

PARALLELIZED CYBER RECONNAISSANCE AUTOMATION: A REAL-TIME AND SCHEDULED SECURITY SCANNER

MSc Thesis

Malek Malkawi

Graduate School of Engineering and Natural Sciences

Electrical-Electronics Engineering and Cyber Systems

August, 2022

PARALLELIZED CYBER RECONNAISSANCE AUTOMATION: A REAL-TIME AND SCHEDULED SECURITY SCANNER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF
ENGINEERING AND NATURAL SCIENCES
OF ISTANBUL MEDIPOL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL, ELECTRONICS ENGINEERING AND CYBER SYSTEMS

By

Malek Malkawi

August, 2022

PARALLELIZED CYBER RECONNAISSANCE AUTOMATION: A REAL-TIME AND SCHEDULED SECURITY SCANNER

By Malek Malkawi

02 Aug 2022

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Prof. Dr. Reda Alhajj (Advisor)

Prof. Dr. Selim Akyokuş

Prof. Dr. Tansel Özyer

Approved by the Graduate School of Engineering and Natural Sciences:

Prof. Dr. Yasemin Yüksel Durmaz

Director of the Graduate School of Engineering and Natural Sciences

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Signature :

Name, Surname: Malek Malkawi

ACKNOWLEDGEMENT

First and foremost, I would like to present my deepest appreciation to my supervisor Prof. Dr. Reda Alhajj for his endless support, guidance, motivation, and encouragement. It was such a great honor experience to work with him that I would never forget throughout my life.

I am honored that Prof. Dr. Selim Akyokuş and Prof. Dr. Tansel Özyer examined this dissertation. I owe them gratitude and respect for their time and insight.

I am grateful for the opportunity that has allowed me to make wonderful friends during my master's studies at Istanbul Medipol University. Many thanks to them and to all those who have supported me during this journey.

Last but not least, I dedicate this thesis to my family, to whom I owe a huge debt for their unwavering support and encouragement throughout my personal and academic lives despite far distances. When it comes to expressing my gratitude for their support, I'm always at a loss for words. Nonetheless, I am sure that without them, I would have accomplished nothing.

Malek Malkawi
August, 2022

CONTENTS

Page

ACKNOWLEDGEMENT ..... **iv**
CONTENTS ..... **v**
LIST OF FIGURES ..... **vii**
LIST OF TABLES ..... **viii**
ABBREVIATIONS ..... **ix**
ÖZET ..... **x**
ABSTRACT ..... **xi**
1. INTRODUCTION **1**
1.1. Related Work 2
1.2. Thesis Structure 4
2. THEORETICAL PART **5**
2.1. IT Security 5
2.2. Python programming language 6
2.3. Network scanner (Nmap) 6
2.4. Nmap script engine (NSE) 6
2.5. Flask Micro Web Framework 7
2.6. Restful API 7
2.7. Celery Task Queue 7
2.8. RabbitMQ Message Broker 9
2.9. Flower Monitoring Tool 9
3. EXPERIMENTAL PART **11**
3.1. General Structure And Workflow 11
3.2. Methods And Implementation 13
3.2.1. Host discovery 14
3.2.2. Port scanning 14
3.2.3. Service and version enumeration 15
3.2.4. OS detection 16
3.2.5. Vulnerability detection 16
3.2.6. Main Endpoint - Scan 17
3.2.7. Code packaging and structuring 18
3.2.8. Schema and data validation 18
3.2.9. Scanning options and workflow 19
3.3. Test Cases 19
3.3.1. Metasploitable-2 19
3.3.2. Bee-box-v1.6 20
3.3.3. Metasploitable-1 20
3.3.4. BTRsys1 and BTRSys2.1 21
3.3.5. g0rmint 21
3.3.6. Hades-v1.0.1 21
3.3.7. vulnVoIP 21
3.3.8. VPLE 21
3.3.9. Tr0ll2 22
3.3.10. sick0s1.1 22
3.3.11. MorningCatch 22

3.3.12. NETinVM UML 2016 22
3.3.13. hackxor1 22
3.3.14. Brainpan2, CySCA2014InABox, DonkeyDockerv1.0, GoatseLinux-1.0-VM, UltimateLAMP-0.2 and w1r3s.v1.0.1 23
4. RESULTS AND DISCUSSION 24
4.1. Results 24
4.2. Discussion 29
5. CONCLUSION AND FUTURE WORK 34
5.1. Summary And Conclusion 34
5.2. Future Research Directions 35
BIBLIOGRAPHY 36
CURRICULUM VITAE 40

LIST OF FIGURES

Figure 2.1: CIA Triad 5
Figure 2.2: API Architecture 8
Figure 2.3: Celery Architecture 8
Figure 2.4: RabbitMQ Message Broker Screenshot during the Testing Process 9
Figure 2.5: Flower Monitoring Tool Screenshot during the Testing Process 10
Figure 3.1: General Workflow and Structure 12
Figure 3.2: General Structure - Initial Version 13
Figure 3.3: Host Discovery Methods 14
Figure 3.4: Port Scan Methods 15
Figure 3.5: OS Detection 16
Figure 3.6: Project Structure 20
Figure 4.1: Total time for each process 30
Figure 4.2: Total time for each option 31
Figure 4.3: Enhancement Ratio for each option from different aspects 32
Figure 4.4: Port Scan and Vuln. Detection Relationship 33

LIST OF TABLES

Table 4.1: Results of Test Cases *the Numbers show the amount of vulnerabilities found after service-version enumeration 25
Table 4.2: Total Elapsed Time in Minutes 26
Table 4.3: Test Cases based Previous and Enhanced Version Scanning Time Comparison *The enhanced version is fully automated and parallelized..... 27
Table 4.4: Test Cases based Previous and Enhanced Version found vulnerabilities Comparison 28
Table 4.5: New Scanning Options Time Results 29

ABBREVIATIONS

IP : Internet Protocol
CIA : Confidentiality, Integrity, and Availability
API : Application Programming Interface
REST : REpresentational State Transfer
IoT : Internet of Things
Nmap : Network Mapper
AMQP : Advanced Message Queuing Protocol
IT : Information Technology
PenTest : Penetration Testing
NSE : Nmap Scripting Engine
CVE : Common Vulnerabilities and Exposures
IDS : Intrusion Detection System
TCP : Transmission Control Protocol
UDP : User Datagram Protocol
OS : Operating system
SSL : Secure Sockets Layer
SQL : Structured Query Language
XSS : Cross-Site Scripting
WWW : World Wide Web
HTTP : HyperText Transfer Protocol
HTTPS : Hypertext Transfer Protocol Secure
URL : Uniform Resource Locators
JSON : JavaScript Object Notation
ARP : Address Resolution Protocol
PING : Packet Internet or Inter-Network Groper
LAN : Local Area Network
ICMP : Internet Control Message Protocol
ACK : Acknowledgement
SYN : Synchronize
RST : Reset
FTP : File Transfer Protocol
SV : Service-Version
RFC : Request for Comments
DB : Database
PEP : Python Enhancement Proposal
VoIP : Voice over Internet Protocol
OSCP : Offensive Security Certified Professional
OWASP : The Open Web Application Security Project
VM : Virtual Machine
DOS : Denial-of-Service
DDOS : Distributed Denial of Service

PARALELLEŞTIRILMIŞ SIBER KEŞIF OTOMASYONU: GERÇEK ZAMANLI VE ZAMANLANMIŞ GÜVENLIK TARAYICISI

ÖZET

Malek Malkawi

Elektrik-Elektronik Mühendisliği ve Siber Sistemler, Yüksek Lisans

Tez Danışmanı: Prof. Dr. Reda Alhajj

Ağustos, 2022

Teknolojinin olağanüstü gelişimi, hala ulaşılması zor olan gerekli bilgi güvenliği düzeyin ulaşmanın önemini arttırmıştır. Son zamanlarda, ağ ve web uygulaması saldırıları sıklıkl meydana gelmekte ve sistemlerdeki mevcut güvenlik açıkları tarafından gizli verileri çalınmasına neden olmaktadır. Saldırıların arkasındaki en dikkat çeken neden, savunmasız servislere sahip açık portlardır. Bu, CIA (Gizlilik, Bütünlük ve Kullanılabilirlik) Üçlü Modelinin kırılmasına neden olur. Sızma testi, sisteme yönelik olası tehditleri v potansiyel saldırıları doğru bir şekilde tespit etmek için gerçek hayatta kullanılan teme tekniklerden biridir ve bilgisayar korsanlarının saldırı gerçekleştirmesinin ilk adımı bilg toplamadır. Bu çalışmada, gerçek dünyadaki sorunları gidermek amacıyla, sızma test sırasında ve sistem yöneticileri tarafından kullanılabilecek aktif bilgi-toplama aşamas için faydalı bir şema sunuyoruz. Uygulanacak bir güvenlik motorunun ilk özelliği olacak. Paralelleştirmeye dayalı otomatik bir güvenlik tarayıcısıdır. Şemamızda sağlana kurallara bağlı olarak, yalnızca gerçek hayattaki durumlar için kolay entegrasyon sağlaya bir RESTful API'ye değil, aynı zamanda duruma bağlı olarak bilgileri yüksek doğrulukla toplamak için Nmap dayalı aktif bilgi toplama aşamasında paralel olarak gerçe zamanlı ve planlı sistem taramalarını destekler. Başlangıçta gelişmiş mesaj kuyruklam protokolünü (AMQP) uygulayan message broker yazılımının (RabbitMQ) entegrasyon ile birlikte kullanıcı, anında özelleştirilmiş taramalar oluşturma ve ilgili sonuçları kontrol etme yeteneğine sahiptir. Bu özellikler, Celery Workers'in, çoklu işlem ve görevleri eşzamanlı yürütülmesini gerçekleştirmek için dağıtılmış ileti geçişine dayanan, eşzamansız bir görev kuyruğu kullanmasına bağlıdır. Sistem sızma test uzmanları, BT departmanları ve sistem yöneticileri tarafından sistemlerini izlemek ve kritik tehditlere karşı yükse güvenlik ve anlık alarmlar vermek için kullanılabilir. Otomatik bir IP ve port taraması, servis-versiyon ve güvenlik açığı tespit sistemi, önerilen şema projesinin çekirdeğini oluşturur. Bu tekniğin doğruluğu ve etkinliği, gerçek dünya olaylarına dayanan çeşitli tes senaryolarıyla kanıtlanmıştır. Bir sunucuyu taramanın ve güvenlik açıklarını tespit etmenin ortalama süresi 1,7 dakikadır. Güvenlik açıklarının sayısından bağımsız olarak, her bir açık port için süre artışı sadece yaklaşık 12 saniyedir. Test senaryosu olarak kullandığımız 20 farklı sunucuyu taramak için ortalama çalışma süresi 35 dakika olup, paralelleştirmenin gücü ile geçen süreyi %90,80 oranında azaltarak 4 dakikaya indirebildik.

Anahtar sözcükler: Güvenlik Tarayıcısı, Siber Keşif, Güvenlik Açığı Değerlendirmesi, Sızma Testi, Otomasyon, Paralelleştirme.

PARALLELIZED CYBER RECONNAISSANCE AUTOMATION: A REAL-TIME AND SCHEDULED SECURITY SCANNER

ABSTRACT

Malek Malkawi

MSc in Electrical, Electronics Engineering and Cyber Systems

Advisor: Prof. Dr. Reda Alhajj

August, 2022

The extraordinary advancement of technology has increased the importance of achieving the required level of information security, which is still difficult to achieve. Recently, network and web application attacks have occurred frequently, causing confidential data to be stolen by the available vulnerabilities in the systems. The most prominent reason behind the attacks is open ports with vulnerable services. This causes the CIA (Confidentiality, Integrity, and Availability) Triad Model to break. Penetration testing is one of the key techniques used in real life to accurately detect possible threats and potential attacks against the system, and the first step for hackers to conduct attacks is information collection. In this work, with the aim of relieving real-world concerns, we present a useful schema for the active information-gathering phase that can be used during penetration testing and by system administrators. It will be the first feature of a security engine to be implemented. It is an automated security scanner based on parallelization. It supports real-time and scheduled system scans in parallel in the phase of active information gathering based on not only a RESTful API allowing easy integration for real-life cases, but also a Network Mapper (Nmap) to collect the information with high accuracy depending on the provided rules in our schema. With the integration of the message-broker software (RabbitMQ) that originally implemented the advanced message queuing protocol (AMQP), the user has the ability to create instant customized scans and check the related results. These features depend on Celery workers using an asynchronous task queue, which is reliant on distributed message passing to perform multiprocessing and concurrent execution of tasks. The system can be used by penetration testers, IT departments, and system administrators to monitor their systems and grant high security and instant alarms on critical threats. An automated IP and port scanning, service-version enumeration, and security vulnerability detection system are the core of the proposed scheme project. The accuracy and efficiency of this technique have been demonstrated through a variety of test cases based on real-world events. The average time of scanning a server and detecting vulnerabilities is 1.7 minutes. Regardless of the number of vulnerabilities, the increase in time for each open port is just about 12 seconds. The average run time for scanning the 20 distinct servers that we have used as test cases is 35 minutes, and with the power of parallelization, we could decrease the elapsed time by 90.80% to 4 minutes.

Keywords: Security Scanner, Cyber Reconnaissance, Vulnerability Assessment, Penetration Testing, Automation, Parallelization.

CHAPTER 1

1. INTRODUCTION

Internet technology has proliferated and progressed at a breakneck pace. The internet connects people's lives, but it also brings with it some threats, such as cyber-attacks [1]–[3]. That may result in data leaking or business interruption, leading to significant financial losses [4] and breaking the CIA (Confidentiality, Integrity, and Availability) Triad, which is the main principle of information security. Penetration Testing (PenTest) is one of the methods used to discover possible attacks and hidden threats. One of the most important aspects of testing is identifying vulnerabilities in a timely manner [5]–[8]. Existing vulnerabilities mean that data processed by web apps is not secure and exposed to the existing exploitation scenario [9]–[11].

Manual testing approaches make it difficult to detect different types of application vulnerabilities. The resources allocated for testing are limited and not only the complexity of modern web applications is increasing but also the number of problems to be solved in terms of the integration among the used technologies. That's why regular and automated vulnerability assessments help to reduce the risks of eventual intrusion and violations of the systems' data integrity, availability, and confidentiality [12]–[19].

Reconnaissance is a fundamental part of the penetration testing methodology in which a professional penetration tester gathers as much information about a target machine or network as possible prior to conducting a guided test in search of vulnerabilities. This process consists of three phases: fingerprinting, scanning, and enumeration.

In this thesis, we are going to propose an parallelized security scanner. It is an API-based automated network scanner including host discovery, open port scanner, service-version enumerator, and vulnerability detection approach using the Network Mapper (Nmap) Scripting Engine (NSE) integrated with VulScan and Nmap-vulners modules databases to

find Common Vulnerabilities and Exposures (CVE). VulScan and Vulners are NSE scripts that make use of publicly available services to provide relevant information about security vulnerabilities [20]. Parallel real-time and scheduled system scans are supported in this schema. With the integration of RabbitMQ message-broker, which originally developed the advanced message queuing protocol (AMQP), the user can create instant customized scans and retrieve the relevant results. These features rely on Celery workers, which use an asynchronous task queue to perform multiprocessing and concurrent task execution in the main endpoint of our RESTful-API server.

The project adheres to Python Enhancement Proposal to enrich readability, as well as modules and packaging, in order to achieve the project's purposes in the most efficient manner feasible. In the final version of our project, a distinct schema has been established for each endpoint to optimize the connectivity. Before being processed, Marshmallow will check the inbound request for compliance. In this work, we have proposed a solution that can easily be integrated with the current systems to solve the difficulties faced while switching from one tool to another to complete the process of scanning as the first feature of an automated security engine. The primary goal is to close the gap by automating the network scanning and vulnerability assessment processes for small and large networks. At the same time, it gives the opportunity for administrators and researchers to prove that a host is vulnerable to certain attacks or not by automated periodic scans. Moreover, the test case outcomes exceeded our expectations. We nearly covered all essential network architectures based on real-life scenarios. The outcomes were significantly better than any manual method. We were able to quickly identify weaknesses with high accuracy.

1.1. Related Work

Reconnaissance is the process by which a hacker acquires information about a target before launching an attack or penetration test. It is carried out in stages prior to exploiting or patching system vulnerabilities. There are various tools for doing specific parts of this process. However, research towards automatic and comprehensive tools is still in its infancy.

Researchers in [21] demonstrate traffic accountability and the time required to perform a specific task during active scanning with the Nmap tool, as well as propose solutions

for dealing with huge volumes of hosts while preserving network traffic and task time. Nmap is also used by certain research groups only as a penetration tool to test intrusion detection systems' (IDS) performance and discover the related effects [22]. In [23] the authors concentrate on Nmap's operating system recognition feature and use it to acquire information about the target OS by matching IP/TCP stacks with the tool's built-in fingerprint database. In [24] ZMap has been developed, which is an internet-scale network scanner that scans single packets quickly. It can be used with ZGrab, a related project that supports stateful application-layer handshakes, grabs banners, and finds Heartbleed vulnerabilities.

Schagen et al. (2018) developed an automated network vulnerability scanning method that employs safe patch fingerprinting. This approach automates input detection and differentiates vulnerabilities across non-patched and patched systems quickly. Furthermore, this method allows specialists to scan and safeguard networks efficiently. In order to ensure the safe and ethical execution of patches, inputs that are suspected to be somewhat malicious are denied [25]. Roy et al. (2017) developed a Java-based command line application that locates and stores an organization's footprint and looks for sensitive information, such as data repositories, to enhance an organization's vulnerability assessment process [26]. To determine the correlation between attacks and port scans, Panjwani, S. et al. [27] employ an experiential technique. To test the theory, they ran trials for 48 days on two target computers on a widely used subnet.

In [28], Jensen J. Zhaoet et al. performed research to evaluate the security of US e-governance sites. Though there were a few best practices in place, such as the usage of SSL and a firewall, areas such as the clarity of confidentiality statements and open port 80 were discovered, and viable steps for strengthening e-governance security were suggested. P. LaRoche et al. [29] presented their research into using generic programming to evolve valid TCP packets, conduct a variety of port scans through an IDS, and remain covert in their activity. In [30], McPherson, J.et al. describe a system called "portvis" that analyzes coarsely detailed security-related network data of a basic sort and visualizes security-related events of interest. The authors in [20] discuss some of the challenges that arise while analyzing and identifying current web application vulnerabilities, as well as some potential solutions.

The study in [31] seeks to conduct a survey of typical consumer computer devices, such as start devices, in order to identify open ports and, as a result, vulnerabilities that could lead to potential attack targets. In [32], the authors describe a scanner that detects SQL Injection, Cross-Site Scripting (XSS), CRLF Injection, and Open Redirect vulnerabilities in web applications. It also has a simple port scanner and a web crawler module that helps find other services running on the web server.

1.2. Thesis Structure

In **Chapter 1**, we demonstrate the literature review and the related work. In **Chapter 2**, we introduce the methodology and the main components used in the research. Then, we illustrate the details of the implementation and the applied approach in **Chapter 3**. **Chapter 4** shows the results we have obtained and the test cases' scenarios and the discussion of the work. Finally, the conclusion and future work are explained in **Chapter 5**.

CHAPTER 2

2. THEORETICAL PART

2.1. IT Security

Computer security, also known as cybersecurity or IT security, is the safeguarding of computer networks and systems against unauthorized access, or damage to their hardware, software, or electronic data, as well as manipulating the services they provide [33]. Due to the growing reliance on computer systems, the Internet, and wireless network standards, as well as the proliferation of "smart" devices that make up the IoT, the field is becoming increasingly important. Its main purpose is to protect the system's CIA triad [34], [35]. CIA triad is described in **Figure 2.1** from [36].



Figure 2.1: CIA Triad.

A flaw in design, function, or implementation is referred to as a vulnerability. The CVE database contains the majority of the vulnerabilities that have been discovered [37]. Vulnerabilities can be detected using automated tools or manual customized approaches. Understanding the types of potential attacks that can be attempted against a computer system is critical to its security. Backdoors, Denial of Service, eavesdropping, phishing, privilege escalation, reverse engineering, social engineering, and malware are some of the most well-known threats.

2.2. Python Programming Language

Python is a high-level programming language with a design philosophy that prioritizes readability. Its features and object-oriented approach are designed to help programmers write clean code for both small and large-scale projects. Besides, it's a beneficial language for cybersecurity since it is capable of doing a wide variety of cybersecurity tasks, such as malware analysis, scanning, and penetration testing [38], [39]. Those are some of the reasons why we have decided to write our project with it.

2.3. Network Scanner (Nmap)

Nmap (Network Mapper) is an open-source network scanning and discovery tool widely used for scanning the state of a target host [40]–[42], and it provides comprehensive scan types and firewall evasion methods. Each technique can be customized and made to be as noticeable or as inconspicuous as possible. It can be used to find open ports, communication protocols (TCP/UDP), services and versions used on each port, as well as vulnerabilities on a remote device. The data acquired can be used to improve the system in terms of preventing future attacks. Our schema relies on Nmap algorithms to gather the required information regarding the target [41], [43].

2.4. Nmap Script Engine (NSE)

One of Nmap's most vital and flexible features is the Nmap Scripting Engine (NSE). It allows users to build simple scripts to automate a wide range of networking functions using a programming language called Lua. Those scripts are run in parallel, with the speed and

efficiency that Nmap is customized for. Users can use the built-in scripts of Nmap's library or develop their own to fulfill their specific needs. NSE's key tasks include network discovery, more sophisticated version detection, vulnerability detection, and backdoor detection. It can even be used for vulnerability exploitation [41], [44].

2.5. Flask Micro Web Framework

Flask is a Python-based micro-framework in that does not necessitate the usage of any specific tools or libraries. It doesn't have a database abstraction layer, form validation, or any other components that rely on third-party libraries to do typical tasks. When it comes to designing web applications, Flask gives developers a lot of options. It includes tools, libraries, and mechanisms that allow you to build a web application, but it doesn't impose any dependencies or tell the user how the project should look [45].

2.6. Restful API

REpresentational **S**tate **T**ransfer is a software architectural style that was developed to help in the design and development of the WWW's architecture. REST establishes a set of guidelines for how an Internet-scale distributed hypermedia system, such as the Web, should be designed. As shown in **Figure 2.2**, REST APIs offer a lot of flexibility, which is one of their biggest benefits. REST can accommodate many sorts of calls, return diverse data formats, and even change fundamentally with the correct implementation of hypermedia because data is not linked to resources or functions. Data and functionality are considered resources in the REST architectural style, and they are accessible via Uniform Resource Identifiers (URIs). The most commonly used protocol is HTTP. Using such an architecture makes projects more appropriate for scaling up and integrating with other systems. All of these characteristics contribute to the simplicity, lightness, and speed of RESTful applications [46].

2.7. Celery Task Queue

Celery is a Python-based open-source asynchronous task queue that uses distributed message passing to divide work across multiple threads or processors. Any language can be

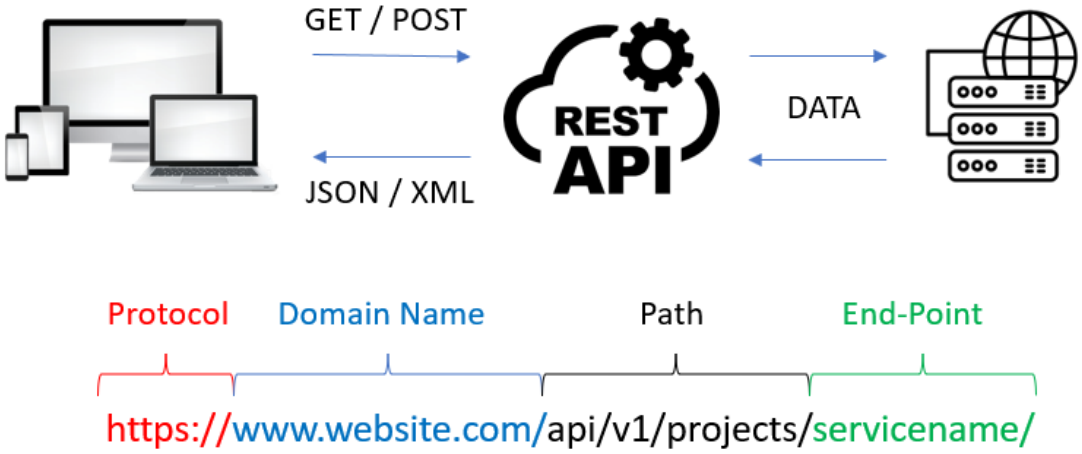


Figure 2.2: API Architecture.

used to implement the protocol. The input to a task queue is a task, which is a unit of work. Using multiprocessing, eventlet, or gevent, these tasks are carried out concurrently on one or more worker nodes. Celery communicates with clients and workers via messages, with a broker acting as a middleman. RabbitMQ and Redis are the suggested message brokers. To begin a job, the client adds a message to the queue, which is then delivered to a worker by the broker as shown in **Figure 2.3**. Multiple workers and brokers can be used in a Celery system, allowing for high availability and horizontal scaling. Celery enables scheduling tasks via Celerybeat and real-time operations via Celery workers [47].

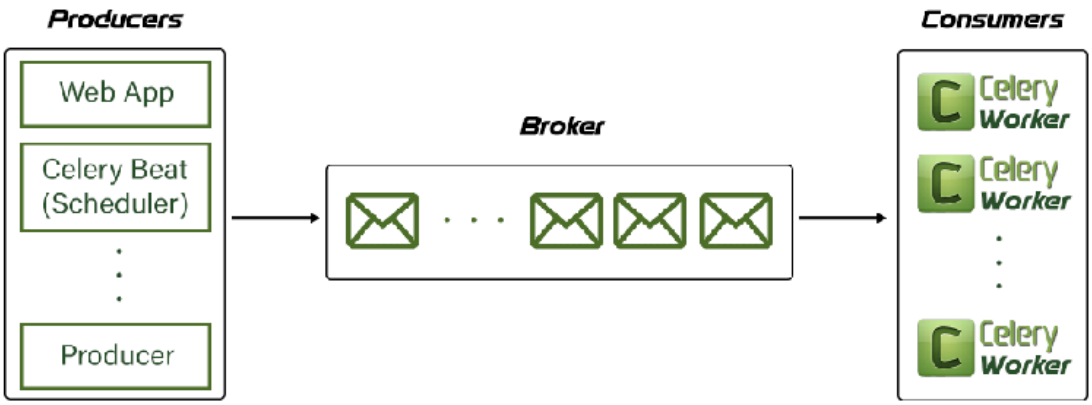


Figure 2.3: Celery Architecture.

2.8. RabbitMQ Message Broker

RabbitMQ is an open-source message broker developed in the Erlang programming language, which was designed to implement the Advanced Message Queuing Protocol (AMQP). Message brokers are responsible for transporting messages and providing producers with the opportunity to begin working on a different task, as shown in **Figure 2.3**. The sender does not have to wait for the message to be received by the recipient. It is a versatile system that can be used to reduce web application server load and delivery times. Persistence, delivery acknowledgments, publisher confirms, and availability are the capabilities available in RabbitMQ to balance performance and reliability. On a local network, several RabbitMQ servers can be clustered together to form a single logical broker, and queues can be mirrored in the cluster to guarantee that messages are safe even if the hardware fails. RabbitMQ supports messaging via a variety of messaging protocols and has an easy-to-use management user interface that allows the user to monitor the message broker [48]–[50]. **Figure 2.4** shows the management panel of the broker.

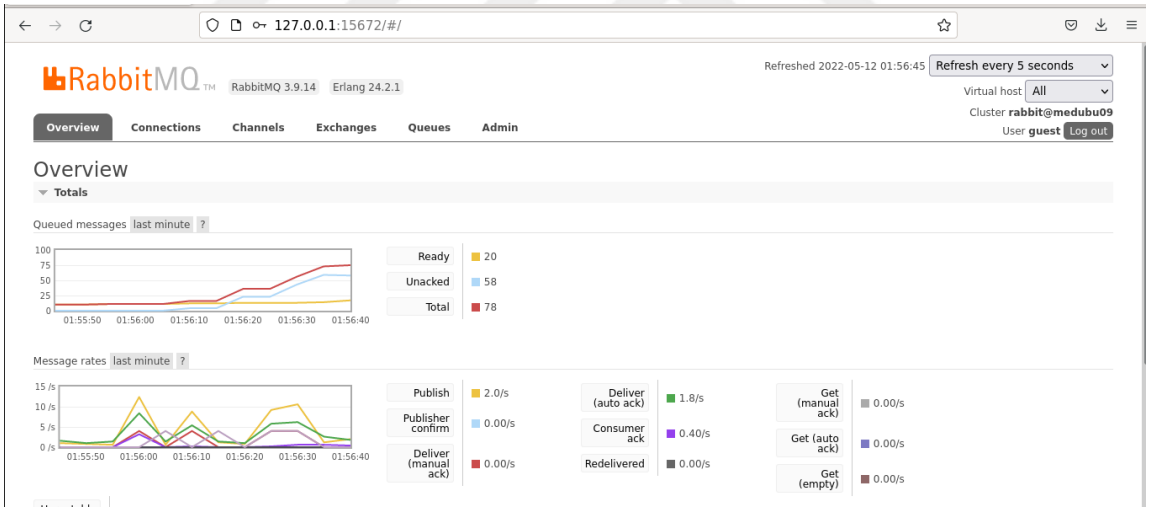


Figure 2.4: RabbitMQ Message Broker Screenshot during the Testing Process.

2.9. Flower Monitoring Tool

Flower is a web-based utility that allows the user to keep track of Celery clusters, task progress, details, and worker status. It has a variety of functions to assist the user in comprehending and being aware of what is going on. Some of the features of Flower include real-time monitoring with task progress and history graphs and statistics, as well

as a remote control with the ability to view worker status and statistics, shutdown and restart worker instances, and monitor scheduled, reserved, and canceled jobs. It also has broker monitoring, which allows the user to view not only Celery queues but also graphs of queue length [51]. **Figure 2.5** shows the administration panel of the Flower monitoring tool.

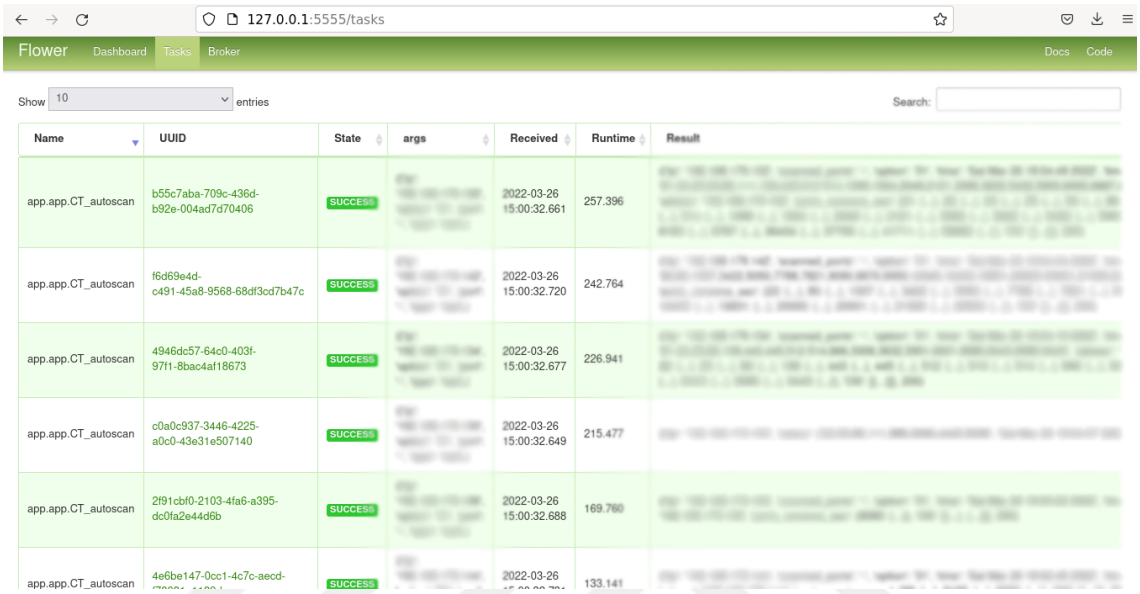


Figure 2.5: Flower Monitoring Tool Screenshot during the Testing Process.

CHAPTER 3

3. EXPERIMENTAL PART

3.1. General Structure And Workflow

The **Figure 3.1** demonstrates the general structure and workflow of the suggested schema. The client has the ability to monitor the system via the web application and to check the status of each step of the process in real-time, including the producers (CeleryBeat, API Server), broker (RabbitMQ), and consumers (CeleryWorkers); as well as, the user is able to launch real-time scans with different options such as auto scan (main endpoint), host discovery, port scanner, SV enumeration, OS detection, and vulnerability detection. As it will be explained in the "Methods And Implementation" section, for each option of scanning, there are multiple parameters and flags that help the user create a customized scan for a specific target. The Python marshmallow library is used as a part of the RESTful API server to validate the received request before the scanning and broker step. The aim of using this library is to enhance the performance of the schema in different aspects, including memory and security. Next, we can see the Celery Task Queue as shown in **Figure 3.1** contains three parts. The first one that we have already mentioned is the producers, which can produce a custom scan specified by the user or trigger the scheduled scans stored in the system depending on the given rules. Each of the requests is sent from the producer to the broker, which is RabbitMQ in our case after the validation step. RabbitMQ will give a ticket ID for each request and add it to the queue to be processed as specified. Then, it starts to assign each task to the Celery workers in the consumers according to their availability. After the worker finishes the given tasks, it stores the results in the database of the system to make the data available to the user from the web app.

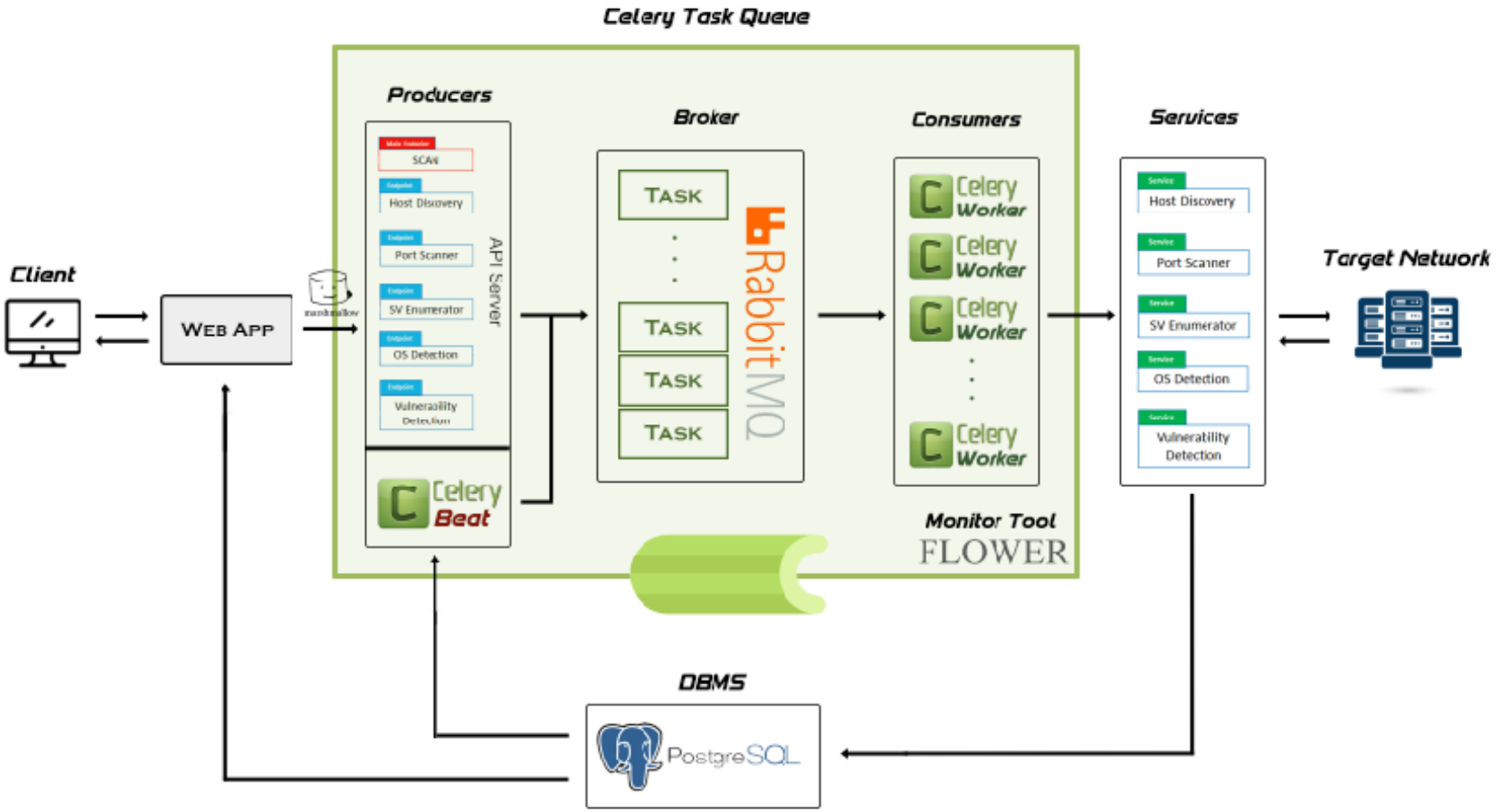


Figure 3.1: General Workflow and Structure.

3.2. Methods And Implementation

The project has been implemented in the Python 3.9 programming language and the Flask Micro Web-Framework as a RESTful-API Server based on Network Mapper (Nmap) algorithms with Celery and RabbitMQ. This approach gives the chance to integrate the project with any system. We have implemented a main endpoint called "AutoScan" requiring IP, port range, scan type and option input to go over the whole process in an automated and parallelized approach, starting with the host discovery, port scanning, service-version enumeration, and ending with vulnerability detection. The detailed result of the scan will be presented as JSON output. Each function of the active reconnaissance process has been implemented as a separate endpoint as shown in **Figure 3.2** to increase the accuracy and optimize the schema. Additionally, it allows advanced users to enter their own customized parameters that might be needed. Each function has been explained in the following subsections.

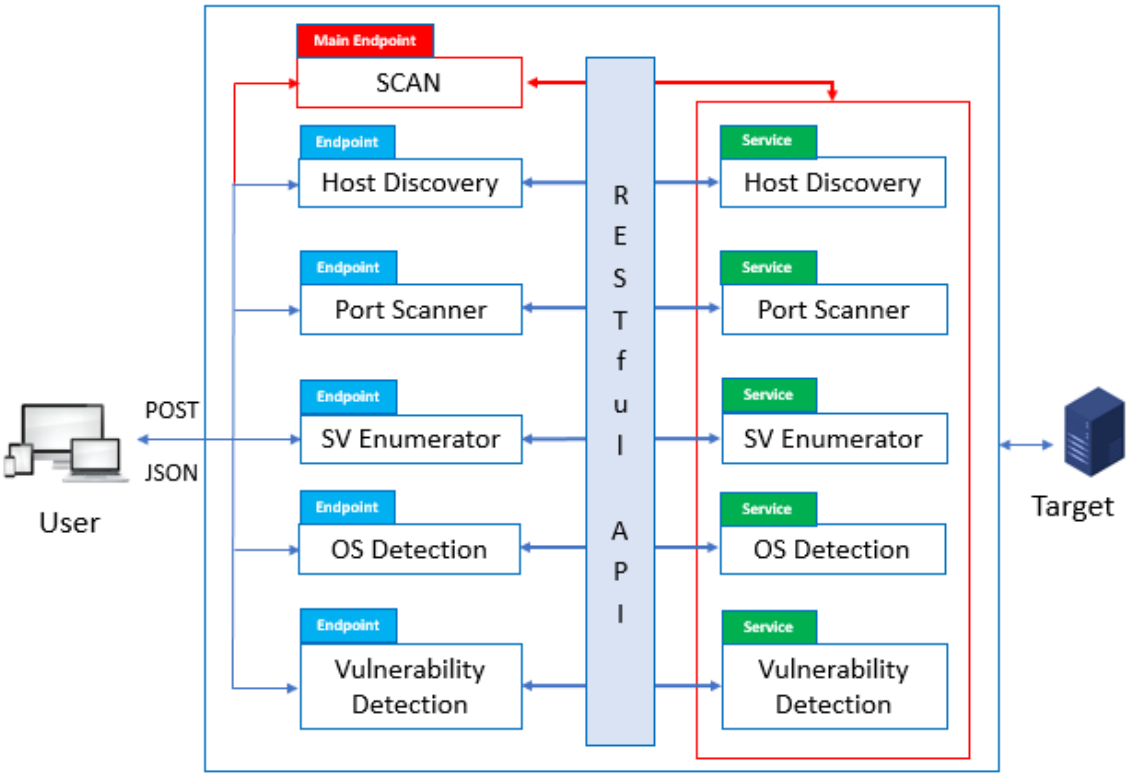


Figure 3.2: General Structure - Initial Version.

3.2.1. Host discovery

Host discovery is the first endpoint in our system. "<server-ip>/api/v1/host-check". At this point, the given IP's (target system) status will be checked to see if it is active or not, and the result will be "Up" or "Down." Accordingly, the system will proceed to the next step (port scan endpoint) automatically. If the result is down then it will go directly to the final step without wasting time with the other scanning methods.

There are three primary host discovery detection methods as demonstrated in **Figure 3.3**. The first one is ARP detection, which in turn broadcasts ARP query packets throughout the LAN. Using a list scan rather than a PING scan by setting the target host's state to "HOST UP" is the second one. The other detection methods include sending four different types of data packets to determine whether the target host is online; An ICMP timestamp request, an ICMP echo request, a TCP SYN packet to port 443, a TCP ACK packet to port 80, and a TCP SYN packet to port 443.

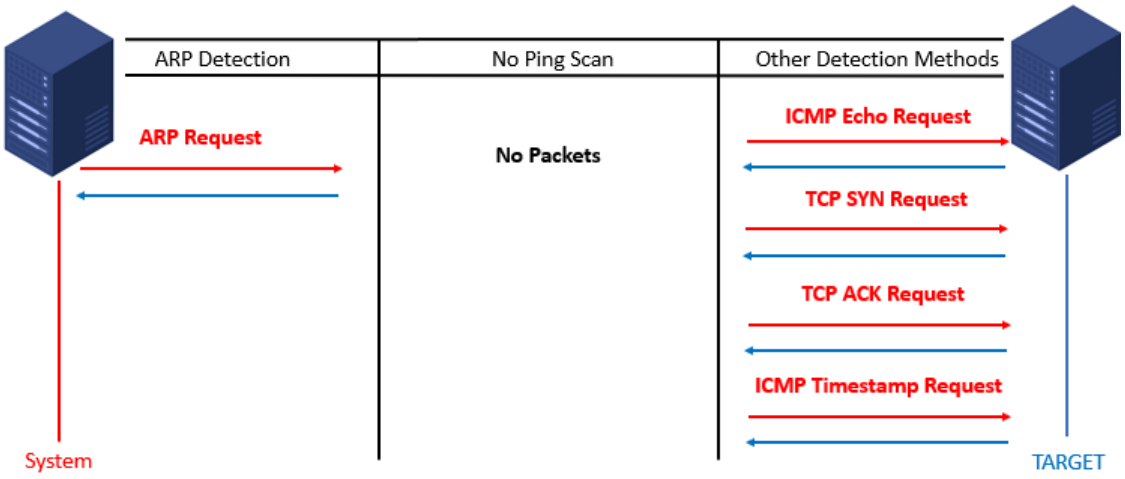


Figure 3.3: Host Discovery Methods.

3.2.2. Port scanning

Port scanning "<server-ip>/api/v1/port-scan" is the schema's second endpoint. By default, the system will start scanning the most known 1000 TCP port, unless the user changes the POST request to scan a specific port range with the demanded network protocol, such as TCP, UDP, or both together. Open ports and the related protocol results will be stored temporarily on the server to be prepared and sent to the next endpoint, which is the service and version enumeration step. In the case of having no open ports detected, the system

will terminate the process and show the results.

The goal of a port scan is to determine the target port's operational condition by returning the packet's characteristics. Different probe packets are created for different scan types. Nmap first runs a PING probe operation and transmits the necessary probe packets to the target machine's specified port, as shown in **Figure 3.4**. Then it either waits for the probe packet to be re-transmitted or sends a fresh probe packet. Finally, it is dependent on several detection mechanisms and waits for various types of response packets.

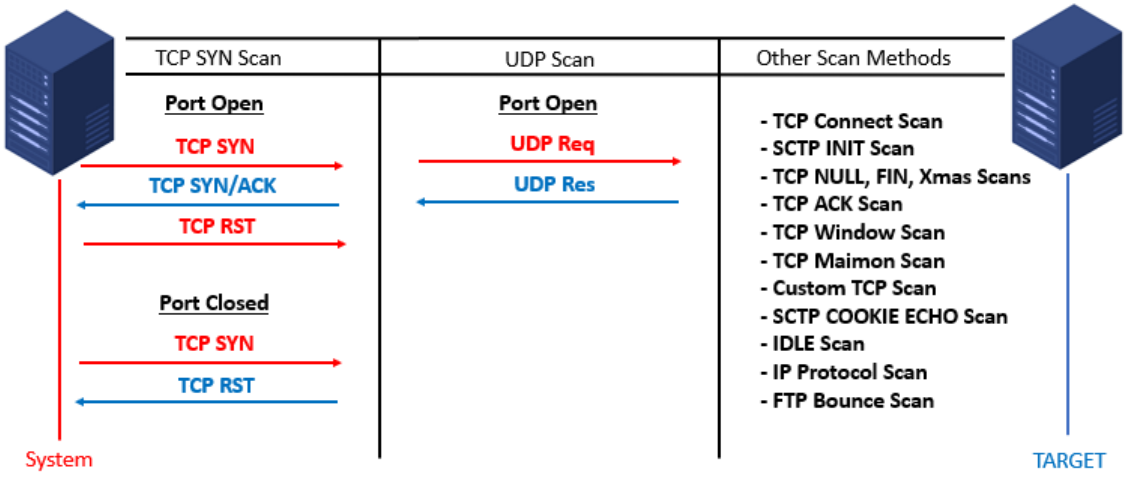


Figure 3.4: Port Scan Methods.

3.2.3. Service and version enumeration

At this endpoint "<server-ip>/api/v1/sv-scan", the system enumerates only the banners of the services on the open ports, then the version of each one. This part is the most vital one, as in the next step we are going to detect the related vulnerabilities. Open ports and the related protocol results will be stored temporarily on the server to be prepared and sent to the next endpoint, which is the service and version enumeration step. In the case of having no open ports detected, the system will terminate the process and show the results.

The premise behind this form of scanning is to match the scan findings to the database's service fingerprint or service signature. Thousands of common service fingerprints or signature traits are integrated into the Nmap database. When a specific target port is asked for, the target system will make and send back the necessary information, which can match the port's service and service version.

3.2.4. OS detection

To increase the accuracy of the results, we have combined the operating system detection with the previous SV Enumeration endpoint. To identify the operating system, the system uses TCP/IP protocol stack fingerprints. Because there are no mandatory specifications for TCP/IP implementation in some areas in the RFC specification, distinct TCP/IP schemes may have their own unique processing mechanisms. The differences in these characteristics are what Nmap uses to determine the type of operating system. Initially, Nmap runs a sequence generation test, sending six TCP probing packets, and extracting the data fingerprint SEQ/OPS/WIN/T1. It then selects a closed UDP port, a closed TCP port, and an open TCP port for complete TCP/UDP/ICMP detection and fingerprint data extraction, respectively. Lastly, it looks at the fingerprints of known systems that came with Nmap and compares them to what it found.

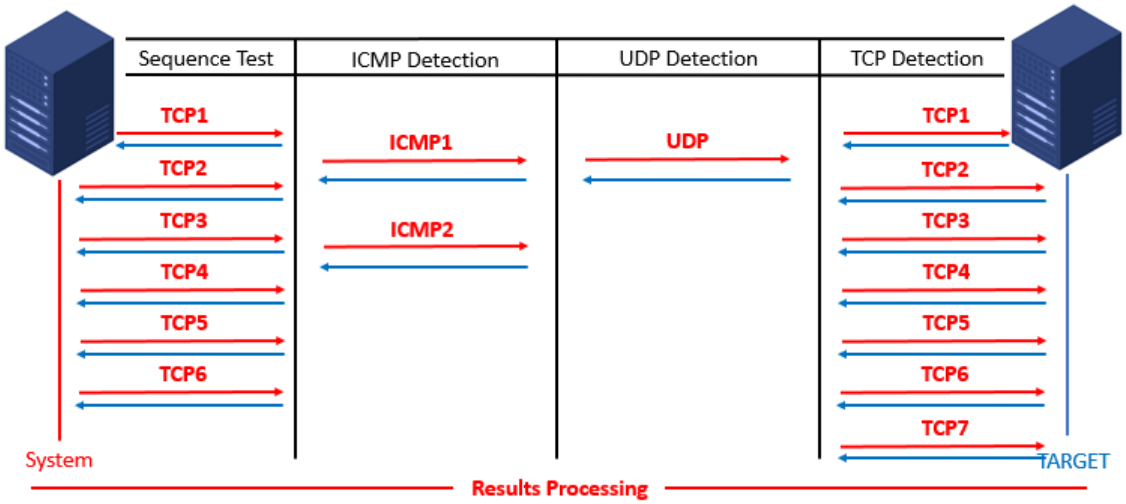


Figure 3.5: OS Detection.

3.2.5. Vulnerability detection

Detecting common vulnerabilities and exposures (CVE) and the possible threats is done at this endpoint via the Nmap Scripting Engine (NSE), and at the end, it returns the final results of the scan. It is divided into three main functions;

- Vulscan: Vulscan is a module that extends Nmap's ability to scan for network vulnerabilities. This is an NSE script that assists in the discovery of vulnerabilities on a single target or a network. This script focuses on service and version detection in

order to determine the severity of vulnerabilities on the target computer or network. The module uses the most famous databases of vulnerabilities which includes all the officially announced security issues. The pre-installed databases are

  - scipvuldb.csv - `https://vuldb.com`
  - cve.csv - `https://cve.mitre.org`
  - securityfocus.csv - `www.securityfocus.com/bid`
  - xforce.csv - `https://exchange.xforce.ibmcloud.com/`
  - expliotdb.csv - `https://www.exploit-db.com`
  - openvas.csv - `http://www.openvas.org`
  - securitytracker.csv - `https://www.securitytracker.com`
  - osvdb.csv - `http://www.osvdb.org`

- Vulners: Vulners is another NSE script that makes use of publicly available services to offer pertinent information about vulnerabilities with more than more than 250GB database size. It is integrated with the nmap libraries in order to improve Nmap's ability to scan for network vulnerabilities.

- Comprehensive: Comprehensive mode makes use of both modules together to detect the highest number possible of vulnerabilities. It checks the results of their return in accordance with their priority in order without any repetitions.

3.2.6. Main Endpoint - Scan

The main endpoint option, which is one of the major goals of our project, provides both complete automation of the active information reconnaissance process and allows parallel processing of requests received with the support of Celery. What users have to do is just send the IP with a scan option request, and they will be able to review the results when they are ready, without having to wait. The main endpoint will be going through all the phases as shown in **Figure 3.2**. It begins with host discovery, followed by port scanning, detection of operating systems, and lastly, vulnerability detection. Depending on the optimization rules we have made, In the case of a negative result at any stage, the process will be finalized without initiating the remaining stages by stating the reason.

3.2.7. Code packaging and structuring

In this project, Python Enhancement Proposals (PEP) have been followed that officially define the coding style that all Python developers should adhere to. As mentioned in the PEP, readability is one of the main features of Python. One of the reasons for trying to make the code as structured as possible is readability. The concept of structure involves creating clean code with clear logic and dependencies, as well as how files and folders are ordered in the file-system to meet the project's objectives in the best way possible. To achieve this intent, Python modules and packaging are used. Python modules are one of the most commonly used abstraction layers for dividing code into pieces that hold related data and functions. For example, one layer would be responsible for interacting with user actions, while another would be in charge of low-level data manipulation, and this is done with the "import" and "from" statements. Python has a very clear packaging system, which is essentially a directory extension of the module mechanism. A Python package is any directory that contains an "__init__.py" file that contains all package-wide declarations. Some of the major factors that play a big role in planning and deciding what the project will look like include implementing the functions in the most appropriate modules, data flow through the project, and the relationship between features and functions. The most important steps toward making work easier, being reliable, and able to grow are putting data and code in modules, putting everything in one directory as a package, using relative paths instead of absolute paths, and choosing file names carefully.

3.2.8. Schema and data validation

A unique schema has been developed for each endpoint in the new version of our project to enhance endpoint connectivity. Prior to processing, the incoming request will be verified for compliance and handled appropriately. This is performed using Marshmallow, a Python package that transforms complicated data types into native Python data types and vice versa [52]. We have four schemas as follow;

- Host_Check schema requires only IP as a string
- Port_Scan schema requires IP, port, type as a string
- SV_Scan schema requires IP, port, option as a string

- AutoScan schema requires IP, port, type, option as a string

3.2.9. Scanning options and workflow

As a result of the improvements we made to the approach, there are now a total of six different vulnerability detection methods available. Each one has a flag to enable the script scan using the default set of scripts of the Nmap Scripting Engine.

- Option one with flag one or zero (#10, #11) enumerates the services running on the opened ports, their versions, and then detects vulnerabilities using the "VulScan" script.
- Option two with flag one or zero (#20, #21) follows the same steps as option one, depending on the "Vulners" script.
- Option three with flag one or zero (#30, #31) checks the system using both scripts at the same time.

Figure **Figure 3.6** indicates the project structure and the flow of the request. It can be received via the producers, in our case, Celery Beat and RESTful-API. Next, it will be sent to the RabbitMQ broker to be then sent to celery workers according to their availability and processed with the related service.

3.3. Test Cases

The scheme's performance was assessed using a variety of scenarios based on real-world situations. We tested the project in a total of 20 different environments, covering the majority of the potential topologies. The following are the test cases;

3.3.1. Metasploitable-2

Metasploitable-2 is a deliberately vulnerable Linux virtual server meant for target practice, training, and exploit testing. Metasploitable2, unlike other vulnerable virtual machines, focuses on operating system and network services layer vulnerabilities rather than custom, vulnerable programs. In terms of vulnerabilities that might be uncovered in a production setting, it has a very similar scenario.

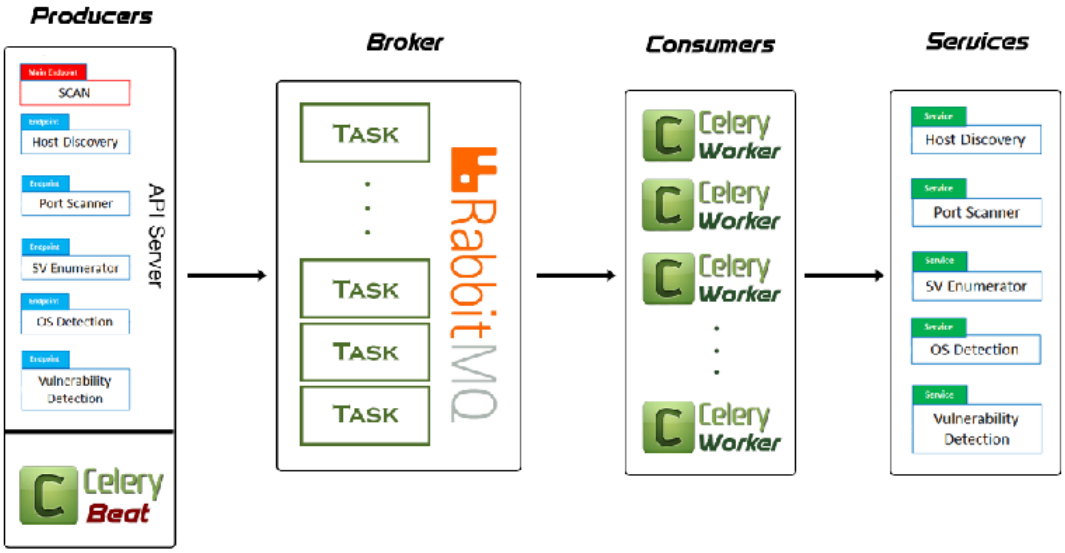


Figure 3.6: Project Structure.

3.3.2. Bee-box-v1.6

Bee-box is a modified Linux virtual machine that comes with a buggy web application pre-installed (bWAPP). It gives the chance to investigate almost all vulnerabilities in web applications. bWAPP is a purposefully unsafe web application that is free and open source. It assists web security enthusiasts, developers, and students in identifying and preventing web vulnerabilities. It can also be beneficial to prepare for effective penetration testing and ethical hacking tasks. The fact that bWAPP has over 100 online vulnerabilities is what sets it apart. It covers all significant known web flaws, as well as all OWASP Top 10 project vulnerabilities. bWAPP is a MySQL database-driven PHP application.

3.3.3. Metasploitable-1

The Metasploit project is a computer security project helping with penetration testing and IDS signature development by providing knowledge about security flaws. This is the initial version of metasploitable, which is based on an Ubuntu 8.04 server that has been customized. Tomcat 5.5 (with weak credentials), distcc, tikiwiki, twiki, and an earlier mysql are among the insecure programs provided. As previously stated, it is a purposely vulnerable virtual system intended for target practice, training, and exploit testing. It focuses on operating system and network services layer vulnerabilities.

3.3.4. BTRsys1 and BTRSys2.1

BTRsys project contains intermediate-level boot-to-root susceptible images, with the primary goal of gaining shell access by attacking vulnerable services on the machine, which is a distinct scenario from the real-world probable risks.

3.3.5. g0rmint

This buggy machine is totally based on a real-world scenario that has been encountered while performing testing for a client's website. It is one of the advanced scenarios in obtaining a limited shell.

3.3.6. Hades-v1.0.1

Hades is yet another boot2root challenge designed mainly for experienced researchers. Reverse engineering, sploit creation, and a thorough understanding of computer architecture will all be required to exploit this machine successfully. The goal of it is to gradually get more access to the box until the root account is reached.

3.3.7. vulnVoIP

VulnVoIP is built on the AsteriskNOW distribution, which has some flaws in it. The goal is to track down VoIP users, crack their passwords, and acquire access to the support account voicemail message system. To add a little spice to the mix, this specific distribution is also vulnerable to a well-known exploit that makes it simple to obtain access to the system's root shell. It is one of the few real-life VoIP test environments available.

3.3.8. VPLE

The Vulnerable Pentesting Lab Environment (VPLE) is a Linux virtual system that is designed to be purposefully vulnerable. This virtual machine can be used for education, tool testing, and standard penetration testing labs. VPLE includes Web-dvwa, Mutillidae, Webgoat, Bwapp, Juice-shop, Security-ninjas, and a Wordpress environment.

3.3.9. Tr0ll2

The Tr0ll machine has been designed to look and act similarly to the Offensive Security Certified Professional (OSCP) system, and it is prepared to troll the penetration tester at some points. Tr0ll2 is a level of difficulty that is higher than the previous level, Tr0ll1. This is a scenario of intermediate difficulty.

3.3.10. sick0s1.1

This server provides a clear example of how hacking tactics can be used to infiltrate a network in a secure setting. This virtual computer is comparable to the ones used in the Offensive Security Certified Professional labs (OSCP). The goal is to get into the network/machine and gain administrative/root access to it.

3.3.11. MorningCatch

Morning Catch is a VMware virtual machine that demonstrates targeted client-side attacks and post-exploitation, akin to Metasploitable. A website for a bogus seafood firm, a self-contained email infrastructure, and vulnerable Linux and Windows client-side desktop environments can all be found on this virtual server. It also uses WINE to run a few weak Windows apps.

3.3.12. NETinVM UML 2016

NETinVM is a VMware virtual machine image that exposes a full computer network to the user. As a result, NETinVM can be used to learn about operating systems, computer networks, and security for systems and networks. Furthermore, NETinVM is a VMware virtual machine image that provides a set of User-mode Linux (UML) virtual machines that are ready to use. When the UML virtual machines are started, they build a whole computer network. The Linux operating system is used by all of the virtual computers. It is a full system that displays a real-world network.

3.3.13. hackxor1

hackxor1: is a webapp testing environment where vulnerabilities must be located and exploited to progress through the system. However, it is similar to WebGoat, with a focus

on realism and challenge. XSS, CSRF, SQLi, ReDoS, DOR, command injection, and other exploits are included. Client attack simulation using HtmlUnit, and realistic vulnerabilities modeled by Google, Mozilla, and other platforms are just a few of the features available.

3.3.14. Brainpan2, CySCA2014InABox, DonkeyDockerv1.0, GoatseLinux-1.0-VM, UltimateLAMP-0.2 and w1r3s.v1.0.1

These machines are also designed to have different schemes from real-life ones to practice and test the new penetration testing tools being developed. These vulnerable servers are based on real-world scenarios and are similar to the OSCP labs.

CHAPTER 4

4. RESULTS AND DISCUSSION

4.1. Results

In each of the test cases mentioned in the previous chapter, we have tested the automation of active reconnaissance in this project, starting with host discovery, port scanning, service-version enumeration, and finalizing with the vulnerability detection option#1/#2/#3. Option one enumerates the services working on the open ports, the versions of each service, and then detects the vulnerabilities based on the "VulScan" script. Option two goes over the same process depending on the "Vulners" script. In the last one (option3), it checks the system with the help of both of the scripts together. In **Table 4.1**, we can see the time elapsed in minutes for each step of the process for each test case without the parallelization. And the second **Table 4.2** shows the total time elapsed in minutes for the whole process with the used vulnerability detection option for each test case without parallelization.

Table 4.1: Results of Test Cases.
*The Numbers show the amount of vulnerabilities found after service-version enumeration.

Virtual Machine Name	Time Elapsed in Minutes					Results				
	Host Discovery	Port Scan	S.V. Enum. Option#1	S.V. Enum. Option#2	S.V. Enum. Option#3	Total Open Ports	S.V. Enum. Option#1*	S.V. Enum. Option#2*	S.V. Enum. Option#3*	OS Detection Accuracy
Metasploitable2	0.025	0.316	5.020	2.718	5.051	30	41692	417	42109	100
Bee-box_v1.6	0.025	0.325	3.835	2.444	3.877	19	3405	544	3949	100
Brainpan2	0.025	0.287	3.183	3.192	3.191	2	6	9	15	100
Metasploitable1	0.025	0.403	4.127	2.720	4.143	13	19794	403	20197	100
BTRsys1	0.023	0.357	0.487	0.358	0.526	3	594	130	724	100
BTRSys2.1	0.115	0.407	0.596	0.281	0.625	3	606	199	805	100
CySCA2014InABox	0.117	0.490	3.98	2.882	4.194	17	52011	395	52406	100
DonkeyDocker_v1.0	0.026	0.336	0.397	0.183	0.391	2	500	150	650	100
g0rmint	0.025	0.302	0.385	0.192	0.414	2	542	199	741	100
Hades_v1.0.1	0.023	0.481	1.872	0.079	0.180	2	370	27	397	100
GoatseLinux_1.0	0.025	0.359	3.977	3.309	3.994	9	5325	228	5553	100
vulnVoIP	0.024	0.398	3.239	2.708	3.244	8	4329	110	4439	100
VPLE	0.029	0.347	1.212	0.291	1.081	8	1249	245	1494	100
Tr0ll2	0.026	0.384	0.559	0.259	0.599	3	373	23	396	100
sick0s1.1	0.039	1.885	0.489	0.292	0.531	3	399	33	432	100
MorningCatch	0.121	0.593	0.867	0.235	0.883	6	1524	34	1558	100
NETinVM 2016	0.081	0.304	0.306	0.179	0.267	2	0	0	0	100
hackxor1	0.038	2.714	0.311	0.183	0.386	1	587	0	587	100
UltimateLAMP-0.2	0.037	0.370	0.276	0.287	0.304	1	35	31	66	100
w1r3s.v1.0.1	0.130	1.065	0.991	0.406	0.890	4	3450	62	3512	100
MIN	**0.023**	**0.287**	**0.276**	**0.079**	**0.180**	**1**	**0**	**0**	**0**	**100**
AVG	**0.049**	**0.606**	**1.806**	**1.160**	**1.738**	**7**	**6840**	**162**	**7002**	**100**
MAX	**0.130**	**2.714**	**5.020**	**3.309**	**5.051**	**30**	**52011**	**544**	**52406**	**100**

Table 4.2: Total Elapsed Time in Minutes.

Virtual Machine Name	Total Time Option#1	Total Time Option#2	Total Time Option#3
Metasploitable2	5.362	3.060	5.393
Bee-box_v1.6	4.185	2.794	4.227
Brainpan2	3.495	3.504	3.503
Metasploitable1	4.554	3.147	4.570
BTRsys1	0.868	0.738	0.906
BTRSys2.1	1.117	0.802	1.146
CySCA2014InABox	4.592	3.489	4.801
DonkeyDocker_v1.0	0.759	0.545	0.753
g0rmint	0.712	0.518	0.740
Hades_v1.0.1	2.377	0.583	0.685
GoatseLinux_1.0_VM	4.360	3.692	4.377
vulnVoIP	3.662	3.131	3.667
VPLE	1.588	0.666	1.457
Tr0ll2	0.969	0.669	1.010
sick0s1.1	2.414	2.216	2.455
MorningCatch	1.581	0.948	1.597
NETinVM UML 2016	0.691	0.564	0.652
hackxor1	3.063	2.935	3.138
UltimateLAMP-0.2	0.683	0.694	0.711
w1r3s.v1.0.1	2.186	1.601	2.084
MIN	**0.683**	**0.518**	**0.652**
AVG	**2.461**	**1.815**	**2.394**
MAX	**5.362**	**3.692**	**5.393**

We have enhanced the results by the parallelization and compared it with the previous results **Table 4.1** and **Table 4.2**. The testing includes all scanning steps; host discovery, port scanning, service-version enumeration, and vulnerability detection with improved six distinct options. The second **Table 4.3** shows the comparison between the enhanced version and the previous work in terms of the scanning time with the enhancement ratio. The vulnerabilities found in each test case with the related improvement ratio are shown in **Table 4.4**. The last **Table 4.5** shows how long it took to run the new scanning options that were added to the schema.

Table 4.3: Test Cases based Previous and Enhanced Version Scanning Time Comparison.
*The enhanced version is fully automated and parallelized.

Virtual Machine Name	Time Elapsed in Minutes								
	Auto Scan - Option#10			Auto Scan - Option#20			Auto Scan - Option#30		
	Previous Version	Enhanced Version*	Enhancement Ratio	Previous Version	Enhanced Version*	Enhancement Ratio	Previous Version	Enhanced Version*	Enhancement Ratio
Metasploitable2	5.362	4.233	21.05%	3.060	2.773	9.36%	5.393	4.334	19.63%
Bee-box_v1.6	4.185	3.711	11.33%	2.794	3.167	-13.37%	4.227	3.767	10.87%
Brainpan2	3.495	3.574	-2.25%	3.504	3.548	-1.27%	3.503	3.588	-2.43%
Metasploitable1	4.554	1.534	66.31%	3.147	0.635	79.83%	4.570	1.567	65.71%
BTRsys1	0.868	0.710	18.21%	0.738	0.563	23.67%	0.906	0.757	16.44%
BTRSys2.1	1.117	0.675	39.60%	0.802	0.538	32.96%	1.146	0.703	38.67%
CySCA2014InABox	4.592	4.070	11.37%	3.489	3.173	9.06%	4.801	4.202	12.48%
DonkeyDocker_v1.0	0.759	0.621	18.13%	0.545	0.534	1.97%	0.753	0.652	13.40%
g0rmint	0.712	0.627	11.95%	0.518	0.495	4.52%	0.740	0.640	13.59%
Hades_v1.0.1	2.377	0.495	79.17%	0.583	0.453	22.41%	0.685	0.515	24.81%
GoatseLinux_1.0	4.360	2.979	31.67%	3.692	3.137	15.04%	4.377	3.208	26.71%
vulnVoIP	3.662	3.385	7.54%	3.131	3.223	-2.94%	3.667	3.575	2.50%
VPLE	1.588	1.041	34.42%	0.666	0.406	39.09%	1.457	0.976	33.03%
Tr0ll2	0.969	0.632	34.77%	0.669	0.494	26.20%	1.010	0.623	38.33%
sick0s1.1	2.414	2.200	8.86%	2.216	2.080	6.13%	2.455	2.216	9.73%
MorningCatch	1.581	0.995	37.02%	0.948	0.716	24.51%	1.597	1.043	34.68%
NETinVM 2016	0.691	0.502	27.23%	0.564	0.518	8.09%	0.652	0.523	19.81%
hackxor1	3.063	2.833	7.51%	2.935	2.672	8.97%	3.138	1.137	63.76%
UltimateLAMP-0.2	0.683	0.667	2.36%	0.694	0.578	16.79%	0.711	0.645	9.20%
w1r3s.v1.0.1	2.186	1.920	12.16%	1.601	1.641	-2.54%	2.084	1.872	10.16%
MIN	**0.683**	**0.495**	**27.56%**	**0.518**	**0.406**	**21.70%**	**0.652**	**0.515**	**21.03%**
AVG	**2.461**	**1.870**	**24.00%**	**1.815**	**1.567**	**13.65%**	**2.394**	**1.827**	**23.66%**
MAX	**5.362**	**4.233**	**21.05%**	**3.692**	**3.548**	**3.90%**	**5.393**	**4.334**	**19.63%**
Total Run Time	**49.215**	**37.405**	**24.00%**	**36.297**	**31.344**	**13.65%**	**47.870**	**36.543**	**23.66%**
Total Elapsed Time	**49.215**	**4.233**	**91.40%**	**36.297**	**3.548**	**90.22%**	**47.870**	**4.334**	**90.95%**

Table 4.4: Test Cases based Previous and Enhanced Version found vulnerabilities Comparison.

Virtual Machine Name	Vulnerabilities Found								
	Auto Scan - Option#10			Auto Scan - Option#20			Auto Scan - Option#30		
	Previous Version	Enhanced Version	Enhancement Ratio	Previous Version	Enhanced Version	Enhancement Ratio	Previous Version	Enhanced Version	Enhancement Ratio
Metasploitable2	41692	41851	0.38%	417	427	2.40%	42109	42278	0.40%
Bee-box_v1.6	3405	3118	-8.43%	544	544	-	3949	3662	-7.27%
Brainpan2	6	6	-	9	10	11.11%	15	16	6.67%
Metasploitable1	19794	19953	0.80%	403	427	5.96%	20197	20380	0.91%
BTRsys1	594	594	-	130	136	4.62%	724	730	0.83%
BTRSys2.1	606	606	-	199	205	3.02%	805	811	0.75%
CySCA2014InABox	52011	59079	13.59%	395	335	-15.19%	52406	59414	13.37%
DonkeyDocker_v1.0	500	500	-	150	159	6.00%	650	659	1.38%
g0rmint	542	542	-	199	205	3.02%	741	747	0.81%
Hades_v1.0.1	370	370	-	27	25	-7.41%	397	395	-0.50%
GoatseLinux_1.0	5325	5325	-	228	228	-	5553	5553	-
vulnVoIP	4329	4329	-	110	110	-	4439	4439	-
VPLE	1249	1249	-	245	245	-	1494	1494	-
Tr0ll2	373	373	-	23	23	-	396	396	-
sick0s1.1	399	465	16.54%	33	79	139.39%	432	544	25.93%
MorningCatch	1524	1613	5.84%	34	128	276.47%	1558	1741	11.75%
NETinVM 2016	0	0	-	0	6	-	0	6	-
hackxor1	587	704	19.93%	0	0	-	578	704	21.80%
UltimateLAMP-0.2	35	86	145.71%	31	109	251.61%	66	195	195.45%
w1r3s.v1.0.1	3450	4128	19.65%	62	205	230.65%	3512	4333	23.38%
MIN	**0**	**0**	-	**0**	**0**	-	**0**	**6**	-
AVG	**6840**	**7245**	**5.92%**	**162**	**180**	**11.33%**	**7001**	**7425**	**6.05%**
MAX	**52011**	**59079**	**13.59%**	**544**	**544**	-	**52406**	**59414**	**13.37%**
Total	**136791**	**144891**	**5.92%**	**3239**	**3606**	**11.33%**	**140021**	**148497**	**6.05%**

Table 4.5: New Scanning Options Time Results.

Virtual Machine Name	Time Elapsed in Minutes		
	Auto Scan (New Scanning Options)		
	Option#11	Option#21	Option#31
Metasploitable2	4.252	2.818	4.290
Bee-box_v1.6	3.772	3.176	3.782
Brainpan2	3.574	3.531	1.419
Metasploitable1	1.522	0.669	1.581
BTRsys1	0.732	0.581	0.764
BTRSys2.1	0.696	0.521	0.715
CySCA2014InABox	3.906	3.170	4.046
DonkeyDocker_v1.0	0.641	0.520	0.668
g0rmint	0.632	0.553	0.662
Hades_v1.0.1	0.456	0.442	0.541
GoatseLinux_1.0	2.955	2.922	2.986
vulnVoIP	3.416	3.288	3.591
VPLE	1.072	0.392	1.163
Tr0ll2	0.611	0.511	0.637
sick0s1.1	2.176	2.068	2.219
MorningCatch	1.002	0.726	1.040
NETinVM 2016	0.487	0.553	0.500
hackxor1	2.791	2.761	2.829
UltimateLAMP-0.2	0.659	0.596	0.646
w1r3s.v1.0.1	1.846	1.692	1.897
MIN	**0.456**	**0.392**	**0.500**
AVG	**1.860**	**1.574**	**1.799**
MAX	**4.252**	**3.531**	**4.290**
Total Run Time	**37.199**	**31.489**	**35.976**
Total Elapsed Time	**4.252**	**3.531**	**4.290**

4.2. Discussion

The evaluation process over numerous test environments outlined in the preceding section demonstrates the project's stability and high performance. Users can check the status of the host (up/down), examine the open ports, find the exposed service and version information, and finally investigate all possible risks, threats, and vulnerabilities in an automated manner using three different detection functions provided as quick, detailed, and comprehensive. The minimum, average, and maximum values of the time elapsed for each endpoint test case are shown in **Figure 4.1**.

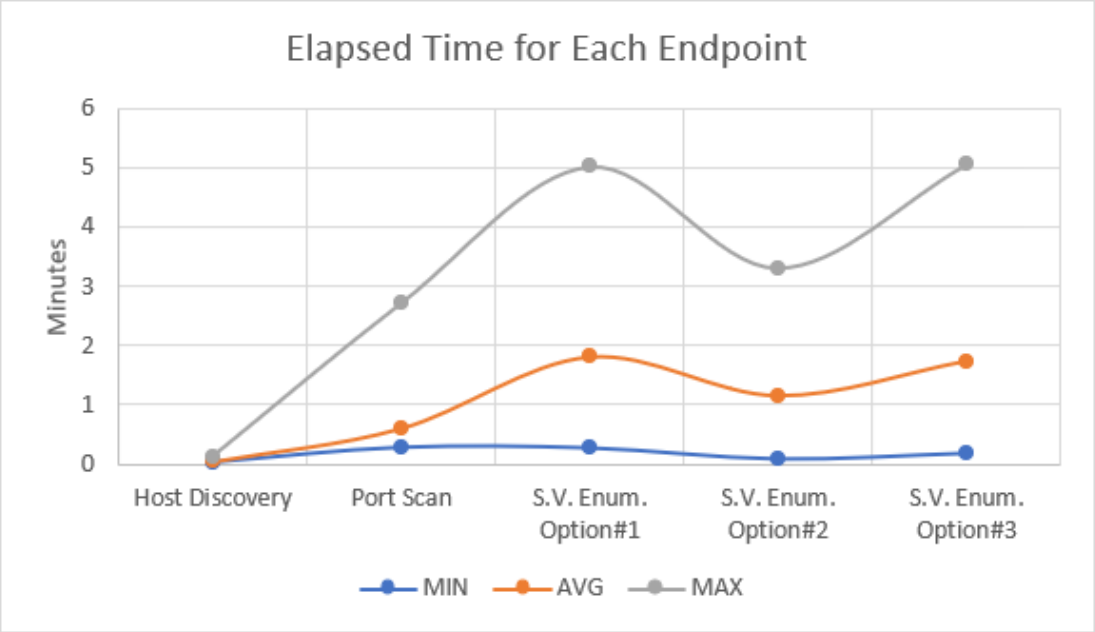


Figure 4.1: Total time for each process.

In our approach, scanning option#10 relies on the Nmap Scripting Engine's Vulners module, which provides the common vulnerabilities and exposures reference numbers related to the enumerated service and version information, as well as the exploit priority. Option#20 is more detailed and based on the VulScan script, which gives a significantly longer list that includes not only the CVE reference number but also additional potential risks classified by various types of reference numbers. As we mentioned in the methods section, this module is integrated with the world's eight most well-known vulnerability databases, which cover nearly every possible risk related to the organization's technology. The final comprehensive method (option#30) combines both scripts and returns the unique findings. Figure **Figure 4.2** shows the high performance of each strategy for the initial version of the study before the parallelization.

For all the options in the schema, we can enable the flag by converting the one to zero to lunch the NSE default scripts. **Figure 4.3** demonstrates the enhancement ratio for each strategy in terms of run time, elapsed time, and the number of detected vulnerabilities for the final version. The parallelization system used in the scheme accelerated the process significantly. It reduced the total elapsed time to scan the 20 machines from 44.46 minutes to 4.03 minutes, which means a 90.80% improvement. The total run time without parallelization has been enhanced by 20.40% from 44.46 minutes to 35.09 minutes as well. We have detected 98998 vulnerabilities in the current version, compared to 93350 in the

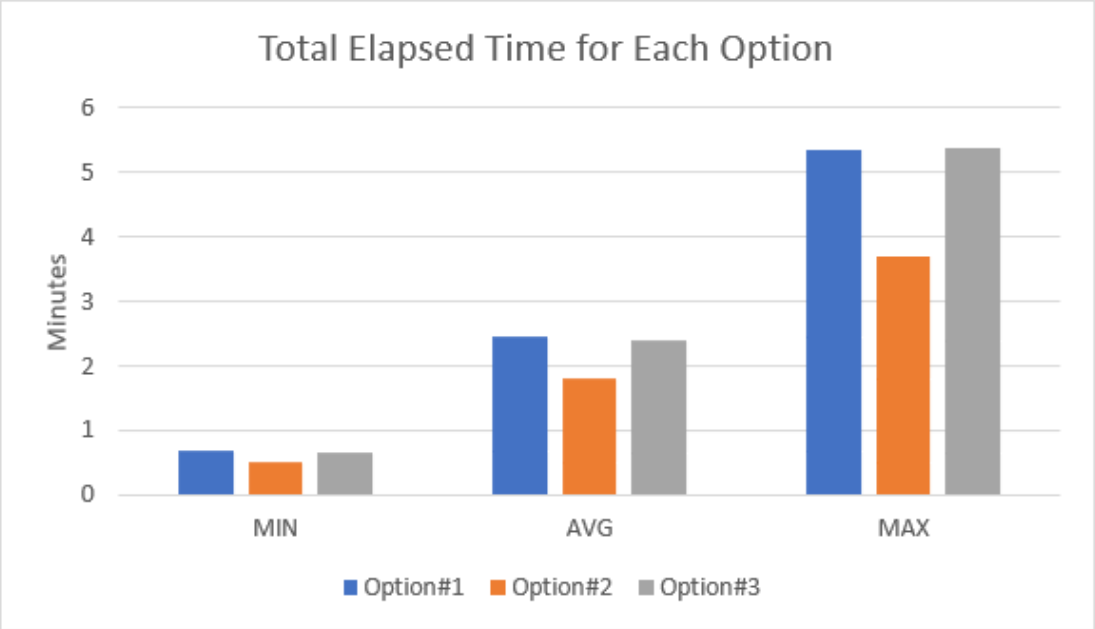


Figure 4.2: Total time for each option.

previous version. We were able to achieve a 7.70% enhancement. Similarly, the average time to scan a server for vulnerabilities was lowered from 2.2 minutes to 1.7 minutes, a 22.73% reduction.

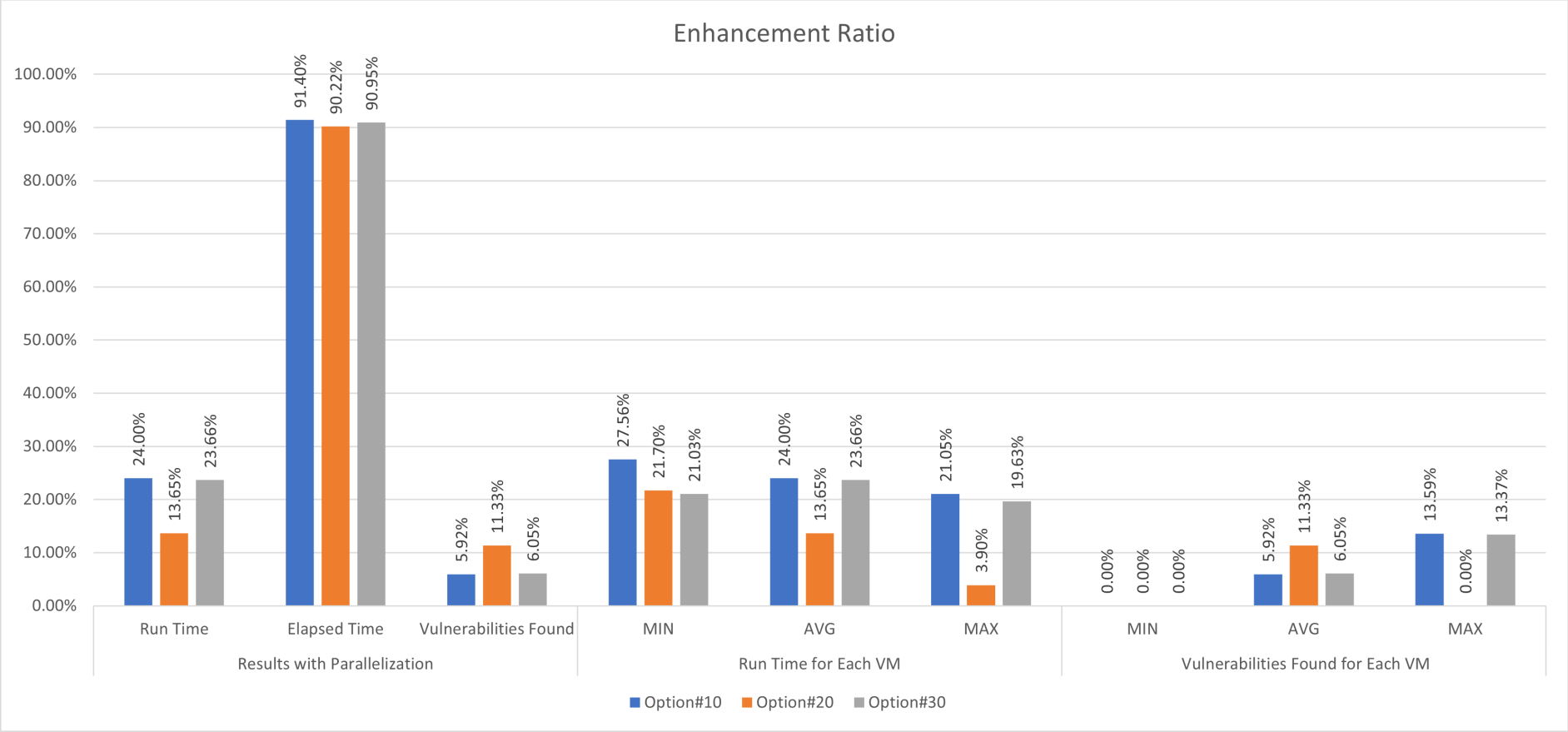


Figure 4.3: Enhancement Ratio for each option from different aspects.

The relationship between the total open port number and the total elapsed time for the enumeration and detection operation is shown in **Figure 4.4**. As it can be expected, the time increases as the total number of open ports grows. However, the increment in time is only about 12 seconds for each port, regardless of the number of vulnerabilities. Castiglione et al. [53] indicated that performing a manual approach would take one minute per vulnerability. For example, our system was able to detect 42109 vulnerabilities in just five minutes for the Metasploitable2 environment. The best case in the increment ratio was 6 seconds and the worst case was 1.5 minutes in the test cases. This shows how stable the suggested schema is and how well it works in terms of its high processing speed.

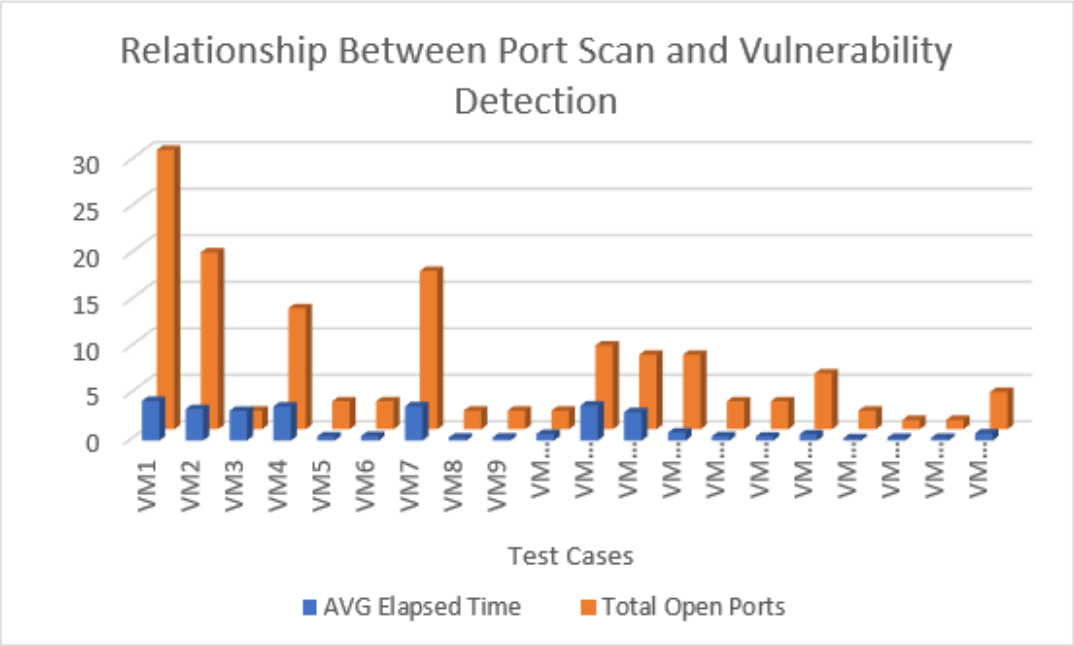


Figure 4.4: Port Scan and Vuln. Detection Relationship.

Last but not least, the suggested technique will play an important role in system security by allowing companies and system administrators to scan their systems on a frequent and regular basis with the minimal resources they have. Furthermore, this allows for rapid updates to newly discovered software vulnerabilities. Our scanner is used to demonstrate how simple it is to scan a complicated enterprise-grade web application. Attempting to enumerate the service-version and vulnerability information from all ports without making any optimization rules to enhance the performance of the operation is the most common mistake made by other researchers.

CHAPTER 5

5. CONCLUSION AND FUTURE WORK

5.1. Summary And Conclusion

Protecting the CIA (Confidentiality, Integrity, and Availability) Triad, which is at the heart of information security, has become an increasingly difficult task as technology advances. Penetration testing is a method of detecting hidden threats and possible attacks. Manual testing and vulnerability detection procedures, on the other hand, take a long time and make it difficult to uncover various sorts of application faults in a wide network. As a result, regular and automated vulnerability assessments can help decrease the risks of eventual penetration, as well as data integrity, availability, and confidentiality violations, while also promptly discovering flaws.

The goal of this research article is to provide a generic and optimized approach for automating the process using the available and dynamic vulnerability and exposure databases. We've made it apparent that our technique is primarily focused on automating network scanning and vulnerability detection. In this paper, we have enhanced the proposed schema in the previous paper for the active information-gathering phase. The project is an API-based automated and parallelized security scanner, including an IP and port scanner, a service-version enumerator, and a vulnerability detection system. In this approach, the Network Mapper is used to collect information with high accuracy using the rules in our schema. Moreover, the work has been developed as a RESTful-API server for easy integration in real-world scenarios, allowing users to scan and protect their networks more rapidly while remaining completely scalable and responsive to their needs and growth.

In the phase of cyber reconnaissance automation, it supports parallel real-time and scheduled system scans. The user can create instant customized scans and view the relevant results with the integration of message-broker software (RabbitMQ) that originally implemented the advanced message queuing protocol (AMQP). These features rely on Celery workers to perform multiprocessing and concurrent task execution in the main endpoint via an asynchronous task queue. The project follows PEP to increase readability, and modules and packaging to meet the project's objectives in the best way possible. To improve endpoint connectivity, a specific schema has been designed for each endpoint in the new version of our project. The inbound request will be checked for compliance using Marshmallow before being processed.

Furthermore, the test case outcomes were satisfactory and beyond our expectations. Based on real-life events, we nearly covered all of the conceivable fundamental conditions. The results were drastically different in comparison to a more manual procedure. In a short period of time, we were able to detect vulnerabilities with high accuracy. We improved the average time of scanning a server and detecting vulnerabilities by 22.73% to become 1.7 minutes in the new version of the schema, as well as the run time, elapsed time, and vulnerability detection by 20.40%, 90.80% and 7.70% respectively.

5.2. Future Research Directions

In the future, our schema will be upgraded to cover the remaining parts of security scanning, as this is just an updated version of the first aspect of the security engine we aim to develop. The next steps will be to build an alarm system, add unit tests for the code, and connect the project to a database. After that, the project will be used in the real world to make sure that the technique works well and is effective.

BIBLIOGRAPHY

[1] A. M. Gamundani and L. M. Nekare, "A review of new trends in cyber attacks: A zoom into distributed database systems," in *2018 IST-Africa Week Conference (IST-Africa)*. IEEE, 2018, pp. Page–1.

[2] D. Arnaldy and A. R. Perdana, "Implementation and analysis of penetration techniques using the man-in-the-middle attack," in *2019 2nd International Conference of Computer and Informatics Engineering (IC2IE)*. IEEE, 2019, pp. 188–192.

[3] N. Zhu, X. Chen, and Y. Zhang, "Construction of overflow attacks based on attack element and attack template," in *2011 Seventh International Conference on Computational Intelligence and Security*. IEEE, 2011, pp. 540–544.

[4] S. Kang, D. Qiaozhong, and Z. WeiQiang, "Space information security and cyberspace defense technology," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 1509–1511.

[5] G. Daria and A. Massel, "Intelligent system for risk identification of cybersecurity violations in energy facility," in *2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)*. IEEE, 2018, pp. 1–5.

[6] A. Markov, A. Fadin, and V. Tsirlov, "Multilevel metamodel for heuristic search of vulnerabilities in the software source code," *International Journal of Control Theory and Applications*, vol. 9, no. 30, pp. 313–320, 2016.

[7] A. I. Pechenkin and D. S. Lavrova, "Modeling the search for vulnerabilities via the fuzzing method using an automation representation of network protocols," *Automatic Control and Computer Sciences*, vol. 49, no. 8, pp. 826–833, 2015.

[8] P. Zegzhda, D. Zegzhda, E. Pavlenko, and A. Dremov, "Detecting android application malicious behaviors based on the analysis of control flows and data flows," in *Proceedings of the 10th International Conference on Security of Information and Networks*, 2017, pp. 280–283.

[9] G. Abramov, L. Korobova, A. Ivashin, and I. Matytsina, "Information system for diagnosis of respiratory system diseases," in *Journal of Physics: Conference Series*, vol. 1015, no. 4. IOP Publishing, 2018, p. 042036.

[10] A. V. Barabanov, A. S. Markov, and V. L. Tsirlov, "Methodological framework for analysis and synthesis of a set of secure software development controls," *Journal of Theoretical and Applied Information Technology*, vol. 88, no. 1, pp. 77–88, 2016.

[11] M. Howard and S. Lipner, *The security development lifecycle: A Process for Developing Demonstrably More Secure Software*. Microsoft Press Redmond, 2006, vol. 8.

[12] S. Calzavara, R. Focardi, M. Nemec, A. Rabitti, and M. Squarcina, "Postcards from the post-http world: Amplification of https vulnerabilities in the web ecosystem," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 281–298.

[13] S. Calzavara, R. Focardi, M. Squarcina, and M. Tempesta, "Surviving the web: A journey into web session security," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 1–34, 2017.

[14] K. Nirmal, B. Janet, and R. Kumar, "Web application vulnerabilities-the hacker's treasure," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2018, pp. 58–62.

[15] A. S. Petrenko, S. A. Petrenko, K. A. Makoveichuk, and P. V. Chetyrbok, "Protection model of pcs of subway from attacks type «wanna cry»,«petya» and «bad rabbit» iot," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 2018, pp. 945–949.

[16] R. Priya, C. Lifna, D. Jagli, and A. Joy, "Rational unified treatment for web application vulnerability assessment," in *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*. IEEE, 2014, pp. 336–340.

[17] R. Bhor and H. Khanuja, "Analysis of web application security mechanism and attack detection using vulnerability injection technique," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*. IEEE, 2016, pp. 1–6.

[18] B. Wang, L. Liu, F. Li, J. Zhang, T. Chen, and Z. Zou, "Research on web application security vulnerability scanning technology," in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1. IEEE, 2019, pp. 1524–1528.

[19] D. Yadav, D. Gupta, D. Singh, D. Kumar, and U. Sharma, "Vulnerabilities and security of web applications," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. IEEE, 2018, pp. 1–5.

[20] V. V. Varenitca, A. S. Markov, and V. V. Savchenko, "Recommended practices for the analysis of web application vulnerabilities," in *10th Anniversary International Scientific and Technical Conference on Secure Information Technologies, BIT 2019 CEUR Workshop Proceedings*, vol. 2603, 2019, pp. 75–78.

[21] M. Shah, S. Ahmed, K. Saeed, M. Junaid, H. Khan *et al.*, "Penetration testing active reconnaissance phase–optimized port scanning with nmap tool," in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2019, pp. 1–6.

[22] S. Chakrabarti, M. Chakraborty, and I. Mukhopadhyay, "Study of snort-based ids," in *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, 2010, pp. 43–47.

[23] G. Kaur and N. Kaur, "Penetration testing–reconnaissance with nmap tool." *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, 2017.

[24] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013, pp. 605–620.

[25] N. Schagen, K. Koning, H. Bos, and C. Giuffrida, "Towards automated vulnerability scanning of network servers," in *Proceedings of the 11th European Workshop on Systems Security*, 2018, pp. 1–6.

[26] A. Roy, L. Mejia, P. Helling, and A. Olmsted, "Automation of cyber-reconnaissance: A java-based open source tool for information gathering," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2017, pp. 424–426.

[27] S. Panjwani, S. Tan, K. M. Jarrin, and M. Cukier, "An experimental evaluation to determine if port scans are precursors to an attack," in *2005 International Conference on Dependable Systems and Networks (DSN'05)*. IEEE, 2005, pp. 602–611.

[28] J. J. Zhao and S. Y. Zhao, "Opportunities and threats: A security assessment of state e-government websites," *Government Information Quarterly*, vol. 27, no. 1, pp. 49–56, 2010.

[29] C. N. Mooers, "Preventing software piracy," *Computer*, vol. 10, no. 3, pp. 29–30, 1977.

[30] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "Portvis: a tool for port-based detection of security events," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, 2004, pp. 73–81.

[31] K. Mathew, M. Tabassum, and M. V. L. A. Siok, "A study of open ports as security vulnerabilities in common user computers," in *2014 International Conference on Computational Science and Technology (ICCST)*. IEEE, 2014, pp. 1–6.

[32] R. Maini, P. BVDUCOEP, R. Pandey, R. Kumar, and R. Gupta, "Automated web vulnerability scanner," *Int. J. Eng. Appl. Sci. Technol*, vol. 4, no. 1, pp. 132–136, 2019.

[33] D. Schatz, R. Bashroush, and J. Wall, "Towards a more representative definition of cyber security," *Journal of Digital Forensics, Security and Law*, vol. 12, no. 2, p. 8, 2017.

[34] M. Kianpour, S. J. Kowalski, and H. Øverby, "Systematically understanding cybersecurity economics: A survey," *Sustainability*, vol. 13, no. 24, p. 13677, 2021.

[35] T. Stevens, "Global cybersecurity: new directions in theory and methods," *Politics and Governance*, vol. 6, no. 2, pp. 1–4, 2018.

[36] Wallarm, Wallarm Learning Center, "Cia triad definition. examples of confidentiality, integrity, and availability," [accessed May 11, 2022]. [Online]. Available: https://www.wallarm.com/what/cia-triad-definition

[37] P. Mell and T. Grance, "Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme," National Inst Of Standards And Technology Gaithersburg Md Computer Security Div, Tech. Rep., 2002.

[38] "What is python? executive summary | python.org." [Online]. Available: https://www.python.org/doc/essays/blurb/

[39] G. Van Rossum *et al.*, "Python programming language." in *USENIX annual technical conference*, vol. 41, no. 1, 2007, pp. 1–36.

[40] A. Orebaugh and B. Pinkard, *Nmap in the enterprise: your guide to network scanning*. Elsevier, 2011.

[41] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US), 2008.

[42] S. Liao, C. Zhou, Y. Zhao, Z. Zhang, C. Zhang, Y. Gao, and G. Zhong, "A comprehensive detection approach of nmap: Principles, rules and experiments," in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2020, pp. 64–71.

[43] "Chapter 15. nmap reference guide | nmap network scanning." [Online]. Available: https://nmap.org/book/man.html#man-description

[44] "Nmap scripting engine (nse) | nmap network scanning." [Online]. Available: https://nmap.org/book/man.html#man-description

[45] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[46] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.", 2011.

[47] C. Burr and B. Couturier, "A gateway between gitlab ci and dirac," in *EPJ Web of Conferences*, vol. 245. EDP Sciences, 2020, p. 05026.

[48] V. M. Ionescu, "The analysis of the performance of rabbitmq and activemq," in *2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER)*. IEEE, 2015, pp. 132–137.

[49] "What can rabbitmq do for you? — rabbitmq." [Online]. Available: https://www.rabbitmq.com/features.html

[50] "Rabbitmq tutorial - "hello world!" — rabbitmq." [Online]. Available: https://www.rabbitmq.com/tutorials/tutorial-one-python.html

[51] "Flower - celery monitoring tool – flower 1.0.1 documentation." [Online]. Available: https://flower.readthedocs.io/en/latest/

[52] "marshmallow: simplified object serialization — marshmallow 3.15.0 documentation." [Online]. Available: https://marshmallow.readthedocs.io/en/stable/

[53] A. Castiglione, F. Palmieri, M. Petraglia, and R. Pizzolante, "Vulsploit: A module for semi-automatic exploitation of vulnerabilities," in *IFIP International Conference on Testing Software and Systems*. Springer, 2020, pp. 89–103.

CURRICULUM VITAE

Name Surname : Malek Malkawi

EDUCATION:

B.Sc. : 2020, Marmara University, Computer Engineering

PUBLICATIONS, PRESENTATIONS, AND PATENTS ON THE THESIS:

- M. Malkawi, T. Özyer, and R. Alhajj, "Automation of active reconnaissance phase: An automated api-based port and vulnerability scanner," in Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ser. ASONAM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 622–629.

PARALLELIZED CYBER RECONNAISSANCE AUTOMATION: A REAL-TIME AND SCHEDULED SECURITY SCANNER

ORIGINALITY REPORT

12%	6%	4%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	acikbilim.yok.gov.tr Internet Source	2%
2	Si Liao, Chenming Zhou, Yonghui Zhao, Zhiyu Zhang, Chengwei Zhang, Yayu Gao, Guohui Zhong. "A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments", 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2020 Publication	1%
3	Kuruvilla Mathew, Mujahid Tabassum, Marlene Valerie Lu Ai Siok. "A study of open ports as security vulnerabilities in common user computers", 2014 International Conference on Computational Science and Technology (ICCST), 2014 Publication	1%
4	Submitted to Loreto College Coorparoo Student Paper	<1%
	Submitted to University of Bahrain	

5	Student Paper	<1 %
6	Submitted to University of Greenwich Student Paper	<1 %
7	www.knowledgehut.com Internet Source	<1 %
8	Submitted to British University in Egypt Student Paper	<1 %
9	Submitted to Sheffield Hallam University Student Paper	<1 %
10	Submitted to UNITEC Institute of Technology Student Paper	<1 %
11	www.rmccurdy.com Internet Source	<1 %
12	Submitted to University of Hertfordshire Student Paper	<1 %
13	medium.com Internet Source	<1 %
14	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
15	Submitted to King's Own Institute Student Paper	<1 %

16	Submitted to The University of the West of Scotland Student Paper	<1 %
17	digitalcommons.unf.edu Internet Source	<1 %
18	Submitted to Coventry University Student Paper	<1 %
19	Submitted to University of Northampton Student Paper	<1 %
20	Submitted to Asian Institute of Technology Student Paper	<1 %
21	Submitted to Gitam University Student Paper	<1 %
22	Submitted to Northern Caribbean University Student Paper	<1 %
23	Submitted to South Thames College Student Paper	<1 %
24	manpages.ubuntu.com Internet Source	<1 %
25	www.ncbi.nlm.nih.gov Internet Source	<1 %
26	Submitted to Purdue University Student Paper	<1 %
27	Submitted to Wright State University	

	Student Paper	<1 %
28	digitalcommons.wku.edu Internet Source	<1 %
29	A. A. Baybulatov, V. G. Promyslov. "Cybersecurity Assessment Using Delay from Backlog Bound Calculation", 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), 2020 Publication	<1 %
30	hdl.handle.net Internet Source	<1 %
31	keep.lib.asu.edu Internet Source	<1 %
32	www.researchgate.net Internet Source	<1 %
33	Submitted to University of Westminster Student Paper	<1 %
34	tel.archives-ouvertes.fr Internet Source	<1 %
35	www.nstec.com Internet Source	<1 %
36	chennaihackers.blogspot.com Internet Source	<1 %

37	insukcho.blogspot.com Internet Source	<1%
38	Katarzyna Wilkosinska, Andreas Aderhold, Holger Graf, Yvonne Jung. "Chapter 36 Towards a Common Implementation Framework for Online Virtual Museums", Springer Science and Business Media LLC, 2013 Publication	<1%
39	github.com Internet Source	<1%
40	pure.tue.nl Internet Source	<1%
41	www.coursehero.com Internet Source	<1%
42	www.scribd.com Internet Source	<1%
43	Submitted to Chester College of Higher Education Student Paper	<1%
44	Corinna Schmitt, Marvin Weber. "Home Automation Solutions for SecureWSN", Wiley, 2022 Publication	<1%
45	circleci.com Internet Source	<1%

46	digitalscholarship.unlv.edu Internet Source	<1 %
47	docs.mipro-proceedings.com Internet Source	<1 %
48	dokumen.pub Internet Source	<1 %
49	uniquesubmission.com Internet Source	<1 %
50	waschman.blogspot.com Internet Source	<1 %
51	G. Sergio Ordoñez, T. Cesar Guerra. "Chapter 4 Prototype of a Security System with Artificial Intelligence Using Neural Networks and Evolutionary Algorithms", Springer Science and Business Media LLC, 2018 Publication	<1 %
52	Pedro Luis González Ramírez. "Interconnection Architecture of Proximity Smart IoE-Networks with Centralised Management", Universitat Politecnica de Valencia, 2022 Publication	<1 %
53	Submitted to Royal Holloway and Bedford New College Student Paper	<1 %
54	id.123dok.com	

	Internet Source	<1%
55	skemman.is Internet Source	<1%
56	su-plus.strathmore.edu Internet Source	<1%
57	Mark Lui. "Enterprise Messaging with JMS and AMQP", Pro Spring Integration, 2011 Publication	<1%
58	kalitut.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		