An Empirical Study of Automation in Software Security Patch Management

Nesara Dissanayake
CREST - Centre for Research on Engineering Software Technologies, The University of Adelaide
Adelaide, Australia
nesara.madugodasdissanayakege@adelaide.edu.au

Asangi Jayatilaka
CREST - Centre for Research on Engineering Software Technologies, The University of Adelaide
Adelaide, Australia
asangi.jayatilaka@adelaide.edu.au

Mansooreh Zahedi
University of Melbourne
Melbourne, Australia
mansooreh.zahedi@unimelb.edu.au

Muhammad Ali Babar
CREST - Centre for Research on Engineering Software Technologies, The University of Adelaide
Adelaide, Australia
ali.babar@adelaide.edu.au

ABSTRACT

Several studies have shown that automated support for different activities of the security patch management process has great potential for reducing delays in installing security patches. However, it is also important to understand how automation is used in practice, its limitations in meeting real-world needs and what practitioners really need, an area that has not been empirically investigated in the existing software engineering literature. This paper reports an empirical study aimed at investigating different aspects of automation for security patch management using semi-structured interviews with 17 practitioners from three different organisations in the healthcare domain. The findings are focused on the role of automation in security patch management for providing insights into the as-is state of automation in practice, the limitations of current automation, how automation support can be enhanced to effectively meet practitioners' needs, and the role of the human in an automated process. Based on the findings, we have derived a set of recommendations for directing future efforts aimed at developing automated support for security patch management.

CCS CONCEPTS

• **Security and privacy** → **Software security engineering**; **Vulnerability management**; • **Software and its engineering** → *Software post-development issues.*

KEYWORDS

security updates, patch management, vulnerability management

ACM Reference Format:
Nesara Dissanayake, Asangi Jayatilaka, Mansooreh Zahedi, and Muhammad Ali Babar. 2022. An Empirical Study of Automation in Software Security Patch Management. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3551349.3556969

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.
ASE '22, October 10–14, 2022, Rochester, MI, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3556969

1 INTRODUCTION

Timely patching of security vulnerabilities is vital for safeguarding software systems against cyberattacks [33, 40]. Failure in promptly applying security patches usually results in devastating consequences as a result of successful cyberattacks that particularly target software systems in mission-critical domains such as healthcare [19, 20]. For example, a failure in the timely installation of an available patch exploited by the WannaCry ransomware cryptoworm brought down several critical systems of the National Health Services (NHS) in the UK [47]. Several studies [26, 40] report that available but not installed security patches are one of the main reasons for successful cyberattacks, e.g., the Heartbleed security bug that remained unpatched in systems for several years [35].

In order to understand the reasons for and provide solutions to the delays in applying security patches, an increased amount of research attention is being targeted at the software security patch management process, also called security patch management [15, 33, 40]. Software security patch management consists of several activities, such as detecting, obtaining, testing, installing and verifying security patches, and associated interdependent artefacts on which different types of stakeholders work [15].

Like other areas of software engineering [60, 62], it has been claimed that automation has great potential for significantly improving the effectiveness and efficiency of security patch management to minimise the delays in applying available security patches [7]. That has led to increased enthusiasm for devising and integrating automation support in different activities of the security patch management workflow. For example, automated approaches/tools for patch information retrieval [2, 49], vulnerability scanning [5, 8], assessment and prioritisation [8, 28], patch testing [11, 18], patch deployment [7, 29] and post-deployment patch verification [9, 48]. Despite these research efforts producing promising results for streamlining different activities of security patch management by providing automated solutions, delays in real-world security patch management still exist [52]. We assert that it is important to empirically understand the current status of automation, the real-world

issues facing the current automation and the practitioners' needs for improved automation support to develop suitable automated approaches and tools for effectively addressing delays in security patch management. To the best of our knowledge, there is little empirically known about these aspects of automation for security patch management.

Motivated by the need to empirically understand the role and user needs of automation for improving security patch management, we conducted an empirical investigation using in-depth semi-structured interviews with 17 practitioners from three organisations in the healthcare sector. We followed the Straussian version of the Grounded Theory method [54] to systematically uncover the findings in a bottom-up approach. Our study findings contribute to the state-of-the-art understanding of research and practice by (i) providing an evidence-based understanding of the as-is state of automation in security patch management describing the current manual and automated tasks in the process, (ii) identifying the limitations of the current automation in practice, (iii) informing how automation support can be enhanced in effectively meeting practitioners' needs, (iv) explaining the role of the human in security patch management automation identifying the tasks and points in the process in which automation needs to facilitate a human-in-the-loop approach and (v) discussing the future avenues of research and providing a set of recommendations that can guide tool designers and researchers to address the identified gaps and user needs.

2 BACKGROUND

Software security patches are *"pieces of code developed to address security problems identified in software"* [40]. Organisations often prioritise security patches over non-security patches (e.g., feature patches) due to their criticality. Security patch management refers to the *"multifaceted process of identifying, acquiring, testing, installing, and verifying security patches for the identified third-party vulnerabilities"* [15]. Existing studies [15, 33, 58] report that the process consists of five phases as described in Table 1.

Although no work has specifically investigated the role of automation in the process in practice, the previous research has invested efforts aimed at providing automation for security patch management to bring efficiency into the patch management process and reduce delays [7, 11, 18, 48]. Most of the previous efforts focus on automating specific tasks in security patch management, rather than supporting the end-to-end process [15]. A few studies have proposed solutions for automated patch information retrieval from multiple sources [2, 43], customised information filtering [2, 59], information validation [49] and patch download and distribution [49]. For vulnerability scanning, a central platform integrating the scan results from multiple information sources has been suggested [5, 8]. Towards vulnerability risk assessment and prioritisation, researchers [4, 8, 28] have proposed customisable tools/approaches for vulnerability risk analysis in line with the industry standard, the Common Vulnerability Scoring System (CVSS) [46].

Towards patch testing, a set of tools have attempted automated detection of faulty [11, 18] and malicious patches [30, 31]. However, only limited attention has been given to recovering from crashes that result from faulty patches with minimum disruption. Several studies focus on automating patch deployment [10, 43]. To minimise downtime and service disruptions, several approaches have been developed; such as dynamic software updating (DSU) [22], JIT patching [7] and instant kernel updates [29]. For post-deployment patch verification, automated approaches for verification of patch deployment [48], detection of exploits [63], and repair of past exploits [9] have been presented. An important observation is that only a few studies have rigorously evaluated the proposed solutions in real-world contexts [15, 25]; an implication of little empirical understanding of how well the solutions have addressed the practitioners' needs.

Table 1: Phases of the security patch management process

Process Phase	Description of the Tasks Undertaken
Patch information retrieval	• Learning about new patch releases • Acquiring patches from third-party vendors and distributing to the target machines
Vulnerability scanning, assessment and prioritisation	• Scanning systems to identify existing vulnerabilities • Assessing the vulnerability risk based on organisational context • Prioritising patch decisions according to the vulnerability risk assessment
Patch testing	• Preparing for patch deployment by configuring the machines, handling patch dependencies and scheduling patch windows • Testing patches for inadvertent errors or side effects
Patch deployment	• Installing patches on the machines
Post-deployment patch verification	• Verifying the patch deployment • Handling post-deployment issues

Meanwhile, another set of studies has investigated the socio-technical aspects relating to security patch management, particularly the patch management process and its challenges [33, 58], the role of coordination and collaboration [16, 23, 44, 45], system administrators' practices, behaviour, and experiences [11, 14, 26, 33, 58], and reasons and mitigation strategies for delays in security patch management [17]. The understanding of the socio-technical aspects is considered important as security patch management is essentially a socio-technical endeavour whereby humans and technologies continuously interact for enabling team members to effectively coordinate and collaborate using the available tools [15]. An important observation is the highlighted need for human involvement in the patch management process despite the advancements in automation to reduce patch management delays [5, 11, 15, 18, 33, 38, 58]. However, why human involvement is needed in process automation and the desired balance between automation and human involvement has not been investigated.

Our work distinguishes itself from the above-mentioned works as it purports to provide a holistic understanding of the role of automation in security patch management grounded in evidence gathered from practitioners. We explain how practitioners have integrated automation into the security patch management process by

describing the automated and manual tasks in the workflow, limitations of the current automation in meeting practitioners' needs, how automation support can be improved to assist practitioners, and the roles of humans in process automation, i.e., why and where is human involvement needed. Given the criticality of timely security patch management, this study's findings will provide a solid foundation to propose practical solutions that can address the limitations of current automation support for security patch management.

3 RESEARCH METHOD

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

RQ1. What is the as-is state of automation in security patch management? This RQ focuses on understanding the current state of automation in practice by identifying which security patch management tasks are currently supported by automation.

RQ2. What are the limitations of current automation? This RQ focuses on identifying the limitations of the current automation/tools to understand how well the current automation meets practitioners' needs.

RQ3. How automation in security patch management can be enhanced to support practitioners? This RQ aims at understanding the kind of support practitioners expect from improved automation. The knowledge is important to design future tools to effectively meet practitioners' needs.

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

3.1 Data Collection

Participants recruitment: We collected data through semi-structured interviews with 17 practitioners from three organisations (Org A, B and C) in Australia. We used a mix of purposive [51] and convenience [36] sampling when selecting the organisations to ensure that our data are representative of the substantial area through which the findings emerge. Org A is a large health services agency responsible for maintaining the IT systems in an Australian state government's healthcare sector. In addition to maintaining third-party vendors' software systems, they also maintained some medical software applications developed in-house. However, security patch management of operating systems (OS) and associated third-party applications were outsourced to Org B and Org C, which are IT services and consulting organisations. The non-security patches were handled by different teams in Org A, which is not included in our scope.

We requested the manager of Org A's security team to suggest potential interviewees engaged in security patch management in the case organisations. Then, we sent email invitations to the nominated candidates. The participant selection covered a diverse set of job roles in security patch management to achieve a rounded perspective. A pre-interview questionnaire was emailed to the participants to collect demographic information about them and their team. The demographics of the participants are presented in Table 2. The participants, on average, had 22 years of experience in security patch management. The participant (PID) and organisation identities have been omitted to preserve participant anonymity as per the human ethics guidelines. The last three columns refer to the number, types of machines or devices, and types of software components managed by the participants.

Semi-structured Interviews: We conducted semi-structured interviews with 17 participants from 10 teams of three organisations from November-December 2021. The interviews lasted between 30 and 60 minutes and were conducted via Microsoft Teams because of Covid-19 restrictions. The first author led all the interviews, while the second and the third authors participated in seven interviews (author2 = 4 interviews and author3 = 3 interviews) along with the first author for helping in asking the follow-up questions and increasing the reliability of the interviewing process.

The interviews were focused on security patch management of OS and third-party software applications in the context of healthcare. The open-ended interview questions covered the following areas: (a) role and responsibilities in the security patch management; (b) the current status of the automation tools used in the process; (c) the challenges and limitations of the current automation tools; (d) role of humans in security patch management automation and the reasons for human involvement, and (e) the need for better automation support (i.e., the envisioned solutions and their features).

All the authors jointly prepared the interview guide which was modified with mutual agreement when appropriate during data collection to reduce the potential bias. All the interviews were recorded and transcribed for analysis by the first author. The interviews were conducted in two phases in parallel with the data analysis. We continued with the data collection until the data analysis confirmed *theoretical saturation* [54], for example, the last three interviews provided more examples for the emerged findings but no new categories or insights emerged.

3.2 Data Analysis

We used Grounded Theory (GT) [21, 54] for data analysis as it aligned well with our study's goals of investigating a phenomenon in a real-world context, in this case, understanding the role of automation in security patch management in practice. We adopted Strauss and Corbin's GT version (Straussian GT) [54] as it provides a well-structured data analysis approach guided by open-ended and practice-based RQs [53, 54]. The data analysis procedure followed different types of coding: ***open coding***, ***axial coding*** and ***selective coding*** [54]. The first author conducted the data analysis and created a Codebook containing all the codes and memos. The Codebook and interview transcripts were shared with all co-authors. The second and third authors cross-validated all the codes, concepts, categories, and core categories in the Codebook against the raw data (i.e., interview transcripts) to reduce bias and increase the reliability of the findings [55]. The categories and their relationships were thoroughly discussed in weekly meetings and finalised after

Table 2: Participant demographics

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

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

several rounds of revisions among all authors. Any disagreements were resolved in weekly detailed discussions between all authors throughout the process. The use of analytical tools such as diagramming and memoing facilitated the data analysis process. The interview transcripts were stored in NVivo, a qualitative analysis tool [24].

During ***open coding***, we read through the interview transcripts line by line summarising the content as *key points*. It was further encapsulated into *codes* with short phrases. Constantly comparing the emerged codes within a single interview transcript and between different interview transcripts, gave rise to *concepts* [54], a higher level of abstraction of the *codes*. Similarly, we developed *categories*, the next level of abstraction, as shown in Figure 1. Next, we applied ***axial coding*** in which we systematically linked the emerged categories at the level of their properties (i.e., "characteristics of a category") and dimensions (i.e., "variations within properties") to uncover the relationships among them [54]. This process of axial coding was guided through activities involving referring back and forth to *memos*, i.e., notes written during open coding explaining the codes and their relationships, drawing the inter-relationships in diagrams, and refining them through frequent team meetings. The process continued until no new properties, dimensions, or relationships emerged for each category, which indicated *theoretical saturation* [54]. Finally, we applied ***selective coding*** where we finalised the categories and integrated them with the main theme of the research, known as the *central category* [54], in this case, the role of automation in security patch management.

The raw data obtained in this study cannot be shared because of confidentiality agreements with the industry collaborators, which happen to be in the health sector where data sensitivity has extra layers of governance. However, we made our interview guide containing the pre-interview questionnaire and interview questions publicly available [1].

4 RESULTS

The sub-sections of this section report our study's findings for different aspects of automation in security patch management.

4.1 As-Is State of Automation in Security Patch Management

Our findings to answer RQ1 reveal that whilst some tasks of security patch management have full or partial automation support, several of the tasks are carried out entirely manually. Figure 2 summarises the current state of the automation reported by the participants characterised by each process phase shown in Table 1. We describe how practitioners have integrated automation into the main tasks and the tools used below.

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

"The [vendor] schedules a call monthly for his premium customers. Then there's a security professionals' community where we share intelligence on incidents happening. For example, mid last year, a

[1] https://doi.org/10.5281/zenodo.6519830

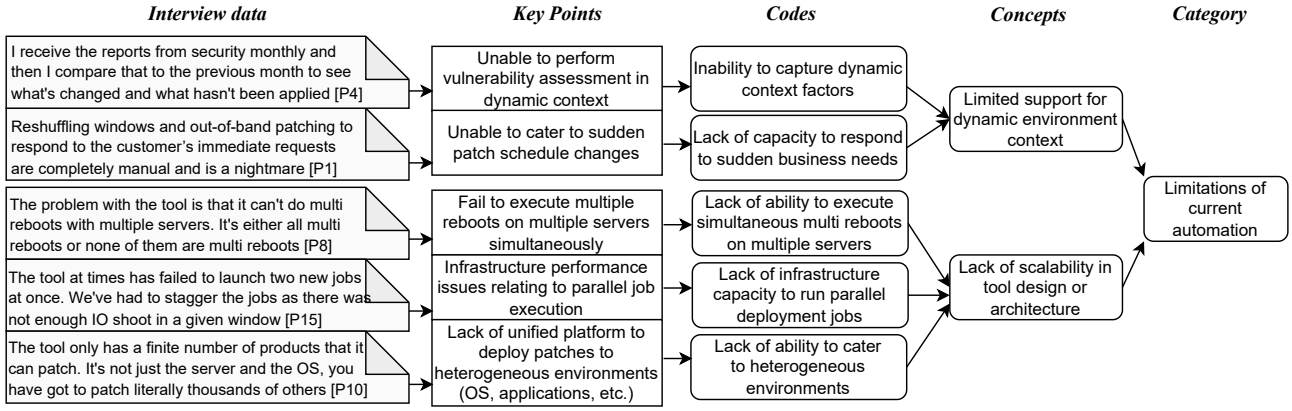

Figure 1: An example of the data analysis steps leading to a category from raw data as evidence.

Microsoft patch caused problems on the printer and it was highlighted in that forum. So at least we had some forewarning to minimise the business impact." - P9

4.1.2 Vulnerability Discovery through Scans. Vulnerability scanning involves performing scans to locate the assets on the network and finding vulnerabilities on the assets which are automated using the tool "Tenable.sc" [57]. The practitioners schedule the vulnerability scanner (e.g., Nessus [56]) to perform the scan and retrieve the results through automatically generated reports. However, identifying the potential locations of a known vulnerability sometimes requires manual effort; for example, writing a script to query what DLLs or java files the machine had in certain locations to find the locations of the Log4j vulnerability [6].

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

4.1.3 Vulnerability Risk Assessment and Prioritisation. Based on the understanding of system vulnerabilities through the scans, practitioners manually perform the risk assessment and prioritisation to decide on patching. This is because the existing scanners fail to incorporate the organisational context and needs which are required for an accurate risk representation. Patch prioritisation is based on the vulnerability severity and impact. Then the need for patch window extensions is decided based on the risk assessment as some security patches can have a wide impact on several other services, for example, Log4j vulnerability [6].

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

Patch prerequisites (e.g., registry changes, group policy object (GPO) changes, preparation package installation) are identified by extensively reading through the patch release notes, knowledge base (KB) articles or information on the vendor's website. On the other hand, scheduling a patch window involves a series of tasks of extensive planning and organising with external stakeholders. Some participants use a third-party integrated risk management solution (e.g., Archer GRC [34]) for organisational governance and compliance with industry regulations; however, configuring the tool is a laborious task. The others manually schedule patches using an Excel spreadsheet.

"Every month, I export the information of 1800 servers from the database to a large sheet with 27 columns and go through it to find the windows." - P13

4.1.5 Testing Patches for Accuracy and Unintended Effects. The participants perform testing in dedicated testing environments (e.g., "Dev", "Test" and "Pre-prod") before rolling out the patches into production ("Prod"). However, due to resource constraints, client machine patching typically employs a staged deployment with a 10:30:60 rollout (i.e., systematically deploying patches to clusters of 10%, 30% and 60% of machines without testing). Preparing the test environment to replicate the "Prod" is also a laborious task due to shared access resulting in too many interdependencies.

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

4.1.6 Deploying Patches to Machines. The participants use several third-party patch management software like Ivanti (server) [1], SCCM (server) [41], WSUS (server) [42], and VMware Workspace ONE (desktop) [61] for deploying patches based on their needs and budget. On average, 67% of the servers (i.e., 1200/1800) are patched automatically. However, the success of automated deployment relies on the accuracy of server information fed into the tool and tool configuration.

In an issue during automated deployment (e.g., the application not automatically restarting upon the server reboot), it is switched to semi-auto allowing practitioners to log into the machines to monitor the patching job while trying to investigate the issue. On average, 11% of the servers (i.e., 200/1800) are patched semi-automatically.

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

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

Figure 2: The as-is state of automation in each phase of the security patch management process. The bold text in the right column denotes the main tasks in each phase and the sub-tasks are listed underneath.

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

4.1.7 Verifying the Success of Patch Deployment. The main verification technique used is scanning the systems post-deployment; the commonly used tools are Ivanti [1] and Tenable.sc [57]. The participants reported that they target 99% coverage of the server fleet as the remaining 1% is usually never completed due to various reasons such as legacy systems.

4.1.8 Handling Post-Deployment Issues. This involves two manually performed activities: identifying what caused the issue and how wide the impact it has caused. Identifying the impact of the issue is important for deciding on the appropriate mitigation actions. The impact is initially measured in terms of the number of client complaints received and vendor announcements. The gathered information is then analysed to calculate the risk and identify the potential workarounds (e.g., rollback) to enable service continuity.

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

4.1.9 Patch Defect Management. To manage the patch defects identified during patch testing through deployment to verification, participants use an in-house application life cycle management (ALM) solution. Incident management (i.e., logging and management of to-do patching requests) is supported by different tool choices such as third-party IT Service Management tools (e.g., Marval [37]) and in-house solutions. However, the coordination of tasks between teams is primarily done manually.

4.2 Limitations of Current Automation

This section presents the limitations of the current automation/tools to answer RQ2. An understanding of the current solutions' limitations is important for identifying the areas of improvement for addressing the identified limitations as discussed in Section 5.

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

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

4.2.2 Lack of Proper Support in Process Workflows. Limited support for patch deployment preparation warrants significant effort for reading through release notes to identify patch prerequisites. Correspondingly, limited support for handling patch dependencies seems to be a common complaint. The lack of a holistic view of the system interdependencies requires practitioners to spend a significant amount of time and effort in identifying the dependencies

during testing and troubleshooting deployment issues. The absence of automation support to handle legacy software dependencies creates further complications, often leading to delays in restarting services to avoid system breakdowns. Another drawback is the lack of support for detecting the need for multiple reboots. Although the tools are capable of automatically executing the scheduled reboots, identifying how many reboots are required is a manual task.

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

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

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

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

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

"The tool has only a finite number of products that it can patch. Imagine the trouble when I have hundreds of products but it can only patch a half." - P10

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

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

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

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

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

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

4.3 Practitioners' Needs for Enhanced Automation

This section answers RQ3, the desires expressed by the participants in response to the interview question *"what tasks do you wish would have been better supported by the technology and how?"*. We raised this question to better understand what practitioners really need to fill the gaps in the current automation to effectively meet their needs.

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

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

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

4.3.2 Central Platform Integrating Vulnerability Scanning and Risk Assessment. The participants reported the need for enhanced filtering and customisation on vulnerability scanning that enables better filtering and information sorting to identify outstanding vulnerabilities at a glance. Additionally, the compiled wish list includes the ability to easily search through a myriad of vulnerabilities to track the remediation status and generate scan reports with better visualisations as the current report only provides summary statistics in Excel spreadsheet format. Another need is the capability to integrate vulnerability scanning and assessment reports

with other tools such as the configuration management database (CMDB). While some existing platforms (e.g., Archer GRC Solution [34]) enable practitioners to view the reports generated through the vulnerability scanner, it requires a lot of manual effort on the customisations.

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

4.3.3 Automated Preparation for Patch Deployment. The need for automation support for identifying the patch prerequisites was highlighted by many participants. As described in Section 4.1.4, detecting prerequisites is currently a manual effort of reading through the release notes. However, the execution of prerequisites is considered difficult to be automated as selecting the optimum configuration needs reasoning based on the environment, therefore demanding human intervention.

"The tools do not consider what vendors have written in their release notes. The release note may say, to complete this patching, you also need to set up this configuration in this manner, and a lot of times this is missed. So a system that can flag which patches require manual intervention to adjust settings would come in handy." - P9

4.3.4 Automation Support to Articulate Patch Scheduling. Another noteworthy desire from many participants was automation support for articulation work in patch scheduling as it involves a series of cooperative tasks (as described in section 4.1.4). The existing tools lack support in managing the integral set of patch scheduling tasks. The participants desire a single dashboard view of all patch schedules enabling easy identification of patch windows based on availability, modification of schedules, tracking status and schedule changes, interactive communication between stakeholders, and integration with the patching tool to export the schedules straight to deployment. Such a platform would assist in articulating patch scheduling and rescheduling tasks among distributed stakeholders to speed up patch deployment.

4.3.5 Automated Patch Deployment With Better User Control. Whilst the current tools are capable of automated patch deployment, the participants expressed the need for enhanced automation with more user control in the capabilities described below. Reduced or no system downtime to address the challenges of service disruptions during reboots and exhaustive manual overhead of the stop-patch-restart process described in Section 4.2.2.

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

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

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

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

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

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

4.4 Role of Human in Process Automation

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

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

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

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

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

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

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

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

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

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

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

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

4.4.3 Handle Legacy Systems In Place. Having *unsupported legacy systems in place* is one of the fundamental reasons requiring human involvement. While the unsupported software poses a huge threat leaving several attack vectors open for exploits, many organisations retain legacy systems because of the service criticality and high cost and complexity involved in migrating legacy systems. Lack of an upgrade path for legacy applications and lack of support from vendors place the burden of handling incompatible legacy software dependencies on the practitioners' shoulders alone.

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

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

4.4.4 Adapt to the Organisational Needs and Culture. Organisational factors including the culture, policies and needs play an important role in the need for human association in process automation. This finding provides additional evidence to previous work [14, 33], which also reported the influence of the organisation's internal policies and management on system administrators in the process.

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

5 DISCUSSION AND FUTURE WORK

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

We present an evidence-based understanding of the role of automation in security patch management that describes the as-is state of automation in practice, its limitations, practitioners' needs

for improved automation support and the role of the human in tsecurity patch management automation. The evidence from Section 4.1 shows that a majority of the tasks in the security patch management process are performed manually (see Figure 2). We find several reasons for the current state of manual work in the process. The key reason is the limitations of the current automation support reported in Section 4.2. This situation stems from different factors such as limitations in the existing tool capabilities (e.g., capturing organisational context in vulnerability assessment), lack of specific features (e.g., identifying prerequisites from patch release notes), performance limitations (e.g., inability to execute parallel deployment jobs), infrastructure constraints (e.g., lack of ability to deploy patches to heterogeneous environments), and usability limitations (e.g., lack of meaningful error messages). These limitations show the gap areas that present excellent opportunities for researchers and tool builders for providing advanced automation/tooling support.

Further, the evidence from Sections 4.2 and 4.3 indicates that certain limitations of the current automation solutions require the practitioners to perform several tasks manually which usually causes delays in patching. We argue that the tool design may have failed to take into consideration why and how automation is actually used in practice. We anticipate our findings would provide a solid understanding of what enhancements in automation support are needed (Sections 4.2 and 4.3) and how those enhancements will be used in practice (Sections 4.1). Such understanding will also be beneficial to shaping future work to effectively address the practical concerns in security patch management. The findings of Section 4.3 can provide an opportunity for tool builders to identify the features that would enable their tools to add more value and practical utility in practice. We recognise that these findings might not capture all the desired tool features as the participants might not be aware of other available tools and their capabilities. This limitation also prompts future work for a mapping of the features of the existing patch management tools to the practitioners' desired needs to scope down *what is missing where*.

Given our findings are context-dependent, i.e., in the healthcare domain, future research is needed for broader validation in different contexts and using additional data sources (e.g., large-scale surveys). The findings from this study can be extended to other domains to identify additional desired features with concrete requirements. Similar to previous work [15, 58], we foresee the value of future work to evaluate the tools in real-world contexts to better understand how well the tool meets the industry needs.

Another set of reasons for the increased manual work stems from the socio-technical implications of human and machine interaction in the security patch management process (reported in Section 2). These reasons demand an in-depth investigation of the critical roles that humans play in gaining control of uncertain situations, understanding the context, making sense of the available information and accordingly making informed decisions as described in Section 4.4. Our findings identify the tasks and points in the security patch management process that require a balance between human control and automation so that future automation can be designed to facilitate a collaborative relationship with humans. The findings present opportunities for future research in *"Human-AI collaboration"* [13], an emerging research paradigm which combines human and machine intelligence to collectively achieve a goal. We also suggest further research to investigate the challenges in developing patch management tools that harness collective human and machine intelligence. Based on our findings and the existing research on "Human-AI collaboration" [3, 12, 27], we propose some recommendations that can guide future tool development to address the limitations of the current automation solutions.

An Integrated Platform Offering Support Across All Process Phases. The lack of an end-to-end automated solution that supports all phases of the security patch management process has been an important anti-pattern recognised in our study, also reported by another study [15]. This situation forces practitioners to use multiple tools for different tasks in diverse environments resulting in increased errors and delays. Since building a unified solution capable of supporting heterogeneous environments (e.g., Windows, Linux, Mac) is highly complex due to the inherent differences in each environment, there is a need for a consolidated platform that can offer end-to-end patch management life cycle support in a single environment to address a majority of the limitations of the existing solutions. Moreover, such a consolidated environment will also benefit from being interoperable with external third-party solutions to cater for the needs of a multi-vendor environment.

Human-Machine Collaboration for Patch Management. We envision the integration of human and machine intelligence can be beneficial in several security patch management tasks reported in Section 4.4. For patch prerequisites identification, we envision a system that not only leverages combined human and machine intelligence but also learns from each of the collaborative partners (i.e., human and machine) to improve future prediction accuracy. Specifically, a machine can convert the unstructured data in patch release notes to a structured format and identify the prerequisites, and provide the summary results to the human partner to ease the selection of the optimum configuration options based on domain knowledge. Further improvements can be made by leveraging machine learning techniques to predict the most suitable configuration options that can guide practitioners to make quick decisions.

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

Human-Centred AI Explanations to Assist Contextual Decision-Making. To support human decision-making based on contextual understanding (Section 4.4.2), we propose that future automation focuses on enabling human-centred explanations. As

suggested by the Human-AI Interaction design guidelines [3] "make clear why the system did what it did", we anticipate such a capability will assist humans in understanding a machine's predictions thereby allowing them to make better decisions based on the context. For example, in selecting the optimum configuration during patch testing, a machine can provide the basis for the prediction. As another important design functionally, such a system shall enable interactive bi-directional communication. Interactive explanations enable humans to interact with machines [32]; for example, by editing input and changing the prediction based on the context. Further, a machine can learn from human interactions and evolve to improve its predictions by leveraging machine learning training techniques such as Reinforcement Learning and Active Learning.

Decision Support for Patch Scheduling. The evidence reported in Section 4.1.4 shows that sudden changes to the patch schedules demand an arduous manual task of patch rescheduling. Further, it can produce a cascading effect on the dependent patch schedules resulting in increased complexity for rescheduling and further delays in patching. Responding to these challenges, we suggest developing a smart decision support system to guide practitioners toward accurate and quick decisions about patch scheduling and rescheduling. The envisioned system should be able to identify the cascading effect on dependent patch schedules, i.e., which schedules will likely get affected and how much is the impact, based on the identified patch dependencies and prerequisites. Accordingly, the system should provide an estimate of the patching delay. We argue that these insights if presented in simple visualisations (e.g., graphs), would lead practitioners to make informed decisions faster about rescheduling.

6 LIMITATIONS

In this section, we discuss the potential threats to validity and how they were mitigated based on the guidelines reported in [39, 50].

External Validity: Our findings do not claim to be generalised from a sample to a population but rather applicable to the studied cases like most qualitative research. Our findings are limited to the practitioners in the case organisations in the domain of healthcare studied in-depth to provide a holistic understanding of the studied topic grounded in evidence that is not attainable through broader but shallower approaches (e.g. surveys). However, we believe that our findings can be recreated and adapted in other similar contexts; for such purpose, we have provided sufficient details about the studied participants and the research methodology (section 3) to facilitate transferability.

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

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

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

Reliability: To ensure the reliability of the findings, the data collection, analysis process, and the emerged findings were thoroughly discussed among all authors in weekly meetings, as described in Section 3.2. Furthermore, two researchers were present in seven interviews to minimise the threat of researcher error.

7 CONCLUSION

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

ACKNOWLEDGMENTS

The authors thank our industry collaborators, without whose help this research would not have been made possible. We sincerely thank all the participants who generously shared their time and experiences in interviews with us. This study was conducted under the University of Adelaide Human Research Ethics Committee Application ID H-2020-035.

REFERENCES

[1] 2022. Ivanti. Retrieved February 17, 2022 from https://www.ivanti.com/
[2] A. Al-Ayed, S.M. Furnell, D. Zhao, and P.S. Dowland. 2005. An automated framework for managing security vulnerabilities. *Information management & computer security* 13, 2 (2005), 156–166. https://doi.org/10.1108/09685220510589334
[3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI*

2019). ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300233

[4] Marco Angelini, Graziano Blasilli, Tiziana Catarci, Simone Lenti, and Giuseppe Santucci. 2018. Vulnus: Visual vulnerability analysis for network security. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 183–192.

[5] Marco Angelini, Silvia Bonomi, Simone Lenti, Giuseppe Santucci, and S Taggi. 2019. MAD: A visual analytics solution for Multi-step cyber Attacks Detection. *Journal of Computer Languages* 52 (2019), 10–24.

[6] Apache. 2022. Apache Log4j. Retrieved April 21, 2022 from https://logging.apache.org/log4j/2.x/security.html

[7] Frederico Araujo and Teryl Taylor. 2020. Improving cybersecurity hygiene through JIT patching. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20)*. ACM, New York, NY, USA, 1421–1432. https://doi.org/10.1145/3368089.3417056

[8] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2010. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 105–114.

[9] Ramesh Chandra, Taesoo Kim, Meelap Shah, Neha Narula, and Nickolai Zeldovich. 2011. Intrusion recovery for database-backed web applications. In *Proceedings of the 26th Annual Computer Security Applications Conference*. 101–114.

[10] Chuan-We Chang, Dwen-Ren Tsai, and Jui-Mi Tsai. 2005. A cross-site patch management model and architecture design for large scale heterogeneous environment. In *Proceedings 39th Annual 2005 International Carnahan Conference on Security Technology*. IEEE, 41–46.

[11] 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.

[12] James A. Crowder, John Carbone, and Shelli Friess. 2020. Human–AI Collaboration. In *Artificial Psychology*. Springer, Cham. https://doi.org/10.1007/978-3-030-17081-3_4

[13] Dominik Dellermann, Adrian Calma, Nikolaus Lipusch, Thorsten Weber, Sascha Weigel, and Philipp Ebel. 2019. The Future of Human-AI Collaboration: A Taxonomy of Design Knowledge for Hybrid Intelligence Systems. In *Proceedings of the 52nd Hawaii International Conference on System Science*.

[14] 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)*. ACM, 1272–1289. https://doi.org/10.1145/3243734.3243794

[15] 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

[16] 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, 793–805. https://doi.org/10.1145/3468264.3468595

[17] 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. *Proceedings of the ACM on Human-Computer Interaction (CSCW)* 6, 362 (2022). https://doi.org/10.1145/3555087

[18] 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.

[19] Chief Healthcare Executive. 2022. *Cyberattacks in healthcare surged last year, and 2022 could be even worse.* Retrieved April 24, 2022 from https://www.chiefhealthcareexecutive.com/view/cyberattacks-in-healthcare-surged-last-year-and-2022-could-be-even-worse

[20] Center for Internet Security (CIS). 2022. *Cyber Attacks: In the Healthcare Sector.* Retrieved April 24, 2022 from https://www.cisecurity.org/insights/blog/cyber-attacks-in-the-healthcare-sector

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

[22] Michael Hicks and Scott Nettles. 2005. Dynamic Software Updating. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 27, 6 (2005), 1049–1096. https://doi.org/10.1145/381694.378798

[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] QSR International. 2022. *Nvivo qualitative data analysis software.* https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home

[25] Chadni Islam, Victor Prokhorenko, and Muhammad Ali Babar. 2022. Runtime Software Patching: Taxonomy, Survey and Future Directions. *arXiv preprint arXiv:2203.12132* (2022). https://arxiv.org/pdf/2203.12132.pdf

[26] 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.

[27] Ece Kamar. 2016. Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence. In *IJCAI*. 4070–4073.

[28] Patrick Kamongi, Srujan Kotikela, Krishna Kavi, Mahadevan Gomathisankaran, and Anoop Singhal. 2013. Vulcan: Vulnerability assessment framework for cloud computing. In *2013 IEEE 7th International Conference on Software Security and Reliability*. IEEE, 218–226.

[29] Sanidhya Kashyap, Changwoo Min, Byoungyoung Lee, Taesoo Kim, and Pavel Emelyanov. 2016. Instant {OS} Updates via Userspace {Checkpoint-and-Restart}. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 605–619.

[30] Kuinam J Kim and Minsu Kim. 2017. A study of integrity on the security patches system using PM-FTS. *Wireless Personal Communications* 94, 2 (2017), 165–173.

[31] Yonggun Kim and Yoojae Won. 2020. A new cost-saving and efficient method for patch management using blockchain. *The Journal of Supercomputing* 76, 7 (2020), 5301–5319.

[32] Vivian Lai, Samuel Carton, and Chenhao Tan. 2020. Harnessing explanations to bridge ai and humans. *arXiv preprint arXiv:2003.07370* (2020). https://arxiv.org/pdf/2003.07370

[33] 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.

[34] RSA Security LLC. 2022. Archer GRC Solution. Retrieved February 21, 2022 from https://www.archerirm.com/content/grc

[35] Gilad Maayan. 2022. *Five years later, Heartbleed vulnerability still unpatched.* Retrieved April 24, 2022 from https://blog.malwarebytes.com/exploits-and-vulnerabilities/2019/09/everything-you-need-to-know-about-the-heartbleed-vulnerability/

[36] Martin N. Marshall. 1996. Sampling for qualitative research. *Family practice* 13, 6 (1996), 522–526. https://doi.org/10.1093/fampra/13.6.522

[37] Marval. 2022. Marval ITSM. Retrieved February 20, 2022 from https://www.marval.co.uk/

[38] Matthew Maurer and David Brumley. 2012. TACHYON: Tandem execution for efficient live patch testing. In *21st USENIX Security Symposium (USENIX Security 12)*. 617–630.

[39] 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

[40] Peter Mell, Tiffany Bergeron, David Henning, et al. 2005. Creating a patch and vulnerability management program. *NIST Special Publication* 800 (2005), 40.

[41] Microsoft. 2022. Microsoft System Center Configuration Manager. Retrieved February 16, 2022 from https://docs.microsoft.com/en-us/mem/configmgr/

[42] Microsoft. 2022. Windows Server Update Services. Retrieved February 17, 2022 from https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus

[43] Soranut Midtrapanon and Gary B Wills. 2019. Linux Patch Management: With Security Assessment Features. In *International Conference on Internet of Things, Big Data and Security*. 270–277.

[44] 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

[45] Felicia M. Nicastro. 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

[46] NIST. 2022. *NVD Common Vulnerability Scoring System.* Retrieved March 10, 2022 from https://nvd.nist.gov/vuln-metrics/cvss

[47] National Audit Office. 2017. *Investigation: WannaCry cyber attack and the NHS.* Report by the Comptroller and Auditor General. Department of Health.

[48] Michal Procházka, Daniel Kouril, Romain Wartel, Christos Kanellopoulos, and Christos Triantafyllidis. 2011. A Race for Security: Identifying Vulnerabilities on 50 000 Hosts Faster than Attackers. In *Proceedings of Science (PoS) International Symposium on Grid and Clouds.*

[49] Md Sazzadur Rahman, Guanhua Yan, Harsha V Madhyastha, Michalis Faloutsos, Stephan Eidenbenz, and Mike Fisk. 2013. iDispatcher: A unified platform for secure planet-scale information dissemination. *Peer-to-Peer Networking and Applications* 6, 1 (2013), 46–60.

[50] 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.

[51] Thomas A. Schwandt. 1997. *Qualitative Inquiry.* Sage, London.

[52] Accenture Security. 2021. *State of Cybersecurity Resilience 2021.* Retrieved March 10, 2022 from https://www.accenture.com/_acnmedia/PDF-165/Accenture-State-Of-Cybersecurity-2021.pdf

[53] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. ACM, 120–131. https://doi.org/10.1145/2884781.2884833

[54] Anselm L. Strauss and Juliet M. Corbin. 1998. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory* (2nd ed.). Sage.

[55] Anselm L. Strauss and Juliet M. Corbin. 2007. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3rd ed.). Sage.

[56] Tenable. 2022. Nessus. Retrieved April 21, 2022 from https://www.tenable.com/products/nessus

[57] Tenable. 2022. tenable.sc. Retrieved February 17, 2022 from https://www.tenable.com/products/tenable-sc

[58] Christian Tiefenau, Maximilian Häring, Katharina Krombholz, and Emanuel von Zezschwitz. 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.

[59] Slim Trabelsi, Henrik Plate, Amine Abida, M Marouane Ben Aoun, Anis Zouaoui, Chedy Missaoui, Sofien Gharbi, and Alaeddine Ayari. 2015. Mining social networks for software vulnerabilities monitoring. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 1–7.

[60] Uraz Cengiz Türker, Robert Hierons, Mohammad Reza Mousavi, and Ivan Tyukin. 2021. Efficient state synchronisation in model-based testing through reinforcement learning. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 368–380.

[61] VMware. 2022. VMware Workspace ONE. Retrieved February 17, 2022 from https://www.vmware.com/products/workspace-one.html

[62] Shangwen Wang, Ming Wen, Bo Lin, Hongjun Wu, Yihao Qin, Deqing Zou, Xiaoguang Mao, and Hai Jin. 2020. Automated Patch Correctness Assessment: How Far Are We?. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*. ACM, New York, NY, USA, 968–980.

[63] Wu Zhou, Peng Ning, Xiaolan Zhang, Glenn Ammons, Ruowen Wang, and Vasanth Bala. 2010. Always up-to-date: scalable offline patching of vm images in a compute cloud. In *Proceedings of the 26th Annual Computer Security Applications Conference*. 377–386.