

A Security Scanner Toolkit

STMU CYBERSECURITY MS COMPREHENSIVE PROJECT

Eluwa Chidera Duanne Anita

August 5th, 2022

St. Mary's University

(210) 995-4053

celuwa@mail.stmarytx.edu

Table of Contents

Overview & Background	3
Problem.....	3
Background.....	3
Existing Products	4
Security Scanner	5
Requirements	7
Developer Software & Hardware Requirements	7
User Software & Hardware Requirements	8
Functional Requirements	9
Design	10
Software Modules.....	10
Interface Design.....	18
Installation	21
Method 1	21
Method 2	22
Sample Sessions.....	23
Home Page.....	23
Network scan results	23
Ping scan results	24
Ping Sweep Results.....	25
Fast Scan Result.....	26
Regular Scan Result.....	26
Extensive Scan Results	27
Vulnerability Scan Results	28
Task List	30
References.....	32
Appendix: Source Code.....	33

Table of Figures

Figure 1 Home Page	18
Figure 2 Network scan.....	18
Figure 3 ping scan.....	19
Figure 4 ping sweep.....	19
Figure 5 Fast scan	19
Figure 6 Regular scan	20
Figure 7 Extensive scan	20
Figure 8 Vulnerability Scan.....	20
Figure 9 Home front page.....	23
Figure 10 network scan result.....	24
Figure 11 network scan result.....	24
Figure 12 ping scan result.....	25
Figure 13 ping scan result.....	25
Figure 14 ping sweep result	26
Figure 15 Fast scan result	26
Figure 16 regular scan result (open ports)	27
Figure 17 regular scan result (no open ports)	27
Figure 18 extensive scan result (port).....	28
Figure 19 extensive scan result (os info)	28
Figure 20 vulnerability scan result	28
Figure 21 vulnerability scan result	29
Figure 22 vulnerability scan result	29
Figure 23 vulnerability scan result	29
Figure 24 vulnerability scan result	30
Figure 25 task list.....	30
Figure 26 task list.....	31
Figure 27 task list.....	31

A Security Scanner Toolkit

Overview & Background

This report is divided into various sections. Background sections cover the history of cyberattacks that occur over time and the motivation behind creating a security scanner. The requirements section highlights the information the user must know i.e., which hardware and software to use, which packages were needed to create this project, and the functional requirements for the software. A detailed design that explains what the software does and how it needs to function is included in the design section. The design section will also cover any links between the components and external interfaces. The installation section contains a listing of the source files and includes information about installation and regular usage for a user. Within the sample sessions section, screenshots of every module of the software are included. It also includes a common problem chart with workable solutions. The references section will have references used throughout the duration of the project. The concluding section in this manual is an appendix containing all source code for each module with accompanying screenshots.

Problem

From past experiences, negligence in securing the system not only results in loss of information but causes huge disruption and loss of funds as well. Any vulnerability in the system allows attackers to exploit it. An attacker can gain access to the system if ports are open, unsecured and if vulnerable applications are running on those ports. The attacker utilizes this opportunity to attack the system and gains access. Thus, securing the system has become a fundamental requirement. The first step towards securing the system and network is to know their vulnerabilities.

Background

With the advent of the internet, there is no denying that it transforms the world mostly for the better, but it also intrudes cyber threats that are continuously evolving. The first-ever computer virus was witnessed in 1988 (Uhde, 2017) when a worm designed by Robert Morris to propagate across the

network failed and results in a clogged network and slowed internet that triggered the system to collapse. In 1999, another virus named Melissa was discovered (Hook, 2019) causing millions of dollars in harm followed by Mafia boy in 2000 which cost multinational companies such as Yahoo, Fifa.com, Dell, and eBay around \$1.2 billion collectively. Yahoo had been a victim of the state-sponsored attack and almost 500 accounts have been compromised (Shahani, 2016). In 2018, an unknown marketing firm leaked the data of 340 million individuals including their personal information such as name, address, interests, habits, religion, occupation, and the number of children. All these were because of poor security practices. Even so, one of the biggest social media platforms such as Facebook could not refrain from these cyberattacks and put the data of about 50 million users at risk. These cyberattacks alarms the government, businesses, organizations and hence, emphasizes the significance of cybersecurity then and now.

Existing Products

With the increasing cyberattacks, the online security industry has grown more sophisticatedly and continues to grow. Several toolkits have been designed over the years to identify complex vulnerabilities.

Core Impact:

This is a centralized testing tool that detects flaws and vulnerabilities in the system. Core Impact claims to have the greatest number of exploits present in the market and does a thorough audit trail containing PowerShell commands, and by replaying the audit trail can automatically retest a client. Commercial Grade exploit ensures quality and offers the technical support to both the exploits and their platforms (Software Testing, 2022).

Nessus:

Nessus is an open-source vulnerability scanner. Its product range includes Nessus Cloud, Nessus Professional, Nessus Manager, and Nessus Home. Malware detection, vulnerability assessment, and auditing of the control system are the main services offered by Nessus. Although it does not provide

penetration testing, the Nessus scanning engine employs plug-ins that are updated regularly to find new vulnerabilities (Tittel, 2022).

APT2

APT2 performs an Nmap scan and integrates results that are imported from the scan. These results are then processed to carry out the attack on the targeted environment. The results are stored on the localhost and becomes a fragment of the knowledge base (KB). In addition, it performs tasks such as detecting SSL protocols and ciphers, getting a list of users from SMB, Nmap VNC brute scan, attempting to authenticate using PSEXEC PTH, attempting to exploit MS08-067, and generating HTML reports.

Acunetix

Acunetix supports HTML5, JavaScript, and CMS systems and is used to identify and generate reports for over 4000 vulnerabilities. It also includes manual tools for penetration testers to integrate with the most popular Issue Trackers and WAFs. They employ modern techniques such as DeepScan. AcuSensor combines black box scanning techniques with sensor feedback included in the source code to create AcuSensor. It's a parental control system based on software. Based on the configurations, it is used to restrict and filter internet traffic and can also prevent PC games from being played.

These products might be beneficial to users sometimes they can be very expensive and difficult to operate for an untrained user.

Security Scanner

Security scanner Toolkit is a smart scanning toolkit with an automated architecture that can actively scan a wireless network for network topology and vulnerabilities using various exploitation techniques. These problems can be caused by misconfigurations such as software faults, missing updates, malware infections, and so on. **This security scanner** conducts computer network scanning using the same techniques as hackers, but with permission from the network owner. This project assists in identifying security vulnerabilities and mapping vulnerabilities to known CVEs and vulnerabilities from other online databases. Existing open-source solutions are manual and command-line based

whereas propriety solutions are very expensive. Security Scanner is a low-cost solution to perform a vulnerability assessment of the devices/systems connected to the network to minimize cyber-attack risk. It aims to provide security to networks, integrity, and confidentiality of data.

This product is not only for security professionals but also for non-technical individuals who want to check the security of their system either at the office or at home because personal laptops may contain details and passwords of the various account and a compromised laptop would result in theft of this confidential information. Also, this scanner can be a useful tool for educational purposes as it just does not perform one scan. The student will be able to scan without correlating the scan to vulnerabilities so that they can learn how to read a simple scan. This scanner has an easy-to-use interface and any individual can run it on their system if the system fulfills the installation requirements.

Requirements

There are three fundamental requirements for a security scanner. The hardware and software requirements, user knowledge, and lastly, the functional requirement.

Practical knowledge of installing scanners is a prerequisite for its users. Users must have sound information on how to install software on their respective devices and have proper credentials to install the software. The requirements vary from user to user for example, users in a corporate environment may not have the permission to install software on their devices unless they have the approval of their employer.

Developer Software & Hardware Requirements

The hardware component required for the implementation of this project requires a PC/Laptop which has at least a core i5 having 8GB RAM and 80 GB hard drive to smoothly run the high processing scanning.

The software requirements are discussed below:

Pycharm IDE professional version is used for coding in the python language.

Python

The security scanner makes use of python to automate tools that are integrated under a user-intuitive Web Application which is developed using React js. The following are the main python libraries used.

Libraries:

The Subprocess module of python is used for accessing system commands. This module permits to initiation of new processes, connecting to their input/output/error pipes, and acquiring their return codes.

Python-Nmap is a python library that aids in making use of the Nmap port scanner. It facilitates easy manipulation of scan results from Nmap and is a suitable tool to automate scan tasks and scan reports. It also supports Nmap script outputs.

Flask is used for creating API. Flask is a widely used micro web framework for creating APIs in Python. It is a simple yet powerful web framework which is designed to get started quick and easy, with the ability to scale up to complex applications.

JSON is a python library which is used to work with JSON data. In this project, it is used to send the results to the frontend in json format and store the results in json file to view it later.

React JS

React JS is used to make user friendly, easy-to-use interface. React js has a component-based architecture. This means that the components can be re-used. It can change data without rendering the page. It is fast, scalable, simple to use and can be integrated with other libraries. All these things make react a great choice for making a good web application. With it, we used java script to render the result that comes from the python API server to display it.

Node js version 16.14.2

User Software & Hardware Requirements

To run the software, the user and developer should have Linux installed as the OS. The user and developer can also work with a virtual machine. Users will need an internet connection. Users will need to see an installation guide on how to install and run the software. Users will also need to understand the concept of vulnerabilities and CVEs, how a certain CVE represents a vulnerability and what a user can do to remediate the vulnerability as the software reports just vulnerabilities. As a list:

- A Linux computer or a computer with Windows 8 or higher as the OS with a Linux VM installed in it.
- Stable Internet connection
- Read installation guide on how to install and run
- Basic concept of Linux OS, security, vulnerabilities and how to prevent them.

Functional Requirements

The security scanner provides several options to work with depending on the nature of the task. The main goal of the product is to not only offer some security product to a pen tester or security engineers but also individuals who have no experience in security. So, they can educate themselves about security principles, correct their security practices and adopt preventive measures to prevent and save themselves from cybercrime.

The users can do the following things:

- Users can perform a network scan and see how many devices are connected to their network.
- Users can see IP addresses, MAC Addresses, device names and vendors of the network interface cards of the devices connected to the network.
- Users can ping a custom IP or IP from the network scan and check its availability, whether the host is up or unreachable.
- Users can ping a range of IPs in one go to check their availability.
- Users can scan ports of the system to check if there are any open ports.
- For the extensive scan, users can see the application and version of the application running on open ports.
- Users can see the Operating System running on the target machine.
- User can map port scanning results against known vulnerabilities.
- User can view a task list of previous scans. Table will show Id, IP, type of scan, and time when scan was run.
- User can view the results of previous scans.

Design

The following section contains information about the detailed design of each module. Each module has a flow chart explaining the background functions. There will also be a written explanation before each chart. At the end of this section, there will be screens of the interface for each component.

Software Modules

Network Scan:

To find IP addresses and other information of all the devices connected to the network, the scanner will perform multiple scans to find all the information. First, the scanner will perform an ARP scan. Arp-scan is a command-line tool that uses the ARP protocol to discover and fingerprint IP hosts on the local network. Arp-scan provides all the IPs present in the Arp table mapping IPs to MAC addresses. Then the scanner will perform an Nmap scan to verify arp-scan results and find out the hostname of devices and the manufacturers. After performing all the scans, the results of the scan will be sorted to show only specific results on the web application. The results are presented in a tabular form on the screen of the application.

Ping Scan and Sweep:

Ping works by sending an Internet Control Message Protocol (ICMP) Echo Request to a specified IP on the network and, in return, sends out replies to validate the connection. The ping response is the most important thing to keep an eye on when performing a ping test. It tells us whether the host is up or not, the host is up but isn't reachable. Destination unreachable can have different causes. Destination unreachable uses several code values to further describe the cause.

- 0 = Network Unreachable
- 1 = Host Unreachable
- 2 = Protocol Unreachable
- 3 = Port Unreachable
- 5 = Source Route Failed

- 6 = Destination Network Unknown
- 7 = Destination Host Unknown
- 9 = Destination Network Administratively Prohibited
- 10 = Destination Host Administratively Prohibited

From these codes in the reply packet, we are able to see what the reason is. The results are displayed on the screen of the user interface.

Ping sweep scans the entire IP subnet provided by the user to check how many hosts are alive. Ping sweep can be done using ICMP packets, TCP, or UDP packets. Ping sweep communicates with multiple hosts at the same time. Ping sweep is done using ping. It checks and compares the results for accuracy. The result of this function is returned to the function of the interface. The web application populates the range of IPs mapped to live hosts in a table.

Quick Port Scan:

Port scanning is a method of determining which ports on a network are open and could be receiving or sending data. It is also a process for sending packets to specific ports on a host and analyzing responses to identify vulnerabilities. The scanner will perform a TCP SYN scan to quickly get the result of open ports, the protocol used, and what service is running on that port. SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. It provides a clear, reliable differentiation between the open, closed, and filtered states. This technique is often referred to as half-open scanning because you don't open a full TCP connection. You send a SYN packet as if you are going to open a real connection and then wait for a response. An SYN/ACK indicates the port is listening (open), while an RST (reset) is indicative of a non-listener. If no response is received after several retransmissions, the port is marked as filtered. The result of this function is returned to the function of the interface of the specified page. The web app will display the list of open ports and respective protocols if any ports are open. Otherwise, it will display that all ports are closed or filtered.

Extensive Port Scan:

In an extensive port scan, the scanner is thoroughly scanning the target IP. In addition, to open ports, it scans for service and which version that service is using, potential OS used by the target. The result can vary due to the use of a firewall in OS detection. At first, it's going to check if the host is up or not, basically a host discovery. Then, the scanner will scan for ports. After TCP and/or UDP ports are discovered using one of the other scan methods, version detection interrogates those ports to determine more about what is actually running. The nmap-service-probes database contains probes for querying various services and match expressions to recognize and parse responses.

The display will list all the results for open ports, protocol service, application, version, and OS family.

Correlating scan results and vulnerabilities:

The Security Scanner correlates the scanned results to get the better understanding of the open ports, services, application name and version found by scan. It checks whether a known vulnerability is present in the system by mapping it to different vulnerability databases. Then the user can search for patches for such vulnerabilities. Or as security official or student, to gain experience in exploiting the system, users can use this scan result to exploit the system. Users can search Metasploit exploits library to find a specific module to use. For finding CVE, NSE scripts are used. Vulscan and vulners scripts are used for this purpose. These scripts match results using their online database or our own database can be used. The database needs to be updated with time as more vulnerabilities will be added to the vulnerabilities list.

The Web app displays the ports and application name with version, vulnerabilities, the, CVE number and vulnerability score.

Security Scanner

```
graph TD; A[Security Scanner] --> B[Network Scan]; A --> C[Quick Port Scan]; A --> D[Ping Scan and Sweep]; A --> E[Extensive Port Scan]; A --> F[Correlating scan results and CVEs];
```

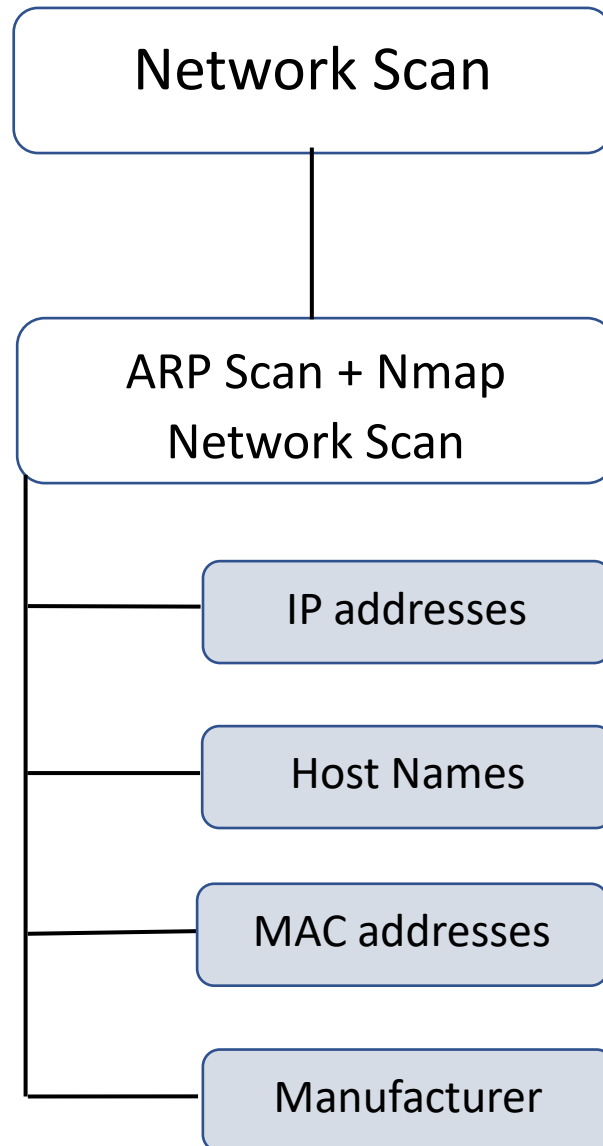
Network Scan

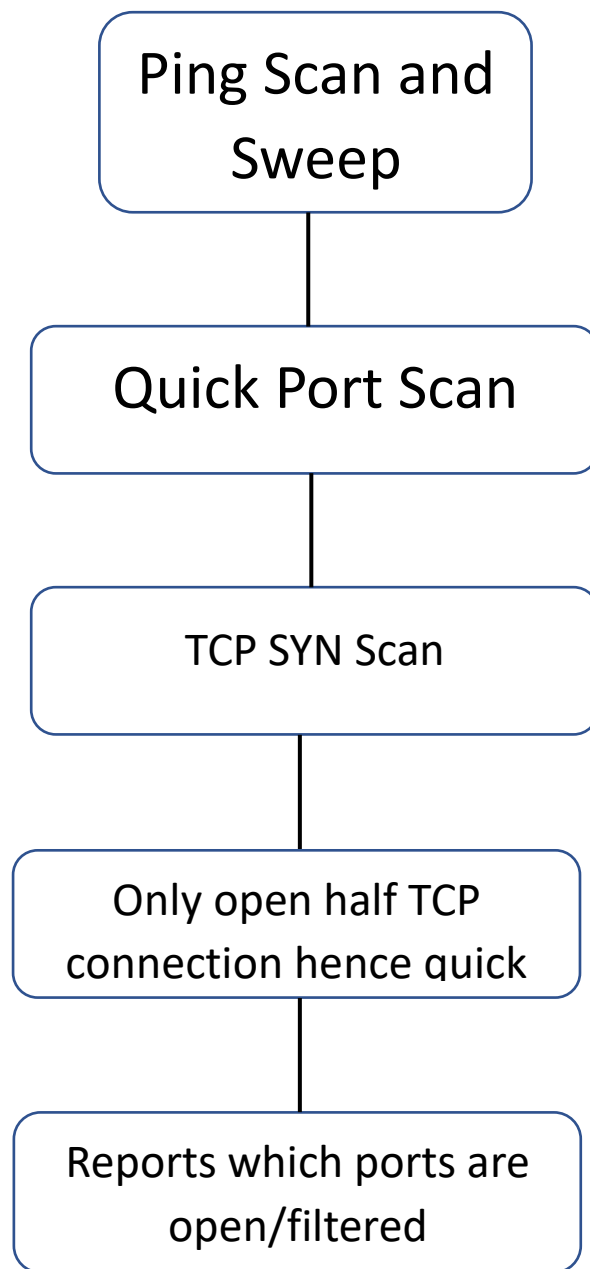
Quick Port Scan

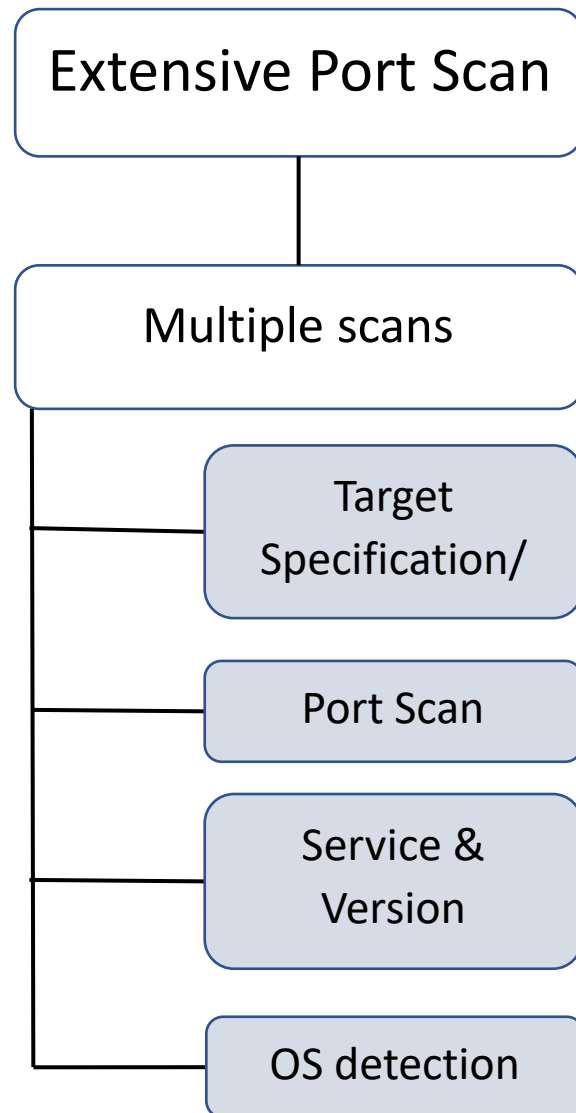
Ping Scan and
Sweep

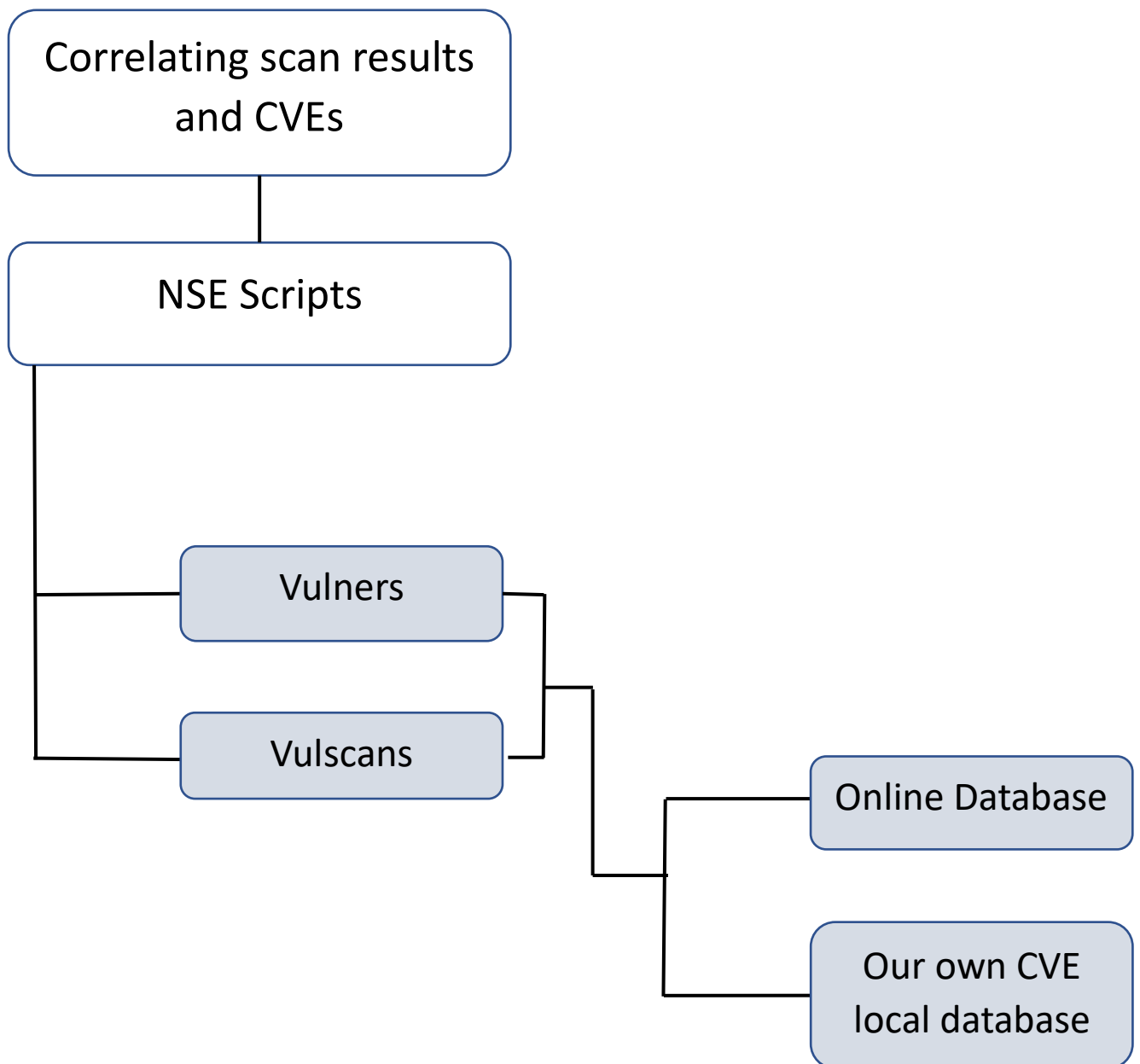
Extensive Port Scan

Correlating scan
results and CVEs









Interface Design

The following section contains screens depicting the interface for each component. The main screen will display title and services offered by the security scanner.

The project has 7 pages, multiple subpages, and tabs. The first is the home page that presents what a security scanner is and what services the project provides. Then the next page shows each module per page. There's also a screen for the task list module. A collapsible side navigation bar that contains the name of each module is included to enable the user navigate through the module easily.

Home screen:

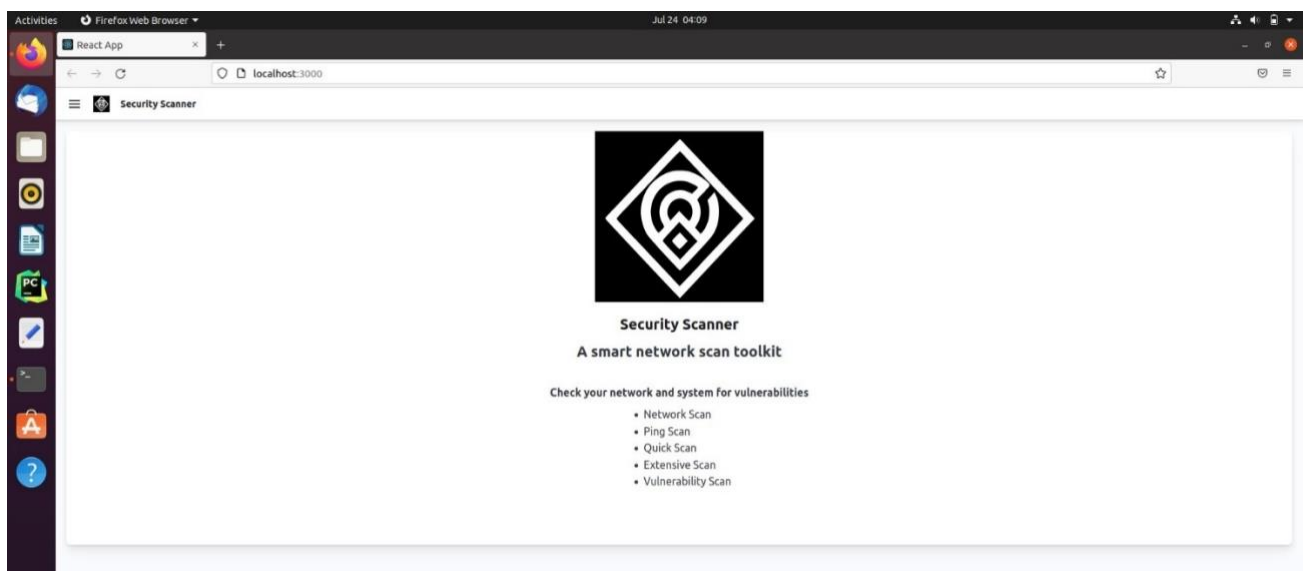


Figure 1 Home Page

Network Scan Page:

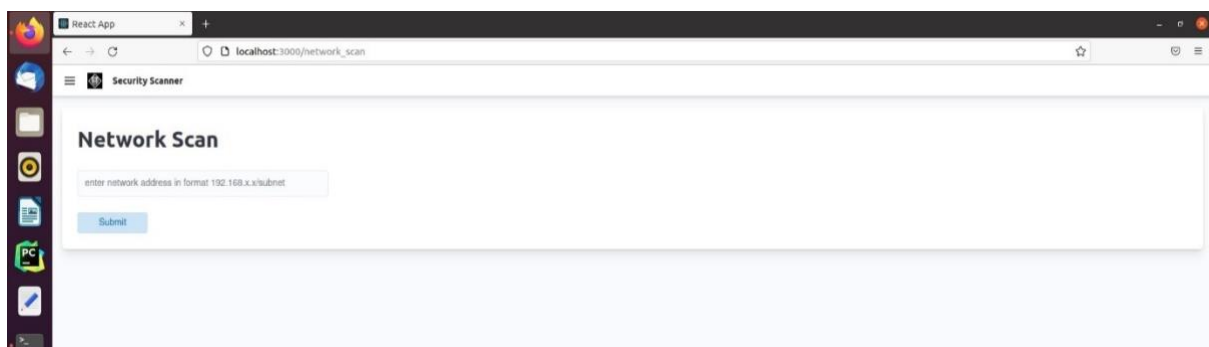
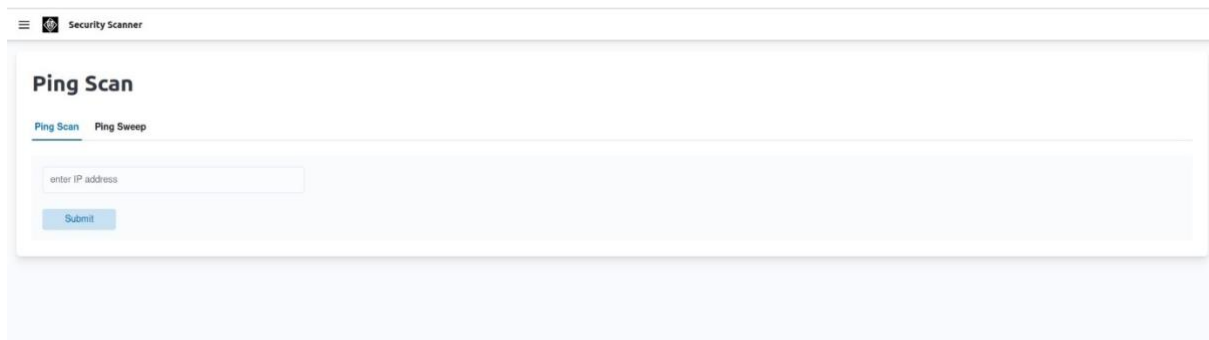


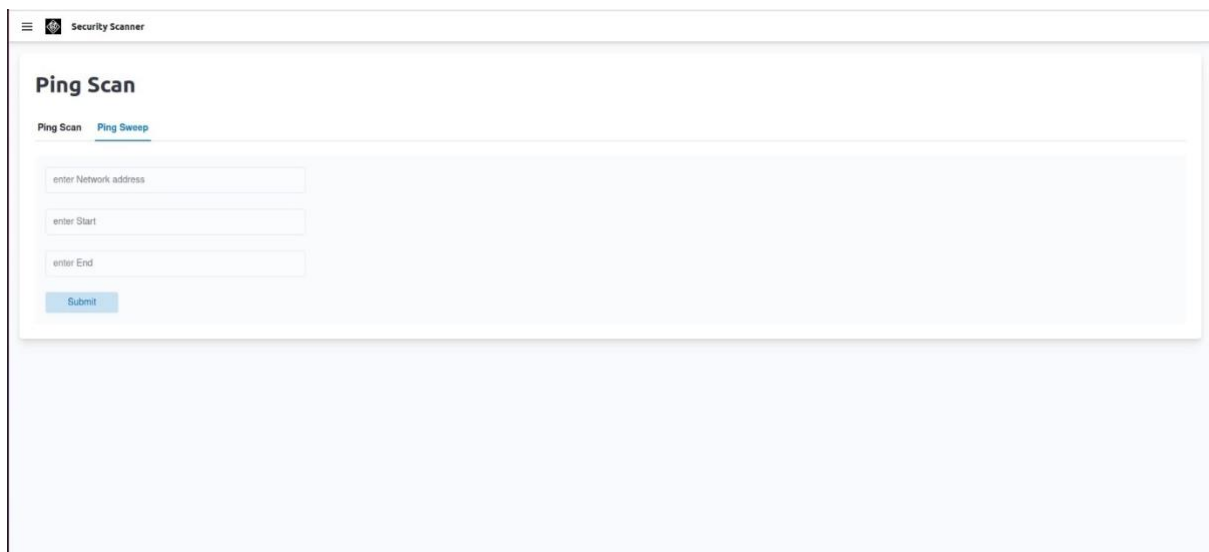
Figure 2 Network scan

Ping scan and sweep page:



The screenshot shows the 'Ping Scan' page of the 'Security Scanner' application. The page has a header with a hamburger menu icon and the text 'Security Scanner'. Below the header, the title 'Ping Scan' is displayed. There are two tabs: 'Ping Scan' (active) and 'Ping Sweep'. The main form area contains a single text input field labeled 'enter IP address' and a blue 'Submit' button below it.

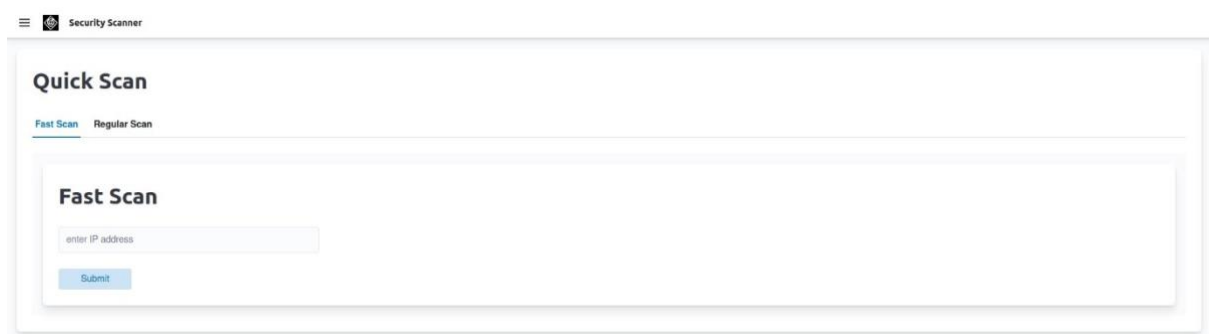
Figure 3 ping scan



The screenshot shows the 'Ping Sweep' page of the 'Security Scanner' application. The page has a header with a hamburger menu icon and the text 'Security Scanner'. Below the header, the title 'Ping Scan' is displayed. There are two tabs: 'Ping Scan' and 'Ping Sweep' (active). The main form area contains three text input fields: 'enter Network address', 'enter Start', and 'enter End', followed by a blue 'Submit' button.

Figure 4 ping sweep

Quick scan (Fast and Regular scan) page:



The screenshot shows the 'Quick Scan' page of the 'Security Scanner' application. The page has a header with a hamburger menu icon and the text 'Security Scanner'. Below the header, the title 'Quick Scan' is displayed. There are two tabs: 'Fast Scan' (active) and 'Regular Scan'. The main form area contains a text input field labeled 'enter IP address' and a blue 'Submit' button below it.

Figure 5 Fast scan

The screenshot shows the 'Security Scanner' application interface. At the top, there is a navigation bar with a hamburger menu icon and the text 'Security Scanner'. Below this, the main heading is 'Quick Scan'. Underneath, there are two tabs: 'Fast Scan' and 'Regular Scan', with 'Regular Scan' being the active tab. The 'Regular Scan' section contains a text input field with the placeholder text 'enter IP address' and a blue 'Submit' button below it.

Figure 6 Regular scan

Extensive Scan Page:

The screenshot shows the 'Security Scanner' application interface for the 'Extensive Scan' page. It features a navigation bar with a hamburger menu icon and the text 'Security Scanner'. The main heading is 'Extensive Scan'. Below the heading, there is a text input field with the placeholder text 'enter IP address' and a blue 'Submit' button positioned directly below it.

Figure 7 Extensive scan

Vulnerability scan page:

The screenshot shows the 'Security Scanner' application interface for the 'Vulnerability Scan' page. It includes a navigation bar with a hamburger menu icon and the text 'Security Scanner'. The main heading is 'Vulnerability Scan'. Below the heading, there is a text input field with the placeholder text 'enter IP address' and a blue 'Submit' button located below the input field.

Figure 8 Vulnerability Scan

Installation

The following section contains information about how to install the software for testing and running. There are two methods to run the security scanner on your system.

Method 1

For installing the scanner manually, follow these steps:

1. Install oracle virtual box
2. Create a virtual machine with the following specification:
 - 4 GB RAM
 - 80 GB memory
 - Atleast 1 CPU core
3. Download ubuntu iso file
4. Install ubuntu in the virtual machine.
5. Open ubuntu terminal and run following commands:
 - `sudo apt update`
 - `sudo apt install net-tools`
 - `sudo apt install Nmap`
 - `sudo apt install python3`
 - `sudo apt install make`
 - `sudo apt install build-essentials`
 - `sudo apt install python3-pip`
 - `sudo apt install git`
 - `sudo apt install curl`
6. Install following python packages:
 - `pip install python-nmap`
 - `pip install json`
 - `pip install flask`
 - `pip install flask-cors`
 - `pip install mac-vendor-lookup`
 - `pip install getmac`
7. Then for install node js version 16.14.2 by following commands:
 - `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/master/install.sh | bash`
 - `source ~/.bashrc`
 - `nvm --version`
 - `nvm install v16.14.2`
8. Install yarn and create-react-app by following commands:
 - `npm install -g yarn`
 - `npm install -g create-react-app`
9. Now, download the given code for scanner. There will be two folders.
 - Sentinel
 - Scanner
10. Sentinel folder contains code for front-end while scanner folder contains code for backend api
11. To run backend api, open scanner folder in terminal and run command: `sudo python3 main.py`
12. To run UI, open sentinel folder in terminal and run command: `yarn start`.
13. A tab in firefox will open and user can now use the scanner to perform different scans.
14. If the user wants to build the production bundle, it can be done with the following command:
`yarn run build`.

Method 2

If the user wants to use the scanner without all the steps mentioned above, they can do following:

1. Install oracle virtual box
2. Import ubuntu VM which will be provided
3. Open ubuntu VM. It is preconfigured.
4. To run backend api, open scanner folder in terminal and run command: `sudo python3 main.py`
5. To run UI, open sentinel folder in terminal and run command: `yarn start`
6. A tab in firefox will open and user can now use the scanner to perform different scans.

Sample Sessions

The following section contains screenshots and information about every component / module and some scan results with details.

Home Page

This is the home page. There is a collapsible navigation bar on the top left to access the menu.

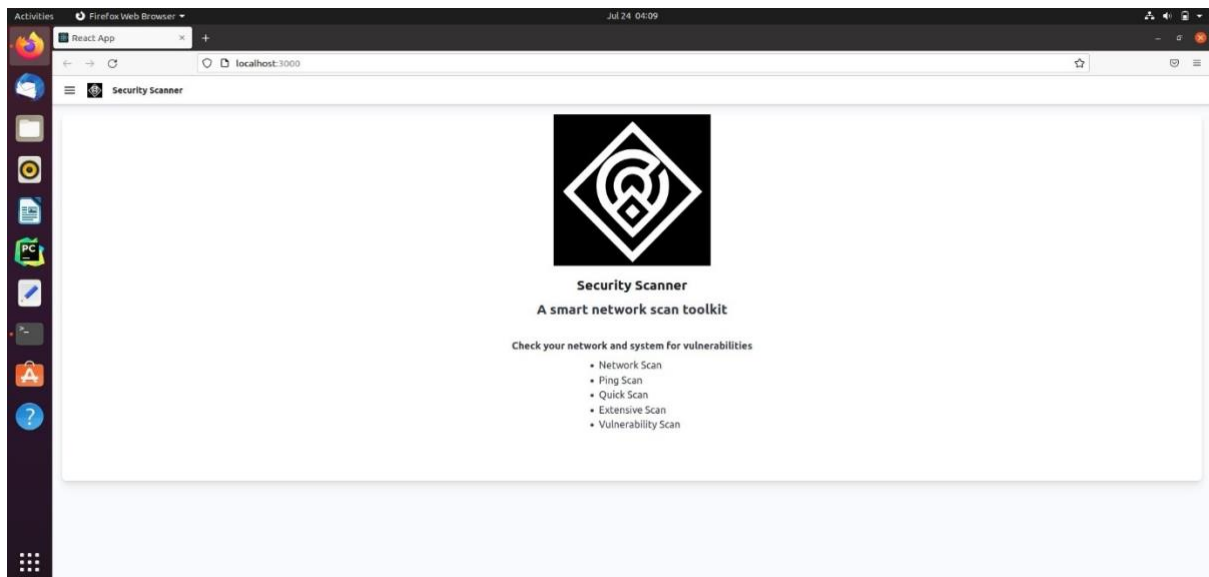


Figure 9 Home front page

Network scan results

In the network scan, you can see the IP address, host name, mac address, mac vendor of all the users connected to the internet.

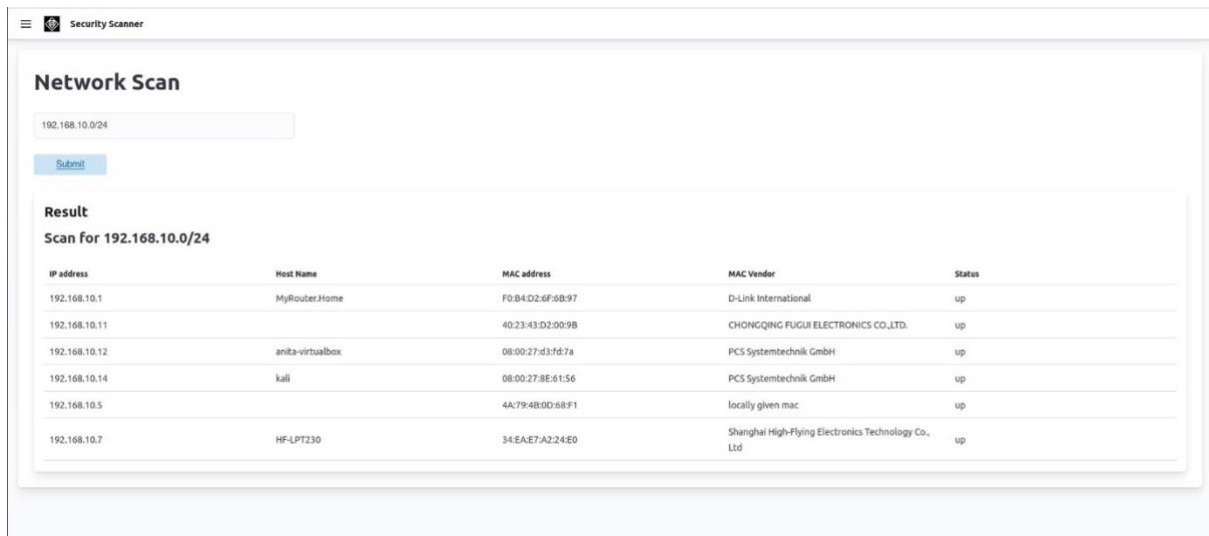


Figure 10 network scan result

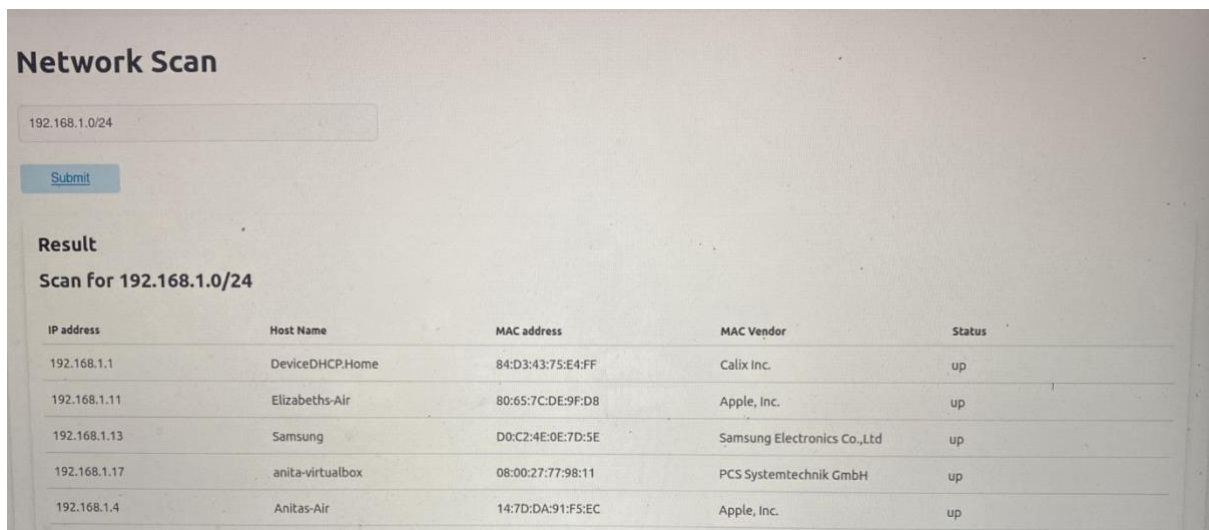


Figure 11 network scan result

Ping scan results

The below image shows when a host is not responding:

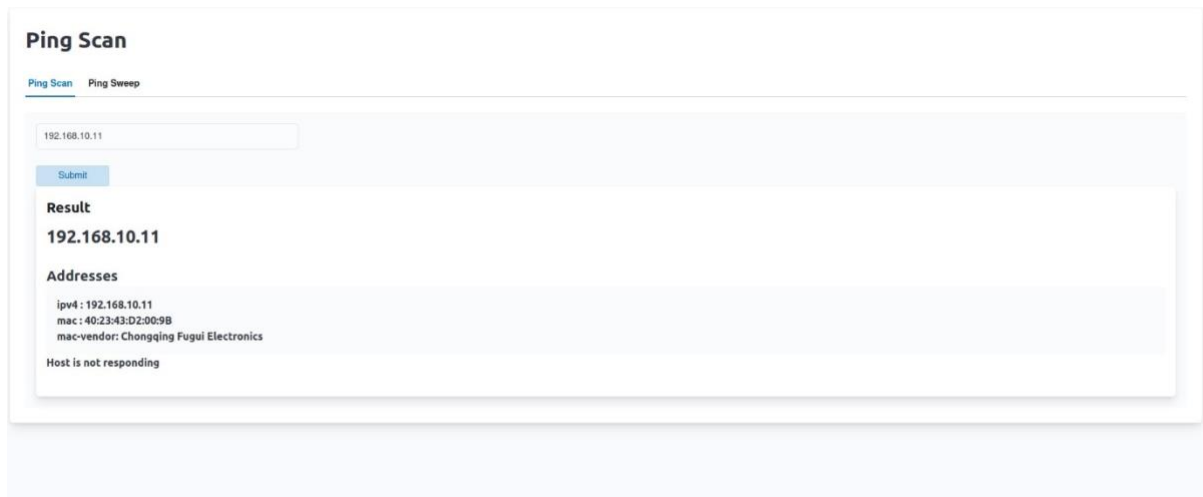


Figure 12 ping scan result

The below image shows when the host is up and responding:

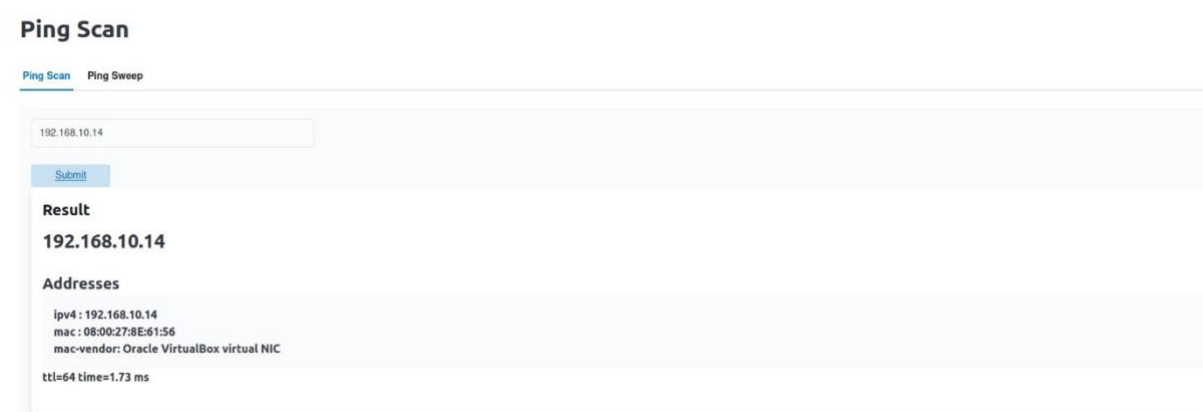


Figure 13 ping scan result

Ping Sweep Results

Ping sweep tells us if a host is not responding, if it is up or when a destination IP is unreachable because it is not assigned to any device.

Ping Scan	
Ping Sweep	
192.168.10.0	
1	
20	
Submit	
Result	
IP Address	Response
192.168.10.1	Host is up : ttl=64 time=11.5 ms
192.168.10.2	Destination Host Unreachable
192.168.10.3	Destination Host Unreachable
192.168.10.4	Host is not responding
192.168.10.5	Host is up : ttl=64 time=353 ms
192.168.10.6	Destination Host Unreachable
192.168.10.7	Host is up : ttl=255 time=212 ms
192.168.10.8	Destination Host Unreachable
192.168.10.9	Host is not responding
192.168.10.10	Destination Host Unreachable

Figure 14 ping sweep result

Fast Scan Result

Fast scan shows open port and name of services.

Quick Scan	
Fast Scan	
Regular Scan	
192.168.10.11	
Submit	
Result	
192.168.10.11	
<div> <div>Address</div> <div> ipv4 : 192.168.10.11 mac : 40:23:43:02:00:9B </div> </div>	
Port	State Reason Name Conf
5357	open syn-ack wsdapi 3

Figure 15 Fast scan result

Regular Scan Result

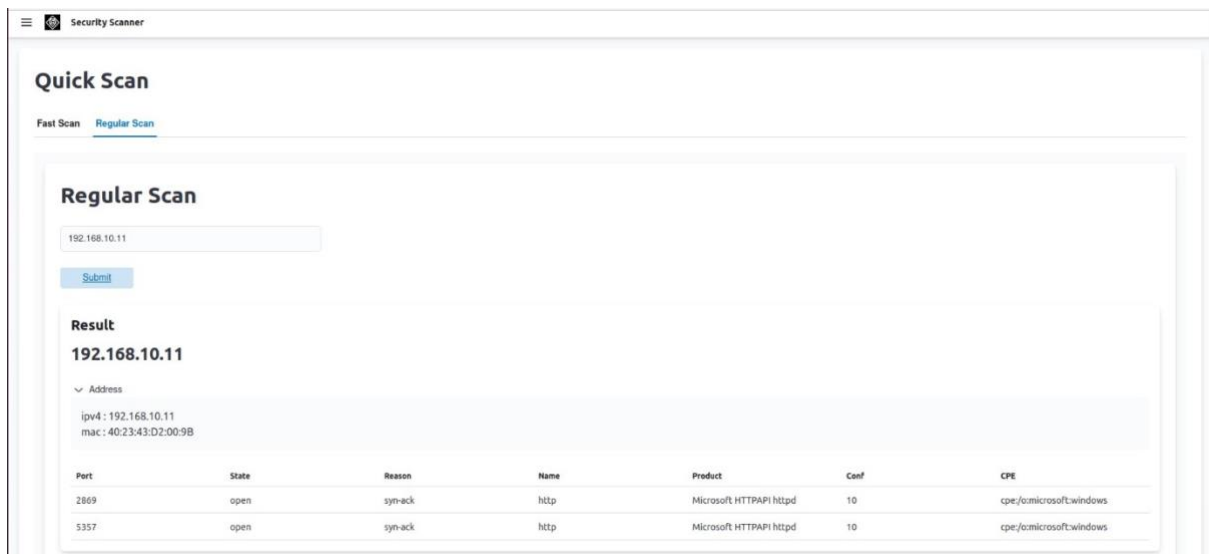


Figure 16 regular scan result (open ports)

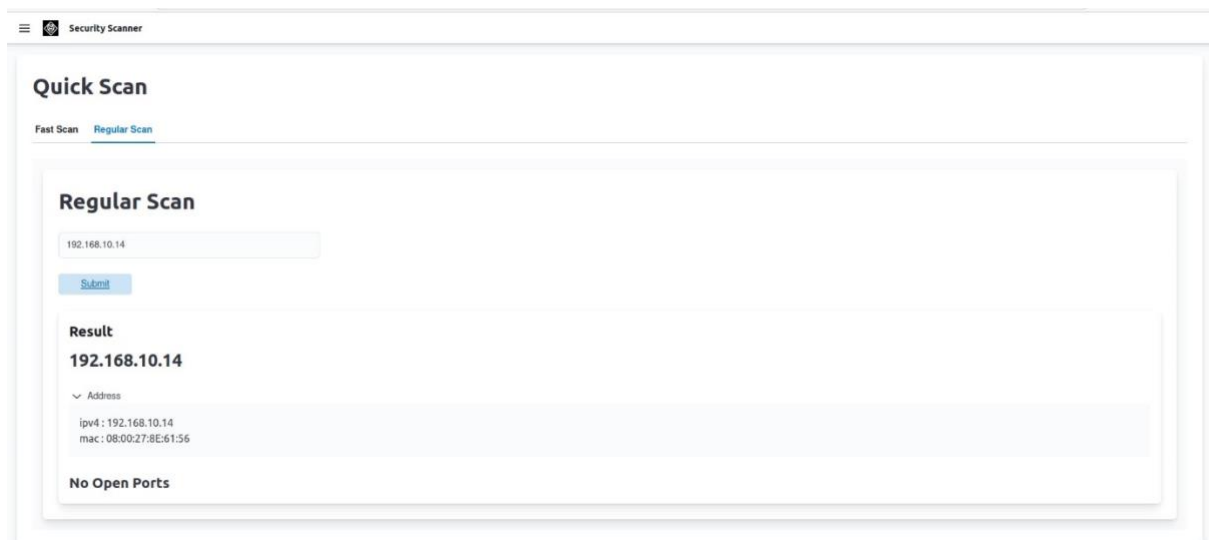


Figure 17 regular scan result (no open ports)

Extensive Scan Results

Security Scanner

Extensive Scan

192.168.10.11

Submit

Result

192.168.10.11

Address

ipv4 : 192.168.10.11
 mac : 40:23:43:D2:00:9B
 mac-vendor: Chongqing Fugui Electronics

Port Information

Port	State	Reason	Name	Product	Version	Extra Information	Conf	CPE	HTTP Title	HTTP Server Header
2869	open	syn-ack	http	Microsoft HTTPAPI httpd	2.0	SSDP/UPnP	10	cpe:/o:microsoft:windows	Service Unavailable	Microsoft-HTTPAPI/2.0
5357	open	syn-ack	http	Microsoft HTTPAPI httpd	2.0	SSDP/UPnP	10	cpe:/o:microsoft:windows	Service Unavailable	Microsoft-HTTPAPI/2.0

OS Information

Figure 18 extensive scan result (port)

Security Scanner

Vulnerability Scan

192.168.10.14

Submit

Result

192.168.10.14

Address

ipv4 : 192.168.10.14
 mac : 08:00:27:8E:61:56
 mac-vendor: Oracle VirtualBox virtual NIC

OS Information

These results may not be 100% accurate

Name	Accuracy	Vendor	Os Family
AVtech Room Alert 26W environmental monitor	87	AVtech	embedded
Microsoft Windows XP SP2	87	Microsoft	Windows
FreeBSD 6.2-RELEASE	86	FreeBSD	FreeBSD
FreeBSD 10.3-STABLE	85	FreeBSD	FreeBSD

Figure 19 extensive scan result (os info)

Vulnerability Scan Results

Security Scanner

Vulnerability Scan

192.168.10.14

Submit

Result

192.168.10.14

Address

ipv4 : 192.168.10.14
 mac : 08:00:27:8E:61:56
 mac-vendor: Oracle VirtualBox virtual NIC

OS Information

These results may not be 100% accurate

Name	Accuracy	Vendor	Os Family
Linux 2.6.32	96	Linux	Linux
Linux 3.2 - 4.9	96	Linux	Linux
Linux 2.6.32 - 3.10	96	Linux	Linux
Linux 3.4 - 3.10	95	Linux	Linux

Figure 20 vulnerability scan result

These results may not be 100% accurate

Open Ports Information

Vulnerabilites

Vulnerabilites

Port number: 80

Figure 22 vulnerability scan result

80

Figure 23 vulnerability scan result

Vulners Result		
Port number: 8000		
CVE number	Level	Link
NODEJS:744	5.0	https://vulners.com/nodejs/NODEJS:744
NODEJS:585	3.5	https://vulners.com/nodejs/NODEJS:585
Vulscan		
✓ 8000 VulDB - https://vuldb.com : [119198] simplehttpserver on Node.js cross site scripting [5572] Python up to 2.7.2 SimpleHTTPServer Module SimpleHTTPServer.py list_directory cross site scripting MITRE CVE - https://cve.mitre.org : [CVE-2011-4940] The list_directory function in Lib/SimpleHTTPServer.py in SimpleHTTPServer in Python before 2.5.6c1, 2.6.x before 2.6.7 rc2, and 2.7.x before 2.7.2 does not place a charset parameter in the Content-Type HTTP header, which makes it easier for remote attackers to conduct cross-site scripting (XSS) attacks against Internet Explorer 7 via UTF-7 encoding. SecurityFocus - https://www.securityfocus.com/bid/ : [54083] Python SimpleHTTPServer 'list_directory()' Function Cross Site Scripting Vulnerability IBM X-Force - https://exchange.xforce.ibmcloud.com : [76525] Python SimpleHTTPServer list_directory() cross-site scripting Exploit-DB - https://www.exploit-db.com : No findings OpenVAS (Nessus) - http://www.openvas.org : No findings SecurityTracker - https://www.securitytracker.com : No findings OSVDB - http://www.osvdb.org : [83057] Python SimpleHTTPServer Module Crafted Filename Upload Directory Listing XSS		

Figure 24 vulnerability scan result

Task List

Task List			
ID	IP address	Scan Type	Time
1	192.168.10.14	vulnerability scan	12/07/2022, 04:41:24
2	192.168.10.14	vulnerability scan	12/07/2022, 04:51:20
3	192.168.10.14	vulnerability scan	12/07/2022, 04:52:50
4	192.168.10.14	vulnerability scan	12/07/2022, 04:56:28
5	192.168.10.14	vulnerability scan	12/07/2022, 04:58:46
6	192.168.10.11	vulnerability scan	12/07/2022, 05:01:28
7	192.168.10.11	vulnerability scan	12/07/2022, 05:03:28
8	192.168.10.14	vulnerability scan	12/07/2022, 05:06:08
9	192.168.10.14	vulnerability scan	12/07/2022, 05:11:13
10	192.168.10.14	vulnerability scan	12/07/2022, 05:12:26

Rows per page: 10

< 1 2 3 4 5 6 7 >

Figure 25 task list

Security Scanner

Task List

ID	IP address	Scan Type	Time
52	192.168.10.4	extensive scan	21/07/2022, 11:18:53
53	192.168.10.4	extensive scan	21/07/2022, 11:19:19
54	192.168.10.4	extensive scan	21/07/2022, 11:20:14
55	192.168.10.0/24	network scan	24/07/2022, 04:10:18
56	192.168.10.11	ping scan	24/07/2022, 04:11:55
57	192.168.10.14	ping scan	24/07/2022, 04:12:46
58	192.168.10.1-21	ping sweep	24/07/2022, 04:13:14
59	192.168.10.11	fast scan	24/07/2022, 04:16:17
60	192.168.10.11	regular scan	24/07/2022, 04:17:17
61	192.168.10.14	regular scan	24/07/2022, 04:18:18

Rows per page: 10

< 1 2 3 4 5 6 7 >

Figure 26 task list

Security Scanner

Task List

ID	IP address	Scan Type	Time
62	192.168.10.11	extensive scan	24/07/2022, 04:19:11
63	192.168.10.14	vulnerability scan	24/07/2022, 04:21:03

Rows per page: 10

< 1 2 3 4 5 6 7 >

Figure 27 task list

References

- Equifax. (2022). *How Cyber Attacks Happen*. From <https://www.equifax.co.uk/resources/identity-protection/how-cyber-attacks-happen.html> Accessed on: April, 7 2022.
- Forte, e. (n.d.). *CORE IMPACT - PENETRATION TESTING TOOL*. From <https://www.esecforte.com/products/core-impact/> Accessed on: April, 12 2022
- Forsey, C. (2020). The Ultimate List of Email Marketing Stats for 2020. Accessed on: April, 6 2022.
- Hancock, M. (2021). The Influence of Cybersecurity on Modern Society (No. 5882). EasyChair.
- Hook, I. (2019, March 26). *How the Melissa Virus Changed the Internet*. From <https://www.insidehook.com/article/history/melissa-virus-changed-internet> Accessed on: May 17, 2022.
- Shahani, A. (2016, September 22). *Yahoo Confirms Massive Data Breach By 'State-Sponsored Actor'*. From <https://www.npr.org/2016/09/22/495069534/yahoo-confirms-massive-data-breach-by-state-sponsored-actor> Accessed on: May 17, 2022.
- Uhde, A. (2017, July 5). *A short history of computer viruses*. From <https://www.sentrion.com.au/blog/a-short-history-of-computer-viruses> Accessed on: May 17, 2022.
- VPN Mentor (2022). The 20 Biggest Hacking Attacks of All Time <https://www.vpnmentor.com/blog/20-biggest-hacking-attacks-time/> Accessed on: April, 6 2022.

Appendix: Source Code

Python API source code:

```
#####  
#Project name: Security Scanner  
#File: main.py // api code  
#Developer: Anita  
#####  
from datetime import datetime  
import nmap  
import os  
from flask import Flask, request  
from flask_cors import CORS  
import json  
import socket  
from mac_vendor_lookup import MacLookup, VendorNotFoundError  
from getmac import get_mac_address as gma  
  
app = Flask(__name__)  
CORS(app)  
  
id= 0  
#Network Scanning  
@app.route('/network_scan', methods=['GET', 'POST'])  
def network_scan():  
    if request.method == 'GET':  
        ip = request.args.get("ip")  
    elif request.method == 'POST':  
        ip = request.form.get("ip")  
    data = [json.loads(line) for line in open('tasks.json', 'r+')]  
    if len(data) == 0:  
        id = 1
```

```

else:
    task_len = len(data[0])
    last_id = data[0][task_len - 1]["id"]
    id = last_id + 1
new_task = {
    "id": id,
    "ip": ip,
    "type": "network scan",
    "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
}
new_data = []
if len(data) == 0:
    new_data.append(new_task)
else:
    new_data = data[0]
    new_data.append(new_task)
with open('tasks.json', 'w') as file_json:
    json.dump(new_data, file_json)
file_json.close()

```

```

nm = nmap.PortScanner()
target = nm.scan(hosts=ip, arguments="-sP")
response = target['scan']
keys = response.keys()
last = list(response.keys())[-1]
response[last]['addresses']['mac'] = gma()
for key in response:
    mac = response[key]['addresses']['mac']
    try:
        vendor = MacLookup().lookup(mac)
        response[key]['vendor'] = vendor
    except VendorNotFoundError:

```

```

        response[key]['vendor'] = "locally given mac"
    with open(f"{id}.json", "w") as result:
        json.dump(response, result)
    result.close()
    return (response)

```

#Ping Sweep

```

@app.route('/ping_sweep', methods=['GET', 'POST'])
def ping_sweep():
    if request.method == 'GET':
        net = request.args.get("net")
        st1 = request.args.get("st1")
        en1 = request.args.get("en1")
    elif request.method == 'POST':
        net = request.form.get("net")
        st1 = request.form.get("st1")
        en1 = request.form.get("en1")
    net1 = net.split(".")
    a = '.'
    net2 = net1[0] + a + net1[1] + a + net1[2] + a
    en1 = int(en1) + 1
    ping1 = 'ping -c 1 '
    result = []
    ip = net2 + st1 + "-" + str(en1)
    data = [json.loads(line) for line in open('tasks.json', 'r+')]
    if len(data) == 0:
        id = 1
    else:
        task_len = len(data[0])
        last_id = data[0][task_len - 1]["id"]
        id = last_id + 1
    new_task = {

```

```

        "id": id,
        "ip": ip,
        "type": "ping sweep",
        "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
    }
    new_data = []
    if len(data) == 0:
        new_data.append(new_task)
    else:
        new_data = data[0]
        new_data.append(new_task)
    with open('tasks.json', 'w') as file_json:
        json.dump(new_data, file_json)
    file_json.close()

    for ip in range(int(st1), en1):
        addr = net2 + str(ip)
        comm = ping1 + addr
        response = os.popen(comm)
        res = response.readlines()
        if (res[1] != "\n"):
            res1 = res[1].split("icmp_seq=1")
            message = res1[1]
            if (message != " Destination Host Unreachable\n"):
                message = "Host is up : " + message
        elif (res[1] == "\n"):
            message = "Host is not responding"
        result.append({
            "address": addr,
            "response": message
        })
    with open(f"{id}.json", "w") as res:

```

```

        json.dump(result, res)
    res.close()
    result = json.dumps(result)
    return (result)

```

#Ping Scan

```
@app.route('/ping_scan', methods=['GET', 'POST'])
```

```
def ping_scan():
```

```
    if request.method == 'GET':
```

```
        ip = request.args.get("ip")
```

```
    elif request.method == 'POST':
```

```
        ip = request.form.get("ip")
```

```
    data = [json.loads(line) for line in open('tasks.json', 'r+')]

```

```
    if len(data) == 0:
```

```
        id = 1
```

```
    else:
```

```
        task_len = len(data[0])
```

```
        last_id = data[0][task_len - 1]["id"]
```

```
        id = last_id + 1
```

```
    new_task = {
```

```
        "id": id,
```

```
        "ip": ip,
```

```
        "type": "ping scan",
```

```
        "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
```

```
    }
```

```
    new_data = []
```

```
    if len(data) == 0:
```

```
        new_data.append(new_task)
```

```
    else:
```

```
        new_data = data[0]
```

```
        new_data.append(new_task)
```

```
    with open('tasks.json', 'w') as file_json:
```

```

        json.dump(new_data, file_json)
    file_json.close()
    ping = 'ping -c 1 ' + ip
    response = os.popen(ping)
    res = response.readlines()
    if (res[1] != "\n"):
        res1 = res[1].split("icmp_seq=1")
        message = res1[1]
    elif (res[1] == "\n"):
        message = "Host is not responding"
    nm = nmap.PortScanner()
    target = nm.scan(hosts=ip, arguments="-sn")
    response = target['scan']
    key = response.keys()
    last = list(response.keys())
    response[last[0]]['message'] = message
    with open(f"{id}.json", "w") as result:
        json.dump(response, result)
    result.close()
    return (response)

```

#Fast Scan

```

@app.route('/fast_scan', methods=['GET', 'POST'])
def fast_scan():
    if request.method == 'GET':
        ip = request.args.get("ip")
    elif request.method == 'POST':
        ip = request.form.get("ip")
    data = [json.loads(line) for line in open('tasks.json', 'r+')]
    if len(data) == 0:
        id = 1
    else:

```

```

    task_len = len(data[0])

    last_id = data[0][task_len - 1]["id"]

    id = last_id + 1

new_task = {
    "id": id,
    "ip": ip,
    "type": "fast scan",
    "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
}

new_data = []

if len(data) == 0:
    new_data.append(new_task)
else:
    new_data = data[0]
    new_data.append(new_task)

with open('tasks.json', 'w') as file_json:
    json.dump(new_data, file_json)

file_json.close()


nm = nmap.PortScanner()

target = nm.scan(hosts=ip, arguments="-F")

response= target['scan']

with open(f"{id}.json", "w") as result:
    json.dump(response, result)

result.close()

return (response)


#Regular Scan

@app.route('/reg_scan', methods=['GET', 'POST'])

def regular_scan():

    if request.method == 'GET':

        ip = request.args.get("ip")

```



```

elif request.method == 'POST':
    ip = request.form.get("ip")
    print(ip)
    if ip == 'localhost':
        hostname = socket.gethostname()
        ip = socket.gethostbyname(hostname)
    data = [json.loads(line) for line in open('tasks.json', 'r+')]
    if len(data) == 0:
        id = 1
    else:
        task_len = len(data[0])
        last_id = data[0][task_len - 1]["id"]
        id = last_id + 1
    new_task = {
        "id": id,
        "ip": ip,
        "type": "regular scan",
        "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
    }
    new_data = []
    if len(data) == 0:
        new_data.append(new_task)
    else:
        new_data = data[0]
        new_data.append(new_task)
    with open('tasks.json', 'w') as file_json:
        json.dump(new_data, file_json)
    file_json.close()

nm = nmap.PortScanner()
target = nm.scan(hosts=ip)
response = target['scan']

```

```
with open(f'{id}.json', "w") as result:
```

```
    json.dump(response, result)
```

```
result.close()
```

```
return (response)
```

```
#Extensive Scan
```

```
@app.route('/extensive_scan', methods=['GET', 'POST'])
```

```
def extensive_scan():
```

```
    if request.method == 'GET':
```

```
        ip = request.args.get("ip")
```

```
    elif request.method == 'POST':
```

```
        ip = request.form.get("ip")
```

```
    data = [json.loads(line) for line in open('tasks.json', 'r+')]

```

```
    if len(data) == 0:
```

```
        id = 1
```

```
    else:
```

```
        task_len = len(data[0])
```

```
        last_id = data[0][task_len - 1]["id"]
```

```
        id = last_id + 1
```

```
    new_task = {
```

```
        "id": id,
```

```
        "ip": ip,
```

```
        "type": "extensive scan",
```

```
        "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
```

```
    }
```

```
    new_data = []
```

```
    if len(data) == 0:
```

```
        new_data.append(new_task)
```

```
    else:
```

```
        new_data = data[0]
```

```
        new_data.append(new_task)
```

```
    with open('tasks.json', 'w') as file_json:
```

```
    json.dump(new_data, file_json)
file_json.close()
```

```
nm = nmap.PortScanner()
target = nm.scan(hosts=ip, arguments="-sV -A -O")
response = target['scan']
with open(f"{id}.json", "w") as result:
    json.dump(response, result)
result.close()
return (response)
```

#Vulnerability Scan

```
@app.route('/vuln_scan', methods=['GET', 'POST'])
def vuln_scan():
    if request.method == 'GET':
        ip = request.args.get("ip")
    elif request.method == 'POST':
        ip = request.form.get("ip")
    data = [json.loads(line) for line in open('tasks.json', 'r+')]
    if len(data) == 0:
        id = 1
    else:
        task_len = len(data[0])
        last_id = data[0][task_len - 1]["id"]
        id = last_id + 1
    new_task = {
        "id": id,
        "ip": ip,
        "type": "vulnerability scan",
        "time": datetime.now().strftime("%d/%m/%Y, %H:%M:%S"),
    }
    new_data = []
```

```

if len(data) == 0:
    new_data.append(new_task)
else:
    new_data = data[0]
    new_data.append(new_task)
with open('tasks.json', 'w') as file_json:
    json.dump(new_data, file_json)
file_json.close()

nm = nmap.PortScanner()
target = nm.scan(hosts=ip, arguments="-sV -A -O --script=vulners.nse,vulscan/vulscan.nse")
response = target['scan']
with open(f"{id}.json", "w") as result:
    json.dump(response, result)
result.close()
return (response)

#Returning saved report
@app.route('/report', methods=['GET', 'POST'])
def report():
    if request.method == 'GET':
        id = request.args.get("id")
    elif request.method == 'POST':
        id = request.form.get("id")
    file_json = open(f"{id}.json")
    data=json.load(file_json)
    result = json.dumps(data)
    return result

#Return Task List
@app.route('/task_list', methods=['GET', 'POST'])
def task_list():

```

```

file_json = open("tasks.json")

data = json.load(file_json)

new_data = json.dumps(data)

return new_data


if __name__ == '__main__':

    app.run(host='localhost', port=4000)

```

React Js Frontend Code:

Public Html file:

```

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8" />

  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <meta name="theme-color" content="#000000" />

  <meta

    name="description"

    content="Web site created using create-react-app"

  />

  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

  <!--

```

manifest.json provides metadata used when your web app is installed on a user's mobile device or desktop. See <https://developers.google.com/web/fundamentals/web-app-manifest/>

```

-->

  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

  <!--

```

Notice the use of %PUBLIC_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike `/favicon.ico` or `favicon.ico`, `%PUBLIC_URL%/favicon.ico` will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running ``npm run build``.

-->

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run ``npm start`` or ``yarn start``.

To create a production bundle, use ``npm run build`` or ``yarn build``.

-->

</body>

</html>

Index.js file:

/*

Project name: Security Scanner

File: index.js // main render file

developer: Anita

*/

import React from 'react';

```
import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { BrowserRouter as Router } from "react-router-dom";
```

```
ReactDOM.render(

  <React.StrictMode>

    <Router>

      <App />

    </Router>

  </React.StrictMode>,

  document.getElementById('root')

);
```

```
// If you want to start measuring performance in your app, pass a function
```

```
// to log results (for example: reportWebVitals(console.log))
```

```
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

```
reportWebVitals();
```

App.js file:

```
/*
```

Project name: Security Scanner

File: app.js // main headers

developer: Anita

*/

import './App.css';

import { Routes } from './routes'

import React from "react";

import {

 EuiHeader,

 EuiHeaderLogo,

 EuiProvider,

} from '@elastic/eui';

import { SideNav } from "../components/templates/sideNav";

import scansvg from "../components/sentinel/scan.svg";

function App() {

 const headers=(

 <

 <EuiHeader

 sections=[

 {


```

        items: [

            <SideNav/>,

            <EuiHeaderLogo iconType={ scansvg }>Security Scanner</EuiHeaderLogo>,

        ],

        borders: 'none',

    },

    ]}

    />

</>

)

return (

    <div>

        <EuiProvider>

            {headers}

            <Routes />

        </EuiProvider>

    </div>

);

}

export default App;

```

Routes.js file:

/*

Project name: Security Scanner

File: routes.js // routing the pages

developer: Anita

*/

import React from "react";

import { Route, Routes as Switch } from "react-router-dom";

import { Network_Scan, Ping_Scan, Quick_Scan, Extensive_Scan, Vulnerability_Scan, Task_List }
from './components/sentinel';

import { Sentinel } from "./pages/sentinel";

import { Regular, Fast, Extensive, Network, Vulnerability, Sweep, Scan } from
"./components/sentinel/results";

export const Routes = () => {

return (

<Switch>

<Route exact path="/" element={ <Sentinel /> } />

<Route exact path="/network_scan" element={ <Network_Scan /> } />

<Route path="/ping_scan" element={ <Ping_Scan /> } />

<Route path="/quick_scan" element={ <Quick_Scan /> } />

<Route path="/extensive_scan" element={ <Extensive_Scan /> } />

```

    <Route path='/vulnerability_scan' element={<Vulnerability_Scan />} />

    <Route path='/task_list' element={<Task_List />} />

    <Route path="/fast/:id" element={<Fast/>} />

    <Route path="/regular/:id" element={<Regular />} />

    <Route path="/extensive/:id" element={<Extensive />} />

    <Route path="/network/:id" element={<Network />} />

    <Route path="/vulnerability/:id" element={<Vulnerability />} />

    <Route path="/sweep/:id" element={<Sweep />} />

    <Route path="/scan/:id" element={<Scan />} />

  </Switch>

)

}

```

Sentinel.js file:

```
/*
```

Project name: Security Scanner

File: sentinel.js // main front page

developer: Anita

```
*/
```

```
import '@elastic/eui/dist/eui_theme_light.css'
```

```
import React from "react";
```

```
import {
```

```

    EuiCard,

    EuiProvider,

    EuiPage,

    EuiPageBody,

    useEuiTheme,

    EuiText,

    EuiSpacer

  } from '@elastic/eui';

import scanner from './scanner.svg';

export const Sentinel = ({ Page }) => {

  const euiTheme = useEuiTheme();

  return (

    <EuiProvider colorMode={euiTheme.colorMode}>

      <EuiPage>

        <EuiPageBody>

          <EuiCard title={"Security Scanner"} image={<div><img src={scanner} alt="scanner"
height={250} /> </div>} >

            <EuiText>

              <h3>A smart network scan toolkit</h3>

              <h4>Check your network and system for vulnerabilities</h4>

```

```

<div style={{ textAlign:"center" }}>

    <ul style={{ display:"inline-table" }}>

        <li style={{ textAlign:"left" }}>Network Scan</li>

        <li style={{ textAlign:"left" }}>Ping Scan</li>

        <li style={{ textAlign:"left" }}>Quick Scan</li>

        <li style={{ textAlign:"left" }}>Extensive Scan</li>

        <li style={{ textAlign:"left" }}>Vulnerability Scan</li>

    </ul>

    <EuiSpacer size="xxl" />

</div>

</EuiText>

</EuiCard>

</EuiPageBody>

</EuiPage>

</EuiProvider>

);

};

```

SideNav.js:

```
/*
```

Project name: Security Scanner

File: sidenav.js // side collapsible nav

developer: Anita

*/

```
import React, { useState } from 'react';
```

```
import {
```

```
  EuiCollapsibleNav,
```

```
  EuiHeaderSectionItemButton,
```

```
  EuiCollapsibleNavGroup,
```

```
  EuiPinnableListGroup,
```

```
  EuiIcon,
```

```
  useGeneratedHtmlId,
```

```
  EuiButton,
```

```
  EuiSpacer
```

```
} from '@elastic/eui';
```

```
import { useNavigate } from "react-router-dom";
```

```
export const SideNav = () => {
```

```
  const navigate = useNavigate();
```

```
  const [navIsOpen, setNavIsOpen] = useState(
```

```
    JSON.parse(
```

```
      String(localStorage.getItem('euiCollapsibleNavExample--isDocked'))
```

```
    ) || false
```

```

);

const [navIsDocked, setNavIsDocked] = useState(

  JSON.parse(

    String(localStorage.getItem('euiCollapsibleNavExample--isDocked'))

  ) || false

);

const collapsibleNavId = useGeneratedHtmlId({ prefix: 'collapsibleNav' });

const TopNavLinks: EuiPinnableListGroupItemProps[] = [

  {

    label: 'Home',

    iconType: 'home',

    isActive: true,

    pinnable: false,

    onClick: () => {navigate("/"); setNavIsOpen(false)},

  },

  { label: 'Network Scan', pinnable: false, onClick: () => {navigate("/network_scan");

setNavIsOpen(false)} },

  { label: 'Ping Scan', pinnable: false, onClick: () => {navigate("/ping_scan"); setNavIsOpen(false)} },

  { label: 'Quick Scan', pinnable: false, onClick: () => {navigate("/quick_scan"); setNavIsOpen(false)

} },

  { label: 'Extensive Scan', pinnable: false, onClick: () => {navigate("/extensive_scan");

setNavIsOpen(false)} },

```

```
    { label: 'Vulnerability scan', pinnable: false, onClick: () => {navigate("/vulnerability_scan");  
setNavIsOpen(false)} },
```

```
    { label: 'Task List', pinnable: false, onClick: () => {navigate("/task_list"); setNavIsOpen(false) } },
```

```
];
```

```
return(  
  <
```

```
<
```

```
<EuiCollapsibleNav
```

```
  id={collapsibleNavId}
```

```
  isOpen={navIsOpen}
```

```
  isDocked={navIsDocked}
```

```
  size={240}
```

```
  button={
```

```
    <EuiHeaderSectionItemButton
```

```
      aria-label="Toggle main navigation"
```

```
      onClick={() => setNavIsOpen(!navIsOpen)}
```

```
    >
```

```
    <EuiIcon type={'menu'} size="m" aria-hidden="true" />
```

```
  </EuiHeaderSectionItemButton>
```

```
}
```



```
onClose={()=>setNavIsOpen(false)}
```

```
>
```

```
<EuiCollapsibleNavGroup background="light">
```

```
  <EuiPinnableListGroup
```

```
    listItems={TopNavLinks}
```

```
    onPinClick={() => {}}
```

```
    maxWidth="none"
```

```
    color="text"
```

```
    gutterSize="none"
```

```
    size="s"
```

```
  />
```

```
</EuiCollapsibleNavGroup>
```

```
<EuiSpacer />
```

```
<span />
```

```
<EuiButton
```

```
  onClick={() => {
```

```
    setNavIsDocked(!navIsDocked);
```

```
    localStorage.setItem(
```

```
      'euiCollapsibleNavExample--isDocked',
```

```
      JSON.stringify(!navIsDocked)
```

```

        );

    }}

    >

    Docked: {navIsDocked ? 'on' : 'off'}

    </EuiButton>

    </EuiCollapsibleNav>

    </>

    );

};

```

Network Scan .js file:

```

/*

Project name: Security Scanner

File: network_scan.js

developer: Anita

*/

import React, {useState} from "react";

import {

    EuiPage,

    EuiPageBody,

    EuiProvider,

    useEuiTheme,

```

```

    EuiCard,

    EuiPageContent,

    EuiPageContentHeader,

    EuiText,

    EuiPageContentBody,

    EuiFieldText,

    EuiSpacer,

    EuiButton, EuiBasicTable

  } from "@elastic/eui";

export const Network_Scan = () => {

  const euiTheme = useEuiTheme();

  const [ipaddr, setIPAddr] = useState("");

  const [scanResult, setScanResult] = useState();

  const data = new FormData();

  const onChangeText = (e) => {

    setIPAddr(e.target.value);

  };

  const onClickHandler = () => {

    data.append('ip', ipaddr);

    fetch('http://localhost:4000/network_scan', {

      method: 'POST',

```

```

        body: data,

    })

    .then((response) => response.json())

    .then((result) => {

        console.log('Success', result);

        const data = Object.keys(result);

        const data_length = data.length;

        let network = [];

        for (let i = 0; i<data_length; i++){

            const ip= data[i];

            network.push({

                ip: result[ip].addresses.ipv4,

                mac: result[ip].addresses.mac,

                hostname: result[ip].hostnames[0].name,

                vendor: result[ip].vendor,

                status: result[ip].status.state,

            })

        }

        console.log(network);

        setScanResult(network);

    })

```

```
        .catch((error) => {

            console.error('Error:', error);

        });

};

const column = [

    {

        field: 'ip',

        name: 'IP address'

    },

    {

        field: 'hostname',

        name: 'Host Name'

    },

    {

        field: 'mac',

        name: 'MAC address'

    },

    {

        field: 'vendor',

        name: 'MAC Vendor'

    },

    ],
```

```

{
    field: 'status',

    name: 'Status'

},

]

if (!scanResult) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Network Scan</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiFieldText

                                placeholder="enter network address in format 192.168.x.x/subnet"

                                value={ipaddr}

                                onChange={(e) => onChangeText(e)}

```

```

        />

        <EuiSpacer />

        <EuiButton type="primary" size="s" onClick={onClickHandler}>

            Submit

        </EuiButton>

    </EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Network Scan</h1>

                        </EuiText>

```

```

</EuiPageContentHeader>

<EuiPageContentBody>

  <EuiFieldText

    placeholder="enter network address in format 192.168.x.x/subnet"

    value={ipaddr}

    onChange={(e) => onChangeText(e)}

  />

  <EuiSpacer />

  <EuiButton type="primary" size="s" onClick={onClickHandler}>

    Submit

  </EuiButton>

  <EuiSpacer />

  <EuiCard title={"Result"} textAlign={"left"}>

    <EuiText>

      <h3>Scan for {ipaddr}</h3>

    </EuiText>

    <EuiSpacer />

    <EuiBasicTable

      items={scanResult}

      columns={column}

    />

```



```

        </EuiCard>

        </EuiPageContentBody>

    </EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Ping_Scan.js file:

```

/*

Project name: Security Scanner

File: ping_scan.js

developer: Anita

*/

import React, {Fragment, useState, useMemo} from "react";

import {

    EuiPage,

    EuiPageBody,

    EuiProvider,

    useEuiTheme, EuiPageContent,

    EuiPageContentHeader,

```

```

    EuiPageContentBody,

    EuiSpacer,

    EuiTab,

    EuiTabs,

    EuiText,

  } from "@elastic/eui";

import {Ping_Sweep} from "../ping_scan/ping_sweep";

import {PingScan} from '../ping_scan/ping_scan';

export const Ping_Scan = () => {

  const euiTheme = useEuiTheme();

  const tabs = [

    {

      id: 'ping',

      name: 'Ping Scan',

      content: (

        <Fragment>

          <EuiSpacer />

          <PingScan />

        </Fragment>

      ),

    },

  ],

```

```

    {

      id: 'sweep',

      name: 'Ping Sweep',

      content: (

        <Fragment>

          <EuiSpacer />

          <Ping_Sweep/>

        </Fragment>

      ),

    }

  ];

  const [selectedTabId, setSelectedTabId] = useState('ping');

  const selectedTabContent = useMemo(() => {

    return tabs.find((obj) => obj.id === selectedTabId)?.content;

  }, [selectedTabId, tabs]);

  const onSelectedTabChange = (id: string) => {

    setSelectedTabId(id);

  };

  const renderTabs = () => {

    return tabs.map((tab, index) => (

      <EuiTab

```

```

    key={index}

    onClick={() => onSelectedTabChange(tab.id)}

    isSelected={tab.id === selectedTabId}

  >

    {tab.name}

  </EuiTab>

));

};

return (

  <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

      <EuiPageBody>

        <EuiPageContent>

          <EuiPageContentHeader>

            <EuiText>

              <h1>Ping Scan</h1>

            </EuiText>

          </EuiPageContentHeader>

          <EuiPageContentBody>

            <EuiTabs>{renderTabs()}</EuiTabs>

            {selectedTabContent}

```

```

        </EuiPageContentBody>

      </EuiPageContent>

    </EuiPageBody>

  </EuiPage>

</EuiProvider>

)

}

```

Pingscan/ping_scan.js file:

```

/*

Project name: Security Scanner

File: pingscan/ping_scan.js

developer: Anita

*/

import React, {Fragment, useState, useMemo} from "react";

import {

  EuiPage,

  EuiPageBody,

  EuiProvider,

  useEuiTheme, EuiPageContent,

  EuiPageContentHeader,

  EuiPageContentBody,

```

```
EuiSpacer,  
  
EuiTab,  
  
EuiTabs,  
  
EuiCard,  
  
EuiBasicTable,  
  
EuiText, EuiFieldText, EuiButton, EuiAccordion, EuiPanel,  
  
} from "@elastic/eui";
```

```
export const PingScan = () => {  
  
  const euiTheme = useEuiTheme();  
  
  const [ipaddr, setIPAddr] = useState("");  
  
  const [scanResult, setScanResult] = useState();  
  
  const data = new FormData();  
  
  const onChangeText = (e) => {  
  
    setIPAddr(e.target.value);  
  
  };  
  
  const onClickHandler = () => {  
  
    data.append('ip', ipaddr);  
  
    fetch('http://localhost:4000/ping_scan', {  
  
      method: 'POST',  
  
      body: data,
```

```

    })

    .then((response) => response.json())

    .then((result) => {

        setScanResult(result)

    })

}

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    const vendorKey= Object.keys(scanResult[ip].vendor);

    let keys: string[]= Object.keys(address);

    return(

        <>

        <EuiText><h3>Addresses</h3></EuiText>

        <EuiPanel color="subdued">

            {keys.map((key) => (

                <EuiText>

                    <h4>{key} : {scanResult[ip].addresses[key]}</h4>

                </EuiText>

            ))}

            <EuiText><h4>mac-vendor: {scanResult[ip].vendor[vendorKey]}</h4></EuiText>

```

```

</EuiPanel>

<EuiText>

  <h4>{scanResult[ip].message}</h4>

</EuiText>

</>

)

}

if(!scanResult) {

  return (

    <EuiProvider colorMode={euiTheme.colorMode}>

      <EuiPage>

        <EuiPageBody>

          <EuiPageContentBody>

            <EuiFieldText

              placeholder="enter IP address"

              value={ipaddr}

              onChange={(e) => onChangeText(e)}

            />

            <EuiSpacer/>

            <EuiButton type="primary" size="s" onClick={onClickHandler}>

              Submit

```



```

        </EuiButton>

    </EuiPageContentBody>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContentBody>

                    <EuiFieldText

                        placeholder="enter IP address"

                        value={ipaddr}

                        onChange={(e) => onChangeText(e)}

                    />

                    <EuiSpacer/>

                    <EuiButton type="primary" size="s" onClick={onClickHandler}>

                        Submit

                    </EuiButton>

```

```

        <EuiCard title={ "Result" } textAlign="left">

            <EuiText>

                <h2>{ Object.keys(scanResult) }</h2>

            </EuiText>

            <EuiSpacer />

            {renderAddress()}

            <EuiSpacer />

        </EuiCard>

    </EuiPageContentBody>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Pingscan/ping_sweep.js file:

```

/*

Project name: Security Scanner

File: pingscan/ping_sweep.js

developer: Anita

*/

import React, {Fragment, useState, useMemo} from "react";

```

```

import {

  EuiPage,

  EuiPageBody,

  EuiProvider,

  useEuiTheme, EuiPageContent,

  EuiPageContentHeader,

  EuiPageContentBody,

  EuiSpacer,

  EuiTab,

  EuiTabs,

  EuiCard,

  EuiBasicTable,

  EuiText, EuiFieldText, EuiButton,

} from "@elastic/eui";

export const Ping_Sweep = () => {

  const euiTheme = useEuiTheme();

  const [net, setNet] = useState("");

  const [start, setStart] = useState("");

  const [end, setEnd] = useState("");

  const [scanResult, setScanResult] = useState("");

```

```

const [pageIndex, setPageIndex] = useState(0);

const [pageSize, setPageSize] = useState(10);

const [showPerPageOptions, setShowPerPageOptions] = useState(true);

const [totalitem, setTotalItem] = useState();

const onTableChange = ({ page = {} }) => {

const { index: pageIndex, size: pageSize } = page;

setPageIndex(pageIndex);

setPageSize(pageSize);

};

const ArraySort = (index, size) => {

const pageStart = size * index;

const pageEnd = pageStart + size;

const result = [];

for (let i = pageStart; i < pageEnd; i++) {

if (i < totalitem) {

result.push(scanResult[i]);

} else {

break;

}

}

return result;

```

```

};

const pagination = {

  pageIndex,

  pageSize,

  totalItemCount: totalItem,

  pageSizeOptions: [50, 10, 0],

  showPerPageOptions,

};

const data = new FormData();

const onClickHandler = () => {

  data.append('net', net);

  data.append('st1', start);

  data.append('en1', end);

  fetch('http://localhost:4000/ping_sweep', {

    method: 'POST',

    body: data,

  })

    .then((response) => response.json())

    .then((result) => {

      setScanResult(result);

      setTotalItem(result.length);

```

```

    })

}

const onChangeTextNet = (e) => {

    setNet(e.target.value);

};

const onChangeTextStart = (e) => {

    setStart(e.target.value);

};

const onChangeTextEnd = (e) => {

    setEnd(e.target.value);

};

const column = [

    {

        field: 'address',

        name: 'IP Address'

    },

    {

        field: 'response',

        name: 'Response',

    }

];

```

```

if(!scanResult) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContentBody>

                        <EuiFieldText

                            placeholder="enter Network address"

                            value={net}

                            onChange={(e) => onChangeTextNet(e)}

                        />

                        <EuiSpacer/>

                        <EuiFieldText

                            placeholder="enter Start"

                            value={start}

                            onChange={(e) => onChangeTextStart(e)}

                        />

                        <EuiSpacer/>

                        <EuiFieldText

                            placeholder="enter End"

                            value={end}

```

```

        onChange={ (e) => onChangeTextEnd(e) }

    />

    <EuiSpacer/>

    <EuiButton type="primary" size="s" onClick={ onClickHandler }>

        Submit

    </EuiButton>

</EuiPageContentBody>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return(

    <EuiProvider colorMode={ euiTheme.colorMode }>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContentBody>

                    <EuiFieldText

                        placeholder="enter Network address"

                        value={ net }

                        onChangeText={ (e) => onChangeTextNet(e) }

```



```

/>

<EuiSpacer/>

<EuiFieldText

  placeholder="enter Start"

  value={start}

  onChangeText={(e) => onChangeTextStart(e)}

/>

<EuiSpacer/>

<EuiFieldText

  placeholder="enter End"

  value={end}

  onChangeText={(e) => onChangeTextEnd(e)}

/>

<EuiSpacer/>

<EuiButton type="primary" size="s" onClick={onClickHandler}>

  Submit

</EuiButton>

<EuiCard title="Result" textAlign="left">

  <EuiBasicTable

    items={ArraySort(pageIndex, pageSize)}

    columns={column}

```

```

        pagination={pagination}

        onChange={onTableChange}

    />

</EuiCard>

</EuiPageContentBody>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Quick_scan.js file:

```

/*

Project name: Security Scanner

File: quick_scan.js

developer: Anita

*/

import React, {Fragment, useState, useMemo} from "react";

import {

    EuiPage,

    EuiPageBody,

    EuiProvider,

```

```

    useEuiTheme, EuiPageContent,

    EuiPageContentHeader,

    EuiPageContentBody,

    EuiSpacer,

    EuiTab,

    EuiTabs,

    EuiText,

  } from "@elastic/eui";

import { Fast_Scan } from "../quick_scan/fast_scan";

import { Regular_Scan } from "../quick_scan/regular_scan";

export const Quick_Scan = () => {

  const euiTheme = useEuiTheme();

  const tabs = [

    {

      id: 'fast',

      name: 'Fast Scan',

      content: (

        <Fragment>

          <EuiSpacer />

          <Fast_Scan />


```

```

        </Fragment>

    ),

},

{

    id: 'regular',

    name: 'Regular Scan',

    content: (

        <Fragment>

            <EuiSpacer />

            <Regular_Scan />

        </Fragment>

    ),

}

];

const [selectedTabId, setSelectedTabId] = useState('fast');

const selectedTabContent = useMemo(() => {

    return tabs.find((obj) => obj.id === selectedTabId)?.content;

}, [selectedTabId, tabs]);

const onSelectedTabChange = (id: string) => {

    setSelectedTabId(id);

};

```

```

const renderTabs = () => {

return tabs.map((tab, index) => (

  <EuiTab

    key={index}

    onClick={() => onSelectedTabChange(tab.id)}

    isSelected={tab.id === selectedTabId}

  >

    {tab.name}

  </EuiTab>

));

});

return (

  <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

      <EuiPageBody>

        <EuiPageContent>

          <EuiPageContentHeader>

            <EuiText>

              <h1>Quick Scan</h1>

            </EuiText>

          </EuiPageContentHeader>

```

```

        <EuiPageContentBody>

            <EuiTabs>{renderTabs()}</EuiTabs>

            {selectedTabContent}

        </EuiPageContentBody>

    </EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Quickscan/fastscan.js file:

```
/*
```

Project name: Security Scanner

File: fast_scan.js

developer: Anita

```
*/
```

```
import React, {useState} from "react";
```

```
import {
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiProvider,
```

```

    useEuiTheme, EuiPageContent,

    EuiPageContentHeader,

    EuiPageContentBody,

    EuiFieldText,

    EuiButton,

    EuiSpacer,

    EuiCard,

    EuiBasicTable,

    EuiText, EuiAccordion, EuiPanel,
  } from "@elastic/eui";

export const Fast_Scan = () => {

  const euiTheme = useEuiTheme();

  const [ipaddr, setIPAddr] = useState("");

  const [scanResult, setScanResult] = useState();

  const [portData, setPortData] = useState();

  const data = new FormData();

  const onChangeText = (e) => {

    setIPAddr(e.target.value);

  };

  const onClickHandler = () => {

    data.append('ip', ipaddr);

```

```

fetch('http://localhost:4000/fast_scan', {

  method: 'POST',

  body: data,

})

.then((response) => response.json())

.then((result) => {

  const ip = Object.keys(result);

  if(result[ip].hasOwnProperty('tcp')) {

    const tcp = result[ip].tcp;

    const port_keys = Object.keys(tcp);

    const port_length = port_keys.length;

    let ports= [];

    for (let i = 0; i< port_length; i++){

      const port= port_keys[i];

      ports.push({

        port: port_keys[i],

        state: tcp[port].state,

        reason: tcp[port].reason,

        name: tcp[port].name,

        conf: tcp[port].conf,

      })

```



```

    }

    setPortData(ports);

    }

    setScanResult(result);

  })

  .catch((error) => {

    console.error('Error:', error);

  });

};

const columns = [

  {

    field: 'port',

    name: 'Port'

  },

  {

    field: 'state',

    name: 'State'

  },

  {

    field: 'reason',

    name: 'Reason'

```

```

    },

    {

      field: 'name',

      name: 'Name'

    },

    {

      field: 'conf',

      name: 'Conf'

    },

  ],

];

const renderAddress = () => {

  const ip= Object.keys(scanResult);

  const address = scanResult[ip].addresses;

  let keys: string[]= Object.keys(address);

  return(

    <>

    <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={ true }>

      <EuiPanel color="subdued">

        {keys.map((key) => (

          <EuiText>

            {key} : {scanResult[ip].addresses[key]}


```

```

        </EuiText>

    )})

</EuiPanel>

</EuiAccordion>

</>

)

}

if (!scanResult) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Fast Scan</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiFieldText

                                placeholder="enter IP address"

```

```

        value={ipaddr}

        onChange={(e) => onChangeText(e)}

    />

    <EuiSpacer />

    <EuiButton type="primary" size="s" onClick={onClickHandler}>

        Submit

    </EuiButton>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

if (scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

```

```

<EuiText>

    <h1>Fast Scan</h1>

</EuiText>

</EuiPageContentHeader>

<EuiPageContentBody>

    <EuiFieldText

        placeholder="enter IP address"

        value={ipaddr}

        onChange={(e) => onChangeText(e)}

    />

    <EuiSpacer />

    <EuiButton type="primary" size="s" onClick={onClickHandler}>

        Submit

    </EuiButton>

    <EuiSpacer />

    <EuiCard title={"Result"} textAlign={"left"}>

        <EuiText>

            <h2>{Object.keys(scanResult)}</h2>

        </EuiText>

        <EuiSpacer />

        {renderAddress()}

```

```

        <EuiSpacer />

        <EuiText>

            <h3>No Open Ports</h3>

        </EuiText>

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Fast Scan</h1>

                        </EuiText>

```

```
</EuiPageContentHeader>
```

```
<EuiPageContentBody>
```

```
  <EuiFieldText
```

```
    placeholder="enter IP address"
```

```
    value={ipaddr}
```

```
    onChange={(e) => onChangeText(e)}
```

```
  <EuiSpacer />
```

```
  <EuiButton type="primary" size="s" onClick={onClickHandler}>
```

```
    Submit
```

```
</EuiButton>
```

```
<EuiSpacer />
```

```
<EuiCard title={"Result"} textAlign={"left"}>
```

```
  <EuiText>
```

```
    <h2>{Object.keys(scanResult)}</h2>
```

```
  </EuiText>
```

```
  <EuiSpacer />
```

```
    {renderAddress() }
```

```
  <EuiSpacer />
```

```
  <EuiBasicTable
```

```
    items={portData}
```

```

        columns={columns}

      />

    </EuiCard>

  </EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Quickscan/regular_scan.js file:

```

/*

Project name: Security Scanner

File: regular_scan.js

developer: Anita

*/

import React, {useState} from "react";

import {

  EuiPage,

  EuiPageBody,

  EuiProvider,

```



```

    useEuiTheme, EuiPageContent,

    EuiPageContentHeader,

    EuiPageContentBody,

    EuiFieldText,

    EuiButton,

    EuiSpacer,

    EuiText,

    EuiCard,

    EuiAccordion,

    EuiPanel,

    EuiBasicTable,

  } from "@elastic/eui";

export const Regular_Scan = () => {

  const euiTheme = useEuiTheme();

  const [ipaddr, setIPAddr] = useState("");

  const [scanResult, setScanResult] = useState();

  const [portData, setPortData] = useState();

  const data = new FormData();

  const onChangeText = (e) => {

    setIPAddr(e.target.value);

  };

```

```

const onClickHandler = () => {

  data.append('ip', ipaddr);

  fetch('http://localhost:4000/reg_scan', {

    method: 'POST',

    body: data,

  })

  .then((response) => response.json())

  .then((result) => {

    const ip = Object.keys(result);

    if(result[ip].hasOwnProperty('tcp')) {

      const tcp = result[ip].tcp;

      const port_keys = Object.keys(tcp);

      const port_length = port_keys.length;

      let ports = [];

      for (let i = 0; i < port_length; i++) {

        const port = port_keys[i];

        ports.push({

          port: port_keys[i],

          state: tcp[port].state,

          reason: tcp[port].reason,

          name: tcp[port].name,

```

```

        product: tcp[port].product,

        conf: tcp[port].conf,

        cpe: tcp[port].cpe

    })

}

setPortData(ports);

}

setScanResult(result);

})

.catch((error) => {

    console.error('Error:', error);

});

};

const columns = [

    {

        field: 'port',

        name: 'Port'

    },

    {

        field: 'state',

        name: 'State'

```

```
},  
  
{  
  
  field: 'reason',  
  
  name: 'Reason'  
  
},  
  
{  
  
  field: 'name',  
  
  name: 'Name'  
  
},  
  
{  
  
  field: 'product',  
  
  name: 'Product'  
  
},  
  
{  
  
  field: 'conf',  
  
  name: 'Conf'  
  
},  
  
{  
  
  field: 'cpe',  
  
  name: 'CPE'  
  
},
```

```

];

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    let keys: string[]= Object.keys(address);

    return(

        <>

        <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={true}>

            <EuiPanel color="subdued">

                {keys.map((key) => (

                    <EuiText>

                        {key} : {scanResult[ip].addresses[key]}

                    </EuiText>

                )))}

            </EuiPanel>

        </EuiAccordion>

        </>

    )

}

if (!scanResult){

    return (

```

```

<EuiProvider colorMode={euiTheme.colorMode}>

  <EuiPage>

    <EuiPageBody>

      <EuiPageContent>

        <EuiPageContentHeader>

          <EuiText>

            <h1>Regular Scan</h1>

          </EuiText>

        </EuiPageContentHeader>

        <EuiPageContentBody>

          <EuiFieldText

            placeholder="enter IP address"

            value={ipaddr}

            onChange={(e) => onChangeText(e)}

          />

          <EuiSpacer />

          <EuiButton type="primary" size="s" onClick={onClickHandler}>

            Submit

          </EuiButton>

        </EuiPageContentBody>

      </EuiPageContent>

```

```

        </EuiPageBody>

    </EuiPage>

</EuiProvider>

)

}

if (scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Regular Scan</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiFieldText

                                placeholder="enter IP address"

                                value={ipaddr}

                                onChange={(e) => onChangeText(e)}

```

```

/>

<EuiSpacer />

<EuiButton type="primary" size="s" onClick={onClickHandler}>

    Submit

</EuiButton>

<EuiSpacer />

<EuiCard title="Result" textAlign="left">

    <EuiText>

        <h2>{Object.keys(scanResult)}</h2>

    </EuiText>

    <EuiSpacer />

    {renderAddress()}

    <EuiSpacer />

    <EuiText>

        <h3>No Open Ports</h3>

    </EuiText>

</EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

```



```

    </EuiProvider>

  )

}

return (

  <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

      <EuiPageBody>

        <EuiPageContent>

          <EuiPageContentHeader>

            <EuiText>

              <h1>Regular Scan</h1>

            </EuiText>

          </EuiPageContentHeader>

          <EuiPageContentBody>

            <EuiFieldText

              placeholder="enter IP address"

              value={ipaddr}

              onChange={(e) => onChangeText(e)}

            />

            <EuiSpacer />

            <EuiButton type="primary" size="s" onClick={onClickHandler}>

```

```

        Submit

    </EuiButton>

    <EuiSpacer />

    <EuiCard title="Result" textAlign="left">

        <EuiText>

            <h2>{ Object.keys(scanResult) }</h2>

        </EuiText>

        <EuiSpacer />

        {renderAddress()}

        <EuiSpacer />

        <EuiBasicTable

            items={portData}

            columns={columns}

        />

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

```

```
}
```

Extensive scan.js file:

```
/*
```

Project name: Security Scanner

File: extensive_scan.js

developer: Anita

```
*/
```

```
import React, {useState} from "react";
```

```
import {
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiProvider,
```

```
    useEuiTheme,
```

```
    EuiPageContentHeader,
```

```
    EuiText,
```

```
    EuiPageContentBody,
```

```
    EuiFieldText,
```

```
    EuiSpacer,
```

```
    EuiButton,
```

```
    EuiPageContent, EuiAccordion, EuiPanel, EuiCard, EuiBasicTable
```

```
} from "@elastic/eui";
```

```

export const Extensive_Scan = () => {

  const euiTheme = useEuiTheme();

  const [ipaddr, setIPAddr] = useState("");

  const [scanResult, setScanResult] = useState();

  const [portData, setPortData] = useState();

  const [osData, setOsData] = useState();

  const data = new FormData();

  const onChangeText = (e) => {

    setIPAddr(e.target.value);

  };

  const onClickHandler = () => {

    data.append('ip', ipaddr);

    fetch('http://localhost:4000/extensive_scan', {

      method: 'POST',

      body: data,

    })

    .then((response) => response.json())

    .then((result) => {

      const ip = Object.keys(result);

      if(result[ip].hasOwnProperty('tcp')) {

        const tcp = result[ip].tcp;

```

```

const port_keys = Object.keys(tcp);

const port_length = port_keys.length;

let ports = [];

for (let i = 0; i < port_length; i++) {

    const port = port_keys[i];

    ports.push({

        port: port_keys[i],

        state: tcp[port].state,

        reason: tcp[port].reason,

        name: tcp[port].name,

        product: tcp[port].product,

        version: tcp[port].version,

        extrainfo: tcp[port].extrainfo,

        conf: tcp[port].conf,

        cpe: tcp[port].cpe,

        title: tcp[port].script['http-title'],

        serverheader: tcp[port].script['http-server-header']

    })

}

setPortData(ports);

}

```

```

const osmatch = result[ip].osmatch;

const os_length = osmatch.length;

let os_result = [];

for (let i = 0; i < os_length; i++) {

    const os = osmatch[i];

    os_result.push({

        name: os.name,

        accuracy: os.accuracy,

        vendor: os.osclass[0].vendor,

        osfamily: os.osclass[0].osfamily,

    })

}

setOsData(os_result);

setScanResult(result);

})

.catch((error) => {

    console.error('Error:', error);

});

};

const columns_os = [

    {

```

```

        field: 'name',

        name: 'Name'

    },

    {

        field: 'accuracy',

        name: 'Accuracy'

    },

    {

        field: 'vendor',

        name: 'Vendor'

    },

    {

        field: 'osfamily',

        name: 'Os Family'

    },

];

const columns_ports = [

    {

        field: 'port',

        name: 'Port'

    },

    },

```

```
{  
  
  field: 'state',  
  
  name: 'State'  
  
},  
  
{  
  
  field: 'reason',  
  
  name: 'Reason'  
  
},  
  
{  
  
  field: 'name',  
  
  name: 'Name'  
  
},  
  
{  
  
  field: 'product',  
  
  name: 'Product'  
  
},  
  
{  
  
  field: 'version',  
  
  name: 'Version'  
  
},  
  
{
```



```
      field: 'extrainfo',

      name: 'Extra Information'

    },

    {

      field: 'conf',

      name: 'Conf'

    },

    {

      field: 'cpe',

      name: 'CPE'

    },

    {

      field: 'title',

      name: 'HTTP Title'

    },

    {

      field: 'serverheader',

      name: 'HTTP Server Header'

    },

  ];

  const renderAddress = () => {
```

```

const ip= Object.keys(scanResult);

const address = scanResult[ip].addresses;

const vendorKey= Object.keys(scanResult[ip].vendor);

let keys: string[]= Object.keys(address);

return(

    <>

    <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={true}>

        <EuiPanel color="subdued">

            {keys.map((key) => (

                <EuiText>

                    {key} : {scanResult[ip].addresses[key]}

                </EuiText>

            ))}

            mac-vendor: {scanResult[ip].vendor[vendorKey]}

        </EuiPanel>

    </EuiAccordion>

    </>

)

}

if(!scanResult) {

    return (

```

```

<EuiProvider colorMode={euiTheme.colorMode}>

  <EuiPage>

    <EuiPageBody>

      <EuiPageContent>

        <EuiPageContentHeader>

          <EuiText>

            <h1>Extensive Scan</h1>

          </EuiText>

        </EuiPageContentHeader>

        <EuiPageContentBody>

          <EuiFieldText

            placeholder="enter IP address"

            value={ipaddr}

            onChange={(e) => onChangeText(e)}

          />

          <EuiSpacer />

          <EuiButton type="primary" size="s" onClick={onClickHandler}>

            Submit

          </EuiButton>

        </EuiPageContentBody>

      </EuiPageContent>

```

```

        </EuiPageBody>

    </EuiPage>

</EuiProvider>

)

}

if (scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Extensive Scan</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiFieldText

                                placeholder="enter IP address"

                                value={ipaddr}

                                onChange={(e) => onChangeText(e)}

```

```
    />

    <EuiSpacer />

    <EuiButton type="primary" size="s" onClick={onClickHandler}>

        Submit

    </EuiButton>

    <EuiSpacer />

    <EuiCard title={ "Result" } textAlign={ "left" }>

        <EuiText>

            <h2>{ Object.keys(scanResult) } </h2>

        </EuiText>

        <EuiSpacer />

        {renderAddress()}

        <EuiSpacer />

        <EuiText>

            <h3>Port Information</h3>

            <h4>No Open Ports</h4>

        </EuiText>

        <EuiSpacer />

        <EuiText>

            <h3>OS Information</h3>

            <h4>These results may not be 100% accurate</h4>


```

```

        </EuiText>

        <EuiBasicTable

            items={osData}

            columns={columns_os}

        />

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Extensive Scan</h1>

```

```
</EuiText>
```

```
</EuiPageContentHeader>
```

```
<EuiPageContentBody>
```

```
<EuiFieldText
```

```
  placeholder="enter IP address"
```

```
  value={ipaddr}
```

```
  onChange={(e) => onChangeText(e)}
```

```
<EuiSpacer />
```

```
<EuiButton type="primary" size="s" onClick={onClickHandler}>
```

```
  Submit
```

```
</EuiButton>
```

```
<EuiSpacer />
```

```
<EuiCard title={"Result"} textAlign={"left"}>
```

```
  <EuiText>
```

```
    <h2>{Object.keys(scanResult)}</h2>
```

```
  </EuiText>
```

```
  <EuiSpacer />
```

```
    {renderAddress() }
```

```
  <EuiSpacer />
```

```
  <EuiText>
```

```

        <h3>Port Information</h3>

        </EuiText>

        <EuiBasicTable

            items={portData}

            columns={columns_ports}

        />

        <EuiText>

            <h3>OS Information</h3>

            <h4>These results may not be 100% accurate</h4>

        </EuiText>

        <EuiBasicTable

            items={osData}

            columns={columns_os}

        />

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

```



```
}
```

Vulnerability_scan.js file:

```
/*
```

Project name: Security Scanner

File: vulnerability_scan.js

developer: Anita

```
*/
```

```
import React, {Fragment, useState} from "react";
```

```
import {
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiProvider,
```

```
    useEuiTheme,
```

```
    EuiCard,
```

```
    EuiPageContentHeader,
```

```
    EuiText,
```

```
    EuiPageContentBody,
```

```
    EuiFieldText,
```

```
    EuiSpacer,
```

```
    EuiButton,
```

```
    EuiLink,
```

EuiPageContent, EuiAccordion, EuiPanel, EuiBasicTable

```
} from "@elastic/eui";

export const Vulnerability_Scan = () => {

  const euiTheme = useEuiTheme();

  const [ipaddr, setIPAddr] = useState("");

  const [scanResult, setScanResult] = useState();

  const [osData, setOsData] = useState();

  const [portData, setPortData] = useState();

  const data = new FormData();

  const onChangeText = (e) => {

    setIPAddr(e.target.value);

  };

  const onClickHandler = () => {

    data.append('ip', ipaddr);

    fetch('http://localhost:4000/vuln_scan', {

      method: 'POST',

      body: data,

    })

    .then((response) => response.json())

    .then((result) => {

      const ip = Object.keys(result);
```

```

const osmatch = result[ip].osmatch;

const os_length = osmatch.length;

let os_result = [];

for (let i=0; i<os_length; i++) {

    const os = osmatch[i];

    os_result.push({

        name: os.name,

        accuracy: os.accuracy,

        vendor: os.osclass[0].vendor,

        osfamily: os.osclass[0].osfamily,

    })

}

if(result[ip].hasOwnProperty('tcp')) {

    const tcp = result[ip].tcp;

    const port_keys = Object.keys(tcp);

    const port_length = port_keys.length;

    let ports = [];

    for (let i = 0; i < port_length; i++) {

        const port = port_keys[i];

        ports.push({

            port: port_keys[i],

```

```

        state: tcp[port].state,

        reason: tcp[port].reason,

        name: tcp[port].name,

        product: tcp[port].product,

        version: tcp[port].version,

        extrainfo: tcp[port].extrainfo,

        conf: tcp[port].conf,

        cpe: tcp[port].cpe,

        serverheader: tcp[port].script['http-server-header']

    ))

}

setPortData(ports);

}

setOsData(os_result);

setScanResult(result);

})

.catch((error) => {

    console.error('Error:', error);

});

};

const columns_os = [

```

```
{  
  
  field: 'name',  
  
  name: 'Name'  
  
},  
  
{  
  
  field: 'accuracy',  
  
  name: 'Accuracy'  
  
},  
  
{  
  
  field: 'vendor',  
  
  name: 'Vendor'  
  
},  
  
{  
  
  field: 'osfamily',  
  
  name: 'Os Family'  
  
},  
  
];  
  
const columns_ports = [  
  
  {  
  
    field: 'port',  
  
    name: 'Port'
```

```
},  
  
{  
  
  field: 'state',  
  
  name: 'State'  
  
},  
  
{  
  
  field: 'reason',  
  
  name: 'Reason'  
  
},  
  
{  
  
  field: 'name',  
  
  name: 'Name'  
  
},  
  
{  
  
  field: 'product',  
  
  name: 'Product'  
  
},  
  
{  
  
  field: 'version',  
  
  name: 'Version'  
  
},
```

```

{
    field: 'extrainfo',

    name: 'Extra Information'

},

{

    field: 'conf',

    name: 'Conf'

},

{

    field: 'cpe',

    name: 'CPE'

},

{

    field: 'serverheader',

    name: 'HTTP Server Header'

},

}
]

```

```

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    const vendorKey= Object.keys(scanResult[ip].vendor);

```

```

let keys: string[] = Object.keys(address);

return (
  <
    <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={ true }>

      <EuiPanel color="subdued">

        {keys.map((key) => (

          <EuiText>

            {key} : {scanResult[ip].addresses[key]}

          </EuiText>

        ))}

        mac-vendor: {scanResult[ip].vendor[vendorKey]}

      </EuiPanel>

    </EuiAccordion>

  </>

)

}

const vulners_list = (port) => {

  let vulner= [];

  const ip = Object.keys(scanResult);

  const tcp = scanResult[ip].tcp;

  if (scanResult[ip].tcp[port].script.vulners) {

```



```

const vulners = scanResult[ip].tcp[port].script.vulners.split('\n');

const vulner_column = [

  {

    field: 'cve',

    name: 'CVE number',

  },

  {

    field: 'level',

    name: 'Level',

  },

  {

    field: 'weblink',

    name: 'Link',

    render: (id, item) => (

      <EuiLink href={item.weblink} target="_blank">

        {item.weblink}

      </EuiLink>

    )

  }

]

for(let i = 2; i < vulners.length; i++){

```

```

const vul = vulners[i].split("\t");

vulner.push({

  cve: vul[1],

  level: vul[2],

  weblink: vul[3]

})

}

return (

  <Fragment>

    <EuiSpacer />

    <EuiText>

      <h4>Vulners Result</h4>

      <h4>Port number: {port} </h4>

    </EuiText>

    <EuiBasicTable

      items={vulner}

      columns={vulner_column}

    />

  </Fragment>

)

}

```

```

return (

  <Fragment>

    <EuiSpacer/>

    <EuiText>

      <h3>Port number: {port}</h3>

      <h4>No result from Vulners Database</h4>

    </EuiText>

  </Fragment>

)

}

const vulscan = (port) => {

  const ip = Object.keys(scanResult);

  const tcp = scanResult[ip].tcp;

  if (scanResult[ip].tcp[port].script.vulscan) {

    const vulscan = scanResult[ip].tcp[port].script.vulscan.split('\n');

    return(

      <Fragment>

        <EuiText><h3>Vulscan</h3></EuiText>

        <EuiAccordion grow={false} buttonContent={port}>

          <EuiPanel grow={false} color="subdued">

            { vulscan.map((vul) => (

```

```

        <EuiText textAlign="left">{ vul}</EuiText>

    )))

</EuiPanel>

</EuiAccordion>

</Fragment>

)

}

return (

    <Fragment>

        <EuiSpacer/>

        <EuiText>

            <h3>Port number: {port}</h3>

            <h4>No result from Vulscan Database</h4>

        </EuiText>

    </Fragment>

)

}

const vulnerability = () => {

    const ip = Object.keys(scanResult);

    const tcp = scanResult[ip].tcp;

    const ports = Object.keys(tcp);

```

```

return(

    <Fragment>

        {ports.map((port) => (

            <>

                {vulners_list(port)}

                {vulscan(port)}

            </>

        ))}

    </Fragment>

)

}

if (!scanResult) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Vulnerability Scan</h1>

```

```

        </EuiText>

    </EuiPageContentHeader>

    <EuiPageContentBody>

        <EuiFieldText

            placeholder="enter IP address"

            value={ipaddr}

            onChange={(e) => onChangeText(e)}

        />

        <EuiSpacer/>

        <EuiButton type="primary" size="s" onClick={onClickHandler}>

            Submit

        </EuiButton>

    </EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

if (scanResult && !portData) {

    return (

```

```

<EuiProvider colorMode={euiTheme.colorMode}>

<EuiPage>

  <EuiPageBody>

    <EuiPageContent>

      <EuiPageContentHeader>

        <EuiText>

          <h1>Vulnerability Scan</h1>

        </EuiText>

      </EuiPageContentHeader>

      <EuiPageContentBody>

        <EuiFieldText

          placeholder="enter IP address"

          value={ipaddr}

          onChange={(e) => onChangeText(e)}

        />

        <EuiSpacer/>

        <EuiButton type="primary" size="s" onClick={onClickHandler}>

          Submit

        </EuiButton>

        <EuiSpacer />

        <EuiCard title={"Result"} textAlign={"left"}>

```

```
<EuiText>

    <h2>{ Object.keys(scanResult)}</h2>

</EuiText>

<EuiSpacer />

{renderAddress()}

<EuiSpacer />

<EuiText>

    <h3>OS Information</h3>

    <h4>These results may not be 100% accurate</h4>

</EuiText>

<EuiBasicTable

    items={osData}

    columns={columns_os}

/>

<EuiSpacer />

<EuiText>

    <h3>Open Ports Information</h3>

    <h4>No Open Ports</h4>

</EuiText>

</EuiCard>

</EuiPageContentBody>
```



```

        </EuiPageContent>

    </EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Vulnerability Scan</h1>

                        </EuiText>

                    </EuiPageContentHeader>

                    <EuiPageContentBody>

                        <EuiFieldText

                            placeholder="enter IP address"

                            value={ipaddr}

                            onChange={(e) => onChangeText(e)}

```

```

/>

<EuiSpacer/>

<EuiButton type="primary" size="s" onClick={onClickHandler}>

    Submit

</EuiButton>

<EuiSpacer />

<EuiCard title={ "Result" } textAlign={ "left" }>

    <EuiText>

        <h2>{ Object.keys(scanResult) } </h2>

    </EuiText>

    <EuiSpacer />

    {renderAddress()}

    <EuiSpacer />

    <EuiText>

        <h3>OS Information</h3>

        <h4>These results may not be 100% accurate</h4>

    </EuiText>

    <EuiBasicTable

        items={osData}

        columns={columns_os}

    />

```

```

        <EuiSpacer />

        <EuiText>

            <h3>Open Ports Information</h3>

        </EuiText>

        <EuiBasicTable

            items={portData}

            columns={columns_ports}

        />

        <EuiText>

            <h3>Vulnerabilites</h3>

        </EuiText>

        {vulnerability()}

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

TaskList.js file:

/*

Project name: Security Scanner

File: task_list.js

developer: Anita

*/

import React, {useEffect, useState} from "react";

import { useNavigate } from "react-router-dom";

import {

 EuiButton,

 EuiButtonEmpty,

 EuiPage,

 EuiPageBody,

 EuiPageContent,

 EuiPageContentBody,

 EuiPageContentHeader,

 EuiPageHeader,

 EuiTitle,

 EuiText,

 EuiBasicTable,

 EuiLink,

 EuiFlexItem, useEuiTheme, EuiProvider, EuiFieldText, EuiSpacer

```

} from '@elastic/eui'

export const Task_List = () => {

  const euiTheme = useEuiTheme();

  const navigate = useNavigate();

  const [list, setList] = useState();

  const [pageIndex, setPageIndex] = useState(0);

  const [pageSize, setPageSize] = useState(10);

  const [showPerPageOptions, setShowPerPageOptions] = useState(true);

  const [totalItem, setTotalItem] = useState();

  const onTableChange = ({ page = { } }) => {

    const { index: pageIndex, size: pageSize } = page;

    setPageIndex(pageIndex);

    setPageSize(pageSize);

  };

  useEffect(() => {

    fetch('http://localhost:4000/task_list')

      .then((response) => response.json())

      .then((result) => {

        setList(result)

        setTotalItem(result.length)

      })
  })
}

```

```

}, []);

const ArraySort = (index, size) => {

const pageStart = size * index;

const pageEnd = pageStart + size;

const result = [];

for (let i = pageStart; i < pageEnd; i++) {

  if (i < totalitem) {

    result.push(list[i]);

  } else {

    break;

  }

}

return result;

};

const columns = [

  {

    field: 'id',

    name: 'ID'

  },

  {

    field: 'ip',

```

```

name: 'IP address',

render: (id, item) => (

  <EuiButtonEmpty

    id="report"

    onClick={() => {

      if (item.type === "fast scan"){

        navigate(`/fast/${item.id}`)

      } else if (item.type === "regular scan") {

        navigate(`/regular/${item.id}`)

      } else if (item.type === "extensive scan") {

        navigate(`/extensive/${item.id}`)

      } else if (item.type === "network scan") {

        navigate(`/network/${item.id}`)

      } else if (item.type === "vulnerability scan") {

        navigate(`/vulnerability/${item.id}`)

      } else if (item.type === "ping sweep") {

        navigate(`/sweep/${item.id}`)

      } else if (item.type === "ping scan") {

        navigate(`/scan/${item.id}`)

      }

    }}

```

```

        >

        {item.ip}

    </EuiButtonEmpty>

)

},

{

    field: 'type',

    name: 'Scan Type'

},

{

    field: 'time',

    name: 'Time'

}

];

const pagination = {

    pageIndex,

    pageSize,

    totalItemCount: totalitem,

    pageSizeOptions: [50, 10, 0],

    showPerPageOptions,

};

```



```
return (  
  
  <EuiProvider colorMode={euiTheme.colorMode}>  
  
    <EuiPage>  
  
      <EuiPageBody>  
  
        <EuiPageContent>  
  
          <EuiPageContentHeader>  
  
            <EuiText>  
  
              <h1>Task List</h1>  
  
            </EuiText>  
  
          </EuiPageContentHeader>  
  
          <EuiPageContentBody>  
  
            <EuiBasicTable  
  
              items={ ArraySort(pageIndex, pageSize) }  
  
              columns={ columns }  
  
              pagination={ pagination }  
  
              onChange={ onTableChange }  
  
            />  
  
          </EuiPageContentBody>  
  
        </EuiPageContent>  
  
      </EuiPageBody>  
  
    </EuiPage>  
  </EuiProvider>  
)
```

```
        </EuiProvider>

    )

}
```

View Save Results file for each scan:

Network.js result file:

```
/*

Project name: Security Scanner

File: network.js

developer: Anita

*/

import React, {useState, useEffect} from 'react';

import {useParams} from "react-router-dom";

import {

    EuiPage,

    EuiProvider,

    useEuiTheme, EuiPageContent,

    EuiPageContentHeader,

    EuiLoadingSpinner,

    EuiSpacer,

    EuiText,

    EuiAccordion,
```

```

    EuiPanel,

    EuiBasicTable,

  } from "@elastic/eui";

export const Network = () => {

  const [scanResult, setScanResult] = useState();

  const euiTheme = useEuiTheme();

  const obj = useParams();

  useEffect(() => {

    fetch('http://localhost:4000/report?id='+obj.id)

      .then((response) => response.json())

      .then((result) => {

        const data = Object.keys(result);

        const data_length = data.length;

        let network = [];

        for (let i = 0; i<data_length; i++){

          const ip= data[i];

          network.push({

            ip: result[ip].addresses.ipv4,

            mac: result[ip].addresses.mac,

            hostname: result[ip].hostnames[0].name,

```

```

        vendor: result[ip].vendor,

        status: result[ip].status.state,

    })

}

setScanResult(network);

})

.catch((error) => {

    console.error('Error:', error);

});

}, [])

const column = [

{

    field: 'ip',

    name: 'IP address'

},

{

    field: 'hostname',

    name: 'Host Name'

},

{

    field: 'mac',

```

```

        name: 'MAC address'

    },

    {

        field: 'vendor',

        name: 'MAC Vendor'

    },

    {

        field: 'status',

        name: 'Status'

    },

]

if(!scanResult){

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Network Scan Result</h1>

                        </EuiText>

                    </EuiPageContentHeader>

```

```

        <EuiLoadingSpinner size="xl" />

    </EuiPageContent>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageContent>

                <EuiPageContentHeader>

                    <EuiText>

                        <h1>Network Scan Result</h1>

                    </EuiText>

                </EuiPageContentHeader>

                <EuiBasicTable

                    items={scanResult}

                    columns={column}

                />

            </EuiPageContent>

        </EuiPage>

```

```
        </EuiProvider>

    )

}
```

Ping scan result js file:

```
/*

Project name: Security Scanner

File: scan.js //ping

developer: Anita

*/

import React, {useState, useEffect} from 'react';

import {useParams} from "react-router-dom";

import {

    EuiPage,

    EuiPageBody,

    EuiProvider,

    useEuiTheme,

    EuiPageContentHeader,

    EuiText,

    EuiPageContentBody,

    EuiFieldText,

    EuiSpacer,
```

```

    EuiButton,

    EuiPageContent, EuiAccordion, EuiPanel, EuiCard, EuiBasicTable, EuiLoadingSpinner

} from "@elastic/eui";

export const Scan = () => {

    const obj = useParams();

    const euiTheme = useEuiTheme();

    const [scanResult, setScanResult] = useState("");

    useEffect(() => {

        fetch('http://localhost:4000/report?id='+ obj.id)

        .then((response) => response.json())

        .then((result) => {

            setScanResult(result)

        })

    }, [])

    const renderAddress = () => {

        const ip= Object.keys(scanResult);

        const address = scanResult[ip].addresses;

        const vendorKey= Object.keys(scanResult[ip].vendor);

        let keys: string[]= Object.keys(address);

        return(

            <>

```



```

<EuiText><h3>Addresses</h3></EuiText>

<EuiPanel color="subdued">

  {keys.map((key) => (

    <EuiText>

      <h4>{key} : {scanResult[ip].addresses[key]}</h4>

    </EuiText>

  ))}

  <EuiText><h4>mac-vendor: {scanResult[ip].vendor[vendorKey]}</h4></EuiText>

</EuiPanel>

<EuiText>

  <h4>{scanResult[ip].message}</h4>

</EuiText>

</>

)

}

if(!scanResult){

  return (

    <EuiProvider colorMode={euiTheme.colorMode}>

      <EuiPage>

        <EuiPageContent>

          <EuiPageContentHeader>

```

```

        <EuiText>

            <h1>Ping Scan Result</h1>

        </EuiText>

    </EuiPageContentHeader>

    <EuiLoadingSpinner size="xl" />

</EuiPageContent>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Ping Scan Result</h1>

                        </EuiText>

                    </EuiPageContentHeader>

                    <EuiPageContentBody>

```

```

        <EuiText>

            <h2>{ Object.keys(scanResult)}</h2>

        </EuiText>

        <EuiSpacer />

        {renderAddress()}

        <EuiSpacer />

        </EuiPageContentBody>

    </EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```

Ping sweep result js file:

```

/*

Project name: Security Scanner

File: sweep.js //ping sweep

developer: Anita

*/

import React, {useState, useEffect} from 'react';

import { useParams } from "react-router-dom";

```

```
import {

  EuiPage,

  EuiPageBody,

  EuiProvider,

  useEuiTheme,

  EuiPageContentHeader,

  EuiText,

  EuiPageContentBody,

  EuiFieldText,

  EuiSpacer,

  EuiButton,

  EuiPageContent, EuiAccordion, EuiPanel, EuiCard, EuiBasicTable, EuiLoadingSpinner

} from "@elastic/eui";
```

```
export const Sweep = () => {

  const obj = useParams();

  const euiTheme = useEuiTheme();

  const [scanResult, setScanResult] = useState("");

  const [pageIndex, setPageIndex] = useState(0);

  const [pageSize, setPageSize] = useState(10);

  const [showPerPageOptions, setShowPerPageOptions] = useState(true);
```

```

const [totalItem, setTotalItem] = useState();

useEffect(() => {

  fetch('http://localhost:4000/report?id='+ obj.id)

    .then((response) => response.json())

    .then((result) => {

      setScanResult(result);

      setTotalItem(result.length);

    })

}, [])

const onTableChange = ({ page = {} }) => {

const { index: pageIndex, size: pageSize } = page;

setPageIndex(pageIndex);

setPageSize(pageSize);

};

const ArraySort = (index, size) => {

const pageStart = size * index;

const pageEnd = pageStart + size;

const result = [];

for (let i = pageStart; i < pageEnd; i++) {

  if (i < totalItem) {

    result.push(scanResult[i]);

```

```
    } else {  
  
        break;  
  
    }  
  
}  
  
return result;  
  
};  
  
const pagination = {  
  
    pageIndex,  
  
    pageSize,  
  
    totalItemCount: totalItem,  
  
    pageSizeOptions: [50, 10, 0],  
  
    showPerPageOptions,  
  
};  
  
const column = [  
  
    {  
  
        field: 'address',  
  
        name: 'IP Address'  
  
    },  
  
    {  
  
        field: 'response',  
  
        name: 'Response',  
  
    },  
  
];
```

```

    }

];

if(!scanResult){

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Ping Sweep Result</h1>

                        </EuiText>

                    </EuiPageContentHeader>

                    <EuiLoadingSpinner size="xl" />

                </EuiPageContent>

            </EuiPage>

        </EuiProvider>

    )

}

return(

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

```

```

<EuiPageBody>

  <EuiPageContent>

    <EuiPageContentHeader>

      <EuiText>

        <h1>Ping Sweep Result</h1>

      </EuiText>

    </EuiPageContentHeader>

    <EuiPageContentBody>

      <EuiBasicTable

        items={ ArraySort(pageIndex, pageSize) }

        columns={ column }

        pagination={ pagination }

        onChange={ onTableChange }

      />

    </EuiPageContentBody>

  </EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```


Fast scan result js file:

```
/*
```

```
Project name: Security Scanner
```

```
File: fast.js
```

```
developer: Anita
```

```
*/
```

```
import React, {useState, useEffect} from 'react';
```

```
import {useParams} from "react-router-dom";
```

```
import {
```

```
    EuiAccordion, EuiBasicTable, EuiButton, EuiCard, EuiFieldText, EuiLoadingSpinner,
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiPageContent, EuiPageContentBody,
```

```
    EuiPageContentHeader,
```

```
    EuiPanel,
```

```
    EuiProvider, EuiSpacer,
```

```
    EuiText, useEuiTheme
```

```
} from "@elastic/eui";
```

```
export const Fast = () => {
```

```
    const obj = useParams();
```

```

const [scanResult, setScanResult] = useState();

const euiTheme = useEuiTheme();

const [portData, setPortData] = useState();

useEffect(() => {

  fetch('http://localhost:4000/report?id='+ obj.id)

    .then((response) => response.json())

    .then((result) => {

      const ip = Object.keys(result);

      if(result[ip].hasOwnProperty('tcp')) {

        const tcp = result[ip].tcp;

        const port_keys = Object.keys(tcp);

        const port_length = port_keys.length;

        let ports = [];

        for (let i = 0; i < port_length; i++) {

          const port = port_keys[i];

          ports.push({

            port: port_keys[i],

            state: tcp[port].state,

            reason: tcp[port].reason,

            name: tcp[port].name,

            conf: tcp[port].conf,

```

```

        })

    }

    setPortData(ports);

}

setScanResult(result);

})

.catch((error) => {

    console.error('Error:', error);

});

}, [])

const columns = [

    {

        field: 'port',

        name: 'Port'

    },

    {

        field: 'state',

        name: 'State'

    },

    {

        field: 'reason',

```

```

        name: 'Reason'

    },

    {

        field: 'name',

        name: 'Name'

    },

    {

        field: 'conf',

        name: 'Conf'

    },

];

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    let keys: string[]= Object.keys(address);

    return(

        <>

        <EuiAccordion id="address" buttonContent={"Address"} initialIsOpen={true}>

            <EuiPanel color="subdued">

                {keys.map((key) => (

                    <EuiText>

```

```

        {key} : {scanResult[ip].addresses[key]}

    </EuiText>

    )}

</EuiPanel>

</EuiAccordion>

</>

)

}

if(!scanResult){

    return (

    <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

    <EuiPageContent>

    <EuiPageContentHeader>

    <EuiText>

    <h1>Fast Scan Result</h1>

    </EuiText>

    </EuiPageContentHeader>

    <EuiLoadingSpinner size="xl" />

    </EuiPageContent>

    </EuiPage>

```

```

        </EuiProvider>

    )

}

if (scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Fast Scan Result</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiCard title="" textAlign="left">

                                <EuiText>

                                    <h2>{Object.keys(scanResult)}</h2>

                                </EuiText>

                                <EuiSpacer />

                                {renderAddress()}


```

```

        <EuiSpacer />

        <EuiText>

            <h3>No Open Ports</h3>

        </EuiText>

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Fast Scan Result</h1>

                        </EuiText>

```

```

</EuiPageContentHeader>

<EuiPageContentBody>

  <EuiCard title={""} textAlign={"left"}>

    <EuiText>

      <h2>{Object.keys(scanResult)}</h2>

    </EuiText>

    <EuiSpacer />

    {renderAddress()}

    <EuiSpacer />

    <EuiBasicTable

      items={portData}

      columns={columns}

    />

  </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```


Regular scan result js file:

```
/*
```

```
Project name: Security Scanner
```

```
File: regular.js
```

```
developer: Anita
```

```
*/
```

```
import React, {useState, useEffect} from 'react';
```

```
import {useParams} from "react-router-dom";
```

```
import {
```

```
    EuiPage,
```

```
    EuiProvider,
```

```
    useEuiTheme, EuiPageContent,
```

```
    EuiPageContentHeader,
```

```
    EuiLoadingSpinner,
```

```
    EuiSpacer,
```

```
    EuiText,
```

```
    EuiAccordion,
```

```
    EuiPanel,
```

```
    EuiBasicTable,
```

```
} from "@elastic/eui";
```

```

export const Regular = () => {

  const [scanResult, setScanResult] = useState();

  const euiTheme = useEuiTheme();

  const [portData, setPortData] = useState();

  const obj = useParams();

  useEffect(() => {

    fetch('http://localhost:4000/report?id='+obj.id)

      .then((response) => response.json())

      .then((result) => {

        const ip = Object.keys(result);

        if(result[ip].hasOwnProperty('tcp')) {

          const tcp = result[ip].tcp;

          const port_keys = Object.keys(tcp);

          const port_length = port_keys.length;

          let ports = [];

          for (let i = 0; i < port_length; i++) {

            const port = port_keys[i];

            ports.push({

              port: port_keys[i],

              state: tcp[port].state,

              reason: tcp[port].reason,

```

```

        name: tcp[port].name,

        product: tcp[port].product,

        conf: tcp[port].conf,

        cpe: tcp[port].cpe

    })

}

setPortData(ports);

}

setScanResult(result);

})

}, []

const columns = [

    {

        field: 'port',

        name: 'Port'

    },

    {

        field: 'state',

        name: 'State'

    },

    {

```

```
      field: 'reason',

      name: 'Reason'

    },

    {

      field: 'name',

      name: 'Name'

    },

    {

      field: 'product',

      name: 'Product'

    },

    {

      field: 'conf',

      name: 'Conf'

    },

    {

      field: 'cpe',

      name: 'CPE'

    },

  ];

  const renderAddress = () => {
```

```

const ip= Object.keys(scanResult);

const address = scanResult[ip].addresses;

let keys: string[]= Object.keys(address);

return(

  <>

  <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={ true }>

    <EuiPanel color="subdued">

      {keys.map((key) => (

        <EuiText>

          {key} : {scanResult[ip].addresses[key]}

        </EuiText>

      ))}

    </EuiPanel>

  </EuiAccordion>

  </>

)

}

if(!scanResult){

  return (

    <EuiProvider colorMode={euiTheme.colorMode}>

      <EuiPage>

```

```

        <EuiPageContent>

            <EuiPageContentHeader>

                <EuiText>

                    <h1>Regular Scan Result</h1>

                </EuiText>

            </EuiPageContentHeader>

            <EuiLoadingSpinner size="xl" />

        </EuiPageContent>

    </EuiPage>

</EuiProvider>

)

}

if (scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Regular Scan Result</h1>

                        </EuiText>

```

```

        </EuiPageContentHeader>

        <EuiText>

            <h2>{ Object.keys(scanResult) }</h2>

        </EuiText>

        <EuiSpacer />

        {renderAddress()}

        <EuiSpacer />

        <EuiText>

            <h3>No Open Ports</h3>

        </EuiText>

    </EuiPageContent>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageContent>

                <EuiPageContentHeader>

                    <EuiText>

```

```

        <h1>Regular Scan Result</h1>

        </EuiText>

    </EuiPageContentHeader>

    <EuiText>

        <h2>{ Object.keys(scanResult) }</h2>

    </EuiText>

    <EuiSpacer />

    {renderAddress()}

    <EuiSpacer />

    <EuiBasicTable

        items={portData}

        columns={ columns}

    />

</EuiPageContent>

</EuiPage>

</EuiProvider>

)

}

```

Extensive scan result js file:

```
/*
```

Project name: Security Scanner

File: extensive.js

developer: Anita

*/

```
import React, {useState, useEffect} from 'react';
```

```
import {useParams} from "react-router-dom";
```

```
import {
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiProvider,
```

```
    useEuiTheme,
```

```
    EuiPageContentHeader,
```

```
    EuiText,
```

```
    EuiPageContentBody,
```

```
    EuiFieldText,
```

```
    EuiSpacer,
```

```
    EuiButton,
```

```
    EuiPageContent, EuiAccordion, EuiPanel, EuiCard, EuiBasicTable, EuiLoadingSpinner
```

```
} from "@elastic/eui";
```

```
export const Extensive = () => {
```

```
    const obj = useParams();
```

```

const euiTheme = useEuiTheme();

const [scanResult, setScanResult] = useState();

const [portData, setPortData] = useState();

const [osData, setOsData] = useState();

useEffect(() => {

  fetch('http://localhost:4000/report?id='+ obj.id)

    .then((response) => response.json())

    .then((result) => {

      const ip = Object.keys(result);

      if(result[ip].hasOwnProperty('tcp')) {

        const tcp = result[ip].tcp;

        const port_keys = Object.keys(tcp);

        const port_length = port_keys.length;

        let ports = [];

        for (let i = 0; i < port_length; i++) {

          const port = port_keys[i];

          ports.push({

            port: port_keys[i],

            state: tcp[port].state,

            reason: tcp[port].reason,

            name: tcp[port].name,

```

```

        product: tcp[port].product,

        version: tcp[port].version,

        extrainfo: tcp[port].extrainfo,

        conf: tcp[port].conf,

        cpe: tcp[port].cpe,

        title: tcp[port].script['http-title'],

        serverheader: tcp[port].script['http-server-header']

    })

}

setPortData(ports);

}

const osmatch = result[ip].osmatch;

const os_length = osmatch.length;

let os_result = [];

for (let i = 0; i < os_length; i++) {

    const os = osmatch[i];

    os_result.push({

        name: os.name,

        accuracy: os.accuracy,

        vendor: os.osclass[0].vendor,

        osfamily: os.osclass[0].osfamily,

```

```

        })

    }

    setOsData(os_result);

    setScanResult(result);

    })

    .catch((error) => {

        console.error('Error:', error);

    });

}, [])

const columns_os = [

    {

        field: 'name',

        name: 'Name'

    },

    {

        field: 'accuracy',

        name: 'Accuracy'

    },

    {

        field: 'vendor',

        name: 'Vendor'

```

```
    },  
  
    {  
  
      field: 'osfamily',  
  
      name: 'Os Family'  
  
    },  
  
  ],  
  
  const columns_ports = [  
  
    {  
  
      field: 'port',  
  
      name: 'Port'  
  
    },  
  
    {  
  
      field: 'state',  
  
      name: 'State'  
  
    },  
  
    {  
  
      field: 'reason',  
  
      name: 'Reason'  
  
    },  
  
    {  
  
      field: 'name',
```

```
    name: 'Name'

  },

  {

    field: 'product',

    name: 'Product'

  },

  {

    field: 'version',

    name: 'Version'

  },

  {

    field: 'extrainfo',

    name: 'Extra Information'

  },

  {

    field: 'conf',

    name: 'Conf'

  },

  {

    field: 'cpe',

    name: 'CPE'

  }
```

```

    },

    {

        field: 'title',

        name: 'HTTP Title'

    },

    {

        field: 'serverheader',

        name: 'HTTP Server Header'

    },

    ],

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    const vendorKey= Object.keys(scanResult[ip].vendor);

    let keys: string[]= Object.keys(address);

    return(

        <>

        <EuiAccordion id="address" buttonContent={"Address"} initialIsOpen={true}>

            <EuiPanel color="subdued">

                {keys.map((key) => (

                    <EuiText>

```

```

        {key} : {scanResult[ip].addresses[key]}

    </EuiText>

    ))}

    mac-vendor: {scanResult[ip].vendor[vendorKey]}

</EuiPanel>

</EuiAccordion>

</>

)

}

if(!scanResult){

    return (

    <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

    <EuiPageContent>

    <EuiPageContentHeader>

    <EuiText>

    <h1>Extensive Scan Result</h1>

    </EuiText>

    </EuiPageContentHeader>

    <EuiLoadingSpinner size="xl" />

    </EuiPageContent>

```



```

        </EuiPage>

    </EuiProvider>

)

}

if(scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageBody>

                    <EuiPageContent>

                        <EuiPageContentHeader>

                            <EuiText>

                                <h1>Extensive Scan Result</h1>

                            </EuiText>

                        </EuiPageContentHeader>

                        <EuiPageContentBody>

                            <EuiCard title="" textAlign="left">

                                <EuiText>

                                    <h2>{Object.keys(scanResult)}</h2>

                                </EuiText>

                                <EuiSpacer />

```

```

        {renderAddress()}

        <EuiSpacer />

        <EuiText>

            <h3>Port Information</h3>

            <h4>No Open Ports</h4>

        </EuiText>

        <EuiSpacer />

        <EuiText>

            <h3>OS Information</h3>

            <h4>These results may not be 100% accurate</h4>

        </EuiText>

        <EuiBasicTable

            items={osData}

            columns={columns_os}

        />

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

```

```

    )
}

return (

  <EuiProvider colorMode={euiTheme.colorMode}>

    <EuiPage>

      <EuiPageBody>

        <EuiPageContent>

          <EuiPageContentHeader>

            <EuiText>

              <h1>Extensive Scan Result</h1>

            </EuiText>

          </EuiPageContentHeader>

          <EuiPageContentBody>

            <EuiCard title="" textAlign="left">

              <EuiText>

                <h2>{Object.keys(scanResult)}</h2>

              </EuiText>

              <EuiSpacer />

              {renderAddress()}

              <EuiSpacer />

              <EuiText>

```

```

        <h3>Port Information</h3>

        </EuiText>

        <EuiBasicTable

            items={portData}

            columns={columns_ports}

        />

        <EuiText>

            <h3>OS Information</h3>

            <h4>These results may not be 100% accurate</h4>

        </EuiText>

        <EuiBasicTable

            items={osData}

            columns={columns_os}

        />

    </EuiCard>

</EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

```

```
}
```

Vulnerability scan result js file:

```
/*
```

```
Project name: Security Scanner
```

```
File: vulnerability.js
```

```
developer: Anita
```

```
*/
```

```
import React, {useState, useEffect, Fragment} from 'react';
```

```
import {useParams} from "react-router-dom";
```

```
import {
```

```
    EuiPage,
```

```
    EuiPageBody,
```

```
    EuiProvider,
```

```
    useEuiTheme,
```

```
    EuiPageContentHeader,
```

```
    EuiText,
```

```
    EuiPageContentBody,
```

```
    EuiFieldText,
```

```
    EuiSpacer,
```

```
    EuiButton,
```

```
    EuiPageContent, EuiAccordion, EuiPanel, EuiCard, EuiBasicTable, EuiLoadingSpinner, EuiLink
```

```
} from "@elastic/eui";
```

```
export const Vulnerability = () => {

  const obj = useParams();

  const euiTheme = useEuiTheme();

  const [scanResult, setScanResult] = useState();

  const [portData, setPortData] = useState();

  const [osData, setOsData] = useState();

  useEffect(() => {

    fetch('http://localhost:4000/report?id='+ obj.id)

      .then((response) => response.json())

      .then((result) => {

        const ip = Object.keys(result);

        const osmatch = result[ip].osmatch;

        const os_length = osmatch.length;

        let os_result = [];

        for (let i =0; i<os_length; i++) {

          const os = osmatch[i];

          os_result.push({

            name: os.name,

            accuracy: os.accuracy,
```

```

        vendor: os.osclass[0].vendor,

        osfamily: os.osclass[0].osfamily,

    })

}

if(result[ip].hasOwnProperty('tcp')) {

    const tcp = result[ip].tcp;

    const port_keys = Object.keys(tcp);

    const port_length = port_keys.length;

    let ports = [];

    for (let i = 0; i < port_length; i++) {

        const port = port_keys[i];

        ports.push({

            port: port_keys[i],

            state: tcp[port].state,

            reason: tcp[port].reason,

            name: tcp[port].name,

            product: tcp[port].product,

            version: tcp[port].version,

            extrainfo: tcp[port].extrainfo,

            conf: tcp[port].conf,

            cpe: tcp[port].cpe,

```

```

        serverheader: tcp[port].script['http-server-header']

    })

}

setPortData(ports);

}

setOsData(os_result);

setScanResult(result);

})

}, [])

const columns_os = [

    {

        field: 'name',

        name: 'Name'

    },

    {

        field: 'accuracy',

        name: 'Accuracy'

    },

    {

        field: 'vendor',

        name: 'Vendor'

    }

]

```



```
    },  
  
    {  
  
      field: 'osfamily',  
  
      name: 'Os Family'  
  
    },  
  
  ],  
  
  const columns_ports = [  
  
    {  
  
      field: 'port',  
  
      name: 'Port'  
  
    },  
  
    {  
  
      field: 'state',  
  
      name: 'State'  
  
    },  
  
    {  
  
      field: 'reason',  
  
      name: 'Reason'  
  
    },  
  
    {  
  
      field: 'name',
```

```
    name: 'Name'

  },

  {

    field: 'product',

    name: 'Product'

  },

  {

    field: 'version',

    name: 'Version'

  },

  {

    field: 'extrainfo',

    name: 'Extra Information'

  },

  {

    field: 'conf',

    name: 'Conf'

  },

  {

    field: 'cpe',

    name: 'CPE'

  }
```

```

    },

    {

        field: 'serverheader',

        name: 'HTTP Server Header'

    },

]

const renderAddress = () => {

    const ip= Object.keys(scanResult);

    const address = scanResult[ip].addresses;

    const vendorKey= Object.keys(scanResult[ip].vendor);

    let keys: string[]= Object.keys(address);

    return (

        <>

        <EuiAccordion id="address" buttonContent={ "Address" } initialIsOpen={ true }>

            <EuiPanel color="subdued">

                {keys.map((key) => (

                    <EuiText>

                        {key} : {scanResult[ip].addresses[key]}

                    </EuiText>

                )))}

                mac-vendor: {scanResult[ip].vendor[vendorKey]}

```

```

        </EuiPanel>

        </EuiAccordion>

    </>

)

}

const vulners_list = (port) => {

    let vulner= [];

    const ip = Object.keys(scanResult);

    const tcp = scanResult[ip].tcp;

    if (scanResult[ip].tcp[port].script.vulners) {

        const vulners = scanResult[ip].tcp[port].script.vulners.split('\n');

        const vulner_column = [

            {

                field: 'cve',

                name: 'CVE number',

            },

            {

                field: 'level',

                name: 'Level',

            },

            {

```

```

    field: 'weblink',

    name: 'Link',

    render: (id, item) => (

        <EuiLink href={item.weblink} target="_blank">

            {item.weblink}

        </EuiLink>

    )

}

]

for(let i = 2; i < vulners.length; i++ ){

    const vul = vulners[i].split("\t");

    vulner.push({

        cve: vul[1],

        level: vul[2],

        weblink: vul[3]

    })

}

return (

    <Fragment>

        <EuiSpacer />

        <EuiText>

```

```

        <h4>Vulners Result</h4>

        <h4>Port number: {port} </h4>

    </EuiText>

    <EuiBasicTable

        items={ vulner}

        columns={ vulner_column}

    />

</Fragment>

)

}

return (

    <Fragment>

        <EuiSpacer/>

        <EuiText>

            <h3>Port number: {port}</h3>

            <h4>No result from Vulners Database</h4>

        </EuiText>

    </Fragment>

)

}

const vulscan = (port) => {

```

```

const ip = Object.keys(scanResult);

const tcp = scanResult[ip].tcp;

if (scanResult[ip].tcp[port].script.vulscan) {

  const vulscan = scanResult[ip].tcp[port].script.vulscan.split('\n');

  return(

    <Fragment>

      <EuiText><h3>Vulscan</h3></EuiText>

      <EuiAccordion grow={false} buttonContent={port}>

        <EuiPanel grow={false} color="subdued">

          { vulscan.map((vul) => (

            <EuiText textAlign="left">{ vul}</EuiText>

          ))}

        </EuiPanel>

      </EuiAccordion>

    </Fragment>

  )

}

return (

  <Fragment>

    <EuiSpacer/>

    <EuiText>

```

```
<h3>Port number: {port}</h3>
```

```
<h4>No result from Vulscan Database</h4>
```

```
</EuiText>
```

```
</Fragment>
```

```
)
```

```
}
```

```
const vulnerability = () => {
```

```
  const ip = Object.keys(scanResult);
```

```
  const tcp = scanResult[ip].tcp;
```

```
  const ports = Object.keys(tcp);
```

```
  return(
```

```
    <Fragment>
```

```
      {ports.map((port) => (
```

```
        <>
```

```
          {vulners_list(port)}
```

```
          {vulscan(port)}
```

```
        </>
```

```
      )))
```

```
    </Fragment>
```

```
)
```



```

}

if(!scanResult){

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Vulnerability Scan Result</h1>

                        </EuiText>

                    </EuiPageContentHeader>

                    <EuiLoadingSpinner size="xl" />

                </EuiPageContent>

            </EuiPage>

        </EuiProvider>

    )

}

if(scanResult && !portData) {

    return (

        <EuiProvider colorMode={euiTheme.colorMode}>

            <EuiPage>

```

```

<EuiPageBody>

  <EuiPageContent>

    <EuiPageContentHeader>

      <EuiText>

        <h1>Vulnerability Scan Result</h1>

      </EuiText>

    </EuiPageContentHeader>

    <EuiPageContentBody>

      <EuiText>

        <h2>{ Object.keys(scanResult) }</h2>

      </EuiText>

      <EuiSpacer />

      {renderAddress()}

      <EuiSpacer />

      <EuiText>

        <h3>OS Information</h3>

        <h4>These results may not be 100% accurate</h4>

      </EuiText>

      <EuiBasicTable

        items={osData}

        columns={columns_os}

```

```

        />

        <EuiSpacer />

        <EuiText>

            <h3>Open Ports Information</h3>

            <h4>No Open Ports</h4>

        </EuiText>

    </EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

return (

    <EuiProvider colorMode={euiTheme.colorMode}>

        <EuiPage>

            <EuiPageBody>

                <EuiPageContent>

                    <EuiPageContentHeader>

                        <EuiText>

                            <h1>Vulnerability Scan Result</h1>

```

</EuiText>

</EuiPageContentHeader>

<EuiPageContentBody>

<EuiText>

<h2>{ Object.keys(scanResult)}</h2>

</EuiText>

<EuiSpacer />

{renderAddress() }

<EuiSpacer />

<EuiText>

<h3>OS Information</h3>

<h4>These results may not be 100% accurate</h4>

</EuiText>

<EuiBasicTable

items={osData}

columns={columns_os}

/>

<EuiSpacer />

<EuiText>

<h3>Open Ports Information</h3>

</EuiText>

```

        <EuiBasicTable

            items={portData}

            columns={columns_ports}

        />

        <EuiText>

            <h3>Vulnerabilites</h3>

        </EuiText>

        {vulnerability()}

    </EuiPageContentBody>

</EuiPageContent>

</EuiPageBody>

</EuiPage>

</EuiProvider>

)

}

```