratively worked towards an end goal of timely application of security patches to ensure systems' security, a delay of one party resulted in delays in task completion. Similarly, a lack of awareness of task progression between teams also created delays in inter-team task progression. As such, the multiteam system [118] in the studied context resulted in delays in decision approvals as they had to go through multiple teams (or levels). In addition, a lack of understanding of shared roles and responsibilities led to delays in coordinating tasks between teams because the task assignee did not know whom to contact in the event of errors or who was handling the interdependent task. We noticed that coordination delays also occurred due to missing information owing to an overload of emails. Email being the primary source of communication between the internal teams, there were cases where some emails had been missed resulting in delays in passing information on time. Moreover, poor communication and information misinterpretation contributed to delays in information passing.

Concerning external stakeholder coordination delays, we found delays attributed to the coordination with customers (e.g., hospitals), end-users (e.g., hospital patients and staff), and vendors (e.g., Microsoft). A dominant reason was the delays in obtaining customers' approval for patch deployment. Since patch deployment usually resulted in system downtime arising from the reboots, obtaining approval for patch deployment schedules was important.

> [Subject - Request to change patch window of [s1] server]
> *"10/5/19 - Currently set to 0000-0300, but the full backup of the server happening during this window causes slowness and issues with patching. Suggest changing the window to 0600-0900. [P1-AT2] checking on the status with business approval*
> *14/6/19 - [P1-AT2] to follow up as no response from the business."* - Win, Task ID 19

The other reason was the delays in coordinating with vendors for support. Coordinating vendor dependencies is integral to security patch management as the practitioners rely on vendors' support for errors encountered during patching and to obtain information about patch releases. Additional delays included the administrative overhead of coordinating with multiple customers for pre and post-patching verification and delegation delays due to conflicts of task ownership with other third-party vendors owing to a lack of accountability.

Another instrumental people-related reason was the *delays in providing input requirements*. This is because the security patch management process represents a sequence of phases with tightly coupled activities whereby the output of one phase is the input to the next phase. Similar to coordination delays, we identified that the input requirements delays emerge internally and externally.

Internal input requirements delays occurred when requested information was not provided by the internal teams on time. This included delays in delivering the reports such as the vulnerability scan reports which led to delays in vulnerability assessment and prioritisation. Similarly, delays in delivering patch schedules-related information led to delays in planning and subsequently deploying patches. Other reasons included delays in supplying other information requisites such as server details and providing the team's requirements in the patch cycle. An important observation was that the teams did not maintain an online repository with the server details which created the need for waiting for information about up-to-date server details (i.e., with the latest patched versions).

External input requirements delays are concerned with the requirements delivered late by vendors. For example, delays in the patch release, particularly the patches for fixing critical security vulnerabilities, can result in a significant increase in the risk of exposure to cyberattacks. Additionally, the delays in receiving vendor support for patching errors and new patch release information caused delays in addressing the vulnerabilities.

[Subject - New zero-day vulnerability warning]
"12/6/20 - Monitor Microsoft patch release for critical vulnerability identified on [T1] servers. Font Type 1 is expected as a zero-day soon, full report is not available yet.
24/7/20 - No update from Microsoft." - EMR, Task ID 43

We noticed that some delays were caused by *failures from poor planning and execution*. Security patch management in large and mission-critical domains like healthcare entails challenging tasks that need to be cautiously planned and executed to avoid system breakdowns. However, the complexity of patches, particularly, the unforeseen errors during deployment presented a major risk to deploying within the planned time frame. With regards to poor planning, inaccurate estimates of patch windows caused patch deployment to exceed the allocated patch windows resulting in inconvenience to customers and end-users. As a consequence, practitioners often halted patch deployment to avoid service disruptions resulting in patching delays.

[Subject - Execution exceeding the patch window]
"31/5/17 - Only 72.9% of scheduled patch deployments were completed as of 11.20 am. Two further windows are to be raised to ensure the appropriate length of time is scheduled due to unknown 2016 updates that were required to be implemented, first window is 1st June 8 am to 12 pm." - Win, Task ID 4

Of poor execution, missing patch prerequisites such as registry changes, GPO configuration and installation of preparation packages halted execution due to errors during the deployment. Similarly, incomplete patch deployment (e.g., failing to reboot after deployment which resulted in the installed patch not taking effect), and inadequate post-deployment patch verification such as failing to monitor the status of patch deployment tasks caused the need to re-execute patch deployment. Insufficient post-deployment patch verification also resulted in operation disruptions due to unexpected errors. For instance, we observed a heated discussion during a patch meeting owing to an issue with the printers not working reported by the customers to Org A caused by a lack of post-deployment verification by Org B.

Organisation-Related Reasons.

This category covers reasons relating to the organisation approvals, schedules, capacity to undertake patch management tasks and policies on service availability.

We found some reasons denoting *organisation delays* resulting from organisation policies and schedules. The need for compliance with organisation policies and the involvement of multiple parties (i.e., two organisations and several teams) has resulted in delays in obtaining approval from organisational management for monthly patch schedules and changes in the process. We also noticed that delays occur due to changes in organisation schedules such as change freeze periods, testing schedules like regression testing plans, and holidays (e.g., year-end shutdown period) during which no patch deployments were allowed to be scheduled.

> [Subject - Patching for December 2019]
> *"18/10/19 - OOB for November patching from 4th December instead of December patching.*
> *31/10/19 - [AT1] patching for December month is off but November Microsoft patches will be applied in the first week of December instead to keep compliance up."* - EMR, Issue ID 29

Further, we noticed a *lack of capacity* concerning human resources, infrastructure and the time leading to delays. With regard to resource constraints, insufficient human resources appeared to be a major factor in delays. For example, unavailability of task assignees due to high work overload and assignee being on leave held up the tasks in progress until the assignee was available. Another root cause was the lack of qualified personnel with sufficient experience to handle complex tasks such as legacy system upgrades, thus leading to an experienced practitioner getting overloaded with tasks that would end up queued for a long time. Regarding infrastructure-related limitations, hardware and network limitations hindered task progression in ways such as performance delays. For example, the high patch load described in the *complexity of patches (R1)* impacted the reboots following deployment and issues with the bandwidth required for patching due to a lack of capacity to handle the load.

> [Subject - Backup server patching]
> *"24/1/20 - Patching cannot go ahead when the active backup is running. The patch load can impact servers before reboot. Need a window change, proposal to be sent by [P1-BT1] to [P2-AT1]."* - EMR, Issue ID 39

Another reason stemmed from the periodic patch cycles as they presented the practitioners with a time-bound restraint to progress with the tasks. In particular, some tasks such as testing the workarounds for failed deployments had to be delayed for weeks given the time-driven (i.e., monthly) patch cycle in practice.

Another crucial cause of delays stemmed from the *service availability restrictions*. We noticed that patch deployment was often delayed due to organisations' inability to allow service downtime from reboots. Reboots were necessary for the patch to take effect after deployment and some patches required multiple reboots or multi reboots depending on the level of complexity involved, for example, the number of patch interdependencies. As such, the multi-reboots required longer and additional patch windows than the usually allocated 4-hour window. Consequently, in most cases, the patch schedules were delayed to be deployed in out-of-band (OOB) windows to reduce service disruptions from longer patch windows during business hours.

> [Subject - [Servers s1 and s2] patching]
> *"26/7/19 - OOB window is needed for the multi reboots to catch up.*
> *9/8/19 - Waiting for the customer's confirmation of the new patch window, pending information from [P1-AT1]."* - EMR, Issue ID 20

However, getting customers' approval for a change of patch window presented an additional challenge to the practitioners as customers were always hesitant about the risk of system downtime. Correspondingly, further delays occurred due to customers' requests to postpone the schedules to allow service continuity.

Summary for RQ2.1: We identified nine causes for patching delays associated with technology, people and organisation-related reasons. In a majority of the delays, we found multiple reasons attributing to one delayed task. Among these reasons, people-related reasons, for example, coordination delays appeared as the most prominent and recurrent reasons for delays. Concerning where the delays occur, we found that the delays are distributed throughout the security patch management process, however, most of the delays, i.e., 54% occurred during one phase, i.e., patch deployment. Yet, regarding the duration of delays, we found that tasks related to vulnerability scanning, assessment and prioritisation and patch testing phases account for the longest delays.

4.4.2 Mitigation Strategies for Delays in Security Patch Management

We identified a group of strategies implemented by the studied teams as corrective/reactive actions to reduce delays. Further investigation enabled us to identify *where* to apply the strategies in the security patch management process. Figure 4.6 presents the strategies grouped by the relevant security patch management process phase with the number of references for each strategy (in parentheses).

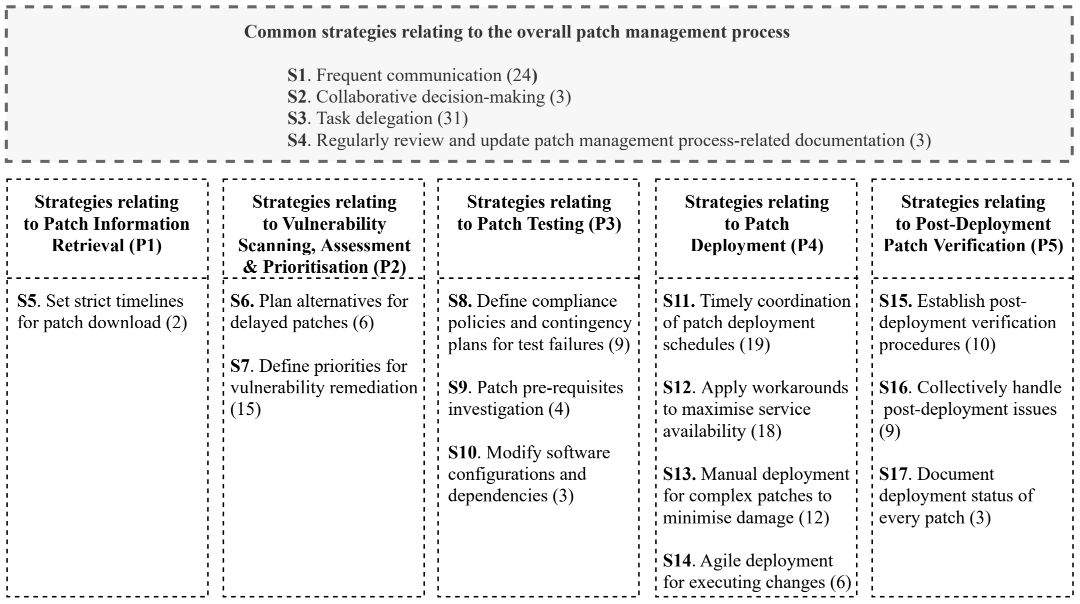

Common strategies relating to the overall patch management process
S1. Frequent communication (24)
S2. Collaborative decision-making (3)
S3. Task delegation (31)
S4. Regularly review and update patch management process-related documentation (3)
Strategies relating to Patch Information Retrieval (P1)
S5. Set strict timelines for patch download (2)
Strategies relating to Vulnerability Scanning, Assessment & Prioritisation (P2)
S6. Plan alternatives for delayed patches (6)
S7. Define priorities for vulnerability remediation (15)
Strategies relating to Patch Testing (P3)
S8. Define compliance policies and contingency plans for test failures (9)
S9. Patch pre-requisites investigation (4)
S10. Modify software configurations and dependencies (3)
Strategies relating to Patch Deployment (P4)
S11. Timely coordination of patch deployment schedules (19)
S12. Apply workarounds to maximise service availability (18)
S13. Manual deployment for complex patches to minimise damage (12)
S14. Agile deployment for executing changes (6)
Strategies relating to Post-Deployment Patch Verification (P5)
S15. Establish post-deployment verification procedures (10)
S16. Collectively handle post-deployment issues (9)
S17. Document deployment status of every patch (3)

FIGURE 4.6. Detailed overview of the strategies applied to mitigate delays in security patch management.

Strategies Relating to the Overall Patch Management Process.

The following set of *common strategies* can be applied across all phases of the security patch management process.

Frequent communication with all internal and external stakeholders is vital in reducing the patching delays as it helped strengthen collaboration and improve mutual understanding by bringing all stakeholders on the same page. Regarding internal communication, the studied practitioners held bi-weekly patch meetings to discuss patching issues, find solutions to the issues, report the status of patching tasks, and measure the progress of the patch cycle. Besides the patch meetings, they held informal discussions on complex and critical issues when required.

> [Subject - Post-deployment issue - Data Capture servers not able to communicate with [system s1]]
> *"7/8/20 - [P1-AT1] checking with [P2-AT2] for the other three servers that do not have a commissioning request.*
> *21/7/20 - Set up another meeting with BT1 to discuss this request (ID 1772737)."* - EMR, Issue ID 42

As to external communication, the practitioners frequently negotiated with customers about the patch deployment schedules. It involved getting consent for patch deployment at customers' premises, agreeing on the patch deployment dates and times (i.e., patch window), establishing contact persons at the customer sites for emergency contact and notifying completion of the patch deployment task. Similarly, the practitioners regularly negotiated with the vendors regarding the delayed patch releases and support cases raised for faulty patches. Frequent communication helped all stakeholders gain awareness of the tasks and schedules, assisting them with up-front planning and coordination of the dependent tasks.

> [Subject - Unix patching schedule confirmation]
> *"24/7/20 - The requirements analysis revealed a major OS upgrade, not simple patching. The schedule is still being negotiated with [customer c1]."*
> - Non-Win, Issue ID 7

Collectively making decisions about patch management, for example, patch prioritisation based on the vulnerability assessment results and organisation needs, and selecting workarounds for delayed patching and post-deployment issues helped the team members gain insight into the prospective plans and activities. In addition, it allowed the individual team members to make well-informed decisions about their task assignments which reduced the impact of the delays from waiting for input from dependent tasks and changes in the organisation's schedules.

> [Subject - Proposal for a patch cycle change in [servers s1 and s2]]
> *"4/4/18 - Discussions are still ongoing for the decision. AT1 is still considering various options and has put them out in slides for discussion at the meeting."* - EMR, Issue ID 2

We observed the patch meeting facilitator *delegating the tasks* to BT1 team members based on their expertise and experience during the patch meetings. In rare cases, the practitioners voluntarily self-assigned the tasks based on their interests and due to the unavailability of task assignees. The delegated tasks including details of the task, task assignee, raised by, and date of the assignment were documented in the patching tracker during the meeting. It appeared a useful strategy to increase dependency awareness of the tasks, particularly, in scenarios where task B is dependent on task

A (A $\rightarrow$ B) and the assignee of task B needs input from task A to progress with the task. Moreover, employing this strategy ensured well-defined roles and responsibilities around patch management activities resulting in increased accountability for actions.

> [Subject - Vulnerabilities in .NET Core]
> *"21/2/20 - .NET Core is not receiving updates. A new process is required to patch this version and a service request (SR) needs to be submitted for review and assessment. [P1-BT1] to raise the SR for the issue raised by BT1 on 7th Feb 2020."* - Win, Issue ID 40

Another common strategy that emerged from the data analysis was having a systematic process to *regularly review and update the documentation* about security patch management process actions and decisions. It is important to consistently review the process and test any process changes internally before documenting them. A well-documented process ensures clarity in the process activities and decisions and eases tracing back during troubleshooting post-deployment errors.

> [Subject - Update documentation for the split of [servers s1 and s2] patching into two procedures]
> *"13/12/19 - Finalising the documentation after testing internally for handover to 24x7.*
> *10/1/20 - Documentation to be tested in February, will be ready for handover in March."* - EMR, Task ID 24

Strategies Relating to Patch Information Retrieval (P1).

Setting tight timelines for patch download, for example, within two days of the *"Patch Tuesday"* when large vendors like Microsoft, Adobe, and others release the patches, was a strategy followed by the studied practitioners. It allowed them sufficient time to plan and coordinate the patch windows, negotiate with customers, obtain organisation approval, and undertake extensive patch testing before deployment. In the studied context, Org B provided a report to Org A teams containing a list of the retrieved patches each month that aided the collaborative assessment of vulnerability risks.

> [Subject - Provide .NET report at the start of the patch cycle]
> *"15/3/19 - Org A requests BT1 to provide an extract of .NET released patches every month and a report including what patches will be applied to what servers."* - EMR, Task ID 53

Strategies Relating to Vulnerability Scanning, Assessment and Prioritisation (P2).

We observed the practitioners *planning alternatives* for scheduled patching that will be delayed due to known reasons. For example, a major upgrade for critical legacy software is a complex and time-consuming process that often involves several challenging subtasks like an intensive assessment of the cost-benefit analysis and impact on other services, and laborious data migration procedures. In such cases, the practitioners planned alternatives (i.e., *what to do* and *when to do it*) for the time being until the software is patched to minimise the risks of attacks. We observed them collaboratively analysing various workarounds for suitability during delayed patch releases and delayed patching and assessing the timing of those alternate remediation plans.

> [Subject - Org A Change Freeze from 14th December to 12th January]
> *"31/10/19 - Due to the change freeze in December, there will be no patching for December.*

> *1/11/19 - AT2 wants to reserve the first two weeks of December for remediation and out-of-band patching of critical vulnerabilities as required."* - Win, Task ID 34

Defining priorities for vulnerability remediation appeared beneficial in reducing the risk of exploitable attack vectors from delayed remediation due to the large number and diversity of patch releases. The studied practitioners prioritised vulnerabilities based on the patch severity and impact. In the studied context, the security team (AT4) prioritised security patches based on the global vulnerability rating and their own risk assessment. High-risk critical patches were prioritised to be deployed within 48 hours while the medium to low-risk patches were deployed in the next patching cycle. Prioritisation based on the patch type, for example, operating system patch vs software application patch, was another strategy employed for defining the priorities. In some cases, we observed them prioritising the operating system security patches over other security patches like .NET, IE, Adobe, and Java.

> [Subject - OS security patches need to be tracked separately in the vulnerability remediation]
> *"15/5/20 - [P1-AT1] requesting the OS security patches to be tracked separately from all other vulnerability remediation. Org B's report should only be addressing OS security patches anyway but can make sure to separate any non-OS remediation tasks."* - EMR, Task ID 45

Strategies Relating to Patch Testing (P3).

Definition of compliance policies, for example, the standards imposed by the security team to reboot every legacy server even if there are no patches, and developing contingency plans in cases of failures appeared beneficial in mitigating the risk of delays caused by the erroneous patches.

> [Subject - [s1] new servers compliance]
> *"4/12/18 - This item remains open until all new [s1] servers are fully compliant including security hardening prior to being pushed into production and support.*
> *18/2/18 - Security approved the new [s1] servers, go-live completed on 18/2."* - EMR, Task ID 1

Patch prerequisites such as the registry changes and preparation package installation represent preconditions that needed to be set up for the patch to take effect during the deployment. As a strategy to avoid possible delays resulting from the runtime errors hindering patch deployment due to missing prerequisites and delays in manual configurations associated with the prerequisites, the BT1 team performed an *investigation of prerequisites* for the patches released every month as a separate task during patch testing.

> [Subject - Registry key missing for Knowledge Base (KB) ID [n] (LDAP)]
> *"2/10/20 - Patches not installed on [servers s1 and s2] due to missing a registry key. [P1-BT1] to check settings and apply where missing."* - Win, Task ID 24

In preparing the machines for patch deployment and avoiding potential delays arising from complexities of patches due to patch dependencies, the practitioners dedicated a specific time to *identifying and modifying the dependencies and configurations* during patch testing. For example, they created patch clusters based on the patch similarity and configured the group settings, also known as Group Policy Object (GPO), to reduce time spent on manual configurations on individual patches.

[Subject - GPO creations (configs) to be done together]
"12/6/20 - [P1-BT1] to create new GPOs at the same time early next week in preparation for the next round of additions for vulnerability remediation." - Win, Task ID 49

Strategies Relating to Patch Deployment (P4).

Well-timed coordination of patch deployment schedules can help mitigate several delays associated with the coordination delays, capacity limitations, organisation policies regarding service availability, organisation schedule changes, failures from poor planning, and increased rate of patch release during patch deployment. The activities involved internal planning and scheduling of the patch windows for each managed system (i.e., *when to patch*), defining the teams' roles and responsibilities for contacting customer sites for patch deployment verification and planning the servers' load to spread evenly through the patch windows to avoid performance issues and unexpected service disruptions during patch deployment (i.e., *how to patch*).

[Subject - Review of [s1] servers' patch windows: re-balancing and extended windows proposed]
"13/3/19 - Org B proposes 4-hour windows starting at 18:00 each night. The first lot of servers are to start Friday week 2 after the "Patch Tuesday". Sample of re-balanced servers provided for Org A's consideration." - Win, Task ID 9

Given the mission-critical nature of healthcare operations, the risk of system downtime from reboots presented a major challenge to the practitioners in reducing the risk of service disruptions during patch deployment. As a strategy to maximise service availability and reduce potential associated delays, they *applied various countermeasures including clustering, load balancing, and failover*. Clustering refers to grouping patches based on their similarity. As such, configuring the group settings and deployment of the patch clusters significantly reduced the time spent in testing, deployment, and rebooting than comparable single-patch work resulting in increased service availability. Similarly, load balancing which refers to balancing the load on servers during deployment helped avoid unnecessary service disruptions. This is because the servers will be patched in batches reducing the risk of all services being interrupted at the same time. Failover or maintaining backup servers to concurrently run the services while being rebooted was another workaround employed to minimise the downtime. Subsequently, the backup servers' patching was carefully planned with separate patch windows. A few other countermeasures included planning extended windows for patches that required multi-reboots in out-of-band windows and preloading the patches offline to avoid patch deployment exceeding the allocated patch window.

[Subject - Patch deployment failed at [server s1]]
"24/1/18 - Single point of failure for [server s1]. AT1 to review the proposed design for clustering for high availability. Currently hard to obtain reboot timings, only one reboot is allowed. Ask the customer for an extended window and move the patching to the weekend." - Win, Task ID 6

The practitioners decided to *shift to manual patch deployment* for business-critical server patching, complex patches that involved multiple version dependencies, multi reboots and legacy software systems, and redeployment of erroneous patches. This strategy was deemed effective in minimising the damage (i.e., service operations left unstable post-deployment) caused by failed deployments and avoiding the risk of further delays. However, we noticed that shifting to the manual deployment itself could lead to delays in patching as described in R3 in Section 4.4.1.

> [Subject - Post-deployment error at [server s1] causing the printing service unavailable]
> *"31/10/19 - This server was patched on 18/10 at 2 am-6 am. Due to the errors, patching will be done manually in November."* - Win, Issue ID 6

Agile deployment was another strategy employed by the teams where they executed the changes to patch deployment procedures in small iterations. This was adopted as a precautionary measure against unexpected breakdowns since a small change in the deployment process could result in disastrous consequences to service continuity and build confidence around the new changes.

> [Subject - Review of patch cycle timings]
> *"22/2/18 - Org A is considering going for bi-monthly deployment cycles for .NET patching and then moving to monthly deployment after the confidence is built."* - EMR, Task ID 2

Strategies Relating to Post-Deployment Patch Verification (P5).

Having a defined set of procedures for post-deployment patch verification helps reduce the risk of delays caused by failures from poor execution due to inadequate post-deployment verification. The studied teams verified the patch deployment status using several approaches such as monitoring the system for any functional, performance, or unexpected issues, analysing the system logs, collecting user feedback (i.e., confirming with customers about any adverse impact on service continuity), and getting periodic scans to verify the targeted security vulnerabilities have been patched.

> [Subject - Automated second rescan for reboots]
> *"31/10/19 - [P1-BT1] raised this issue, he has configured the window to rescan for missing patches and conduct a second reboot if required. No issues during patching, seeking client feedback for verification."* - EMR, Task ID 28

Post-deployment issues such as an unresponsive server or unavailability of service may have developed due to failures during patch testing and deployment, or lack of proper post-deployment verification. To avoid such issues leading to long delays causing unexpected service disruptions, the practitioners engaged in a *collaborative problem handling approach.* We observed long discussions at the patch meetings about the investigations of the root causes for post-deployment issues and finding workarounds for failed deployments. The most commonly used workarounds in the studied context included reverting to the previous working software version, restoring from the backup, and patch redeployment in out-of-band windows.

> [Subject - Tracking of GPO applications that are not intended]
> *"10/7/20 - Information sent to [P1-AT2] and [P2-AT2]. [Patch p1] will need a rollback.*
> *24/7/20 - Rollback executed, no reported issue due to the rollback. Keep*

open for one more meeting for monitoring." - Win, Task ID 51

The team members *documented the deployment status of every patch* in the patching tracker. It served useful as a vulnerability wiki to keep track of the progress of every patch and as a reference in cases of errors encountered during the execution. Further, employing this strategy during post-deployment patch verification ensured all patches are properly deployed and audited. As a result, the delays that occurred due to tool limitations, for example, missing patches during deployment were minimised.

[Subject - [Servers s1 and s2] successfully patched]
"24/7/20 - [...] No further issues experienced since patching. Manual instructions and deployment status are updated in the shared tracker. Will be kept in-monitor for another couple of weeks." - EMR, Issue ID 47

Summary for RQ2.2: We identified 17 strategies applied by the practitioners as corrective/reactive actions to mitigate the delays. Among these strategies, frequent communication, collaborative decision-making, task delegation, and regularly reviewing and updating the documentation were common strategies applied across all phases of the security patch management process. Further, we found a group of strategies executed at each phase of the process to mitigate the delays that occurred during each step.

4.4.3 Findings from Member Checking

The participants provided positive feedback on the study findings and agreed with the accuracy of the results. Several participants including the executive complimented our research, saying *"Thanks for all the information. Very interesting analysis"*- P1-Org A, *"From my point of view, I think your analysis is very good and useful because it's not just looking at how good or bad things are but also highlights where the improvement could be"*- Executive-Org A. Further, it was interesting to see their motivation to improve the delays following the presentation. *"I hate to see this good work going wasted, a really good analysis where we got some really good insights. So, I'd like to see our teams taking these on board, then revisit this to see how the pie chart changes when we address the top reasons for delays"*- Executive-Org A. The participants did not mention any new information or variations to the findings but explained the challenges of dealing with some of the delays, for example, *"The patching timeline is fixed by vendors such as Microsoft who use a monthly schedule so reducing the time frame of getting appropriate approvals and executing is an absolute necessity. And getting new patches tested, confirmed, and approved in a week is always a challenge before they are rolled out confidently to production"*- P1-Org A, *"Also, not all environments have testing environments to test these patches. So, in a fair few cases application testing actually occurs in deployment environments which can cause many failures leading to delays"*- P2-Org B, *"Yes, to add to it, vendors introducing application patches at the same time as OS patches can also cause delays and conflicts with OS security patching"*- P3-Org B. They also asked us several questions including how they can reduce the delays further, to which we suggested some improvements which are discussed in Section 4.5.

4.5 Discussion

In this section, we reflect upon our findings and discuss them in light of the existing literature. Further, we present the implications for research and practice.

Mitigating delays in security patch management is instrumental in maintaining the security, availability, and confidentiality of information technology (IT) systems [19], and failure to do so has resulted in several devastating outcomes [102]. Yet, the topic remains less explored in the literature, particularly, in understanding the practical reasons for delays in applying the patches. Based on a comprehensive analysis of the gathered artefacts over a period of four years, we have identified why, how and where delays happen in security patch management in practice and a set of corrective strategies to mitigate them.

Our findings unveil that the primary cause of the most prevalent delays (24.9%) is coordination delays in the patch management process (Figure 4.4). An interesting finding is that internal coordination delays were more recurrent than the delays occurring from external stakeholders, i.e., customers and vendors. This was confirmed during the member checking as described by the executive, *"I'm not surprised by some of these reasons, especially the coordination delays as the difficulties in collaborating and communicating between the teams are evident in almost every aspect of the process."* Although it appears that such delays are within the control of the practitioners, our findings emphasise the need for further support on coordination across patch management tasks and stakeholders. Similarly, with regards to the second most recurrent reason, the input requirements delays (16.8%), a majority of the delays emerged from internal teams as opposed to external vendors, indicating that adopting strategies like frequent communication (S1) and task delegation at meetings (S3) can help reduce such delays.

The next prominent reason, the complexity of patches (13.7%) can be attributed to the inherent complex patch dependencies and unknown risks of faulty patches. Although the intrinsic factors are essentially in control of the third-party vendors in charge of patch development, strategies like extensive patch testing to identify the prerequisites and inherent patch dependencies (S9, S10) [2, 54] and defining contingency plans to handle faulty patch errors (S8) can help reduce delays arising from the patch complexity. The socio-technical endeavour in security patch management constituting the fourth-most recurring delay (13.4%) can be explained by the inevitable need for human intervention in the process. While it suggests a need for a better understanding of human interaction in security patch management, our findings can guide practitioners in the planning of patch schedules allowing sufficient time for manual intervention (S11). Regarding the delays caused by capacity limitations in human resources, infrastructure, and time (9.6%), properly planning the task assignments with minimum task dependencies (S3), patch clustering and load balancing (S12), and implementing patch deployment changes in an agile manner (S14) can be helpful.

While organisation-related delays (7.4%) can be implied to be within the control of practitioners, service availability restrictions (6%) may appear difficult to always be taken control of. This is because service continuity presents a pressing need for modern enterprises, particularly in the context of mission-critical domains for which service disruptions even for a few seconds can result in severe consequences. As described by a participant during member checking, *"it is very challenging with the service availability restrictions, one example is the ambulance service, even though we have received approval, we always have to call the service just to confirm if it's okay to patch because we don't want to shut down the system in the middle of an operation"*. However, applying workarounds such as failover, clustering, and load balancing (S12) can help reduce such delays. Concerning the least occurring delays, limitations of existing tools (4.6%), although reflect reasons not within practitioners' control, having well-established roles, patch management practices, and policies can help mitigate such delays. Finally, the delays emerging from failures in poor planning and execution (3.6%) can be addressed with careful planning and execution (S9-11, S17).

Further reflecting upon our findings and in comparison to previous works, we discuss that some of the identified reasons are not necessarily specific to security patching in the domain of healthcare, but could be also observed in other domains. For example, the **complexity of patches (R1)**. The patch interdependencies are found to be intrinsic characteristics present in the patches released by the vendors [103]. Therefore, the resultant delays from managing these patch interdependencies could be challenging in other domains as well. Similarly, the need of human expertise is a standard notion accepted in security patch management because the process is inherently a socio-technical endeavour, where the human and technical interactions are tightly interconnected [54, 59]. Therefore, we find the reasons relating to the **need of human intervention (R3)** as reasons that could also apply beyond the studied context. In addition, we recognise the reasons relating to the **service availability restrictions (R9)** could be present in other domains as well. This is because the reboots following patch deployment are necessary for the applied patch to take effect. Further, the service interruptions caused by the reboots have been widely acknowledged as a major obstacle in patch management across several domains [2, 20, 106, 107, 108].

In contrast, we believe that some of the reasons are likely to be specific to the domain of healthcare and the context of studied organisations. For example, the reasons attributing to **organisation delays (R7)** and **capacity limitations (R8)**. Concerning organisation-related reasons, patching delays resulting from delayed approval from higher management may not directly apply to a small organisation with one team or to an organisation with a flat hierarchy where no line approvals are needed. Although these reasons may not necessarily represent reasons beyond the studied cases or the context, an understanding of the context-specific reasons enables researchers and practitioners to better appreciate the practical utility of the solutions and formulate appropriate plans for mitigating potential delays. We believe there are possibilities for future research to explore the reasons for delays in a broader context using these categories.

4.5.1 Related Works

Our study confirms the findings of the previous studies that suggest some challenges in security patch management could contribute to delays in patching. For example, our finding of coordination delays contributing to the majority of the delays complements the existing research [2, 22, 59, 103, 106], which reported that coordination is one of the most pressing challenges of timely patch management. Our analysis extends the knowledge by showing how coordination delays are introduced internally and externally. Additionally, our findings further highlight the importance and the need to focus more on the socio-technical aspects such as coordination in the time-critical security patch management process as mentioned by previous literature [2, 20, 54, 59].

Our analysis reveals that the complexity of patches causes the third-most frequent reason for delays; it complements the previous work [20, 25, 59, 103, 104, 110, 112], which has mentioned that faulty patches and configuring patch dependencies are challenging as they often lead to breakdowns during patch deployment. Similarly, the need of human expertise in the process [2, 20, 25] and capacity limitations, specifically, lack of human resources [20, 110, 111] are mentioned as challenges in security patch management in the related studies. Further, several studies (e.g., [2, 20, 106, 107, 108]) have highlighted service disruption as a central challenge of patch deployment. Our study extends the knowledge of these challenges by showing how, why and when they contribute towards patching delays.

Alternatively, previous studies [60, 61] have predominantly focused on achieving timely patch management through optimising the process by attaining a balance between an organisation's patch cycle and a vendor's patch release cycle. Dey et al. [24] have developed a quantitative framework that analyses and compares various patching policies to find the optimum policy considering the costs of periodic patching against the security risks from patching delays. By investigating vendors' patch release and practitioners' patch deployment practices, Nappa et al. [103] revealed that only 14% of the patches are deployed on time and the patching mechanism (e.g., automated vs manual patch deployment) impacts the rate of patch deployment. Despite the widespread attention towards timely security patch management, an important observation is the absence of an investigation of the root causes (i.e., reasons) for delays in security patch management. To the best of our knowledge, the existing studies have not explored why the application of patches is delayed but rather proposed approaches to achieve a timely patch management process. Hence, our study contributes to the existing body of knowledge by:

- Providing a taxonomy of reasons explaining why delays occur when applying security patches in practice;
- Reporting what reasons for delays are more prominent based on frequency analysis;
- Demonstrating where the delays occur in the security patch management process;
- Presenting a set of strategies to mitigate the delays and describing when they can be applied in the security patching process;
- Providing practical implications for practitioners to identify and mitigate delays;
- Establishing a foundation for future research towards effective management of patching delays.

4.5.2 Implications for Practitioners

Our findings reveal why delays happen when applying security patches in practice with a set of reasons contributing to the delays, explain how the reasons vary, and how delays are distributed in the security patch management process. As a direct practical implication of the provided understanding, the security analysts and system administrators will be able to identify and assess the factors associated with the causes of delays and take precautions to mitigate potential delays. Further, the understanding of the frequency analysis of reasons and distribution of the delays highlights *what* reasons need practitioners' immediate attention and *where* is improvement needed to overcome the delays. In addition, the knowledge will help practitioners in suitable decision-making, prioritisation, and planning of security patch management tasks with minimal impediments.

In addition to explaining why, how, and where delays occur in patching, our findings describe how the delays can be mitigated. We present a set of strategies employed by the studied practitioners to rectify the delays. Knowing what to do and when to do can be useful for practitioners and organisations in taking prompt actions to mitigate the impact of delays. The findings may also help predict a delay in a given scenario whereby practitioners can better plan patch cycles and refine the patching process in light of their organisational contexts. For example, practitioners can consider the development of new tools like Environment Diagrams as a visualisation tool, to keep track of the system dependencies that would save time in patch testing and deployment. Other approaches like maintaining an online shared repository documenting organisation schedules and regularly documenting patch exemptions in detail would assist teams with accurate planning of patch schedules. Towards overcoming delays of coordination in patching, adopting computer-supported collaborative tools like “Slack" can benefit from accomplishing timely communication, collaboration, and information sharing between all stakeholders [119]. In this way, our findings offer guidance to practitioners to make suitable decisions to alleviate the threat of cyberattacks from delayed patching.

4.5.3 Implications for Researchers

Given our findings are based on the cases studied limited to the domain of healthcare, other researchers can extend and adapt the results through future studies within the same domain involving different stakeholders or different domains. Further, future research exploring the viability of the findings based on the contextual factors, for example, variations in context-specific reasons for delays like capacity limitations (R8) and organisation delays (R7), can result in useful insights from additional cases with extended scope. With regards to the reported strategies for mitigating the delays, future studies can investigate their suitability and effectiveness depending on the context and organisation policies (e.g., similar to future work of [2, 20]). In addition, the findings can be used in potential interview guides and surveys to verify the findings in other contexts and discover variations within them. Another possibility is to investigate the impact of patching delays on organisations and other stakeholders such as end-users.

Considering the most recurrent delays occur due to a lack of coordination in the patching process and delays in providing the required inputs, future research can invest efforts into understanding in depth how and why ineffective coordination cause delays in security patch management in practice and what can be done to address the coordination delays in the security patch management process. Another direction is developing computer-supported tools and platforms that can support better coordination across patching tasks and reduce delays in collaborative tasks.

Further, the data analysis has revealed that the limitations in current tools contribute to delays in applying the patches. We believe that future research can address this limitation by developing advanced tools leveraging deep learning techniques. For example, an automated tool that provides dependency visibility by highlighting mismatches of patch dependencies. Solutions could be further investigated on how automation support can be extended to assist the decision-making in patch management, for example, developing intelligent interactive systems like software bots [120] for collaborating with practitioners that guide them to decisions by asking rational questions. This further opens up an avenue for future research to explore how *“human-AI collaboration"* [91], an emerging research paradigm [121], can be extended to a crucial topic like security patch management. Moreover, there is room for research to explore how to improve the performance and accuracy of the patch management tools. Tool development needs to consider the diversity of operating systems, software applications, platforms, and programming languages in vulnerability patch management to overcome the obstacles of lack of accuracy and scalability in the current tools (R2). This presents an excellent opportunity for future researchers to investigate the role of automation in security patch management in practice to better understand practitioners' needs and wants in order to cater to them. In addition, the researchers, particularly the usable security researchers, can study how to improve the design of such smart tools.

4.6 Threats to Validity

In this section, we discuss the potential threats to validity and how they were mitigated following the guidelines proposed by [33, 122, 123].

4.6.1 External Validity - Generalisability

This study is based on the empirical data collected from a particular context, i.e., security patch management in the healthcare domain. Hence, our findings do not claim for generalisation to all other contexts of patch management, instead, this study focuses on performing a comprehensive investigation of the delays in security patch management within the studied setting to provide detailed explanations through rigorous data analysis. However, we do not assert the results to be absolute or final, rather they can be recreated and adapted in other contexts [34, 35, 45].

Regarding data representativeness, the study includes data collected limited to the patching tracker. However, collecting data from two organisations with multiple teams including participants with diverse roles and wide experience increased the data reliability and assisted in ensuring participant triangulation [123]. Although we have analysed data spanning over four years from October 2016 to May 2021, it is possible that we may have missed some variations in the findings, specifically the context-specific reasons and strategies. We suggest that any future studies on this topic include more data sources such as additional cases or interviews to extend the scope of our findings and verify their explanatory power in other contexts.

4.6.2 Reliability

To mitigate the threat of subjectivity and ensure reliability in the data analysis, all the data collection and analysis procedures, emerged codes, and identified relationships were discussed in detail among all researchers and finalised through multiple revisions. In addition, related to interpretive validity [123], we conducted member checking to verify the accuracy of our findings, which was attended by the present author and two other researchers, further ensuring investigator triangulation.

4.6.3 Construct Validity

To address the threat of construct validity, we used multiple sources of evidence, i.e., analysis of artefacts and observations, and multiple stakeholders, and maintained a chain of evidence (e.g., the coding procedure following the Grounded Theory data analysis method [124]), and had the findings reviewed through member checking.

4.6.4 Internal Validity

To mitigate the threat of internal validity and misrepresentation, we ensured participant triangulation by covering the entire population involved in security patch management representing all teams in both organisations. In addition, the participants had a wide experience in security patch management, which helps mitigate the risk of participants' lack of expertise.

4.6.5 Evaluative Validity

The verifiability of the findings that emerged from a Grounded Theory data analysis can be attained from the adequacy and soundness of the research methodology through which the findings emerge [34, 35]. To achieve this, we have detailed our data analysis process of the application of the Straussian GT data analysis procedures in Section 2.2.2. Further, to alleviate the reporting bias, we have included quotes from the patching tracker in Section 4.4.

4.7 Chapter Summary

In this chapter, we empirically explore and systemically explain why, how, and where delays occur when applying security patches in practice, and how the delays can be mitigated. Through a longitudinal industrial case study representing eight different teams from two organisations in the domain of healthcare, and based on a Grounded Theory data analysis of 132 delayed tasks documented in the patching tracker over a period of four years from October 2016 to May 2021, we identify a set of reasons relating to technology, people and organisation that cause delays in security patch management. We also provide an analysis of the frequency distribution of reasons for delays and distribution of delays over the security patch management process. Such information highlights the reasons that need immediate attention and the areas of improvement in the security patch management process. Additionally, we report a set of strategies that can be used for mitigating the delays in applying security patches by practitioners.

Compared to the related literature, our study provides an evidence-based understanding of the reasons for delays when applying security patches in practice, and to the best of our knowledge, it is the first attempt to empirically investigate the topic in-depth. We assert that the reported understanding of why, how, and where delays occur during patching and how they can be mitigated will help practitioners take suitable decisions to mitigate delays and guide them towards taking timely actions to avoid potentially disastrous consequences from delays in patching. Furthermore, our findings lay the foundation for future research to investigate and develop computer-supported tools that can address the practical concerns causing delays in security patch management, drawing attention to a topic, critical and timely, yet less explored in the software engineering community.

Chapter 5

A Grounded Theory of the Role of Coordination in Security Patch Management

Related publication: This chapter is based on our paper titled *"A Grounded Theory of the Role of Coordination in Software Security Patch Management", published in the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2021 (CORE Ranking A*) [56].*

In Chapter 4, we have identified a set of reasons relating to the socio-technical aspects of security patch management that are key explanations that cause delays in security patching and discovered that the most prominent cause of delays relates to a lack of coordination in the security patch management process. While researchers and practitioners have paid attention to studying the socio-technical challenges in patch management, there has been relatively little effort dedicated to gaining an in-depth understanding of how and why the socio-technical aspects, e.g., coordination of interdependent activities of the patching process and patching decisions impact patching delays. In this Chapter, we address this gap by presenting a grounded theory of the role of coordination in security patch management. The reported theory consists of four interrelated dimensions, i.e., causes, breakdowns, constraints, and mechanisms. The theory explains the causes that define the need for coordination among interdependent software/hardware components and multiple stakeholders' decisions, the constraints that can negatively impact coordination, the breakdowns in coordination, and the potential corrective measures for reducing coordination delays. This study provides potentially useful insights for researchers and practitioners who can carefully consider the needs of and devise suitable solutions for supporting the coordination of interdependencies involved in security patch management.

5.1 Introduction

Timely development and application of security patches to the identified vulnerabilities are considered critically important to avoid potentially successful security attacks [17]. As revealed in Chapter 4, a majority of the delays in the security patch management process emerge from socio-technical aspects such as coordination [2, 20, 54, 59]. As described in previous chapters, security patch management activities entail several socio-technical aspects that underpin some of the critical decision points that make security patch management a complex and challenging undertaking [2, 20, 22]. Further, it needs the coordination of the efforts and decisions of multiple stakeholders with conflicting interests and several interdependencies [100].

While it is widely understood that effective coordination is needed for timely decisions and actions by the involved stakeholders, there is not much empirically known about why and how coordination causes delays in applying patches as described in Section 4.5.3. That means researchers and practitioners may not find much guidance in gaining a good understanding of the role and impact of coordination in security patch management to answer some critically important questions: What is the role of coordination in security patch management? How and why does the coordination aspect cause delays in security patch management? What can be done for addressing coordination delays in security patch management? These questions motivated us to carry out a Grounded Theory study of the role of coordination in security patch management.

In this Chapter, we present the first, to the best of our knowledge, Grounded Theory study exploring the role of coordination in security patch management. It is based on observations of 51 patch meetings over a period of 9 months, which involved 21 industry practitioners from two organisations in the mission-critical healthcare domain. We explain how coordination impacts the security patch management process in four interrelated dimensions: causes, constraints, breakdowns, and mechanisms. Grounded in the evidence from industrial practices of security patch management, the theory aspires to enhance the state-of-the-art understanding of researchers and practitioners in several ways:

1. The theory highlights the importance of gaining a deep understanding of the interdependencies before applying security patches and how improved support in coordination can help reduce the delays in security patch management.
2. It structures the knowledge about the unexplored phenomenon of security patching in the mission-critical healthcare domain.
3. Provides a theoretical model to shape future Software Engineering (SE) research to address the practical concerns in security patching.
4. Practitioners can leverage the understanding to reduce patching delays attributable to coordination delays in the process.
5. The theory can also be useful to practitioners as guidance to enhance confidence in patching decisions.

Chapter Organisation: Section 5.2 summarises the related work. Section 5.3 presents the application of the Grounded Theory research method. Section 5.4 reports the results of the theory. In Section 5.5, we discuss the contributions of our theory in comparison with prior related work and its implications for practice and research. The threats to the validity and verifiability of the findings are presented in Section 5.6. Finally, Section 5.7 concludes the study.

5.2 Related Work

There has been extensive research [25, 22] on improving automation support in the security patch management process, however, we noticed a scarcity of empirical studies investigating the socio-technical aspects of security patch management as described in previous chapters. Existing empirical studies on socio-technical aspects of security patch management have primarily focused on studying system administrators [2, 20, 25, 109], the patch management process and its challenges [2, 20, 54], and patch information retrieval behaviours and approaches of system administrators [20, 110].

We found several studies (e.g., [2, 22, 54, 103, 106]) reporting coordination and collaboration challenges in the patch management process. However, they lack a comprehensive investigation of what causes the coordination needs and related challenges, its effects on the security patch management process, and the impact on delays of patch application. Some studies [20, 59, 125, 126] have described at an abstract level the dependencies between multiple stakeholders, such as vendors and organisations. Similarly, Nappa et al. [103] reported that the coordination challenges concerning vendor dependencies arise from a lack of synchronised patch releases from different vendors because of shared vulnerabilities in the software code. Their analysis was based on a large data set of deployed patches in client-side vulnerabilities. Similarly, quantitative models and frameworks presented by a few other studies [24, 60, 61] focused on optimising patch management by synchronising an organisation's patch cycle with the vendor's patch release cycle to reduce patching costs and risks. As such, the reported dependencies with software vendors raise important concerns about the need for an in-depth understanding of the role and impact of such dependencies on security patch management.

However, coordination has been studied extensively across various dimensions in related domains such as software development over the last decades [127, 128, 129, 130, 131, 132]. The literature defines coordination as *the management of interdependencies* [133, 134, 135] and describes different types of coordination as explicit coordination and implicit coordination. Similarly, Crowston [133] provided a categorisation of the types of dependencies based on the context such as task, knowledge, resource, and technical dependencies. Furthermore, previous work [129] on the coordination challenges in software development processes has demonstrated that ineffective coordination of dependencies represents a major cause of project failure, justifying the need for effective coordination to manage various interdependencies. Correspondingly, a comprehensive understanding of the role of coordination in patch application presents a critical research gap [54, 100], which is fulfilled by this study.

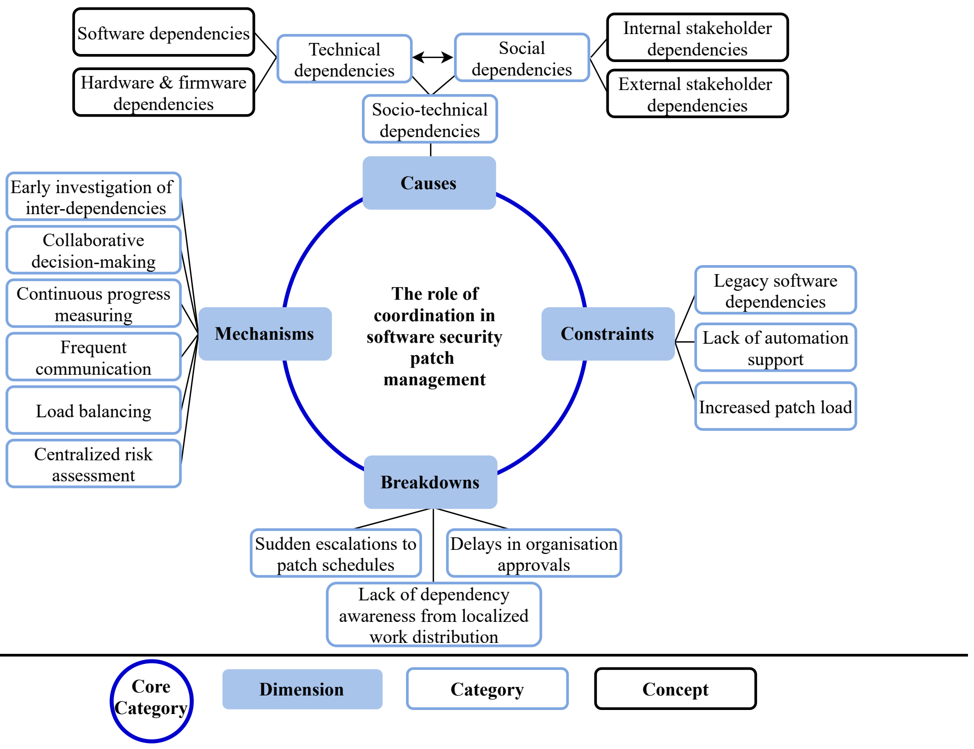
5.3 Research Method

We used Grounded Theory (GT) [35, 44] for data collection, analysis, theory development, and reporting. The details of the application of the Grounded Theory research method with examples and the reasons for the choice of this research method were described in detail in Section 2.2.3.

5.4 Findings

In this section, we describe the theory of the role of coordination in security patch management using four interrelated dimensions: Causes, Breakdowns, Mechanisms, and Constraints, providing evidence with grounded quotes from the underlying data.

Figure 5.1 presents the theory of the role of coordination in security patch management. For ease of reference, we used unique identifiers to refer to participants, for example, *AT5, M10-dis* refers to a participant from Org A's Security team in the 10th post-meeting discussion, and *BT1, M4* refers to a participant from Org B's Server team in the 4th meeting (see Table 2.1 and Table 2.4).



Software dependencies
Hardware & firmware dependencies
Technical dependencies
Social dependencies
Internal stakeholder dependencies
External stakeholder dependencies
Socio-technical dependencies
Causes
Early investigation of inter-dependencies
Collaborative decision-making
Continuous progress measuring
Frequent communication
Load balancing
Centralized risk assessment
Mechanisms
The role of coordination in software security patch management
Constraints
Legacy software dependencies
Lack of automation support
Increased patch load
Breakdowns
Sudden escalations to patch schedules
Delays in organisation approvals
Lack of dependency awareness from localized work distribution
Core Category
Dimension
Category
Concept

FIGURE 5.1. A grounded theory of the role of coordination in security patch management.

5.4.1 Causes - Socio-Technical Dependencies

We identified several socio-technical interdependencies inherent in the security patch management process that define the need for coordination. We recorded discussions about several scenarios exemplifying the potentially disastrous implications as a result of failing to identify dependencies before applying a patch. Through the data analysis, we have placed the observed interdependencies in two categories: *Technical* and *Social Dependencies.*

Technical Dependencies

The interdependencies between the software and the associated hardware and firmware give rise to technical dependencies, arising as a result of dependencies in software code. For example, a security vulnerability present in multiple functions in the source code may include dependencies among the functions. As a result, a security patch developed to fix the identified vulnerability also inherits these function-level dependencies creating complexities that must be carefully handled during the security patch management process.

In terms of **software-related patch dependencies,** several factors such as operating system (OS) dependencies, software application dependencies, and prerequisites for patch installation fostered the conditions for coordinating the dependencies. Security patch management in large software systems involves managing multiple software components (or services) with different OS versions. The existence of the interdependencies between OS versions and other software applications built on top of a particular OS like web browsers may create additional tasks for practitioners as all of them need to be synchronised. For example, the inability to detect incompatible version dependencies of different OSs prior to patch installation led to unexpected errors that often needed time-consuming manual investigations.

"We have about 15-16 versions of Windows 10. So, before patching we need to see which version is running on which server? What is the build number? Are we running the latest? It's a lot!" - AT5, M14-dis

Additionally, some security patches contained interdependencies with legacy OS. It presented a much more arduous task to the studied practitioners since some critical emergency medical services were running on Org A's legacy systems that were not supported by large vendors, e.g., Microsoft. In such cases, the participants often felt forced to delay patch installation as the available solutions like decommissioning or upgrading legacy systems presented high risks of operation interruptions. Consequently, this practice of delayed patching of security vulnerabilities in systems would significantly increase the risk of exposure to attacks.

Similarly, interdependencies between software applications, platforms, and tools presented another major category of software-related dependencies. This is because the build dependencies between the software application and patch sources require the versions to be in sync before attempting patch installation [136]. As the size of an organisation grows, managing these dependencies appeared difficult with a large number of diverse applications installed. For example, Org A's software applications ranged from general applications, e.g., Java, .NET to specific medical applications, e.g., Electronic Medical Record Software. As such, Org B's teams spent most of their time detecting the existing interdependencies such as version incompatibilities between various software applications.

"There's an old HP tools version and a new version, and the vulnerabilities are coming up on the scan as with the new version. But the issue is because the old version is still there which we should have got rid of earlier." - BT1, M14

The circular dependency represented a more complex semantic dependency in security patch management. An example scenario was when software B is dependent on software A, and software A uses software B to function (B $\leftrightarrow$ A). In such cases applying a security patch to A led to service unavailability of B as a result of rebooting A to make the security patch take effect. In particular, effectively coordinating circular dependencies was crucial in the healthcare domain as A and B could represent critical medical services like emergency life support or surgery equipment.

"And there could be like A needs B to run, and vice versa but when we accidentally took B offline that day, A didn't work. That was when we all got goosebumps." - AT5, M10-dis

On the other hand, some security patches required prerequisites to be established before installation for the patch to take effect. In most cases, the prerequisites comprised registry changes and preparation package installation. We identify that it is resulting from the patches that do not contain source code modifications, as explained by Li and Paxson [137]. To investigate the prerequisites of the security patches retrieved each month, Org B allocated a specific time frame before patch testing and discussed with Org A during the patch meetings how and when they would handle the identified prerequisites. Coordinating the prerequisites was often a manual task as it involved decisions about suitable configurations based on the organisation's needs and the other associated software dependencies. Failing to configure the prerequisites led to errors that would halt a patch installation. However, we found that the teams became more receptive to detecting prerequisite-related dependencies with the continuous early investigation approach employed.

"The patches listed here needed a preparation package installed before the patching window and then the reboot would have applied the patch. We'll do that just before the current patching window and then patching should proceed as normal without errors." - BT1, M11

Besides the most common software-related dependencies, some security patches also contained dependencies with the associated hardware and firmware giving rise to **hardware and firmware-related dependencies**. For example, in one instance, practitioners were unable to patch the security vulnerabilities found in virtual machine (VM) software as the VM-related firmware was not up to date. So, they had to regularly keep track of the existing dependencies and update the supporting hardware and firmware accordingly before attempting patch installation.

"Some patches need a certain type of hardware to be at a certain level. There was a 2008 security patch which we couldn't install until we updated the firmware or the utilities." - AT5, M10-dis

Social Dependencies

Social dependencies that stem from interdependencies between stakeholders are another major category of dependencies integral to security patch management. Security patching in large organisations is challenging due to the increased complexity stemming from a high number of stakeholders. Therefore, effectively coordinating the dependencies between multiple stakeholders is important for successful patch management. Our data analysis led to the emergence of two subcategories of social dependencies: Internal stakeholder dependencies and External stakeholder dependencies as illustrated in Figure 5.2.

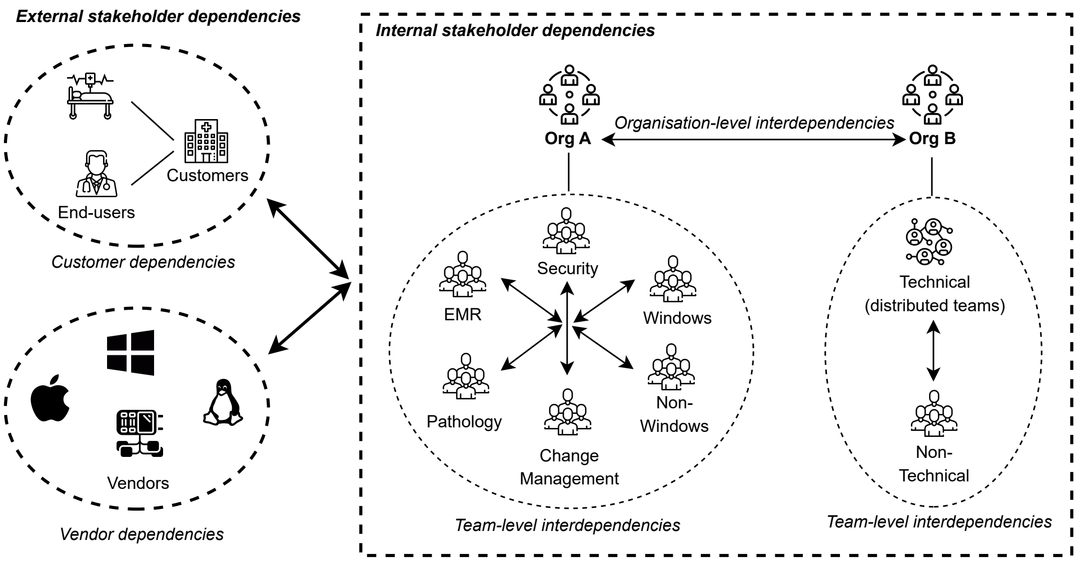

External stakeholder dependencies
Customers
End-users
Customer dependencies
Vendors
Vendor dependencies
Internal stakeholder dependencies
Org A
Organisation-level interdependencies
Org B
Security
EMR
Windows
Pathology
Non-Windows
Change Management
Team-level interdependencies
Technical (distributed teams)
Non-Technical
Team-level interdependencies

FIGURE 5.2. The social dependencies present in the studied context.

Internal Stakeholder Dependencies - The two organisation's stakeholders worked together to achieve a common goal of securing the state's healthcare system by timely installation of security patches. Hence, the internal stakeholder dependencies, in this context, relate to the dependencies stemming from the interactions between stakeholders across Org A and Org B. We identified two layers of dependencies namely *team-level dependencies* and *organisation-level dependencies.*

The current context displays a *multiteam system* (MTS) structure [118] as multiple interdependent teams within each organisation collaborated towards a collective goal. The interdependent team interactions gave rise to *team-level dependencies.* Since each organisational team had assigned responsibilities, coordination between and within teams remained pivotal to achieving the goals. In most cases, the inter-team tasks contained dependencies that required management of the cross-team interconnections. For example, Org A teams T1, T2, and T3, often depended on T6 to approve security patch schedules before assigning them to Org B to be executed. Lack of awareness of the roles and responsibilities complicated the coordination of team dependencies causing delays of several weeks in patching known vulnerabilities.

"We are still waiting for an email from [P1] approving the manual patching process. (BT1)
Well, I'm not sure whether it should come from [P1] or some other guy. I will confirm it with [P2] and get to back you." - AT2, M8

Similar to the inter-team dependencies, *organisation-level dependencies* also created several challenges for security patch management. The large-scale and heterogeneous nature of Org A created additional complexities to coordinating organisation-level dependencies that often resulted in delays in patch testing and deployment. For example, every security patch required approval from Org A's Change Management (T6) before they could be installed at the customer's sites (i.e., hospitals). Given the critical nature of the task, it was important to effectively coordinate it and leave sufficient time for the customers' agreements. It was needed to reduce the time of service interruptions and manage technical dependencies.

"This morning [Org A's] change manager said that the change hasn't been approved yet. So, we had to suddenly change plans just minutes before the [scheduled] patch window." - BT1, M10

External Stakeholder Dependencies - The involvement of external stakeholders such as customers, end-users, and vendors is integral to security patch management. Effectively coordinating the collaborative relationships and the dependencies with external stakeholders is vital to a sound patch management process. In this context, the external stakeholders consisted of *customers* (e.g., hospitals) and *vendors* (e.g., Microsoft). The end-user dependencies consisted of hospital patients and staff, hence, these were not directly linked to Org A and Org B. However, ineffective coordination of customer dependencies negatively impacted end-users. For example, uninformed operation interruptions to medical equipment resulted in inconveniences to patients and medical staff. In contrast to internal stakeholder dependencies, managing external stakeholder dependencies presented a much more difficult challenge to practitioners. The main reason was the lack of a shared understanding of the importance of security patching and the visibility of the existing process interdependencies.

Vendor-Related Dependencies refer to the dependencies that are created due to the need of installing security patches received from vendors. Management of vendor dependencies became difficult with the presence of shared vulnerabilities and associated technical dependencies in software applications. This demanded coordination of patch releases from the vendors of different software applications. Additionally, some of Org A's third-party applications were patched by external vendors, for example, the medical application providers, as per the agreement at the point of purchase. Thus, it required synchronising each vendor's patching cycles to avoid the unavailability of the interdependent systems.

"Regarding the recent concern from Security (T5) on [S1 server] patching is one month behind, can we confirm this with [third-party vendor's] requirements? Because this vendor's patching cycle is always one month behind, every month they release the patches for the last month." - BT1, M17

Furthermore, missing, faulty, or exempted security patches and unknown errors during patch installation increased the need for coordinating vendor dependencies during patch testing, deployment, and post-deployment patch verification. Patch exemption was when selected security patches were excluded from installation due to legitimate reasons approved by Org A's Security team. We observed several scenarios of patching delays due to a lack of coordination of software dependencies with exempted security patches that were managed by external vendors.

Managing *customer-related dependencies* with hospitals presented a challenge to Org A and Org B teams, particularly when negotiating the patch schedules. Reaching a consensus on a patch installation time was essential to minimise any potential impact of service disruptions from reboots. However, an interesting observation was that in a majority of cases, practitioners spent most of the time trying to communicate to customers on the need to patch systems, as opposed to agreeing on the patch installation time. It was due to the lack of understanding of the need to apply security patches and the inability to accept the high risks of service downtime.

"A lot of customers don't always understand the worth of security patching, they just want to use the server, and keep asking; "why do you want to reboot it every then and there, or why you got to update it? It's working so leave it alone!" - AT5, M4-dis

Detailed planning and proper coordination were essential during patch installations at hospitals. Since service interruptions are critical in the health domain, Org B team members were required to complete the patch installations within a specified patch window. Failure to do so, at times, resulted in even heated discussions between Org A and Org B teams. An example scenario that resulted in an emergency at a hospital due to ineffective coordination with hospital staff was that one of Org B's system administrators installing a patch to a surgical medical device forgot to inform the hospital's medical staff about the additional service interruption.

"For mission-critical and time-sensitive medical devices, we will actually ring and check up with the nurse to make sure it's not an issue to patch." - AT5, M3-dis

5.4.2 Constraints

This section presents the constraints that hindered the coordination. When the constraints affected the socio-technical dependencies, they caused coordination breakdowns. Hence, it is important to devise suitable approaches and tools for identifying and managing the potential impact of constraints.

Legacy Software-Related Dependencies

Legacy systems pose a security threat to organisational ICT infrastructures, particularly in mission-critical domains like healthcare. This is because most of the critical services that run on legacy systems remain unsupported by vendors leaving them vulnerable to security attacks. Org A had several legacy systems like 2008 servers that were no longer receiving security patches from Microsoft. Furthermore, the dependencies with these legacy systems produced wider implications for security patching.

It resulted in practitioners being unable to patch until the dependent legacy systems were upgraded to the current version or offered extended support from vendors. However, upgrading legacy systems required a critical evaluation of the impact on other important services. In some cases, Org A had acquired extended support for critical legacy systems through negotiations with the vendors. However, it presented practitioners with the additional challenges of having to perform manual configurations to install security patches as the current configuration settings did not work and some installed patches got rolled back.

"There are 26 2008 servers still waiting to be patched this month. But there are some servers that we need to have a look at why the patches aren't applying. Even though we have installed all the required preparation packages, they keep rolling back." - BT1, M12

Similarly, prerequisite dependencies associated with legacy systems create hurdles for timely patch installation. The major hindrance factor is the incompatibility of software versions as security patches often contain interdependencies that require associated software versions to be up-to-date. Such uninformed dependencies led to practitioners spending a considerable amount of time in patch meetings trying to find solutions for version dependency-related challenges. Moreover, the incompatibilities with version dependencies created additional difficulties during patch testing. Investigating the right pre-installation packages with the correct versions presented a daunting manual task for many practitioners that resulted in delays in patch installation and increased security risk.

"I found that those 46 IE-related vulnerabilities are 2008-related and everything is running on IE 11. That was why the patch installation looked to be corrupt as we couldn't update it to IE 11 back when the updates happened." - BT1, M10.

Lack of Automation Support

The lack of suitable tool support presented a major constraint for the coordination of the dependencies. One of the key constraints was the inability to oversee technical dependencies across all systems inventory. The inability to identify the existing software dependencies from the current tools resulted in practitioners spending hours trying to find the current software versions in the event of errors during a patch installation. Additionally, the lack of automation support to investigate the patch prerequisites as mentioned in Section 5.4.1, caused delays and sometimes errors in installing patches due to missing out on some registry changes.

On the other hand, the limitations on the features of the available support tools presented constraints in detecting specific dependencies such as legacy dependencies and their contextual categorisation. This could be because most of the tools available to the studied teams focused on function-level patching assuming that vulnerable code resides within only one function [136, 137]. For example, Org A's decisions were largely based on vulnerability scan reports. However, the existing scanning tool was unable to filter the unpatched vulnerabilities resulting from legacy dependencies and exempted patches, which constrained accurate decision-making.

"These vulnerability numbers will go down by as much as half since [scanning tool] captures 2008 servers' vulnerabilities as well. I don't know whether we can do exceptions through the tool, like flag things that are legacy, to make the numbers reflect what we see." - BT1, M12

Increased Patch Load

As the organisation size grew, the number and diversity of systems also expanded which resulted in increased complexity in the patching. Hence, the practitioners often faced difficulties in keeping up with the patch release rate. Accumulated patch load due to the previous patch exclusions added to the challenges as they had to patch previously excluded patches in the following month. Correspondingly, an increased patch load led to more socio-technical dependencies creating additional constraints on coordination. Overall, it led to an increased risk of exposure to attacks as installing security patches was often delayed.

"There's just too much to check! We're dealing with 1500 servers, we don't have time to look at each patch for every server like, "Yeah, this one is right, this isn't. Which servers have interdependencies that can't be patched at the same time? Which server has which version of this software, that software..." - AT5, M14-dis

5.4.3 Breakdowns

In this subsection, we report the scenarios that exemplify the breakdowns resulting in the security patch management process from ineffective coordination of the socio-technical dependencies and related constraints.

Sudden Escalations to Patch Schedules

Security patch installation within the allocated patch window is critical in a mission-critical domain like healthcare to avoid unexpected service disruptions. Org A provided specific patch windows to Org B, usually 4 hours, to install the security patches on production servers (i.e., operating in hospitals) after agreeing with the customers. To adhere to the specified patch window while installing patches, Org B teams were required to plan well ahead to establish each patch's prerequisites, identify interdependencies, test the patches, and obtain management approval. As such, the miscoordination of these conditions would usually lead to unexpected escalations in the planned patch schedules. Given the critical nature of the healthcare operations, security patch installations going beyond the scheduled patch window resulted in devastating consequences such as life-threatening risks to critical patients from additional service unavailability even for a couple of minutes.

"We realised there's a need to do a sudden change of configuration for [ISP] servers at the time of patching, so our team had to escalate immediately to switch to manual patching because some servers needed Windows approved patches to be rolled out." - BT1, M12

Delays in the Organisation Approvals

All security patches needed to be approved by Org A's change manager a month before a patch installation to allow Org B teams to prepare in advance for installing a patch. However, we observed some delays in approvals from the Change Management team resulting in abrupt changes to patch installation plans such as shorter patch windows. Such unforeseen changes invoked changes to the dynamics of socio-technical dependencies resulting in breakdowns in the process. For example, having to install the emergency security patches in shorter patch windows warranted re-testing of patches to confirm that patch installation can fit into the shorter patch window, and obtaining reapproval from customers to avoid unexpected service interruptions.

"We were ready for the patch deployment. But this morning [Org A's] change manager said that the change hasn't been approved yet. So, if it isn't approved prior to the scheduled start time, we will have to reschedule it." - BT1, M10

Lack of Dependency Awareness from Localised Work Distribution

Org A and Org B teams had localised work distribution settings within their team structures. Org A teams were located in the same office and Org B teams were distributed across different offices in the same city. This structure led to the creation of a lack of dependency awareness of the task assignments and progression between teams. We observed this during status reporting in the meetings as some members were unaware of the tasks progressing in the teams. Their lack of understanding resulted in added challenges of coordinating the inter-team dependencies that inhibited measuring the progression of security patch management tasks.

"I believe [P1 from AT2] is working on this issue at the moment. Do you know when he's likely to get that done? - (BT1)
Not sure, but I would be hoping by next week. I would contact him and let you know." - AT2, M13

5.4.4 Mechanisms

This subsection presents a collection of strategies that have emerged from analysing the data; the studied teams practised these mechanisms to manage the dependencies while mediating the constraints.

Early Investigation of Interdependencies

Org A and Org B teams used patch meetings to discuss the findings from the investigations of the technical dependencies. Since the participants with diverse technical backgrounds and expertise attended the meetings, this configuration enabled knowledge sharing to collaboratively identify dependencies upfront. Failing to identify the dependencies resulted in scenarios of installed patches not working as intended, patches refusing to get installed resulting in rollbacks, and patches going beyond the allocated patch window during installation. Early identification of dependencies helped practitioners to coordinate the task dependencies among teams, and make timely decisions to address the usual problems such as raising support cases to vendors seeking expert advice and finding workaround solutions.

"From troubleshooting why the last two weeks' patches hadn't worked properly, we can realise that each patch needed to be rebooted at the beginning. Since we failed to do so, the current reboot may have applied the last month's patches." - AT5, M14

Collaborative Decision-Making

Accurate and timely decision-making is pivotal throughout the patching process. The studied practitioners used patch meetings as a platform to collaboratively decide about vulnerability risk assessment and prioritisation, and approval of the patch decisions and schedules. Collaborative decision-making helped the teams to maintain dependency awareness about the decisions and to plan the associated tasks with minimum impact of dependencies. For example, the decisions about patch exemptions involved a collective assessment of the requests. Thus, the awareness of patch exemptions helped the teams to plan their other patch schedules with limited dependencies to the exempted patches, keep track of the exempted patches in the month and organise the to-do patch list in the following month including them.

"The last item on the agenda is about the [s1] servers that are marked as excluded from patching. We need to decide if they are being exempt from our patching list this month or not because we haven't got the official confirmation whether we're doing the patching or if the [third-party] vendor is doing it again?" - BT1, M13

Other examples of collaborative decision-making involved selecting the optimum patch configurations based on organisational needs, and managing legacy software dependencies. In most cases, the decisions for legacy software dependencies revolved around the need for decommissioning or rebuilding legacy systems, when and how to do it, and how to patch them following the rebuilds. Decisions about the patch schedules for the approved patches helped the teams to coordinate the planning upfront and identify the need for *out-of-band (OOB) patching.* OOB patching refers to the need to allocate an additional patch window when some security patches require more installation time than the allocated patch window due to the compound dependencies involved.

Continuous Measuring of Progression

Org A teams measured the continual progress of vulnerability remediation through Org B's status reports in meetings and regular vulnerability scan reports. When the scans indicated an increase in the number of vulnerabilities present in the systems, the matters were discussed extensively to remedy the situation. The continuous measurement of the progression enabled the identification of the outliers such as missing patches resulting in the investigation of the causes and coordinating the associated stakeholder dependencies with the third-party vendors.

*"What is the status of internal [s1] server security vulnerabilities? - (AT5)
We're getting our regular scans to measure that. That one is progressing quickly. We will share the report next week" - BT1, M9*

Frequent Communication

Frequent communication appeared to be essential for the effective coordination of dependencies. It helped to erase boundaries between roles, teams, and organisations, and increase cohesion and trust between stakeholders. Teams used various communication mediums such as bi-weekly patch meetings, email, and Skype. Additionally, the studied practitioners held separate meetings to discuss critical and urgent matters that emerged in between patch meetings or when patch meeting discussions were dragged beyond the allocated time. Patch meetings were the most preferred communication medium as the teams felt more comfortable with direct communication. Communication during patch meetings facilitated collaboration, knowledge sharing, and information exchange about technical and socio-technical matters affecting the patching process, for example, upcoming patch schedules, changes to patching plans such as out-of-band patch schedules, and vendors' patch release information. Regular patch meetings benefited the teams in numerous ways such as allowing visibility into task progression and assignments, staying proactive to potential issues about critical security vulnerabilities, and effectively coordinating security patch management activities.

"[Security Advisor shares the vulnerability remediation progress report on screen] We were averaging 75 high-risk vulnerabilities per server back in 2016 when I joined. As you can see now, we're down to 5 per server. Given the mix of environments we

are dealing with, this is amazing. You can see that the frequent patch meetings making a big difference!" - AT5, M10

Load Balancing

An important strategy employed by the studied teams to coordinate patch schedules was load balancing. It was used to balance the patch load in servers at any given patch installation time. Balancing the server load helped reduce service interruptions. Patching dozens of servers at the same time significantly increased the risk of system failure as all the servers go offline at the same time during reboots. Load balancing, on the other hand, helped to run the critical medical services concurrently on another server while the desired server(s) is being rebooted. However, the presence of technical dependencies created difficulties in load balancing. In particular, for instances with one-to-one dependencies such as (A $\rightarrow$ B), the practitioners had to rigorously analyse the interdependencies before planning the load on servers to avoid unexpected system downtime.

"Before we started with the load balancing, we patched 50 servers one night, and just two the next night. So, I suggested we plan to load balance. But there's a lot to manage, especially when we have systems like system A that is redundant to B, and oops! we accidentally took both of them down at the same time to patch." - AT5, M10-dis

Centralised Vulnerability Risk Assessment

Regularly performing vulnerability risk assessment and prioritisation was necessary as it could potentially differ from that of the vendor's assessment based on the organisation's environment. It aided practitioners to plan well in advance to promptly respond to critical security vulnerabilities. To regularly monitor security vulnerabilities, the teams devised a centralised role in Org A's Security team (Security Advisor) responsible for scanning and categorising the vulnerabilities based on teams' ownership. Having a centralised structure helped maintain consistency in vulnerability risk assessments across teams as well as reduce delays in vulnerability assessment and prioritisation decisions. Additionally, frequent comparisons with the previous scans assisted with evaluating the vulnerability remediation performance.

"[Security Advisor] gets the global rating of a vulnerability risk and re-assess it to see if it's critical to us and how it can be exploited. For medium to low risks, we patch in the next cycle, but if it's critical or we're under attack, we'll patch within 48 hours." - AT5, M10-dis

5.5 Discussion

In this section, we discuss our theory in comparison with the prior related work and elaborate on the broader implications of our theory for practitioners and researchers.

5.5.1 Comparing to Related Work

Following Glaser's advice [35, 36], we compare our theory with the existing literature. The prior related work on this topic [2, 22, 54, 103, 106] primarily focuses on the coordination and collaboration challenges but does not provide in-depth details of the causes of coordination challenges or the potential strategies to address the coordination challenges leading to delays in applying security patches. A few studies [59, 125, 126] investigate the social dependencies concerning the involvement of multiple internal and external stakeholders. Another set of studies [24, 60, 61, 103] exclusively focuses on vendor dependencies that might arise from shared vulnerabilities in software code. In particular, their main focus remains on optimising patch management by obtaining an equilibrium of an organisation's patch cycle with a vendor's patch-release cycle to minimise cost. An important observation is the absence of *theories* that focus on the socio-technical aspects concerning security patch management delays in contrast to quantitative *models* and *frameworks* [138]. This is an important point as the theories provide *"basic concepts and underlying mechanisms which constitute an important counterpart to the knowledge of passing trends"* [139]. In contrast, our theory derived from the gathered data differs from these existing works in several ways as it:

- Explains the role of coordination in security patch management as incorporating a multi-dimensional nature across four interrelated dimensions in contrast to focusing on one type such as vendor dependencies reported in [24, 60, 61, 103];
- Explains the socio-technical dependencies that create the need for effective coordination in security patch management going beyond reporting just the challenges with lack of coordination in the process [59, 125, 126];
- Explains the constraints that hinder the coordination of interdependencies and shows scenarios of breakdowns resulting from ineffective coordination of causes and constraints;
- Suggests strategies that can be used for effective coordination in the process;
- Offers a comprehensive overview of the impact of (in)effective coordination in timely security patch management in a mission-critical domain like healthcare;
- Presents a theoretical model for future research, and,
- Provides guidance to practitioners to overcome patching delays and increase confidence in their decisions.

5.5.2 Implications for Practitioners

The reported theory can be used to gain an in-depth understanding of the significance and impact of the role of coordination in security patch management. Practitioners can use this understanding to realise their roles and responsibilities in ensuring coordination effectiveness across different dimensions of the role of coordination. Moreover, practitioners can use the theory as a guide to identify the related dependencies and how they might affect their security patching process. Further, we have observed that adopting these coordination mechanisms has resulted in a reduction in patching delays and the number of unpatched security vulnerabilities in Org A's systems. Hence, our findings may also be useful in exploring the suggested mechanisms in other organisational settings. Additionally, practitioners can benefit from the early detection of constraints and breakdowns to avoid failures and delays in security patch management.

5.5.3 Implications for Researchers

Given a Grounded Theory study is considered to produce a "mid-ranged" theory based on the contexts studied [44], other researchers can carry out an extension through future research including a more detailed analysis of the present dimensions, new dimensions discovered, or different contexts. Context-specific research investigating how the role of coordination is impacted by contextual factors can result in useful models of coordination in patch management [124, 140]. For example, organisation-level dependencies may not directly apply to small organisations where security patch management is usually handled by one team. Future studies can also investigate the effectiveness of the coordination mechanisms and the context in which they should be employed. The impact of organisational policies is often cited as one of the dominant socio-technical challenges in security patch management [2, 20, 54, 59, 125, 141]. Similarly, future studies can explore how organisational culture affects the role of coordination. Another possibility is to employ the findings in large-scale surveys to evaluate the theory and identify variations in different organisation settings such as in DevOps processes.

While this study is based on data collected from security patch management, the findings can be directly beneficial to software development research. This is because patch application is inherently dependent on patch development. An important point to note from our theory is the need to consider the *socio-technical* aspect intrinsic to patch management when developing patches (e.g., future work similar to Li et al. [2]). We show that early identification of the dependencies is the key to avoiding coordination-related patching delays and failures, but lack of automation support presents a key constraint as previously mentioned for timely identification of the dependencies [100]. Therefore, there is a need for research on how automation support can be employed in dependency detection in patch development and management. The findings also highlight the important need for further research and development for advanced security patch management tools. For example, scanning tools can be enhanced to customise the software dependencies such as excluding exempted patches to provide real-time feedback that assists practitioners with accurate decision-making. Accordingly, an evidence-based understanding of the role of automation in security patch management would be beneficial to identify other areas of improvement relating to advancing automation support in the process. Furthermore, our theory provides an in-depth understanding of how the role of coordination impacts a mission-critical domain, particularly, healthcare. An understanding of the causes of unexpected service interruptions can help researchers to devise strategies to avoid such downtime. While the research into dynamic software updating (DSU) [63] attempting to address this issue is progressing, our results can be useful information for future research that investigates the effectiveness of the developed strategy in mission-critical contexts.

5.6 Threats to Validity

In this section, we reflect upon the potential threats to the validity of this study and discuss how we have mitigated them.

A Grounded Theory study does not affirm generalisation as the theory formulation is pertinent to the studied context [35, 45]. The context of this study is limited to the cases studied in security patch management in the domain of healthcare. Nevertheless, we believe that our theory can be recreated in other contexts and modified.

In terms of data representativeness, our data collection is limited to the observations of patch meetings, post-meeting discussions, and analysis of meeting minutes and patch mailing threads as described in Section 2.2.3 (Data Collection). We acknowledge that more data sources such as interviews and surveys can be incorporated into future studies to increase the scope of the analysis and verifiability of the theory [142].

While employing Grounded Theory procedures permits the data analysis to be grounded in collected data, there is a threat of subjectivity of the data analysis referred to as the "uncodifiable step" [143, 144]. To alleviate this threat, we regularly held internal discussions on the emergent findings throughout the study as described in Section 2.2.3 (Data Analysis). In addition, the findings were further cross-checked with a senior member of Org A's security team to ensure we have accurately interpreted the theory from the observed practices.

The verifiability of a grounded theory can be deduced from the robustness of the research method, and evidence of theory formulation from its application [44]. To confirm verifiability, we have described our application of the Glaserian version of GT in detail (Section 2.2.3) and included quotations from the underlying data in the findings (Section 5.4). These details provide evidence of how our theory meets the GT evaluation criteria: the generated categories *fit* the underlying data (see Figure 2.5); the theory can *work* as it explains the main concerns of the participants in patch meetings; the theory has *relevance* to the domain of security patch management; and the theory is open to *modification* based on future studies in different contexts [36, 45].

5.7 Chapter Summary

In this Chapter, we present a grounded theory of the role of coordination in security patch management. The developed theory explains the effects of coordination in the security patch management process across four interrelated dimensions namely *causes, breakdowns, constraints,* and *mechanisms.* Our theory is based on a longitudinal Grounded Theory study of 51 patch meeting observations involving 21 industry practitioners in two organisations in the healthcare domain over a duration of 9 months. We provide grounded evidence that the role of coordination in security patch management represents a core concern, contrasting with a perception among the software engineering community that automation and tooling alone can be sufficient to achieve success in timely security patching and highlight the need to have a delicate balance between the socio-technical concerns such as coordination and automation to reduce delays in security patch management, which is often unrecognised in the existing literature.

Overall, besides providing a holistic understanding of the role of coordination in security patch management that is based on empirical evidence and grounded in practice, our study is the first attempt to investigate in-depth the socio-technical aspects of security patch management in the mission-critical healthcare domain. The theory provides important insights for practitioners to avoid patching delays and failures and enhance confidence in their decisions, and for researchers to shape their work on patch development to address the practical concerns in patch application. The findings can also be used for developing the next generation of AI-enabled tools for supporting effective coordination of the security patch management process.

Chapter 6

Automation in Security Patch Management

Related publication: This chapter is based on our paper titled *"An Empirical Study of Automation in Software Security Patch Management", published in the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2022 (CORE Ranking A*) [145].*

In the previous Chapters 4 and 5, we observed that tools limitations and lack of suitable automation have largely contributed to delays in security patch management and caused constraints in effectively coordinating the process interdependencies leading to further delays. Additionally, several other studies have also shown that automation support for different activities of the security patch management process has great potential for reducing delays in applying security patches. However, as described in Chapter 5, it is also important to understand how automation is used in practice, its limitations in meeting real-world needs and what practitioners really need, an area that has not been empirically investigated in the existing software engineering literature. This chapter addresses this gap by reporting an empirical study aimed at investigating different aspects of automation in security patch management using semi-structured interviews with 17 practitioners from three different organisations in the healthcare domain. The findings are focused on the role of automation in security patch management providing insights into the as-is state of automation in practice, the limitations of current automation, how automation support can be enhanced to effectively meet practitioners' needs, and the role of the human in an automated security patch management process. Based on the findings, we have derived a set of recommendations for directing future efforts aimed at developing automated support facilitating human-automation collaboration for addressing the current gaps and user needs for timely security patch management.

6.1 Introduction

Failure to install security patches to the identified software vulnerabilities usually results in disastrous consequences as a result of successful cyberattacks that particularly target software systems in mission-critical domains such as healthcare [10, 11]. Several studies [1, 110] report that available but not installed security patches are one of the main reasons for such successful cyberattacks, for example, the Heartbleed security bug that remained unpatched in systems for several years [146]. In our study reported in Chapter 4 to understand the reasons for delays in applying security patches in practice, we identified that the limitations in current tools and lack of suitable automation leading to the need for human intervention in several activities of the security patch management process give rise to many delays in security patching. Further, in our attempt in Chapter 5 to understand and provide a solution framework addressing the most prominent cause of the delays, i.e., ineffective coordination in the process, the lack of automation support presented a major constraint in effectively managing the socio-technical interdependencies leading to further delays.

Additionally, like other areas of software engineering [147, 148], it has been claimed that automation has great potential for significantly improving the effectiveness and efficiency of the security patch management process to minimise the delays in applying available security patches [108]. That has led to increased enthusiasm for devising and integrating automation support in different activities of the security patch management workflow. For example, automated approaches/tools for patch information retrieval [149, 150], vulnerability scanning [151, 152], assessment and prioritisation [151, 153], patch testing [25, 104], patch deployment [108, 154] and post-deployment patch verification [125, 155]. Despite these research efforts producing promising results for streamlining different activities of security patch management by providing automated solutions, delays in real-world security patch management still exist [23, 156]. We assert that it is important to empirically understand the current status of automation, the real-world issues facing the current automation and the practitioners' needs for improved automation support to develop suitable automated approaches and tools for effectively reducing delays in security patch management. To the best of our knowledge, there is little empirically known about these aspects of automation for security patch management.

Motivated by the need to empirically understand the role and user needs of automation for reducing delays in security patch management, in this chapter, we conducted an empirical investigation using in-depth semi-structured interviews with 17 practitioners from three organisations in the healthcare sector. We followed the Straussian version of the Grounded Theory data analysis method [34] to systematically uncover the findings in a bottom-up approach. Our study findings contribute to the state-of-the-art understanding of research and practice by:

1. Providing an evidence-based understanding of the as-is state of automation in security patch management by describing the current manual and automated tasks in the process;
2. Identifying the limitations of the current automation in practice;
3. Informing how automation support can be enhanced in effectively meeting practitioners' needs;
4. Explaining the role of the human in security patch management automation identifying the tasks and points in the process in which automation needs to facilitate a human-in-the-loop approach; and

5. Discussing the future avenues of research and providing a set of recommendations that can guide tool designers and researchers to address the identified gaps and user needs towards reducing patching delays.

Chapter Organisation: Section 6.2 summarises the related work. Section 6.3 presents the research questions of the study and describes the research method. Section 6.4 reports the results of the study. In Section 6.5, we discuss the study findings and their implications for practice and research and present our recommendations for future work. The limitations of the findings are presented in Section 6.6. Finally, Section 6.7 concludes the study.

6.2 Related Work

Although no work has specifically investigated the role of automation in the process in practice, the previous research has invested efforts aimed at providing automation for security patch management activities to bring efficiency into the patch management process and reduce delays [25, 104, 108, 125]. Most of the previous efforts focus on automating specific tasks in security patch management, rather than supporting the end-to-end process [54]. A few studies have proposed solutions for automated patch information retrieval from multiple sources [149, 157], customised information filtering [149, 158], information validation [150] and patch download and distribution [150]. For vulnerability scanning, a central platform integrating the scan results from multiple information sources has been suggested [152, 151]. Towards vulnerability risk assessment and prioritisation, researchers [151, 153, 159] have proposed customisable tools/approaches for vulnerability risk analysis in line with the industry standard, the Common Vulnerability Scoring System (CVSS) [86].

Towards patch testing, a set of tools have attempted automated detection of faulty [25, 104] and malicious patches [160, 161]. However, only limited attention has been given to recovering from crashes that result from faulty patches with minimum disruption. Several studies focus on automating patch deployment [157, 162]. To minimise downtime and service disruptions, several approaches have been developed; such as dynamic software updating (DSU) [63], JIT patching [108] and instant kernel updates [154]. For post-deployment patch verification, automated approaches for verification of patch deployment [125], detection of exploits [163], and repair of past exploits [155] have been presented. An important observation is that only a few studies have rigorously evaluated the proposed solutions in real-world contexts [54, 69]; an implication of little empirical understanding of how well the solutions have addressed the practitioners' needs.

Meanwhile, another set of studies has investigated the socio-technical aspects relating to security patch management, particularly the patch management process and its challenges [2, 20], the role of coordination and collaboration [22, 56, 59, 103], system administrators' practices, behaviour and experiences [2, 20, 25, 109, 110], and reasons and mitigation strategies for delays in security patch management [100]. The understanding of the socio-technical aspects is considered important as security patch management is essentially a socio-technical endeavour whereby humans and technologies continuously interact for enabling team members to effectively coordinate and collaborate using the available tools [54]. An important observation is the highlighted need for human involvement in the patch management process despite the advancements in automation to reduce patch management delays [2, 20, 25, 54, 104, 105, 152]. However, why human involvement is needed in process automation and the desired balance between automation and human involvement has not been investigated.

Our work distinguishes itself from the above-mentioned works as it purports to provide a holistic understanding of the role of automation in security patch management grounded in evidence gathered from practitioners. We explain how practitioners have integrated automation into the security patch management process by describing the automated and manual tasks in the workflow, the limitations of the current automation in meeting practitioners' needs, how automation support can be improved to assist practitioners, and the roles of humans in process automation, i.e., why and where is human involvement needed. Given the criticality of timely security patch management, these findings will provide a solid foundation to propose practical solutions that can address the limitations of current automation support for timely security patch management.

6.3 Research Method

This study was aimed at understanding the role of automation in security patch management in practice. We conducted a qualitative study using semi-structured interviews that are expected to enable us to gain a better understanding of real-world practices through practitioners' perspectives. To achieve this goal, we developed the following research questions (RQs):

RQ4.1. What is the as-is state of automation in security patch management?
This RQ focuses on understanding the current state of automation in practice by identifying which security patch management tasks are currently supported by automation.

RQ4.2. What are the limitations of current automation?
This RQ focuses on identifying the limitations of the current automation/tools to understand how well the current automation meets practitioners' needs.

RQ4.3. How automation in security patch management can be enhanced to support practitioners?
This RQ aims at understanding the kind of support practitioners expect from improved automation. This knowledge is important to design future tools to effectively meet practitioners' needs.

RQ4.4. What is the role of the human in security patch management automation?
This RQ focuses on understanding the roles humans play in security patch management automation and the reasons demanding human involvement. Such understanding can help in identifying the tasks for which future automation can be designed to facilitate a human-in-the-loop approach.

We collected data through semi-structured interviews with 17 practitioners from Org A, B and C, and used Strauss and Corbin's GT version (Straussian GT) [34] for data analysis. The details of the data collection and analysis were described in Section 2.2.2.

6.4 Results

This section reports our study's findings for different aspects of automation in security patch management. We describe the as-is state of automation in the security patch management process in Section 6.4.1, limitations of existing automation in Section 6.4.2, practitioners' desires for enhanced automation support in Section 6.4.3, and the role of the human in process automation in Section 6.4.4.

6.4.1 As-Is State of Automation in Security Patch Management

Our findings to answer RQ4.1 reveal that whilst some tasks of security patch management have full or partial automation support, several of the tasks are carried out entirely manually. Figure 6.1 summarises the current state of the automation reported by the participants characterised by each process phase (shown in Figure 3.1(b)). We describe how practitioners have integrated automation into the main tasks and the tools below.

Patch Information Retrieval from Third-Party Vendors

Most patch deployment tools automatically retrieve the patches from third-party vendor sites and distribute them to the patching towers. However, learning about new patch releases is a manual task whereby practitioners proactively search for patch release information using various sources such as direct vendor calls, security professional mailing lists, community Slack channels and online forums.

"The [vendor] schedules a call monthly for his premium customers. Then there's a security professionals' community where we share intelligence on incidents happening. For example, mid-last year, a Microsoft patch caused problems on the printer and it was highlighted in that forum. So at least we had some forewarning to minimise the business impact." - P9

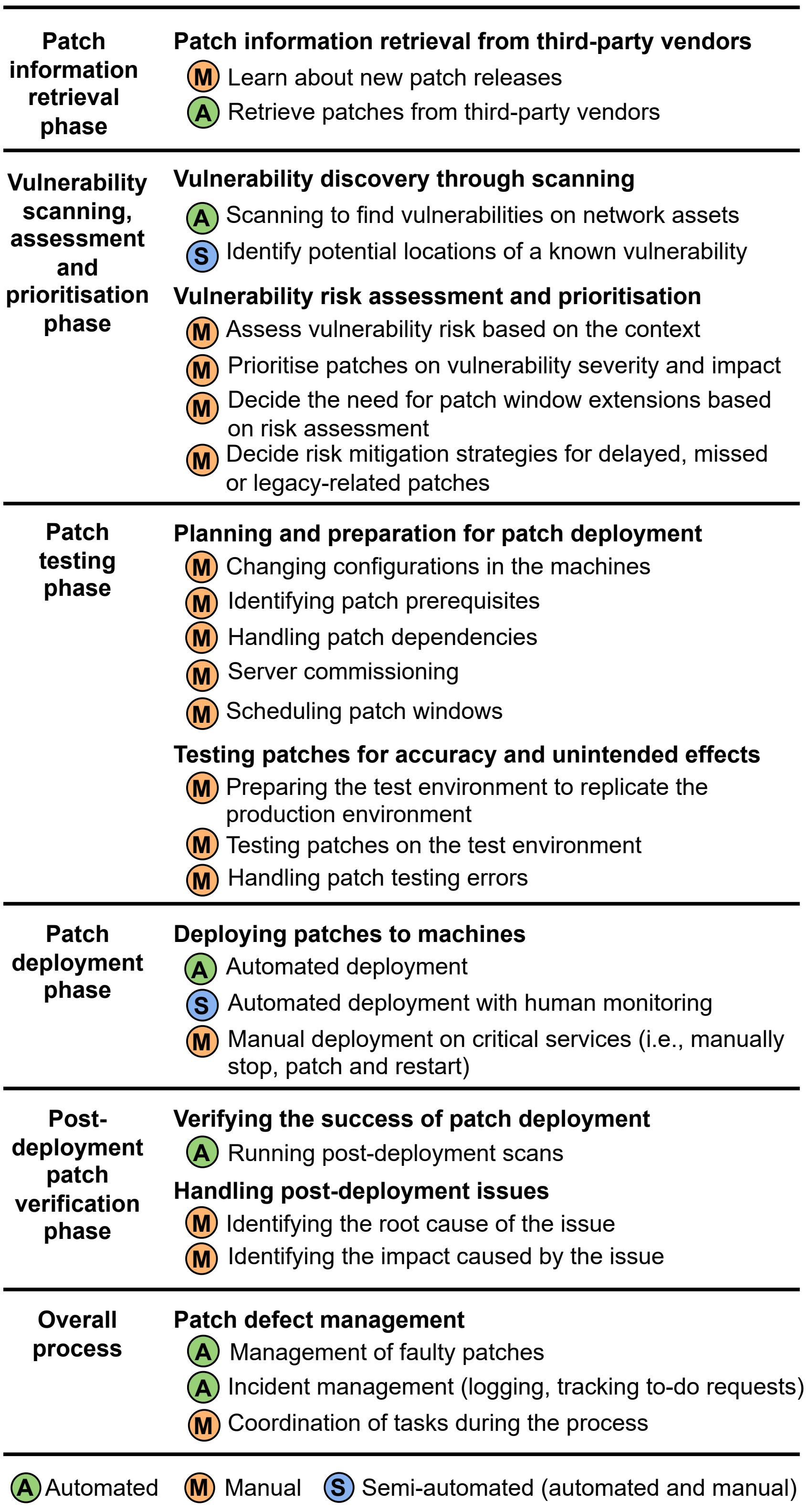
Vulnerability Discovery through Scans

A dedicated role is established for centralised vulnerability scanning for increased reliability in vulnerability risk assessment and reducing delays in patch decisions. Vulnerability scanning involves performing scans to locate the assets on the network and finding vulnerabilities in the assets which are automated using the tool "Tenable.sc" [164]. The practitioners schedule the vulnerability scanner (e.g., Nessus [165]) to perform the scan and retrieve the results through automatically generated reports. However, identifying the potential locations of a known vulnerability sometimes requires manual effort; for example, writing a script to query what DLLs or java files the machine had in certain locations to find the locations of the Log4j vulnerability [166] as echoed by P13:

"We wrote a small batch file to query what DLLs or java files the machine had in certain locations and pushed out to all the servers and executed it without the tool. It created a results file locally on the machine and that was scraped and analysed later." - P13

Vulnerability Risk Assessment and Prioritisation

Based on the understanding of existing system vulnerabilities through the scan reports, practitioners manually perform the risk assessment and prioritisation to decide



Patch information retrieval phase	**Patch information retrieval from third-party vendors** Ⓜ Learn about new patch releases Ⓐ Retrieve patches from third-party vendors
Vulnerability scanning, assessment and prioritisation phase	**Vulnerability discovery through scanning** Ⓐ Scanning to find vulnerabilities on network assets Ⓢ Identify potential locations of a known vulnerability **Vulnerability risk assessment and prioritisation** Ⓜ Assess vulnerability risk based on the context Ⓜ Prioritise patches on vulnerability severity and impact Ⓜ Decide the need for patch window extensions based on risk assessment Ⓜ Decide risk mitigation strategies for delayed, missed or legacy-related patches
Patch testing phase	**Planning and preparation for patch deployment** Ⓜ Changing configurations in the machines Ⓜ Identifying patch prerequisites Ⓜ Handling patch dependencies Ⓜ Server commissioning Ⓜ Scheduling patch windows **Testing patches for accuracy and unintended effects** Ⓜ Preparing the test environment to replicate the production environment Ⓜ Testing patches on the test environment Ⓜ Handling patch testing errors
Patch deployment phase	**Deploying patches to machines** Ⓐ Automated deployment Ⓢ Automated deployment with human monitoring Ⓜ Manual deployment on critical services (i.e., manually stop, patch and restart)
Post-deployment patch verification phase	**Verifying the success of patch deployment** Ⓐ Running post-deployment scans **Handling post-deployment issues** Ⓜ Identifying the root cause of the issue Ⓜ Identifying the impact caused by the issue
Overall process	**Patch defect management** Ⓐ Management of faulty patches Ⓐ Incident management (logging, tracking to-do requests) Ⓜ Coordination of tasks during the process

Ⓐ Automated Ⓜ Manual Ⓢ Semi-automated (automated and manual)

FIGURE 6.1. The as-is state of automation in each phase of the security patch management process. The **bold** text in the right column denotes the main tasks in each phase and the sub-tasks are listed underneath.

on patching. This is because the existing scanners fail to incorporate the organisational context and needs which are required for an accurate representation of the risk.

"The security risk assessment is based on multiple factors, for example, the CVSS score and how easy it is to exploit within our organisation. We also look at if there's a particular exploit in the wild that will put more urgency on applying this patch." - P9

Patch prioritisation is based on the vulnerability severity and impact. The need for patch window extensions is decided based on the risk assessment as some security patches could have a wide impact on several other services, for example, Log4j vulnerability [166]. The security team then delegates the vulnerability remediation tasks to the relevant teams based on ownership. In cases such as delayed patch releases, missed patches in the previous cycle or legacy software-related vulnerabilities, practitioners collectively discuss suitable risk mitigation strategies (e.g., building firewalls, changing registry settings, configuring OS to harden against vulnerabilities, server clustering and upgrading the legacy systems). The vulnerability remediation progress is evaluated by comparing the vulnerability scans over time for strategic planning to improve the process efficiency.

Planning and Preparation for Patch Deployment

In preparing for patch deployment, the practitioners engage in a set of tedious manual tasks involving changing configurations in the machines, identifying the patch prerequisites, handling patch dependencies, server commissioning (i.e., the process of building servers ready for deployment), and planning the patch schedules. In this, identifying patch prerequisites and scheduling tasks are perceived as the most challenging, necessitating significant manual overhead.

A lot of security patches generally have prerequisites that demand a set of configurations (e.g., registry changes, group policy object (GPO) changes, preparation package installation) to be configured prior to patch deployment for the patch to take effect during the deployment. Identifying prerequisites is considered a daunting manual task because they are identified by extensively reading through the patch release notes, knowledge base (KB) articles or information on the vendor's website. Besides, it is prone to missing some prerequisites, which results in deployment errors. As a result, significant effort is needed to redeploy the patches through out-of-band (OOB) patching (i.e., outside the scheduled windows). On top of that, identifying the cause of deployment errors as missing prerequisites is an equally challenging task since the tools are incapable of detecting the issue.

On the other hand, scheduling a patch window involves a series of tasks of extensive planning and organising with external stakeholders. Some participants use a third-party integrated risk management solution (e.g., Archer GRC [167]) for organisational governance and compliance with industry regulations; however, configuring the tool is a laborious task. The others manually schedule patches using an Excel spreadsheet. Due to the vast information and lack of automation support, identifying which servers need which patches, approving the changes and, load balancing (i.e., balancing the patch load on servers to reduce service interruptions) require arduous administrative overhead and a lot of discussions during patch meetings.

"I do a lot of manual work to get the tool ready for patching. Every month, I export the information of 1800 servers from CMDB to a large sheet with 27 columns and go through it to find the windows. It's quite a complex and time-consuming task, and if I make a mistake with one formula, it can affect the whole process." - P13

Testing Patches for Accuracy and Unintended Effects

For most participants, patch testing is a challenge as it is a time-consuming and strenuous manual process. The participants perform testing in dedicated testing environments (e.g., “Dev", “Test" and “Pre-prod") before rolling out the patches into production (“Prod"). However, due to resource constraints, client machine patching typically employs a staged deployment with a 10:30:60 rollout (i.e., systematically deploying patches to clusters of 10%, 30% and 60% of machines without testing). Preparing the test environment to replicate the “Prod" is also a laborious task due to shared access resulting in too many interdependencies. Additionally, patch testing errors are handled manually, which generally occur owing to a broken dependency. As a result, the practitioners spend hours organising the testing with other teams that owned the dependent services.

“If we find that the patch has an interdependency with an application, then my team has to work internally with the related teams to deploy the patch from Dev, Test, Pre-prod through to Prod which needs a lot of coordination." - P11

Additionally, errors during patch testing are also handled manually. Generally, errors occur because of a broken dependency. When the practitioners are unable to fix the error, they aim at finding workarounds to resolve the vulnerability risk without installing that particular patch or replan the testing.

Deploying Patches to Machines

Patch deployment is largely automated, however, in critical scenarios, it is shifted to semi-auto or manual depending on the level of human intervention needed. Automated deployment, referred to as auto windows by the participants, is when the patch deployment and rebooting of the machine are done with no human interference. The participants use several third-party patch management software like Ivanti (server) [168], SCCM (server) [169], WSUS (server) [170], and VMware Workspace ONE (desktop) [171] for deploying patches based on their needs and budget. On average, 67% of the servers (i.e., 1200/1800) are patched automatically. However, the success of automated deployment relies on the accuracy of server information fed into the tool and tool configuration.

In an issue during automated deployment (e.g., the application not automatically restarting upon the server reboot), it is switched to semi-auto allowing practitioners to log into the machines to monitor the patching job while trying to investigate the issue. Semi-automatically patched machines are assessed for any recurring errors in the next patch cycle and moved to the auto group based on the success. On average, 11% of the servers (i.e., 200/1800) are patched semi-automatically.

Meanwhile, 22% of deployments, i.e., 400 of 1800 servers on average rely on decision-making that is extremely challenging or impossible to be automated; hence those systems are patched manually. One example is when a patch is deployed to a critical server that provides life support to patients or an ambulance service. For example, *“we can’t automatically shut down a system in the middle of an ongoing mission. So we tell the client we need to reboot the system and it’ll be out for X time. They might say no, we’ve got four missions going on. So in such cases, there’s no way automation is ever going to work there."* - P6

Verifying the Success of Patch Deployment

The main verification technique used is scanning the systems post-deployment; the commonly used tools are Ivanti [168] and Tenable.sc [164]. The practitioners run a post-deployment scan overnight which generates a report presenting the summary statistics. The report is then analysed to identify the success of deployment and if any patches have been missed. Interestingly, the participants reported that they target 99% coverage of the server fleet as the remaining 1% is usually never completed due to various reasons such as legacy systems. Other verification techniques are monitoring the system performance for a few weeks and collecting user feedback after an issue has been identified post-deployment.

Handling Post-Deployment Issues

Handling post-deployment issues involves two manually performed activities: identifying what caused the issue and how wide the impact it has caused. Some common troubleshooting incidents are identifying why the patched systems are not (re)starting or performing as expected and why some patches have been missed during the deployment. Participants use various troubleshooting mechanisms such as collecting system log information, gathering end-user feedback, obtaining information from online sources (e.g. forums, news, mailing lists), investigating for errors in the process, patching tool and production environment, recreating the issue to monitor its symptoms (e.g., when and how was the issue created), debugging error messages, shutting down service to identify the impact on dependent systems, and reaching out to the software vendor for assistance. Identifying the impact of the issue is important for deciding on the appropriate mitigation actions. The impact is initially measured in terms of the number of client complaints received and vendor announcements. The gathered information is then analysed to calculate the risk and identify the potential workarounds (e.g., rollback) to enable service continuity.

"It's a real pain in the bum to have patches fail because it can be quite intensive. We've had months where we had to roll back 20, 30, and 40 servers. And that's when it gets really ugly because the rollbacks are expensive like a full virtual server restore." - P15

Patch Defect Management

Since there is no end-to-end solution that caters to the needs of all phases in the process, the participants use multiple tools to handle different tasks in the workflow. To manage the patch defects identified during patch testing through deployment to verification, participants use an in-house application life cycle management (ALM) solution. Incident management (i.e., logging and management of to-do patching requests) is supported by different tool choices such as third-party IT Service Management tools (e.g., Marval [172]) and in-house solutions. However, the coordination of tasks between teams is primarily done manually. General communication around patch management is through regular patch meetings, email and Microsoft Teams chats while keeping track of task progress is achieved through Word documents and Excel spreadsheets.

6.4.2 Limitations of Current Automation

This subsection presents the limitations of the current automation/tools to answer RQ4.2. An understanding of the current solutions' limitations is important for identifying the areas of improvement for addressing the identified limitations as discussed in Section 6.5.

Limited Support for Dynamic Environment Conditions

Whilst many vulnerability management tools claim to have supported automated vulnerability assessment, they do not capture the dynamic context factors in vulnerability assessment leading to an inaccurate risk assessment. That is why practitioners are required to do the heavy lifting for risk assessment incorporating the organisational context. In addition, the current automation lacks adequate support for accommodating sudden changes in the schedules, for example, out-of-band (OOB) patching (i.e., outside the scheduled windows). Whilst many participants desire automation support to adapt to unforeseen changes, some believe that the coordination among different stakeholders and decision-making in such cases depend on human intuition which is difficult to be automated.

"Automation, while great and saving time, also reduces the chance for special requirements, interaction and coordination with the business when things suddenly change. We have to work out what we can patch automatically and what still requires coordination with the business and all those things." - P14

Lack of Proper Support in Process Workflows

Limited support for patch deployment preparation warrants significant effort for reading through release notes to identify patch prerequisites. Correspondingly, limited support for handling patch dependencies seems to be a common complaint. The lack of a holistic view of the system interdependencies requires practitioners to spend a significant amount of time and effort in identifying the dependencies during testing and troubleshooting deployment issues. The absence of automation support to handle legacy software dependencies creates further complications, often leading to delays in restarting services to avoid system breakdowns. Another drawback is the lack of support for detecting the need for multiple reboots. Although the tools are capable of automatically executing the scheduled reboots, identifying how many reboots are required is a manual task.

The current tools do not support identifying and remedying the service interruptions caused by incompatible dependencies during deployment. As a result, practitioners are forced to adopt effort-intensive workarounds to minimise resultant service disruptions. A lack of real-time report generation detailing the deployment errors leads to a lack of interpretability in verification leaving practitioners to spend hours troubleshooting the root cause of an issue.

"Some tools are chosen because of the practicality, not because it's the evergreen solution. Sometimes automatically deploying a patch becomes impossible when the application does not restart correctly due to a broken dependency. So, we often do a very tedious manual process to stop, patch and restart the services." - P2

Lack of Accuracy of Output

Another deficiency is missing information (e.g., skipping some patches) in scanning leading to incorrect vulnerability reports. According to P4, discrepancies in the scan reports can be caused by a failure in running the scan completely or some information may be missing from the scan report. Inaccurate scan reports thus result in inaccurate vulnerability assessment leaving the system exposed to a myriad of attack opportunities. Lack of capacity to detect mid-cycle patch releases (i.e., superseding patches released during the patch cycle) is another limitation that leads to false positives in the scan reports.

"In cases of superseded patches, the tool reports them as missing patches. So we had to go through Microsoft's catalogue to find out that those patches have superseded the previous patches and notify the security team that this is what has happened." - P5

Lack of Scalability in Tool Design/Architecture

The lack of a unified platform to deploy patches to heterogeneous environments (i.e., multiple operating systems, dependent applications, etc.) is an important infrastructure limitation of existing tools. Microsoft SQL is a classic example that results in great annoyance for practitioners, forcing them to shift to manual deployment or juggle between several tools leading to often missing out on patches during deployment.

"The tool has only a finite number of products that it can patch. Imagine the trouble when I have hundreds of products but it can only patch half. It's not just the OS that you patch, you've gotta patch Office, Adobe drivers and whatnot, has it got WinZip? Creative Writer? or other third-party Oracle, Java, you know, there are so many little things to think about. " - P10

Another concern is the performance limitations in terms of the lack of capacity to run parallel deployment jobs and execute multiple reboots on different servers simultaneously. These limitations demand practitioners to spend a lot of effort in careful planning to cope with the time and service availability constraints.

"The tool at times has failed to launch two new jobs at once so we've had to stagger the jobs. In other words, there wasn't enough IO shoot enough jobs out in a given window." - P15

Service Disruptions During Patch Deployment

A key limitation of the current automation is the service downtime resulting from reboots forcing practitioners to rely on workarounds like server clustering and failovers. Despite the constant struggle to minimise service disruptions, the workarounds also require extensive effort for planning and execution. Similarly, the necessity to do multiple reboots in some cases is a frequent frustration given the narrow patch windows and extended service interruptions.

"It's very challenging as the application is unable to support redundancy or high availability. If you take one server down then the technology should be able to continue to run." - P2

Lack of Usability

The notifications indicating an error in the current patch deployment tools do not suffice for the level of understanding required for participants to detect errors. The lack of meaningful error messages leads to insufficient information for troubleshooting deployment failures causing practitioners to spend significant time and manual effort "*finding where to look for what*" (P10).

"It's scary when something breaks after patching because no one knows where to start looking. So someone has to look at the error messages to understand what it means. How do you automate something if you don't know what you're looking for?" - P10

6.4.3 Practitioners' Needs for Enhanced Automation

This section answers RQ4.3, the desires expressed by the participants in response to the interview question *"what tasks do you wish would have been better supported by the technology and how?"*. We raised this question to better understand what practitioners really need to fill the gaps in the current automation to effectively meet their needs.

Automation Support for Patch Information Management

The participants desire a single platform for retrieving trusted patch information from multiple sources covering new patch releases, mid-cycle releases, delays in patch releases, and potential patch adverse effects. Such a platform would assist them in making informed decisions about the patch application, the level of testing required and finding workarounds for potential adverse effects.

Additionally, the participants wish that a system provides an analysis of the potential impact of new patches based on the patch information retrieved from external sources. P3 recalled a situation wherein they spent significant time searching through public forums to find out about a faulty patch release.

"There was a Windows OS security patch rolled out this week that prevented Windows 10 desktops from being able to print on Windows 2003. The patch didn't say exactly what it was doing, just that it's a security patch, but behind the scenes, it has changed the protocol." - P3

Central Platform Integrating Vulnerability Scanning and Risk Assessment

The participants reported the need for enhanced filtering and customisation on vulnerability scanning that enables better filtering and information sorting to identify outstanding vulnerabilities at a glance. Additionally, the compiled wish list includes the ability to easily search through a myriad of vulnerabilities to track the remediation status and generate scan reports with better visualisations as the current report only provides summary statistics in Excel spreadsheet format. Another need is the capability to integrate vulnerability scanning and assessment reports with other tools such as the configuration management database (CMDB). While some existing platforms (e.g., Archer GRC Solution [167]) enable practitioners to view the reports generated through the vulnerability scanner, it requires a lot of manual effort on the customisations.

"If the tool could communicate directly to [the scanner], then we can do some better filtering and information sorting. And if it can be linked to our internal knowledge bases, no need for manually tracking or storing with any individuals." - P4

Automated Preparation for Patch Deployment

The need for automation support for identifying the patch prerequisites was highlighted by many participants. As described in Section 6.4.1, detecting prerequisites is currently a manual effort of reading through the release notes. However, the execution of prerequisites is considered difficult to be automated as selecting the optimum configuration needs reasoning based on the environment, therefore demanding human intervention.

"The tools do not consider what vendors have written in their release notes. The release note may say, to complete this patching, you also need to set up this configuration in this manner, and a lot of times this is missed. So a system that

can flag which patches require manual intervention to adjust settings would come in handy." - P9

Another requirement is better automation support for patch testing with the flexibility to easily configure the test environment depending on the patch dependencies. This presents an additional requirement which is automation support for detecting dependencies, for example, a dependency graph visualising the system interdependencies.

Automation Support to Articulate Patch Scheduling

Another noteworthy desire from many participants was automation support for articulation work in patch scheduling as it involves a series of cooperative tasks (as described in Section 6.4.1). The existing tools lack support in managing the integral set of patch scheduling tasks. The participants desire a single dashboard view of all patch schedules enabling easy identification of patch windows based on availability, modification of schedules, tracking status and schedule changes, interactive communication between stakeholders, and integration with the patching tool to export the schedules straight to deployment. Such a platform would assist in articulating patch scheduling and rescheduling tasks among distributed stakeholders to speed up patch deployment.

Automated Patch Deployment With Better User Control

Whilst the current tools are capable of automated patch deployment, the participants expressed the need for enhanced automation with more user control in the capabilities described below. Reduced or no system downtime to address the challenges of service disruptions during reboots and exhaustive manual overhead of the stop-patch-restart process described in Section 6.4.2.

"It would help a lot if there's a way of figuring out non-rebooting patches so that we could patch and keep going and it wouldn't matter when you patch as much." - P8

The capability to automatically execute simultaneous multiple reboots on multiple servers is preferred as the current manual activity often leads to missing them leaving practitioners with "no way to catch up" due to time and resource constraints. Another need is a unified platform capable of supporting patch deployment across heterogeneous environments, particularly beneficial in alleviating obstacles associated with shared access in an environment. In addition, a few participants indicated the need for improved usability of patch deployment tools with increased efficiency.

"I guess the idea of automated patching is at a more holistic level and it's not well known in the enterprise context. It's an interesting one because you don't like a tool because of technical limitations is one thing but an experience of a customer is an entirely different vector of thought associated with a tool. I mean, yes, we can tick some boxes in the tool and be happy because they're all ticked but if it's not easy to use, what's the point then." - P1

Automated Patch Deployment Verification And Recovery

Several participants expressed the need for improved automation for verifying patch deployment and detecting post-deployment issues, particularly reporting. This need was emphasised by many as the current verification tasks require significant manual overhead thus often leaving them neglected. Real-time reporting is preferred by the participants over monthly summary reports as it enables detecting and responding to issues promptly. Additionally, providing meaningful error messages to aid the troubleshooting and automated recovery of patch deployment failures is also desired.

"Verification of the deployment would be the biggest plus, so we wouldn't find an issue a week later. And then better reporting on the patching success and the exceptions. A summary at the end of the month to me is a waste of time, I want as close to real-time." - P10

Improved Configuration Management Database With an Overview of System Interdependencies

A configuration management database (CMDB) displays the configuration items (e.g., server, application, router, etc.) in a managed environment and how they interact with each other. As mentioned by P1, *"a good CMDB will have whatever you want in there, whereas at best we have got a list of servers that may or may not be up to date with no relationships between them. It is something that helps you make informed decisions, especially around change management, be it patching or otherwise. I think that is an important vector into patching and the culture of how stressful it makes people"*. The participants indicated the need for an improved CMDB providing a real-time overview of the system's patch state including information about the server status, patched date, decisions (e.g., server exemptions), and better filtering.

A single view of system interdependencies would be helpful to reduce the manual overhead of handling issues, particularly legacy software dependencies wherein the current process relies on human knowledge and expertise. It also facilitates the understanding and coordination of manual patching and patch scheduling between teams, thereby reducing the delays and risks of additional outages to critical services. A few expressed their desire for predictive analysis on the impact of patch dependencies, for example, predicting the list of interdependencies that could potentially be impacted by a particular patch to guide practitioners in patch testing.

"For systems that are 20 years old, it's pure luck if there's somebody who knows it is still around. If not, there's no formal process to follow in the absence of architectural design documents. So during decommissioning, we will shut them down for two weeks and then wait to hear who complains because we don't know who relies on what." - P14

6.4.4 Role of Human in Process Automation

This section answers RQ4.4 by describing the crucial roles of humans in the security patch management process automation and explaining why and where human involvement is needed.

Gain Control Over Uncertain and Dynamic Environment Conditions

The evolving conditions of the environment resulting in unpredictable changes to schedules are one of the main reasons demanding human involvement in automation. In such cases, the participants revealed switching to manual patch deployment to obtain increased control of the situation. Below are some *use cases* that describe the roles humans play in a dynamic context.

Emergency patches released to fix critical vulnerabilities require urgent attention as the patches need to be deployed within 48 hours of release. The unanticipated event and urgency of the task calls for human involvement in careful planning and execution in an OOB window, which is not possible through automation.

Lack of visibility into patch load dynamics during the patch window (e.g., how many patches get applied, how long it takes to apply) is another reason that requires humans to be involved in instructing the tool of the subsequent actions (e.g., revert the installed patches, extend the window by x hours) to maintain system availability.

Unforeseen errors in patch deployment resulting from faulty patches create the need for understanding the severity and impact of the situation, wherein humans are accountable for service continuity. Despite the uncertainty, incompatible dependencies exacerbate the challenges demanding an increased level of human involvement to resolve the broken dependencies. This is because new patch releases only consider compatibility with the most recent vendor-supported software versions.

The *dynamic environment context calls for a high degree of coordination* to manage the interrelated tasks between stakeholders. In mission-critical contexts, some coordinating tasks such as safety checks (e.g., time to deploy the patch) are extremely difficult to automate. Many participants do not believe that coordination tasks such as finding an outage window or negotiating a patch window extension can be automated as expected.

"From my point of view, the biggest problem is coordination and that's a purely human-driven process. I can't see how the AI or a machine with various tools can help us find an outage window." - P11

Contextual Awareness-Based Decision-Making

Increased complexity of tasks together with existing tools' lack of domain expertise and inflexibility demand human involvement in the appropriate decision-making. Humans' contextual awareness, which cannot be fully automated away, is key for making the right decisions during uncertain and complex events. For example, *shared services* necessitate the need to handle a large number of interdependencies in managed systems. Certain interdependencies such as legacy software dependencies entail significant domain expertise in assessing the risks of service interruptions.

"One of the biggest things is understanding and trying to work out all the combinations of interdependencies. If we patch this one, will it break the other one or do we have to upgrade this one and this one? That is a big challenge as the more applications you put on one server, the more possibilities of interactions, hence more combinations that you've got to consider when patching." - P10

High severity and impact of services exacerbate the challenge of service downtime resulting from reboots. The criticality of service downtime requires an increased sense of agency in initiating, executing and controlling the patch deployment, where human sense-making is needed to decide on the right action.

Further, the large volume of machines with myriad software versions, machine clusters and patch levels result in *too many configuration options* leading to cognitive overload. The patch exemption requests from clients further complicate the issue. Hence, it necessitates human involvement in selecting and implementing the suitable configurations as the automation is not capable and trusted in reasoning, classifying or predicting the configuration options. Other example scenarios include making informed decisions about the need and number of extra reboots during a patch window and implementing workarounds to maintain system availability during post-deployment errors based on the severity and impact.

"Very often human decision-making is needed. That extra setting really depends on your environment. And to see how your environment may be relevant, you have to assess how the risk is applied to your context." - P9

Handle Legacy Systems In Place.

Having *unsupported legacy systems in place* is one of the fundamental reasons requiring human involvement. While the unsupported software poses a huge threat leaving several attack vectors open for exploits, many organisations retain legacy systems because of the service criticality and high cost and complexity involved in migrating legacy systems. The lack of an upgrade path for legacy applications and the lack of support from vendors place the burden of handling incompatible legacy software dependencies on the practitioners' shoulders alone. An additional challenge in this is finding human resources with the knowledge and capability of resolving the legacy system-related issue.

"When we have problems with legacy systems, we find someone who knows the system to get it working again and then we don't touch it. If we can find someone who knows to fix it, beauty but if we don't, we wouldn't know what to do." - P14

Another instance is *executing multiple reboots*. The legacy software requires multiple reboots for getting the software up to date. As the number of reboots required depends on the system being patched, a human understanding of the context is needed to identify the exact number of reboots. Further, since multiple reboots can potentially exceed the patch window, human involvement is needed to take remediation action depending on the conditions.

Adapt to the Organisational Needs and Culture

Organisational factors including culture, policies and needs play an important role in the need for human association in process automation. This finding provides additional evidence to previous work [2, 109], which also reported the influence of the organisation's internal policies and management on system administrators in the process.

Handling negative perceptions about security in the organisational culture, for example, lack of interest, freedom and priority for security, results in poor practices or decisions that adversely impact security patch management. For example, non-security teams tend to neglect security patching, higher management delays patch approval decisions, and stakeholders refuse to corporate. In such cases, human involvement is essential in getting people to understand the need for security patching and negotiating the patch schedules maintaining a balance between the need to patch and maintaining system availability. Another instance is the *need for managing resistance to change*. Interestingly, in some cases like patch scheduling, some participants reported they are resistant to shifting to an automated solution as they have got used to manually doing it with Excel spreadsheets.

"Trying to get some people to understand why you need to patch is a big challenge. You can tell all the scare stories about people being hacked by ransomware but does it mean my server has to be down twice even between 2 and 4 a.m., I need it up 24/7. I can't afford you to take it down." - P8

6.5 Discussion

We discuss the findings from this study and its broader implications for practitioners and researchers to derive some actionable insights and identify the areas for future efforts aimed at providing automation solutions for security patch management.

We present an evidence-based understanding of the role of automation in security patch management that describes the as-is state of automation in practice, its limitations, practitioners' needs for improved automation support and the role of the human in security patch management automation. The evidence from Section 6.4.1 shows that a majority of the tasks in the security patch management process are performed manually (see Figure 6.1).

We find several reasons for the current state of manual work in the process. The key reason is the limitations of the current automation support reported in Section 6.4.2. This situation stems from different factors such as limitations in the existing tool capabilities, lack of specific features, performance limitations, infrastructure constraints, and usability limitations as shown in Table 6.1. These limitations show the gap areas that present excellent opportunities for researchers and tool builders for providing advanced automation/tooling support.

Table 6.1. Classification of limitations of current automation

Category	**Example limitations**
Limitations in existing tool capabilities	• Capture organisational context in risk assessment • Accommodate sudden changes to patch schedules • Real-time detailed reports for deployment errors • Miss out information in scans • Service downtime resulting from reboots
Lack of specific features	• Identify prerequisites from patch release notes • Handle patch interdependencies • Detect the need for multiple reboots • Handle service interruptions from incompatible dependencies during deployment • Detect mid-cycle patch releases
Performance limitations	• Execute parallel deployment jobs • Run simultaneous multiple reboots on multiple servers
Infrastructure constraints	• Deploy patches to heterogeneous environments
Usability limitations	• Lack of meaningful error messages

Further, the evidence from Sections 6.4.2 and 6.4.3 indicates that certain limitations of the current automation solutions require the practitioners to perform several tasks manually which usually causes delays in patching. We argue that the tool design may have failed to take into consideration why and how automation is actually used in practice. We anticipate our findings would provide a solid understanding of what enhancements in automation support are needed (Sections 6.4.2 and 6.4.3) and how those enhancements will be used in practice (Section 6.4.1). Such understanding will also be beneficial to shaping future work to effectively address the practical concerns in security patch management. The findings of Section 6.4.3 can provide an opportunity for tool builders to identify the features that would enable their tools to add more value and practical utility in practice. We recognise that these findings might not capture all the desired tool features as the participants might not be aware of other available tools and their capabilities. This limitation also prompts future work for a mapping of the features of the existing patch management tools to the practitioners' desired needs to scope down *what is missing where.*

Given our findings are context-dependent, i.e., in the healthcare domain, future research is needed for broader validation in different contexts and using additional data sources (e.g., large-scale surveys). The findings from this study can be extended to other domains to identify additional desired features with concrete requirements. Similar to previous work [20, 54], we foresee the value of future work to evaluate the tools in real-world contexts to better understand how well the tool meets the industry needs.

Another set of reasons for the increased manual work stems from the socio-technical implications of human and machine interaction in the security patch management process (reported in Section 6.2). These reasons demand an in-depth investigation of the critical roles that humans play in gaining control of uncertain situations, understanding the context, making sense of the available information and accordingly making informed decisions as described in Section 6.4.4. Our findings identify the tasks and points in the security patch management process that require a balance between human control and automation so that future automation can be designed to facilitate a collaborative relationship with humans. The findings present opportunities for future research in *"Human-AI collaboration"* [92], an emerging research paradigm which combines human and machine intelligence to collectively achieve a goal. We also suggest further research to investigate the challenges in developing patch management tools that harness collective human and machine intelligence. Based on our findings and the existing research on "Human-AI collaboration" [91, 173, 174], we propose some recommendations that can guide future tool development to address the limitations of the current automation solutions.

6.5.1 An Integrated Platform Offering Support Across All Process Phases

The lack of an end-to-end automated solution that supports all phases of the security patch management process has been an important anti-pattern recognised in our study, also reported in Chapter 3. This situation forces practitioners to use multiple tools for different tasks in diverse environments resulting in increased errors and delays. Since building a unified solution capable of supporting heterogeneous environments (e.g., Windows, Linux, Mac) is highly complex due to the inherent differences in each environment, there is a need for a consolidated platform that can offer end-to-end patch management life cycle support in a single environment to address a majority of the limitations of the existing solutions. Moreover, such a consolidated environment will also benefit from being interoperable with external third-party solutions to cater for the needs of a multi-vendor environment.

6.5.2 Human-Machine Collaboration for Patch Management

We envision the integration of human and machine intelligence can be beneficial in several security patch management tasks reported in Section 6.4.4. For patch prerequisites identification, we envision a system that not only leverages combined human and machine intelligence but also learns from each of the collaborative partners (i.e., human and machine) to improve future prediction accuracy. Specifically, a machine can convert the unstructured data in patch release notes to a structured format and identify the prerequisites, and provide the summary results to the human partner to ease the selection of the optimum configuration options based on domain knowledge. Given the recent advances of Large Language Models (LLMs) such as ChatGPT and

GPT-4 [175] in diverse tasks and contexts, future research can investigate the application of LLMs for human-machine collaborative patch prerequisite management. Further improvements can be made by leveraging machine learning techniques to predict the most suitable configuration options that can guide practitioners to make quick decisions.

Similarly, joint human and machine intelligence can benefit in identifying and handling machine configurations and patch dependencies. A machine can identify the interconnections between configurations and patch dependencies, and predict the potential dependency breakdowns to reduce the possibilities of human errors in handling them and the expected delays in manual work. Another task that can benefit from human and machine collaboration is patch deployment. When there are errors during patch deployment, a machine can inform the dependent stakeholders of deployment failures and guide them about the potential subsequent actions (e.g., revert deployment, estimates of patch window extensions).

Human-Centred AI Explanations to Assist Contextual Decision-Making

To support human decision-making based on contextual understanding (Section 6.4.4), we propose that future automation focuses on enabling human-centred explanations. As suggested by the Human-AI Interaction design guidelines [173] “make clear why the system did what it did", we anticipate such a capability will assist humans in understanding a machine’s predictions thereby allowing them to make better decisions based on the context. For example, in selecting the optimum configuration during patch testing, a machine can provide the basis for the prediction. As another important design functionally, such a system shall enable interactive bi-directional communication. Interactive explanations enable humans to interact with machines [176]; for example, by editing input and changing the prediction based on the context. Further, a machine can learn from human interactions and evolve to improve its predictions by leveraging machine learning training techniques such as Reinforcement Learning and Active Learning. Future research can further investigate how information (e.g., visualisations, chat summaries) and notifications should be best presented to ensure effective human-machine communication.

Decision Support for Patch Scheduling

The evidence reported in Section 6.4.1 shows that sudden changes to the patch schedules demand an arduous manual task of patch rescheduling. Further, it can produce a cascading effect on the dependent patch schedules resulting in increased complexity for rescheduling and further delays in patching. Responding to these challenges, we suggest developing a smart decision support system to guide practitioners toward accurate and quick decisions about patch scheduling and rescheduling. The envisioned system should be able to identify the cascading effect on dependent patch schedules, i.e., which schedules will likely get affected and how much the impact, based on the identified patch dependencies and prerequisites. Accordingly, the system should provide an estimate of the patching delay. We argue that these insights if presented in simple visualisations (e.g., graphs), would lead practitioners to make informed decisions faster about rescheduling.

6.6 Threats to Validity

In this section, we discuss the potential threats to validity and how they were mitigated based on the guidelines reported in [33, 123].

6.6.1 External Validity

Our findings do not claim to be generalised from a sample to a population but rather applicable to the studied cases like most qualitative research. Our findings are limited to the practitioners in the case organisations in the domain of healthcare studied in-depth to provide a holistic understanding of the studied topic grounded in evidence that is not attainable through broader but shallower approaches (e.g. surveys). However, we believe that our findings can be recreated and adapted in other similar contexts; for such purposes, we have provided sufficient details about the studied participants and the research methodology (Section 2.2.2) to facilitate transferability.

Concerning data representativeness, we are aware that our data collection is limited to the interviews although it covers a wide range of the participants' roles and experiences. We acknowledge that data triangulation, e.g., additional cases, observations or surveys, will be useful for future studies to verify this study's findings and also extend the scope.

6.6.2 Construct Validity

To mitigate this validity threat, all researchers collectively prepared the interview guide and pre-interview questionnaire. The interview questions were revised through several iterations following a pilot interview which was held between researchers (i.e., the present author as the interviewee, the third researcher as the interviewer, and others as observers). It assisted in finalising the questions' scope, clarity and duration. However, as the findings were not verified with the participants through member checking, there is a threat of potential misinterpretations from the interviews.

6.6.3 Internal Validity

To alleviate potential internal validity threats, we included the participants who are involved in security patch management and ensured a representation covering all aspects of security patching in the case organisations, i.e., server and desktop. Further, the participants had 22 years of experience on average, mitigating the risk of the participants' lack of expertise.

6.6.4 Reliability

To ensure the reliability of the findings, the data collection, analysis process, and the emerged findings were thoroughly discussed among all researchers in weekly meetings, as described in Section 2.2.2. Furthermore, two researchers were present in seven interviews to minimise the threat of researcher error.

6.7 Chapter Summary

In this chapter, we report an empirical study aimed at providing an evidence-based holistic understanding of the role of automation in security patch management. Based on semi-structured interviews of 17 practitioners, we conclude that while the security patch management workflow in practice incorporates manual and automated approaches, the majority of the tasks are performed entirely manually due to the limitations of the existing automation/tools and the lack of capacity of the available automation to handle certain use cases demanding human involvement. Based on an improved understanding of the practitioners' perceived needs for enhanced automation support, this study enables us to explain why and where human involvement is needed in security patch management automation. Our findings indicate that human involvement in certain tasks in the workflow, particularly concerning contextual awareness-based decision-making and human intuition, is crucial in security patch management. That is why we propose that future patch management tool development should aim at supporting human-machine collaboration by leveraging the best of both capabilities. For future work, we outline how and what functionalities may be needed to effectively address the current gaps and user needs whilst embracing the desired balance between automation and human control in the security patch management process for reducing patching delays.

Chapter 7

Conclusions and Future Work

Timely security patch management is vital for safeguarding software systems against cyberattacks. Despite the practical importance, security patch management remains one of the most complex challenges facing modern organisations due to the inherent technical and socio-technical interdependencies involved in the process. As a result, many organisations often delay or avoid the installation of security patches leaving a multitude of attack vectors vulnerable to exploits. While researchers and practitioners have paid significant attention to the technical aspects of security patch management to address delays, the socio-technical aspects of security patch management delays have received relatively limited attention. This is an important limitation because security patch management is an inherently socio-technical endeavour involving complex interactions between humans, machines and environmental aspects. Therefore, knowledge of the socio-technical aspects is essential to gain a complete understanding of how human, organisational, and social factors and technological systems in security patch management lead to delays. This understanding can contribute to the design of solutions involving the process, organisational structures and technical advancements to address the root causes of patching delays.

This thesis aimed to address this gap by providing an evidence-based understanding of the socio-technical aspects of security patch management. Specifically, this thesis presents a set of empirical studies to identify, understand and address the role and impact of socio-technical aspects on delays in security patch management. To that end, we have first provided an organised evidential body of knowledge on the topic by identifying and categorising the socio-technical challenges and available solutions, and analysing how well the reported solutions have been assessed with their level of rigour and industrial relevance. Then we conducted a longitudinal field study with 10 teams from three organisations in the healthcare domain to gain an in-depth understanding of delays in real-world security patch management from a socio-technical perspective. Based on longitudinal data collected from artefact analysis, observations of patch meetings, and semi-structured interviews, we have identified the root causes and mitigation strategies for delays in security patch management, proposed a grounded theory of the role of coordination in security patch management that explains and addresses the most prominent cause of delays in the process and provided an evidence-based understanding of the role of automation in security patch management. This chapter first summarises the main findings and key contributions of this thesis and then suggests some avenues for future research.

7.1 Summary of Findings and Contributions

This section summarises the main findings and key contributions of this thesis.

7.1.1 A Systematisation of Knowledge of Security Patch Management

In Chapter 3, we have provided a holistic insight into the state of the art of socio-technical aspects of security patch management. From a comprehensive review of 72 primary studies on security patch management, we have identified and categorised the socio-technical challenges and available solutions and analysed the reported solutions based on their level of rigour and industrial relevance. Given there is no commonly known/accepted definition, we have provided an operational definition of security patch management based on our evidence-based understanding from a longitudinal field study and the related literature.

From the analysis, we have identified and classified 14 socio-technical challenges as six challenges common across all phases of the security patch management process and eight others as specific to each phase of the process. Further, we have identified 18 solutions including 6 themes of approaches and tools and 12 practices as common ones affecting all phases of the security patch management process and those specific to each process phase. We have also provided a mapping between the solutions and challenges to provide an overview of the gap areas. The findings have revealed that 50% of the common challenges have not been directly addressed in the solutions and that 37.5% of the solutions address the challenges in one phase of the process, namely vulnerability scanning, assessment and prioritisation. Moreover, we found that only 20.8% of the reported solutions have been rigorously evaluated in industrial settings indicating a need for more empirical investigations using real-world representative evaluation approaches. Our findings have shown that the socio-technical aspects have a wide-ranging effect across all phases of the process and that human-in-the-loop is inevitable in security patch management due to its inherent complexity and dynamic nature. However, an important observation has been the lack of evidence-based understanding of the role and impact of socio-technical aspects on the delays in security patch management. It presents an important limitation as evidence-based research with industrial relevance produces outcomes having higher value addition and practical utility. This observation motivated us to conduct a longitudinal empirical investigation into the socio-technical aspects of the delays in security patch management presented in Chapter 4.

7.1.2 An Evidence-Based Understanding of the Reasons and Mitigation Strategies for Security Patching Delays

In Chapter 4, we have conducted a longitudinal case study to explain why, how and where delays occur when applying security patches in practice, and how the delays can be mitigated. Based on longitudinal data collected from artefacts consisting of 132 delayed patching tasks over a period of four years and 66 observations of patch meetings involving eight teams from two organisations, and using quantitative and qualitative data analysis approaches, we have presented a taxonomy of reasons relating to technology, people and organisation as key explanations that cause delays in security patch management. Further, we have provided an analysis of the frequency distribution of reasons for delays and the distribution of delays over the patch management process. The findings have revealed that the most prominent cause of delay is ascribed to coordination delays in the security patch management process and a majority of delays occur during the patch deployment phase. Such knowledge is useful in identifying the reasons that need immediate attention and the areas of improvement in the security patch management process. We have also provided a classification of strategies that can be applied in practice to mitigate the delays including when to apply them during the security patch management process.

To the best of our knowledge, this is the first research attempt to provide an evidence-based understanding of the causes and mitigation strategies for delays in real-world security patch management. The reported understanding helps practitioners to understand what and where improvement is needed in the security patch management process and guide them towards taking suitable decisions and timely actions to avoid potentially catastrophic consequences from delays in patching. Furthermore, our findings lay the foundation for future researchers and tool designers to design and develop computer-supported solutions to address the practical concerns causing delays in security patch management.

7.1.3 A Grounded Theory of the Role of Coordination in Security Patch Management

Motivated by the need to gain a thorough understanding to address the most prominent cause of delay in security patch management as revealed in Chapter 4, we have conducted a Grounded Theory study of the role of coordination in security patch management in Chapter 5. Based on observations of 51 patch meetings over a period of 9 months involving 21 practitioners from eight teams in two organisations, we have provided answers to some important questions such as how and why coordination causes delays in security patch management and what can be done for addressing the coordination delays in security patch management? Following the guidelines of Glaser's Grounded Theory [44] research method, we have uncovered a grounded theory of the role of coordination in security patch management consisting of four interrelated dimensions, i.e., causes, constraints, breakdowns, and mechanisms. The theory explains the causes that define the need for coordination among interdependent software/hardware components and multiple stakeholders' decisions, the constraints that can negatively impact the effective coordination of the causes, the breakdowns resulting from ineffective coordination of causes and constraints, and the potential corrective measures for enabling effective coordination in the process.

Overall, we have provided evidence that the role of coordination represents a core concern in timely security patch management and provide emphasis on the need to have a delicate balance between socio-technical concerns such as coordination and automation in the process to reduce delays. The reported theory provides several useful insights for practitioners to avoid patching delays and failures and increase confidence in their security patch management decisions. The theoretical model can be useful in shaping future research to address the practical concerns in coordination delays in security patching and for developing automated solutions for supporting effective coordination in the security patch management process.

7.1.4 An Empirical Understanding of Automation in Security Patch Management

Another key reason for security patching delays as described in Chapter 4 attributed to technology-related reasons owing to the limitations of current tools and the need for human intervention. While it is widely understood in the existing software engineering literature that automation support for different activities of the security patch management process has great potential for reducing security patching delays, there is not much known about how automation is used in practice, its limitations in meeting real-world needs and what practitioners really need. To find answers to these questions, we have conducted an empirical investigation in Chapter 6 of the role of automation in security patch management. Based on semi-structured interviews with 17 practitioners from three organisations in the healthcare domain, we have provided a holistic insight into the role of automation in security patch management explaining the as-is state of automation in practice, the limitations of current automation, how automation support can be enhanced to effectively meet practitioners' needs and the role of the human in an automated security patch management process.

Concerning the as-is state of security patch management, we have found that whilst the security patch management process incorporates manual and automated approaches, the majority of the tasks are performed entirely manually due to the limitations of the existing automation/tools and the lack of capacity of the available automation to handle certain use cases resulting in the need for human intervention. Based on an understanding of the practitioners' needs for improved automation support, we have explained why and where human involvement is needed in the security patch management process automation. Our findings have shown that human involvement in certain tasks in the process, for example, decision-making based on contextual awareness and human intuition, is critical in security patch management. Accordingly, we have proposed a set of recommendations for directing future work to address identified gaps and user needs. In these recommendations, we have described how and what functionalities may be needed to effectively address the current gaps and user needs whilst embracing the desired balance between automation and human control in the security patch management process.

7.2 Opportunities for Future Research

This thesis has made several contributions towards gaining an evidence-based understanding of security patch management from a socio-technical perspective. However, there are still several opportunities for future research to address and further advance the field. While some suggestions for future work have already been described in the previous chapters, we summarise them in the following areas below.

7.2.1 Replicating the Study

While our research is the first attempt to incorporate a longitudinal approach to empirically explore delays in security patch management from a socio-technical perspective, the replication of our research with larger samples and different contexts can be an important direction for future research. Given our findings are based on the contexts studied limited to the domain of healthcare, researchers can extend and adapt the findings through future research within the same domain involving different stakeholders or different domains. This presents an excellent opportunity for the generalisation of the findings presented in this thesis. Furthermore, future research can explore what other reasons for delays exist, and the viability of the findings based on contextual factors, for example, variations in organisation-related reasons for delays and how the role of coordination and automation is impacted by contextual factors. Such context-specific research can result in useful insights and models from additional cases with extended scope. Another direction of future research is investigating the effectiveness of the solutions proposed in this thesis (e.g., mitigation strategies for delays in security patch management and coordination mechanisms) and the context in which they should be deployed. Additionally, future research can use the findings in potential interview guides and large-scale surveys to evaluate the findings in different organisation settings and identify variations within them.

7.2.2 Technological Support for Dependency Management

As explained in Chapter 5, the security patch management process involves intricate technical and social interdependencies among various software and hardware/firmware components, as well as multiple stakeholders. These dependencies create the need for effective coordination, which represents a crucial factor in reducing patching delays and ensuring smooth security updates. In our research, we further discovered in Chapters 5 and 6 that a lack of automation support hinders the effective management of both technical and social interdependencies. This limitation is a significant concern highlighted by practitioners who express the need for technological tools to assist in dependency management, as reported in Chapter 6. This opens up exciting opportunities for future research to explore and develop innovative solutions in this area. For example, one potential avenue for future research is to investigate the creation of automation support or tools specifically tailored to manage socio-technical dependencies. These tools could facilitate collaboration among various stakeholders involved in the patch management process, ensuring smoother communication and decision-making. Such support could improve the overall efficiency of the patch management process and reduce the likelihood of delays that might otherwise lead to potential exploits.

Moreover, concerning technical dependencies, future research can delve into designing systems capable of automatically identifying complex system configurations and patch interdependencies. By doing so, these systems could preemptively identify potential technical dependency breakdowns, which are often responsible for delays and errors in patch testing and deployment. Automating this process could significantly expedite the identification and resolution of technical dependencies, leading to faster and more reliable patch deployment.

7.2.3 Decision Support for Security Patch Management

Effective decision-making plays a pivotal role in ensuring timely and efficient security patch management. As highlighted in Chapter 6, the practical implementation of security patch management involves several decision points that necessitate human intervention. These decisions are often impeded by task complexity, a lack of domain expertise, and the limited flexibility of existing tools, resulting in significant delays in the patch management process. Consequently, there is a pressing need for an evidence-based understanding of the role of decision-making in security patch management. This need for comprehensive knowledge presents an opportunity for future research to focus on developing intelligent decision-support tools and technologies that can assist human decision-makers in making accurate and timely patch management choices.

Another promising avenue for future research is the design and implementation of decision support systems that harness the power of Artificial Intelligence (AI). By integrating machine learning algorithms, these systems can learn from historical security patch management data and offer real-time insights and recommendations to human decision-makers. For instance, AI-based decision support can be particularly valuable in activities such as customised vulnerability assessment and prioritisation, dynamic patch scheduling, and post-deployment patch verification. The application of machine learning algorithms can aid in identifying critical vulnerabilities, assessing their potential impact, and prioritising patch deployment accordingly. This not only streamlines the decision-making process but also ensures that resources are allocated efficiently to address the most critical security vulnerabilities.

Moreover, the design of these decision support systems should be centred around facilitating effective collaboration and coordination among different stakeholders involved in the security patch management process. By fostering seamless communication and knowledge sharing, these tools can foster a more collaborative and cooperative approach to patch management, leading to improved overall security posture with reduced delays.

7.2.4 Studying the Influencing Factors and Impact of External Stakeholders on Patching Delays

As demonstrated in Chapter 4, our investigation revealed several contributing factors to patching delays, such as input requirement delays, service availability restrictions, and the complexity of patches relate to interdependencies with external stakeholders, including vendors and customers. While this thesis presents compelling evidence regarding the influencing impact of external stakeholders on security patching delays, further research is warranted to gain a comprehensive understanding of the underlying factors and the extent of their impact on patching delays. For example, future research can adopt a stakeholder-centred approach and study the perspectives of external stakeholders to identify additional socio-technical aspects that may influence patching delays. By capturing the insights and experiences of these stakeholders, researchers can identify previously unexplored factors and uncover potential solutions to mitigate delays.

Moreover, a critical area for investigation lies in exploring methods to enhance the awareness of the importance of timely security patch management among external stakeholders, such as customers and end-users. By examining the knowledge gaps and perception barriers surrounding patch management, future research can devise strategies to promote understanding and instil a culture of prioritising timely security patches. For instance, employing targeted educational campaigns, interactive workshops, or informative materials can be effective ways to communicate the significance of security patches to customers and end-users. Increasing awareness can lead to a more proactive approach towards patch management, reducing delays and minimising the window of vulnerability to potential cyber threats.

Appendix A

Codebook for the Reasons for Security Patching Delays

Category	Subcategory	Concept	Code	Description	Number of references	Examples
Reasons for Delay	Technology-related reasons	Complexity of patches	Patch interdependencies	The software, hardware, and firmware interdependencies. The dependencies consisting in the source code (functions, library), versions and legacy software.	30	[Subject - Software version issues on the system [EPAS 17.3] to resolve] 24/1/19 - 47 Office patches on servers [s1-6] (multi-languages and x32 and x64 bit versions) and 7 Visual Studio-related security patches having errors while testing. 6/2/19 - A ticket raised to investigate Visual Studio patches and manually apply if required. - EMR, Issue ID 9
			Faulty patches	Unknown errors during patch testing, deployment, and post-deployment arising from faulty patches.	12	[Subject - [Server s1] non-functional after patching] 30/10/18 - Org A raised a support case to the [vendor v1] as the server was left in an unusable state with no communication post-patching. 4/4/19 - No further update regarding the support case. - Win, Issue ID 6
			Extensive monitoring for faulty patch fixes	Issue kept open and under monitor until the practitioners confirm that the patch poses no unanticipated adverse effects.	8	[Subject - Patch deployment error at the [server s1] 13/12/19 - Workaround applied and timings were all good. Keep open till January run for confirmation. - EMR, Issue ID 35
			Patch heterogeneity	Diverse set of patch types increasing the complexity.	5	[Subject - Type 4a vs Type 4 for .NET patches] 28/6/19 - Confirm if all servers have moved to Type 4 vs 4A - all needs to be Type 4 with .NET patching as it is -2 months behind (Pre-Prod, Prod) 12/7/19 - Miscommunication regarding patch types 4a and 4. From July onwards Type 4A in previous format will be defunct and Type 4A profile will be used for Type 4 .Net-2 months. 9/8/19 - Type 4A is now named Type 4EMR, [P1-AT1] will send out email to advise wider audience. Item to close. – EMR, Issue ID 24
			Increasing rate of patch release	A large number of patches and patch types.	2	[Subject - PreProd and Prod of [T1] to be patched monthly and in separate machine group] 22/2/19 – Org A considering going bi-monthly with PreProd and Prod to keep up with patches. Then moving monthly after confidence is built around for .NET patching 04/04 – Team [T1] still considering the patch cycles and windows for the PreProd and Prod [T1] servers. Various options were requested and were put out in slides for their consideration- T&M apply as outside standard patching methods. – EMR, Issue ID 2
		Limitations of current tools	Lack of accuracy	Lack of accuracy in the output of current tools - missing vulnerabilities in scans, omitting patches during patch deployment.	9	[Subject - Issues with IE patches] 7/8/19 - [...] The Windows update is showing zero missing patches but the [patch deployment tool] is showing 15 missing patches related to IE. [P1-AT2] to raise a support case for Microsoft account manager attention. - EMR, Issue ID 35
			Lack of scalability	Lack of scalability to handle diverse types of patches and their features.	3	[Subject - 2019 SHA-2 Code Signing Support requirement for Windows and WSUS] 8/3/19 – The patch team following MS release dates for those required updates and will apply to affected OS - win2008R2 SP1 & SP2. 12/3/19 - MS released stand alone security updates: KB4474419 and KB4490628 released to introduce SHA-2 code sign support - for Win2008R2 SP1. Will be applied with March cycle to affected servers. 3/5/19 - Excluded against 2008 servers as the tool doesn't support it, [B-T1] working in reports on how it can be applied in near future in May. – EMR, Issue ID 24
			Functionality limitations	Functional incapabilities in the tools.	7	[Subject - Additional reboot required for .NET patching] 7/2/20 - Investigation is needed around the number of required reboots for EMR patching and window requirements as a result if more reboots are required. A new process needs to be fleshed out when patching is postponed to accommodate the identification of the number of reboots required. - EMR, Task ID 35
		Need of human intervention	Troubleshooting	Troubleshooting the issues, unknown errors during and post-deployment and faulty patches.	23	[Subject - Issues with patch installation at [server s1]] 17/4/19 - Installed patches reverting and some patches not getting installed, need further investigation. 3/5/19 - [P1-BT1] ran troubleshooting, no success. Need to raise a support case with Microsoft for advice. - Win, Issue ID 20
			Manual patch deployment	Manual patch deployment during complex, erroneous, or business-critical patch installations.	17	[Subject - [Hospital h1] patching stage 3 on 27th November] 18/10/19 - Patching needs to be moved to OOB due to the change freeze from 15th November to 3rd December. 31/10/19 - [P1-BT1] team putting in significant amounts of work, like 15-20 hours per month, to redo the schedules on custom dates each time the deployments move off standard windows. - EMR, Task ID 30
			Decision approvals need thorough assessment of patch impact	Thorough assessment of the impact on multiple aspects to avoid breakdowns.	10	[Subject – Decision on pre-Download of patches for [A-T1]] 29/11/19 - Pre-download to be reverted, no Pre-Download going forward until we can confirm no impact on windows. - EMR, Issue ID 38
			Manual configurations	Manual configurations based on the organisation needs.	6	[Subject – Decision on pre-Download of patches for [A-T2]] 31/10/19 – [P1-AT2] raised issue, Org B has configured 2nd job 2 hours into the window that rescans for missing patches and conducts a second cycle/post-reboot if required. No issues experienced in October to our knowledge, seeking client feedback. 1/11/19 – [P2-AT2] will check how many servers needed the 2nd reboot and configure accordingly. - Win, Issue ID 38
		Coordination delays	Delays in obtaining approval	Delays in decision approvals as they had to go through multiple teams (or levels).	37	[Subject - IE version upgrade approval required] 7/9/18 - quote now with [A-T2] team to approve. 3/5/19 - [P1-AT4] to follow up with [P2-AT2] on the status of this. 10/5/19 - [P1] following up, still waiting for approval. - Win, Task ID 1
			Lack of awareness of task progression	Lack of awareness of task progression between teams and organisations.	22	[Subject - Need Service Packs updated to resolve software version incompatibility.] 31/8/18 - Not sure if [A-T1] team has raised Non-Standard Service Request (NSSR) ticket to upgrade 32-bit clients that are lower than IE8. 14/9/18 - Well, [P1-AT1] must know but he's not at the meeting, I'm not sure about it. - Win, Task ID 2
			Lack of understanding of roles and responsibilities	Lack of understanding of shared roles and responsibilities between teams.	6	[Subject - Investigation of post-deployment issue] 18/10/19 - [P1-BT1] says Org B should not be accountable for apps that don't start correctly. 31/10/19 - Decided that Org A or third-party vendor X should be fixing this error, Org B should not be accountable for application problems. - EMR, Issue ID 26
			Poor communication and information misinterpretation	Lack of communication and misinterpretation of communication creating delays as it creates the need to constantly follow up to get responses.	5	[Subject - Cluster-based patching (Type 4a vs Type 4 .NET patches)] 12/7/19 - miscommunication regarding Types 4a and 4. 9/8/19 - Type 4a is now named Type 4-EMR. [P1-AT1] to send out email to advise wider audience. - EMR, Issue ID 18
			Missing information due to overload of emails	Missed emails resulting in delays in passing information on time.	2	[Subject - 2008 patching in Week 2] 1/4/21 - Check emails for update KB5000851. The plan had been already emailed to the teams. - EMR, Issue ID 5
			Delays in obtaining customers' approval	Delays in obtaining customers' approval for patch deployment.	14	[Subject - Request to change patch window of [s1] server] 10/5/19 - currently set to 0000-0300, but the full backup of the server happening during this window causes slowness and issues with patching. Suggest changing the window to 0600-0900. [P1-AT2] checking on the status with business approval. 14/6/19 - [P1-AT2] to follow up as no response from the business. - Win, Task ID 19
			Delays in coordinating with vendors for support	Delays in coordinating with vendors for support cases.	12	[Subject - Patch deployment issue - file validation errors in [s1] server] 25/2/19 - [Vendor v1] case [n] - number of fixes suggested, completed last tests on server [s1], still no fix found. 3/5/19 - awaiting full response from [Vendor v1], investigations still underway. - Win, Issue ID 17
			Administrative overhead of coordinating with multiple customers	Administrative overhead of coordinating with multiple customers for pre and post-patching verification.	3	[Subject - Verify client email contact for the task [T1]] 27/2/19 - Provided email contact gives an error message stating that the email is restricted. So [Org B] has altered the special instructions to send emails to [P1] until sorted. 1/3/19 - Org A to contact the client and verify the client's email address. - Win, Issue ID 14

s			Delegation delays due to conflicts of task ownership	Delegation delays due to conflicts of task ownership with other third-party vendors because of lack of accountability.	3	[Subject – Investigation into SRM services not starting after patching] 31/10/19 - SRM process instructions were added for manual checks and run in October. Org B should not be accountable for apps that don't start correctly. SRM Prod boxes don't seem to be doing this, [A-T1]/[Vendor V1] should be fixing this at right level, Org B should not be accountable for application problems. - Win, Issue ID 36
	People-related reasons		Delays in delivering reports	Delays in delivering the reports such as the vulnerability scan reports.	16	[Subject – Awaiting reports on missing patches on EPAS] 24/1/19 – [P1-AT1] waiting for his request on a monthly report in missed patches - compliance report. Org B to organise and send weekly group reports for missed patches + monthly compliance report. Will consider also list of January released patches required on each server to be produced and communicated. - EMR, Issue ID 10
			Delays in delivering patch schedule information	Delays in providing information about patch cycle changes, patch windows, server lists and organisational schedule changes like change freeze delaying planning and execution of patch deployment.	15	[Subject - Scheduling Out-Of-Band (OOB) patching for exempted servers] 14/6/19 - An email sent to [P1-AT2] asking for patch window information, pending response. 28/6/19 - A follow-up email was sent to [P1-AT2] as no information was received. 26/7/19 - The patch window is still pending. - Win, Issue ID 22
		Input requirement delays	Delays in providing team requirements	Delays in providing team requirements for patching tasks.	4	[Subject - SQL Security Patching] 1/5/19 - No update since 17/4. Seeking confirmation on whether this is still a requirement from us [BT1]. - Non-Win, Issue ID 1
			Delays in patch release by vendors	Delays in the patch release.	11	[Subject - Investigation of missing patches from March patch run] 1/4/21 - OOB investigation results. Microsoft patch releases on patch Tuesday included listing KB5000803 (also applies to KB5000851 and KB5000853) but the patch was not available to download through [tool T1] until the 16th March. This patch also had a pre-requisite KB5001078 released in Feb that was also released late after Feb patch Tuesday. - EMR, Issue ID 54
			Delays in providing input for support cases	Delays in receiving vendor's support for support cases.	8	[Subject - New zero-day vulnerability warning] 12/6/20 - Monitor Microsoft patch release for critical vulnerability identified on [T1] servers. Font Type 1 expected as a zero-day soon, full report not available yet. 24/7/20 - No update from Microsoft. - EMR, Task ID 43
			Missing patch pre-requisites during installation	Missing patch pre-requisites such as registry changes, GPO configuration and installation of preparation packages. As a result, errors in patch deployment.	1	[Subject -IE version upgrades resulting in additional patches] 1/5/20 - Review status of the IE patching after the current patching run. Will generate status report from [tool T1] now as the patching run is complete. [P1-AT5] to also run a new report. 15/5/20 - Found 3 outstanding servers with patches to be applied for IE. [P2-AT1] to review the list of vulnerabilities from [P3-AT5] to confirm. 12/6/20 - 3 vulnerabilities identified under a new yet to be created GPO. [P4-BT1] scheduling the implementation of this. 24/07 - IE patches up to date, 0 missing patches. But keep under observation for one more patch cycle please. [P4-BT1] doing a final scan in relation to IE patching. Item can close. - EMR, Issue ID 57
		Failures due to poor planning and execution	Inaccurate estimates of patch windows	Inaccurate predictions of timing of patch installations.	3	[Subject - Execution exceeding the patch window] 31/5/17 - Only 72.9% of scheduled patch deployments were completed as of 11.20 am. Two further windows to be raised to ensure the appropriate length of time is scheduled due to unknown 2016 updates that were required to be implemented, first window is 1st June 8 am to 12 pm. - Win, Task ID 4
			Incomplete patch deployment	Patch deployment task left incomplete, such as required rebooting not completed.	5	[Subject – 2008 Critical patches from March] 16/4/21 – Server [s1] is the last remaining EMR 2008 server with missing patches. 30/4/21 – The last outstanding server is being tracked separately; item can close. - EMR, Issue ID 56
			Inadequate post-patch deployment verification	Failing to monitor the status of patch deployment tasks.	6	[Subject - [Server s1] post-patching functionality issues] 8/10/19 - [Server s1] has not shut down properly during the patch window which caused the printers to be unavailable for [customer c1]. It needs to move to semi-auto patching for November to monitor. - Win, Issue ID 32
		Organisation delays	Delays in getting approval from higher management	Delays in obtaining approval from organisational management for monthly patch schedules and changes in the process.	18	[Subject - RITM for new bespoke solution] 8/10/19 - Approval received from AT1, has been sent across to [P1-AT2] to raise the purchase order. 29/11/19 - The requested item is still with Finance (team) for processing. - EMR, Issue ID 23
			Delays due to changes in company schedules	Delays due to changes in organisation schedules like change freeze, testing plans, and holidays/shot down plans.	13	[Subject - Patching for December 2019] 18/10/19 - OOB for November patching from 4th December instead of December patching. 31/10/19 - EMR patching for December month is off but November Microsoft patches will be applied in the first week of December instead to keep compliance up. - EMR, Issue ID 29
			Resource limitations	Insufficient human resources to carry out certain tasks.	24	[Subject - L1TF spectre update] 16/5/19 - Task 0117374 assigned to [P1-BT1] (a senior server engineer) for implementation. 14/6/19 - Task on hold until July due to resource (i.e., P1-BT1) being borrowed by [AT4] team (for another critical task). - Win, Issue ID 21
	Organisation-related reasons	Capacity limitations	Infrastructure limitations	Infrastructure-related limitations leading to performance delays.	11	[Subject - Backup server patching] 24/1/20 - Patching cannot go ahead when the active backup is running. The patch load can impact servers before reboot. Need a window change, proposal to be sent by [P1-BT1] to [P2-AT1]. - EMR, Issue ID 39
			Time limitations	Time-bound limitations, e.g., monthly patch cycles, to progress with the tasks.	5	[Subject - [Server s1] storage failover issue] 7/2/19 - A support case was raised with [vendor v1]. Waiting for the next failure to occur to submit the logs, the next patch run is due in late March. - EMR, Issue ID 33
			Inability to allow service downtime from reboots	Organisations' restrictions/inability to allow service downtime from reboots.	13	[Subject - Go-Live for [Hospital h1] on 19th, need to postpone any patching happening on that day] 14/6/19 - OOB to be raised to patch servers - list of impacted servers to be sent to [P1-BT1] from [P2-AT1]. 28/6 - OOB scheduled on the next go-live 24th July. 12/7 – Customer proposing date 7th August. Need to reschedule. - EMR, Issue ID 22
		Service availability restrictions	Multi reboots requiring longer and additional patch windows	The patch schedules scheduled in out-of-band windows to reduce service disruptions from additional patch windows during business hours.	8	[Subject - [Servers s1 and s2] patching] 26/7/19 - OOB window is needed for the multi reboots to catch up. 9/8/19 - Waiting for the customer's confirmation of the new patch window, pending information from [P1-AT1]. - EMR, Issue ID 20
			Customer requests to postpone patch deployment schedules	Delays due to accommodating customer requests to delay patch deployment to avoid service interruptions.	4	[Subject - Go-Live for [hospital h1] on 19th July] 12/7/19 - Customer [hospital h1] asking to postpone patching until 7th August. - EMR, Issue ID 16

Appendix B

Codebook for the Mitigation Strategies for Security Patching Delays

Category	Subcategory	Mapping to the patch management process	Concept	Code	Description	Number of references	Examples
Strateg	Common strategies	All phases of the process	Frequent communication	Regular patch meetings	Bi-weekly patch meetings to discuss patching issues, find solutions, report the patching status, and measure the progress of the patch cycle.	8	[Subject - SQL Security Patching RITM# 52442] 12/7/19 - Patch meeting to be booked for 9am Friday 19th July to define requirements. 26/7/19 – [P1-BT1] putting together test run & proposal for discussion. - Win, Issue ID 1
				Informal team discussions for complex, critical issues	Scheduling different meetings for issues that need further discussion (Shifting long standing issues to other forums for future tracking).	6	[Subject - Post-deployment issue - Data Capture servers not able to communicate with [system s1]] 7/8/20 - [P1-AT1] checking with [P2-AT2] for the other three servers that do not have a commissioning request. 21/7/20 - Set up another meeting with BT1 to discuss this request (ID 1772737). - EMR, Issue ID 42
				Negotiation with customers about procedures and responsibilities	Getting consent for patch deployment at customers' premises, agreeing on the patch deployment dates and times, and establishing contact persons at the customer sites.	5	[Subject - Unix patching schedule confirmation] 24/7/20 - The requirements analysis revealed a major OS upgrade, not simple patching. The schedule is still being negotiated with [customer c1]. - Non-Win, Issue ID 7
				Frequent negotiation with vendors for support	Negotiate with vendors regarding delayed patch releases and support cases.	5	[Subject - hlt100hvd001 reverting, post patch-deployment issue] 17/4/19 – Incident INC0117732 raised, need further investigation, need to troubleshoot, and organise OOB. 3/5/19 – [P1-BT1] patched, ran troubleshooting, will need to raise with Microsoft and investigate. [P2-BT1] to send info to [P3-AT2] to raise with Microsoft. 10/5 - Email sent 9/5 to [P3-AT2] with info to raise Microsoft support case. - Win, Issue ID 39
			Collaborative decision-making	Making decisions about patch management activities	Team members making decisions about patch management tasks and task assignments.	2	[Subject - Proposal for a patch cycle change in [servers s1 and s2]] 4/4/18 - Discussions still ongoing for the decision. AT1 is still considering various options and has put them out in slides for discussion at the meeting. - EMR, Issue ID 2
				Making decisions about task assignments		1	
			Task delegation	Task assignment	Delegating the tasks to teams/team members	21	[Subject - EPAS and .NET patching - with no upgrade of .NET version] 14/2/18 - Apps team to do due diligence and test after each night patch group in week2-3 and if any identified issues - to advise for week4 pre-prod and prod servers (pull out of patching if needed). Ownership/investigation if any reported issues to be agreed - apps team as a first point? - EMR, Issue ID 3
				Define roles and responsibilities around tasks	Well-defined roles and responsibilities around patch management activities resulting in increased accountability.	10	[Subject - Vulnerabilities in .NET Core] 21/2/20 - .NET Core is not receiving updates. A new process is required to patch this version and a service request (SR) needs to be submitted for review and assessment. [P1-BT1] to raise the SR for the issue raised by BT1 on 7th Feb 2020. - Win, Issue ID 40
			Regularly review and update patch management process-related documentation	Regularly document patching tasks and decisions	Constantly document the decisions and activities to ease tracing back during troubleshooting post-deployment errors.	1	[Subject - New process provided for Server [sx016] but still require updates for [sx018]] 29/1/20 - Email received with manual instructions about the process. Instructions have been updated in the patching master sheet. - Win, Issue ID 54
				Regularly update the documentation	Consistently review the documentation and test any process changes internally before updating the documentation.	2	[Subject - Update documentation for the split of [servers s1 and s2] patching into two procedures] 13/12/19 - Finalising the documentation after testing internally for handover to 24x7. 10/1/20 - Documentation to be tested in February, will be ready for handover in March. - EMR, Task ID 24
		Patch information retrieval	Set strict timelines for patch download	Report which patches are download from patch releases	Each month after "patch Tue", Org B is to provide extract to Org A of the released patches for that month and a report including what patches will be applied in the month.	1	[Subject - Provide .NET report at the start of the patch cycle] 15/3/19 - Org A requests BT1 to provide an extract of .NET released patches every month and a report including what patches will be applied to what servers [...]. - EMR, Task ID 53
				Classify set of patches for application		1	
		Vulnerability Scanning, Assessment & Prioritisation	Plan alternatives for delayed patches	Analyse workarounds for suitability (what)	Alternatives for delayed patch releases and patching. e.g., review compliance to security when legacy systems cannot be upgraded.	3	[Subject - New 0-day vulnerability warning] 7/8/20 – The article points to an updated workaround with monthly rollup KB links, not a specific patch. 21/8/20 - Confirmation that this applies to 2012 R2,2016 and 2008 R2. Workaround decision to be addressed in the security forum. - EMR, Issue ID 56
				Assess timing of alternative remediation plans (when)	When to execute the alternative plans.	3	[Subject - Org A Change Freeze from 14th December to 12th January] 31/10/19 - Due to the change freeze in December, there will be no patching for December. 1/11/19 - AT2 wants to reserve the first two weeks of December for remediation and out-of-band patching of critical vulnerabilities as required. - Win, Task ID 34
			Define priorities for vulnerability remediation	Prioritisation based on patch severity and impact	Prioritising patches based on severity (vulnerability risk assessment) and its impact.	11	[Subject - LDAP GPO issues] 15/5/20 - GPO creation required for testing. Proceed with GPO creation part of the request so servers can be tested before wider rollout. DCs to be updated progressively with communication to Org A's ICT and Apps teams for verification. 12/6/20 - GPO updates to proceed again next week and creation of new GPO's required to be done first. 26/06 – This has taken a backseat due to critical patches being addressed as a priority. - Win, Issue ID 85
				Prioritisation based on patch type	Prioritising patches based on the patch type.	4	[Subject - OS security patches need to be tracked separated in the vulnerability remediation] 15/5/20 - [P1-AT1] requesting the OS security patches to be tracked separately to all other vulnerability remediation. Org B's report should only be addressing OS security patches anyway but can make sure to separate any non-OS remediation tasks. - EMR, Task ID 45
		Patch Testing	Define compliance policies and contingency plans for test failures	Develop security compliance policies	Compliance with security standards before deployment.	5	[Subject - [s1] new servers compliance] 4/12/18 - This item remains open until all new [s1] servers are fully compliant including security hardening prior to being pushed into production and support. 18/2/18 - Security approved the new [s1] servers, go-live completed on 18/2. - EMR, Task ID 1
				Develop contingency plan for test failures	Preparing plan B for patch testing errors.	4	[Subject - EPAS and .NET patching - with no upgrade of Net version] 14/2/18 - As discussed with the security team, EPAS17.3 Pre-Prod and Prod will go into 2-monthly cycle, starting March. Non-Prod will continue monthly. Plan is to move all EPAS 17.3 to monthly patching after 3 consecutive successful cycles. Success criteria = no business impact AND no major .NET version upgrades (e.g., updates within the .NET version). If any issues are identified after due diligence and testing, week4 pre-prod and prod servers to be pulled out of patching. - EMR, Issue ID 3
			Patch pre-requisites investigation	Pre-investigation of requisites	Set up the required requisites for patch deployment.	2	[Subject - Registry key missing for Knowledge Base (KB) ID in LDAP] 2/10/20 - Patches not installed on [servers s1 and s2] due to missing a registry key. [P1-BT1] to check settings and apply where missing. - Win, Task ID 24
				Update the progress of pre-requisite investigation		2	[Subject - March Pre-requisite Investigation relating to OS Patching] 17/4/20 - 3 new SSU pre-requisites to be applied. Updates have been populated on the SharePoint, but servers identified will be rectified as part of normal monthly patching cycle. 29 Servers currently require this patch and no registry changes or reboots required. 16 servers were flagged as auto and changed to semi auto to confirm application of the patches. Remaining 13 were manual. - Win, Issue ID 79
			Modify software configurations and	Identify the dependency changes	Identify and prepare the required dependencies for patch deployment.	2	[Subject - OCS Dependencies & Risks] 28/6/19 - Information required from OCS around any dependencies and risks of security patching (OS) to progress with patching. - Non-Win, Issue ID 2

gies	Strategies specific to each phase of the process		configurations and dependencies	Modify the configurations	Change the configurations to suit patch deployment.	1	[Subject - GPO creations (configs) to be done together] 12/6/20 - [P1-BT1] to create new GPOs at the same time early next week in preparation for the next round of additions for vulnerability remediation. - Win, Task ID 49
		Patch Deployment	Timely coordination of patch deployment schedules	Planning patch windows	Planning when to patch.	8	[Subject - Review of [s1] servers' patch windows: re-balancing and extended windows proposed] 13/3/19 - Org B proposes 4-hour windows starting at 18:00 each night. The first lot of servers to start Friday week 2 after the "Patch Tuesday". Proposal provided for Org A's consideration. - Win, Task ID 9
				Establish contact roles during and post-deployment for verification	Defining who will contact the external parties to confirm the deployment.	6	[Subject – Scheduled deployment at hospital [h1]] 24/1/19 - [P1-BT1] to ensure phone call after patching is completed tonight for [server s1]. 7/2/19 - Information provided this morning for build process to commence. - EMR, Issue ID 34
				Planning load balance	Planning the spread of servers evenly through patch windows (how to patch).	5	[Subject - Review of [s1] server's patch windows - rebalancing & extended windows proposed] 11/1/19 - Need to move back the start time for [s1] servers patching to "patch Tue" + 2days @21:00. For this we need to review the current schedule, agree on new extended patch windows (proposed 4-6 hrs windows) and to spread servers evenly through the windows. Ongoing work - for consideration by Org A once the new patch windows are clarified. 31/5/19 - Email directly to [P1-AT4] to advise that the change can be accommodated while awaiting 4h window approval and overall rebalancing approach approval. - Win, Issue ID 14
			Apply workarounds to maximise service availability	Cluster-based patching (based on patch types)	Configuring the group settings and deployment of the patch clusters for higher availability.	7	[Subject - Patch deployment failed at [server s1]] 24/1/18 - Single point of failure for [server s1]. AT1 to review the proposed design for clustering for high availability. Currently hard to obtain reboot timings and only one reboot is allowed. Ask the customer for an extended window and move the patching to the weekend. - Win, Task ID 6
				Extend patch windows for critical patches with multi reboots	Planning extended windows for patches that required multi reboots in out-of-band windows.	4	[Subject - Smartlink Servers extended patch windows] 31/5/19 - OOB patch needed to catch up. [P1-AT1] to advise the window, may need longer window (5-6 hrs suggested) and multi reboot. Note: Server [s1] is not under support. 14/6/19 - OOB patching completed, reintroduced to standard schedule. - EMR, Issue ID 20
				Balancing the load on servers	Balancing the load on servers during deployment to reduce service unavailability.	4	[Subject - Auto Rebalancing required] 3/10/19 - Optimal amount per window for auto patching is 32 servers. 8/10/19 – The spreadsheet sent to [P1-AT2] and [P2-AT5] for redistribution. 18/10 - Review in progress. 31/10 – [P3-BT1] suggesting maximum limit of servers per window is 32 per 4-hour block, client looking at which windows can shuffle. - Win, Issue ID 57
				Pre-load the patches offline to avoid exceeding patch windows	Pre-loading the patches offline to avoid patch deployment exceeding the allocated patch window.	1	[Subject -Pre-download for auto patching required by management] 31/10/19 – Org B is expecting formal NSSR from Org A - T2 to advise that pre-staging of patches or downloading "earlier" than the patching window to allow earlier patch download/multi-reboots. Org B is suggesting patches can be predownloaded on week 2 Thursday as there is no patching on that day. - Win, Issue ID 53
				Separate patch windows for failover patching	Maintaining backup servers to concurrently run the services while being rebooted.	2	[Subject - Request to change the patch window of HLT667U003] 11/4/19 – Currently set to 0000-0300, but the full backup of server patching happening during this window which causes slowness/issues with patching. Suggest changing the window to 0600-0900. - Win, Issue ID 36
			Manual deployment for complex patches to minimise damage	Redeployment of failed patching for unknown reasons	Redeployment of erroneous patches.	7	[Subject - Post-deployment error at [server s1] causing the printing service unavailable] 31/10/19 - This server was patched on 18/10 at 2 am-6 am. Due to the errors, patching will be done manually in November. - Win, Issue ID 6
				Manual patching of business critical servers	Manual patching for business-critical servers (e.g., life support machines).	3	[Subject - Software version issues on some servers] Issues with 47 Office patches on [server s1]-5 (multi-languages and x32 and x64 bit versions) and 7 Visual Studio related patches on SCMPRODCPMW001. 24/1/19 - A ticket raised for [server s1] for Org B to investigate Visual Studio patches and manually apply if required. 3/8/19 – Patches are installed by [P1-BT1] as OOB with [change request ch1]. This item can be closed now. - EMR, Issue ID 11
				Manual patching of complex legacy systems	Manually patching legacy software systems.	2	[Subject – 2008 patching update] 18/11/20 - Currently down to 11 servers out of the 273 with patching issues. It'll be a manual patching in OOB. Also, someone will be sitting there watching to rectify if any issues arise during the deployment. - EMR, Issue ID 69
			Agile deployment for executing changes	Execute complex patch deployment changes in small iterations	This is done as a precautionary measure and to build confidence around the changes. Also, because a change in the patching process could create devastating consequences.	6	[Subject - Review of patch cycle timings] 22/2/18 - Org A is considering going for bi-monthly deployment cycles for .NET patching and then move to monthly deployment after the confidence is built. - EMR, Task ID 2
		Post-Deployment Patch Verification	Establish post-deployment verification procedures	Verification through gathering client feedback	Define set of procedures for post-deployment patch verification (how to verify).	4	[Subject - Automated second rescan for reboots] 31/10/19 - [P1-BT1] raised this issue, he has configured the window to rescan for missing patches and conduct a second reboot if required. No issues during patching, seeking client feedback for verification. - EMR, Task ID 28
				Verification through monitoring		2	[Subject – Tracking of GPO applications not intended] 24/07 - Rollback status: no reported issue due to the rollout, rollback not required at this point. Keep monitoring for one more week. - Win, Issue ID 91
				Verification through analysis of log system status		2	[Subject – SQL tlogs and IIS logs on C drives - capacity issue] 22/2/19 - Org B to run report of servers that have any SQL or IIS logs/data on C drive and provide to [P1-AT2]. Then, apps teams to look at moving off C. - Win, Issue ID 19
				Verification through periodic scans		2	[Subject – Internal DXC server vulnerabilities] 20/3/20 – The scans identified application vulnerabilities on internal Org B's Jump hosts. The remediation is being investigated. – Non-Win, Issue ID 15
			Collectively handle post-deployment issues	Find workarounds for failed deployments (countermeasures)	Workarounds such as Out-Of-Band (OOB) patching to catch up with missing patches, reverting back installed erroneous patches, and uninstalling EOL and unnecessary software.	6	[Subject - Tracking of GPO applications that are not intended] 10/7/20 - Information sent to [P1-AT2] and [P2-AT2]. Patch [p1] will need a rollback. 24/7/20 - Rollback executed, no reported issue due to the rollback. Keep open for one more meeting for monitoring. - Win, Task ID 51
				Investigation of root causes (troubleshooting)	Identify the root causes for issues identified post-deployment.	3	[Subject – Issue with Print Servers Patching] 16/10/20 - [P1-BT1] did some investigation and it was found that a KB was missing in one server. Since this KB is only for 2016 server, it is inapplicable. Need verification on that. [P1-AT5] to double check on the vulnerabilities scan. - Win, Issue ID 54
			Document deployment status of every patch	Document the status of each task	Record the status of each patch in the Patching Tracker.	3	[Subject - [Servers s1 and s2] successfully patched] 24/7/20 - [...] No further issues experienced since patching. Manual instructions and deployment status updated in the shared tracker. Will be kept in-monitor for another couple of weeks. - EMR, Issue ID 47

Appendix C

The Interview Guide

PRE-INTERVIEW QUESTIONNAIRE TO COLLECT THE DEMOGRAPHICS

Question	**Choices Provided**
Full name	
What is your current role (job title)?	
How long have you been in the industry (total industry experience)?	
What is your team (team name)?	
What is the team size (number of people)?	
Which types of patches are you working with?	Security patches, Non-security patches, Both, Other (please specify)
What types of software components are you in charge of patching?	Operating System (OS), Applications, Custom Inhouse Software Applications, End of Life (EOL) Software, Other (please specify)
How many machines/devices do you manage?	
What type of machines/devices do you manage?	Client machines (e.g., desktops, laptops, etc.), Servers, Mobile devices (e.g., phones and tablets), Routers/network devices, Embedded devices/Internet of Things, Other (please specify)
What are the operating systems on the machines/devices that you manage?	

INTERVIEW QUESTIONS

Warm-up Questions

Q1. Can you briefly walk me through your responsibilities in the security patch management process?

Current tools/automation support in security patch management

Q2. What tools do you use in security patch management tasks?
- When do you use them (i.e., which task in which phase)?
- How does the tool help in security patch management tasks?
- Does the tool require human involvement at any point to proceed with the task?
- If yes, can you explain why? When does it need human input?

Limitations of current tools/automation support in security patch management

Q3. What challenges do you experience in dealing with these tools?
- Can you give me some examples of such challenges you have experienced?
- Why do you think they are challenging?
- In your experience, what situations are the most challenging to handle and why?
- How do you overcome these challenges? Can you please provide some examples?

Needs for better automation/tool support in security patch management

Q4.1. What are the situations or security patch management tasks that are currently not supported by automation in your process?

- Can you explain why they are manual?

Q4.2. What tasks do you wish would have been better supported by the technology, if any?

- What kind of support do you expect from an improved tool?
- How do you anticipate such kind of a tool will help reduce patching delays?
- Any specific features you think are important to have in such a tool?

Q4.3. Have you heard about artificial intelligence (AI) or machine learning?

- How do you think AI can help bring efficiency in the process?
- In your opinion, do you think it is possible to replace human involvement with AI in the security patch management process?

Q4.4: Any related points that we missed but you would like to reflect on?

Appendix D

The Observation Protocol for the Grounded Theory Study

TABLE D.1. Observation protocol

Topic	Question
Participants	What are the roles and other details of the participants? • Number of attendees • Role (manager, system administrator, etc.) • Affiliation (Org A or Org B) • Team (Security, Windows, Server, etc.) Is someone acting as a facilitator? • Who? • How is he/she facilitating the meeting?
Communication	What is the communication channel? How does the communication happen? • Directed / indirect questions? • Active participation in communication? • Any roles that are most active in communication?
Activities	What are the various discussions and activities? • Topics discussed • Challenges discussed • Activities (demonstrations etc.)
Objects	What resources/media are used? • Presentation slides, excel sheets, tools, etc.
Collaboration	How do the participants interact and corporate with each other?
Events	Are there any particular events or anything unanticipated?
Time	When does the meeting start? What is the sequence of events? When does the meeting end?
Goals	What are the participants trying to accomplish?
Feelings	How is the atmosphere?
Closing	How is the meeting ended? Is there a post-meeting planned? Is there anything discussed about the next meeting?

Appendix E

Selected Primary Studies in the Literature Review

TABLE E.1. Selected primary studies in the SLR.

ID	Title	Author(s)	Venue	Year
P1	Computer security and operating system updates	G. Post, A. Kagan	Information and Software Technology	2003
P2	Reducing internet-based intrusions: Effective security patch management	B. Brykczynski, R.A. Small	IEEE Software	2003
P3	Safe software updates via multi-version execution	P. Hosek, C. Cadar	International Conference on Software Engineering	2013
P4	The attack of the clones: A study of the impact of shared code on vulnerability patching	A. Nappa, R. Johnson, L. Bilge, J. Caballero, T. Dumitraș	IEEE Symposium on Security and Privacy	2015
P5	Identifying Information Disclosure in Web Applications with Retroactive Auditing	H. Chen, T. Kim, X. Wang, N. Zeldovich, M. F. Kaashoek	USENIX Symposium on Operating Systems Design and Implementation	2014
P6	Improving VRSS-based vulnerability prioritization using analytic hierarchy process	Q. Liua, Y. Zhanga, Y. Konga, Q. Wu	Journal of Systems and Software	2012
P7	Improving CVSS-based vulnerability prioritization and response with context information	C. Frühwirth and T. Männistö	International Symposium on Empirical Software Engineering and Measurement	2009
P8	Keepers of the Machines: Examining How System Administrators Manage Software Updates	F. Li, L. Rogers, A. Mathur, N. Malkin and M. Chetty	USENIX Conference on Usable Privacy and Security	2019
P9	Towards A Self-Managing Software Patching Process Using Black-Box Persistent-State Manifests	J. Dunagan, R. Roussev, B. Daniels, A. Johnson, C. Verbowski and Y.M. Wang	IEEE International Conference on Autonomic Computing	2004
P10	Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators	C. Tiefenau, M. Häring, K. Krombholz, E. von Zezschwitz	USENIX Symposium on Usable Privacy and Security	2020
P11	Patch management automation for enterprise cloud	H. Huang, S. Baset, C. Tang, A. Gupta, K.M. Sudhan, F. Feroze, R. Garg, S. Ravichandran	IEEE Network Operations and Management Symposium	2012
P12	Shadow Patching: Minimizing Maintenance Windows in a Virtualized Enterprise Environment	D. Le, J. Xiao, H. Huangy, H. Wang	International Conference on Network and Service Management	2014

ID	Title	Author(s)	Venue	Year
P13	VULCAN: Vulnerability Assessment Framework for Cloud Computing	P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, A. Singhal	International Conference on Software Security and Reliability	2013
P14	VRank: A Context-Aware Approach to Vulnerability Scoring and Ranking in SOA	J. Jiang, L. Ding, E. Zhai, T. Yu	International Conference on Software Security and Reliability	2012
P15	"Anyone Else Seeing this Error?": Community, System Administrators, and Patch Information	A. Jenkins, P. Kalligeros, K. Vaniea, M. K. Wolters	IEEE European Symposium on Security and Privacy	2020
P16	Staged Deployment in Mirage, an Integrated Software Upgrade Testing and Distribution System	O. Crameri, N. Knezevic, D. Kostic, R. Bianchini, W. Zwaenepoel	ACM SIGOPS Operating Systems Review	2007
P17	Devirtualizable Virtual Machines Enabling General, Single-Node, Online Maintenance	D.E. Lowell, Y. Saito, E.J. Samberg	ACM SIGARCH Computer Architecture News	2004
P18	An automated framework for managing security vulnerabilities	A. Al-Ayed, S.M. Furnell, D. Zhao, P.S. Dowland	Information Management and Computer Security	2005
P19	A cross-site patch management model and architecture design for large scale heterogeneous environment	C.W. Chang, D.R. Tsai, J.M. Tsai	International Carnahan Conference on Security Technology	2005
P20	Security patch management	F.M. Nicastro	Information Systems Security	2003
P21	Intrusion recovery for database-backed web applications	R. Chandra, T. Kim, M. Shah, N. Narula, N. Zeldovich	ACM Symposium on Operating Systems Principles	2011
P22	MAD: A visual analytics solution for Multi-step cyber Attacks Detection	M. Angelini, S. Bonomi, S. Lenti, G. Santucci, S. Taggi	Journal of Computer Languages	2019
P23	Designing an efficient framework for vulnerability assessment and patching (VAP) in virtual environment of cloud computing	R. Patil, C. Modi	Journal of Supercomputing	2019
P24	A new cost-saving and efficient method for patch management using blockchain	Y.Kim, Y. Won	Journal of Supercomputing	2019
P25	Linux patch management: With security assessment features	S. Midtrapanon, G. Wills	International Conference on Internet of Things, Big Data and Security	2019
P26	Vulnus: Visual vulnerability analysis for network security	M. Angelini, G. Blasilli, T. Catarci, S. Lenti, G. Santucci	IEEE Transactions on Visualization and Computer Graphics	2018
P27	Handling vulnerabilities with mobile agents in order to consider the delay and disruption tolerant characteristic of military networks	T. Aurisch, A. Jacke	International Conference on Military Communications and Information Systems	2018
P28	Green WSUS	S. Kadry, C. Jouma	International Conference on Future Energy, Environment and Materials	2012

ID	Title	Author(s)	Venue	Year
P29	Checking running and dormant virtual machines for the necessity of security updates in cloud environments	R. Schwarzkopf, M. Schmidt, C. Strack, B. Freisleben	IEEE International Conference on Cloud Computing Technology and Science	2011
P30	A race for security: Identifying vulnerabilities on 50 000 hosts faster than attackers	M. Procházka, D. Koüril, R. Wartel, C. Kanellopoulos, C. Triantafyllidis	The International Symposium on Grids and Clouds and the Open Grid Forum	2011
P31	Multi-layered virtual machines for security updates in grid environments	R. Schwarzkopf, M. Schmidt, N. Fallenbeck, B. Freisleben	Euromicro Conference on Software Engineering and Advanced Applications	2009
P32	A study and implementation of vulnerability assessment and misconfiguration detection	C.H. Lin, C.H. Chen, C.S. Laih	IEEE Asia-Pacific Services Computing Conference	2019
P33	Using the vulnerability information of computer systems to improve the network security	Y.P. Lai, P.L. Hsia	Computer Communications	2007
P34	Analyzing enterprise network vulnerabilities	M. Nyanchama, M. Stefaniu	Information Systems Security	2003
P35	The dilemma of security patches	G. Post, A. Kagan	Information Systems Security	2002
P36	Always up-to-date: Scalable offline patching of VM images in a compute cloud	W. Zhou, P. Ning, X. Zhang, G. Ammons, R. Wang, V. Bala	Annual Computer Security Applications Conference	2010
P37	A process framework for stakeholder-specific visualization of security metrics	T. Hanauer, W. Hommel, S. Metzger, D. Pöhn	International Conference on Availability, Reliability and Security	2018
P38	VULCON: A system for vulnerability prioritization, mitigation, and management	K.A. Farris, A. Shah, G. Cybenko, R. Ganesan, S. Jajodia	ACM Transactions on Privacy and Securitys	2018
P39	Patch auditing in infrastructure as a service clouds	L. Litty, D. Lie	International Conference on Virtual Execution Environments	2011
P40	Designing a distributed patch management security system	Y. Nunez, F. Gustavson, F. Grossman, C. Tappert	International Conference on Information Society	2010
P41	Beyond heuristics: Learning to classify vulnerabilities and predict exploits	M. Bozorgi, L.K. Saul, S. Savage, G.M. Voelker	ACM SIGKDD International Conference on Knowledge Discovery and Data Mining	2010
P42	NetGlean: A methodology for distributed network security scanning	G.W. Manes, D. Schulte, S. Guenther, S. Shenoi	Journal of Network and Systems Management	2005
P43	RL-BAGS: A tool for smart grid risk assessment	Y. Wadhawan, C. Neuman	International Conference on Smart Grid and Clean Energy Technologies	2018
P44	From patching delays to infection symptoms: Using risk profiles for an early discovery of vulnerabilities exploited in the wild	C. Xiao, A. Sarabi, Y. Liu, B. Li, M. Liu, T. Dumitraș	USENIX Security Symposium	2018
P45	PKG-VUL: Security vulnerability evaluation and patch framework for package-based systems	J.H. Lee, S.G. Sohn, B.H. Chang, T.M. Chung	ETRI Journal	2009

ID	Title	Author(s)	Venue	Year
P46	A Study of Integrity on the Security Patches System Using PM-FTS	K.J. Kim, M. Kim	Wireless Personal Communications	2017
P47	Patch integrity verification method using dual electronic signatures	J. Kim, Y. Won	Journal of Information Processing Systems	2017
P48	Software asset analyzer: A system for detecting configuration anomalies	X. Li, P. Avellino, J. Janies, M.P. Collins	IEEE Military Communications Conference	2016
P49	Vulnerabilities scoring approach for cloud saas	Z. Li, C. Tang, J. Hu, Z. Chen	International Conference on Ubiquitous Intelligence and Computing, International Conference on Advanced and Trusted Computing, International Conference on Scalable Computing and Communications and its associated Workshops	2015
P50	Risk assessment and mitigation at the information technology companies	B. Marx, D. Oosthuizen	Risk Governance & Control: Financial Markets and Institutions	2016
P51	A proposed framework for proactive vulnerability assessments in cloud deployments	K.A. Torkura, F. Cheng, C. Meinel	International Conference for Internet Technology and Secured Transactions	2015
P52	Elementary Risks: Bridging Operational and Strategic Security Realms	W. Kanoun, S. Papillon, S. Dubus	International Conference on Signal-Image Technology and Internet-Based Systems	2015
P53	Mining social networks for software vulnerabilities monitoring	S. Trabelsi, H. Plate, A. Abida, M.M. Ben Aoun, A. Zouaoui, C. Missaoui, S. Gharbi, A. Ayari	International Conference on New Technologies, Mobility and Security	2015
P54	A VMM-level approach to shortening downtime of operating systems reboots in software updates	H. Yamada, K. Kono	IEICE Transactions on Information and Systems	2014
P55	A vulnerability life cycle-based security modeling and evaluation approach	G.V. Marconato, M. Kaâniche, V. Nicomette	The Computer Journal	2013
P56	iDispatcher: A unified platform for secure planet-scale information dissemination	M.S. Rahman, G. Yan, H.V. Madhyastha, M. Faloutsos, S. Eidenbenz, M. Fisk	Peer-to-Peer Networking and Applications	2013
P57	Efficient patch-based auditing for web application vulnerabilities	T. Kim, R. Chandra, N. Zeldovich	USENIX Symposium on Operating Systems Design and Implementation	2012
P58	Instant OS updates via userspace checkpoint-and-restart	S. Kashyap, C. Min, B. Lee, T. Kim, P. Emelyanov	USENIX Annual Technical Conference	2016
P59	Tachyon: Tandem execution for efficient live patch testing	M. Maurer, D. Brumley	USENIX Security Symposium	2012
P60	Efficient online validation with delta execution	J. Tucek, W. Xiong, Y. Zhou	International Conference on Architectural Support for Programming Languages and Operating Systems	2009

ID	Title	Author(s)	Venue	Year
P61	Enterprise Vulnerability Management and Its Role in Information Security Management	M. Nyanchama	Information Security Management	2005
P62	A Machine Learning-based Approach for Automated Vulnerability Remediation Analysis	F. Zhang, P. Huff, K. McClanahan, Q. Li	IEEE Conference on Communications and Network Security	2020
P63	Reducing Downtime Due to System Maintenance and Upgrades	S. Potter and J. Nieh	USENIX Systems Administration Conference	2005
P64	Increasing virtual machine security in cloud environments	R. Schwarzkopf, M. Schmidt, C. Strack, S. Martin and B. Freisleben	Journal of Cloud Computing: Advances, Systems and Applications	2012
P65	Understanding Software Patching	J. Dadzie	ACM Queue	2005
P66	Patching the Enterprise	G. Brandman	ACM Queue	2005
P67	Why Do Upgrades Fail and What Can We Do about It?	T. Dumitras, P. Narasimhan	ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing	2009
P68	Transparent Mutable Replay for Multicore Debugging and Patch Validation	N. Viennot, S. Nair, J. Nieh	ACM SIGARCH computer architecture news	2013
P69	A quantitative evaluation of vulnerability scanning	H. Holm, T. Sommestad	Information Management & Computer Security	2011
P70	Evaluation of Security Vulnerability Scanners for Small and Medium Enterprises Business Networks Resilience towards Risk Assessment	I. Chalvatzis, D. A. Karras, R. C. Papademetriou	IEEE International Conference on Artificial Intelligence and Computer Applications	2019
P71	SLA-driven Applicability Analysis for Patch Management	B. Yang, N. Ayachitula (Arun), S. Zeng, R. Puri	IFIP/IEEE International Symposium on Integrated Network Management	2011
P72	A Design for a Hyperledger Fabric Blockchain-Based Patch-Management System	K. T. Song, S. I. Kim, S. H. Kim	Journal of Information Processing Systems	2020

Appendix F

Data Extraction Form in the Literature Review

Table F.1. Data extraction form used in the SLR.

ID	Data item	Description	RQ (Section 3.3.1)
D1	Title	The title of the paper	Demographic data
D2	Author(s)	The author(s) of the paper	Demographic data
D3	Venue	The publication venue	Demographic data
D4	Year	The year of the publication	Demographic data
D5	Publication type	The type of publication (e.g., conference paper, journal paper)	Demographic data
D6	Area of Focus	The focus of the paper in the security patch management process	Demographic data
D7	Target User(s)	The intended users (e.g., security manager)	Demographic data
D8	Application Domain	The target application domain (e.g., cloud)	Demographic data
D9	Solution Type	The type of solution i.e., Practice, Approach, Tool	Demographic data
D10	Research Type	The type of research i.e., Validation Research, Evaluation Research, Solution Proposal, Experience Paper, Philosophical Paper, Opinion Paper	Demographic data
D11	Challenges	The reported socio-technical challenges	RQ1.1
D12	Solutions: Approaches and tools	The proposed approaches and tools (key elements), and their strengths and capabilities documenting how the solution addresses the reported challenges	RQ1.2.1
D13	Solutions: Practices	The reported practices to successfully implement security patch management	RQ1.2.2
D14	Evaluation	The type of evaluation used to assess the reported solutions, and the level of rigour and industrial relevance	RQ1.3
D15	Limitations and Threats to Validity	The limitations of the solution and the reported threats to validity	Discussion

ID	Data item	Description	RQ (Section 3.3.1)
D16	Future Work	The reported future work	Discussion

Appendix G

Approved Ethics Application

RESEARCH SERVICES
OFFICE OF RESEARCH ETHICS, COMPLIANCE AND INTEGRITY
THE UNIVERSITY OF ADELAIDE

LEVEL 4, RUNDLE MALL PLAZA
50 RUNDLE MALL
ADELAIDE SA 5000 AUSTRALIA

TELEPHONE +61 8 8313 5137
FACSIMILE +61 8 8313 3700
EMAIL hrec@adelaide.edu.au

CRICOS Provider Number 00123M

Our reference 34332

02 April 2020

Professor Ali Babar
School of Computer Science

Dear Professor Babar

ETHICS APPROVAL No:	H-2020-035
PROJECT TITLE:	Human-AI Collaboration for Securing Organizational Systems: A Case of Software Security Patch Management

The ethics application for the above project has been reviewed by the Executive, Human Research Ethics Committee and is deemed to meet the requirements of the *National Statement on Ethical Conduct in Human Research 2007 (Updated 2018)* involving no more than low risk for research participants.

You are authorised to commence your research on:	02/04/2020
The ethics expiry date for this project is:	30/04/2023

NAMED INVESTIGATORS:

Chief Investigator:	Professor Ali Babar
Student - Postgraduate Doctorate by Research (PhD):	Mrs Nesara Madugoda S. Dissanayakege
Associate Investigator:	Dr Asangi Jayatilaka
Associate Investigator:	Dr Mansooreh Zahedi

CONDITIONS OF APPROVAL: Thank you for addressing the feedback. The revised ethics application provided on the 17th of March 2020 has been approved.

Ethics approval is granted for three years and is subject to satisfactory annual reporting. The form titled Annual Report on Project Status is to be used when reporting annual progress and project completion and can be downloaded at http://www.adelaide.edu.au/research-services/oreci/human/reporting/. Prior to expiry, ethics approval may be extended for a further period.

Participants in the study are to be given a copy of the information sheet and the signed consent form to retain. It is also a condition of approval that you immediately report anything which might warrant review of ethical approval including:

- serious or unexpected adverse effects on participants,
- previously unforeseen events which might affect continued ethical acceptability of the project,
- proposed changes to the protocol or project investigators; and

- the project is discontinued before the expected date of completion.

Yours sincerely,

Professor Paul Delfabbro
Convenor

The University of Adelaide

References

[1] P. Mell, T. Bergeron, D. Henning *et al.*, “Creating a patch and vulnerability management program,” *NIST Special Publication*, vol. 800, p. 40, 2005.

[2] F. Li, L. Rogers, A. Mathur, N. Malkin, and M. Chetty, “Keepers of the machines: Examining how system administrators manage software updates,” in *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. USENIX Association, 2019, pp. 273–288.

[3] Accenture, “2020 cyber threatscape report,” https://www.accenture.com/_acnmedia/PDF-136/Accenture-2020-Cyber-Threatscape-Full-Report.pdf.

[4] B. Thomas, “New windows vulnerabilities highlight patch management challenges,” https://www.bitsight.com/blog/new-windows-vulnerabilities-highlight-patch-management-challenges.

[5] S. Coleman, “Cyber security review,” https://www.cybersecurity-review.com/what-if-you-cant-patch/.

[6] R. Brandom, “Former equifax ceo blames breach on a single person who failed to deploy patch,” https://www.theverge.com/2017/10/3/16410806/equifax-ceo-blame-breach-patch-congress-testimony.

[7] M. Lee, “Equifax data breach impacts 143 million americans,” https://www.forbes.com/sites/leemathews/2017/09/07/equifax-data-breach-impacts-143-million-americans/#7aa9c117356f.

[8] L. H. Newman, “Equifax officially has no excuse,” https://www.wired.com/story/equifax-breach-no-excuse/.

[9] N. P. Melissa Eddy, “Cyber attack suspected in german woman’s death,” https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomeware-death.html?smid=tw-share.

[10] C. for Internet Security (CIS). (2022) Cyber attacks: In the healthcare sector. [Online]. Available: https://www.cisecurity.org/insights/blog/cyber-attacks-in-the-healthcare-sector

[11] R. Southwick. (2022) Cyberattacks in healthcare surged last year, and 2022 could be even worse. [Online]. Available: https://www.chiefhealthcareexecutive.com/view/cyberattacks-in-healthcare-surged-last-year-and-2022-could-be-even-worse

[12] N. A. Office, “Investigation: Wannacry cyber attack and the nhs,” Department of Health, Report by the Comptroller and Auditor General, October 2017.

[13] S. Ghafur, S. Kristensen, K. Honeyford, G. Martin, A. Darzi, and P. Aylin, “A retrospective impact analysis of the wannacry cyberattack on the nhs,” *NPJ Digital Medicine*, vol. 2, p. 98, 2019.

[14] Kaspersky, "What is wannacry ransomware?" https://www.kaspersky.com/resource-center/threats/ransomware-wannacry.

[15] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, 2018, pp. 973–990. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[16] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (S&P'19)*. IEEE, 2019, pp. 1–19.

[17] M. Souppaya and K. Scarfone, "Guide to enterprise patch management technologies," *NIST Special Publication*, vol. 800, p. 40, 2013.

[18] NIST, "Procedures for handling security patches," *Special Publication (SP) 800-40*, 2002.

[19] P. Mell, T. Bergeron, and D. Henning, "Creating a patch and vulnerability management program," *NIST Special Publication (SP) 800-40 Revision 2*, 2005.

[20] C. Tiefenau, M. Häring, K. Krombholz, and E. von Zezschwitz, "Security, availability, and multiple information sources: Exploring update behavior of system administrators," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 2020, pp. 239–258.

[21] U. Gentile and L. Serio, "Survey on international standards and best practices for patch management of complex industrial control systems: the critical infrastructure of particle accelerators case study," *International Journal of Critical Computer-Based Systems*, vol. 9, no. 1-2, pp. 115–132, 2019.

[22] H. Huang, S. Baset, C. Tang, A. Gupta, K. M. Sudhan, F. Feroze, R. Garg, and S. Ravichandran, "Patch management automation for enterprise cloud," in *IEEE Network Operations and Management Symposium*. IEEE, 2012, pp. 691–705.

[23] Automox, "Palo alto 2022 incident response report," https://www.automox.com/blog/palo-alto-2022-incident-response-report.

[24] D. Dey, A. Lahiri, and G. Zhang, "Optimal policies for security patch management," *INFORMS Journal on Computing*, vol. 27, no. 3, pp. 462–477, 2015.

[25] O. Crameri, N. Knezevic, D. Kostic, R. Bianchini, and W. Zwaenepoel, "Staged deployment in mirage, an integrated software upgrade testing and distribution system," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 221–236, 2007.

[26] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–45, 2019.

[27] F. E. Emery and E. L. Trist, "Socio-technical systems," *Churchman, C.W., Verhulst, M. (Eds.), Management Science Models and Techniques*, vol. 2, p. 83–97, 1960.

[28] T. D. Barbara A Kitchenham and M. Jorgensen, “Evidence-based software engineering,” in *Proceedings. 26th International Conference on Software Engineering*. IEEE, 2004, pp. 273–281.

[29] S. Keele *et al.*, “Guidelines for performing systematic literature reviews in software engineering,” Technical report, Version 2.3 EBSE Technical Report. EBSE, Tech. Rep., 2007.

[30] T. A. Schwandt, *Qualitative Inquiry*. London: Sage, 1997.

[31] M. N. Marshall, “Sampling for qualitative research,” *Family practice*, vol. 13, no. 6, pp. 522–526, 1996. [Online]. Available: https://doi.org/10.1093/fampra/13.6.522

[32] R. K. Yin, *Case study research and applications: Design and methods*. Sage publications, 2017.

[33] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[34] A. L. Strauss and J. M. Corbin, *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory*, 2nd ed. Sage, 1998.

[35] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Transaction, 1967.

[36] K.-J. Stol, P. Ralph, and B. Fitzgerald, “Grounded theory in software engineering research: A critical review and guidelines,” in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. ACM, 2016, pp. 120–131.

[37] A. L. Strauss and J. M. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 3rd ed. Sage, 2007.

[38] Q. International, “Nvivo qualitative data analysis software,” https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home, 2023.

[39] M. V. Genuchten, “Why is software late? an empirical study of reasons for delay in software development,” *IEEE Transactions on Software Engineering*, vol. 17, no. 6, pp. 582–590, 1991.

[40] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. London: Adisson-Wesley, 1975.

[41] S. B. Merriam, *Qualitative Research and Case Study Applications in Education. Revised and Expanded from*. ERIC, 1998.

[42] K. Charmaz, *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage, 2006.

[43] L. Birt, S. Scott, D. Cavers, and C. C. F. Walter, “Member checking: A tool to enhance trustworthiness or merely a nod to validation?” *Qualitative Health Research*, vol. 26, no. 13, pp. 1802–1811, 2016. [Online]. Available: https://doi.org/10.1177/1049732316654870

[44] B. G. Glaser, *Basics of Grounded Theory Analysis: Emergence vs Forcing.* Mill Valley, CA: Sociology Press, 1992.

[45] B. G. Glaser, *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory.* Mill Valley, CA: Sociology Press, 1978.

[46] R. Hoda, J. Noble, and S. Marshall, "The impact of inadequate customer collaboration on self-organizing agile teams," *Information and Software Technology*, vol. 53, no. 5, pp. 521–534, 2011. [Online]. Available: https://doi.org/10.1016/j.infsof.2010.10.009

[47] I. Cook and M. Crang, *Doing ethnographies*, U. of East Anglia Norwich, Ed. Sociology Press, 1995.

[48] H. Sharp, Y. Dittrich, and C. R. B. de Souza, "The role of ethnographic studies in empirical software engineering," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 786–804, 2016.

[49] J. P. Spradley, *Participant Observation*, 1st ed. Austin, TX: Rinehart and Winston, 1980.

[50] C. Urquhart, *Grounded Theory for Qualitative Research: A Practical Guide.* Sage, 2013.

[51] S. Georgieva and G. Allan, "Best practices in project management through a grounded theory lens," *Electronic Journal of Business Research Methods*, vol. 6, no. 1, 2008.

[52] R. Hoda, J. Noble, and S. Marshall, "Developing a grounded theory to explain the practices of self-organizing agile teams," *Empirical Software Engineering*, vol. 17, no. 6, pp. 609–639, 2012.

[53] B. G. Glaser, *The Grounded Theory Perspective III: Theoretical Coding.* Mill Valley, CA: Sociology Press, 2005.

[54] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management-a systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, p. 106771, 2021. [Online]. Available: https://doi.org/10.1016/j.infsof.2021.106771

[55] A. Jenkins, P. Kalligeros, K. Vaniea, and M. K. Wolters, "“anyone else seeing this error?”: Community, system administrators, and patch information," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P).* IEEE, 2020, pp. 105–119.

[56] N. Dissanayake, M. Zahedi, A. Jayatilaka, and M. A. Babar, "A grounded theory of the role of coordination in software security patch management," in *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21).* ACM, New York, NY, USA, 2021. [Online]. Available: https://doi.org/10.1145/3468264.3468595

[57] T. Gerace and H. Cavusoglu, "The critical elements of patch management," in *Proceedings of the 33rd annual ACM SIGUCCS conference on User services*, 2005, pp. 98–101.

[58] B. Brykczynski and R. A. Small, "Reducing internet-based intrusions: Effective security patch management," *IEEE software*, vol. 20, no. 1, pp. 50–57, 2003.

[59] F. M. Nicastro, "Security patch management," *Inf. Secur. J. A Glob. Perspect.*, vol. 12, no. 5, pp. 5–18, 2003.

[60] H. Cavusoglu, H. Cavusoglu, and J. Zhang, "Security patch management: Share the burden or share the damage?" *Management Science*, vol. 54, no. 4, pp. 657–670, 2008.

[61] H. Cavusoglu, H. Cavusoglu, and J. Zhang, "Economics of security patch management," in *WEIS*. Citeseer, 2006.

[62] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser, "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 942–953.

[63] M. Hicks and S. Nettles, "Dynamic software updating," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 27, no. 6, pp. 1049–1096, 2005.

[64] E. Miedes and F. D. Munoz-Escoı, "Dynamic software update," *Instituto Universitario Mixto Tecnológico de Informática, Universitat Politècnica de València, Technical Report ITI-SIDI-2012/004*, 2012.

[65] H. Seifzadeh, H. Abolhassani, and M. S. Moshkenani, "A survey of dynamic software updating," *Journal of Software: Evolution and Process*, vol. 25, no. 5, pp. 535–568, 2013.

[66] A. R. Gregersen, M. Rasmussen, and B. N. Jørgensen, "State of the art of dynamic software updating in java," in *International Conference on Software Technologies*. Springer, 2013, pp. 99–113.

[67] I. Mugarza, J. Parra, and E. Jacob, "Analysis of existing dynamic software updating techniques for safe and secure industrial control systems," *International journal of safety and security engineering*, vol. 8, no. 1, pp. 121–131, 2018.

[68] B. H. Ahmed, S. P. Lee, M. T. Su, and A. Zakari, "Dynamic software updating: a systematic mapping study," *IET Software*, vol. 14, no. 5, pp. 468–481, 2020.

[69] C. Islam, V. Prokhorenko, and M. A. Babar, "Runtime software patching: Taxonomy, survey and future directions," *Journal of Systems and Software*, p. 111652, 2023.

[70] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[71] "Cambridge dictionary," https://dictionary.cambridge.org/.

[72] Y. Dittrich, "What does it mean to use a method? towards a practice theory for software engineering," *Information and Software Technology*, vol. 70, pp. 220–231, 2016.

[73] K. Schmidt, "The concept of 'practice': What's the point?" in *COOP 2014-Proceedings of the 11th International Conference on the Design of Cooperative Systems, 27-30 May 2014, Nice (France).* Springer, 2014, pp. 427–444.

[74] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering–a tertiary study," *Information and software technology*, vol. 52, no. 8, pp. 792–805, 2010.

[75] M. Zahedi, M. Shahin, and M. A. Babar, "A systematic review of knowledge sharing challenges and practices in global software development," *International Journal of Information Management*, vol. 36, no. 6, pp. 995–1019, 2016.

[76] M. Shahin, M. A. Babar, and M. A. Chauhan, "Architectural design space for modelling and simulation as a service: A review," *Journal of Systems and Software*, vol. 170, p. 110752, 2020.

[77] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[78] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[79] M. Shahin, P. Liang, and M. A. Babar, "A systematic review of software architecture visualization techniques," *Journal of Systems and Software*, vol. 94, pp. 161–185, 2014.

[80] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019.

[81] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[82] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[83] D. S. Cruzes and T. Dybå, "Research synthesis in software engineering: A tertiary study," *Information and Software Technology*, vol. 53, no. 5, pp. 440–455, 2011.

[84] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, 2008, pp. 1–10.

[85] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[86] "Common vulnerability scoring system," https://www.first.org/cvss/.

[87] L. Chen and M. A. Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology*, vol. 53, no. 4, pp. 344–362, 2011.

[88] M. Shaw, “Writing good software engineering research papers,” in *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE, 2003, pp. 726–736.

[89] C. Zannier, G. Melnik, and F. Maurer, “On the success of empirical studies in the international conference on software engineering,” in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 341–350.

[90] N. Fenton, S. L. Pfleeger, and R. L. Glass, “Science and substance: A challenge to software engineers,” *IEEE software*, vol. 11, no. 4, pp. 86–95, 1994.

[91] E. Kamar, “Directions in hybrid intelligence: Complementing ai systems with human intelligence.” in *IJCAI*, 2016, pp. 4070–4073.

[92] D. Dellermann, A. Calma, N. Lipusch, T. Weber, S. Weigel, and P. Ebel, “The future of human-ai collaboration: a taxonomy of design knowledge for hybrid intelligence systems,” in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[93] J. A. Crowder, J. Carbone, and S. Friess, “Human–ai collaboration,” in *Artificial Psychology.* Springer, 2020, pp. 35–50.

[94] D. Kirk and S. G. MacDonell, “Investigating a conceptual construct for software context,” in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[95] T. Dybå, D. I. Sjøberg, and D. S. Cruzes, “What works for whom, where, when, and why? on the role of context in empirical software engineering,” in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, 2012, pp. 19–28.

[96] K. Petersen and C. Wohlin, “Context in industrial software engineering research,” in *2009 3rd International Symposium on Empirical Software Engineering and Measurement.* IEEE, 2009, pp. 401–404.

[97] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, and A. Chatzigeorgiou, “Identifying, categorizing and mitigating threats to validity in software engineering secondary studies,” *Information and Software Technology*, vol. 106, pp. 201–230, 2019.

[98] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, “A map of threats to validity of systematic literature reviews in software engineering,” in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC).* IEEE, 2016, pp. 153–160.

[99] M. Unterkalmsteiner, T. Gorschek, A. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, “Evaluation and measurement of software process improvement—a systematic literature review,” *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 398–424, 2011.

[100] N. Dissanayake, M. Zahedi, A. Jayatilaka, and M. A. Babar, “Why, how and where of delays in software security patch management: An empirical investigation in the healthcare sector,” *Proceedings of the ACM on Human-Computer Interaction (CSCW)*, vol. 6, no. 362, 2022. [Online]. Available: https://doi.org/10.1145/3555087

[101] L. H. Newman. (2017) Equifax officially has no excuse. [Online]. Available: https://www.wired.com/story/equifax-breach-no-excuse/

[102] D. Goodin. (2017) Failure to patch two-month-old bug led to massive equifax breach. [Online]. Available: https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/

[103] A. Nappa, R. Johnson, L. Bilge, J. Caballero, and T. Dumitras, "The attack of the clones: A study of the impact of shared code on vulnerability patching," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2015, pp. 692–708.

[104] J. Dunagan, R. Roussev, B. Daniels, A. Johnson, C. Verbowski, and Y.-M. Wang, "Towards a self-managing software patching process using black-box persistent-state manifests," in *International Conference on Autonomic Computing, 2004. Proceedings.* IEEE, 2004, pp. 106–113.

[105] M. Maurer and D. Brumley, "Tachyon: Tandem execution for efficient live patch testing," in *21st {USENIX} Security Symposium ({USENIX} Security 12*. Bellevue, WA: {USENIX} Association, 2012, pp. 617–630. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/maurer

[106] S. Potter and J. Nieh, "Reducing downtime due to system maintenance and upgrades," in *Proceedings of the 19th USENIX Systems Administration Conference.* IEEE, 2005, pp. 6–6.

[107] T. Dumitraş and P. Narasimhan, "Why do upgrades fail and what can we do about it?" in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, ser. Middleware 2009. Lecture Notes in Computer Science, vol. 5896. Berlin: Springer, 2009, pp. 349–372.

[108] F. Araujo and T. Taylor, "Improving cybersecurity hygiene through jit patching," in *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20).* ACM, 2020, pp. 1421–1432.

[109] C. Dietrich, K. Krombholz, K. Borgolte, and T. Fiebig, "Investigating system operators' perspective on security misconfigurations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18).* ACM, 2018, pp. 1272–1289. [Online]. Available: https://doi.org/10.1145/3243734.3243794

[110] A. Jenkins, P. Kalligeros, K. Vaniea, and M. K. Wolters, """Anyone Else Seeing this Error?": Community, system administrators, and patch information," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P).* IEEE, 2020, pp. 105–119.

[111] G. Post and A. Kagan, "Computer security and operating system updates," *Information and Software Technology*, vol. 45, no. 8, pp. 461–467, 2003.

[112] J. Tucek, W. Xiong, and Y. Zhou, "Efficient online validation with delta execution," in *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, 2009, pp. 193–204.

[113] H. Chen, T. Kim, X. Wang, N. Zeldovich, and M. F. Kaashoek, "Identifying information disclosure in web applications with retroactive auditing," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, 2014, pp. 555–569. [Online]. Available: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/chen_haogang

[114] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: Distance and speed," in *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001*. IEEE, 2001, pp. 81–90.

[115] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, dependencies, and delay in a global collaboration," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW)*. Association for Computing Machinery, 2000, p. 319–328.

[116] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481–494, 2003.

[117] T. Nguyen, T. Wolf, and D. Damian, "Global software development and delay: Does distance still matter?" in *2008 IEEE International Conference on Global Software Engineering*. IEEE, 2008, pp. 45–54.

[118] J. E. Mathieu, M. A. Marks, and S. J. Zaccaro, "Multiteam systems," *Handbook of Industrial, Work and Organizational Psychology*, vol. 2, pp. 289–313, 2001.

[119] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why developers are slacking off: Understanding how software teams use slack," in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16 Companion)*. ACM, New York, NY, USA, 2016, p. 333–336.

[120] M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in oss projects," *Proceedings of the ACM Conference on Computer Supported Cooperative Work Social Computing*, vol. 2, no. CSCW, 2018. [Online]. Available: https://doi.org/10.1145/3274451

[121] D. Wang, J. D. Weisz, M. Muller, P. Ram, W. Geyer, C. Dugan, Y. R. Tausczik, H. Samulowitz, and A. Gray, "Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, 2019.

[122] R. K. Yin, "Case study research: Design and methods," 1994.

[123] J. A. Maxwell, "Understanding and validity in qualitative research," *Harvard Educational Review*, vol. 62, no. 3, pp. 279–301, 1992. [Online]. Available: https://doi.org/10.17763/haer.62.3.8323320856251826

[124] P. Rodriguez, C. Urquhart, and E. Mendes, "A theory of value for value-based feature selection in software engineering," *IEEE Transactions on Software Engineering*, 2020.

[125] M. Procházka, D. Kouril, R. Wartel, C. Kanellopoulos, and C. Triantafyllidis, “A race for security: Identifying vulnerabilities on 50 000 hosts faster than attackers,” in *Proceedings of Science (PoS). International Symposium on Grid and Clouds*, 2011.

[126] T. Hanauer, W. Hommel, S. Metzger, and D. Pöhn, “A process framework for stakeholder-specific visualization of security metrics,” in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3230833.3232855

[127] M. Cataldo and J. D. Herbsleb, “Coordination breakdowns and their impact on development productivity and software failures,” *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 343–360, 2012.

[128] D. E. Strode, S. L. Huff, B. Hope, and S. Link, “Coordination in co-located agile software development projects,” *Journal of Systems and Software*, vol. 85, no. 6, pp. 1222–1238, 2012. [Online]. Available: https://doi.org/10.1016/j.jss.2012.02.017

[129] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, and A. Heinzl, “Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings,” *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 932–950, 2017.

[130] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley, “Identification of coordination requirements: Implications for the design of collaboration and awareness tools,” in *Proceedings of the 2006 20th anniversary Conference on Computer Supported Cooperative Work (CSCW)*. ACM, 2006, pp. 353–362.

[131] R. E. Kraut and L. A. Streeter, “Coordination in software development,” *Communications of the ACM*, vol. 38, no. 3, pp. 69–82, 1995.

[132] A. H. V. de Ven, A. L. Delbecq, and R. K. Jr, “Determinants of coordination modes within organizations,” *American Sociological Review*, pp. 322–338, 1976.

[133] K. Crowston, “A taxonomy of organizational dependencies and coordination mechanisms,” *Organizing Business Knowledge: The MIT Process Handbook*, p. 85–108, 2003.

[134] T. W. Malone and K. Crowston, “Toward an interdisciplinary theory of coordination,” *Organizing Business Knowledge: The MIT Process Handbook*, 1991.

[135] T. W. Malone and K. Crowston, “What is coordination theory and how can it help design cooperative work systems?” in *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work (CSCW)*. ACM, 1990, pp. 357–370. [Online]. Available: https://doi.org/10.1145/99332.99367

[136] R. Duan, A. Bijlani, Y. Ji, O. Alrawi, Y. Xiong, M. Ike, B. Saltaformaggio, and W. Lee, “Automating patching of vulnerable open-source software versions in application binaries,” in *NDSS*, 2019. [Online]. Available: https://dx.doi.org/10.14722/ndss.2019.23126

[137] F. Li and V. Paxson, “A large-scale empirical study of security patches,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 2201–2215. [Online]. Available: https://doi.org/10.1145/3133956.3134072

[138] P. Ralph, "Toward methodological guidelines for process theories and taxonomies in software engineering," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 712–735, 2019.

[139] J. E. Hannay, D. I. Sjøberg, and T. Dybå, "A systematic review of theory use in software engineering experiments," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 87–107, 2007.

[140] Z. Cheng, A. Dimoka, and P. A. Pavlou, "Context may be king, but generalizability is the emperor!" *Journal of Information Technology*, vol. 31, no. 3, pp. 257–264, 2016. [Online]. Available: https://doi.org/10.1057/s41265-016-0005-7

[141] B. Yang, N. Ayachitula, S. Zeng, and R. Puri, "Sla-driven applicability analysis for patch management," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops.* IEEE, 2011, pp. 438–445.

[142] J. Silva, I. Wiese, D. M. German, C. Treude, M. A. Gerosa, and I. Steinmacher, "A theory of the engagement in open source projects via summer of code programs," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20).* ACM, 2020, pp. 421–431.

[143] A. Langley, "Strategies for theorizing from process data," *Academy of Management review*, vol. 24, no. 4, pp. 691–710, 1999.

[144] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18).* ACM, 2018, pp. 187–200.

[145] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "An empirical study of automation in software security patch management," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22).* ACM, New York, NY, USA, 2022, pp. 1–13. [Online]. Available: https://doi.org/10.1145/3551349.3556969

[146] G. Maayan. (2022) Five years later, heartbleed vulnerability still unpatched. [Online]. Available: https://blog.malwarebytes.com/exploits-and-vulnerabilities/2019/09/everything-you-need-to-know-about-the-heartbleed-vulnerability/

[147] S. Wang, M. Wen, B. Lin, H. Wu, Y. Qin, D. Zou, X. Mao, and H. Jin, "Automated patch correctness assessment: How far are we?" in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20).* ACM, New York, NY, USA, 2020, p. 968–980.

[148] U. C. Türker, R. Hierons, M. R. Mousavi, and I. Tyukin, "Efficient state synchronisation in model-based testing through reinforcement learning," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE).* IEEE, 2021, pp. 368–380.

[149] A. Al-Ayed, S. Furnell, D. Zhao, and P. Dowland, "An automated framework for managing security vulnerabilities," *Information management*

& computer security, vol. 13, no. 2, pp. 156–166, 2005. [Online]. Available: https://doi.org/10.1108/09685220510589334

[150] M. S. Rahman, G. Yan, H. V. Madhyastha, M. Faloutsos, S. Eidenbenz, and M. Fisk, "idispatcher: A unified platform for secure planet-scale information dissemination," *Peer-to-Peer Networking and Applications*, vol. 6, no. 1, pp. 46–60, 2013.

[151] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 105–114.

[152] M. Angelini, S. Bonomi, S. Lenti, G. Santucci, and S. Taggi, "Mad: A visual analytics solution for multi-step cyber attacks detection," *Journal of Computer Languages*, vol. 52, pp. 10–24, 2019.

[153] P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, and A. Singhal, "Vulcan: Vulnerability assessment framework for cloud computing," in *2013 IEEE 7th International Conference on Software Security and Reliability*. IEEE, 2013, pp. 218–226.

[154] S. Kashyap, C. Min, B. Lee, T. Kim, and P. Emelyanov, "Instant {OS} updates via userspace {Checkpoint-and-Restart}," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, 2016, pp. 605–619.

[155] R. Chandra, T. Kim, M. Shah, N. Narula, and N. Zeldovich, "Intrusion recovery for database-backed web applications," in *Proceedings of the 26th Annual Computer Security Applications Conference*, 2011, pp. 101–114.

[156] A. Security. (2021) State of cybersecurity resilience 2021. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-165/Accenture-State-Of-Cybersecurity-2021.pdf

[157] S. Midtrapanon and G. B. Wills, "Linux patch management: With security assessment features," in *International Conference on Internet of Things, Big Data and Security*, 2019, pp. 270–277.

[158] S. Trabelsi, H. Plate, A. Abida, M. M. B. Aoun, A. Zouaoui, C. Missaoui, S. Gharbi, and A. Ayari, "Mining social networks for software vulnerabilities monitoring," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2015, pp. 1–7.

[159] M. Angelini, G. Blasilli, T. Catarci, S. Lenti, and G. Santucci, "Vulnus: Visual vulnerability analysis for network security," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 183–192, 2018.

[160] Y. Kim and Y. Won, "A new cost-saving and efficient method for patch management using blockchain," *The Journal of Supercomputing*, vol. 76, no. 7, pp. 5301–5319, 2020.

[161] K. J. Kim and M. Kim, "A study of integrity on the security patches system using pm-fts," *Wireless Personal Communications*, vol. 94, no. 2, pp. 165–173, 2017.

[162] C.-W. Chang, D.-R. Tsai, and J.-M. Tsai, “A cross-site patch management model and architecture design for large scale heterogeneous environment,” in *Proceedings 39th Annual 2005 International Carnahan Conference on Security Technology*. IEEE, 2005, pp. 41–46.

[163] W. Zhou, P. Ning, X. Zhang, G. Ammons, R. Wang, and V. Bala, “Always up-to-date: scalable offline patching of vm images in a compute cloud,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, 2010, pp. 377–386.

[164] Tenable, “tenable.sc,” 2023. [Online]. Available: https://www.tenable.com/products/tenable-sc

[165] Tenable, “Nessus,” 2023. [Online]. Available: https://www.tenable.com/products/nessus

[166] Apache, “Apache log4j,” 2023. [Online]. Available: https://logging.apache.org/log4j/2.x/security.html

[167] R. S. LLC, “Archer grc solution,” 2023. [Online]. Available: https://www.archerirm.com/content/grc

[168] “Ivanti,” 2023. [Online]. Available: https://www.ivanti.com/

[169] Microsoft, “Microsoft system center configuration manager,” 2023. [Online]. Available: https://docs.microsoft.com/en-us/mem/configmgr/

[170] Microsoft, “Windows server update services,” 2023. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus

[171] VMware, “Vmware workspace one,” 2023. [Online]. Available: https://www.vmware.com/products/workspace-one.html

[172] Marval, “Marval itsm,” 2023. [Online]. Available: https://marvalglobal.com/

[173] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, “Guidelines for human-ai interaction,” in *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019)*. ACM, New York, NY, USA, 2019, pp. 1–13. [Online]. Available: https://doi.org/10.1145/3290605.3300233

[174] J. A. Crowder, J. Carbone, and S. Friess, “Human–ai collaboration,” in *Artificial Psychology*. Springer, Cham, 2020.

[175] OpenAI, “Gpt-4 technical report,” *arXiv:2303.08774*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.08774

[176] V. Lai, S. Carton, and C. Tan, “Harnessing explanations to bridge ai and humans,” *arXiv preprint arXiv:2003.07370*, 2020. [Online]. Available: https://arxiv.org/pdf/2003.07370