

Why, How and Where of Delays in Software Security Patch Management: An Empirical Investigation in the Healthcare Sector

NESARA DISSANAYAKE, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia

MANSOOREH ZAHEDI, The University of Melbourne, Australia

ASANGI JAYATILAKA, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia

MUHAMMAD ALI BABAR, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia

Numerous security attacks that resulted in devastating consequences can be traced back to a delay in applying a security patch. 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. 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 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.

CCS Concepts: • **Security and privacy** → **Software security engineering**; *Vulnerability management*; • **Software and its engineering** → **Maintaining software**.

Additional Key Words and Phrases: patch management, security updates, delays, socio-technical research

ACM Reference Format:

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. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 362 (November 2022), 29 pages. <https://doi.org/10.1145/3555087>

Authors' addresses: Nesara Dissanayake, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia, nesara.madugodasdissanayake@adelaide.edu.au; Mansooreh Zahedi, The University of Melbourne, Australia, mansooreh.zahedi@unimelb.edu.au; Asangi Jayatilaka, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia, asangi.jayatilaka@adelaide.edu.au; Muhammad Ali Babar, CREST – Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia, ali.babar@adelaide.edu.au.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

2573-0142/2022/11-ART362 \$15.00

<https://doi.org/10.1145/3555087>

1 INTRODUCTION

Cyberattacks breaching corporate networks often result in catastrophic consequences ranging from exposure of sensitive and confidential data [19, 38] and betrayal of client trust to even human death [15]. The most effective remediation of this problem is to apply security patches to the identified vulnerabilities through a process called software security patch management, referred to as security patch management hereafter, consisting of detecting, retrieving, assessing, installing, and verifying security patches [48]. Despite the gravity of the process, security patch management 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 [12, 27, 51]. 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 [47] 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 have investigated the socio-technical aspects of security patch management, particularly, the process followed [27, 51] and the role of collaboration in the process [40], 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 [4, 5, 37]. Focused on the coordination aspect, our previous study [12] presented a grounded theory of the role of coordination in security patch management based on observations of 51 patch meetings between two case organisations over nine months. The theory explains the causes that define the need for coordination in the process (i.e., the socio-technical dependencies), constraints that can hinder effective coordination, the breakdowns resulting from ineffective coordination of the causes and constraints, and the mechanisms to manage the causes, constraints and breakdowns. Although previous studies have focused on approaches to reduce delays in security patch management and the effects of ineffective coordination on timely security patch management, to date, there has been no study that comprehensively 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. Motivated by the need, we extended the longitudinal study of the two case organisations focusing on the delays in security patch management. The study findings are based on the analysis of the artefacts over four years following the Straussian Grounded Theory method for the data analysis. Our study was guided by the following key research questions (RQs):

RQ1. *Why, how, and where do delays occur in software security patch management?*

RQ2. *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 in the healthcare domain, 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 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 a comprehensive 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: (i) identifies a set of reasons for delays when applying security patches in practice; (ii) describes the most prominent reasons for delays with rationales explaining their variations; (iii) reports where a majority of delays occur in the patch management process presenting their distribution over the process phases; (iv) presents a collection of strategies employed in practice to mitigate the delays including when to apply them in the patch management process; (v) structures the understanding about delays in vulnerability patch management, drawing attention to a critical yet less explored phenomenon in the CSCW community; (vi) 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, and (vii) offers practical guidance for practitioners to identify what and where is improvement needed to mitigate patching delays and drive their decisions appropriately.

2 BACKGROUND AND MOTIVATION

Software security patch management is defined as “a multifaceted process of identifying, acquiring, testing, installing, and verifying security patches for software products and systems” [11]. A security patch is an additional piece of code developed to address security vulnerabilities identified in software [34]. 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 [29] and Spectre [26] 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 [48]. Similarly, applying security patches with minimum delays is instrumental in significantly reducing the risks of cyberattacks that exploit software vulnerabilities (see Figure 1) [48]. Despite the importance of timely patch management, it remains one of the most challenging processes facing modern organisations. To guide the process, several guidelines such as the National Institute of Standards and Technology (NIST)’s Special Publication (SP) 800-40 [35, 41, 48] have been published over the years.

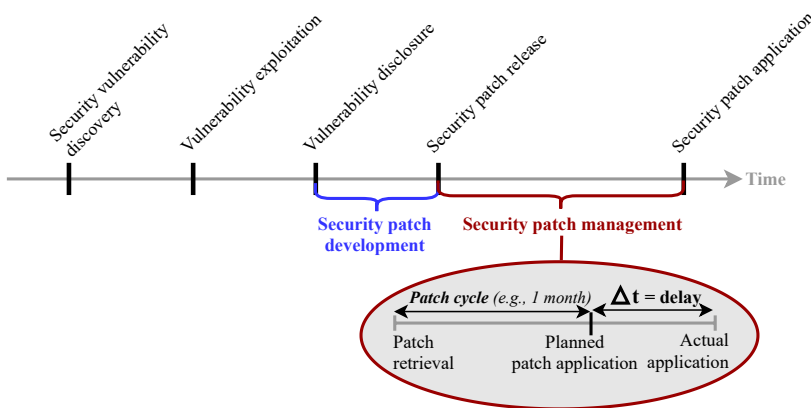


Fig. 1. The focus of the study in the vulnerability timeline.

There have been several research efforts undertaken to explore the patch management process. Two recent studies [27, 51] have investigated the stages the practitioners proceed through patch management. Figure 2 shows an overview of the five main phases in the process. These phases represent the general workflow in a patch cycle (i.e., from the patch retrieval to planned patch application) depicted in Figure 1.

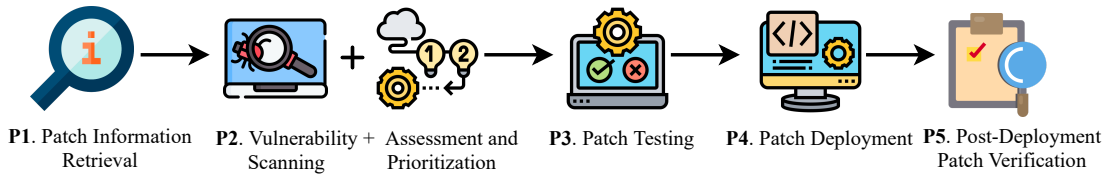


Fig. 2. The five phases of the security patch management process.

The patch information retrieval (P1) phase refers to learning about new patches and downloading them from vendor websites. Next, practitioners scan systems to identify the existing vulnerabilities, assess them based on their applicability to managed systems, and prioritise based on vulnerability severity and patch type when deciding to patch (P2). Once the decision to patch is finalised, they prepare to deploy the patches whereby testing the patches for accuracy (P3) and preparing machines by changing configurations, followed by the patch deployment (P4) phase in which the patches are installed and rebooted. As the final step, the patch deployment is verified and post-deployment issues are handled, if any (P5).

In addition, 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 [8, 14, 32] and mechanisms for reducing system downtime in reboots [1, 13, 43], 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. Cramer et al. [8] were among the first to investigate system administrators' patch management practices. From 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. [10] 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 Cramer et al., two recent studies [27, 51] 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. [24] 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 [11, 23, 27, 37, 43, 51] has explored the challenges in the patch management process. For example, the impact of organisational policies and culture [27, 40, 51], collaboration and coordination challenges due to conflicts between stakeholders [23, 27, 37, 43], lack of resources in terms of skills and expertise required for handling complex patching tasks [24, 42, 51], and the increasing rate of patch release [42, 43, 51] are some of the most common challenges faced by practitioners. In addition, challenges relating to the lack of dedicated patch testing environments [27, 32, 52], post-deployment patch verification [7, 27], and system downtime during patch deployment [13, 27, 43, 51] 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 [11, 14, 27, 42, 51].

To address some of these pressing socio-technical challenges, several studies have proposed tools, frameworks and practices. For example, a set of studies [4, 5, 9] 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. Concerning the coordination challenges, Dissanayake et. al [12] have proposed a grounded theory of the role of coordination in security patch management explaining the causes that create the need for coordinating in the security patch management process, constraints for effective coordination, breakdowns resulting from ineffective handling of the coordination causes and constraints, and mechanisms for managing the causes while mediating the constraints. 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 [19]?

In contrast, delays have been widely studied in related fields like software development. In the majority of these studies [20–22, 39], 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 [16]. 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 take appropriate actions, which is accomplished by this study.

3 RESEARCH METHOD

To understand why and how delays occur in practice, we conducted a longitudinal study with two organisations (Org A and B) involving 21 participants from 8 teams in Australia. In selecting the case organisations, we used a combination of purposive [46] and convenience [30] sampling to ensure that our data are representative of the substantive area through which the findings emerge. The demographics of the studied teams are illustrated in Table 1. Org A is an Australian state government health services agency that outsourced its **OS security patching** to Org B, an American multinational corporation. Org B was responsible for patching Org A's 1500 servers

representing the entire health sector in the state government. In addition, the non-security patches were handled by different teams in Org A, which is not included in our analysis. Org A consisted of several teams each managing different modules, for example, the teams T1-3 represented the main modules in Org A while the 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 two organisations, attended by key stakeholders representing each team detailed in Table 1. Abiding by the human ethics guidelines, the details of the companies, teams and participants have been kept confidential. Figure 3 shows the organisational setup in the studied cases.

Table 1. Demographics of participants

Organisation	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

* The team size refers to the number of team participants in the patch meeting.

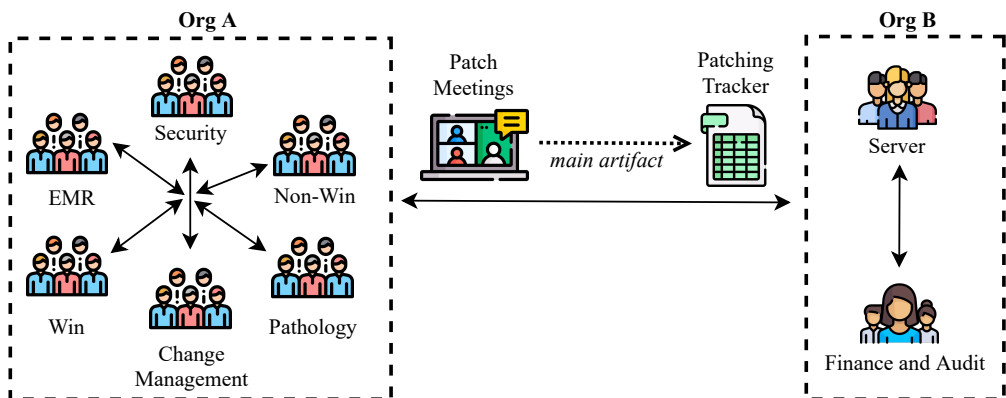


Fig. 3. The organisational setup present in the studied context.

3.1 Data Collection

We collected longitudinal data 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.

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 a 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 4. Each tracker was updated regularly with the date 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 upgrade 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 deploying 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

Fig. 4. A screenshot of an extract from the Patching Tracker - 19.05.2021.

Additionally, to understand what occurs in practice 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 meetings provided a collaborative platform for the participants to discuss and refine the patching process, plan monthly patch schedules, assess the progress, resolve problems, and make decisions about patch exemptions. The fortnightly meetings were held online through Microsoft Teams due to COVID-19 and lasted approximately an hour and a half. Additionally, the observations increased analytical validity and ensured triangulation in our findings [57].

3.2 Data Analysis

First, we qualitatively analysed the data employing the Grounded Theory’s (GT) [18, 49] data analysis procedures, particularly Strauss and Corbin’s version of GT procedures (*Straussian GT*) [49], as they offer well-structured and rigorous data analysis techniques well-suited to answer complex and practice-based *why* and *how* type questions [49]. Second, to identify how the delays and the causes are distributed, we quantitatively analysed the data using frequency analysis, a widely used technique for producing descriptive statistics derived from the data.

We analysed the data at the task level as previous studies [3, 16] 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 4 shows a screenshot of an extract from the patching tracker. To define a delay according to

the studied context, the first author held a discussion with Org A’s Security Advisor about their policies to understand the defined time frames for any given task during the monthly patch cycle practised. Table 2 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 [27, 51].

Table 2. Definition of standard time frames in the studied organisation

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 [49], as shown in Figure 5. The first author performed the data analysis while the second and third authors cross-checked all the codes throughout the process to increase the reliability of the findings and reduce bias [50]. Any disagreements in the coding were resolved through weekly discussions among all authors involving multiple rounds of revisions. The patching tracker, codes, and memos were stored in NVivo, the data analysis tool, and shared with all authors.

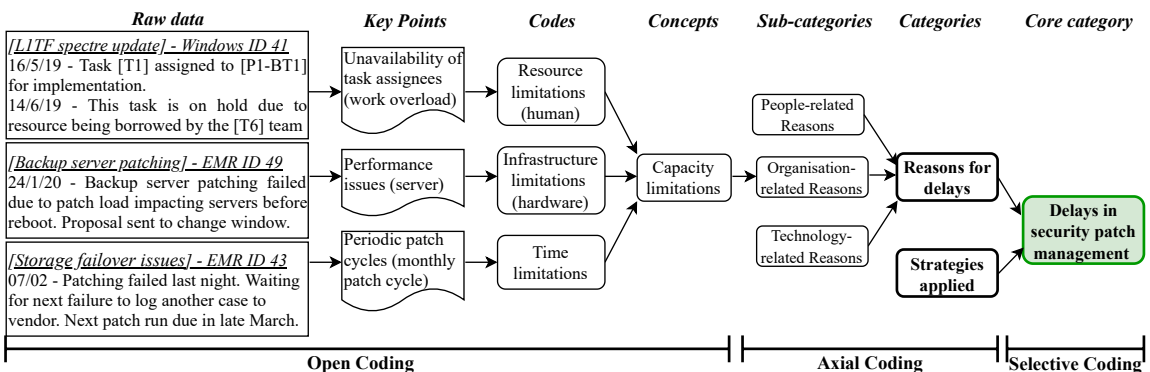


Fig. 5. Emergence of the category *Reasons for delays* from the underlying concept of *Capacity limitations* and codes.

We started with **open coding**, whereby we analysed all columns row by row in the spreadsheet to identify *key points* summarising the content. It was further summarised 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* [49], 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 linked categories to their subcategories based on the relationships between categories relating to their *properties* (i.e., "characteristics of a category") and *dimensions* (i.e., "variations within properties") [49]. During the analysis, we created *memos* for explaining the codes and their relationships which helped us during this process. During the final phase of the analysis, we applied **selective coding** by which we identified 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*" [49], in this case, **delays in security patch management**.

For confidentiality reasons, we do not share the raw data. However, we made our codebook containing the codes, descriptions and examples of raw data publicly available ¹.

3.3 Member checking

We conducted a member checking [36] session to ensure the credibility, accuracy, validity, and transferability of our study findings. Member checking, a technique of "*taking ideas back to research participants for the confirmation*" [6], provides an opportunity to validate the findings with participants and resonance with their experiences [2]. We presented the study findings at a session held at Org A. Three authors 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 A were present physically. In addition, seven patch meeting participants (four from Org A and three from Org B) attended the session virtually. The first 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 RQ 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 first author. The feedback and comments from member checking are presented in Section 4.5.

4 FINDINGS

In this section, we present the findings of our study. Figure 6 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 [53]. In the examples, we include the subject of the task (see Figure 4) 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.

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 7. In summary, we found nine reasons, grouped into three main categories: technology-related reasons, people-related reasons and organisation-related reasons. Next, we quantitatively analysed the identified set of reasons to understand the *most prominent reasons* that

¹<https://doi.org/10.5281/zenodo.5635608>

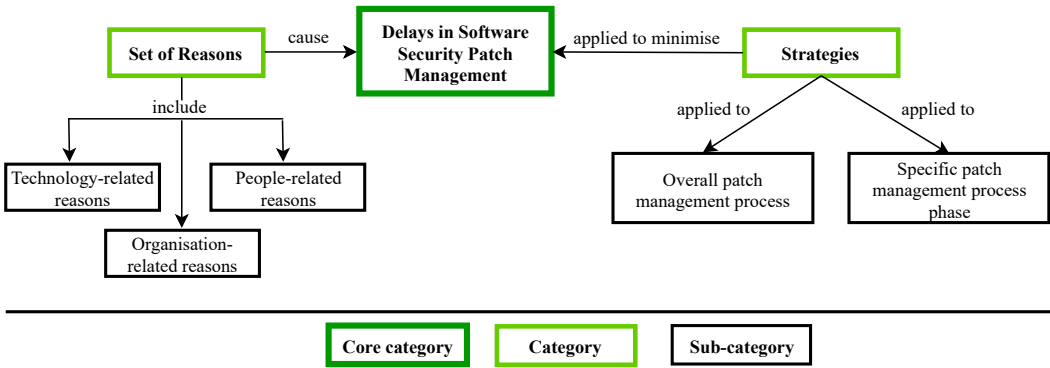


Fig. 6. High-level overview of the findings from the qualitative data analysis.

need practitioners' and researchers' attention. To achieve this, we conducted a frequency analysis on the reasons for delays. An important observation was that in the majority of the delays, we found multiple reasons attributing 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 8 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 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 9. 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 9, 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.

4.1.1 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 of 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 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 [12]. Patching large and complex software

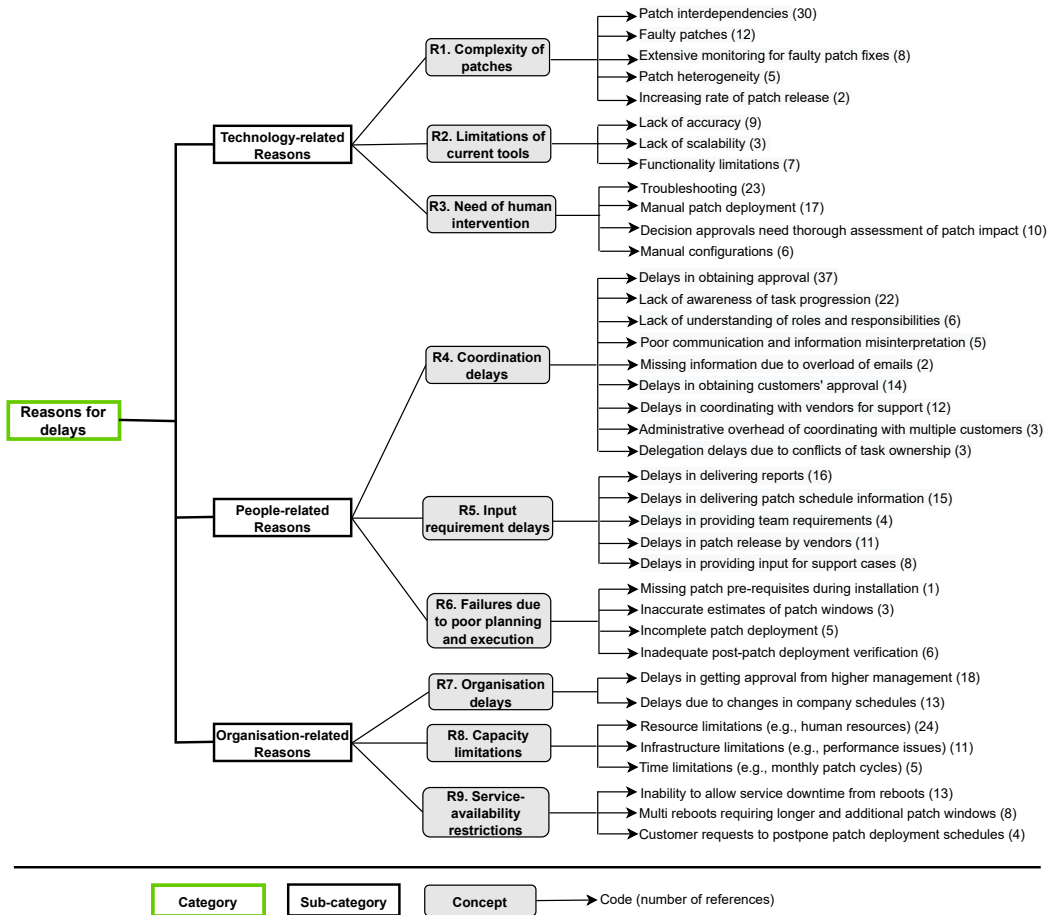


Fig. 7. Detailed overview of the causes of delays in software security patch management.

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., obtain 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

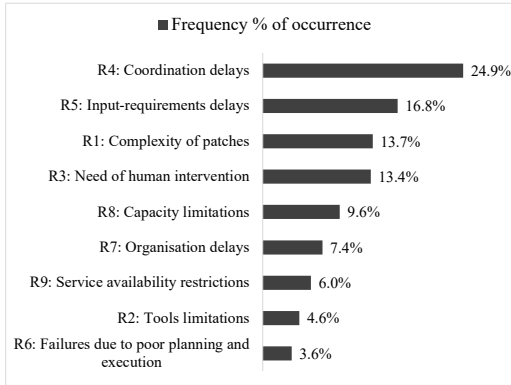


Fig. 8. Frequency distribution of the reasons for delays in security patch management from a total of 417 occurrences of delayed reasons.

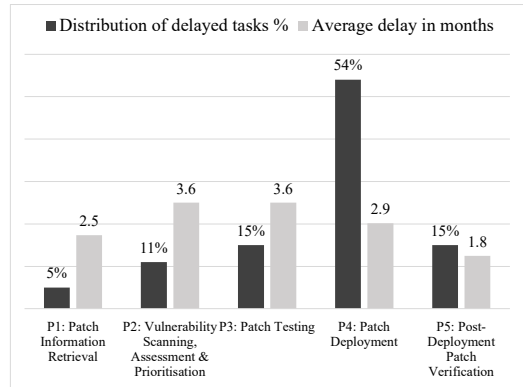


Fig. 9. Distribution of delays over the security patch management process and average delay duration in months in each phase. Total number of delayed tasks = 132.

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 to 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 the 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 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

4.1.2 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 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 [31] 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 the 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 lack of accountability.

Another instrumental people-related reason was the *delays in providing input requirements*. This is because the patch management process represents a sequence of phases with tightly coupled activities whereby an 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's 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 expected as a zero-day soon, full report 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 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.

4.1.3 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 time leading to delays. With regards 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 it 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 RQ1: 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 and input requirement delays appeared as the most prominent and recurrent reasons. 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.

4.2 Mitigation strategies for delays in software security patch management

We identified a group of strategies implemented by the studied teams as corrective/reactive actions to reduce the delays. Further investigation enabled us to identify *where* to apply the strategies in the patch management process. Figure 10 presents the strategies grouped by the relevant patch management process phase with the number of references for each strategy (in parentheses).

4.2.1 Strategies relating to the overall patch management process. The following set of *common strategies* can be applied across all phases of the patch management process.

Frequent communication with all internal and external stakeholders is vital in reducing the patching delays as it helped strengthen the 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'

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)	Strategies relating to Vulnerability Scanning, Assessment & Prioritisation (P2)	Strategies relating to Patch Testing (P3)	Strategies relating to Patch Deployment (P4)	Strategies relating to Post-Deployment Patch Verification (P5)
S5. Set strict timelines for patch download (2)	S6. Plan alternatives for delayed patches (6) S7. Define priorities for vulnerability remediation (15)	S8. Define compliance policies and contingency plans for test failures (9) S9. Patch pre-requisites investigation (4) S10. Modify software configurations and dependencies (3)	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)	S15. Establish post-deployment verification procedures (10) S16. Collectively handle post-deployment issues (9) S17. Document deployment status of every patch (3)

Fig. 10. Detailed overview of the strategies applied to mitigate delays in software security patch management.

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, 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 that 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 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 like task B is dependent

on task A (A → 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 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

4.2.2 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 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

4.2.3 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.

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

4.2.4 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.

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.

4.2.5 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*).

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 pre-loading 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 and 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 4.1.1.

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.

4.2.6 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 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 to failed deployments. 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.

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.

Summary for RQ2: We identified 17 strategies applied by the practitioners as corrective/reactive actions to manage 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.

4.3 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 and 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 5.

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 [35], and failure to do so has resulted in several devastating outcomes [19]. 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 8). The need for effective coordination in patch management appears from a combination of complex technical dependencies inherent in patches

and the collaborative environment in the patch management process demanding orchestrated social dependencies between a diverse group of stakeholders [12]. 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 emphasize 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) [11, 12, 27] and defining contingency plans to handle faulty patch errors (S8) can help reduce delays arising from the patch complexity. The socio-technical endeavour in 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 the human interaction in 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 [12]. 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 [11, 40]. 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 [1, 13, 27, 43, 51].

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.

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 [23, 27, 37, 40, 43], 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 [11, 12, 27, 40, 51].

Our analysis reveals that the complexity of patches causes the third-most frequent reason for delays; it complements the previous work [8, 14, 24, 37, 40, 51, 52], 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 [8, 12, 27, 51] and capacity limitations, specifically, lack of human resources [24, 42, 51] are mentioned as challenges in security patch management in the related studies. Further, several studies (e.g., [1, 13, 27, 43, 51]) 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 [4, 5] 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. [9] 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. [37] 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 patch management process,
- presenting a set of strategies to mitigate the delays and describing when they can be applied in the patching process,
- providing practical implications for practitioners to identify and mitigate delays, and,
- establishing a foundation for future research towards effective management of patching delays.

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 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 patch management tasks with minimal impediments.

In addition to explaining why, how, and where do 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 the 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 accomplishing timely communication, collaboration, and information sharing between all stakeholders [28]. In this way, our findings offer guidance to practitioners to make suitable decisions to alleviate the threat of cyberattacks from delayed patching.

5.3 Implications for Researchers

Given our findings are based on the cases studied limiting 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 [12, 27, 51]). 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.

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. 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 developing computer-supported tools and platforms that can support better coordination across patching tasks and reducing delays in collaborative tasks. 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 [55] 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*” [25], an emerging research paradigm in the CSCW community [54], 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). In addition, the researchers, particularly the usable security researchers, can study how to improve the design of such smart tools.

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, 45, 56].

External validity - Generalizability: 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 generalization 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 [17, 18, 49].

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 [33]. 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.

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 authors and finalised through multiple revisions. In addition, related to interpretive validity [33], we conducted member checking to verify the accuracy of our findings, which was attended by three authors, further ensuring investigator triangulation.

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, maintained a chain of evidence (e.g., the coding procedure following the Grounded Theory method [44]), and had the findings reviewed through member checking.

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.

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 [18, 49]. To achieve this, we have detailed our data analysis process of the application of the Straussian GT procedures in Section 3. Further, to alleviate the reporting bias, we have included quotes from the patching tracker in Section 4.

7 CONCLUSION

In this study, 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 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 evidence-based understanding of the frequency distribution of reasons for delays and distribution of delays over the patch management process. Such information highlights the reasons that need immediate attention and the areas of improvement in the 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 a holistic understanding of the delays when applying security patches in practice; 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 patch management, drawing attention to a topic, critical and timely, yet less explored in the CSCW community.

ACKNOWLEDGMENTS

The authors sincerely thank the industry collaborators of CREST, without whose support this research would not have been possible. Thank you for allowing us to participate in your process of security patch management, and for providing us with great feedback on our presentation. We also thank the reviewers for their valuable insights and feedback.

REFERENCES

- [1] Frederico Araujo and Teryl Taylor. 2020. 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, 1421–1432. <https://doi.org/10.1145/3368089.3417056>
- [2] Linda Birt, Suzanne Scott, Debbie Cavers, and Christine Campbell Fiona Walter. 2016. Member Checking: A Tool to Enhance Trustworthiness or Merely a Nod to Validation? *Qualitative Health Research* 26, 13 (2016), 1802–1811. <https://doi.org/10.1177/1049732316654870>
- [3] Frederick P. Brooks. 1975. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, London.
- [4] Huseyin Cavusoglu, Hasan Cavusoglu, and Jun Zhang. 2006. Economics of Security Patch Management. In *WEIS*. Citeseer, 1–10.
- [5] Hasan Cavusoglu, Huseyin Cavusoglu, and Jun Zhang. 2008. Security Patch Management: Share the Burden or Share the Damage? *Management Science* 54, 4 (2008), 657–670. <https://doi.org/10.1287/mnsc.1070.0794>
- [6] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage.

- [7] Haogang Chen, Taesoo Kim, Xi Wang, Nickolai Zeldovich, and M. Frans Kaashoek. 2014. Identifying Information Disclosure in Web Applications with Retroactive Auditing. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, 555–569. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/chen_haogang
- [8] Olivier Crameri, Nikola Knezevic, Dejan Kostic, Ricardo Bianchini, and Willy Zwaenepoel. 2007. Staged Deployment in Mirage, an Integrated Software Upgrade Testing and Distribution System. *ACM SIGOPS Operating Systems Review* 41, 6 (2007), 221–236. <https://doi.org/10.1145/1323293.1294283>
- [9] Debabrata Dey, Atanu Lahiri, and Guoying Zhang. 2015. Optimal Policies for Security Patch Management. *INFORMS Journal on Computing* 27, 3 (2015), 462–477. <https://doi.org/10.1287/ijoc.2014.0638>
- [10] Constanze Dietrich, Katharina Krombholz, Kevin Borgolte, and Tobias Fiebig. 2018. Investigating System Operators’ Perspective on Security Misconfigurations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS’18)*. ACM, 1272–1289. <https://doi.org/10.1145/3243734.3243794>
- [11] Nesara Dissanayake, Asangi Jayatilaka, Mansooreh Zahedi, and Muhammad Ali Babar. 2021. Software security patch management-A systematic literature review of challenges, approaches, tools and practices. *Information and Software Technology* 144 (2021), 106771. <https://doi.org/10.1016/j.infsof.2021.106771>
- [12] 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)*. ACM, New York, NY, USA. <https://doi.org/10.1145/3468264.3468595>
- [13] Tudor Dumitraş and Priya Narasimhan. 2009. 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 (Middleware 2009. Lecture Notes in Computer Science, Vol. 5896)*. Springer, Berlin, 349–372. https://doi.org/10.1007/978-3-642-10445-9_18
- [14] John Dunagan, Roussi Roussev, Brad Daniels, Aaron Johnson, Chad Verbowski, and Yi-Min Wang. 2004. Towards a self-managing software patching process using black-box persistent-state manifests. In *International Conference on Autonomic Computing, 2004. Proceedings*. IEEE, 106–113. <https://doi.org/10.1109/ICAC.2004.1301353>
- [15] Melissa Eddy and Nicole Perloth. 2020. *Cyber Attack Suspected in German Woman’s Death*. Retrieved June 23, 2021 from <https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomware-death.html?smid=tw-share>
- [16] Michiel Van Genuchten. 1991. Why is Software Late? An Empirical Study of Reasons for Delay in Software Development. *IEEE Transactions on Software Engineering* 17, 6 (1991), 582–590.
- [17] Barney G. Glaser. 1978. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Sociology Press, Mill Valley, CA.
- [18] Barney G. Glaser and Anselmo L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, Chicago.
- [19] Dan Goodin. 2017. *Failure to patch two-month-old bug led to massive Equifax breach*. Retrieved June 23, 2021 from <https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/>
- [20] James D. Herbsleb and Audris Mockus. 2003. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering* 29, 6 (2003), 481–494. <https://doi.org/10.1109/TSE.2003.1205177>
- [21] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. 2000. 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, 319–328. <https://doi.org/10.1145/358916.359003>
- [22] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. 2001. An Empirical Study of Global Software Development: Distance and Speed. In *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001*. IEEE, 81–90. <https://doi.org/10.1109/ICSE.2001.919083>
- [23] Hai Huang, Salman Baset, Chunqiang Tang, Ashu Gupta, KN Madhu Sudhan, Fazal Feroze, Rajesh Garg, and Sumithra Ravichandran. 2012. Patch Management Automation for Enterprise Cloud. In *IEEE Network Operations and Management Symposium*. IEEE, 691–705. <https://doi.org/10.1109/NOMS.2012.6211988>
- [24] Adam Jenkins, Pieris Kalligeros, Kami Vaniea, and Maria K. Wolters. 2020. “Anyone Else Seeing this Error?”: Community, System Administrators, and Patch Information. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 105–119. <https://doi.org/10.1109/EuroSP48549.2020.00015>
- [25] Ece Kamar. 2016. Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence. In *IJCAI*. IEEE, 4070–4073.
- [26] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative

- Execution. In *2019 IEEE Symposium on Security and Privacy (S&P'19)*. IEEE, 1–19. <https://doi.org/10.1109/SP.2019.00002>
- [27] Frank Li, Lisa Rogers, Arunesh Mathur, Nathan Malkin, and Marshini Chetty. 2019. Keepers of the Machines: Examining How System Administrators Manage Software Updates. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. USENIX Association, 273–288.
- [28] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. 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, 333–336. <https://doi.org/10.1145/2818052.2869117>
- [29] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, 973–990. <https://www.usenix.org/conference/usenixsecurity18/presentation/lipp>
- [30] Martin N. Marshall. 1996. Sampling for qualitative research. *Family practice* 13, 6 (1996), 522–526. <https://doi.org/10.1093/fampra/13.6.522>
- [31] John E. Mathieu, Michelle A. Marks, and Stephen J. Zaccaro. 2001. Multiteam systems. *Handbook of Industrial, Work and Organizational Psychology* 2 (2001), 289–313.
- [32] Matthew Maurer and David Brumley. 2012. TACHYON: Tandem execution for efficient live patch testing. In *21st {USENIX} Security Symposium ({USENIX} Security 12)*. {USENIX} Association, Bellevue, WA, 617–630. <https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/maurer>
- [33] Joseph A. Maxwell. 1992. Understanding and Validity in Qualitative Research. *Harvard Educational Review* 62, 3 (1992), 279–301. <https://doi.org/10.17763/haer.62.3.8323320856251826>
- [34] Peter Mell, Tiffany Bergeron, and David Henning. 2005. Creating a patch and vulnerability management program. *NIST Special Publication* 800 (2005), 40.
- [35] Peter Mell, Tiffany Bergeron, and David Henning. 2005. Creating a Patch and Vulnerability Management Program. *NIST Special Publication (SP) 800-40 Revision 2* (2005).
- [36] Sharan B. Merriam. 1998. *Qualitative Research and Case Study Applications in Education. Revised and Expanded from ERIC*.
- [37] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. 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, 692–708. <https://doi.org/10.1109/SP.2015.48>
- [38] Lily Hay Newman. 2017. *Equifax Officially Has No Excuse*. Retrieved June 23, 2021 from <https://www.wired.com/story/equifax-breach-no-excuse/>
- [39] Thanh Nguyen, Timo Wolf, and Daniela Damian. 2008. Global Software Development and Delay: Does Distance Still Matter?. In *2008 IEEE International Conference on Global Software Engineering*. IEEE, 45–54. <https://doi.org/10.1109/ICGSE.2008.39>
- [40] Felicia M. Nicasastro. 2003. Security Patch Management. *Inf. Secur. J. A Glob. Perspect.* 12, 5 (2003), 5–18. <https://doi.org/10.1201/1086/43808.12.5.20031101/78486.2>
- [41] NIST. 2002. Procedures for Handling Security Patches. *Special Publication (SP) 800-40* (2002).
- [42] Gerald Post and Albert Kagan. 2003. Computer security and operating system updates. *Information and Software Technology* 45, 8 (2003), 461–467.
- [43] Shaya Potter and Jason Nieh. 2005. Reducing Downtime Due to System Maintenance and Upgrades. In *Proceedings of the 19th USENIX Systems Administration Conference*. IEEE, 6–6.
- [44] Pilar Rodriguez, Cathy Urquhart, and Emilia Mendes. 2020. A Theory of Value for Value-based Feature Selection in Software Engineering. *IEEE Transactions on Software Engineering* (2020). <https://doi.org/10.1109/TSE.2020.2989666>
- [45] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164.
- [46] Thomas A. Schwandt. 1997. *Qualitative Inquiry*. Sage, London.
- [47] Accenture Security. 2020. *2020 Cyber Threatscape Report*. Retrieved June 23, 2021 from https://www.accenture.com/_acnmedia/PDF-136/Accenture-2020-Cyber-Threatscape-Full-Report.pdf
- [48] Murugiah Souppaya and Karen Scarfone. 2013. Guide to Enterprise Patch Management Technologies. *NIST Special Publication 800-40 Revision 3* (2013), 40. <http://dx.doi.org/10.6028/NIST.SP.800-40r3>
- [49] Anselm L. Strauss and Juliet M. Corbin. 1998. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory* (2nd ed.). Sage.
- [50] Anselm L. Strauss and Juliet M. Corbin. 2007. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3rd ed.). Sage.
- [51] Christian Tiefenau, Maximilian Häring, Katharina Krombolz, and Emanuel von Zeszschwitz. 2020. Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators. In *Sixteenth Symposium on*

Usable Privacy and Security (SOUPS 2020). USENIX Association, 239–258.

- [52] Joseph Tucek, Weiwei Xiong, and Yuanyuan Zhou. 2009. Efficient online validation with delta execution. In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*. 193–204.
- [53] Cathy Urquhart. 2013. *Grounded Theory for Qualitative Research: A Practical Guide*. Sage.
- [54] Dakuo Wang, Justin D. Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla R Tausczik, Horst Samulowitz, and Alexander Gray. 2019. Human-AI Collaboration in Data Science: Exploring Data Scientists' Perceptions of Automated AI. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019). <https://doi.org/10.1145/3359313>
- [55] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Characterizing and Understanding Bots in OSS Projects. *Proceedings of the ACM Conference on Computer Supported Cooperative Work: Social Computing* 2, CSCW (2018). <https://doi.org/10.1145/3274451>
- [56] Robert K. Yin. 1994. *Case Study Research: Design and Methods*.
- [57] Robert K. Yin. 2009. *Case Study Research: Design and Methods* (4 ed.). Sage, Thousand Oaks, CA, USA.

Received July 2021; revised November 2021; accepted February 2022