University of Amsterdam

SNE MSc - Research Project 1

The Pipeline of Mass Disclosure

February 2024

Students:
Karlo Drvoderić
14482959

Wouter Honselaar
15112616

Abstract

This project explores the possibility of the introduction of automation to the mass-scale Coordinated Vulnerability Disclosure (CVD) process. The research conducted by this project is inspired by the mass-scale CVD process conducted by the Dutch Institute for Vulnerability Disclosure (DIVD). This study recognizes the importance of introducing technological advances and process improvements to CVD, because as the adoption rate of new technologies increases, so does society's exposure to vulnerabilities. The current CVD process is manual, which leads to human resource limitations as organizations such as DIVD are run by volunteers. To combat such limitations, this project aims to explore the opportunities of introducing automation to the CVD process, to introduce efficiency, speed, and accuracy. To this end this study mapped out the steps, processes, and tools used by DIVD to conduct the mass-scale CVD process. Based on these findings, requirements for implementation of the automation solution were defined to streamline the development process. Finally, after a clear set of requirements has been defined, this project set out to identify the appropriate technology stack for implementation of the requirements, which culminated in the development of a proof-of-concept automation pipeline for the mass-scale CVD process. The proposed automation pipeline utilizes Apache Airflow to streamline the scanning and notification stages of the CVD process, introducing potential gains in efficiency and speed of the CVD process.

Keywords: Coordinated Vulnerability Disclosure, Automated Vulnerability Disclosure, Cybersecurity Process Optimization, Vulnerability Management, CVD Process Enhancement, Apache Airflow Automation

`The project repository can be found at the following link:`
`https://github.com/karlokarlo/cvd_automation_pipeline`

1 Introduction

As technology becomes more widely used in society, it is becoming an irreplaceable part of people's personal and professional lives. As a result of the increasing adoption rate of new technology, the exposure to various information security threats also grows as malicious actors are presented with a larger attack surface for various kinds of cyber attacks [9]. Cyber attacks often result in negative consequences for the victims, such as loss of private data, reputational damage to individuals and organizations, financial loss or even posing threats to national security [33][14][8]. This is why cyber security research is increasingly focused on the prevention of such attacks before the damage is done [13]. To help guide this effort, the Coordinated Vulnerability Disclosure (CVD) process has been developed.

CVD is a process that aims to spread awareness of newly identified vulnerabilities while giving developers sufficient time to address the discovered vulnerabilities and release relevant security patches [16]. It is important to give the developers time to correct the found vulnerabilities before making them public to prevent malicious actors from exploiting the discovered vulnerability before a security patch is available. Another reason is to prevent developers from rushing, as that might introduce new vulnerabilities due to an increased chance of mistakes. Once the responsible organization has had enough time to develop a security patch, the CVD process concludes by publicly disclosing the information surrounding the found vulnerability, raising awareness around the found vulnerability, and encouraging affected users to apply the security patch [27]. By conducting the vulnerability disclosure in such a manner, the responsible organizations are allowed to correct their mistakes while the users are provided with the necessary transparency and safety.

Dutch Institute for Vulnerability Disclosure (DIVD)[1] is an organization that participates in the CVD process at a mass scale. This means that on top of conducting vulnerability research and notifying the responsible organizations of the found results, DIVD aims to spread knowledge about the vulnerabilities to the affected parties by encouraging them to take preventative measures. DIVD mostly participates in the later stages of the CVD process, focusing on timely notification of the vulnerable hosts after the vulnerabilities have been made public. By performing the CVD process at a mass scale, DIVD is capable of raising awareness surrounding dozens of vulnerabilities by notifying hundreds of thousands of affected IPs every year [7]. However, as DIVD is a volunteer organization that consists of cybersecurity professionals, it experiences certain limitations in terms of personnel available to fully follow through on the CVD process. The researchers involved in the CVD process must carefully prioritize between conducting research surrounding a specific vulnerability and coordinating communication between responsible and affected parties.

The primary goal of this research is to identify the opportunities to automate the parts of the CVD process that focus on vulnerability scanning and notification of the affected targets, allowing researchers to give more time and focus to the analysis of the vulnerabilities themselves. This research also aims to provide insight into how DIVD conducts the mass-scale CVD process, to encourage discussion surrounding the CVD process, and to lower the barrier to entry for other cybersecurity organizations that may also wish to participate in CVD and make the Internet a safer place.

2 Related Work

The research performed by [31] identified the challenges researchers face when conducting the CVD process at a mass scale. These include handling large volumes of data in scanning and email notification phases, finding the correct contact information of the vulnerable parties, identifying new communication channels, and encouraging people responsible for vulnerable hosts to take action in mitigating the found vulnerabilities. The paper suggests dealing with the identified challenges through the standardization of practices and procedures undertaken during the CVD process and proposes an architecture design for an automation system to streamline the scanning and communication part of the CVD process. The findings presented in this paper serve as a foundation for this research, which aims to build on the suggested strategies for improvement of how the CVD process is conducted by creating a proof-of-concept system for its automation.

The work done by [32] investigates the policies and rules various organizations share with CVD researchers when inviting them to participate in the CVD process. The research identified the lack of a clear definition of policies and legal constraints surrounding the process researchers were engaging in. This can leave the researchers questioning the legality of the steps involved in the CVD process and leaves them unsure of the extent to which they can probe the system to identify vulnerabilities. Our research aims to reduce this gap by providing insight into how the mass-scale CVD process is conducted at DIVD and to provide standardization as to how the CVD process is conducted by introducing automation of the recognized practices.

[1] `https://www.divd.nl/`

The paper [5] investigates zero-day attacks and the behavior of malicious actors before and after a vulnerability has been disclosed. Their research has found that the use of zero-day attacks increases by five times after the vulnerability has been publicly disclosed. This finding signifies the importance of contacting vulnerable targets as a part of the CVD process, as once a vulnerability is made public, it is crucial to raise awareness and apply relevant security patches. The automation pipeline proposed by our research aims to automate the notification part of the CVD process, therefore simplifying the distribution of information to vulnerable parties.

A paper [17] has been identified that attempts to automate and improve the responsible vulnerability disclosure process. This research explores the use of distributed ledger technology for a new approach to vulnerability disclosure called Automated Responsible Disclosure. A problem recognized in this work is the tendency of vendors to intimidate and work against researchers who disclose vulnerabilities they have found, as associating their brand with vulnerable systems could cause reputational damage. The main aim of the proposed automation system is to protect the identity of the researchers responsible for disclosing vulnerabilities while avoiding unnecessarily long delays before the vulnerability found can be responsibly disclosed. While the automation pipeline our research attempts to design focuses on automating the traditional CVD, it is valuable to note that technological advances are having an impact on refining the efficiency and security provided by the current CVD process.

The research done by [23] utilizes natural language processing and machine learning to automate the vulnerability characterization process. The proof of concept system that was developed obtains CVE information from multiple sources. With this information, it is capable of detecting the source of an attack, assessing the context of the vulnerability, examining the methods used in the exploits, considering the potential impact of the vulnerabilities, and suggesting mitigation strategies. The research showed that using the developed system instead of manual work resulted in a significant decrease in the time required to categorize the vulnerabilities.

3 Problem statement

There are other organizations such as DIVD attempt to make the Internet a safer place [30]. As the work these organizations face becoming increasingly challenging with the advancement of cyber threats, the amount of work that their activities require increases proportionally. However, due to the volunteering nature of such organizations, cyber security professionals must often balance their professional and volunteering responsibilities, resulting in limited time to fully commit to the CVD process. This means that such organizations often face limitations in terms of human resources available to them, and must reduce the amount of time spent on researching vulnerabilities in favor of organizing data and coordinating communication with the vulnerable parties. Furthermore, the researchers involved in the CVD process must ensure that there are no false positives in the hosts determined to be vulnerable, as contacting them would be a waste of resources, and could cause reputational damage to the organization conducting CVD. Another key factor to consider is that CVD is a time-sensitive process, as once a vulnerability is made public, it draws the attention of threat actors, increasing the likelihood of exploitation. For all these reasons it is important that CVD is conducted efficiently and consistently, and that the vulnerable parties are contacted as soon as vulnerability data is available.

One of the ways to increase the efficiency and consistency of the CVD process is by introducing automation to the parts of the CVD process that are repetitive and can be streamlined [31]. Introducing automation could help such organizations process more data in shorter periods, while at the same time allowing researchers to put more focus on the analytical parts of the CVD process. Automation in data processing could also mitigate the number of mistakes caused by manual work by introducing consistency in the performed tasks. Finally, automation of the CVD process will result in faster notification of vulnerable hosts, reducing the time malicious actors have to exploit the vulnerable targets.

3.1 Research Questions

The objective of this research is to identify opportunities to automate the CVD process to streamline the more repetitive and predictable parts of the process. In doing so, this research aims to increase the speed and consistency with which organizations conducting the CVD process operate by proposing a proof-of-concept of an automation pipeline for the vulnerability scanning and email notification part of the CVD process. With this in mind, the following main research question has been defined:

"To what extent can a mass-scale coordinated vulnerability disclosure process be automated?"

To implement an automation pipeline for the CVD process, it is first necessary to understand what the CVD process consists of. This research aims to answer this question by exploring the steps and processes involved in CVD such as those used by DIVD. Therefore, the following sub-research question has been defined:

SRQ1: "What are the key components and processes of mass-scale Coordinated Vulnerability Disclosure as implemented by the Dutch Institute for Vulnerability Disclosure?"

Once the steps and processes of the mass-scale CVD are clearly defined, it becomes more feasible to determine what an automation pipeline for the mass-scale CVD process may require. To clearly define the requirements for an automation solution, the following sub-research question has been defined:

SRQ2: "What are the requirements for automation of the mass-scale coordinated vulnerability disclosure process?"

Finally, once the requirements for automation of the mass-scale CVD process are known, this research aims to explore the possibilities to design and implement such a solution. This should result in a proof-of-concept automation pipeline for the mass-scale CVD process. This process should lead to an answer to the final sub-research question:

SRQ3: "How can a solution for the automation of the mass-scale coordinated vulnerability disclosure be realized?"

3.2 Ethical Considerations

What is considered acceptable in cybersecurity practices always varies between global standards and the regulations of the country in which the vulnerability assessment is performed. Active scanning for vulnerability disclosure is a sensitive issue in many countries. However, the Dutch government recognizes the importance of such activities and is more lenient towards vulnerability scanning when it is done to inform vulnerable parties about vulnerabilities and encourage them to apply security patches [25][26]. As DIVD is based in the Netherlands, it follows Dutch law in determining what vulnerability disclosure practices are considered acceptable. While Dutch law does not explicitly define what is allowed in terms of vulnerability scanning and disclosure, the guidelines that DIVD (and by extension this project) follows are based on criminal law and jurisprudence that define the boundaries of what is considered acceptable. Accordingly, DIVD maintains a public Code of Conduct document [6] that defines the organization's mission statement and details how a researcher conducting a CVD process should behave. The Code of Conduct states that all vulnerability scans conducted by DIVD will be done for vulnerability disclosure and protection of vulnerable hosts. In addition, any data collected by DIVD will be stored in a GDPR-compliant manner, ensuring fair use, appropriate storage, and minimal retention of personal data.

The implications of publicly sharing the details of the mass-scale CVD process and the developed automation pipeline solution were also considered. It was concluded that as the main contribution of the pipeline is in chaining the publicly available tools and utilizing them for timely notification of vulnerable hosts, the proposed proof-of-concept does not allow threat actors to do anything they could not have done had the structure of the CVD process and the automation pipeline solution remained private. Furthermore, to effectively use the pipeline, it is necessary to define an effective vulnerability fingerprint, which is a challenging task and requires expert knowledge from the person operating the pipeline. This is why it was concluded that the findings are beneficial for furthering the understanding of how mass-scale CVD process is conducted and potential in its automation, while not giving any additional benefits and advantages to threat actors.

4 Methodology

This section aims to provide an overview of the methodological framework used to design and implement the proposed CVD automation pipeline. The following subsections describe the steps taken in this study, starting from understanding what the mass-scale CVD process consists of, to defining the requirements for its automation and implementing a proof-of-concept solution.

4.1 Analysis of the DIVD Vulnerability Disclosure Case Guide

To find an optimal solution for automation of the CVD process, it is necessary to understand what the manual CVD process consists of. To this end, this study aims to investigate and map out the workflow through which CVD conducts vulnerability scanning and notification of vulnerable parties. To gain an overview of the step-by-step CVD workflow as used by DIVD, a procedural (operational) analysis was conducted on DIVD's case manual. Through this document, DIVD aims to guide its researchers through the CVD process, from opening the case and defining the initial scan surface to notifying the parties responsible for vulnerability patching and closing the case. The identified process was used as the foundation for the proposed automation solution, as by having an overview of what the CVD process consists of, it was possible to recognize which areas of the process allow for automation.

To perform the procedural analysis in a structured manner, a note was taken of each step that the researcher needs to take when conducting vulnerability scanning and notification aspects of the CVD process. In addition to analyzing the steps a researcher is guided through, a detailed record was made of the tools that were used during each step, including the specific commands and command parameters. This analysis aims to identify the tasks and tools used in the CVD process with the end goal of enabling the reader to reproduce the mass-scale CVD process as it is conducted by DIVD.

4.2 Requirements Engineering and Interviews

Requirement engineering is a process of collecting and analyzing all information relevant to the system being built to accurately define the needs that the desired system needs to fulfill [4]. The requirements engineering process for this research was based on the detailed procedural analysis of the DIVD case guide described in the previous chapter. Analysis of this document allowed for the recognition of the key tools, dependencies, and outcomes involved in each step of the CVD process. As a result, this served as a strong foundation for defining how an automation solution for the CVD process should behave and what qualities it should exhibit.

To build upon the insights gained from the case guide, a series of interviews were organized with the DIVD members. The goal of these interviews was to refine the findings from the procedural analysis step and to gain insights from people practically involved in the CVD process. The interviewers were asked several predefined questions, however, the interviews were not strictly kept in a Q&A format and served as an opportunity for brainstorming and the free exchange of ideas. The set of questions that the researchers were asked can be found in the appendix A. The resulting potential requirements were noted down and subsequently expanded on through the detailed comparison with the findings of the first step.

The practical insights of the DIVD members helped complete the full picture of what the CVD process consists of and what its automation may require. This process resulted in a set of requirements that reflected both the documented procedures and the practical insights, making the requirements relevant, specific, and realistic. The defined requirements were separated into functional and non-functional requirements to make a clear distinction between what capabilities and qualities a mass-scale CVD automation pipeline should have. Furthermore, the defined requirements were systematically categorized according to the Must, Should, Could, or Will Not

(MoSCoW) requirements prioritization structure [3]. This prioritization aims to identify which requirements are the most critical for the operation of the automation pipeline while keeping in mind the opportunities for optimization of the pipeline's performance.

4.3 Choice of Technology Stack and Development

With the defined requirements this project aimed to develop a proof-of-concept for the automation pipeline of the mass-scale CVD process. To achieve this, it was necessary to identify the appropriate technology stack through which the defined requirements could be implemented. The identification of appropriate technology was conducted through a brief literature review covering literature exploring the automation of complex workflows. Although a detailed investigation of the available workflow automation frameworks was beyond the scope of this project, the review of relevant literature narrowed down the frameworks through which automation could be achieved. In addition to selecting a workflow automation framework that best met the needs of this project, the ease of use and flexibility of the technology was also considered, as the ultimate goal of this project was to develop a functional proof of concept rather than a finished and optimized solution. Once the appropriate technology stack was identified, it was used to translate the previously defined requirements into a working proof-of-concept of the automation pipeline. The system developed aims to imitate the manual CVD process as closely as possible while fulfilling all of the requirements identified by the previous step of the research.

5 Results

5.1 Analysis of the DIVD Vulnerability Disclosure Case Guide

The procedural analysis of the DIVD case manual revealed 8 primary steps for replicating the mass-scale CVD workflow as conducted by DIVD. During the procedural analysis, a record was taken of the steps taken, tools used, and the reasoning behind the actions that a researcher should take. The identified steps are as follows:

1. **Opening the Case** - The researcher must first open a case on the shared ticketing system that corresponds to a particular CVE. While this step is crucial for assigning tasks, progress tracking, and accountability, this is a step related to DIVD's internal case management and is not necessary for the reproduction of the mass-scale CVD process.

2. **Defining the scan surface** - The next step a DIVD researcher takes is related to creating a list of the devices that could potentially be vulnerable to the specified vulnerability. This is done by identifying the type of devices that a vulnerability targets, and by creating a list of such devices available on the public IPv4 address space. This is done through open-source intelligence (OSINT) tools and platforms that can scan the entire IPv4 space, such as Shodan and Censys. In case clear search settings can not be defined for creating a subset of potentially vulnerable devices, researchers are given the option to scan the entire IPv4 range (0.0.0.0/0). However, researchers are discouraged from doing so as such scans are resource-intensive. The primary platform that DIVD uses for defining potentially vulnerable hosts is Shodan. Shodan provides a search engine where hosts can be filtered based on their characteristics, such as the type of software that is running on the host. This allows for a narrowing down of the initial scan surface, making the vulnerability scanning process more efficient. To initiate a Shodan query, DIVD uses Shodan CLI by executing the following command: `"shodan download --limit -1 filename 'query'"`. This command creates a list of the devices matching a predefined query, while the –limit flag set to -1 ensures that all publicly known devices are included in the list. In case a researcher would like to obtain the initial scan surface list from additional tools, they are recommended to use the tool called Uncover[2]. Uncover server as a wrapper for IPv4 space scanning tools such as Shodan[3] and Censys[4], and allows for concurrently running a query

[2]https://github.com/projectdiscovery/uncover
[3]https://www.shodan.io/
[4]https://search.censys.io/

on multiple platforms. This can result in a more complete list of potentially vulnerable devices, as different platforms may return different results. An example command a researcher would run when performing an Uncover search is the following: `"uncover -cs 'http.title:"Log In - Juniper Web Device Manager"' -l 9999999 >shodan-results.txt"` This command searches through the Censys database and creates a list of Juniper devices, which may be particularly useful when conducting a CVD process on a vulnerability targeting Juniper devices.

3. **Defining the Vulnerability Fingerprint** - Once the initial scan surface is clearly defined, the researcher must use the information obtained to create a vulnerability scanning template for a mass vulnerability scanning tool such as Nuclei. [5]. This is normally done through information gathered from public exploits, publicly available version numbers of services running on the target system, entity tags in an HTTP header, or dependency hashing. The process of defining the vulnerability fingerprint is unique to each vulnerability and cannot be easily generalized [29]. The vulnerability fingerprint contains the set of behaviors that are used to probe the potentially vulnerable device and the set of responses that are to be expected from a vulnerable device. When defining a vulnerability fingerprint it is crucial to distinguish between the behavior that a vulnerable and non-vulnerable device would display, to avoid false positives when identifying vulnerable devices.

4. **Performing the Vulnerability Scan** - The previously defined Nuclei template is used in combination with the list of devices identified in step 2 to determine which of these devices are susceptible to the vulnerability. An example of initiating a Nuclei vulnerability scan can be seen in the following command: `"nuclei -H "User-Agent:  DIVD-2023-00020" -t ./divd-2023-00009.yaml -l date-targetlist.txt -retries 2 -timeout 6 -output date-results.json -j"`. This command executes a vulnerability scan by using a previously defined vulnerability fingerprint file referenced by the *-t* flag of the command. Furthermore, DIVD attempts to differentiate itself from other traffic by setting a "User-Agent" header field to the DIVD case number referencing the particular vulnerability that is being investigated. Finally, the results of this step are stored in JSON format by the inclusion of the *"-j"* flag to support the tools used in the subsequent steps of the CVD process.

5. **Enriching Scan Results** - To obtain the contact information of the vulnerable hosts DIVD uses an open source tool called nuclei-parse-enrich [6]. This tool takes Nuclei vulnerability scan results in JSON format and parses them into a JSON object containing relevant vulnerability data. At the same time as parsing takes place, the tool also attempts to retrieve contact information of vulnerable targets. This is done by the tool first attempting to retrieve the geolocation and abuse information from RipeStat, and falling back to a Whois query if no useful contact information was returned by RipeStat. According to the DIVD case manual, the current success rate for retrieving contact information with this tool is 95%. The success rate of email retrieval should however be differentiated from obtaining the correct contact information, as DIVD currently reports a 42% false positive ratio on WHOIS queries, that has been increasing since the implementation of the GDPR. In case contact information is not found for all IP addresses, DIVD researchers are instructed to manually attempt to find the remaining contact information through traditional search engines such as Google.

6. **Sending Vulnerability Notification Emails** - After contact information for vulnerable hosts has been obtained, it is necessary to notify the hosts about the discovered vulnerability. For this purpose, DIVD uses a tool called Mailmerge[7], which connects to the DIVD SMTP server and sends out vulnerability notification emails. The tool uses a predefined email template which is filled out with the vulnerability information unique for each host that is being contacted.

[5]https://github.com/projectdiscovery/nuclei
[6]https://github.com/DIVD-NL/nuclei-parse-enrich
[7]https://github.com/awdeorio/mailmerge

7. **Sharing Data with the Government CSIRT Partners** - After the vulnerability notifications have been sent, DIVD researchers have an option to separate the vulnerability data of the vulnerable hosts based on the country a vulnerable IP address belongs to. Then, the researcher is tasked with informing an organization that is responsible for the mitigation of vulnerabilities within that country.

8. **Closing the case** - After several mail runs, the researcher responsible for the case may decide to close the case. To close the case, the researcher will create a separate GitHub branch and create a pull request with the updated case file. In the updated case file, the case status will be set to close, the information about the final number of reported IPs will be entered, and in the event of the DIVD closing the case will be added to the case timeline.

5.2 Definition of Requirements

Functional requirements reflect the capabilities that the developed system must have by clearly defining the desired system functionality [18]. By having a clear definition of what the end product should be capable of, these requirements help guide the development and design of the final solution. A clear definition of functional requirements also helps in establishing clear expectations for stakeholders and easily verifying the capabilities of the developed solution [15]. The functional requirements defined for a pipeline automating the mass-scale CVD process are as follows:

1. **Initial Scan Surface Definition:** *"The initial scan surface shall be determined by utilizing IPv4 space scanning tools such as Shodan or Censys. The system shall integrate with at least one of these tools to define the scan surface."* - The automation pipeline must first create a list of devices that were identified as the type of devices that could be vulnerable to the specific vulnerability. To create such a list, IPv4 space scanning tools such as Shodan or Censys should be utilized. This list serves as a basis for further vulnerability scanning and notification efforts and is therefore categorized as a MUST have requirement.

2. **Vulnerability Scanning:** *The system shall implement a vulnerability scanning tool that can operate on mass-scale, to identify vulnerable devices among the predefined scan surface.* - The automation pipeline MUST implement a tool (such as Nuclei) that is capable of scanning a large number of devices for a specific vulnerability. The tool must be able to identify vulnerable devices based on the predetermined behaviour, and must not be more intrusive than what the laws of the country under which the CVD operation is conducted allow. The final result of this process should culminate in a list of devices that were determined to be vulnerable, containing at least their IP and the description of behaviour by which the device was determined to be vulnerable.

3. **Standardized Vulnerability Template:** *The system shall use a standardized YAML vulnerability template for conducting Nuclei scans, ensuring consistency with the predefined scan criteria.* - On top of containing the necessary information for recognizing vulnerable devices, a vulnerability fingerprint file SHOULD contain other information relevant to the vulnerability, such as the number of its CVE record, vulnerability severity, and references for related information, to use this data in the notification stages of the CVD process. This ensures consistency in scanning and clarity in communication with the affected targets.

4. **Fingerprinting:** *The system shall automatically fetch the Nuclei template from a predetermined source and perform fingerprinting processes.* - To maximize the automation of the CVD process, the system SHOULD have a predetermined location from which it will retrieve a fingerprint vulnerability template that will be used for Nuclei scan. By knowing where to look for the fingerprint file, the system will not need any manual input during its operation, aiding it in uninterrupted performance.

5. **Data Enrichment:** *Once the Nuclei scan produces results, the system shall enrich the gathered data with additional contact information obtained from RIPEstat, WHOIS, or*

security.txt file. - To fully follow through on the CVD process, it is mandatory to have a way of reaching out to the parties affected by the vulnerability. The proposed automation system MUST implement a method of obtaining contact information of vulnerable parties, either through internet information databases such as RIPEstat or WHOIS or by checking the presence of a security.txt file for a particular host.

6. **Handling Unresolved Contacts:** *IP addresses for which contact information cannot be found shall be stored separately for manual processing and review.* - During the enrichment process, not all vulnerable devices will have publicly available contact information. In such an event, a researcher responsible for the CVD process COULD be made aware of these cases and be given the option to follow through with a manual investigation. This gives the researcher and organization conducting the CVD process an opportunity to maximize the number of contacted vulnerable parties even when the traditional methods of obtaining contact information fail.

7. **Data Security:** *Confidential and non-confidential scan data shall be securely stored and automatically destroyed after it is no longer needed.* - The data obtained by the automation pipeline is to be considered sensitive as it contains information that could be exploited by malicious actors. The automation system MUST be designed in a way that minimizes access to and retention of such data. Therefore, the system must safely remove the data containing sensitive information once the data is no longer needed for the uninterrupted execution of the CVD process.

8. **Degree of Manual Control:** *The scanning process shall be started manually by the user while maintaining the ability to stop the process at any time.* - The researcher responsible for the CVD process MUST have full control over the execution of the automation pipeline. The researcher is the one to decide when the pipeline is to be started and must be able to stop the execution of the pipeline at any point for any reason. This provides the researcher with full control over the pipeline and prevents unintentional actions from being executed by the automated system.

9. **Human Confirmation for Email Notification:** *After the completion of the scanning process, the system shall wait for human confirmation before starting the emailing process.* - As the mass-scale CVD automation system is designed to contact large quantities of vulnerable hosts, it is necessary to have an overview and insight into what data gets sent out on the behalf of the organization conducting the CVD process. As sending out correct information is crucial for the reputation of such an organization, the researchers involved in the CVD process MUST have an option to manually validate the correctness of the data that is to be communicated with the vulnerable parties. This is why a system that automates a CVD process must implement an option for researchers to manually confirm the vulnerability scan results before continuing to the notification stage of the CVD process.

10. **Notifications:** *Automatic notifications shall be sent by email to the identified contact information of the target that is found vulnerable.* - The CVD automation pipeline MUST automate the way through which the vulnerable IP addresses are contacted. As the mass-scale CVD process identifies large quantities of vulnerable devices, the vulnerability notifications must be sent out in an automated manner. This ensures consistency in the type of information that is shared, removes the possibility of mistakes introduced by repetitive manual work, and ultimately speeds up the notification process.

11. **Notification Standardization:** *Notifications shall follow a standardized template to ensure consistent and clear communication.* - The information communicated to vulnerable parties SHOULD be formatted using a standardized template. Following a standardized template ensures that the vulnerability information is formatted in a clear way when it is being presented to the vulnerable party. Furthermore, if the template used to construct the message is well thought out, it can be perceived as a message coming from a trusted source, ultimately increasing the chances of the vulnerable party taking the vulnerability seriously and following the vulnerability remediation suggestions.

12. **Status Accessibility:** *Researchers conducting the CVD process shall have an overview of the real-time status of the scan processes.* - The developed solution for the automation of the CVD process should not behave as a black box. The researcher responsible for conducting the CVD process must have an insight into the current state of executing automated tasks at all times. Therefore, the automated CVD pipeline MUST implement a way for the researchers to have an oversight of the current state of the task execution within the pipeline so that they can make informed decisions on how to proceed with the CVD process.

The non-functional requirements are used to describe the performance goals of the developed system [10]. These requirements focus on the quality aspects of the developed system, such as reliability, usability, maintainability, and security which the system should aim towards [11]. The defined non-functional requirements that a CVD automation pipeline should follow are as follows:

1. **Scalability:** *The system shall support concurrent execution of multiple CVD cases without a drop in performance.* - The organization conducting the CVD process SHOULD have the ability to conduct multiple vulnerability scanning and emailing processes at the same time. The developed automation pipeline should be easily scalable so that it can support the necessary amount of workload that the organization requires.

2. **Availability:** *The system shall be operational 24/7, with a maximum allowed downtime of 1 week per year for maintenance.* - The developed automation pipeline COULD be available to the researchers at their convenience, without unintended interruptions to their work. The pipeline's reliability should be conducive to the system upon which it runs, as opposed to the underlying mechanism of the pipeline itself.

3. **Security:** *The system shall implement a secure authentication mechanism through a user-friendly login interface.* - As the developed system will be handling potentially sensitive data, it is crucial that the access to this data is limited, so that only the researchers working on a particular case have access to it. Therefore, the system MUST implement some form of an authentication and access control mechanism that can dictate who can access the data gathered by the automation pipeline.

4. **Modularity:** *The system shall support the addition, modification, or removal of its functions and capabilities.* - The needs of a CVD process may vary from case to case, and the developed system SHOULD be able to be adjusted to those needs. For this reason, the automation pipeline should be designed so that functionality can be added, removed, or adjusted within a reasonable time frame.

5. **Usability:** *Comprehensive documentation and training materials shall be provided to ensure the system is practically usable by the organization performing the CVD process.* - In addition to the developed system to behave as expected, it is important for the researchers using the automation pipeline to be familiar with its functionality and operation. For this reason, the developed system SHOULD come with a user manual that will guide the researcher through the functionality of the developed system and its adequate operation.

6. **Interoperability:** *The system shall be compatible with the tools used by the current CVD process, ensuring easy data exchange and functionality.* - As the proof-of-concept that is being developed by this project will be further expanded on by DIVD, the developed system SHOULD be compatible with the tools and processes that DIVD implements in the manual CVD process. This will ensure a smooth integration of the proposed proof-of-concept into the already existing CVD workflow, allowing for immediate assessment of the developed pipeline's functionality.

7. **Auditability:** *The system shall log all user actions and system events which shall be stored for at least one year.* - The developed system MUST log the actions it performs together with the user responsible for executing those actions. Keeping a detailed note of the events

that happen within the automation system is important for timely error detection, tracking the status of the execution of the CVD process, and accountability of the researchers operating the pipeline.

The defined requirements served as the basis when considering the choice of the technology stack for the proof of concept of the CVD automation solution. The following chapter explores the rationale for identifying the solution that could best support all of the defined requirements.

5.3 Workflow Management

Multiple solutions were compared with each other. The most universal solution should be to write a shell script that runs the tasks individually, passing the output as inputs to the following tasks. However, according to the requirements, the system should be scalable and managed in such a way that independent modules can be used. Logging is also important. While strictly speaking a shell script should suffice, a more robust system was proposed as this means parts of well-known mature current systems can be re-used to fulfill the requirements, preferably in a way that implementing requirements like access control and logging is more convenient with less customization than a shell script.

The compared solutions are Ansible and Apache Airflow. Airflow is considered the best-suited solution, as it directly implements a pipeline. It is a Workflow Management System (WMS), contrary to automation solutions that focus on Infrastructure as Code (IaC). Ansible is such an IaC solution.

Inspiration to take Airflow into consideration is gained after reading papers. Ansible works with Playbooks and corresponding tasks with the possibility to use variables. It is idempotent and declarative and aimed at setting up configurations of infrastructure, while Airflow is aimed at using tasks structured and re-schedules in a certain manner.

As the system should consist of a pipeline, desk research was conducted to search for existing automated pipeline solutions. A pipeline is defined as a system that consists of multiple dependent steps, but they can be run independently, multiple times, and with different variables.

Table 1 shows the feature-by-feature comparison between Apache Airflow and Ansible, two potential solutions. It shows the more suitable solution according to key points, not whether a solution supports a functionality or goal. Due to the requirements, Apache Airflow is a better solution than Ansible as it is suited to perform a set of repetitive tasks in a logical order. Ansible is aimed on automated configuration management, which is not important in this case.

Comparison	Apache Airflow	Ansible
Type of solution	WMS	IaC
Automation	✗	✓
Infrastructure Config.	✗	✓
Flexibility	✓	✗
Use Cases	Repetitive tasks	System configuration
Extensibility	✓	✓
Monitoring	✓	✗
Logging	✓	✗
Scalability	✓	✓
Modularity	✓	✓
Community Support	✓	✓
Learning Curve	✗	✓
Language Support	✓	✗
External Integration	✓	✗
Configuration Management	✗	✓
Agentless	✓	✓
State Management	✓	✗
Orchestration	✓	✗

Table 1: Comparison between Ansible and Apache Airflow. Checkmarks indicate either better suitability or the functionality being absent in the alternative

Once the requirements have been defined, it was necessary to identify the appropriate technology stack that can enable their implementation. Interviews with DIVD staff revealed that DIVD had previously attempted to design and implement a CVD automation pipeline in the programming language Go. However, due to a lack of human resources and the complexity of the proposed solution, the final automation pipeline was never realized. Therefore, due to the complex nature of the requirements and time limitations of this project, an effort was made to determine whether an already existing WMS could fulfill the defined requirements. WMS are tools and frameworks that help streamline business processes by simplifying their implementation and subsequent management [20].

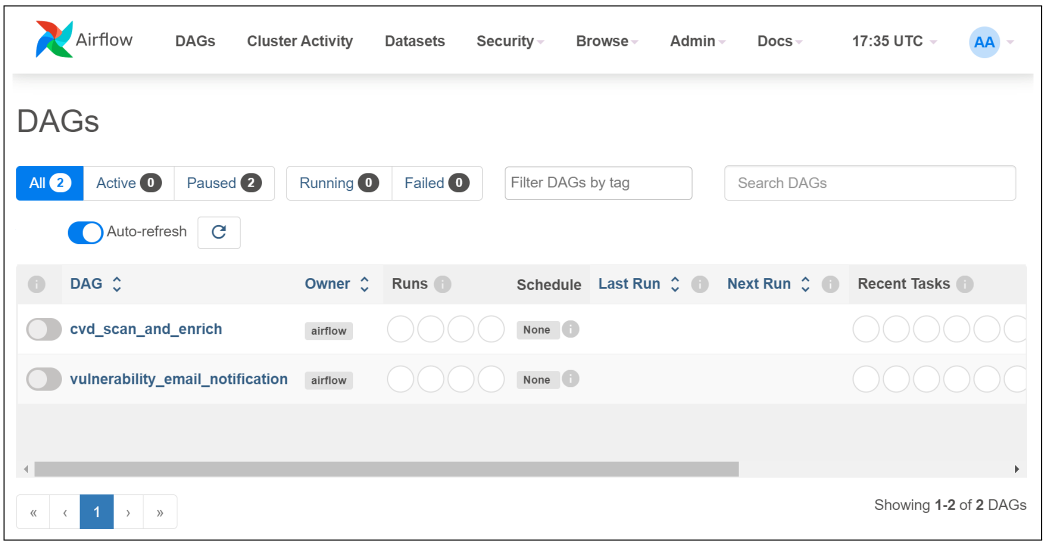
During the search for the most appropriate WMS for the CVD automation pipeline, a paper [19] was identified covering different aspects of four WMSs, namely Pegasus, Makeflow, Apache Airflow, and Pachyderm. The paper compares the four WMS solutions based on their usability, ease of deployment, performance, and relevance. This paper was used as the basis for the choice of the WMS for automation of the CVD pipeline.

Through the analysis of [19], Apache Airflow was chosen as the most fitting solution for the task of building a CVD automation pipeline for the following reasons: Firstly, Apache Airflow offers capabilities of advanced workflow management through the use of Directed Acyclic Graphs (DAGs), which can allow for flexibility and precision when replicating the CVD process as identified by the defined requirements. [21] Furthermore, Airflow offers high modularity by introducing hooks and operators through which it is capable of integration with multiple tools that a scanning entity might be using, such as various data storage solutions and scanning, data enriching, and emailing tools. [21] Finally, Airflow offers advanced error management, rescheduling, and real-time monitoring of ongoing tasks, which may be a crucial feature to have when working on a task-sensitive process such as CVD [21].

When analyzing other WMS solutions, it was recognized that while Makeflow features simplicity and ease of use, it lacks features that are crucial for automating the CVD process. Some of these features include support for integration with various tools, advanced error handling, and real-time monitoring of the execution. Pachyderm is a WMS solution that heavily focuses on data versioning and data lineage, which are features that were not identified as something of high importance when automating the CVD process. Furthermore, Pachyderm features containerization and Kubernetes integration, which are also features that are not identified as crucial for the CVD pipeline and may unnecessarily increase the final solution's complexity. At the same time, Apache Airflow outperforms Pachyderm in areas more relevant to the CVD process, such as detailed task scheduling and real-time monitoring.

5.3.1 Apache Airflow

Apache Airflow works by implementing complex workflows in the form of DAGs. Each node within a DAG represents a task that is to be executed by Apache Airflow, while every edge between the tasks dictates the dependencies between the tasks [28]. The edges that a DAG consists of must always be directed, and the complete DAG must not introduce any loops between the tasks that it implements. The tasks that Apache Airflow executes are implemented through Operators, which serve as a template for the logic that the task is supposed to execute. There are three different types of Airflow Operators, namely BashOperator, PythonOperator and EmailOperator [1]. As writing functionality through Python scripts offers the most flexibility and specificity by accessing the full power of Python programming language, this is the operator through which tasks executed by the CVD automation pipeline were implemented. Furthermore, a significant feature of Apache Airflow is its implementation of the Airflow UI Dashboard. A working instance of Apache Airflow can be accessed through a web server, which provides access to a graphical user interface providing a clear overview of the existing DAGs and their current execution status. Airflow functionality is accessed via a login form, providing access control, a comprehensive view of recorded logs, and the ability to fine-tune the execution process. An example view of Airflow UI can be seen in figure 1.



DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
cvd_scan_and_enrich	airflow		None			
vulnerability_email_notification	airflow		None			

Showing 1-2 of 2 DAGs

Figure 1: Airflow UI with the Implemented DAGs

5.4 Pipeline Design

The proposed solution for the CVD automation pipeline was built by developing two separate DAGs. The first DAG is responsible for defining the scan surface, identifying the vulnerable devices, and finding their corresponding contact information. The second DAG is used to fill in the email template used to notify the targets about the found vulnerability, send the emails, and perform a scheduled re-scan of the vulnerable addresses one month after the email is sent. If the address is found vulnerable once again after one month, a follow-up email is sent to encourage the target to patch the vulnerability. The reason for the separation of the two DAGs is the need for manual control of the targets being contacted, as one of the defined requirements is the need to minimize false positives in vulnerability scanning. This is done by researchers manually performing statistical analysis of the found vulnerable targets, confirming that they are indeed vulnerable.

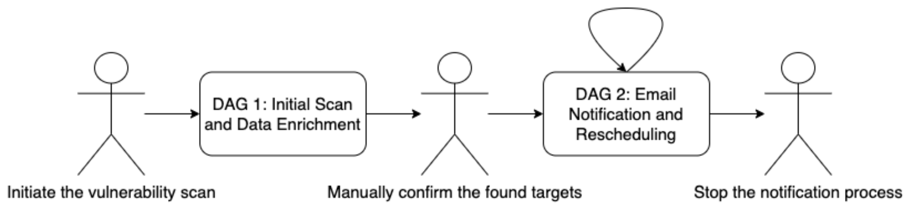

DAG 1: Initial Scan and Data Enrichment
DAG 2: Email Notification and Rescheduling
Initiate the vulnerability scan
Manually confirm the found targets
Stop the notification process

Figure 2: Overview of the Automated CVD Pipeline with Points of Manual Intervention

The first DAG focuses on identifying the vulnerable targets and finding their contact information. It consists of three separate tasks, all of which use Python operators to perform their functions. The execution of each task is dependent on the successful completion of the previous task. The tasks that make up the DAG "Initial Scan and Data Enrichment" are as follows:

1. **Define Scan Surface:** This task is responsible for the definition of the initial scan surface. It uses the Shodan CLI tool to identify IP addresses of the devices that may be susceptible to a particular vulnerability based on a specific search phrase. For this task to work as intended, it is also necessary to provide a valid Shodan API key by defining it as one of the Airflow Variables. Airflow Variables are stored in an encrypted format which ensures secure handling of the provided API key [2]. A capability for integration with the Uncover tool has also been implemented in case the scanning entity has valid API keys for more than one IPv4 space scanning tool.
2. **Nuclei Scan:** This task initiates a Nuclei scan to assess the vulnerability of the targets defined in the previous task. It takes a predefined YAML vulnerability fingerprint file as input and performs the vulnerability assessment of the targets based on whether their properties match those defined in the fingerprint file. If they do, the information gathered by this process is recorded and stored for further processing.
3. **Enrich Information:** The final task in this DAG takes the list of vulnerable IPs from the previous task and enriches it with contact information. This is done by first checking whether a security.txt file exists for each host. Security.txt file contains contact information of a specific domain that can be used to securely report found vulnerabilities [24]. As the adoption rate of the security.txt file is still low [12], a backup option was implemented through which an attempt is made to identify the host's contact information by querying the WHOIS and RIPEstat databases. If the relevant contact information is found, it is saved with relevant vulnerability data. However, if the task fails to find the relevant contact information, it stores the vulnerability data in a separate location for manual inspection by the researcher working on the case.

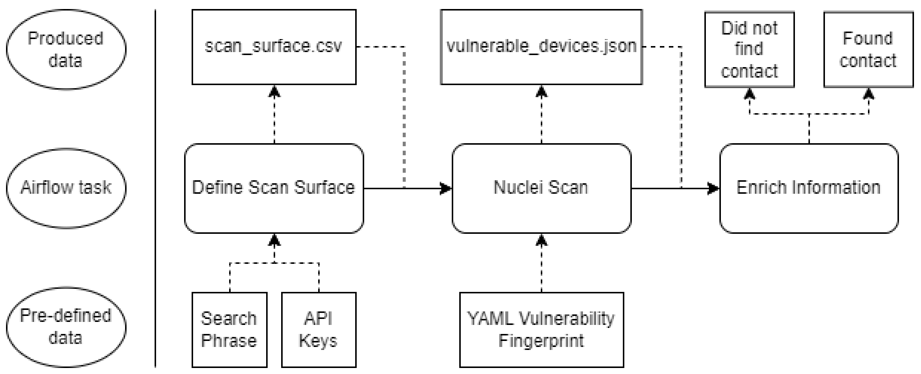


Figure 3: DAG 1: Initial Scan and Data Enrichment

The second DAG was named "Email Notification and Rescheduling", and it is responsible for sending out the first batch of notification emails and subsequent vulnerability assessments and notification of the contacted domains. It must be manually started after the first DAG has finished its execution. The data generated by the first DAG can be directly used by the second DAG and requires no additional configuration. The second DAG consists of five Airflow tasks, which are responsible for sending the first round of emails containing vulnerability notification information, rescheduling the scanning and notification process, and executing it once again. The repeated execution of the scanning and notification process after a predetermined time delay is performed to keep track of the patching process and to further encourage vulnerable hosts to take preventive measures. The tasks that the DAG "Email Notification and Rescheduling" consists of are as follows:

1. **Email Task:** The first task performed by this DAG is responsible for emailing vulnerability notifications to the email addresses belonging to vulnerable hosts identified by the previous DAG. To do this, the task first connects to an SMTP server through which the email notifications are sent. The task then loops through all the IP addresses identified as vulnerable and sends the relevant vulnerability information. The content of the email is formatted using a pre-defined Jinja2 template, which includes information about the vulnerability found, information about the host on which it was found, and the timestamp of the scan. The automation solution is currently being adapted to the DIVD workflow, so it includes a reference to the DIVD case number and a link to the website where the person can find more information about the particular vulnerability and the organization performing the vulnerability disclosure process.

2. **Wait for 1 month:** After the emails have been sent, the proposed implementation of the automation pipeline is configured to perform a wait of 1 month before the scanning and notification process is repeated. The waiting period is configurable and can be adjusted according to the needs of a particular CVD case that is being worked on.

3. **Nuclei Scan:** Once the waiting period has expired, the vulnerability scan is repeated by the task that follows. The scanning process implemented by this task is identical to that implemented by the Nucleus Scan task of the first DAG, with a slight adjustment. The differentiating factor between these two tasks is that the task implemented by the first DAG performs the vulnerability assessment on the entire initial scan surface, whereas the Nuclei Scan task implemented by this DAG focuses on the vulnerability assessment of only those IPs that were contacted during the first email phase of this task. This ensures that the second round of scanning and notification process only targets those hosts that were determined to be vulnerable during the initial scan. This ensures that the progress of vulnerability patching and the success rate of the CVD process can be tracked.

4. **Enrich Information:** Once again, the vulnerability scanning process is followed by the data enrichment task. This task behaves identically to the enrichment task executed by the first DAG, intending to retrieve the most recent contact information of vulnerable hosts. The up-to-date contact information is then added to the hosts determined to be vulnerable by the previous task.

5. **Email Task:** The final task of this DAG processes the most recent vulnerability information and sends this to the email addresses found by the previous task. The emailing process is identical to the first task implemented by this DAG, resulting in all vulnerable hosts being notified of the found vulnerabilities once again.

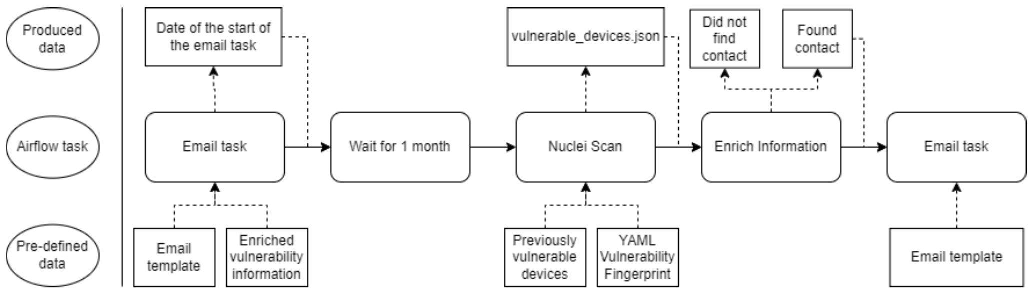


Figure 4: DAG 2: Email Notification and Rescheduling

Due to the modular nature of Airflow tasks [22], the second DAG can be extended with subsequent scan and notification phases. This enables the researcher working on the CVD case to have full control over how many scan and notification phases, and over what periods they would like to conduct the CVD process before closing the case.

6 Discussion

This study identified the exact steps and tools used by DIVD when conducting a mass CVD process. The results obtained provide insight into how DIVD conducts vulnerability disclosure cases. These findings allow for the exact replication of the mass-scale CVD process as conducted by DIVD, providing solutions to the identified problem of the CVD process lacking process standardization. In addition, any company or organization wishing to conduct the CVD process at a large scale can follow the steps defined in this study and use them as an accepted practice. The identification of the steps through which the mass-scale CVD process can be performed was followed by the definition of the requirements through which such a process can be automated. By separating the automation requirements into functional and non-functional, this study provides a clear overview of how an automation solution should behave, both in terms of its functionality and the qualities of the underlying system. This also enabled the identification of an appropriate technology stack to develop the proof-of-concept of the automation pipeline, while paving the way for future automation research.

The proposed proof of concept solution implements all identified automation requirements categorized as "must have". In doing so, the developed automation pipeline fully replicates the manual mass-scale CVD process identified by SRQ1, thereby achieving automation of the identified manual process. As stated in the requirements, the implemented automation solution starts by defining the scan area where vulnerable devices will be identified. This is followed by a vulnerability scan, implemented by Nuclei, which identifies the vulnerable devices based on their behavior. The developed solution expects a vulnerability fingerprint to be written through a template compatible with the Nuclei scanning tool, enforcing standardization of the vulnerability template structure. Once the vulnerable devices have been identified, the vulnerability information is enriched with the associated abuse email addresses. The hosts whose email addresses were not found are separated into a separate list and stored for further processing by the researcher, implementing the requirement to differentiate between vulnerable hosts with and without contact information. For the automated pipeline to start, the researcher responsible for the CVD process must start it manually. During the execution of the automated processes, the researcher can stop the active pipeline at any point. This implements the requirement for manual control of the automated pipeline. Before the automated emails are sent, the researcher responsible for the CVD case is allowed to review the vulnerability and contact information, thereby implementing the need for human verification of the data before the email notification process starts. In addition, the email notification process is automated, fulfilling the requirement for automated contact of vulnerable IP addresses.

The proposed proof-of-concept mass-scale CVD process automation pipeline was developed by utilizing the WMS Apache Airflow. As the developed system comes with a custom-written Docker file, it can be deployed through Docker containers, achieving easy deployment of the developed solution and potential for scalability. Apache Airflow comes with a login form and access control system, through which the security of the system has been implemented. Furthermore, as Apache Airflow performs complex workflow orchestration by placing tasks within a DAG, the developed solution achieves modularity, allowing easy customization of already developed DAGs. New functionality can be added by introducing new tasks to the existing workflow, and existing functionality can be adapted by adjusting the order of execution of tasks within the DAG. Apache Airflow also provides a comprehensive overview of the current status of task execution and detailed logging, allowing researchers to keep up to date with the execution status of the automation pipeline. As the developed system has been implemented using the same tools identified to be used by DIVD, the developed solution can be immediately integrated into DIVD's existing workflow, allowing for testing and validation of the proposed solution in real-world scenarios. Therefore, all non-functional requirements identified as MUST have been implemented by the proposed automation pipeline.

After the development of the proposed automation solution was completed, a meeting was held with DIVD researchers to test the functionality of the developed automation pipeline. During the test of the proof-of-concept automation pipeline, the researchers expressed an overwhelming interest in implementing the proposed automation solution into DIVD's CVD workflow and further testing it on real CVD cases. The researchers also noticed a significant potential in increasing the efficiency of the CVD process and decreasing the amount of manual labor required to completely follow through on the CVD process. The full feedback about the experience of a researcher testing the automation pipeline can be found in the appendix **??**. This initial test of the proposed solution demonstrates the feasibility of automating the CVD process and the potential of automation solutions to streamline and increase the efficiency of the CVD process. Due to the time limitations of this project and limited access to data, this study did not attempt to create a dataset covering the timeframes of DIVD's CVD cases. However, observation of DIVD's publicly available case timelines recognized delays in the DIVD's CVD vulnerability detection process. Currently, there is a 1 to 15-day gap between defining vulnerabilities and initiating the scanning procedures which presents an opportunity for improved efficiency. Cases have also remained open without progress nor follow-up actions after several months. Therefore, integrating an automation solution into the CVD workflow presents the potential to significantly reduce processing times by automatically scheduling further rounds of scans as well as sending appropriate notifications when required. Consequently, such improvements would ensure consistency while preventing periods of stagnation during each case's lifecycle.

7 Conclusion

This chapter provides a summary of key findings, gained insights, and contribution of the project regarding automation of the mass-scale CVD processes.

The objective of the study conducted by this research project was to determine the feasibility of creating an automation solution for the mass-scale CVD process. To this end, this study defined three sub-questions, and answering each question brought the project closer to realizing the proof-of-concept of the mass-scale CVD automation pipeline. By answering the SRQ1, this study identified the key tools and processes that DIVD employs when conducting a mass-scale CVD process. The insight gained from these findings provides a clear definition of what a mass-scale CVD process entails and enables replication of the CVD process as it is performed by DIVD. The answer to SRQ2 identified the requirements that a solution attempting to automate the mass-scale CVD process should follow, enabling the implementation of such a solution. The identified requirements reflect both the behaviors and qualities that the automation system should aim to achieve, paving the way for a straightforward development process. SRQ3 explored the technologies through which the automation pipeline could be realized, resulting in a working proof-of-concept of the mass-scale CVD automation pipeline. The automation solution proposed by this study replicates the scanning and notification stages of the manual mass-scale CVD process, introducing the potential for gain in efficiency, speed, and accuracy of the CVD process. By answering these questions the study identified significant potential for automation and technical improvement of the CVD process while laying groundwork for further automation and optimization efforts.

A pilot has been done with two researchers from DIVD. They both mention that an average case will take 4 hours of manual investment. With the proposed automated solution, it will take 30 minutes, according to their runs in production. This means the speedup is approximately 800

8 Future Work

For future work, this study recommends evaluating the impact that the introduction of the automation pipeline had on the DIVD's CVD workflow. The future study should compare the performance of the manual CVD workflow and the automated CVD workflow introduced by the proposed automation pipeline, comparing their speed and efficiency. Such a study could confirm the interpretation of the results of this study, which is that the introduction of automation to the CVD process will increase its speed and efficiency. Furthermore, this study recommends further exploration of available automation technologies and workflow management systems, intending to further optimise the CVD process automation. As this study only briefly introduced a comparison of workflow management systems, a more in-depth effort is proposed to identify the optimal automation solution. Finally, this study suggests conducting additional research on automating the detection and categorization of vulnerabilities. Advanced machine learning solutions, as identified by [23], may be necessary for this task, and have therefore been beyond the scope of this study. Nevertheless, integrating such technology into the pipeline could further optimize the performance of the CVD workflow.

References

[1] Operators - airflow documentation. `https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/operators.html`, 2023. Accessed: 2024-02-10.

[2] Understanding airflow variables. `https://docs.astronomer.io/learn/airflow-variables`, 2023. Accessed: 2024-02-10.

[3] Khadija Sania Ahmad, Nazia Ahmad, Hina Tahir, and Shaista Khan. Fuzzy_moscow: A fuzzy based moscow method for the prioritization of software requirements. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 433–437, 2017.

[4] Sehrish Alam, Shahid Nazir Bhatti, S. Asim Ali Shah, and Dr. Amr Mohsen Jadi. Impact and challenges of requirement engineering in agile methodologies: A systematic review. *International Journal of Advanced Computer Science and Applications*, 8(4), 2017.

[5] Leyla Bilge and Tudor Dumitraş. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 833–844, New York, NY, USA, 2012. Association for Computing Machinery.

[6] Dutch Institute for Vulnerability Disclosure. Code of conduct. `https://www.divd.nl/code/`, 2023. Accessed: 2024-02-13.

[7] Dutch Institute for Vulnerability Disclosure (DIVD) Computer Security Incident Response Team (CSIRT). Cases handled by divd csirt in 2022. `https://csirt.divd.nl/cases/`, Nov 2022. Accessed: 2024-02-10.

[8] Robin A. Gandhi, Anup C. Sharma, William R. Mahoney, William L. Sousan, Qiuming A. Zhu, and Phillip A. Laplante. Dimensions of cyber-attacks: Cultural, social, economic, and political. *IEEE Technology and Society Magazine*, 30:28–38, 2011.

[9] Suraj Gangwar and Vinayak Narang. A survey on emerging cyber crimes and their impact worldwide. 2020.

[10] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26, 2007.

[11] D. Gross and E. Yu. From non-functional requirements to design through patterns. *Requirements Engineering*, 6(1):18–36, 2001.

[12] Tobias Hilbig, Thomas Geras, Erwin Kupris, and Thomas Schreck. security.txt revisited: Analysis of prevalence and conformity in 2022. *Digital Threats*, 4(3), oct 2023.

[13] Ayei E. Ibor, Florence A. Oladeji, and Olusoji B. Okunoye. A survey of cyber security approaches for attack detection, prediction, and prevention. *International Journal of Security and Its Applications*, 2018.

[14] Youngsoo Kim, Ikkyun Kim, and Namje Park. Analysis of cyber attacks and security intelligence. In *De musica*, 2013.

[15] Gerald Kotonya and Ian Sommerville. *Requirements Engineering: Processes and Techniques.* John Wiley Sons, 1998.

[16] Marleen Weulen Kranenbarg, Thomas J. Holt, and Jeroen van der Ham. Don't shoot the messenger! a criminological and computer science perspective on coordinated vulnerability disclosure. *Crime Science*, 7, 2018.

[17] Andrea Lisi, Prateeti Mukherjee, Laura De Santis, Lei Wu, Dmitrij Lagutin, and Yki Kortesniemi. Automated responsible disclosure of security vulnerabilities. *IEEE Access*, 10:10472–10489, 2022.

[18] Ruth Malan and Dana Bredemeyer. Functional requirements and use cases. 12 2001.

[19] Ryan Mitchell, Loc Pottier, Steve Jacobs, Rafael Ferreira da Silva, Mats Rynge, Karan Vahi, and Ewa Deelman. Exploration of workflow management systems emerging features from users perspectives. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4537–4544, 2019.

[20] C. Mohan, G. Alonso, Roger Günthör, and Mohan Kamath. Exotica : A research perspective on workflow management systems. 1995.

[21] Mohit Nara, Aquila Shaikh, and Rashmita Pradhan. Managing data pipeline with apache airflow. *International Journal of Advanced Research in Science, Communication and Technology*, 2023.

[22] Mohit Nara, Aquila Shaikh, and Rashmita Pradhan. Managing data pipeline with apache airflow. *International Journal of Advanced Research in Science, Communication and Technology*, 2023.

[23] Ahmet Okutan, Peter Mell, Mehdi Mirakhorli, Igor Khokhlov, Joanna C. S. Santos, Danielle Gonzalez, and Steven Simmons. Empirical validation of automated vulnerability curation and characterization. *IEEE Transactions on Software Engineering*, 49(5):3241–3260, 2023.

[24] Tara Poteat and Frank Li. Who you gonna call? an empirical evaluation of website security.txt deployment. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 526–532, New York, NY, USA, 2021. Association for Computing Machinery.

[25] Rechtbank Arnhem. European code law identifier: Ecli:nl:rbarn:2008:bd7578. `https://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:RBARN:2008:BD7578`, 2008. Accessed: 2024-02-13.

[26] Rechtbank Den Haag. Ecli:nl:rbdha:2014:15611. `https://www.recht.nl/rechtspraak/?ecli=NL:RBDHA:2014:15611`, 2014. Accessed: 2024-02-13.

[27] Jukka Ruohonen, Sampsa Rauti, Sami Hyrynsalmi, and Ville Leppänen. A case study on software vulnerability coordination. *Inf. Softw. Technol.*, 103:239–257, 2018.

[28] Sameer Shukla. Developing pragmatic data pipelines using apache airflow on google cloud platform. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, 10:1–8, 08 2022.

[29] Georgios Spanos and Lefteris Angelis. A multi-target approach to estimate software vulnerability characteristics and severity scores. *J. Syst. Softw.*, 146:152–166, 2018.

[30] Kiran Sridhar, Jonathan M. Spring, and Daniel W. Woods. Cybersecurity information sharing: Analysing an email corpus of coordinated vulnerability disclosure. 2021.

[31] Max van der Horst. Global vulnerability vigilance: Timely disaster notification using internet-scale coordinated vulnerability disclosure. *Dutch Institute for Vulnerability Disclosure*, October 2023. Available online: `https://drive.google.com/file/d/1iAxgmuL9kbbj1ONEZMrkSLtX52ajUegC/view?usp=sharing`.

[32] Thomas Walshe and Andrew Simpson. Towards a greater understanding of coordinated vulnerability disclosure policy documents. *Digital Threats*, 4(2), aug 2023.

[33] Merve Şener. Economic impact of cyber attacks on critical infrastructures. *Advances in Digital Crime, Forensics, and Cyber Terrorism*, 2019.

A Questions that were asked during interviews

- What steps does the process consist of?
- What kind of scanning services are used?
- What happens after a scanning service is done with their job?
- What is the collector and what does it consist of?
- Can you explain the controller part, please?
- What parts are useful for the pipeline?
- What will be part of the MVP?
- Are all these services meant to be running on one service?
- What is the current status of development?
- Are we supposed to build on top of the current architecture/codebase/ideas of development?
- What are the intentions of the MVP?
- Do you have some functional requirements defined?
- Do you have some technical requirements defined?
- Can you explain the part stating different contact details per country?
- Is the end goal to automate all of these steps in the pipeline, going from identifying to contacting parties?
- Can you show us the current code that has been developed?
- Shall we make an example using NMAP to base the pipeline one?
- What technology should be used? Ansible and Terraform are mentioned in the project description, but can we just use the most appropriate solution?
- What programming language(s) is/are currently used in the process?
- How much time does it take to open or close a case?

B Automation Pipeline Test - Researcher Feedback

The researcher conducting the test of the automation pipeline provided the following feedback on their experience and observations when testing the CVD process with the proposed automation solution. The feedback has been directly translated from Dutch to English while making minimal changes to the meaning and context of the text:

> *"Currently, this whole process is very time intensive. And it often takes several days until we get around to mailing, for example, after enriching. By automating this process, we take away some very time-intensive tasks. Namely enriching, sending out, and rescanning. Enriching can sometimes take up to several hours and because someone is not constantly checking if it is ready, we save a lot of time with the created pipeline! The same with rescanning and resending, we save a lot of time with the pipeline. Mainly because it is then standardized and automated.*
>
> *After all, we already have the data, only the researcher has to make time for it. With automation, you need the volunteer less and he can focus on the main point, which is researching how to identify a device and creating the template. Once these are correct, the pipeline means you don't have to worry about the other points and this is all done automatically. As a result, I can now research a fingerprint and create a template in two hours. Where first I need a few hours spread over several days, for scanning, enriching the next day, and emailing the day after that. Not to mention the rescan process, which will also take additional hours.*
>
> *Of course, we are now at a stage where we also check that the pipeline as a whole is working properly. But here you only need to check randomly if the data is accurate. So far I have not seen any incorrect data, so possibly in the future, we can run the entire process on its own. So in my opinion, the delivered Proof-of-Concept is very successful and we from the DIVD CSIRT can move forward with this! The code is very clearly written and we can incorporate improvements and features later if necessary."*