National College of Ireland

The Hyperautomation of Software Security Patch Management in Enterprise Networks: A Case Study at the Central Bank of Ireland.

MSc Industrial Internship
Msc. Cybersecurity

Oluwasefunmi Alabi
Student ID: x21130094

School of Computing
National College of Ireland

Supervisor: Vikas Shani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	OLUWASEFUNMI MOYINOLUWA ALABI
Student ID:	X21130094
Programme:	MSc CYBERSECURITY **Year:** 2022
Module:	MSc INTERNSHIP
Supervisor:	VIKAS SHANI
Submission Due Date:	6TH JANUARY, 2023
Project Title:	THE HYPERAUTOMATION OF SOFTWARE SECURITY PATCH MANAGEMENT IN ENTERPRISE NETWORKS: A CASE STUDY AT THE CENTRAL BANK OF IRELAND.
Word Count:	6601 **Page Count** 18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	OLUWASEFUNMI MOYINOLUWA ALABI
Date:	3RD JANUARY, 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Hyperautomation of Software Security Patch Management in Enterprise Networks: A case study within the Central Bank of Ireland.

Oluwasefunmi Alabi
X21130094

Abstract

Several studies have shown that delays in patching software vulnerabilities results in a number of devastating cybersecurity attacks. While software security patch management automation has received a lot of recognition from practitioners and researchers, there has been relatively little effort put into gaining automated assistance for various security patch management processes and tasks which have the potential to significantly shorten the time it takes to install security fixes and further improve software security management. However, it is additionally essential to comprehend how automation is applied in practise, its limitations in meeting real-world objectives, and what security practitioners actually want. This is an area of study that has yet to be experimentally explored in the research on software engineering.

In order to investigate various elements of hyperautomation and automation for security patch management, this study administered research interview to 10 security professionals working in the technology delivery and Information security teams respectively within the Central Bank of Ireland, after which a proof-of-concept implementation of an improved solution is implemented from its findings. The findings focus on the role of automation and hyperautomation in security patch management for unveiling data about the present status of automation in practise, its constraints and how hyperautomation support can be improved to successfully meet practitioners' demands, and the role of humans in an automated process. A series of recommendations is developed on how to focus future work on creating automated assistance for security patch management to support this based on the findings.

1. Introduction

The timely deployment and use of security updates for identified security flaws are viewed as being extremely crucial in mission-critical or production areas within an enterprise, to prevent security breaches that could succeed and have fatal repercussions (Souppaya & Scarfone, 2013). Delays in fixing security flaws may result in serious data losses, as seen in the Equifax data breach instance[1] (Goodin 2017), or even fatalities as seen when some crucial systems of the UK's National Health Services (NHS) and hospitals in Düsseldorf Germany, were shut down due to a delay in installing a patch that the WannaCry ransomware cryptoworm exploited[2] .

According to (Jenkins et al., 2020), one of the primary causes of successful sophisticated cyberattacks is the availability of security fixes but their non-installation. For instance, the Heartbleed security vulnerability that went unpatched in systems for years is an example of this[3]. Dissanayake et al., (2022) highlights how a growing body of research is focusing on the software security patch management process, also known as security patch

management, in an effort to comprehend the causes of and offer remedies to the delays in implementing security patches. Detecting, acquiring, testing, deploying, and confirming security patches are only a few of the operations that comprise software security patch management, which involves collaboration between several stakeholders.

Araujo and Taylor (2020) asserted that automation, like most other software engineering disciplines, has enormous potential for greatly enhancing the efficacy and efficiency of security patch management in order to minimise application delays. This has spurred interest for developing and incorporating automation assistance into various process steps for security patch management. These automated methods/tools include, for instance, those for patch information retrieval, vulnerability scanning, evaluation and prioritisation, patch testing, patch deployment, and post-deployment patch verification. Delays in actual security patch management still remain, notwithstanding these research efforts showing promise for expediting various operations of security patch administration by offering automated solutions.

To develop appropriate automated approaches and tools for successfully addressing delays in security patch management, this research contended that it is crucial to experimentally assess the current status of automation, the real-world challenges it faced, and the practitioners' needs for improved automation support. To the best of knowledge when this research was conducted, there wasn't any actual data on these aspects of automation for security patch administration.

An extensive analysis utilising in-depth semi structured interviews was administered to 10 practitioners from within the IT directorate in the Central Bank of Ireland, a leading organisation in the financial regulatory sector, to better understand the function and user demands of automation for enhancing security patch management. To methodically discover the results in a bottom-up fashion, Corbin and Strauss (2008) straussian adaptation of the Grounded Theory technique was used in this study.

The findings of this study contribute to the state-of-the-art understanding of practice and research by (i) providing an evidence-based knowledge on the as-is state of security patch management automation, by describing the current manual and automated tasks in the process, (ii) recognising the shortcomings of the conventional automation in practise, (iii) Addressing how automation support might be improved to better meet the demands of security practitioners, (iv) Highlighting the roles of people in the automation process, and (iv) describing the role of the human in automating security patch management, highlighting the responsibilities and points in the process where automation is needed to facilitate a "human in the loop" approach, and (v) discussing the future research directions and offering a set of recommendations and proof of concept that can direct software developers and security practitioners to address the found gaps and user needs.

2. Related Work

2.1. Background on automating software security patching

Mell et.al (2005) describes patches for software security in research as bits of code created to resolve security issues found in software. Due to their importance, organisations frequently give security patches priority over non-security updates (such as feature patches) in their patch management. In related research by Dissanayake et al (2022), the phrase "multifaceted process of detecting, purchasing, testing, installing, and confirming security fixes for the discovered third-party vulnerabilities" is used to describe security patch management. It is further explained that the procedure comprises of five steps, as listed in Table 1.

Although no study has explicitly investigated the function of automation in the process in practise, research by Araujo and Taylor (2020) reveals efforts focused on providing automation for security patch management in order to increase the efficiency and reduce delays in the patch management process. Instead of assisting the entire process, the majority of earlier initiatives have concentrated on automating particular security patch administration duties. For automated patch information extraction from different sources, tailored information sifting, information validation, and patch download and dissemination, an automated framework by Al-Ayed et al (2005) have suggested methods. A central platform that integrates the scan findings from several information sources has been recommended for vulnerability scanning, Angelini et al (2019). Angelini et al (2019), Bozorgi et al (2010) and Kamongi (2013) have suggested adaptable tools/approaches for vulnerability risk assessments based on the industry benchmark, the Common Vulnerability Scoring System (CVSS), for vulnerability risk assessment and prioritisation.

Table 1: steps involved in managing security patches.

Process Phase	Description of tasks
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

A number of programmes have tried automated detection of flawed and malicious patches, in the context of patch testing Crameri et al (2007). However, only a small amount of focus has been placed on minimising disturbance when recovering from crashes brought on by flawed updates. According to Kim and Won (2019), automating patch deployment is the subject of several research. Araujo and Taylor (2020) explain how several strategies have been developed to reduce downtime and service interruptions, including dynamic software updating (DSU), JIT patching, and fast kernel upgrades. According to Procházka et al (2011), for post-deployment patch verification, automated methods for patch deployment verification, exploit discovery, and patch repair have been provided. An important observation is that only a few studies have thoroughly assessed the offered solutions in real-world settings, which implies that there is limited experimental understanding of how well the solutions have met the demands of the practitioners.

2.2. Role of coordination in Software security patch management

High-level descriptions of the interdependencies between various stakeholders, including Security vendors and organisations, provided in a study by Hanauer et al (2018), similarly, Nappa et al (2015) revealed that vendor dependency coordination concerns are created by inconsistent coordination in patch rollout from several vendors, as a result of shared security flaws in the application software. A study was designed around a large data collection of client-side vulnerability patches. In comparison, a few other studies (Dey et al 2015, Cavusoglu et al 2006, and Cavusoglu, Cavusoglu, and Zhang, 2008) focused on optimising patch management through synchronization the organization's patch cycle with the vendor's patch rollout to reduce patching expenses and risks. As a result, there is an urgent need for comprehensive understanding of the function and impacts of such dependencies on security patch management. This is because the stated dependencies with software suppliers pose serious issues.

Research by Hai Huang et al.,(2012) also reports difficulties with collaboration and coordination during patch management processes. Nevertheless, the research did not go into great detail on the causes for the coordination needs and associated issues, the implications for the patch management procedure, and the impact of delays on patch deployment. Another group of studies from Tiefenau & Krombholz (2020) looks into the socio-technical elements of security patch management, including the process and difficulties of patch management, the significance of coordination and collaboration, system administrators' practises, behaviours, and experiences, as well as the causes of and mitigation measures for security patch delays. Understanding the socio-technical components is deemed crucial since security patch management is really a socio-technical endeavour in which people and technology constantly interact to allow team members to efficiently coordinate and cooperate utilising the available tools (Zhou et al 2010). A crucial finding in a study by Dissanayake et al (2021) is the emphasis on the necessity of human engagement in the patch management process despite developments in automation to shorten patch management delays. It has not been looked into why human engagement in process automation is necessary or how to strike the ideal balance between automation and human involvement.

In contrast to the previous works, this study's effort aims to provide a full understanding and role of automation in security patch administration, supported by data obtained from practitioners. By describing the automated and manual tasks in the workflow, the limitations of the current automation in meeting practitioners' needs, ways to increase automation support to help practitioners, and the responsibilities of people in process automation, i.e., It illustrates why and where human intervention is required, as well as how practitioners have blended automation into the security patch management process. Given the significance of timely security patch management, the findings of this study provide a solid foundation for suggesting practical alternatives to address the deficiencies of the current automated help for security patch management.

3. Research Methodology

This study intends to comprehend the practical application of automation in security patch administration. Semi-structured interviews are used in the qualitative study. It intends to assist better comprehending real-world activities from the viewpoints of practitioners. Closed and open-ended research questions (RQs) were administered to accomplish this goal.

3.1. Data Collection

Recruitment of participants: 10 practitioners from three teams within the IMTD directorate of the central bank were interviewed for the purpose of gathering data (Team A, B and C). When choosing the teams, Schwandt (2014) purposive selection method combined with Marshall (1996) convenience selection method were used to ascertain that the data are representatives of the important region through which the findings arise. The central Bank's Service Lifecycle team (Team A), is in charge of maintaining its IT systems and providing IT services. They not only managed the bank's own internal bespoke financial regulatory software applications, but also the software systems of third-party providers. However, Team B (Information security) is exclusively responsible for all security operations, and Team C (an IT services and consulting firm) was hired to oversee patch management of operating systems (OS) and related third-party apps and servers. Other smaller teams in the bank handled the non-security updates, which are outside of the study's purview.

The respective team managers were tasked to recommend suitable interview subjects who work in the case organisations' security patch management and deployment. Then, invited the nominated individuals through email. For a well-rounded viewpoint, participants were chosen from a variety of security patch management job roles. The participants were emailed a pre-interview questionnaire to gather demographic data about them and their team.

Semi-structured Interviews: Conducting semi-structured interviews from November to December 2022, 10 participants from 3 teams within the bank were interviewed. Due to work from home limitations, the interviews were done in a hybrid format, physically and through Webex and lasted between 30 and 60 minutes. The interview was aimed to help increase the reliability of the interviewing process by asking follow-up questions. Software-related security patch management for operating systems and third-party software programmes within the financial regulatory sector was the main topic of the interviews.

Below are some of the topics addressed by the open-ended interview questions:

(a) The roles and responsibilities involved in security patch management

(b) The state of the automation tools currently being used in the process

(c) The weakness and restrictions posed by the current automation tools

(d) The part played by humans in security patch management automation and the justifications for their presence

(e) The requirement for better automation support (i.e., the proposed solutions and their features)

The interview guide was created and changed as necessary to prevent any bias during data collection. The results were documented and typed out into transcripts of every interview for analysis. Parallel to the data processing, there were two phases to the interview process. Implementing a qualitative research technique by Corbin et al. (2008), until the data analysis ascertained theoretical saturation, continuous gathering of data was done, for instance, the final three interviews added additional examples to the results that had already emerged but no new categories or perspectives emerged.

3.2. Data Analysis

For data analysis, Strauss and Corbin, (2008) Grounded Theory (GT) was utilised, since it aligned nicely with the objectives of the study, which were to investigate a phenomenon, knowing the function of automation in this situation's real-world setting is important in actual security patch management. The Straussian GT version, developed by Strauss and Corbin, offers a method for data analysis that is organised and directed by open-ended and RQs depending on practise as advised by Stol et al.(2016). Various coding methods, including open, axial, and selective Coding. The data analysis was carried out, after which its codes and memoranda compiled into a Codebook. To decrease bias and improve the reliability of the results, cross-validation of codes, ideas and categories presented in the Codebook were done against the unprocessed data (i.e., interview transcripts). The categories and their connections were exhaustively reviewed during weekly meetings before being decided upon following multiple iterations. Weekly, in-depth meetings to resolve any issues that arose throughout the process was done. The process of data analysis was made easier by the introduction of analytical tools like memoing and diagramming. A qualitative analysis programme called NVivo was used to store the interview transcripts (NVivo, 2022).

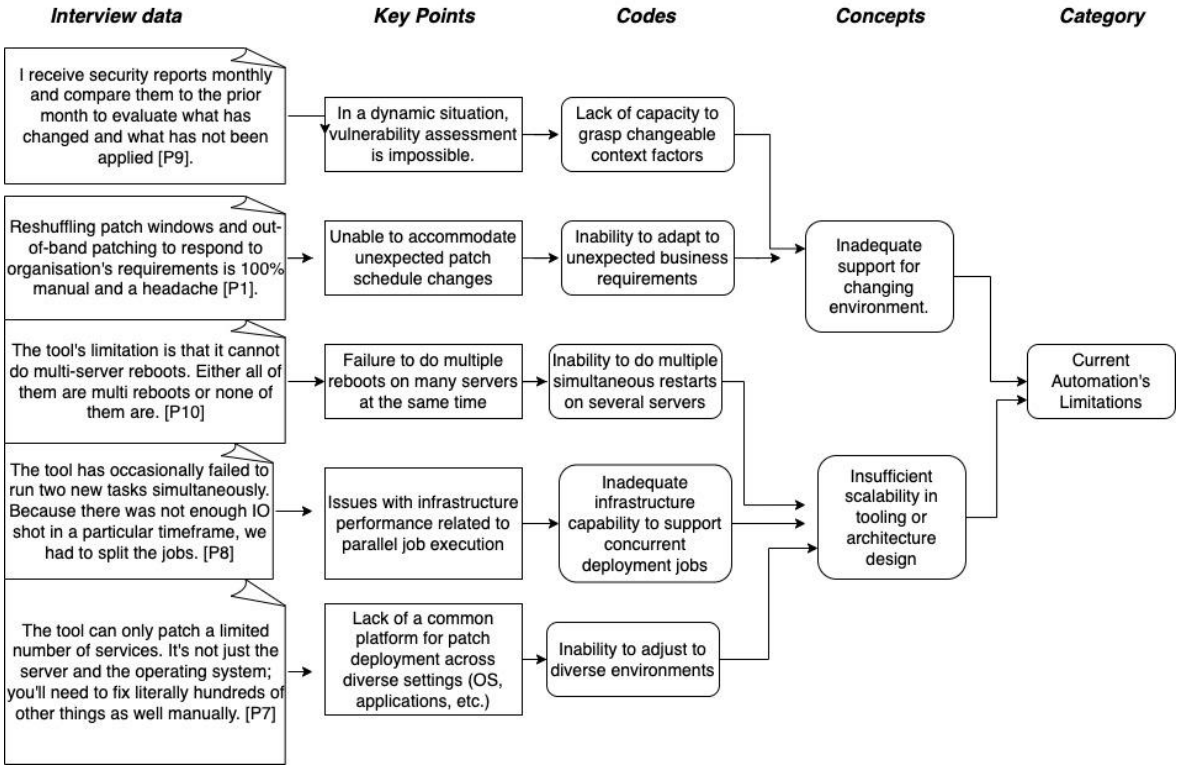

Figure 1: Data analysis processes leading to a category as proof from source data.

The interview transcripts were reviewed line by line during **open coding** and the information summarized as important points. Short sentences were used to further encapsulate it into codes. The codes were continually compared between interview transcripts and within one transcript to produce ideas (Strauss and Corbin, 2008), a more abstracted version of the codes. Similar to this, categories were created, which represented the following degree of abstraction (see Figure 1). Axial coding was utilised to uncover the relationships between the developing categories, connecting them methodically at the level of their attributes (i.e., properties of a category) and sizes (i.e., Dissimilarities within attributes).

The **axial coding** process was aided by activities such as going back and forth to memos, or notes made during open coding that explained the codes and their linkages, by creating diagrams of the interrelationships, and by fine-tuning them at regular team meetings. The method was carried out until no further characteristics, dimensions, or connections could be found for any of the categories, which indicated theoretical saturation (Strauss and Corbin, 2008). Finally, **selective coding** was implemented, in which the categories were finalised and merged with the major issue of the study, in this case, the function of automation in security patch administration, also known as the key category (Strauss and Corbin, 2008).

Due to confidentiality agreements with the industry collaborators, who happen to be in the financial regulatory sector where data sensitivity has additional levels of oversight, the raw data acquired for this study cannot be disclosed. However, the interview guide and configuration manual, which included the pre-interview form and interview questions, are released to the public.

4. Design Specification

The design specification depicts the overall flow of the study in diagrammatic form. As part of the research process, many stages of this investigation have been discussed. The project begins by distributing a questionnaire to determine which aspects of the present automation process need to be improved. The obtained and analysed results are converted into a proof-of-concept enhancement that is evaluated in an AWS cloud tenant through the deployment of the enhanced patch management system. Figure 2 displays the configuration flow of a proof of concept.

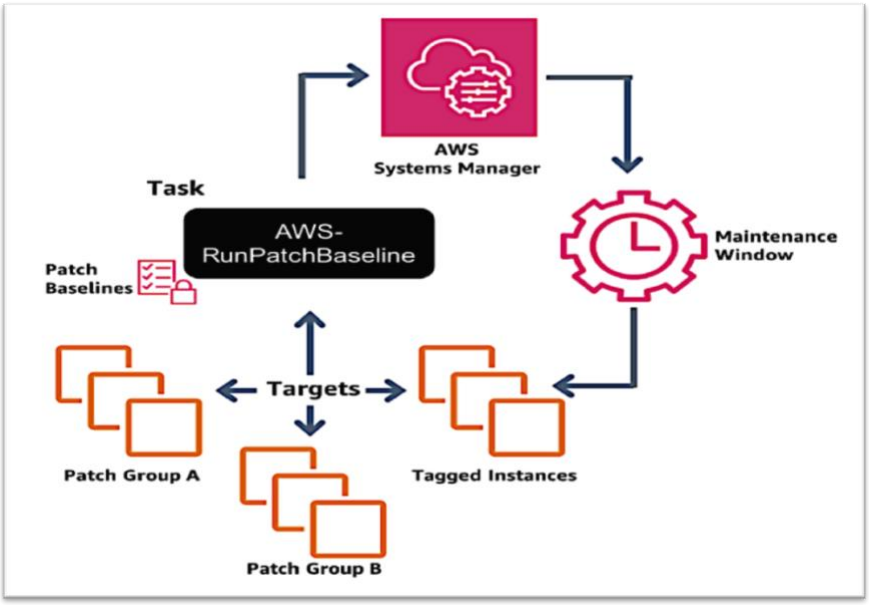

Figure 2: Automated process flow of AWS cloud tenant patch Management.

5. Project Prerequisite

Provisioning an AWS cloud tenant is required. This research utilized AWS free tier subscription for the design and implementation of this proof of concept. The cloud tenant consisting of 3 Linux web servers each configured on a t2.micro EC2 instance running Amazon Linux 2 OS with 8 GB of RAM, and a 30 GB SSD storage attached. 1 Active Directory Domain Controller configured on a t2.micro EC2 instance running Windows Server 2022 Data Centre OS with each configured with 8 GB of RAM, and a 30 GB SSD attached, and 2 Windows 10 Enterprise client instances each running on a t2.micro instance with 8 GB of RAM, and 30 GB SSD storage attached. The following AWS tools are required to run the project:

5.1. AWS Systems Manager (SSM): SSM organises the IT inventory by categorising resources by application or environment, and it integrates with CloudWatch to allow access to dashboards and operating data. SSM may automate operational operations including running pre-defined commands on one or more EC2 instances, altering the instance state, attaching/detaching EBS volumes, making snapshots, and applying patches and upgrades to increase efficiency.

5.2. AWS Patch Manager: This is used to automate the task of patching managed instances with security fixes. This also deploys fixes for non-security upgrades on Linux instances. Windows, Ubuntu Server, various major Linux distributions, and Amazon Linux are all supported. This is configured to have options of scanning instances to present merely a report of missing patches, or automatically scan and apply all needed patches.

The other steps in this process entailed (i) Establishing an IAM role for the Systems Manager to undertake patch operations (ii) Associating patch baselines for EC2 instances to specify what patches are be deployed (iii) Programming Lamda functions, using Python ro trigger automation workflow and establishing a maintenance window to ensure instances are patched when specified. The Cron job is triggered by AWS Event Bridge (iv) Creating a CloudWatch task to monitor patch compliance and ensure the instances are patched.

6. Implementation

6.1. IDE and packages

An IDE (Integrated Development Environment) is a piece of software that allows the execution of programming languages. Several IDEs are available, such as pycharm, vscode, and jupyter. Jupyter notebook, which offers a web interface for authoring, executing, and reviewing code, was chosen as the IDE of choice for this research. Python was utilised throughout this research since it enables for the manipulation of big datasets as well as its versatility in automating tasks and assessment (Raschka, 2015). During this research, the following libraries and packages were utilized.

6.2. Experiment 1: AWS tenant patching in a homogenous environment

As explained in the research methodology, the questionnaire was administered after which its results and conclusions lead to a patching proof of concept within a homogenous IT tenant containing devices of same operating systems and services. The AWS Lambda function was programmed to apply patch baselines specific to the type of environment before execution. This experiment was conducted with 4 homogenous server types and 2 client machines present in the tenant. It successfully automates high-end tests on the cloud to speed up release cycles. Many frameworks and technologies for web testing and application testing automation, such as Selenium, are supported by the tool. The test execution platform simplifies the execution of

scripts. It further minimised test execution time by employing same patch baseline policy on concurrent patch jobs, as well as track the issues that can be remedied.

6.3. Experiment 2: AWS tenant patching in a dynamic environment

Unlike the first iteration, this proof of concept, focused on applying different patch baseline policies according to device specification due to it being a dynamic environment consisting of devices of various operating systems and services. The AWS Lambda function was utilized to program patch baselines in python, specific to the type of operating system before execution. It was used to organise test cases logically. Furthermore, the solution enables the testing team to instantly begin site monitoring and load testing without the need for any difficult setups or instals.

This experiment was conducted with 2 Linux web servers, 1 Active Directory Domain controller, 2 Windows 10 Enterprise client machines and 1 Mac OS client device present in the tenant. To prevent duplication among several storage devices, patch activities and logs are mapped dynamically and stored in an S3 bucket upon completion of the process. This was done to fulfil the need for a central knowledge repository in the patch management process.

7. Evaluation

The subsections of this part present the outcomes from this study for several areas of automation in security patch management.

7.1. The Current State of Automation in Security Patch Management

Findings in response to RQ1 show that, although certain security patch administration procedures have full or partial automated assistance, others are wholly human. Figure 3 summarises the current state of automation as stated by participants, as represented by each process phase in Table 1. It also shows how practitioners have integrated automation into key processes and the technologies they utilise.

7.1.1. Retrieving Patch Information from Third-Party Vendors

Retrieving Patch Information from Third-Party Vendors. Most patch deployment technologies obtain fixes from third-party vendor sites and deliver them to patching towers automatically. Learning about new patch releases, on the other hand, is a manual process in which practitioners proactively look for patch release information via multiple sources such as direct vendor calls, security professional mailing lists, community Slack channels, and official vendor website or internet forums.

7.1.2. Patch Planning, Testing and Preparation for Deployment

The practitioners do a series of time-consuming manual operations such as modifying machine configurations, identifying patch requirements, managing patch dependencies, server commissioning, and organising patch schedules. Identifying patch requirements and scheduling work are seen as the most difficult, requiring substantial manual overhead. The practitioners test patches in specific testing environments (e.g., "Dev," "Test," and "Pre-prod") before releasing them into production. Due to resource restrictions, client machine patching is often done in stages using a 10:30:60 rollout (i.e., progressively sending fixes to clusters of 10%, 30%, and 60% of machines without testing). Preparing the test environment to duplicate the "Prod" environment is likewise a time-consuming operation owing to shared access, which results in excessive interdependencies

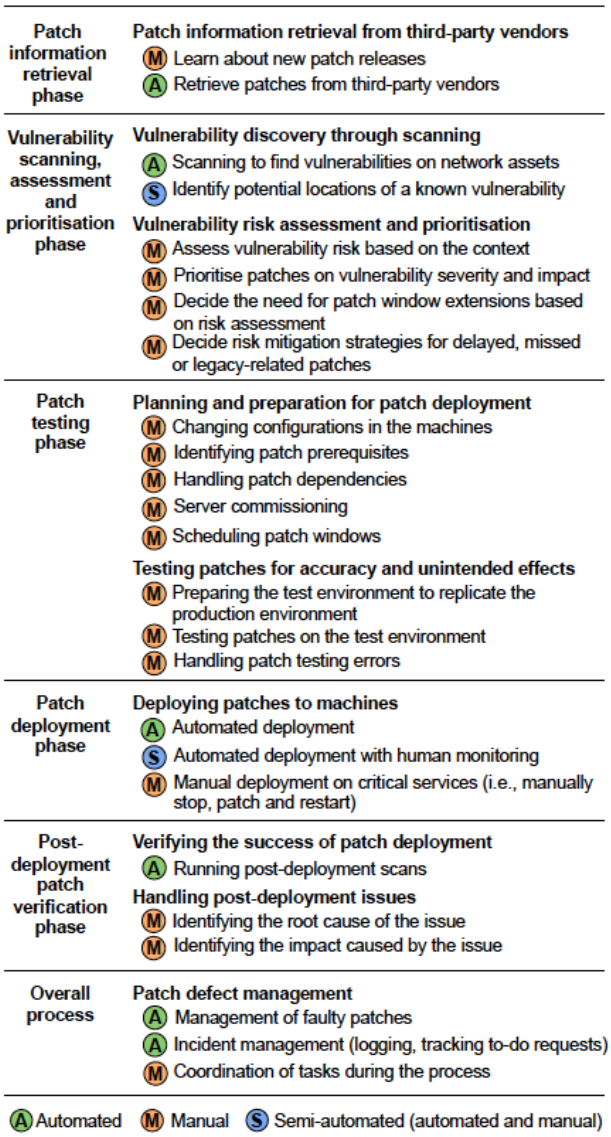

Figure 3: The current state of automation in security patch management process

7.1.3. Managing Patch Deployment and Post-deployment Issues

The practitioners apply fixes depending on their needs and budget using third-party patch management tools such as Microsoft System Centre Configuration Manager (SCCM) (server) and VMware Workspace ONE (desktop). On average, 68% of servers (2176/3200) are automatically patched. The effectiveness of automated deployment, on the other hand, is dependent on the correctness of the server information provided into the tool and the tool settings. When there is a problem during automated deployment (for example, the programme does not resume automatically when the server reboots), it is converted to semi-auto, allowing practitioners to connect into the machines to watch the patching operation while investigating the problem. Semi-automatic patching affects 13% of servers (416/3200).

Handling post deployment issues which entailed two manual tasks: determining what triggered the problem and how widespread its impact was. Identifying the issue's impact is critical for deciding on appropriate mitigating steps. The first effect is assessed in terms of the number of client complaints and vendor announcements. The collected data is then analysed in

order to evaluate the risk and suggest viable solutions (e.g., rollback) to ensure service continuity.

Practitioners utilise an in-house application life cycle management (ALM) system to manage patch faults discovered during patch testing from deployment through verification. Third-party IT Service Management products and in-house solutions both enable incident management (i.e., logging and managing to-do patching requests). However, job coordination amongst teams is mostly performed manually.

7.2. Current Automation Limitations

This section describes the limits of the current automation/tools for responding to RQ2. Recognizing the weaknesses of present solutions is critical for finding areas for improvement for resolving the identified constraints, as stated in Section 5.

7.2.1. Support for Dynamic Environment Conditions is limited

While many vulnerability management solutions promise to provide automated vulnerability assessment, many fail to capture changing context elements, resulting in an erroneous risk assessment. That is why practitioners must undertake the heavy work for risk assessment while taking into account the organisational environment. Furthermore, present automation does not provide appropriate assistance for managing rapid changes in schedules, such as out-of-band (OOB) patching. While many participants want automation to help them adapt to unanticipated developments, others argue that coordination among stakeholders and decision-making in such instances rely on human intuition, which is challenging to automate.

7.2.2. Scalability issues in tool design/architecture

A key infrastructural restriction of present solutions is the lack of a consistent platform for deploying fixes to heterogeneous contexts (i.e., numerous operating systems, dependent apps, etc.). Microsoft SQL is a famous example that causes tremendous irritation for practitioners by requiring them to switch to manual deployment or juggle amongst numerous tools, resulting in frequent patch misses during rollout.

7.2.3. Inadequate Support in Process Workflows

Because of the limited support for patch deployment preparation, looking through release notes to determine patch dependencies necessitates considerable work.

Similarly, poor assistance for dealing with patch dependencies appears to be a widespread issue. Due to the lack of a comprehensive perspective of system interdependencies, practitioners must devote substantial time and effort to discovering dependencies during testing and debugging deployment difficulties. The lack of automated capabilities to address older programme dependencies complicates matters further, frequently resulting in delays in resuming services to avoid system failures. Another disadvantage is the absence of detection of the requirement for numerous reboots. Although the tools may execute scheduled reboots automatically, determining how many reboots are necessary is difficult and a manual task.

7.3. Practitioners' Need for Improved Automated processes

This segment addresses RQ3, the desires expressed by the respondents in response to the question asked "what tasks do you wish would have been better supported by the technology and how?". The question was asked to properly appreciate what practitioners truly require to bridge the gaps in current automation and satisfy their demands effectively.

7.3.1. Support for Patch Management Automation

Participants want a single platform for collecting verified patch information from many sources, including new patch releases, mid-cycle releases, patch release delays, and probable patch adverse effects. A platform like this would help them make educated judgements regarding the patch application, the extent of testing necessary, and identifying solutions for anticipated negative consequences. Furthermore, the participants would want a system that

analyses the possible impact of future patches based on patch information collected from other sources.

7.3.2. Improved System Interdependencies in Configuration Management Database

Improved Configuration Management Database with System Interdependencies. A configuration management database (CMDB) displays the configuration objects in a controlled environment (e.g., server, application, router, etc.) and how they interact with one another. "A decent CMDB will have anything you want in there, whereas at best we have a list of servers that may or may not be up to date with no relationships between them," A practitioner explains. It is something that assists you in making educated judgements, particularly when it comes to change management, whether patching or otherwise. It is a major vector into patching and the culture of how stressful it makes individuals.

7.4. The Human Role in Process Automation

This section responds

This section responds to RQ4 by outlining the crucial roles of humans in the automation of the security patch management automation process, as well as explaining why and when human input is necessary.

7.4.1. Better manage uncertain conditions in dynamic environments

One of the primary reasons for requiring human engagement in automation is changing environmental circumstances, which result in unforeseen alterations to timetables. In such circumstances, the participants reported that they switched to manual patch deployment to get more control over the problem as the unexpected incidents and the task's urgency necessitate human involvement in meticulous preparation and execution in an OOB window, which automation cannot provide.

7.4.2. Adapt to the needs and culture of the organisation

Organizational elements like as culture, policies, and needs all play a part in the necessity for human collaboration in process automation. This conclusion adds to prior research (JasonGerend, n.d.), which also highlighted the impact of the organization's internal rules and management on system administrators during the process.

Handling unfavourable security perceptions in organisational culture, such as a lack of enthusiasm, freedom, and priority for security, results in bad behaviours or decisions that negatively influence security patch management. Non-security teams, for example, tend to ignore security patching, while top management delays patch approval decisions and stakeholders refuse to cooperate. In such instances, human engagement is critical in educating individuals about the importance of security patching and establishing patch schedules that strike a compromise between the requirement to patch and system availability. Another example is the necessity to manage change opposition. Interestingly, in certain circumstances, such as patch scheduling, several participants reported being averse to switching to an automated solution since they have grown accustomed to performing it manually with Excel.

8. Conclusion and Future Work

The study's findings were examined, and their larger implications for researchers and practitioners in order to glean some practical lessons and recommend areas for future studies targeted at delivering automation solutions for security patch management.

An evidence-based understanding of the role of automation in security patch management that highlights the current state of practise automation is given, its limits, practitioners' desires for enhanced automation assistance, and the role of humans in security patch management automation. Evidence indicates that the bulk of tasks in the security patch management process

are completed manually (see Figure 3). In the process, various causes for the current situation of manual labour were discovered. The main reason for this is the limits of the present automation support, as described in Section 7.2.

This situation is the result of a variety of factors, including limitations in existing tool capabilities (e.g., capturing organisational context in vulnerability assessment), a lack of particular features (e.g., identifying prerequisites from patch release notes), performance constraints (e.g., inability to perform simultaneous deployment tasks), infrastructure constraints (e.g., inability to deploy patches to heterogeneous environments), and usability limitations (e.g., lack of meaningful error messages). These constraints highlight the areas where researchers and tool builders may provide enhanced automation/tooling help.

Furthermore, research from Sections 7.2 and 7.3 suggests that some limits of present automation systems necessitate practitioners doing certain operations manually, which typically results in patching delays. This study contends that its findings will offer a good knowledge of what automation support additions are required and how those enhancements will be used in reality. This information is also be useful in designing future work to solve practical challenges in security patch administration. The results from this study can help tool creators discover characteristics that will allow their products to offer greater value and practical utility in practise. This study acknowledges that its findings may not capture all required tool characteristics since participants may be unaware of alternative accessible tools and their capabilities. This constraint also necessitates future effort to match the functionality of existing patch management solutions to the practitioners' intended needs in order to pinpoint what is lacking and where.

Given that the findings are context-dependent, i.e., in the financial regulatory sector, more research in diverse contexts and utilising additional data sources (e.g., large-scale surveys) is required. This study's findings may be applied to other areas to uncover additional desired traits with explicit needs. It anticipates the value of future work to evaluate the tools in real-world scenarios, similar to earlier work (Dissanayake et al., 2021), to better understand how well the tool satisfies industrial demands.

9. References

Al-Ayed, A., Furnell, S.M., Zhao, D. and Dowland, P.S. (2005). An automated framework for managing security vulnerabilities. Information Management & Computer Security, 13(2), pp.156–166. doi:10.1108/09685220510589334.

Araujo, F. and Taylor, T. (2020). Improving cybersecurity hygiene through JIT patching. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. doi:10.1145/3368089.3417056.

Bozorgi, M., Saul, L., Savage, S. and Voelker, G. (2010). Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. KDD, [online] 10. Available at: https://cseweb.ucsd.edu/~saul/papers/kdd10_exploit.pdf.

Chuan-Wen Chang, Dwen-Ren Tsai and Jui-Mi Tsai (2005). A cross-site patch management model and architecture design for large scale heterogeneous environment. Proceedings 39th Annual 2005 International Carnahan Conference on Security Technology. doi:10.1109/ccst.2005.1594837.

Corbin, J. and Strauss, A. (2008). Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory. doi:10.4135/9781452230153.

Crameri, O., Knezevic, N., Kostic, D., Bianchini, R. and Zwaenepoel, W. (2007). Staged deployment in mirage, an integrated software upgrade testing and distribution system. ACM SIGOPS Operating Systems Review, 41(6), pp.221–236. doi:10.1145/1323293.1294283.

Dissanayake, N., Jayatilaka, A., Zahedi, M. and Babar, M.A. (2021). Software security patch management—A systematic literature review of challenges, approaches, tools and practices. Information and Software Technology, 144, p.106771. doi:10.1016/j.infsof.2021.106771.

Dunagan, J., Roussev, R., Daniels, B., Johnson, A., Verbowski, C. and Yi-Min Wang (n.d.). Towards a self-managing software patching process using black-box persistent-state manifests. International Conference on Autonomic Computing, 2004. Proceedings. doi:10.1109/icac.2004.1301353.

Glaser 1967. (2011). [online] Available at: http://www.sxf.uevora.pt/wp-content/uploads/2013/03/Glaser_1967.pdf.

Hai Huang, Baset, S., Chunqiang Tang, Gupta, A., Sudhan, K.N.M., Feroze, F., Garg, R. and Ravichandran, S. (2012). Patch management automation for enterprise cloud. 2012 IEEE Network Operations and Management Symposium. doi:10.1109/noms.2012.6211988.

Hicks, M., Moore, J.T. and Nettles, S. (2001). Dynamic software updating. ACM SIGPLAN Notices, [online] 36(5), pp.13–23. doi:10.1145/381694.378798.

JasonGerend (n.d.). Get started with Windows Server Update Services (WSUS). [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus.

Kamongi, P., Kotikela, S., Kavi, K., Gomathisankaran, M. and Singhal, A. (2013). VULCAN: Vulnerability Assessment Framework for Cloud Computing. 2013 IEEE 7th International Conference on Software Security and Reliability. doi:10.1109/sere.2013.31.

Kim, Y. and Won, Y. (2019). A new cost-saving and efficient method for patch management using blockchain. The Journal of Supercomputing. doi:10.1007/s11227-019-02946-y.

Li, F., Chetty, M., Rogers, L., Mathur, A. and Malkin, N. (2019). Keepers of the Machines: Examining How System Administrators Manage Software Updates This paper is included in the Proceedings of the Fifteenth Symposium on Usable Privacy and Security. Keepers of the Machines: Examining How System Administrators Manage Software Updates. [online] Available at: https://www.usenix.org/system/files/soups2019-li.pdf.

Mell, P., Bergeron, T. and Henning, D. (n.d.). Creating a Patch and Vulnerability Management Program Recommendations of the National Institute of Standards and Technology (NIST) Special Publication 800-40 Version 2.0. [online] Available at: https://csrc.nist.rip/library/alt-SP800-40v2.pdf.

Marshall, M.N. (1996). Sampling for Qualitative Research. Family Practice, [online] 13(6), pp.522–526. doi:10.1093/fampra/13.6.522.

Midtrapanon, S. and Wills, G. (2019). Linux Patch Management: With Security Assessment Features. Proceedings of the 4th International Conference on Internet of Things, Big Data and Security. doi:10.5220/0007712502700277.

Nappa, A., Johnson, R., Bilge, L., Caballero, J. and Dumitras, T. (2015). The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. [online] IEEE Xplore. doi:10.1109/SP.2015.48.

NIST (2019). NVD - Vulnerability Metrics. [online] Nist.gov. Available at: https://nvd.nist.gov/vuln-metrics/cvss.

NVivo (2021). Qualitative Data Analysis Software | NVivo. [online] www.qsrinternational.com. Available at: https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home.

Procházka, M., Wartel, R., Auth, H., Thessaloniki, Greece and Triantafyllidis, C. (n.d.). A Race for Security: Identifying Vulnerabilities on 50 000 Hosts Faster than Attackers Daniel Kouřil Christos Kanellopoulos. [online] PoS. Available at: https://pos.sissa.it/133/031/pdf.

Souppaya, M. and Scarfone, K. (2013). Guidelines for Managing the Security of Mobile Devices in the Enterprise. National Institute of Standards and Technology. [online] doi:10.6028/nist.sp.800-124r1.

Tiefenau, C., Häring, M. and Krombholz, K. (2020). Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators. [online] Available at: https://www.usenix.org/system/files/soups2020-tiefenau.pdf.

Trabelsi, S., Plate, H., Abida, A., Ben Aoun, M.M., Zouaoui, A., Missaoui, C., Gharbi, S. and Ayari, A. (2015). Mining social networks for software vulnerabilities monitoring. 2015 7th International Conference on New Technologies, Mobility and Security (NTMS). doi:10.1109/ntms.2015.7266506.

Turker, U.C., Hierons, R.M., Mousavi, M.R. and Tyukin, I.Y. (2021). Efficient state synchronisation in model-based testing through reinforcement learning. 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). doi:10.1109/ase51524.2021.9678566.

Wang, S., Wen, M., Lin, B., Wu, H., Qin, Y., Zou, D., Mao, X. and Jin, H. (2020). Automated patch correctness assessment. Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. doi:10.1145/3324884.3416590.