Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2020.013678
Article

Tech Science Press

Framework for Cybersecurity Centers to Mass Scan Networks

Waiel M. Eid[1,2], Samer Atawneh[1] and Mousa Al-Akhras[1,3,*]

[1]College of Computing and Informatics, Saudi Electronic University, Riyadh, 11673, Saudi Arabia
[2]National Cybersecurity Authority, Riyadh, Saudi Arabia
[3]Computer Information Systems Department, King Abdullah II School for Information Technology, University of Jordan, Amman, 11942, Jordan
[*]Corresponding Author: Mousa Al-Akhras. Email: m.akhras@seu.edu.sa; mousa.akhras@ju.edu.jo
Received: 17 August 2020; Accepted: 22 September 2020

Abstract: The huge number of devices available in cyberspace and the increasing number of security vulnerabilities discovered daily have added many difficulties in keeping track of security vulnerabilities, especially when not using special security tools and software. Mass scanning of the Internet has opened a broad range of possibilities for security tools that help cybersecurity centers detect weaknesses and vulnerabilities in cyberspace. However, one critical issue faced by national cybersecurity centers is the collection of information about IP addresses and subnet ranges. To develop a data collection mechanism for such information and maintain this information with continuous updates, a scanning system is needed. Therefore, this research creates a novel mass scanning framework that collects the information needed for any security investigation of cyberspace as well as preserving the obtained information for any future research analysis by the Saudi National Cybersecurity Center (NCSC), now part of the National Cybersecurity Authority (NCA). In the proposed framework, multiple instances of the scan are distributed across hosts acting as virtual scan engines, and network ranges are split among the hosts to schedule daily or weekly scanning for a wide range of IP addresses among a possible 65,535 ports per IP. In comparison with other benchmarking tools such as Nmap scanner, our proposed framework leads to faster scanning of cyberspace and greatly reduces scanning time to 30% of Nmap's scanning time.

Keywords: Network mass scan; Nmap; cybersecurity; national cybersecurity authority; National Cybersecurity Center (NCSC)

1 Introduction

With the vast number of devices available in cyberspace and the increasing number of security vulnerabilities discovered daily, it is hard to keep track of security vulnerabilities without the use of specialized tools and software. National cybersecurity centers around the world—for example, the Saudi National Cybersecurity Authority (NCA)—aim to protect national cyberspace by constantly monitoring targeted attacks and keeping up with security changes and threats [1]. For these cybersecurity centers to be proactive in protecting countries' cyberspace, they must oversee mass networks and know what vulnerabilities exist and what services are available. Nationwide cyberspace is considered a mass network based on the number of hosts and terminals that utilize a nation's network. Therefore, cybersecurity centers require customized frameworks and tools to help them achieve their goals.

Network scanning is the probing process of any network host, where a host is any computer or device connected to a network. The scanning process is performed based on an understanding of how TCP/IP protocol works to determine the information required for network security [2]. This information includes the services running on the scanned hosts, the IP address assigned to the host, and any other related information that can be collected from the host that helps the probing process. A standard scan is performed by doing a full TCP connection handshake or half-open SYN scanning [3]. However, this scanning method takes a longer time to provide results. One well-known open-source application used for port and network scanning is Network Mapper (Nmap), developed by Gordon Lyon in 1997 [4]. Since then, the Nmap tool has seen wide use in network security scans [5]. With the limitations of Internet bandwidth in its early stages, the idea of scanning the Internet seemed impossible; the time needed to scan the Internet is very long to be feasibly useful.

One issue faced by cybersecurity centers is the collection of information from specific IP addresses or subnet ranges that belong to a particular cyberspace, such as the owner of the IP address, open ports, or the services running on the cyberspace. Building a data collection method for the necessary information and maintaining this information by regularly updating it require a scanning tool to perform the necessary actions. Therefore, this research creates a framework that would include the tools necessary to provide the information required to scan the national cyberspace. These tools apply mass scanning on IP ranges reserved for a specific cyberspace. Mass scanning tools can enable multiple scanners to simultaneously investigate all ports available in the IP range given by the user, including identifying running services while avoiding redundancy among hosts during data collection. However, challenges might be encountered while creating the required framework. One challenge when scanning many subnets is that it is a time-consuming process: A scan of one normal IP can take up to five minutes; a whole subnet can take more than 20 hours using regular scanning methods. The proposed framework provides cybersecurity centers with a fast and fully functional scanner to scan any given IP ranges. The framework can collect the information needed for any required security investigation as well as preserving the obtained information in a structured database for future use/research. Users' anonymity will be maintained for research purposes.

The rest of this paper is organized as follows: Section 2 presents the necessary background and reviews the current network mass scanning literature. The methodology and the implementation details of the proposed framework are given in Section 3. The framework, experimental results, and challenges faced are presented in Section 4. Finally, Section 5 concludes the work and suggests possible avenues of future work.

2 Background and Literature Review

This section presents current networking and Internet scanning issues and reviews the available mass scanning approaches.

2.1 Network Scanning

Network scanning is the process of probing devices connected to the Internet. In recent years, several studies regarding the scanning of Internet cyberspace have been conducted. In 2010, Dereck Leonard and Dmitri Loguinov searched for a way to implement an Internet scanner service tool. They developed a way to maximize the politeness (i.e., to lessen complaints resulting from scanning the IPs) of a remote network to scan the Internet in minutes or hours instead of weeks or months [6]. Their research resulted in a tool called IRLScanner, which is a high-performance, source-IP scalable tool for service discovery on the Internet based on the measurement conducted [7]. IRLScanner's custom network driver can transmit SYN packets at 5.25 Mb/s using a modern quad-core CPU, assuming 3.5 Gbps of network bandwidth is available. IRLScanner can cover the entire BGP space in 6.7 minutes from a single server. Fig. 1 describes the implementation of the IRLScanner tool [6].

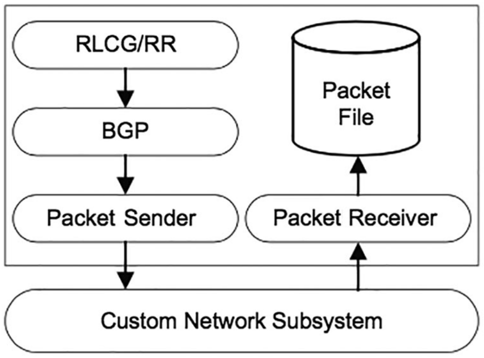

Figure 1: IRLScanner implementation [6]

2.2 Mass Scanning Methods

In 2013, Durumeric et al. [8] developed a modular open-source tool called ZMap to perform fast Internet-wide scanning. ZMap is capable of scanning the entire Internet for one port on a single probe in around 45 minutes using a gigabit Ethernet connection. This tool was built based on the only available network tool and widely used Nmap network scanning [9]. However, the tool framework was constructed based on an asynchronous connection with a customized TCP request. ZMap eliminates the local pre-connection state. It scans widely dispersed targets as fast as the network allows and probes optimized network stacks to bypass inefficiencies by generating Ethernet frames. Fig. 2 shows the ZMap tool framework.

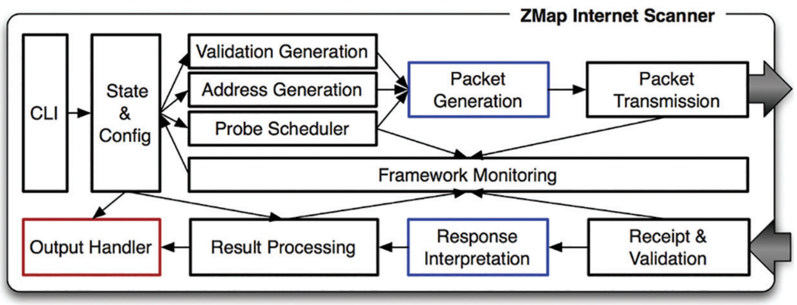

Figure 2: ZMap framework [8]

In the ZMap framework, the packet generation and response interpretation modules support multiple probe types, such as TCP SYN scans and ICMP echo scans. The output handler module allows users to output or act on scan results in application-specific ways. This framework allows sending and receiving components to run asynchronously. It also enables a single source machine to comprehensively scan every host in a public IPv4 address space for a particular open TCP port in less than 45 minutes using a 1 Gbps Ethernet link [8]. The ZMap tool has seen extensive usage in monitoring security applications and tracking availability and protocol adoption, and can be used to enumerate vulnerable host lists and discover unannounced services [8].

In 2014, Robert Graham developed MassScan [10]. The MassScan tool is an open-source Internet modular scan developed based on asynchronous transmission. The fundamental difference between MassScan and other scanners is in the way that MassScan randomizes targets by scanning one port at a time. The main principle is to have a single index variable that starts at zero and is incremented by one for every probe. In addition, MassScan uses a custom TCP/IP stack (mainly for the port scan) [11]. The tool can scan the entire IPv4 Internet in less than 6 minutes for the range of known ports (65,535 ports), but it requires a 10 gigabit Ethernet adapter and a connection with similar bandwidth. In addition, MassScan can fetch the information from scanning ports, such as banner checking. The tool can support SCTP scanning and UDP Nmap payloads, and can also parse Nmap options and produce formats similar to Nmap [9,11]. MassScan also relies on a single port at a time instead of one host at a time. The MassScan tool can do spoof scanning, as it can set the scan on one IP and receive the scan results on a different IP. The tool also has the ability to split the scan among multiple machines [11].

In 2015, Durumeric et al. [12] noticed growth in Internet-wide scanning, which drove security research based on scanning techniques and tools. Based on this, Durumeric et al. built a search engine backed by data from extensive Internet data collection and made it available to the public. The resultant search engine, given the name Censys, supports researchers in answering security-related questions [12]. The Censys search engine collects the data through scheduled host discovery scans using ZMap. It performs a full scan and stores the data required by the search engine in a custom database engine. After that, the collected data is available to researchers through a web application program interface (API) using Google's data interchange format [13] and big data query system, as shown in Fig. 3. The resultant work produced a custom scanning tool called ZGrap. The ZGrap scanner supports the application handshakes of HTTP, HTTP Proxy, HTTPS, SMTP(S), IMAP(S), POP3(S), FTP, CWMP, SSH, and Modbus, as well as StartTLS, Heartbleed, SSLv3, and specific cipher suite checks [12].

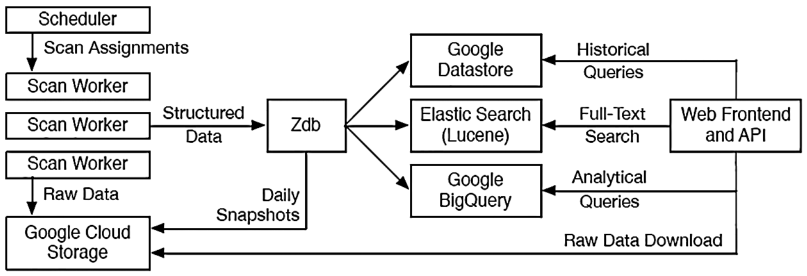

Figure 3: Censys system architecture [12]

In 2016, Jicha et al. [14] identified the devices that run through IPv4 address space. They determined the type and vulnerability level of each device, with a focus on supervisory control and data acquisition (SCADA) devices. The researchers produced a tool they called Amap by combining a connectionless scan and connection-oriented scanners to reduce the time needed for the scan and assure its accuracy [15]. Nmap was used for connection-oriented scans, and ZMap for mass or connectionless scans [14].

In 2017, Rohrmann et al. [16] conducted a large-scale port scanning through the Tor network. The attempt was to achieve anonymization over the scan to avoid detection by the host. The drawback of the tool was that it was not scalable to scan the entire IPv4 range on multiple ports [16]. However, port scanning was possible for specific ports while anonymous. The scan was performed using multiple instances of Nmap tool over a purpose-built virtual machine. In 2019, another attempt by Shi et al. [17]

was conducted to scan a large-scale network to build a penetration-testing framework based on network fingerprint. Although the focus was on penetration testing, they needed a way to get results for a large-scale network. Their framework needed a way to do a stateless scanning of the mass network which was achieved using ZMap and Mass scan tool.

In summary, the literature shows that the development of mass scanning is growing where different scanning tools and methods are available for scanning the cyber/Internet space. The available mass scanning methods and tools help security industries in data gathering and monitor the information security in cyberspace.

The need for mass scanning can be fulfilled using the framework proposed in this paper. The goal is to offer a mechanism for scanning nationwide cyberspace, as commonly needed in national cybersecurity centers for national security needs. Multiple instances of the scan are distributed across hosts, which can be virtual, and network ranges are split among the hosts to schedule IP scanning for a wide range on the 65,535 ports per IP in a daily or weekly matter as needed. The National Cybersecurity Centre (NCSC) in the Kingdom of Saudi Arabia—An organization that has become part of the Saudi National Cybersecurity Authority (NCA)—and similar national cybersecurity centers in the world may offer services such as scanning cyberspace for the state for many considerations regarding national security. According to our best knowledge, there is no similar study or published implementation regarding the use of mass scanning. We implement and test a software tool to offer mass scanning services within the Kingdom of Saudi Arabia.

3 The Proposed Framework

This section presents the proposed framework, which uses mass scanning tools to scan massive ranges of IPs to collect information that would help cybersecurity centers in achieving their national security needs. Extensive research in implementing mass scanning of the Internet with different frameworks has been applied. These implementations led us to create multiple security applications with specific goals. This paper aims to build a framework that satisfies the needs for the NCSC security research. The proposed framework must achieve the following main capabilities:

- The ability to identify all active hosts within the Saudi cyberspace, which is estimated to be around 5 million hosts.
- Port scanning is required for all available ports (65,535 ports).
- The ability to perform scans periodically, where scanning speed is critical (e.g., weekly, monthly, or by request).
- The ability to run multiple scanning engines and pull results back to a centralized management console.
- The ability to grab banners during scanning in order to identify running services.
- The ability to scan TCP and UDP ports.
- A standardized scanning output for scanning purposes derived from multiple agents that enable collecting different results.
- The ability to save and browse the previous scan history.
- Results stored in a big-data ready database, enabling fast queries.
- A visualization process for the scanned results.

The above capabilities are based on daily operations related to national cybersecurity centers. Speed and ability to expand the functionality of the framework are key elements. The design should follow a modular approach to achieve these characteristics. The needed capabilities specify the need for a scanner that can identify active hosts and perform port scanning. Additionally, the framework should be able to run multiple scanned instances. However, there is no specification regarding whether the instances of the scan must run on the same machine or on multiple machines. In addition to scanning functionality, other types of information are required to be fetched, such that the requirements can only specify the exact port banner. The possibility of obtaining other types of information is offered when using a modular build approach. The output of the scan needs to be standardized and formatted in a specific format that can be stored in the database. The expected volume of output varies based on the number of agents; hence, the design must consider a huge number of results, which are streamlined from many agents. The data storage should also be able to keep all received results. Some of these requirements can be delivered by open-source applications, which can be customized or grouped to create the required framework.

3.1 Design Phase

This section describes the system architecture in terms of components that constitute the framework.

3.1.1 Framework Architecture

The proposed framework scans cyberspace by distributing instances of the scan across multiple hosts.

Distributed Environment

We followed a modular approach to design the framework's architecture. This structure allows us to expand the framework in the future by adding further features as needed. The proposed design divides the framework's architecture into three main components: the main manager (edge manager), scan engine, and a backend database. The design of the framework follows a multilayered architecture in order to achieve the most effective security. Therefore, the main framework components operate in a distributed environment. The system architecture components, delineated in Fig. 4, are presented in the following subsections.

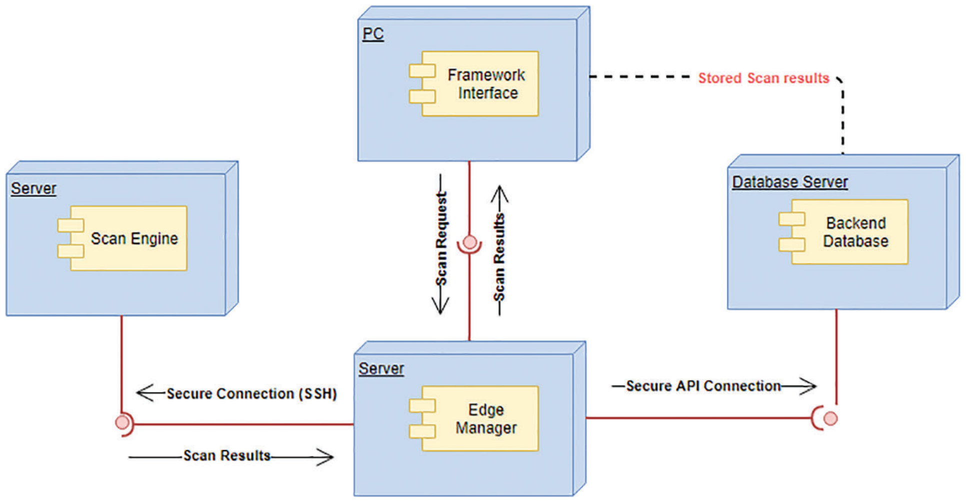

Figure 4: Overall framework architecture

Main Manager (Edge Manager)

The user interacts with the framework through the edge manager interface, which acts as the main interface. The edge manager acts as a middleware between the other components of the application, linking all the components together. The edge manager sets the required actions to other components and processes the output from other components. The main functions of the edge manager are to accept scan requests from the user, manage the scan engines, receive results, and organize their storage in the database. The edge manager possesses a logical tool for determining how to achieve the scanning process and searches only for the required tools that should be run. This logic tool also understands when to execute the required actions in the scanning process. Additionally, the main manager processes the output provided to users, then formats it for storage in the backend database. Fig. 5 highlights the relationships among the framework objects and shows the roles of the edge manager relative to other objects.

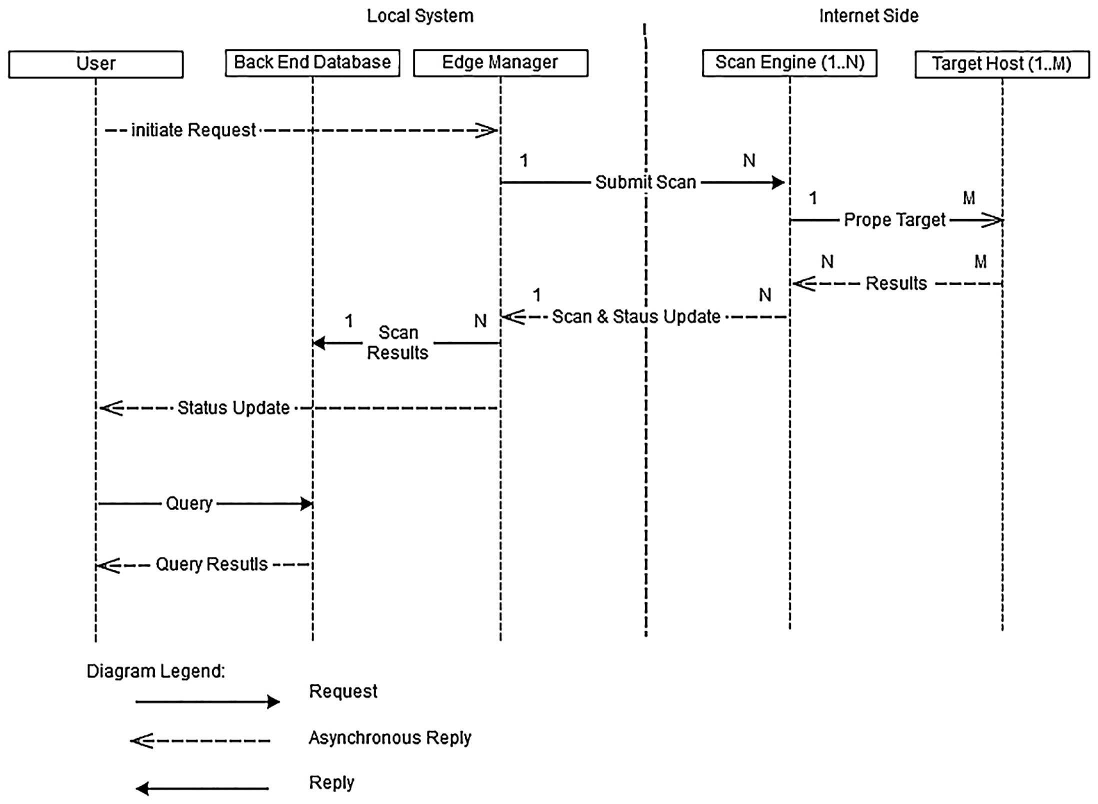

Figure 5: Edge framework design

Scan Engine

The scan engine controls process of probing the targeted hosts over the Internet. The scan engine is required to be online and have sufficient bandwidth for running the mass scan of the provided IPs. A user submits the request from the edge manager, and the scan engine performs the required scan in order to collect any existing information it can obtain about the targeted hosts during the probing process. At this stage, a main prerequisite specified in the requirements is to check whether the status of the targeted host is "alive." After that, the port scans are applied and the information about them is collected. While scanning and probing the targeted host, the scan engine updates the edge manager with the status of the scan instantly, enabling the edge manager to show the status and the results obtained across different scan engines. The scan engine should also be able to probe multiple targets at the same time and collect information for each host separately. At this stage, the scan engine should run multiple scanned jobs and differentiate between the jobs and their results. It should also report each task's status and results to the edge manager. All operations should be performed asynchronously to achieve the maximum speed in running the jobs out of the scan engine.

Backend Database

Scan results are saved in the backend database for future use by the cybersecurity center and in anonymously for data-mining research. In this stage, the collected information are stored asynchronously as received from the multiple scan engine through the edge manager. Therefore, the backend database handles persistent data input (streams of input) with different structures or records and dynamically processes it. Further, the backend database supports API in order to store the data. The backend database should also be able to provide historical results when queried by a user, while the query should not affect storing the streamed results. Moreover, supporting a large data structure (data that is too varied, fast-changing or enormous for conventional technologies) is essential for future research developments.

3.1.2 Framework Process Flow

As mentioned earlier, the overall framework design follows a layered structure (see Fig. 4). The framework is created based on a multilayer architecture in which the edge manger runs on a server located in the demilitarized zone (DMZ). In fact, the framework requires being available on the Internet to receive the results from the scan engines when they are initiated. The results return on various ports to support the asynchronous connection, which is kept in the DMZ farm. Consequently, the firewall that controls access to the zones for network security purposes separates the DMZ farm, user farm, and systems farm.

The scan engine is hosted on a virtual private server (VPS) farm that is available through any cloud service provider, such as Amazon Web Services [18] or Digital Ocean [19]. The edge manager communicates with the scan engine through a secure connection using Secure Shell (SSH), which is used to secure the communication between the two components (manager and scanners). The results obtained from the scan engine are also transmitted through the SSH connection. The backend (i.e., the database) is hosted at the server farm, where it does not have any access from or to the Internet. The edge manager pushes the results through a secure API connection (usually through HTTPS) to store the results in the database. The pushed data are at a dynamic length and are represented and sent in JavaScript Object Notation (JSON) format.

Fig. 6 illustrates the connection flow among the framework components in the Internet space. In fact, this architecture starts from the user side by sending a request to the edge manager to initialize the scan request. The edge manager performs the scan process and initializes the SSH connection to the VPS farm to request the scanning of the targeted host. The scan engine in the VPS farm begins to probe the targeted hosts in the subnets and receives the obtained results, which are accordingly redirected to the edge manger. Once the scan is conducted, the connection with the scan engine is terminated. The edge manager pushes any received result from the scan engine during the scanning process and towards the backend database so that it stores it effectively and displays the results for the user through the graphic user interface (GUI) interface.

3.2 Implementation

This section presents the fundamental processes and techniques used to create the proposed framework. The framework offers mass scanning functionalities using benchmark tools and technologies. The approach to be followed for implementing the framework is to utilize open-source technologies that can assist in building the framework to enhance the framework capabilities and reduce the development time. The development focuses on three modules: The edge manager, the scan engine, and the backend database. The following subsections present the technologies and tools used for implementing the proposed framework.

3.2.1 Technologies Used (Masscan and Elastic Stack)

The technologies used for developing the proposed framework are Masscan and Elastic Stack, which are discussed below.

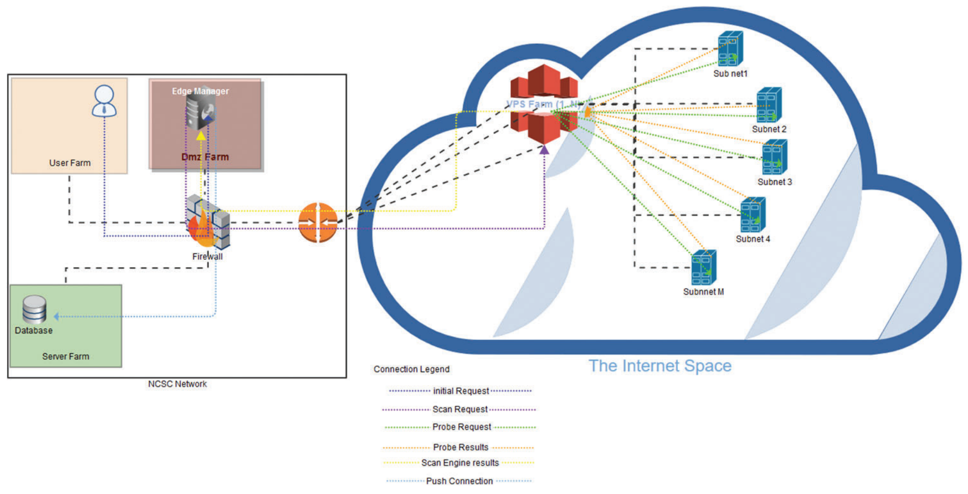

Figure 6: Connection flow among framework components

Masscan

The scan engine relies on using the Masscan tool as the main IP port scanner. As indicated in the literature, the Masscan is an open-source port-scanner tool that provides similar results to those of the Nmap tool. This tool is chosen as a feature scanner because it relies on using asynchronous transmission, which implies that separate transmit threads and receive threads are largely independent of each other [11]. Therefore, the tool helps with speeding up the scanning results. Additionally, such a tool focuses on port scanning rather than host scanning, which implies that it takes the ports and scans the ranges for hosts with the ports open. The tool can also support scanning a wide number of ports. Another reason for using Masscan is that it enables one to have the output in multiple formats. This assists in customizing the format of the obtained results according to users' needs. Moreover, this tool provides the ability to control the transmission rate by allowing users to control the bandwidth within a limited bandwidth environment, if needed [11].

ELK Stack (Elastic Stack)

Elastic Stack is an integrated solution that contains three open-source products consisting of the **E**lasticsearch, **L**ogstash, and **K**ibana [20]. It functions as an end-to-end stack that delivers actionable insights in real-time systems from almost any type of structured and unstructured data source. Elasticsearch is an open-source, readily scalable, broadly distributed, and enterprise-grade search engine that is accessible through an elaborated and extensive API. It can perform extremely fast searches to support user data discovery applications [21]. Elasticsearch analytics provides mathematical transformations on data, including enhanced aggregation processes.

Logstash is an open-source tool for log data intakes, processing, and the output of various systems. This tool collects and parses the data, then stores them in an indexed way to be retrieved more quickly. Additionally, it can ingest data from multiple sources simultaneously. Moreover, it can parse and dynamically transform the received data to a structured form.

Kibana is an open-source real-time data visualizer for Elasticsearch, as well as an exploration platform. Such a visualizer is particularly designed for a large volume of streams; its aim is to visualize the logs in a time-stamped way [21]. Kibana integrates with the Elasticsearch search engine to represent the aggregated data into a graphical representation for various analytics purposes.

3.2.2 Backend Implementation

To implement the proposed framework, we used the Go programming language [22], which represents a simple open-source programming language that assists developers in making an easy and reliable software. The main features of the Go language comprise automatic performance, efficient memory management, and concurrency primitives [23]. Go's functions can run in parallel due to its modelling types of threads, which are called Goroutines. Compared with other languages, Goroutines are considered to be lightweight. The synchronization method used among Goroutines represents a channel of communications. The channel is primitive and provides a way for two concurrently executing functions to synchronize and communicate by passing the value of a specific element type, including the send operation, in a single channel.

3.2.3 Frontend Implementation

The frontend of the implementation represents a web-based system, which uses HTML5 to support the latest technologies, including Javascript. Because Javascript supports asynchronous API calls, including methods for handling the data that it receives, it was used for the Vue.JS framework as a means of implementing the frontend. Several reasons exist for using the web-based frontend according to [24]. These include the following:

- Web-based frontend systems are easier to develop and customize.
- It is cross platform, where systems can easily be ported to any virtual platform with a web browser.
- It is easier to expand as the program grows.
- It can operate on any device (desktop, mobile, or tablet) based on a responsive design.

3.2.4 Utilized Frameworks

The proposed framework uses Vue.JS and AdminLTE frameworks, which helps with the implementation and speeds up the development process. These frameworks are chosen with the ability to be integrated with each other as one product, as well as to provide the required functionality. The frameworks are presented in the following paragraphs.

Vue.JS

Vue.JS is a progressive framework that helps with building the user interface. In fact, this framework is designed to be adaptable to and focused on the view layer that is easy to be integrated with other libraries and projects [25]. Vue.JS framework is applied due to its rendering performance, API simplicity, size, and learning curve compared with other related frameworks. Additionally, the framework supports components that can be added to the framework for the purpose of increasing its functionality.

AdminLTE

AdminLTE is an open-source template for control panels. It uses HTML5 and CSS3, and it supports responsive design, which makes it fast and lightweight. Furthermore, it supports most of the major browsers by accepting a plugin software to enhance its features [26]. Our proposed framework uses AdminLTE to provide a unified interface across the media and to provide a unified user experience by using responsive design. AdminLTE is combined with Vue.JS to provide the frontend control panel for the proposed framework. Vue.JS acts as the data parser when the data are passed from the edge manger, and it renders the data in the AdminLTE template to obtain the final result. Fig. 7 illustrates the flow of data that is received from the edge manager to the Vue.JS framework, which parses the data and passes it to the AdminLTE template. The AdminLTE framework renders the data and the AdminLTE template to display the result to the end user in HTML5 format, which is updated when data from the edge manager are requested directly. In fact, this request does not require rendering the AdminLTE template again through the Vue.JS scripts. AdminLTE and Vue.JS are merged in a fully responsive admin template called Copilot [27].

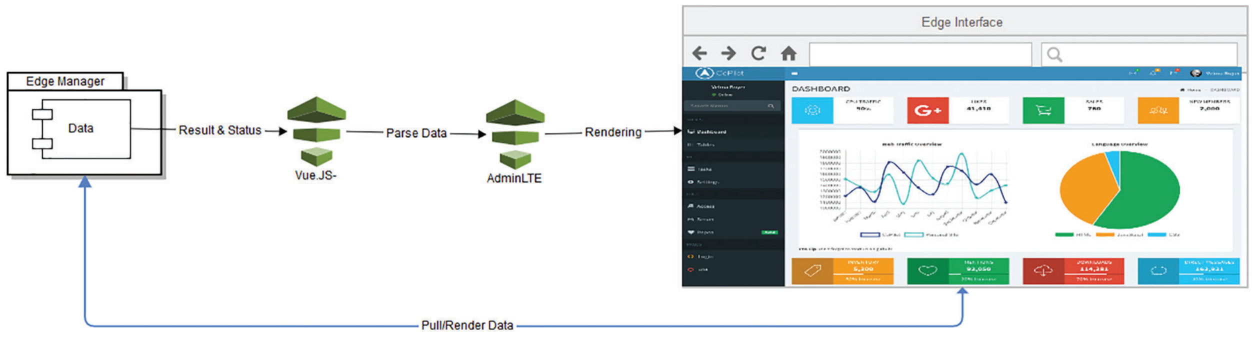

Figure 7: Illustration of data flow from edge manager to Vue.JS framework

3.2.5 Edge Manager Modules

The edge manager compromises four main modules as depicted in Fig. 8. These modules are as follows:

1. The Controllers module represents the main control of the program, which controls the execution of the scan jobs and keeps prioritizing the execution of the job.
2. The Job module contains the scanner definition and the interfaces for each involved job while keeping track of the state of the scanner.
3. The Model module manages the scan servers, keeps track of each server's status, and executes tasks through them. It also provides the statuses of the connections and sessions.
4. The Handler module contains the API web services that handle the requests coming from the GUI interface.

Each module has a set of classes that identify the purpose of the module. The three main classes are the Server class, the Scanner class, and the Job class.

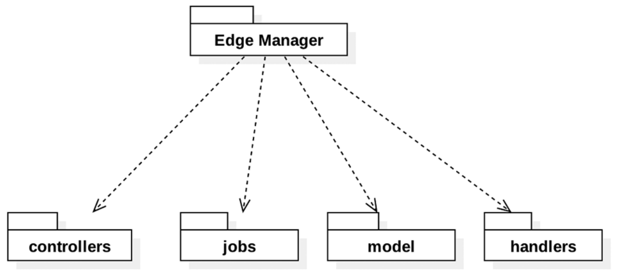

Figure 8: Edge manager modules

Server Class

The Server class is located under the Model module and contains the information related to the server (VPS server) by which the edge manager will connect to perform the scanning request. The server information consists of the username, password, and IP address. Additionally, it has status information consisting of the currently running jobs on the server, the network status (bandwidth), and the connection client (SSH client). Fig. 9 shows the Server class properties. Both *RunningJobs* and *NetworkStats* attributes form a single access value.

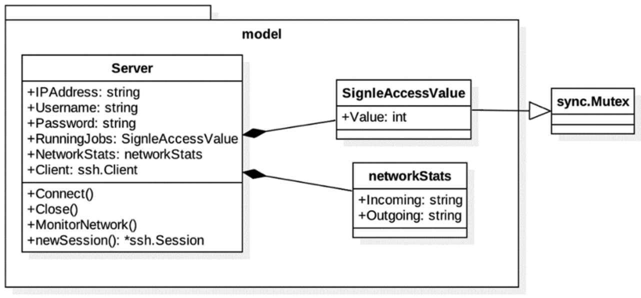

Figure 9: Server class properties

Scanner Class

The Scanner class is located under the Jobs module and contains information related to the scan itself. The Scanner class collects the timestamp, the process identification number, and the status of the running scans. The Scanner class is a special form of the Job class and is made available to implement it. Fig. 10 shows the Scanner class and its relation to the Job class.

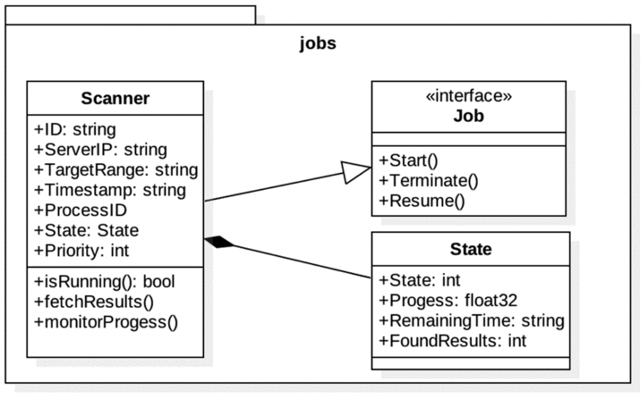

Figure 10: Scanner and Job classes in Job module

Job Class

The Job class represents the actual job request. Whenever a request comes toward the edge manager, a job record is created. The edge manager determines the number of jobs that are required to run the request as it divides the request into multiple jobs based on the following criteria: If the scanned subnet is bigger than /24, the subnet is divided into multiple/24 subnets, where each subnet creates a job request for it. The idea behind splitting the subnets refers to the efficient speed in scanning a range of the same subnet (/24), which usually belongs to the same entity.

3.2.6 Edge Scanner for NCA

The developed framework combines the edge manager and the GUI interface by which the user gains access to perform the scanning request. This produces a scanning tool, given the name Edge Scanner, for the NCA at Saudi Arabia. The user performs an interfacing process through the web GUI. The flow of the traffic begins with the web GUI once a user provides the subnet that should be scanned. The web GUI sends the request to the edge manager, which divides the subnet into 24 subnets when needed, where each subnet is deemed a scanning job. The edge manager assigns the required resources to run the scanning job (VPS server) and adds the job into the queue such that this job awaits the user's command to start the scanning process. Once the user's command is received, the edge manager connects to the resource server through SSH and executes the search job. While the resource server executes the jobs, the results and the status of the jobs are sent in real time to the edge manger. The edge manager sends the job updates to the GUI interface so that it can be displayed to the user. In addition, the edge manager sends the results of the scan to the backend database through the backend APIs. Fig. 11 illustrates the components of the Edge Scanner and shows how they work to perform the required task.

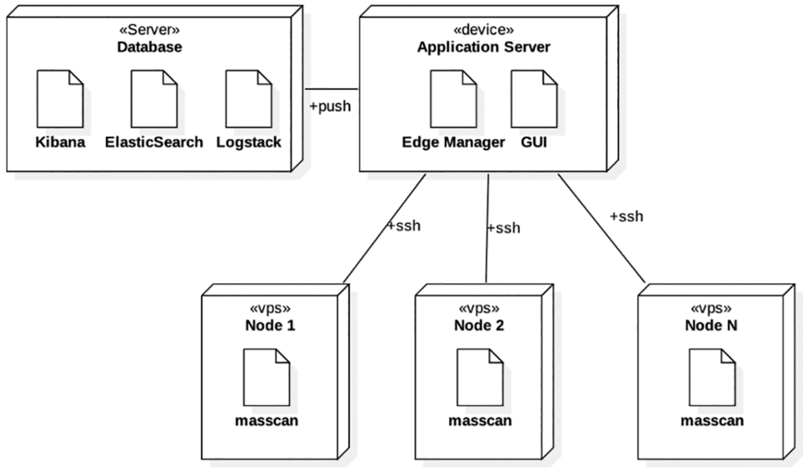

Figure 11: Edge Scanner components

4 Experimental Analysis and Results

The methodology of the developed framework and its architecture, as well as the fundamental processes and techniques used to implement the proposed framework, are presented in Section 3. This section presents various challenges that emerged during the development of the framework. Moreover, several issues also emerged during the testing phase, which have been managed by using the most effective solutions. Finally, the results of the test were validated by confirming the approval of accuracy for the obtained results.

4.1 Challenges

During a mass scan on an IP range, the host or service provider faces a high risk of being reported when it comes to various unauthorized network-scanning activities. In fact, this action may result in legal and financial complications. During the development of the framework, an approach to testing the scanning build was taken into account. While applying the actual scanning and probing on a range of IPs, host owners could contact the service provider used in the test to complain about the probing activity of the developed system. To avoid such issues, the NCSC, which is now a part of the NCA, provides an IP range that acquires an owner's approval to perform multiple network scanning processes and probe their live hosts.

4.2 Results Validation

For the purpose of assuring the accuracy of the Edge Scanner implementation, the results should be validated. According to the NCSC, the results must be at least 90% accurate when being recognized. Validating the results involves taking a sample of the results of any random IPs from the framework scan. Another full connection scan is performed on the IPs using the Nmap scanner. The results from the Nmap are used as the baseline for the validation. A comparison between the results of both scans (the developed framework and Nmap) is performed, and two outcomes should not differ by more than 10%. Exceeding a 10% threshold is considered to be unacceptable according to the NCSC standard.

4.3 Results Evaluation

The developed Edge Scanner was tested to assure its validity. Due to the legality challenges mentioned in Section 4.1, the test was performed on the subnet 176.xxx.yyy.0/20 (IPs masked for security reasons), which targets around 4096 possible hosts. This subnet assures that a certain number of ports are open on a certain number of IPs. The edge manager splits the subnet into multiple subnets of base /24 and starts the scan on each subnet. Tab. 1 shows the results of the scan process and the number of ports detected on each subnet.

Table 1: Samples of scan results

Subnet	No. Hosts	Proposed edge scanner		Nmap scanner	
		No. Ports	Time	No. Ports	Time
176.xxx.yyy.0/24	256	0	00:01:02	0	03:02:46
176.xxx.yyy+1.0/24	256	0	00:01:02	0	
176.xxx.yyy+2.0/24	256	0	00:01:00	0	
176.xxx.yyy+3.0/24	256	0	00:00:58	0	
176.xxx.yyy+4.0/24	256	2	00:01:07	2	
176.xxx.yyy+5.0/24	256	2	00:01:12	2	
176.xxx.yyy+6.0/24	256	2	00:01:09	2	
176.xxx.yyy+7.0/24	256	0	00:01:02	0	
176.xxx.yyy+8.0/24	256	0	00:01:03	0	
176.xxx.yyy+9.0/24	256	0	00:01:02	0	
176.xxx.yyy+10.0/24	256	0	00:01:02	0	
176.xxx.yyy+11.0/24	256	0	00:01:02	0	
176.xxx.yyy+12.0/24	256	0	00:01:02	0	
176.xxx.yyy+13.0/24	256	0	00:01:04	0	
176.xxx.yyy+14.0/24	256	0	00:01:02	0	
176.xxx.yyy+15.0/24	256	0	00:01:02	0	
Total	4096	6		6	
Scanning time average			00:01:03		

The findings from the proposed Edge Scanner demonstrate six ports open on the entire subnet. These results are consistent with the devices on the network because the six ports are not commonly open and are open only based on requests originating from these subnets. This implies that the host should not have a port open unless necessary for particular management purposes. The entire scanning operation that was conducted with the framework (scanning 4096 hosts) took around eight minutes.

The same subnet was scanned by using the benchmark Nmap scanner for validation purposes, where similar results were obtained. However, the process took longer (around three hours). The reason behind this delay has to do with the full sync scanning connection, which ensures that the ports are open. In addition, various tests were conducted on different subnets, and similar results to those of the Nmap scan were obtained. This concludes that the proposed framework not only can work per the NCSC acceptance standards but also can provide full functions and be applied to scan networks via cybersecurity centers, including the NCSC.

5 Conclusions and Future Work

The scanning framework proposed in this paper contributed to the efficient collection of the information needed to investigate cyberspace. The findings revealed that the framework led to faster scanning for cyberspace and greatly reduced the scanning time compared with other benchmarking tools. The main contributions of this research are as follows: (1) Developing a full functional framework that establishes a national cybersecurity database of IPs, ports, and services; (2) Reducing the time needed to collect data when cyberspace is scanned through multiple VPSs; and (3) Allowing cybersecurity centers to become proactive in detecting possible threats that target cyberspace.

One possible enhancement over the developed framework is to create a modular threat assessment, which can provide additional information about possible threats to the IP address within the database. Additionally, it is possible to automate the creation of the VPS through the API provider to automate the creation of scan servers, as well as the distribution of the IPs in it. Consequently, this automation provides a wide coverage and speeds up the scanning process. Another possible enhancement is to generalize the framework by including machine learning to assist cybersecurity centers in detecting various possible attacks. At the same time, it is important to understand the risks and to assist in the necessary security measures to protect against possible cyber terrorist attacks. This would include alerts regarding a pattern of possible ongoing attacks.

Acknowledgement: The APC was funded by the Deanship of Scientific Research, Saudi Electronic University.

Funding Statement: The authors received no specific funding for this study.

Conflict of Interest: All authors declare that they have no conflict of interest.

References

[1] N. C. Authority, “NCSC: National cyber security centre.” 2020. Available: https://nca.gov.sa/en/pages/ncsc.html.

[2] R. Ritchey, B. O’Berry and S. Noel, “Representing TCP/IP connectivity for topological analysis of network security,” in *IEEE Proc. 18th Annual Computer Security Applications Conf.*, pp. 25–31, 2002.

[3] C. Roger, *Port Scanning Techniques and the Defense Against Them*, Bethesda, MD: SANS Institute, 2001.

[4] G. Lyon, “Nmap: The art of port scanning.” 2020. [Online]. Available: https://nmap.org/nmap_doc.html.

[5] G. Kaur and N. Kaur, “Penetration testing-reconnaissance with NMAP tool,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, pp. 844–846, 2017.

[6] D. Leonard and D. Loguinov, “Demystifying service discovery: Implementing an internet-wide scanner,” in *Proc. of the 10th ACM SIGCOMM Conf. on Internet Measurement*, pp. 109–122, 2010.

[7] E. Shakshuki, D. Benoit and A. Trudel, “World’s first web census,” *International Journal of Web Information Systems*, vol. 3, no. 4, pp. 378–389, 2007.

[8] Z. Durumeric, E. Wustrow and J. A. Halderman, “ZMap: Fast internet-wide scanning and its security applications,” in *Presented as Part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13),* pp. 605–620, 2013.

[9] G. F. Lyon, “Nmap network scanning: the official Nmap project guide to network discovery and security scanning,” *Insecure*, 2009.

[10] R. Graham, "TCP port scanner, spews SYN packets asynchronously, scanning entire Internet in under 5 minutes." 2020. [Online]. Available: https://github.com/robertdavidgraham/masscan.

[11] R. Graham, P. McMillan and D. Tentler, "Mass scanning the internet." 2020. [Online]. Available: https://www.defcon.org/images/defcon-22/dc-22-presentations/Graham-McMillan-Tentler/DEFCON-22-Graham-McMillan-Tentler-Masscaning-the-Internet.pdf.

[12] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security*, pp. 542–553, 2015.

[13] K. Varda, "Protocol buffers: Google's data interchange format." 2020. [Online]. Available: https://github.com/google/protobuf.

[14] R. Jicha, M. W. Patton and H. Chen, "Identifying devices across the IPv4 address space," in *2016 IEEE Conf. on Intelligence and Security Informatics: IEEE*, pp. 199–201, 2016.

[15] W. A. H. Ghanem and B. Belaton, "Improving accuracy of applications fingerprinting on local networks using NMAP-AMAP-ETTERCAP as a hybrid framework," in *2013 IEEE Int. Conf. on Control System, Computing and Engineering: IEEE*, pp. 403–407, 2013.

[16] R. R. Rohrmann, V. J. Ercolani and M. W. Patton, "Large scale port scanning through tor using parallel Nmap scans to scan large portions of the IPv4 range," in *2017 IEEE Int. Conf. on Intelligence and Security Informatics: IEEE*, pp. 185–187, 2017.

[17] P. Shi, F. Qin, R. Cheng and K. Zhu, "The penetration testing framework for large-scale network based on network fingerprint," in *2019 Int. Conf. on Communications, Information System and Computer Engineering (CISCE): IEEE*, pp. 378–381, 2019.

[18] Amazon AWS, 2020. [Online]. Available: https://aws.amazon.com/.

[19] Digital Ocean, 2020. [Online]. Available: https://www.digitalocean.com/.

[20] Elastic.co, "The elastic stack: open source search & analytics." 2020. [Online]. Available: https://www.elastic.co/.

[21] J. Vanderzyden, "Welcome to the ELK stack: Elasticsearch, logstash and kibana." 2020. [Online]. Available: https://qbox.io/blog/welcome-to-the-elk-stack-elasticsearch-logstash-kibana/.

[22] Golang, 2020. [Online]. Available: https://www.golang.org/.

[23] S. Hawthorne, "A language comparison between parallel programming features of Go and C," 2014. [Online]. Available: http://www.sjsu.edu/people/robert.chun/courses/cs159/s3/S.pdf.

[24] Core Solution, "The benefits of web-based applications." Accessed May 1, 2020. Available: https://coresolutions.ca/blogs/core-web/the-benefits-of-web-based-applications.

[25] Vue.js, "Introduction - What is Vue.js?." 2020. [Online]. Available: https://vuejs.org/v2/guide.

[26] AdminLTE, "AdminLTE bootstrap admin dashboard template." 2020. [Online]. Available: https://www.adminlte.io/.

[27] GitHub, "misterGF/CoPilot." 2020. [Online]. Available: https://www.github.com/misterGF/CoPilot.