Cyber Security Assessment Automated System

BY

NATTHA SIRIBOON
PAWEE TANTIVASDAKARN
SOPONPAKORN SUTTIKAO

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN COMPUTER INNOVATION ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2020

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Cyber Security Assessment Automated System
Student Name	Miss Nattha Siriboon Student ID. 60011240 Mr. Pawee Tantivasdakarn Student ID. 60011252 Mr. Soponpakorn Suttikao Student ID. 60011271
Degree	Bachelor of Engineering in Computer Innovation Engineering
Project Advisor	Signed: Orathai Sangpetch (Asst. Prof. Dr. Orathai Sangpetch)
Project Co-Advisor	Signed: Akkarit Sangpetch (Asst. Prof. Dr. Akkarit Sangpetch)

Project Title	Cyber Security Assessment Automated System
Student Name	Miss Nattha Siriboon Mr. Pawee Tantivasdakarn Mr. Soponpakorn Suttikao
Degree	Bachelor of Engineering in Computer Engineering
Project Advisor	Asst. Prof. Dr. Orathai Sangpetch
Project Co-Advisor	Asst. Prof. Dr. Akkarit Sangpetch
Academic Years	2020

ABSTRACT

Security hygiene is important, especially within the finance industry, as user privacy is a major priority for finance services. To ensure the highest levels of cybersecurity, security tests must be carried out regularly. Under normal conditions, security testing is carried out by humans. As such, it is time-consuming, costly, and difficult to carry out on a regular and frequent basis. The aim of this project is to combat these problems using machines to run the process automatically, allowing assessment to be undertaken more frequently. The system developed, Cyber Security Assessment Automated System (CSAAS), can run security tests at all hours of the day, making use of a black-box network vulnerability assessment technique and various open-source tools (Hydra, DIRB, Nmap, SQLmap, Amass, Searchsploit, and XSSsniper). A broker company should provide the endpoints for running security tests. After which the system will run all necessary tests, generating a report to send back to the broker company. This report includes general information (such as IP address, port, and OS) as well as the vulnerabilities of the given endpoints.

ACKNOWLEDGEMENTS

Firstly, we would like to express our sincere gratitude to our advisors, Asst. Prof. Dr. Akkarit Sangpetch and Asst. Prof. Dr. Orathai Sangpetch, for their patience, constant support, and encouragement. Their guidance helped us to learn from the problems we faced, allowing continuous development of the project.

Besides our advisors, we would like to thank the rest of the CIE capstone project committee for their comments and questions.

Our sincere thanks also go to the seniors at CMKL labs fortheir discussion on how to implement the system, and their other suggestions, which helped us significantly in our decision-making.

Finally, thanks to each of us for the contributions we made and the sleepless nights that we worked together.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
CIE	Computer Innovation Engineering
SIIE	School of International I Interdisciplinary Engineering Programs
CSAAS	Cyber Security Assessment Automated System
DOM	Document Object Model
DOS	Denial-of-service
DDOS	Distributed Denial-of-service
OWAP	Open Web Application Security Project
SQL	Structure Query Language
JSON	JavaScript Object Notation
IP	Internet Protocol
ICMP	Internet Control Message Protocol
HTTP	Hypertext Transfer Protocol
DAG	Directed Acyclic Graph
SSH	Secure Shell
VM	Virtual Machine
ID	Identifier
CPU	Central Processing Unit
VPC	Virtual Private Cloud
PTR	Pointer
CPU	Central Processing Unit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SMTP	Simple Mail Transfer Protocol
URL	Uniform Resource Locator

CHAPTER 1

INTRODUCTION

This chapter introduces the overarching themes of the reports and places the motivation for the work into context. Thereafter, the rationale and goals for the investigation of the project are discussed and defined, followed by a summary of the overall project. Finally, an overview of the dissertation is given on a per chapter basis.

The internet has become a significant part of our daily lives. Many services operate via internet. The number of internet users has grown substantially. Meanwhile, the number of people with malicious intentions has also increased dramatically. Security can be a big concern when exploring the cyber world. Likewise, trading is another service which has been continuously and rapidly changing, with online trading becoming more popular over time. It can be accessed via websites or mobile applications, with the amount of money circulating within the broker trading system, becoming a major target for hackers wishing to exploit the system and steal precious information for various purposes including making demands for ransom payments. Hernandez's (2017) interesting article on the security of mobile based trading indicated that 62% of the apps investigated sent sensitive data to log files, while 67% stored this data in an unencrypted format. However, physical access to the device was required to extract this data. [1]

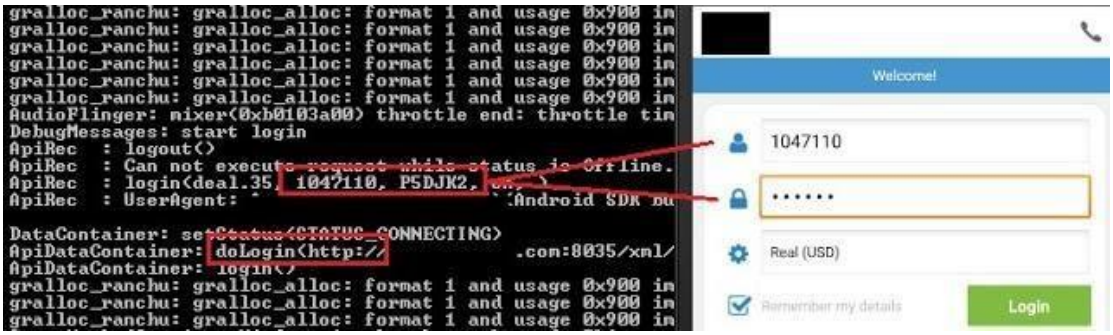


Figure 1.1 Unencrypted log file, exposing the username and password.

1.1 Background

Application of best practices, and proper configuration of the broker system are necessary in order to effectively prevent cyber-attacks. The most important matter to be considered is the hiring of a specialized company able to perform regular and effective security testing for the system. Nevertheless, as this can be expensive and time-consuming, it is hard for many companies to hire and maintain their system using a security testing service. The consequent lack of maintenance could lead to various security vulnerabilities.

1.2 Objective

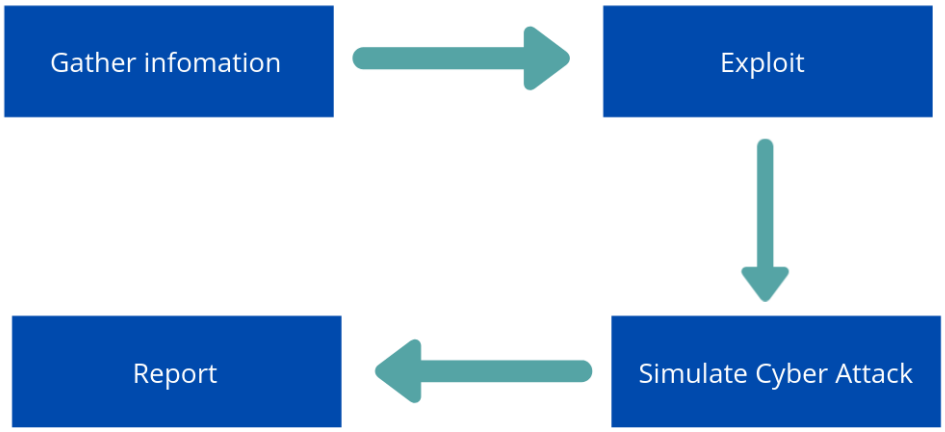


Figure 1.2 Steps of Vulnerability Assessment

An alternative to security testing is network scanning. Scanning via the internet is not a highly complex scanning test option, when compared with a security test. On the contrary, it provides businesses with the convenience to scan regularly. Additionally, it can show the difference between the system hygiene of each scan. The aim of CSAAS is to provide an automated vulnerabilities assessment service which meets the following 2 objectives:

- The service can be run automatically with endpoints provided by the broker company following the procedure shown in Figure 1.2

- The broker company can access reports for any new scan as well as previous records for the purpose of auditing and improvement of the broker's system.

1.3 Scope

- The scanning process includes the process of subdomain enumeration (Amass), enables port scanning (Nmap), SQL injection (SQLMap), cross-site scripting (XSSSniper), brute-force on available services (Hydra), and brute-force of web service directories (DIRB).
- The scanning process will be performed on the network scanning where the service knows only the given endpoints from the broker's company.
- The scanning process will run automatically from gathering information to generating the report as illustrated in Figure 1.2
- The broker company can add to or remove from the list of endpoints, access the records of all scans, and will be able to download the report whenever a scan is finished.

1.4 Methodology

- The broker company's employees can add and remove endpoints of the services CSAAS is required to scan, from the website scan page. The endpoints will be shared within the company.
- CSAAS admin will access the website to see the list of each company's endpoints on the website admin page.
- Admin can generate a scan from a template which contains the pipeline of the scanning process within the orchestrating parallel jobs service, Argo Workflow. The scan is generated with a unique ID and recorded on the database. The broker company's employees can acknowledge the scan's initiation from the website history page.
- The scanning process generates a log file from tools in the pipeline, uploading it to Object Storage. At the end of the pipeline, the JSON file

containing the scan results are uploaded to the management database. Moreover, the scan is marked as “success”.

- The website history page for the broker company will indicate that the scan has finished, and the report will be available for download.
- When the broker company’s employees search for the report on the website history page, the JSON file for the selected scan in the database will be used on the report generator service to create a real-time PDF file from the template which will then be displayed on the browser.

1.5 Expected Outcomes

- The broker company will gain knowledge about their system’s vulnerabilities through the reports and will be able to continuously improve their system.
- At the end of the project, the following functionalities of the system should be achieved:
    1. Broker company employees are able to add and remove from the list of endpoints.
    2. Broker company employees are able to access the records and reports of all previous scans.
    3. The scanning process is automated and detects the vulnerabilities precisely.

1.6 Table of Operation

Table 1.1 Operating Time Frame

Topic	Month									
	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
1. Preparation										
1.1 Research and find suitable tools for inspecting vulnerabilities.										
1.2 Find examples of systems and reports for reference.										
1.3 Design architecture and database										
1.4 Design the first attack on the first vulnerability.										
1.5 Create Kubernetes one node cluster.										
1.6 Create cloud Postgresql cluster.										

Topic	Month									
	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
2. Implement Application										
2.1 Upload each tool image and report generator.										
2.2 Implement Argo workflow report for reference.										

2.3 Implement Argo workflow template.										
2.4 Implement data cleaner.										
2.5 Deploy web server for user interaction.										
2.6 Implement backend service.										
2.7 Implement Argo middleware service.										
3. Testing and debugging										
3.1 Test and improve the performance of the workflow.										
3.2 Test and debug each tool.										
3.3 Test and debug workflow										
3.4 Test with endpoints provided by the company.										
3.5 Get feedback from the company.										
3.6 Debug workflow after applying web application security risk tools.										
3.7 Test with endpoints provided by company II.										
3.8 Test, Debug, and improve the workflow performance.										

1.7 Report Outline

The remainder of this report is organized as follows:

- Chapter 2 reviews state-of-the-art vulnerability assessments
- Chapter 3 describes the design and implementation of automated security scan systems.
- Chapter 4 demonstrates how tools have been validated and presents the results of the system evaluation.
- Chapter 5 closes the report, reviews the work undertaken, and draws conclusions.

CHAPTER 2

LITERATURE REVIEW

Chapter 2 discusses the state-of-the-art security, vulnerabilities, and tools considered during the analysis and design phase of the project. The associated investigation served 3 purposes: firstly, it is important to research the Theoretical Background behind the project (section 2.1); secondly, it assists in identifying the tools most appropriate for use in the project (section 2.2); Thirdly, it is also important to research the previous academic work within the project's field of research (section 2.3); Finally, section 2.4 summarizes the whole chapter.

2.1 Theoretical Background

- **Detection of XSS**

  As shown by Liu et al. (2019) [17], cross-site scripting has been used for several types of attacks, such as phishing, cookie stealing, DOS attacks, DDOS attacks, and XSS worms. It is possible to inject data into the client DOM or the server using different techniques. To detect exploitation, various methods can be applied. Static Analysis Methods are applied by analyzing the code of the application. Dynamic Analysis Methods detect problems by injecting data into the website and observing whether an attack is triggered. This project uses a black-box network vulnerability assessment which's scope cannot access the client source code; dynamic analysis methods are therefore more appropriate.

- **Index Calculation**

  The Index is an acceptable number from the consumers that determines the overall system with iterations and calculations. The index is calculated from what has been found and how it impacts the system by the impactful factors. The vulnerabilities are classified by level to measure cybersecurity and then produce the total index to indicate the overall hygiene. For instance, if one of the indexes shows below standard then the indicator would show how and where to prevent any exploits. [15]

- **Security Vulnerability Category and Black-box testing**

  There are several types of security vulnerabilities. Some of them can only attack from the device itself, while others can attack directly from the internet. According to the OWAP top 10 web security risks [16], some of these potential risks cannot be efficiently tested via the black-box perspective e.g., Broken Access Control, Security Misconfiguration, Insecure Deserialization, and Insufficient Logging & Monitoring. OWAP additionally provides a vulnerability image for testing called OWAP juice shop [3], which can be installed on any machine web server and carry out testing with a different difficulty to the security setup.

- **SQL injections**

  SQL injections are performed by injecting code using a form which asks the user for input. The most basic and common form of SQL injection is to enter (' OR '1' = '1) into a username or password. If the target application is vulnerable to this attack, the attacker will gain access to information in the target database. [18] If the target application is immune to basic SQL injections, blind SQL injections can be used to gain more information about the database. Blind SQL injection is a way to ask a database true/false questions, thus gaining information. [19]

- **Network topology**

  Network reconnaissance usually checks the target IP's status to identify its online status. Nmap is a tool to carry out a reconnaissance process. It uses an ICMP protocol to ping the target IP by sending echo requests to port 80, analyze if they respond, and hence let the Nmap know which ports are active or down. This collects information about the target IP.

2.2 Tool Stacks

Tools and implementations, which were used in the project, were classified into 4 categories: Cloud-structure, Front-end, Workflow, and Back-end.

2.2.1 Tools for cloud-structure

- **Digital Ocean**

  This is one of the largest clouds platforms and has the best price in relation to CPU performance compared to its competitors. 7 services from Digital Ocean have been used for the project including Kubernetes, Droplet, Networking, Container, Registry, Spaces, and Database.

- **Kubernetes(K8s)**

  This is a system for automating deployment, scaling, and management of containerized applications. It was used for its containerizing services.

- **Droplet**

  This is a cost-efficient virtual machine with the ability to run an existing image or private image. During the project droplet was used by Kubernetes to create a Kubernetes's Node Cluster.

- **Networking**

  Networking includes many services such as Domain, Floating IPs, Load Balancers, VPC, Firewall, and PTR record. Floating IPs were used to reserve a static IP, and VPC for the private network, while Load Balancers was used by Kubernetes resources, combined with NGINX Ingress Controller, a reverse proxy.

- **Container Registry**

  This is a private repository hosting private images with the version record on the cloud. The container registry was used to host the workflow container images.

- **Spaces**

  Object storage is compatible with S3. Spaces were used for storing the log file outputs from each tool in the workflow, and the report JSON files.

- **Databases**

  A managed database cluster has options for PostgreSQL, MySQL, and Redis. For this project, PostgreSQL was used as the storage system.

2.2.2 Tools for Front-End

- **NuxtJS**

  This is a powerful framework for simple and powerful web development, building on top of Vue.js and supporting SSR. NuxtJS was used for making the web application.

- **Keycloak**

  Keycloak is an open-source Identity and Access Management solution aimed at modern applications and services. It makes it easy to secure applications and services with little to no code.

2.2.3 Tools for Workflow

Tools from Kali Linux [13] and Argo workflow have been used.

- **Argo Workflow**

  Argo Workflow is a workflow engine that runs as a container native. It can be easily integrated with Kubernetes. Argo Workflow orchestrates highly parallel jobs, in which each step is built from a container as a multi-step or DAG in the workflow without the overhead of legacy VM. For this project, Argo was installed as a Custom Resources Definition in Kubernetes and used to run the workflow (from the template) as a DAG with various tools.

- **Nmap**

  Nmap was used to scan at the communication level of protocols to find the status of the ports that are scanned, such as the TCP and UDP port. It scans in secret, making only a two-way handshake

(normally scan is a three-way handshake), and scanning OS information of the target IP or domain name. Furthermore, Nmap uses scripts within its own resources such as vulnerability script.

- **SQLmap**

  The SQLmap is used for detecting and exploiting SQL injection flaws. SQL injection flaws can be exploited by inserting SQL statements as user input to manipulate target databases. If the target does not protect itself against SQL injection, SQLmap can gain data from the said database.

- **Amass**

  Amass is used to scan for IP addresses and enumerate subdomains from the given domain name using a dictionary.

- **Searchsploit**

  Searchsploit is used to search known vulnerabilities in the OS or platform, storing a copy of the exploit database in local storage for quick search. It must be updated once every two weeks to get the most up-to-date information.

- **CVE**

  CVE is a list of records that contain an identification number, a summary, and at least one public reference for known cybersecurity vulnerabilities. CVE records are used in numerous cybersecurity products and services around the world.

- **Hydra**

  Hydra is a parallelized login cracker that supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add. The Hydra tool makes it possible for researchers and security consultants to show how easy it is to gain unauthorized access to a system remotely [13].

- **DIRB**

  DIRB is a Web Content Scanner. It looks for existing (and/or hidden) Web Objects. It works by launching a directory-based attack against a web server and analyzing the response [13].

- **XSSSniper**

  XSSSniper is a handy XSS discovery tool with mass scanning functionalities. It scans the target URL for GET parameters and then injects an XSS payload and parse the response for artifacts of the injection [20].

2.2.4 Tools for back-end

- **PostgreSQL**

  The database will be connected to the API server and any service needs to access the database via API. Some tables in the database will need admin privilege to make insertions, updates, or deletions in order to ensure maximum security. The database will be routinely backed up in case of situations where the main database goes down.

- **Go-gin**

  The Golang framework is used to make this API server. This framework is extremely fast (40 times faster than martini). The server manages to get packages for the programmer. It has a built-in net/HTTP library for Golang, containing every function needed for routing and rendering. The framework has good error management. [14]

2.3 Related Work

2.3.1 Scoring System for Vulnerability

The Common Vulnerability Scoring System (CVSS) [12] expresses a way to capture the principal characteristics of a vulnerability, producing a numerical score reflecting the severity of the vulnerability. The numerical score representation in the enumeration which can be easily seen as low, medium,

high, or critical, helping organizations to properly assess and prioritize their vulnerability management processes.

2.3.2 Security Tests and Security Assessment Report

The summary of vulnerabilities encountered in the system shows the company how many severe vulnerabilities require immediate investigation of details after scanning. [7]

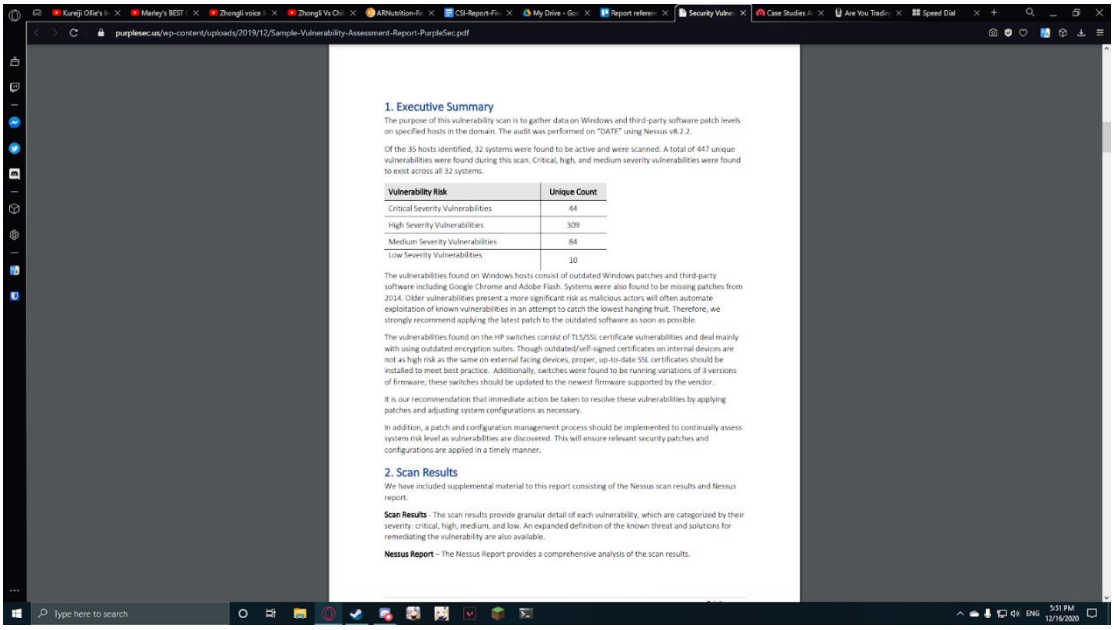


Figure 2.1 The counter of severity in the target system.

2.3.3 Security Tests and Security Assessment Scanner

A web application scanner is an example of a scanner that can produce a report and analyze results. [11]

2.4 Chapter Summary

In Chapter 1 we proposed the importance of cybersecurity to finance businesses, especially in the broker field, also proposing how our CSAAS could improve a project's ecosystem.

In this chapter, recent research, and advances in the field, was categorized into Theoretical Background (section 2.1), Tool Stacks (section 2.2), and Related Work (section 2.3). Observations were made on the various theories (section 2.1), tools and their definitions (section 2.2), and the relevance of the previous works reviewed in section 2.1(section 2.3).

The following chapter presents the design of CSAAS, a system intended to automate cybersecurity assessment.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In Chapter 2 we identified that automated security scans with a black-box perspective can be an interesting solution compared to the traditional penetration test.

This chapter describes the design of CSAAS, a system that assesses the vulnerabilities automatically. First, the chapter describes the vulnerabilities assessment approach (section 3.2);second, the system requirements (section 3.3);third, design and diagrams (section 3.4-3.5);fourth, storage technology (section 3.6); and finally, the proposed flow is discussed in section 3.7.

3.2 Vulnerability Assessment Approach

The Vulnerability Assessment Approach follows three steps: Reconnaissance, Exploitation, and Reporting. Reconnaissance refers to investigation of the target and retrieval of information regarding the endpoint's protocols and subdomain by Nmap and Amass sequentially. Exploitation refers to the search for vulnerabilities in the system including finding the directory and obtaining a list of usernames and passwords via brute force. The report loads the inputs to generate the PDF report from the template using the Carbone tool as shown in Figure 3.1.

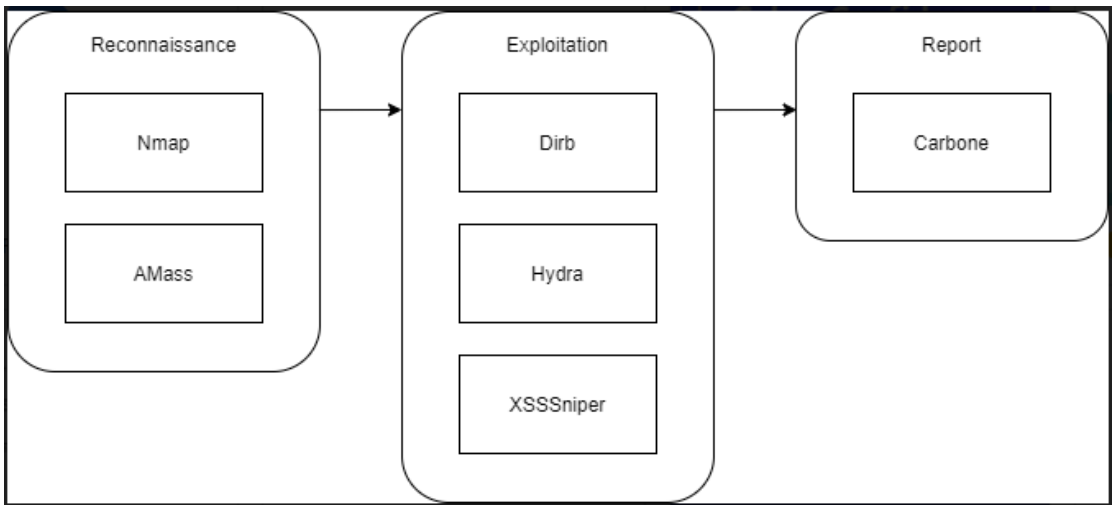


Figure 3.1 Scan Method used in the project.

3.3 System Requirements

- The system can identify the target's vulnerabilities.
- The system can test in a specific window of time.
- The system can perform tests automatically.
- The system can produce a security assessment report in the form of a PDF file.
- The system's services are secure.
- The system is easy to maintain.
- The system is scalable.
- The system is run on the cloud.
- The system's microservices are stateless.
- The system's users can log-in to the system using their username and password.
- The system's users can insert and delete endpoints from their endpoint list.

3.4 Overall Architecture System

Figure 3.2 shows the overall architecture of the CSAAS system. The user accesses the website csi.cmkl.ac.th on their browser. The domain name is resolved by Cloudflare DNS service and points to our Digital Ocean Kubernetes resources. Every request from the user's browser goes to the Ingress NGINX Controller (the integration of the Nginx on Kubernetes with Digital Ocean's load balancer) and passes the request to the service which is either a Web Server or Backend services depending on the request URL. All requests are authenticated and authorized via the Keycloak server.

The broker company's employees are allowed to add, modify, and delete, the endpoints on the browser. The record is stored on the PostgreSQL cluster via the backend service. Admin can start a scan on the Argo Workflow service template. The process of scanning shows on the history page, while the log for each tool can be found on the Digital Ocean space object storage.

The finished scan creates a JSON file on the database for generating the report. The broker company's employees can download the report on the website history page which pulls the JSON data, creating a report directly via the Report Generator.

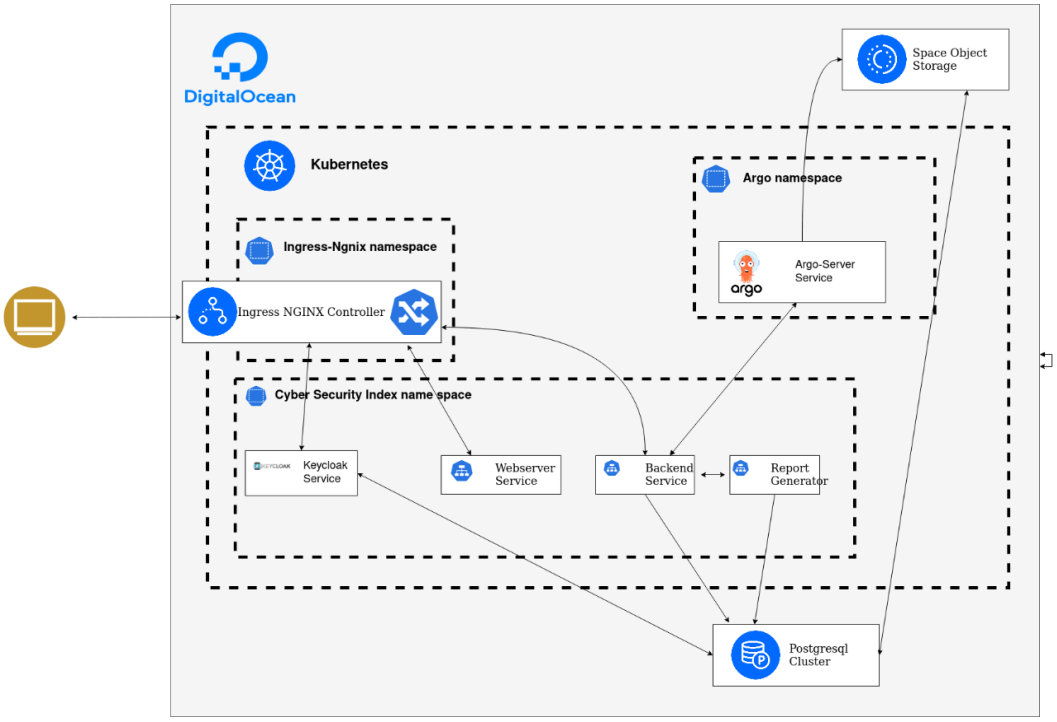


Figure 3.2 Overall System Architecture

3.5 Diagram

3.5.1 Use Case Diagram

As shown in Figure 3.3, The user starts by signing-in with Keycloak OAuth Authentication. The token validation consequently verifies the identity of the user. Once verified, users are able to Add Scan Endpoint, Edit Scan Endpoint, Delete Scan Endpoint, Get Scan History, or Download Report according to their authority.

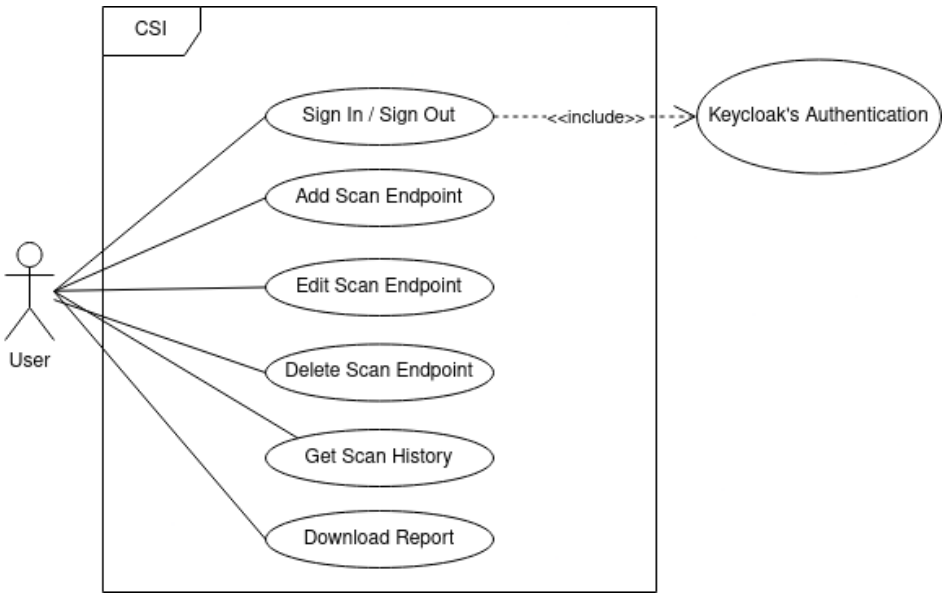


Figure 3.3 Use Case Diagram of the system.

3.5.2 Activity Diagram

The flow of the Argo workflow is illustrated in Figure 3.3. While registering the input ‘Endpoint’, the logic will check whether the endpoint is a domain name or IP. If the endpoint is a domain, the system will run Amass, otherwise, it will run Nmap. In both conditions, the tool will run along with the other tools, SQLmap and XSSsniper in parallel. The result of the Amass will be processed by the Nmap. Searchsploit runs after the Nmap, using the OS info to find the CVE entries which will then be processed by the tool CVE. Searchsploit runs in parallel with hydra or DIRB depending on the result of the Nmap.

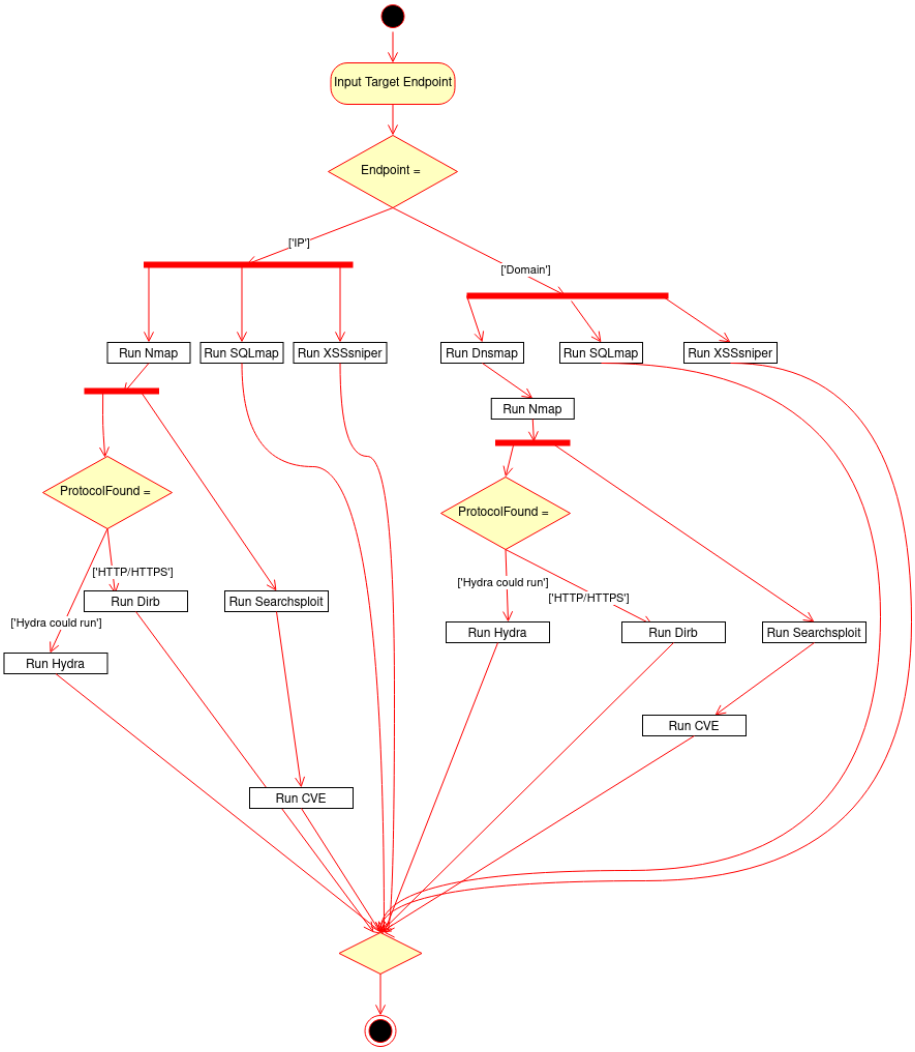


Figure 3.4 Activity Diagram of the scanning workflow with a single endpoint

3.6 Storage

3.6.1 Object Storage

The full log file from each tool is kept for further audit according to the structure design indicated in Figure 3.6. Each company has a different directory indicated by the company name. The company directory contains the logs of the scans indicated by the unique scan identifier.

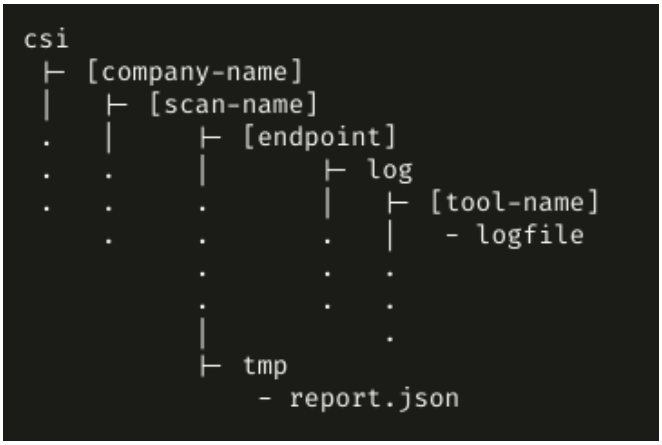


Figure 3.5 Object Storage Structure

3.6.2 Database

The Postgresql database is used to store the user and scanning information. The data for generating the report is stored as a JSON object while other information is stored as relational. SQL databases consist of four tables. All databases run on the Digital Ocean.

Table 3.1 List of scans

Attribute	Data type	Description
scan id	varchar	unique id for each scan, surrogate key. PK
company	varchar	Owner of the scan
status	varchar	The status of the scan has five potential values: failed, success, running, terminated, suspended. failed: scan has failed. success: scan is done successfully. running: scan in-process, scanning. terminated: scan is terminated by the user. suspended: scan in-process, a scan has

		stopped waiting to resume.
Start	Time stamp	Start of scanning time.
complete	Time stamp	Time of scan completion.

Table 3.2 List of endpoints

Attribute	Data type	Description
endpoint	varchar	endpoints that users own.
company	varchar	owner of endpoints.

Table 3.3 List of reports

Attribute	Data type	Description
Scan id	varchar	Produce from which scan
DIR	varchar	The directory that keeps the report

Table 3.4 List of results

Attribute	Data type	Description
Scan id	varchar	Result of which scan
Data	JSON	Data from tools

Table 3.5 List of APIs

APIs	Description	input	output
/addEndpoint	Allow POST method, used by the user for adding a new endpoint.	Header contains user's token. JSON body contains endpoint.	-
/addScan	Allow POST method, used by our system for creating a new scan.	JSON body contains ScanID (auto generated by our system), owner is company's name .	-
/updateScan/:status	Allow PATCH method, used by our system for updating a scan status.	The parameter contains status. JSON body contains scan id and status. If status is changed to 'success' or	-

		‘terminated’ API server will automatically add current time stamp to complete attribute of scan in database.	
/getAllScans	Allow GET method, used by the user to get all their scan information.	Header contains user’s token.	List of scans along with corresponding information: scan id, status, start time and end time.
/getEndpoints	Allow GET method, used by the user to get endpoints in a user's endpoint list.	Header contains user’s token. The parameter contains id of the owner of the endpoint list.	list of endpoints
/DeleteEndpoint	Allow DELETE method, used for deleting endpoint from user’s endpoint list.	Header contains user’s token. JSON body contains endpoint.	-
/uploadResult	Allow POST method, used by our system for upload result of a scan in JSON format.	JSON body contains scan id and result of scan in JSON format.	-
/getAllEndpointsAdmin	Allow GET method, used by admin to get all registered endpoints in system.	-	List of all endpoints in the database and its owners.
/generateReport	Allow POST method, used by our system to create report from a scan.	JSON body contains scan id.	Security assessment report in pdf file.

3.7 Summary

This chapter describes the high-level requirements and design of the CSAAS system. The chapter starts by describing the approach of vulnerability assessment and discusses more regarding the design of the system.

The System features are covered in further detail in Chapter 4 which describes the results of the implementation and experimental results.

CHAPTER 4

EXPERIMENTAL RESULT

4.1 Introduction

Chapters 3 and 4 described the design of CSAAS, a system that assesses the vulnerabilities automatically. In this chapter, we present an implementation and testing method and the results which show the functionality of the system, tool validation, and evaluation of the overall system performance against the test. The chapter is organized as follows: section 4.2 contains a walkthrough of the functionality of the implemented system; section 4.3 measures the reliability of the tools; section 4.4 evaluates the system against the real-world environment.

4.2 Functionality of the System

4.2.1 Backend API

Tokens are used to identify users. Users are only able to gain access to their endpoints list and their scan history. They can also perform some functions, allowing them to add, remove, or update items on their endpoint list. Go-gin and GORM are used to connect with the PostgreSQL database. The figures below show the functionality testing of the backend API.

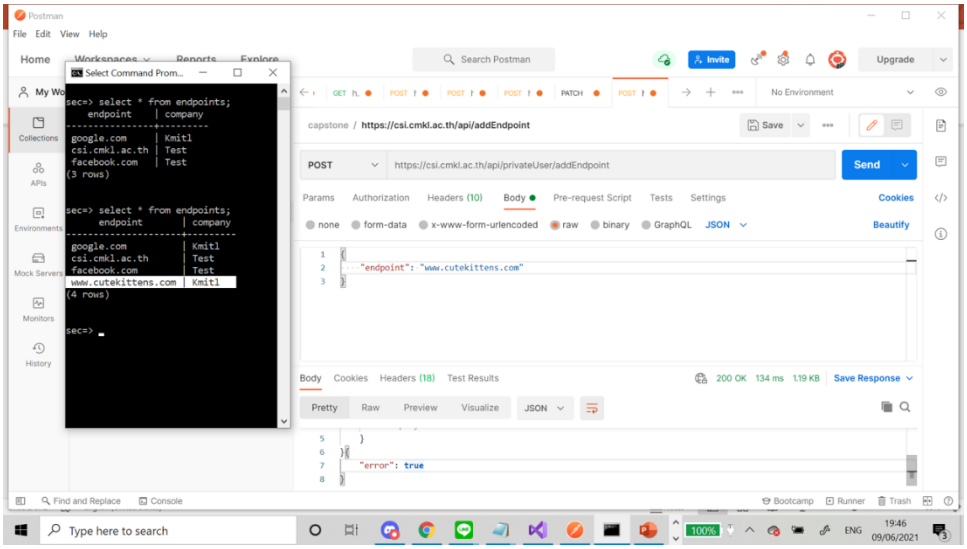


Figure 4.1 Add new endpoint to user's endpoint list. The command line shows the database before and after insertion, a new endpoint highlighted in white.

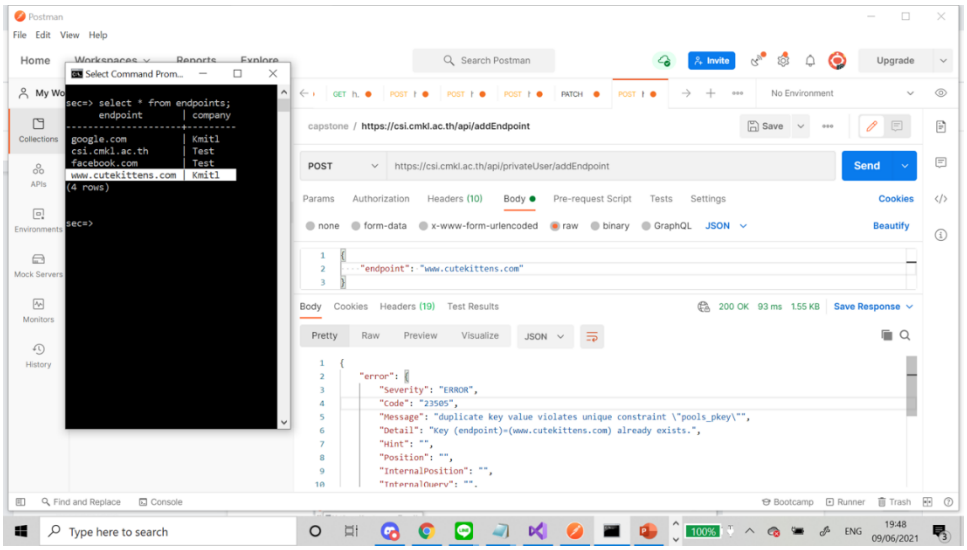


Figure 4.2 Return error (primary key conflict) after an attempt to insert an existing endpoint.

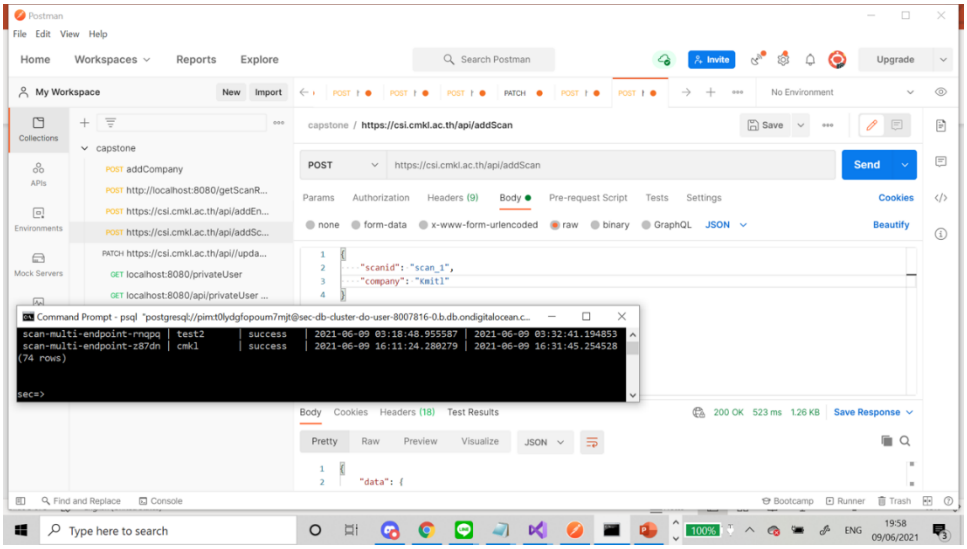


Figure 4.3 Command-line showing the database before adding a new scan. The adding process is done by the system.

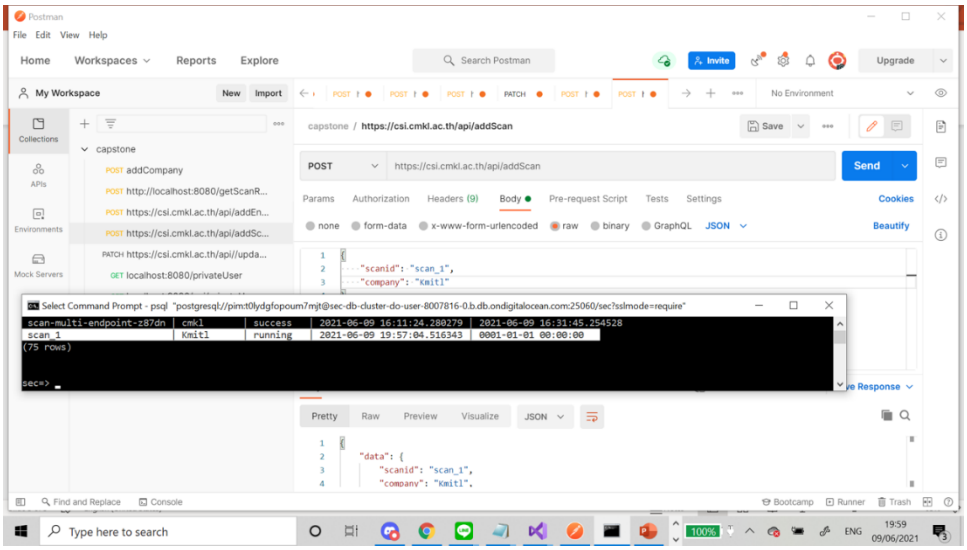


Figure 4.4 Database after adding a new scan. Scan ID will normally be autogenerated. However, for the purpose of demonstration, a new scan was added using scan ID ‘scan_1’

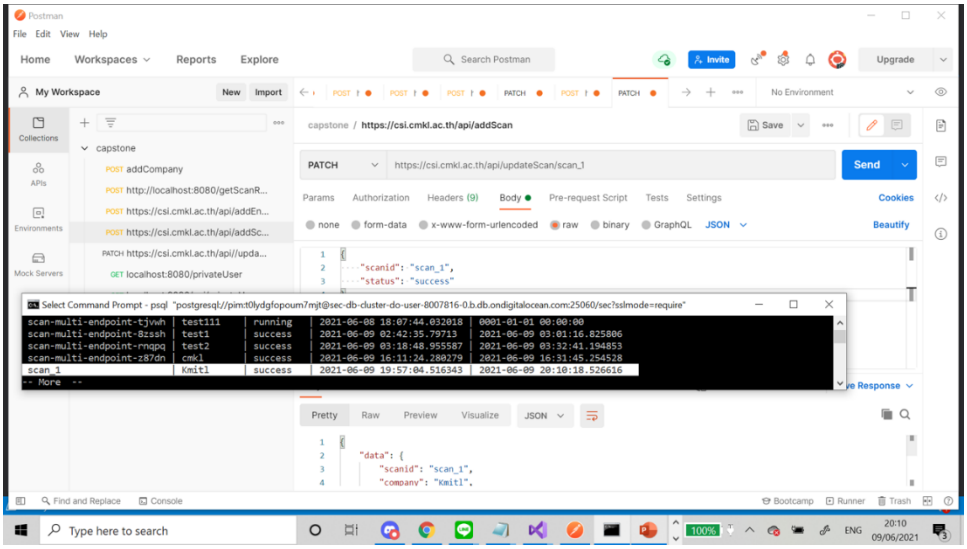


Figure 4.5 As a result of changing the status of 'scan_1' from running to success, the completed time stamp changed from the default value to the current timestamp.

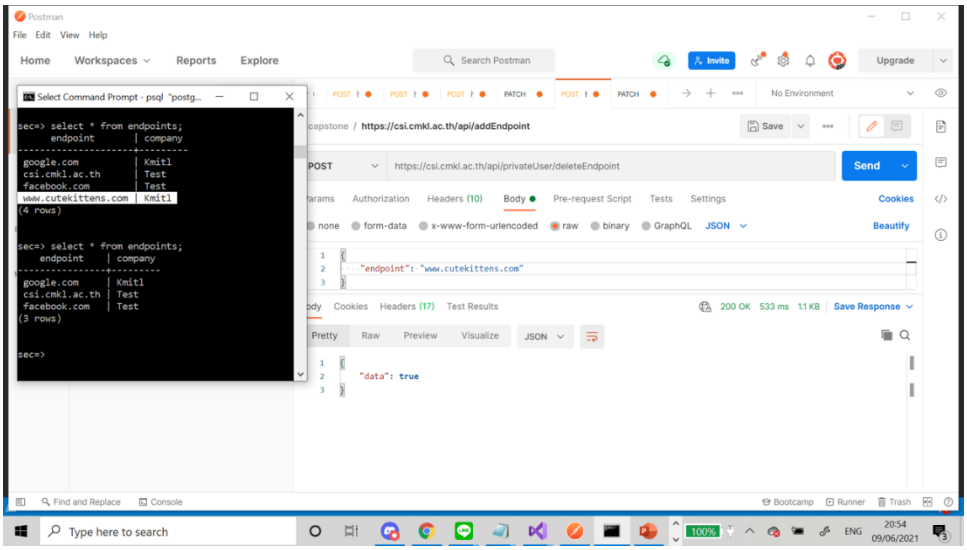


Figure 4.6 Deleted endpoints highlighted in white. The command line shows the database before and after endpoint 'www.cutekittens.com' was deleted.

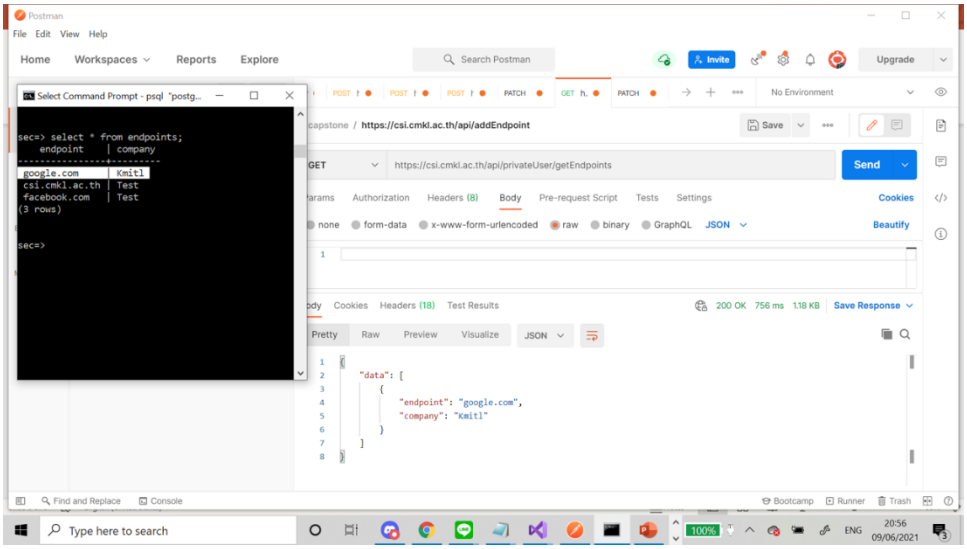


Figure 4.7 The user who belongs to 'Kmitl' company recieves their endpoints list. The company currently has one endpoint in the list as highlighted in white.

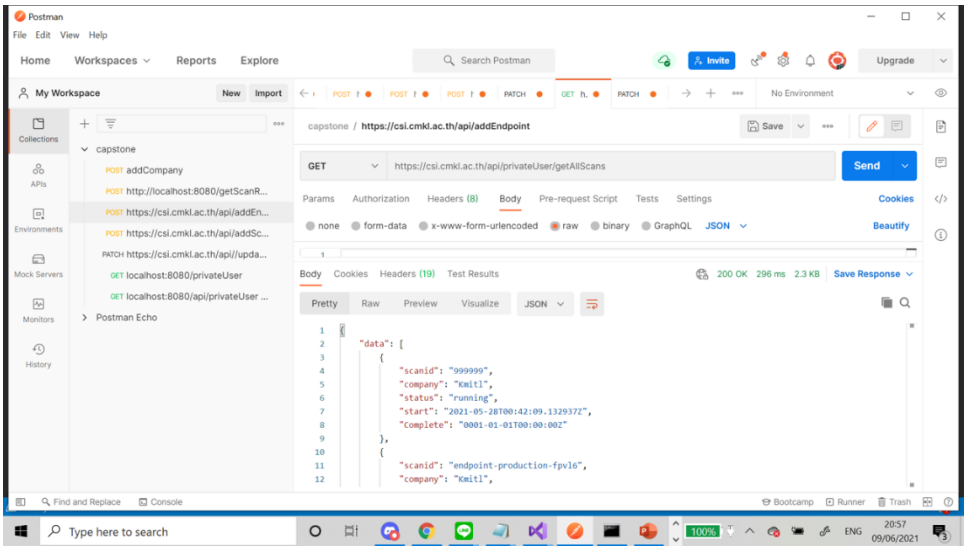


Figure 4.8 The user who belongs to 'Kmitl' company recieves their scan history.

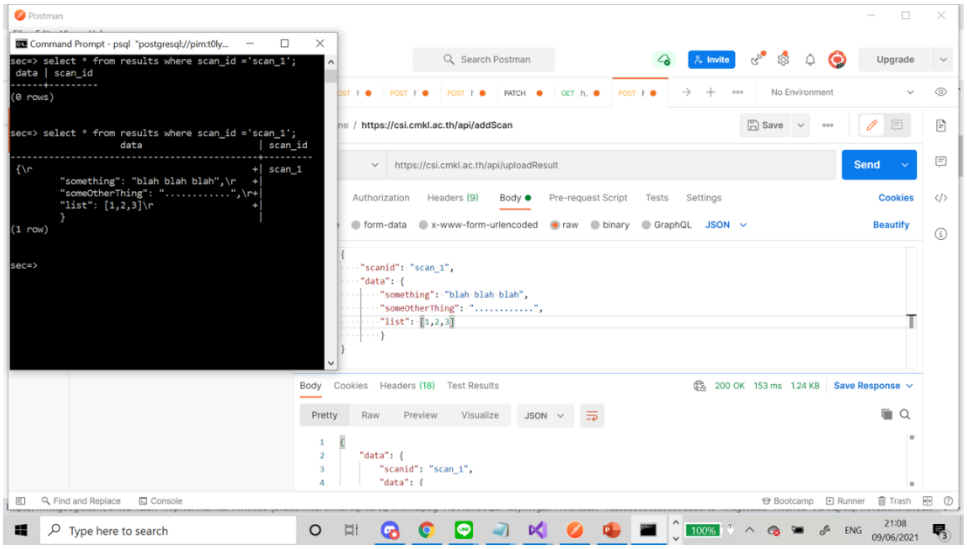


Figure 4.9 Upload scan results to the database. The upload was successful, and the command line shows the database before and after upload.

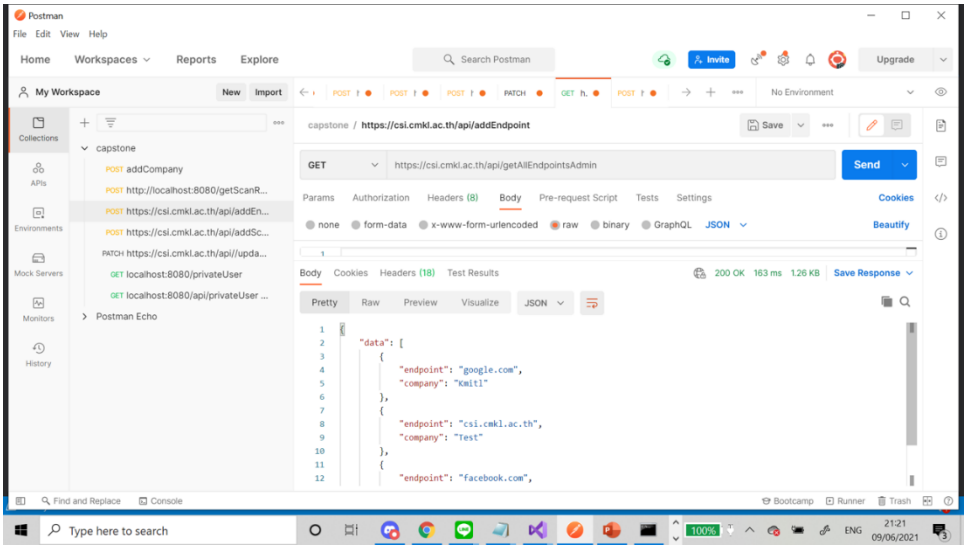


Figure 4.10 Admin gain access to information of all user endpoint lists.

Figure 4.11 Generating a report from the scan, returns a pdf file.

4.2.2 Front End

The user can log-in from the front-end page and will be redirected to /auth path. The auth path will then handle the Keycloak's OAuth Authentication (Figure 4.12). After logging in, the user receives a token identifying their role and group (assigned by the Keycloak admin page). Following the token authorization, the user will land on the page according to their role, as shown in Figure 4.13 (user role) and Figure 4.15 (admin role). Users in the company are able to add some endpoints which are shared in the pool within the company (Figure 4.13). Users can access the scan history as shown in Figure 4.14, while admin can see all request endpoints from all companies (Figure 4.15).

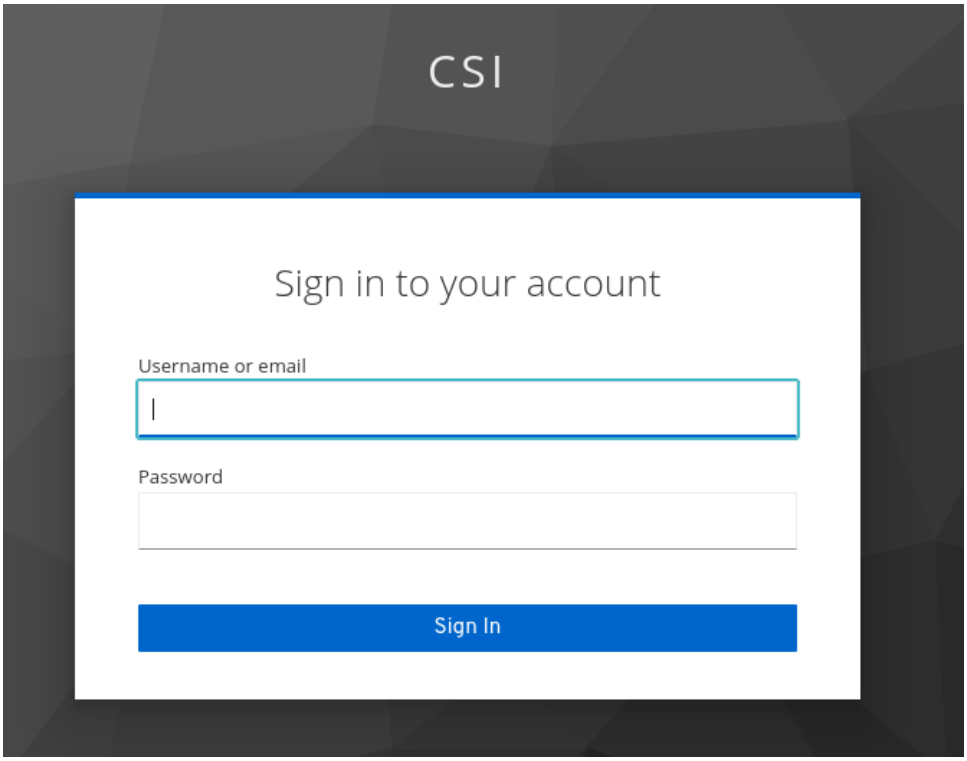


Figure 4.12 OAuth login page

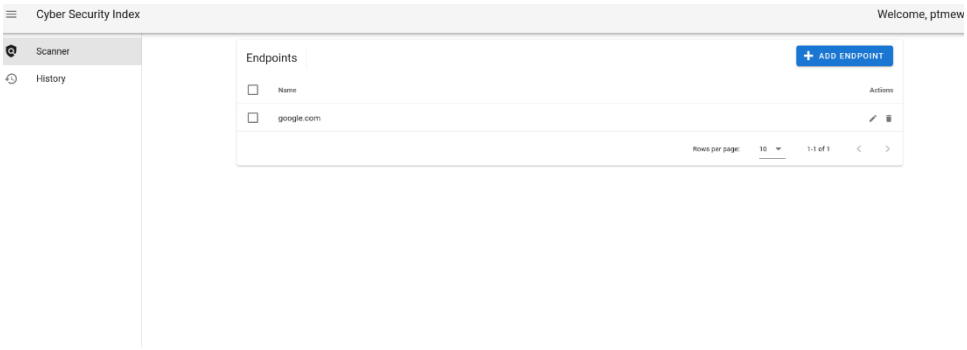


Figure 4.13 Company scanner page

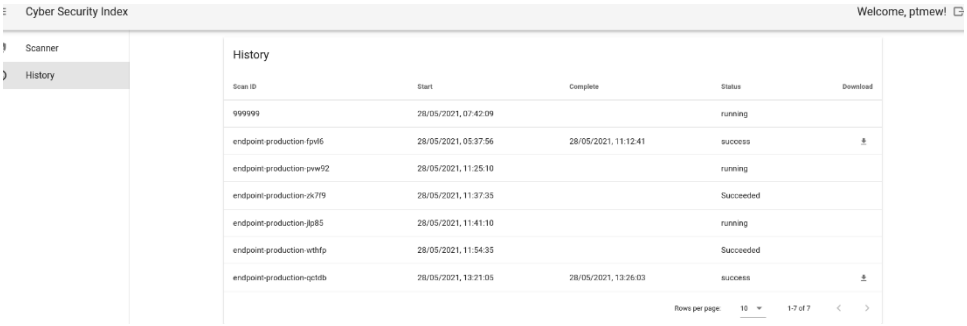


Figure 4.14 Company history page

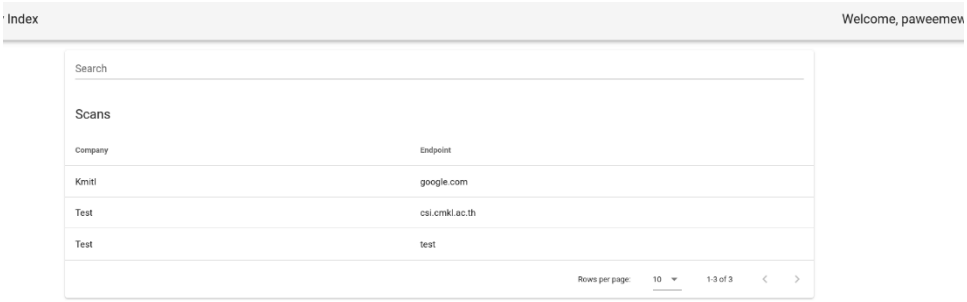


Figure 4.15 Admin page

4.2.3 Workflow

Argo Workflow can be written in the form of a template for workflows that are repeatedly invoked. Tools templates, a single endpoint scan template, and multiple endpoints scan template were created as shown in Figure 4.16. The Digital Ocean Container Registry was used to store the container images which templates could pull, as shown in Figure 4.21 The details of the single endpoint scan and multiple endpoints scan are shown in Figure 4.18 and Figure 4.19 respectively. The resulting logs are saved in the Digital Ocean Space as shown in Figure 4.20.

```
paweemew@MEW-LAPTOP:~$ argo -n argo template list
NAME
dnsmap
hydra
nmap
nmap-ip
scan-endpoint
scan-multi-endpoint
test
tool
validate-ip
```

Figure 4.16 Ago Template YAML file was submitted to Kubernetes and stored in the Argo namespace. It can be viewed via Argo CLI.

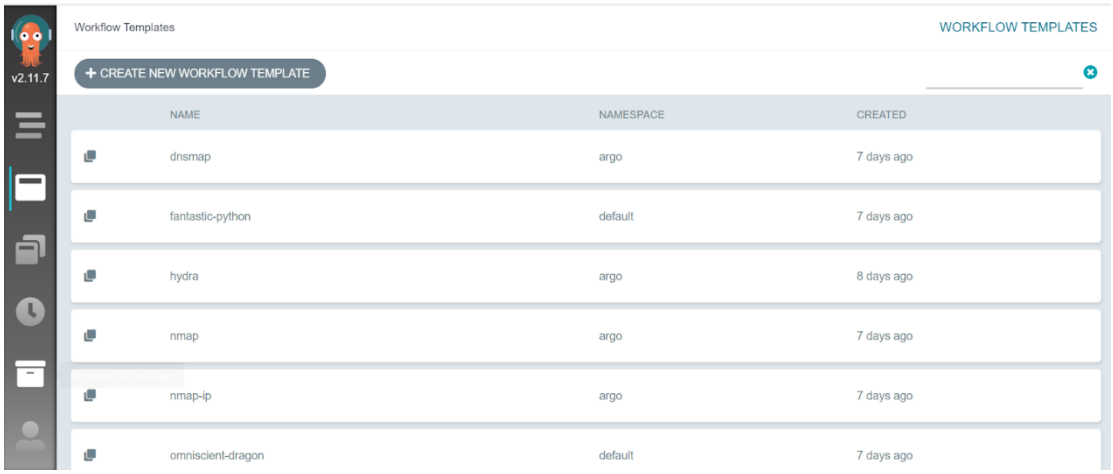


Figure 4.17 Ago Template YAML file was submitted to Kubernetes and stored in the Argo namespace. The template can be viewed as a GUI by using the port-forwarder command of Kubectl to port forward the service from Kubernetes to a localhost.

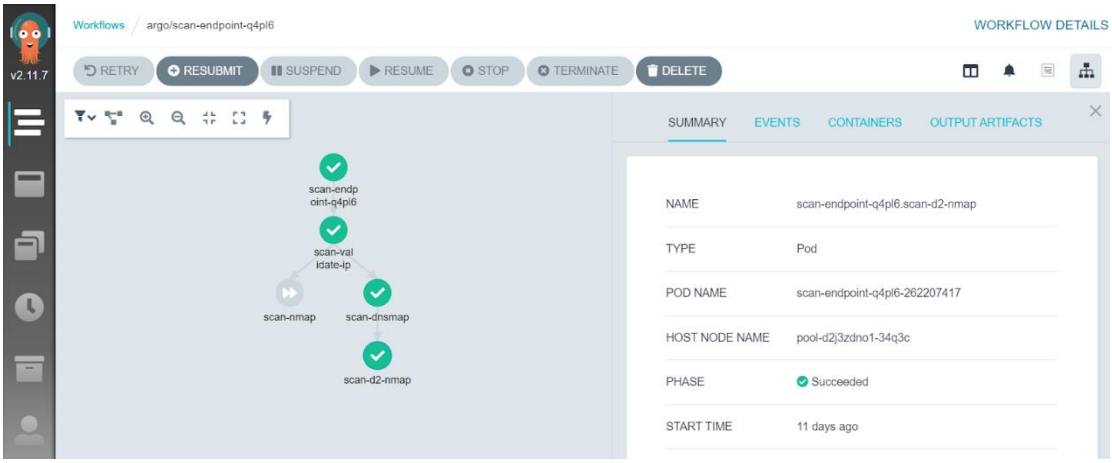


Figure 4.18 An Argo submitted scan for a single endpoint from the template on the Argo server client. The scan process can be undertaken in the form of submitting the template, the "scan one endpoint" option allows you to submit the form with the parameter of the target endpoint. On this figure, the input was a domain name, followed by Nmap.

.

Figure 4.19 An Argo submitted scan of multiple endpoints from the template on the Argo server client. This template was built on top of the single endpoint scan template.

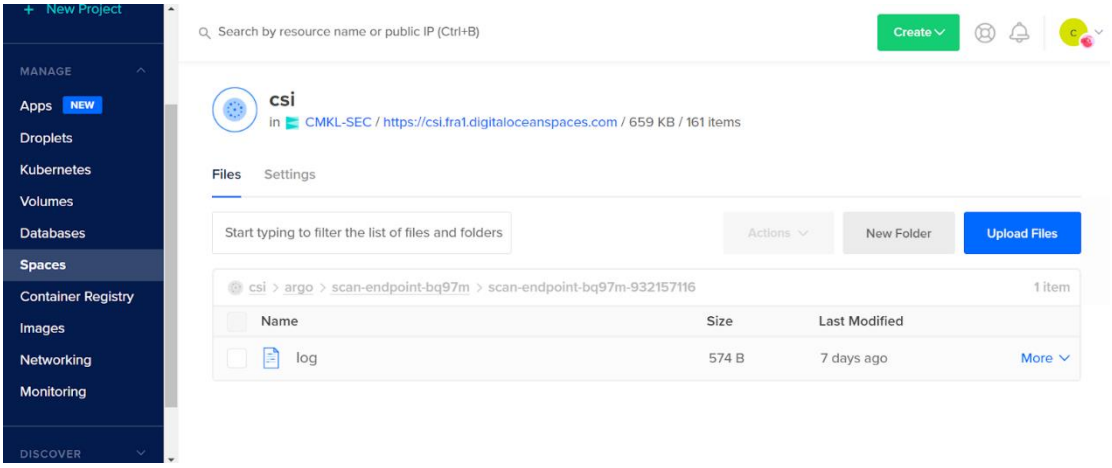


Figure 4.20 Digital Ocean Space stores log files from the scanning process.

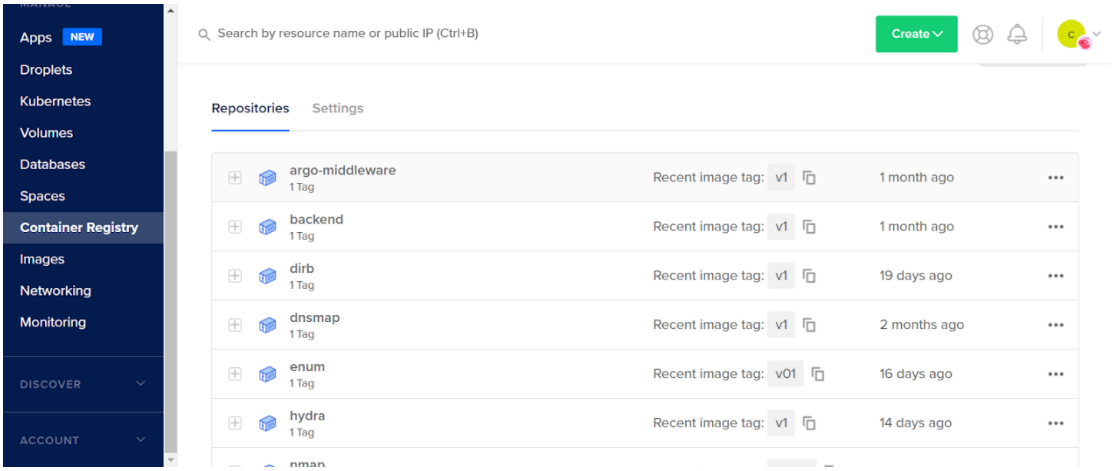


Figure 4.21 Digital Ocean Container Registry stores the private container images for Kubernetes Deployment and the Argo Workflow.

4.2.4 Filtering and Report Generator

The filtering service filters the log file and transforms it to the JSON format that the report generator can apply to the report template. Firstly, the scan log directory is mounted to the filtering service container and begins the process by recursively searching for the log file, then filtering it with a different method according to the tool name. At the end of the process, the

result is returned in JSON format following Figure 4.22. The JSON is recorded on the database for future usage on Carbone, the report generator.

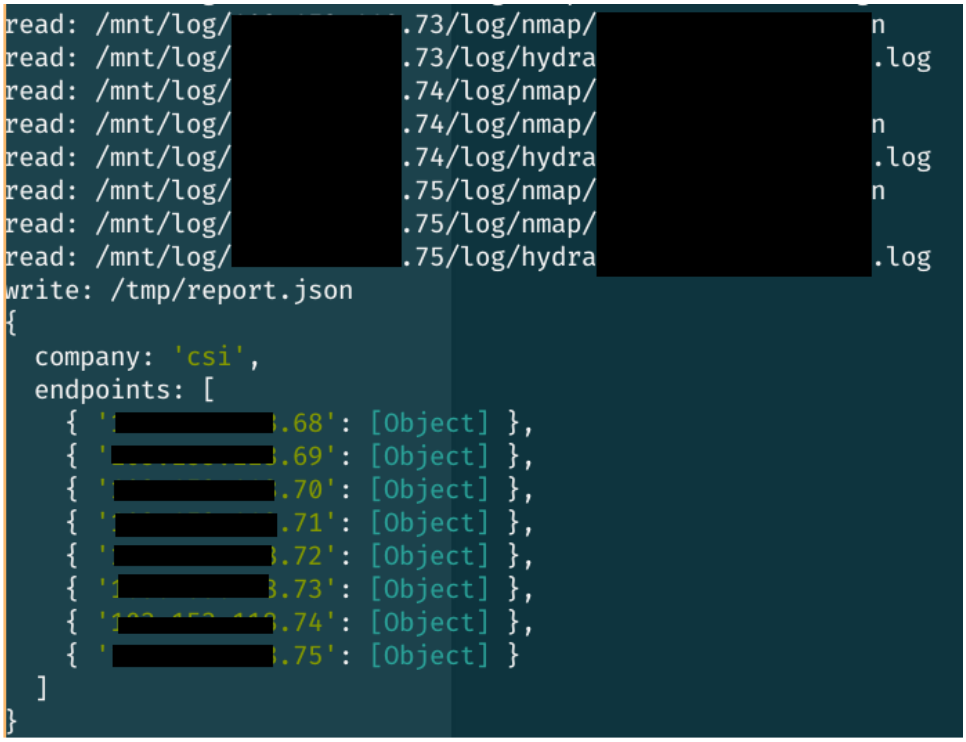


Figure 4.22 Output from the terminal showing the process of filtering the data from the log file.

4.3 Tool Validation

To determine if the chosen tools work with accurate results, validation of the tools is a necessity. The procedure is to run the tools against targets with known vulnerabilities and determine the accuracy of identifying vulnerabilities. Double-checking is also carried out by manually checking the result the tools produce and measuring the system resource usage.

4.3.1 DIRB

This tool was validated using the differences in 2 sets of testing on the testing Apache server. First, files were created in the directory with different

file permissions and settings, such as redirecting the page as shown in Figure 4.23. Secondly, files were created in a directory that returns a different response as shown in Figure 4.24.

File	Directory	Permission	Expect Response	Result Response
Root		644	200	200 OK
admin		0	403	403 Forbidden, 404 Not Found
admin.php		0	403	403 Forbidden, 404 Not Found
auth		666	200	200 OK
index.html		666	200	200 OK
	com2	0	Directory	403 Forbidden
	com1	644	Directory	No reponse (Directory found)
com1/1		644	200	200 OK
com1/2		644	200	200 OK
com1/3		644	200	200 OK
com3		644	200	302 Temporary Redirect

Figure 4.23 Testing files with different permissions

As shown in Figure 4.23, all files are empty except for com3 which contains a code to redirect to google.com. The expected HTTP response corresponded to the resulting HTTP response except for the directory com2. Directory com2 should be detected as a directory in the same way as directory com1.Instead, with the configuration of no permission (0) the result was 403 Forbidden.

Reponse	File	Result
100 Continue	100.php	100
101 Switching Protocols	101.php	101
103 Early Hints	103.php	500
200 OK	200.php	200
201 Created	201.php	201
202 Accepted	202.php	202
203 Non-Authoritative Information	203.php	203
204 No Content	204.php	204
205 Reset Content	205.php	205
206 Partial Content	206.php	206
300 Multiple Choices	300.php	300
301 Moved Permanently	301.php	301
302 Found	302.php	302
303 See Other	303.php	303
304 Not Modified	304.php	304
307 Temporary Redirect	307.php	307
308 Permanent Redirect	308.php	308
400 Bad Request	400.php	400
401 Unauthorized	401.php	401
402 Payment Required	402.php	402
403 Forbidden	403.php	403
404 Not Found	404.php	404 (There is an option not to show 404)
405 Method Not Allowed	405.php	405
406 Not Acceptable	406.php	406
407 Proxy Authentication Required	407.php	407
408 Request Timeout	408.php	408
409 Conflict	409.php	409
410 Gone	410.php	410
411 Length Required	411.php	411
412 Precondition Failed	412.php	412
413 Payload Too Large	413.php	413
414 URI Too Long	414.php	414

Figure 4.24 Files in the directory with different response

According to the experimental results, the response from the tool shows the same response from the .php files with a different return configuration as shown in Figure 4.24. The exceptions are 103 (Early Hints), 418 (I'm a teapot - this response is not common as it was intended for use on April fools day), and 425 (Too early - which returns 500 (Internal Server Error Response instead of the response from the configuration). From the sample, the tools are shown to have an accuracy of 94%.

4.3.2 SQLmap

The SQLmap was tested against a server with known SQL injection vulnerabilities to determine if the SQLmap could identify vulnerabilities

correctly. In this experiment, the Damn Vulnerable Web Application (DVWA) wasused as the target.

DVWA is a web application designed with vulnerabilities to help hackers test their skills legally. DVWA has 4 different security settings (Low, Medium, High, Impossible); the highest setting (Impossible) is meant to be vulnerability-free and cannot be hacked.

The results demonstrated in the figure below (Figure 4.25, Figure 4.26, Figure 4.27) verify that the SQLmap can accurately identify DVWA's vulnerabilities at all hackable levels and was able to retrieve information from DVWA's databases.

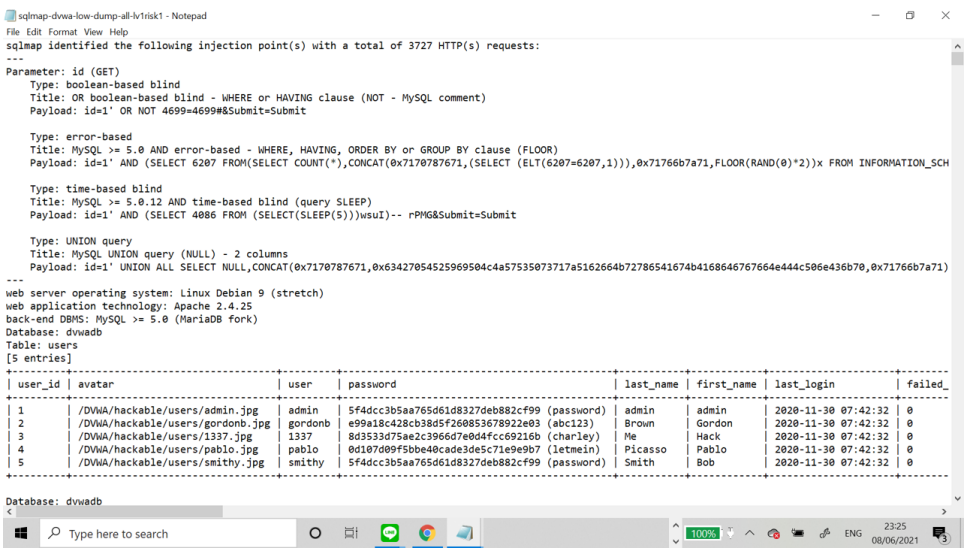


Figure 4.25 SQLmap result against DVWA on Low.

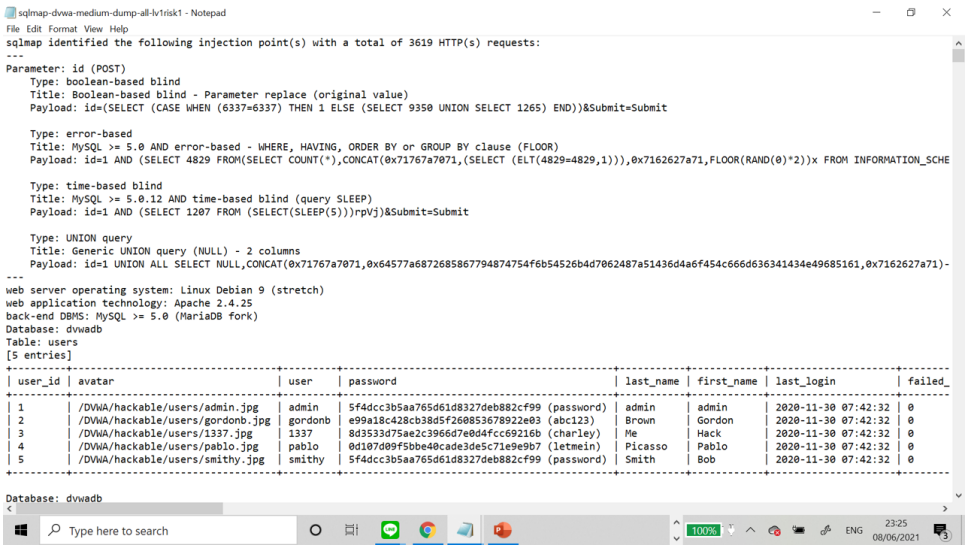


Figure 4.26 SQLmap result against DVWA on Medium

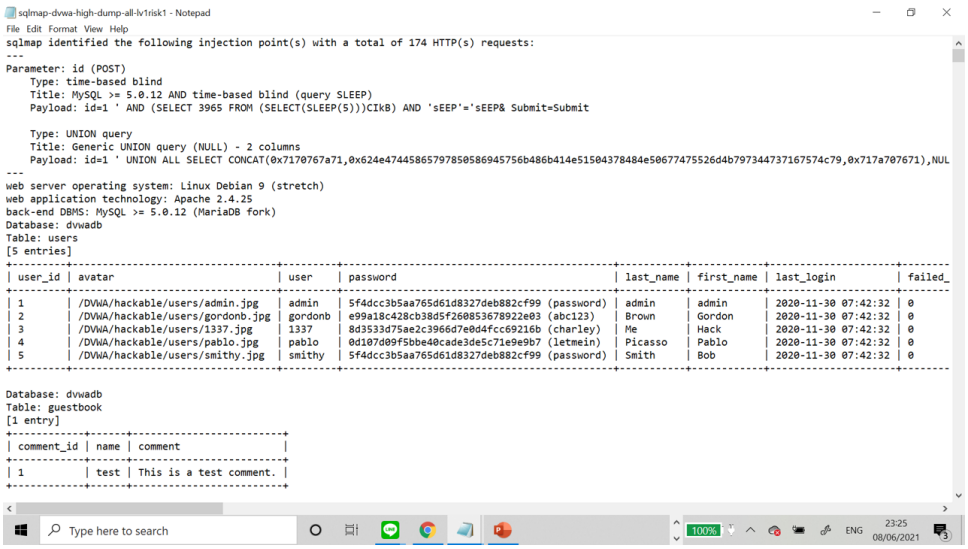


Figure 4.27 SQLmap result against DVWA on High.

4.3.3 Hydra

Hydra attacks by dictionary wordlist. The success rate of this tool was dependent on the consistency of the word list, as the length of the word list affects the processing speed. The first experiment uses a built-in password for each protocol generated by the Hydra, providing a small size word list with the

name related to the protocol. The test was conducted on three protocols, namely FTP, SSH, and Telnet, as shown in Figure 4.28. The built-in file could guess for the FTP and SSH but not for the telnet. The consistency of the wordlist and the length of the wordlist should be reconsidered. The second experiment tested hydra with the key-based authentication as shown in Figure 4.28. The result was immediately exiting the program due to the different system.

Files Input Entries(number)	Status	Protocol.	Port	User	Password	Result	Duration	CPU	Memory	Space Log Size
31	Open	FTP	21	user	pass	Succeeded	1 minute 36 seconds	26.71%	85%	698B
21	Open	SSH	22	local	ssh	Succeeded	2 minutes 0 seconds	28.24%	87%	697B
111	Open	Telnet	23	telnet	root	Failed	5 minutes 58 seconds	27.14%	86%	928B

Files Input Entries(number)	Status	Protocol.	Port	User	Password	Result	Duration	CPU	Memory	Space Log Size
21	Open	SSH	22	local	ssh	ERROR	11 seconds	31.36%	86%	417B

Figure 4.28 Hydra experiments result

4.3.4 Amass

The test was conducted against a known domain name with known subdomains (cmkl.ac.th). Every subdomain on the list was manually pinged to get their IP address and all of them were correct. The result did not cover all CMKL's subdomains since Amass managed to enumerate only 30 subdomains as shown in Figure 4.29.However, the actual domain names for CMKL includes more than 50 subdomains under CMKL.

4.3.5 Nmap

Nmap was validated by scanning the created instances for monitoring and configuration. In the experiment, there are 5 OSs to perform the validation including Ubuntu, Fedora, FreeBSD, CentOS, and Debian for cloud-based systems as shown below. The log file of a finished scan is shown in detail in Figure 4.29, including the information of ports opened, service name, and version, which are taken as useful information. In addition, other information besides host ports can be compared between setup instances and what Nmap has found. Although, the result shows the performance on the other four

sections (OS, Kernel, Fingerprinting, Encryption) is less accurate or unreliable.

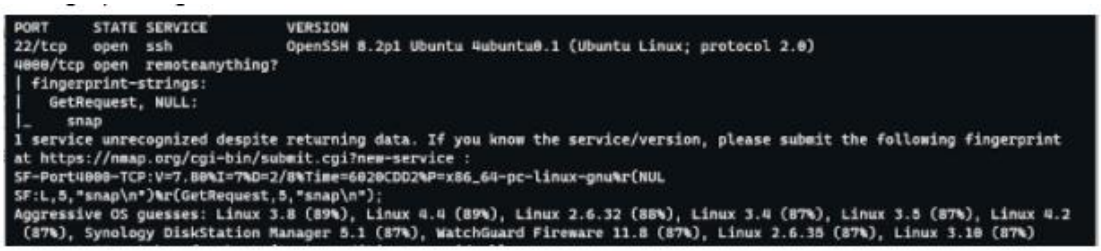


Figure 4.29 Nmap finding on Ubuntu instance.

	Setup	Scan Find
OS	Ubuntu 20.04	-
Kernel	LTS Linux 5.4.0	Linux 3.8 (89%), Linux 4.4 (89%)
Fingerprinting	Yes	No
Encryption	SHA-256	-
Active Port (LISTEN only)	22, 4000	22, 4000

Figure 4.30 Comparison in Ubuntu

	Setup	Scan Find
OS	Fedora 33	-
Kernel	Linux	Linux
Fingerprinting	Yes	No
Encryption	N/A	-
Active Port (LISTEN only)	22	22

Figure 4.31 Comparison in Fedora

	Setup	Scan Find
OS	FreeBSD 12.2	FreeBSD 11.x
Kernel	Linux	Linux
Fingerprinting	Yes	Yes
Encryption	N/A	-
Active Port (LISTEN only)	22	22

Figure 4.32 Comparison in FreeBSD

	Setup	Scan Find
OS	CentOS 8.3	-
Kernel	Linux	Linux
Fingerprinting	Yes	Yes
Encryption	N/A	-
Active Port (LISTEN only)	22	22

Figure 4.33 Comparison in CentOS

	Setup	Scan Find
OS	Debian 9	-
Kernel	Linux	Linux
Fingerprinting	Yes	Yes
Encryption	N/A	-
Active Port (LISTEN only)	22, 111	22, 11

Figure 4.34 Comparison in Debian

4.3.6 Searchsploit

Searchsploit is one tool to find vulnerabilities in software programs by querying the protocol name and version, to which it then displays the information fetched from the Searchsploit database. Only a set of limited protocols can be detected in the database. The result informs the

vulnerability's name and URL to the website of the database for more details, as shown in Figure 4.35 below.


```
{
    "SEARCH": "Memcached 1.5.5",
    "DB_PATH_EXPLOIT": "/opt/exploitdb",
    "RESULTS_EXPLOIT": [
        {"Title":"Memcached 1.5.5 - 'Memcrashed ' Insufficient Control of Network Message Volume Denial of Service With Shodan API","URL":"https://www.exploit-db.com/exploits/44265"},
        {"Title":"Memcached 1.5.5 - 'Memcrashed' Insufficient Control Network Message Volume Denial of Service (1)","URL":"https://www.exploit-db.com/exploits/44264"},
        {"Title":"Memcached 1.5.5 - 'Memcrashed' Insufficient Control Network Message Volume Denial of Service (2)","URL":"https://www.exploit-db.com/exploits/44254"}
    ],
    "DB_PATH_SHELLCODE": "/opt/exploitdb",
    "RESULTS_SHELLCODE": [  ]
}
```


Figure 4.35 Getting information on an outdated or unpatched program.

4.3.7 Resource Usage

Resource usage is measured by monitoring the graphs from the Kubernetes dashboard while running tools on the Argo workflow as shown in Figure 4.36.

Figure 4.36 The sample of droplet resource usage when scanning the endpoint.

Different tools have different resource usage and time duration. Figure 4.36 shows the VM data usage dashboard used for the measurement shown in Figure 4.37.

Nothing				
CPU: 20.93%				
MEMORY%: 64%				
Endpoints	**DNSMAP**	**NMAP**	**DIRB**	**XSSSNIPER**
188.166.231.190 (Juice Shop)		**ID: test-tool-bwtlv**	**ID: test-tool-ss7ld**	**ID: test-tool-wgmwb**
		Duration: 32s	**Duration: 2 minutes 57 seco**	**Duration: 31s**
		CPU:31.57%	**CPU: 25.63%**	**CPU:36.84%**
		Memory:69.70%	**Memory: 67%**	**Memory:64.35%**
		Space size:1.3KB	**Space size: 852B**	**Space size: 1.4KB**
159.65.14.172 (BWAPP)		**ID: test-tool-c65p9**	**ID: test-tool-hxlff**	**ID: test-tool-qklrm**
		Duration: 30s	**Duration: 11 seconds**	**Duration: 9s**
		CPU:27.46%	**CPU:25.88%**	**CPU:36.84%**
		MEMORY:69.70%	**MEMORY:65.57%**	**MEMORY:64.35%**
		Space size:1.7KB	**Space size:756B**	**Space size:1.6KB**
[redacted]	**ID:** test-tool-w8bw5	**ID: test-tool-b4gtt**	**ID: test-tool-hbvnr**	**ID: test-tool-z8m8w**
	Duration: 36s	**Duration: 44s**	**Duration: 1 hour 42 minutes**	**Duration: 9s**
	CPU: 32.92%	**CPU: 24.89%**	**CPU: 23.43% - 27.45%**	**CPU:34.17%**
	MEMORY: 64%	**MEMORY: 66%**	**MEMORY: 65-67%**	**MEMORY:65.12%**
	Space size: 0 B	**Space size: 22.1 KB**	**Space size: 1.2 KB**	**Space size: 1.5 KB**
[redacted]	**ID:** test-tool-5cqp8	**ID:** test-tool-s7mv5	**ID: test-tool-zldsr**	**ID: test-tool-chcr5**
	Duration: 27s	**Duration:** 47s	**Duration: 1 hour 16 minutes**	**Duration: 18s**
	CPU: 31.77%	**CPU:** 29.86%	**CPU: 27.72 %**	**CPU:34.17%**
	MEMORY: 64%	**MEMORY:** 64%	**MEMORY: 65 %**	**MEMORY:65.12%**
	Space size: 0 B	**Space size:** 2 KB	**Space size: 1.8 KB**	**Space size:1.5KB**

Figure 4.37 A sample of droplet data usage when scanning the endpoint. The run times varied from a few minutes to an hour depending on the characteristics of the endpoints.

4.4 System Evaluation

The evaluation of the system was carried out by testing the system against the real-world endpoints of a security company (section 4.4.1) and the organization endpoints (sections 4.4.2)

4.4.1 Security company endpoints

The CSAAS system IP had been whitelisted to reach the company's services during the time window provided, allowing permission to scan the company's endpoints. The company endpoints were scanned using Amass, Nmap, and Hydra. The following scan results apply to the company endpoint in addition to the DIRB tool. The company gave 2 endpoints which were domain names. For these, Amass ran first, however, it could not find the subdomains. Nmap was then run to find the protocol name SSL/HTTP-proxy, which was not supported by the current hydra, so it did not pass the parameter

to hydra for processing. During the given time window, DIRB was also tested, with the resulting log as shown in Figure 4.39.

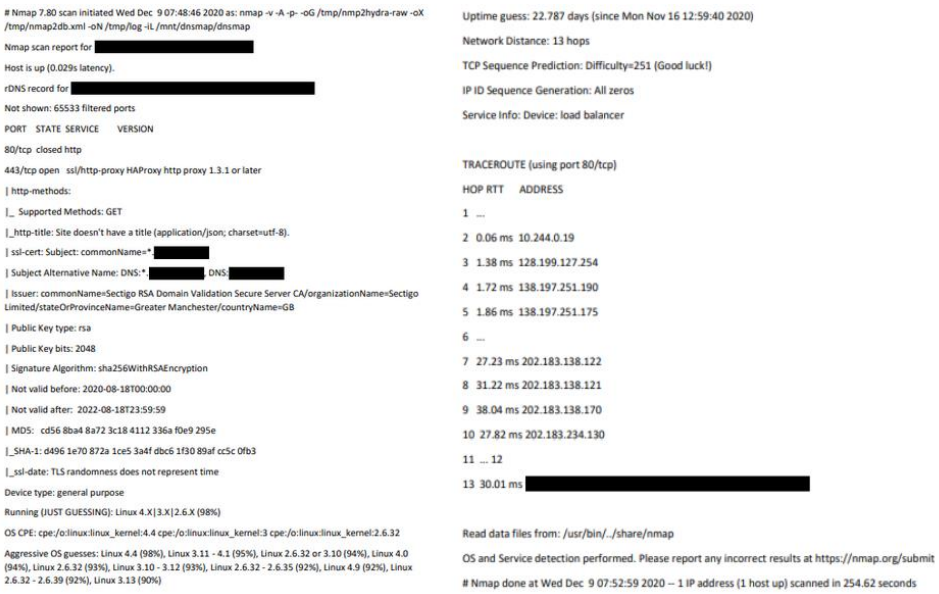


Figure 4.38 Nmap result on the company endpoint. Found the only port opened is 443 running on the HA Proxy load balancer version 1.3.1. This information can be used for checking on the outdated software. Moreover, it provides the size and algorithm of the key bit that can further be tested for outdated implementation. The OS found is Linux with a specific version that is based on guessing.

Figure 4.39 DIRB found that directories on the company endpoint mostly show HTTP response 302 which means the page has been moved and HTTP 400 which means the page exists, however, something went wrong with the request e.g., invalid request syntax on the client-side.

4.4.2 The organization endpoints

A scan was performed for the total list of 10 endpoints provided by the organization according to the flow shown in Figure 4.41. Six endpoints blocked the IP probe (Nmap could not run through). However, four endpoints continued through the assessment. The insights from scanning the organization endpoints were the names of products used in the system, which were then checked in the Exploit-Database to let the company know to update their software.

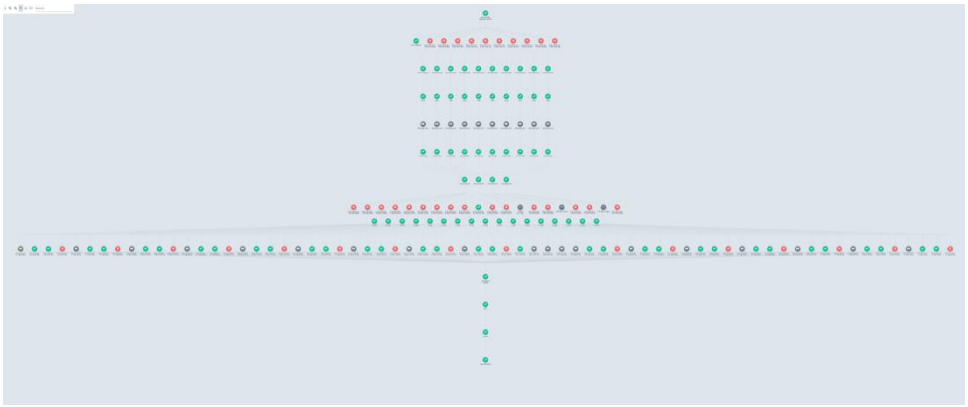


Figure 4.40 The Argo workflow scan was run from the Argo template with the parameters of a list of the endpoints and company names.

CHAPTER 5

CONCLUSION

5.1 Introduction

In this Chapter, we first summarize the work described in this report (section 5.2). Then we draw several conclusions about key parts of the work undertaken and further discussion in section 5.3.

5.2 Summary

The content below is a conclusion of each chapter's intro and summary.

Chapter 1 introduces the cybersecurity world and the importance of the project.

Chapter 2 reviews and considers the recent theory and advancements in security vulnerabilities and associated tools. Security testing tools were introduced, and security testing methods were described. The potential for an automated security assessment system was highlighted.

Chapter 3 describes the design of the system. The functions which support the requirements were then described in more detail, including the website, the Argo workflow, and the report generator. The implementation of the tools in the system is described.

Chapter 4 presents a series of tests which demonstrate that the tools chosen are accurate and the system can correctly identify vulnerabilities in target systems.

Chapter 5 provides the conclusions of this report.

5.3 Conclusions

The project aims to improve the security hygiene in the trading world. Investigating the problem, it was found that automating security testing would make the testing more continuous and could therefore help improve a broker company's system simultaneously.

The designed and implemented system can:

1. Get information from the target.
2. Automatically perform a network reconnaissance for the target system and generate a report.
3. Feedback a report to the broker company.

To maintain simplicity this project used the black-box network vulnerability assessment perspective and open-source tools e.g., Hydra, DIRB, Nmap, SQLmap, Amass, CVE, Searchsploit, and XSSsniper. All the tools run as a service on Kubernetes orchestrated by Argo workflow. After all services ran their necessary tests, the logs were filtered. Carbone used the filtered JSON file and generated reports for the users.

It is good to be able to identify the vulnerabilities in the system. However, this does not show overall hygiene to reach a higher audience e.g., the manager. Further work, such as creating the security index is needed to make a more reliable cyber world.

REFERENCES

[1] Alejandro Hernandez (2017, September 26). *Are You Trading Securely? Insights into the (In)Security of Mobile Trading Apps* [Blog Post]. Retrieved from: <https://ioactive.com/are-you-trading-securely-insights-into> [Accessed 26 September 2020]

[2] netsparker. *HIPAA Compliance Report* <https://www.netsparker.com/support/hipaa-compliance-report/#hipaa-compliance-report-sections> [Accessed 29 October 2020]

[3] OWASP. *OWASP Juice Shop* <https://owasp.org/www-project-juice-shop/> [Accessed 28 Nov 2020]

[4] KALI. *Our Most Advanced Penetration Testing Distribution, Ever.* <https://www.kali.org/> [Accessed 3 October 2020]

[5] OWASP. *OWASP WebGoat* <https://owasp.org/www-project-webgoat/> [Accessed 21 Nov 2020]

[6] RAPID7. *Understanding the reporting data model: Facts* <https://docs.rapid7.com/nexpose/understanding-the-reporting-data-model-facts/> [Accessed 10 November 2020]

[7] Purplesec**.** *Sample Vulnerability Assessment Report - Example Institute* **<**https://purplesec.us/wp-content/uploads/2019/12/Sample-Vulnerability-Assessment-Report-PurpleSec.pdf> [Accessed 20 October 2020]

[8] Microsoft (2020, September 21). *SQL Vulnerability Assessment helps you identify database vulnerabilities* <https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-vulnerability-assessment> [Accessed 17 October 2020]

[9] Infosec Insight (2020, july 1). *13 Vulnerable Websites & Web Apps for Pen Testing and Research* <https://sectigostore.com/blog/13-vulnerable-websites-web-apps-for-pen-testing-and-research/> [Accessed 17 October 2020]

[10] nixCraft (2007, june 7). *Understanding the Linux kernel – Anatomy of the Linux kernel*

<https://www.cyberciti.biz/tips/understanding-the-linux-kernel.html> [Accessed 26 October 2020]

[11] Acunetix, *Acunetix Web Application Vulnerability Report 2020* <https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2020/#methodology> [Accessed 9 October 2020]

[12] CVSS**.** *Common Vulnerability Scoring System SIG* <https://www.first.org/cvss/> [Accessed 24 October 2020]

[13] Kali Linux Tools Listing - *Offsec Services* <https://tools.kali.org/tools-listing> [Accessed 10 October 2020]

[14] Kulshekhar Kabra (2020, January 23), *Building Go Web Applications and Microservices Using Gin* <https://semaphoreci.com/community/tutorials/building-go-web-applications-and-microservices-using-gin>

[15] National Cyber Security Index. *NCSI Methodology* <NCSI :: Methodology (ega.ee)> [Accessed 16 December 2020]

[16] OWASP, *Top Ten Web Application Security Risks | OWASP* <https://owasp.org/www-project-top-ten/> [Accessed 16 December 2020]

[17] M. Liu, B. Zhang, W. Chen and X. Zhang, "A Survey of Exploitation and Detection Methods of XSS Vulnerabilities", *IEEE Access*, vol. 7, pp. 182004-182016, 2019.

[18] Mohd Amin Mohd Yunus, Muhammad Zainulariff Brohan, Nazri Mohd Nawi, Ely Salwana Mat Surin, Nurhakimah Azwani Md Najib, Chan Wei Liang, “Review of SQL Injection : Problems and Prevention”, INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION, 2018.

[19] Jai Puneet Singh “Analysis of SQL Injection Detection Techniques”, CIISE, Concordia University, Montreal, Qu’ebec, Canada.

[20] D. SON, "xsssniper: An automatic XSS discovery tool • Penetration Testing", *Penetration Testing*, 2021. [Online]. Available: https://securityonline.info/xsssniper-automatic-xss-discovery-tool/. [Accessed: 14 June 2021].

APPENDIX A

The example of report in every page

TABLE OF CONTENTS

01 EXECUTIVE SUMMARY

Cyber Security Index powered by CMKL is a conducted service where a black-box perspective security assessment, to perform a network reconnaissance in the given endpoints and show details, including ports, protocols, and DIRB afterward. Our project's purposes are for the company to be able to continuously access our service for continuous improvement for the company's security system and gain reputation.

02 SCOPE

This test scope is engaged on only a black-box perspective(zero-knowledge) with a blind security assessment test on the network area. Testing was performed on Wed Jun 9 15:02:36 2021 with industry-standard tools and frameworks, including, Amass, Nmap, Drib, and Argo.

END POINTS
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

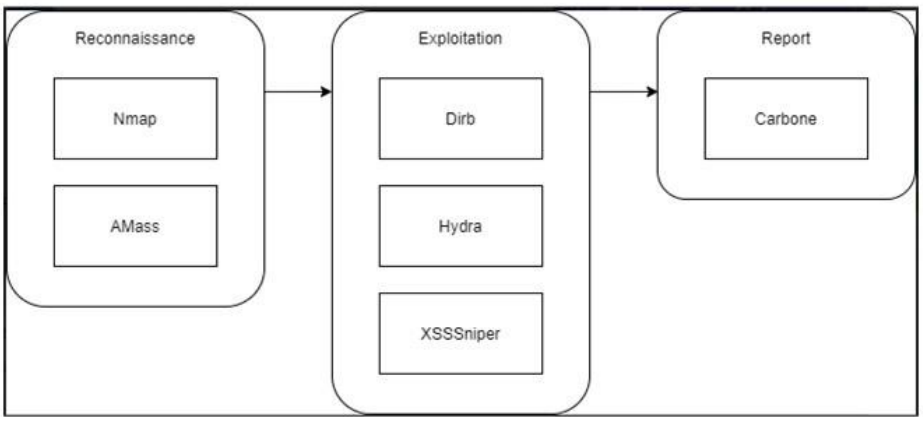
03 METHODOLOGY

As the starting point, the black-box or blinded scan was performed on the provided target server with following these steps:

1) Reconnaissance
2) Exploitation
3) Report

Retrieve information from it by using Nmap, for protocols, and an Amass, for the subdomain. Exploitation or search vulnerabilities in the system including find directory, brute force list of usernames and passwords.

.



04 FINDINGS

Port and Protocol Found

IP Address	Hostname	Protocol / Port
[redacted]		ssh:22http:80rpcbind:111tcpwrapped:179http:443 shell:514nfs_acl:2049http:5000sun-sr-https:6443pentbox-sim:6817unknown:6819http:8080http:8181https-alt:8443jetdirect:9100http:9253http:9353http:10254http:10256unknown:10257unknown:10259memcache:11211unknown:30627unknown:30956http:31351http:31443http:31500http:31559http:32000mountd:33837nlockmgr:35527mountd:38343mountd:54285:
[redacted]		ssh:22rpcbind:111tcpwrapped:179http:5000sun-sr-https:6443unknown:6819jetdirect:9100http:9253http:9353ssh:10004http:10256unknown:10257unknown:10259memcache:11211unknown:30627unknown:30956http:31351http:31443http:31500http:31559http:32000:34575:35907:36293:38633dgld:39381:39479dgld:39803:39937unknown:40011ssh:40122unknown:40513:40587:41873:45017:45763status:52609:
[redacted]		ssh:22rpcbind:111tcpwrapped:179nfs_acl:2049http:5000iperf3:5201sun-sr-https:6443pentbox-sim:6817unknown:6819jetdirect:9100http:9253http:9353http:10256unknown:10257unknown:10259unknown:10554unknown:10555memcache:11211:30627:30628:30956:31351:31443:31500:31559:32000:32037:32256mountd:37207nlockmgr:42919mountd:44499mountd:57449:
[redacted]		:
[redacted]		:
[redacted]		:
[redacted]		:
[redacted]		:
[redacted]		:
[redacted]		ssh:22rpcbind:111tcpwrapped:179http:443nfs_acl:2049http:5000iperf3:5201sun-sr-https:6443pentbox-sim:6817unknown:6819http:8443jetdirect:9100http:9253http:9353http:10256unknown:10257unknown:10259unknown:10554unknown:10555memcache:11211:30627:30628:30956:31351:31443:31500:31559:32000:32037:32256mountd:37207nlockmgr:42919mountd:44499mountd:57449:

05 FINDINGS

Path Found

URL BASE: 1

PATH	HTTP RESPONSE
[redacted]render/https://www.google.com	301
[redacted]/v2	301
[redacted]render/https://www.google.com	301
[redacted]/translations/render/https://www.google.com	301
[redacted]/translations/render/https://www.google.com	301
[redacted]/render/https://www.google.com	301