TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Toyib Olamide Ahmed 177782IVSB

Open-Source Penetration Testing Tool For Beginners

Bachelor's thesis

Supervisor: Kaido Kikkas
Ph.D

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Toyib Olamide Ahmed 177782IVSB

Avatud lähtekoodiga läbistustestimisvahend algajaile

bakalaureusetöö

Juhendaja: Kaido Kikkas

Tehnikateaduste doktor

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Toyib Olamide Ahmed

29.04.2020

Abstract

The objective of this thesis is to create an application that's free for anyone to test the vulnerability in their web server and open to anyone willing to improve usage or download the application. This solution will help small businesses that can't afford to pay a professional for the evaluation of their system.

The result of this work is a full-stack web application, where user can input and submit their domain name and email via user interface and get their system tested with the basic phases of penetration testing. The result will then be forwarded to their respective email on the completion of the test. Back-end is written using Node.js while front-end is implemented with React.js.

This thesis is written in English and is 45 pages long, including 5 chapters, 17 figures and 1 table.

Annotatsioon

Avatud lähtekoodiga läbistustestimisvahend algajaile

Küberturbetehnika bakalaureuse programm.

Lõputöö eesmärk on luua tasuta äpp, mida võib kasutada igaüks, kes soovib oma veebiserveri haavatavust testida, lisaks on äpi kood avatud kõigile, kes soovivad selle kasutusmugavust parandada või isiklikuks kasutuseks alla laadida. See veebilahendus võiks aidata väikeettevõtteid, kellel puudub rahaline ressurss, et palgata professionaal oma süsteemi hindama.

Käesoleva lõputöö tulemuseks on full-stack veebiäpp, kuhu kasutaja sisestab kasutajaliidese kaudu domeeninime ja emaili, seejärel testitakse süsteemi läbistustestimise algtasemel. Testi lõppedes saadetakse tulemus sisestatud emailile. Backend‘i kirjutamiseks oli kasutuses Node.js ja frontend‘i jaoks React.js.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 45 leheküljel, 5 peatükki, 17 joonist ja 1 tabelit.

List of abbreviations and terms

Table of Contents

List of figures

List of tables

1 Introduction

This thesis is focusing on penetration testing and building a simple penetration tool for beginners. A penetration test is a procedure that is used to assess the security of a computer system by acting as an attacker who's trying to gain access into the system. The result of the test always shows if the system is vulnerable to an attack or not. This thesis will focus on building an application to implement network-based penetration testing, as it is one of the most common types of penetration testing. The main reason for choosing network-based penetration testing is because the testing that will be performed in the app involves several repetitive tasks that can be performed remotely via a network connection, so for this reason it is necessary to automate them.

The purpose of penetration testing automation is to reduce the costs in terms of and people needed to perform the test. However, there are also some disadvantages of penetration testing automation like the generation of false positives, limited pivoting, less analysis of sensitive data and stability issues.

1.1 Problem statement

There is already a number of tools that can be used to perform web automated penetration testing and some of these tools will be described in section 2.4. These tools perform well and minimize the amount of work needed to perform a penetration test, but these user-friendly web tools are often monetized and usually give little to no information if you are not paying to use the tool. An undeniable question is "Why are there not enough open-source web penetration testing tools for beginners?". The answer is most companies or institutions behind these web applications are doing it solely for the purpose of monetizing them.

Exploiting is a vulnerability that can often cause a system or service to crash or fail to perform its legitimate purpose. This makes penetration testing a difficult practice since the tests are usually performed in a production environment. The stability and integrity of the target system are very important in almost every situation and one cannot simply accept that the tools may cause the system to stop operating. Therefore, penetration testers mostly prefer manual testing so that the tester has all control of the testing process and can maintain and assure that only safe techniques are used.

1.2 Goals of the Thesis

The main goal of this thesis is to build a web application that's completely open-source and suitable for non-tech personnel to use in order to check the security flaws of a domain or a system.

This new tool will evaluate systems with a non-aggressive approach and should be suitable to use in production environment, should automate everything that can be done safely without risk of service interruption. The new tool will only focus on implementing the first three steps of penetration testing which will be explained in section 2.2. The tool would serve as an initial step in the testing process by eliminating repetitive testing and dependencies. The system under test should be exploited to extend the traded off area and to securely get extra valuable data.

Additionally, the tool should provide the user with a descriptive report of the testing results and this report can be download from the webpage on the completion of the test or sent to the user email. The choice will be made by the user on how to they want to receive their system testing report.

Furthermore, the user will not be able to influence or specify what kind of test they want to run. The test will be conducted based on the set of rules implemented by the author of this project, but the fact that the project is open source allows people with technical knowledge to manipulate, add or remove from the series of testing that has been originally specified.

1.3 Structure of the Thesis

Chapter 1 will be describing the problem and the specific goal of this. Chapter 2 provides the background necessary to understand the problem and the specific knowledge that the reader will need to understand the rest of this thesis. The standard penetration testing process is explained, and some common tools are briefly described. Chapter 3 elaborates on the methodology and explains the steps followed during this thesis work. A description of the web application built is based on research and observation. Chapter 4 talks about the initial consideration, approach and application scenario of this thesis work. In chapter

5, the design of the architecture of the new web penetration testing tool is presented and the main aspect deriving from the implementation of the new tool. Chapter 6 describes the current state of the implemented application, as well as the testing of the new tool and the result of these tests. Finally, chapter 7 reports the conclusion of this thesis project, and suggests possible future improvements and extensions to the new tool.

2 Background

This chapter gives an overview of the primary component to fully understand the rest of the thesis. Section 2.1 presents the primary inspiration driving the choice of performing a penetration test. In section 2.2 the standard penetration testing process is depicted. Understanding this part is fundamental for the development of a tool to automate the process. Several tools, utilities, and frameworks are then introduced. Some are part of the common toolkit of a penetration tester while others (section 2.4) will be analyzing the process of automated penetration testing.

2.1 Main concepts of penetration testing

There are a few reasons why an association/organization should procure a security expert to perform a penetration test. The fundamental explanation is that security breaches can do incredibly exorbitant damage. An effective attack may lead to direct financial losses, hurt the association, trigger fines and so on. With a proper penetration test, it is possible to identify security vulnerabilities and afterwards take countermeasures before a genuine assault happens.

Another reason behind performing penetration testing is that it very well may be a driving function to make the system operator keep the system up to date with respect to the most recent vulnerabilities. New bugs and security issues are often found. An association may utilize intermittent penetration testing to maintain an updated security level.

The aftereffect of a penetration test encourages an association to organize their risks. An explicit security break creates specific harm to the association. Depending on the seriousness of the issues that are identified, it is conceivable to fittingly design a relief procedure with a stronger focus on more critical issues.

Since penetration testing is just a simulation of a real attack, it can also help in assessing the readiness of the organization technical employees in such situations. For example, if the security expert will be able to infiltrate the system without anyone noticing, it is a good sign that more awareness should be put on security and incident handling.

2.2 The penetration testing process

The purpose of penetration testing is to assess the degree of exposure of the system under test and to decide if any approaches to break into the system exist. In request to properly perform an important and authentic test, a couple of activities should be performed in addition to the actual testing phase as described in this chapter. The process of an expert penetration test can be divided into four fundamental stages: commencement, planning, testing, and reporting [3].

2.2.1 Commencement

The commencement or initiation phase includes an underlying conversation with the client (the owner of the system to be tested) planned for setting up an agreement with the penetration tester. Both parties define and set the scope of the test, the individual responsible for each task and the activities that the testers are permitted to take. A team is also set up in case of emergencies like server shutting down or unforeseen circumstances.

2.2.2 Planning

Before the commencement of penetration testing, both parties need to establish an agreement during the commencement phase. If they're more than one tester involved in the testing process, then the processed is usually organized and shared between the team. Tools to be used for the testing process often depends on the task that needs to be executed. This phase allows the testers to put into consideration the system they are testing by considering the integrity and stability of the system.

2.2.3 Testing

This stage is where the actual testing is carried out. All actions taken during this stage must be logged so that there are possibilities to check back if anything goes wrong. The testing phase is one of the most important parts of penetration testing and a lot of steps are taken to achieve the set goals. The steps to follow will be analyzed carefully in the subsections below.

2.2.3.1 Target identification

Target identification is information gathering of the system currently tested and its basic information like available domain, IP addresses, active services, open ports, security

policy etc. The importance of target identification depends on the information that was available to the testers at the beginning of the test. Identifying the target is a crucial step, particularly with regards to an external test, i.e. the tester has no internal resources. Useful information can only be found out using other types of techniques such as information gathering through search engines, probing the website, or performing social engineering [2].

2.2.3.2 Port scanning

Port scanning is the part of penetration testing process that includes establishing a connection with the system under test. It comprises of examining the system to discover which hosts are available, what ports are open and what types of services are running. A tool is always used for this process and one of the most popular tools for this process is described in section 2.3.2.

2.2.3.3 Enumeration

Enumeration is the process of gathering additional information based on the result of port scanning i.e. analyzing the services detected in the previous step to find out which of the services is vulnerable and can be easily used as an entry into the system. This step requires some previous experience, but there are always tools that can be handy for this process.

2.2.3.4 Penetration

According to the definition of penetration it is the act of exploiting weaknesses that have been identified in the system under test. Based on the result of the previous step, the phase focuses on the exploitation part. As stated in [3] an exploit means a tester or an attacker takes advantage of a security flaw within the system, resulting in the behavior that the developers did not intend.

2.2.3.5 Escalation

The escalation process consists of further exploiting the vulnerable service to increment the impact of the tester on the compromised machine. For instance, when a vulnerability is successfully exploited, the access gained is always limited (e.g. ordinary user privileges), but to have a real impact on the system the superuser privilege is needed to exploit the system even further.

2.2.3.6 Getting interactive

The fact that a host in the system under testing is compromised doesn't mean that the system can be easily controlled. An interactive component is needed for the tester to analyze and give commands on the system the same way a system administrator does. But in most cases exploits automatically provide the tester with an interactive interface such as shell to remotely control the system but in case this doesn't happen automatically, an additional phase to gain interactive access is needed so the tester can control and manipulate the system in whatever way they want.

2.2.3.7 Pillaging

This phase is mostly skipped if the above processes went well and full access is gained, but in case of limited access of the system under test, the stage at hand is needed to get more information about comprised resources. The goal of this phase is to gather more information without the need to exploit them. For example, tester analyzes firewalls or extracts credentials from the database and tries to decrypt the hash values.

2.2.3.8 Clean up

The purpose of this stage is to clean up all the software, commands or script used in exploiting the system, this is done to avoid additional vulnerabilities and the goal of the phase is completely different from a hacker's perspective. A hacker is only concerned about not leaving any trace in the system so that the administrator will not notice any changes that would lead to any suspicions, but the hacker might leave a backdoor in the system (a mechanism to be later used to get into the system without having to exploit the system again).

2.2.4 Reporting

The final phase of a penetration test is to report the consequences of the test. These reports include all the vulnerabilities that were experienced during the test, how it's possible to exploit them and recommendations on how they could be fixed. The clients often don't appreciate reports, so its recommended to explain by financially stating how much they could lose if those kinds of attacks were performed by a professional hacker. This way the client gets to understand how important the vulnerability issue is and what the consequences will be if it's not fixed.

2.3 Tools for penetration testing

This section explains some of the most common tools for penetration testing used by security professionals. These tools have their specific tasks and although they can perform the test on their own once automated, they are still not considered to be automated tools because they still need influence from a tester.

2.3.1 Metasploit Framework

Metasploit [3] [4] is an exploitation framework which provides several tools, scripts and utilities to develop and execute exploits against the targeted machine, referencing the penetration testing process in section 2.2, Metasploit is often used for Penetration, Escalation and Getting interactive. There are also some other tools included in the framework to perform a more specific attack.

The Metasploit Project is an open-source computer security project that gives information about the security vulnerability. This tool is developed by the Massachusetts-based security company called Rapid7 and it is best known for executing exploits against a remote target machine using anti-forensic and evasion tools.

Before using Metasploit to exploit a remote target, basic information of the target should be available in advance. Most penetration testers use tools like Nmap since it doesn't only help collect the information, but also identify a potential vulnerability in the target machine. Metasploit has a large number of exploits for different applications and operating systems that can be accessed once a vulnerability is spotted. It is up to the tester to manually choose what kind of attack to launch.

2.3.2 Nmap

Nmap [5] [6] is also a free and open-source network scanner created by Gordon Lyon to discover hosts and services on a computer network by sending packets and analyzing the response. This tool can also be used for network monitoring and inventory.

Nmap injects uniquely created parcels as system traffic and by breaking down the reactions to these packets, it derives several pieces of data about the system. For example, what hosts are available, what administration is running on the machine, the operating system installed, what firewalls are in use, etc. Nmap is an incredible utility that gives the

client extraordinary adaptability with more than 100 command-line options. Below is the following process that takes place during a normal scanning process:

1. Pre-scanning
2. Target enumeration
3. Host discovery
4. Reverse-DNS resolution
5. Port scanning
6. Version detection
7. Operating system detection
8. Traceroute
9. Script scanning
10. Output
11. Post-scanning

Some of this process will be used later in this thesis to gain knowledge about the target.

2.3.3 Wireshark

Wireshark [7] [8] is a free and open-source packet analyzer tool; its main usage is network troubleshooting and analyzing. The project was originally named Ethereal but was renamed in May 2006 due to trademark issues. Wireshark is a cross-platform tool and using QT widget toolkit in its current releases to implement its user interface and using pcap (packet capture) to capture packets.

2.3.4 Burp Suite

Burp Suite [19] is a Java application designed for a major reason: to perform security testing on web applications. Burp Suite consists of different components that together constitute a platform for web application assessment. Below is the list of Burp Suite components:

1. Burp Proxy
2. Burp Spider
3. Burp Scanner
4. Burp Intruder
5. Burp Repeater

2.4 Related Work

This section presents a few related works to this thesis (web-based open-source penetration testing tools). These existing tools will give a basic understanding of this thesis work and also the limitations.

2.4.1 Penetration Testing of Web Applications in a Bug Bounty Program

This thesis work focused on using the readily available penetration testing too called the Bugcrowd, since web application security cannot be guaranteed it is always recommended to used penetration testing method to evaluate new applications. The author used the software to test different web applications and compare the results with the statistics provided by other penetration testing companies and using that to determine the average web application security level [10].

2.4.2 Penetration Testing on Domain Name Service

This thesis work is explaining and simulating the risk associated with threats experienced by domain name service (DNS) inside a private system or web. The author used a virtual environment (VMware workstation) and a kali Linux operating system to achieve this.

The goal of the project is to analyze and process the vulnerabilities associated with DNS through penetration testing and try to mitigate the damage or eliminate any possible risks [11].

2.4.3 Summary of related work

Studying the behavior of the research's described in this section, a typical way to deal with penetration testing emerged. The method followed by these tools comprises of three primary stages:

1. Scan the hosts in the system under evaluation to gather information about the target.
2. Identify vulnerabilities by comparing scan result with vulnerabilities database.
3. Exploit the vulnerability to gain access into the system.

Depending on the tool the tester decides to use, other steps might be needed, however the basic steps for any successful penetration testing are always the three steps mentioned above.

3 Methodology

In this chapter author describe the methodology of the web application, technologies, framework and services used in achieving this thesis work.

3.1 Single Page Application (SPA)

A single page application is a web application that dynamically rewrites the current page with new data from the web server based on the client interaction, instead of the default method of the browser loading the entire new pages [12].

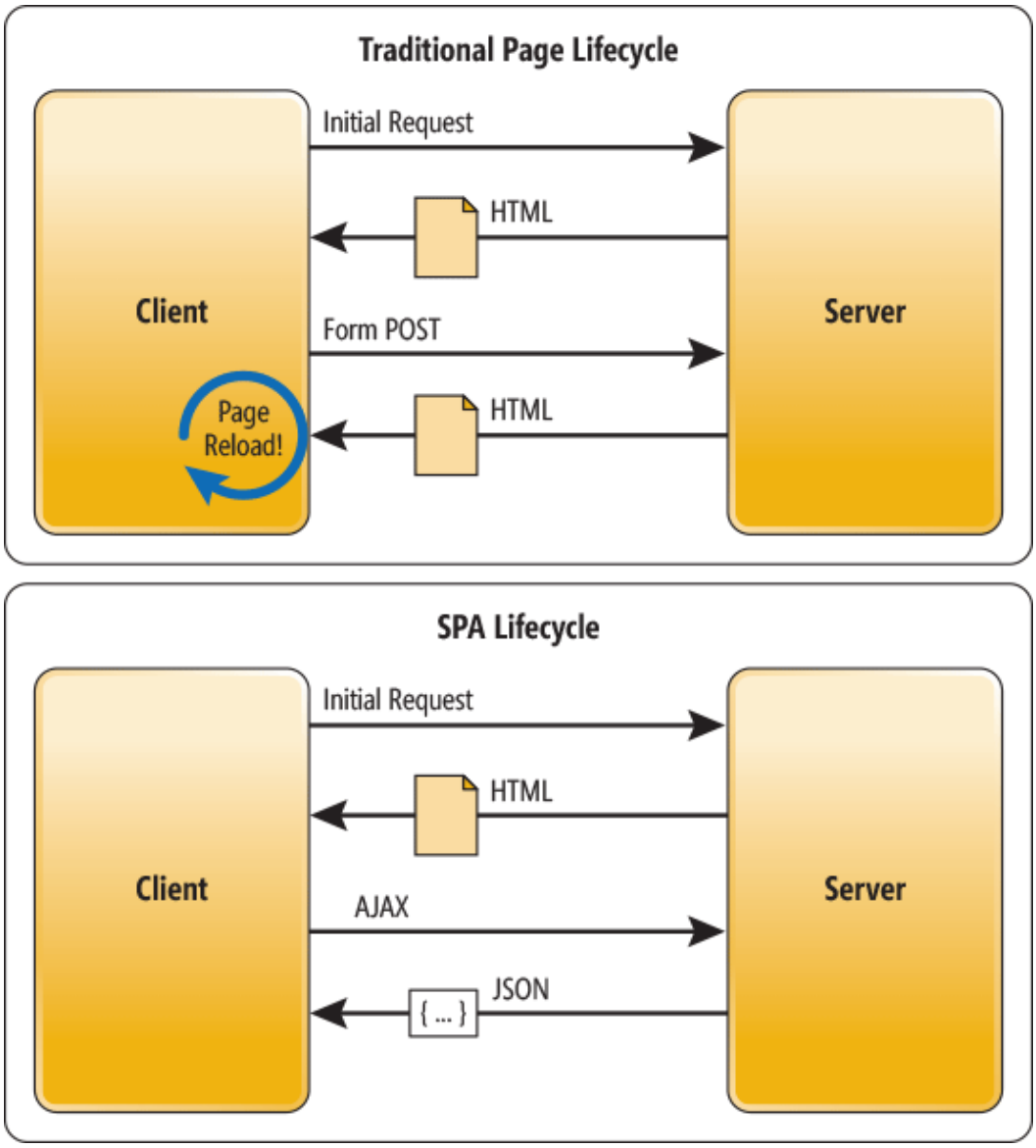

Figure 1 Traditional page and SPA lifecycles [28].

This approach is possible due to JavaScript ability to manipulate DOM elements. It works in such a way that the client send request to the server and the server respond with just an HTML page for the first time and all subsequent request with the server are being made via AJAX request to fetch content and update the page upon client's interaction. This approach is not possible when it come to a multi-page application, the server will respond with a whole new page whenever there's an update (Figure 1). In addition, the SPA history API allow a smooth navigation process without the need for reloading the page, this means that the URL might change a bit due to the route but there will be no page reload to, but instead just replace the content of the page based on the application architecture.

3.2 Model-View-Controller (MVC)

Model-View-Controller is a software design pattern ordinarily utilized for creating UIs which partitions the related program rationale into three interconnected components. This is done to separate internal representations of information from the way's information is presented to and accepted from the users [13].

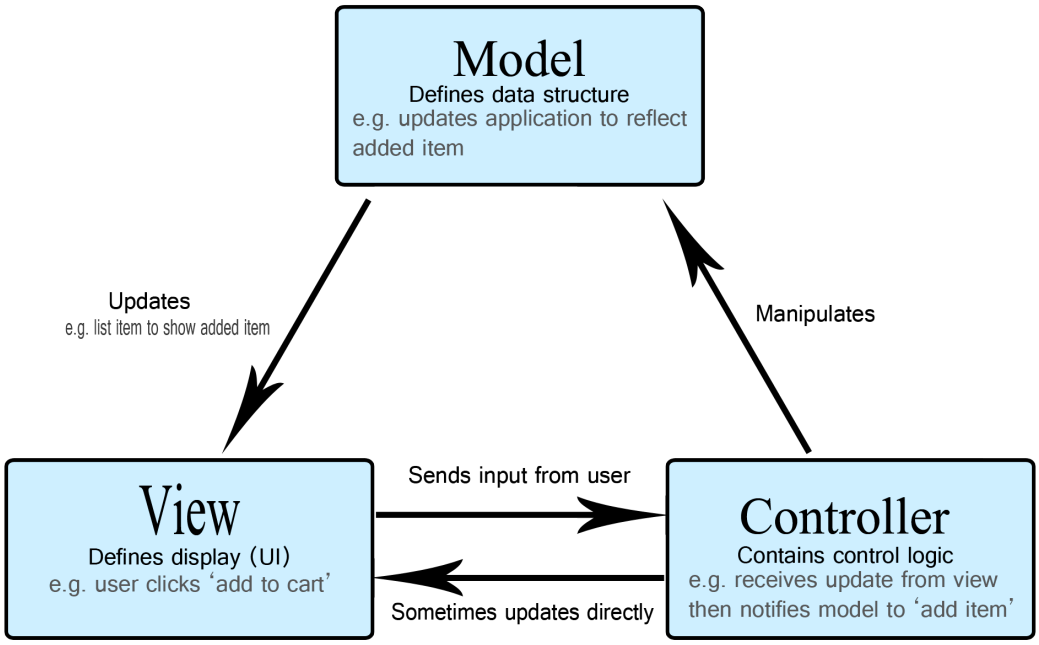

Figure 2 The MVC Pattern [29].

The MVC Pattern is used for applications with GUI. The Model represents the application object while the View represents that object information and the Controller specify the method in which the user interface reacts with the user input.

3.3 Model – View – View-Model (MVVM)

MVVM is another type of software architectural pattern that facilitates the separation of the development of the view from the development of the backend logic (model) just to make sure the view is not depending on any specific model platform. The VM (view model) of the MVVM is a value converter i.e. its responsible for changing over the data objects from the model so that items are effectively overseen and introduced [14].

This model is more common nowadays when it comes to using JavaScript library or framework for frontend because developers don't want to do too much input control anymore like it would be when using the MVC pattern. Below is a diagrammatic representation of MVVM (Figure 3).

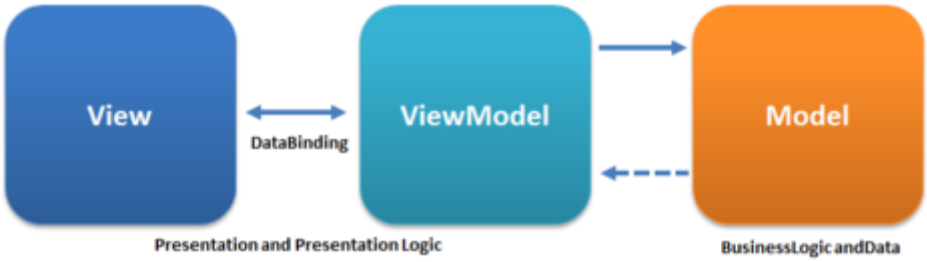

Figure 3 The MVVM pattern [26]

3.4 Frontend with React.js

React is a JavaScript library for building user interfaces. Its maintained by Facebook and a community of individual / companies [15].

React is mostly used as a base for developing a single page application but its only concerned with rendering the DOM i.e. react is focused on the MV layer in the MVVM pattern because it only connects Model and View through the two-way data bindings.

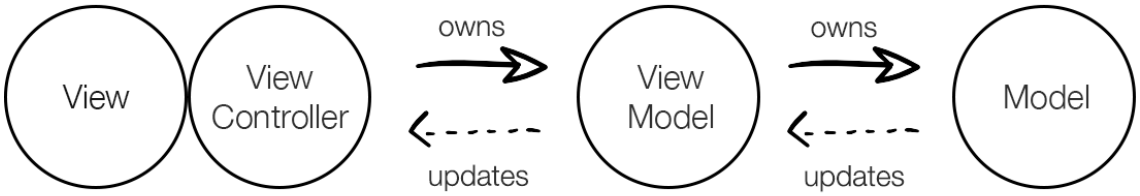

Figure 4 The MVVM pattern with React.js [27]

React manages the data in the HTML element because of its reactiveness and this helps to automatically re-render elements whenever there's any changes [16].

3.4.1 Additional used libraries

- React-bootstrap is a front-end CSS library for React.js to build responsive web design
- Axios is a popular JavaScript library for performing HTTP request that works in both browser and Node.
- Mdbreact is a material design CSS library to enhance web design
- Formik is a JavaScript used to handle form input, author prefer using this than just regular input because it helps to keep track of values, errors and visited fields.

3.5 Backend with Node.js

Node.js is an open-source, cross-platform, JavaScript runtime built on Chrome's V8 JavaScript engine that executes JavaScript code outside of a web browser [17].

Node.js allows developer to use JavaScript to write server-side script to produce a dynamic web page. This represent a "JavaScript everywhere" paradigm which is basically about use a single programming language to build dynamic web application. Node.js also have an event-driven architecture that allows it to perform asynchronous operations [17].

Author choice of this solution was based on the possibilities to use node modules instead on running the whole command on Kali Linux, this module provides the possibilities to

do network scan, dns lookup, etc. And also, the desire to use the same programming language is also considered.

3.5.1 Additional used modules

- Express.js is a framework in Node.js which is use for building web applications and APIs. It's an essential part of the server because it helps to handle all kinds of HTTP requests and also with the implementation of APIs [18].

- CORS is a cross-origin resource sharing module that allows AJAX request to ignore the same-origin policy and allow connections from other remote hosts. Without enabling this module, author will not be able to put frontend and backend on different host [19].

- Mongoose is an ODM module for MongoDB, it helps to facilitate the relationship between data and also provides schema validation. Author use this module to design the database model for the Application [20].

- Nodemailer is a node module that helps to manage email sending. In this Application it is used to send test result to client email after the completion of the testing face

3.5.1.1 Penetration testing modules

- Node-nmap a node module for working with NMAP in Node.js, more explanation about nmap can be found at section 2.3.2 [23].

- DNS a node module uses to get information about a particular domain, for example it gets domain IP address, address and also family depending on the code [24].

3.6 Database with MongoDB

MongoDB is a cross-platform data-oriented database classified as NoSQL database, it uses JSON-like documents with schema and it stores documents in collections, it supports the major SQL queries expression [21].

Data coming from the frontend of the App is in JSON object that's why a NoSQL database like MongoDB suites this project development.

3.7 Heroku Cloud Application Platform

Heroku is a container-based cloud Platform which developers use to deploy, oversee and scale present day application. Heroku is a very big and popular platform that is well documented, easy and to some extent free to use [25].

Heroku is completely overseen, giving developers the opportunity to concentrate on their core project without the interruption of looking after servers [25].

Both client and server side are deployed on Heroku to share application between course-mates and also for demonstration purpose. For simplicity and easy to manage application author have chosen to put the frontend and the backend on different server, the frontend (client) is available at https://pen-testing-frontend.herokuapp.com/ while the backend (server) is available at https://pen-testing-backend.herokuapp.com/.

4 Implementation

Application is constantly developed with the visual studio code editor and source code deployed to GitHub repository available at https://github.com/Tobzzy/Thesis. How to run both server and client locally is carefully explained in the readme.md file upload alongside all other files.

4.1 Service architecture

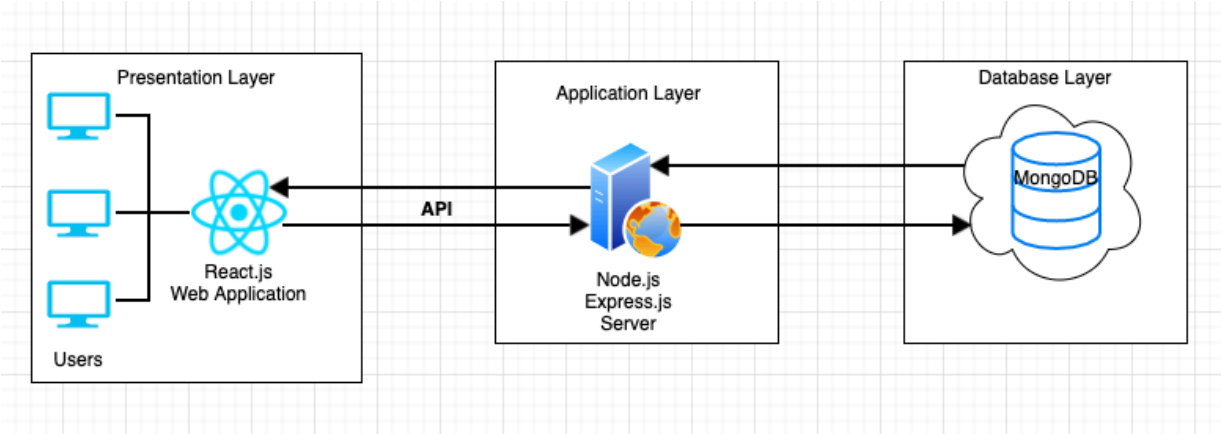

Figure 5 Client-Server Architecture of the Service

It has a presentation layer of 3 client machine, that are requesting web application via browser using the user interface. The application layer is the server that processes and respond to the client data. Server is able to grab the data being passed from the client through the API and store the data in the database represented as the database layer. All communication between client and server is done via HTTP connection.

Client-side web application and Server are both hosted on Heroku and for easier accessibility, explanation about Heroku and link to web application can be found at section 3.7.

4.2 Server-side

Backend is written using Node.js.

Node.js uses package.json file is know as the core of Node.js ecosystem because it's the most basic and fundamental part of the working with Node.js, the package.json file can be called the manifest of the application as it handles modules, packages and etc.

package.json > ...

```
{
  "name": "pentesting-app-backend",
  "version": "1.0.0",
  "description": "Backend for my Thesis project",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Express",
    "RestAPI",
    "MongoDB",
    "Mongoose",
    "PenTesting"
  ],
  "author": "Toyib Olamide Ahmed",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "dns-lookup": "^0.1.0",
    "express": "^4.17.1",
    "mongoose": "^5.9.5",
    "node-nmap": "^4.0.0",
    "nodemailer": "^6.4.6",
    "nodemailer-express-handlebars": "^4.0.0"
  }
}
```

Figure 6 The package.json file

As shown in the figure 6. The package.json file helps to facilitates the whole of the Node.js server, from specifying the name of the application, version of the software, the description of the application, entry scripts, dependencies and etc.

4.2.1 API

The communication between the server and the client is built on HTTP, API was implemented on the server so as to get the endpoints to be passed to the client side of the application, table 1 shows the endpoints defined. Express.js a Node module was used to help configure the server and also manage the route. GET method are used where the data

is only retrieved, POST method are used for creating new data, PUT method is used for updating existing data and DELETE is used for removing existing data from the database.

URL route	Request method	Description
/tests	GET	Responds with the list of all existing test run
/tests/:testId	GET	Responds with the details of the particular test with parameter (testId)
/tests	POST	Create new test
/tests/:testId	PUT	Update an existing test by specifying with parameter (testId)
/tests/:testId	DELETE	Delete an existing test by specifying with parameter (testId)

Table 1 API endpoints

app > routes > JS test.routes.js > <unknown> > module.exports

```
1   module.exports = (app) => {
2       const tests = require('../controllers/test.controller.js');
3
4       app.post('/tests', tests.create);
5
6       app.get('/tests', tests.findAll);
7
8       app.get('/tests/:testId', tests.findOne);
9
10      app.put('/tests/:testId', tests.update);
11
12      app.delete('/tests/:testId', tests.delete);
13  }
```

Figure 7 API routes from server

4.2.2 Model for database

Describing the model when using MongoDB is a very important part of building the API for the server.

app > models > JS test.model.js > ...

```
1   const mongoose = require('mongoose');
2
3   const TestSchema = mongoose.Schema({
4       domain: String,
5       email: String,
6       test: Object
7   }, {
8       timestamps: true
9   });
10
11  module.exports = mongoose.model('Test', TestSchema);
```

Figure 8 Database model

From figure 7, we can already what kind of datatypes is required from the client when using the application. The domain and email are required by the client while the test being an object is computed in the server based on the parameter passed from the client.

4.2.3 Creating new test

The two main parameter requested form user in order to run the test from the client side are the parameter needed on the server in order for the test to be conducted.

Domain – the domain name or IP address the client wishes to test and see it vulnerabilities.

Email – needed for sending the test result to the client upon the completion of the test.

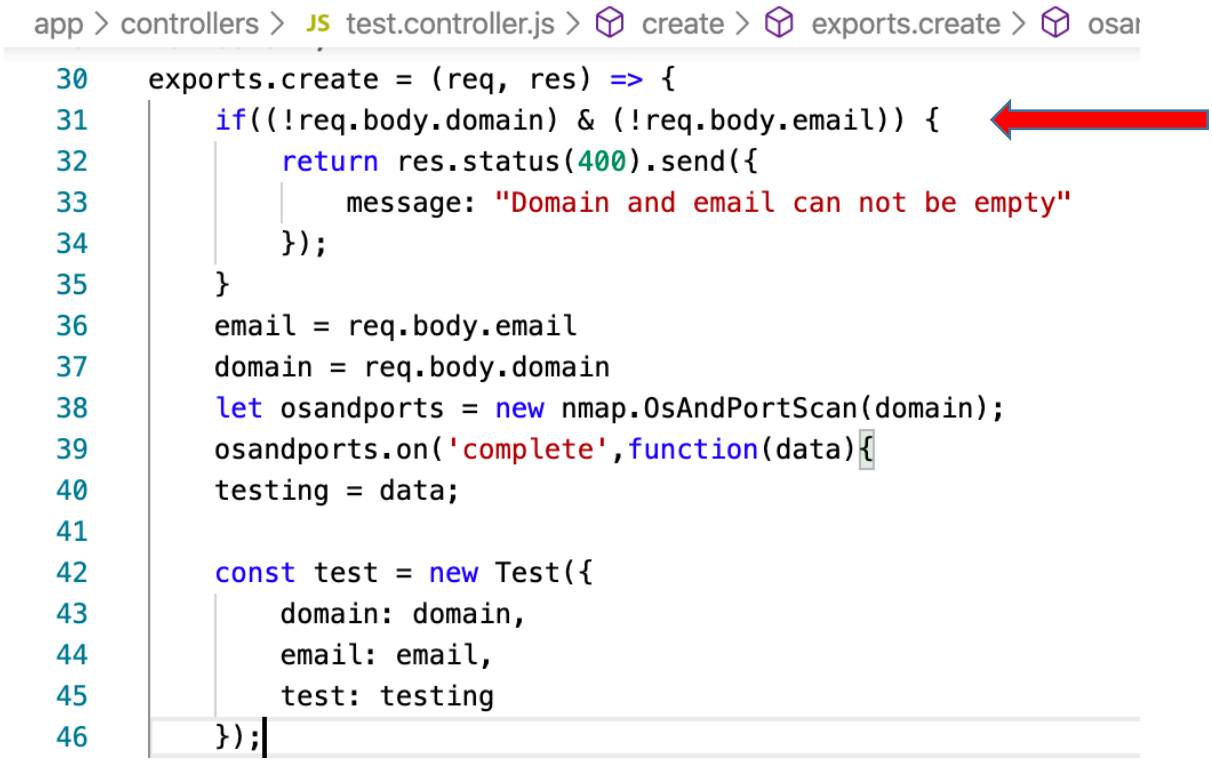

Figure 9 Create test controller

As seen in the figure 9 above both parameters are requested for the test to be conducted.

4.2.4 Running test and sending result

Running the test and send the rest result was integrated into the export.create function. In order to test the domain, the value of that particular domain need to be passed from the frontend.

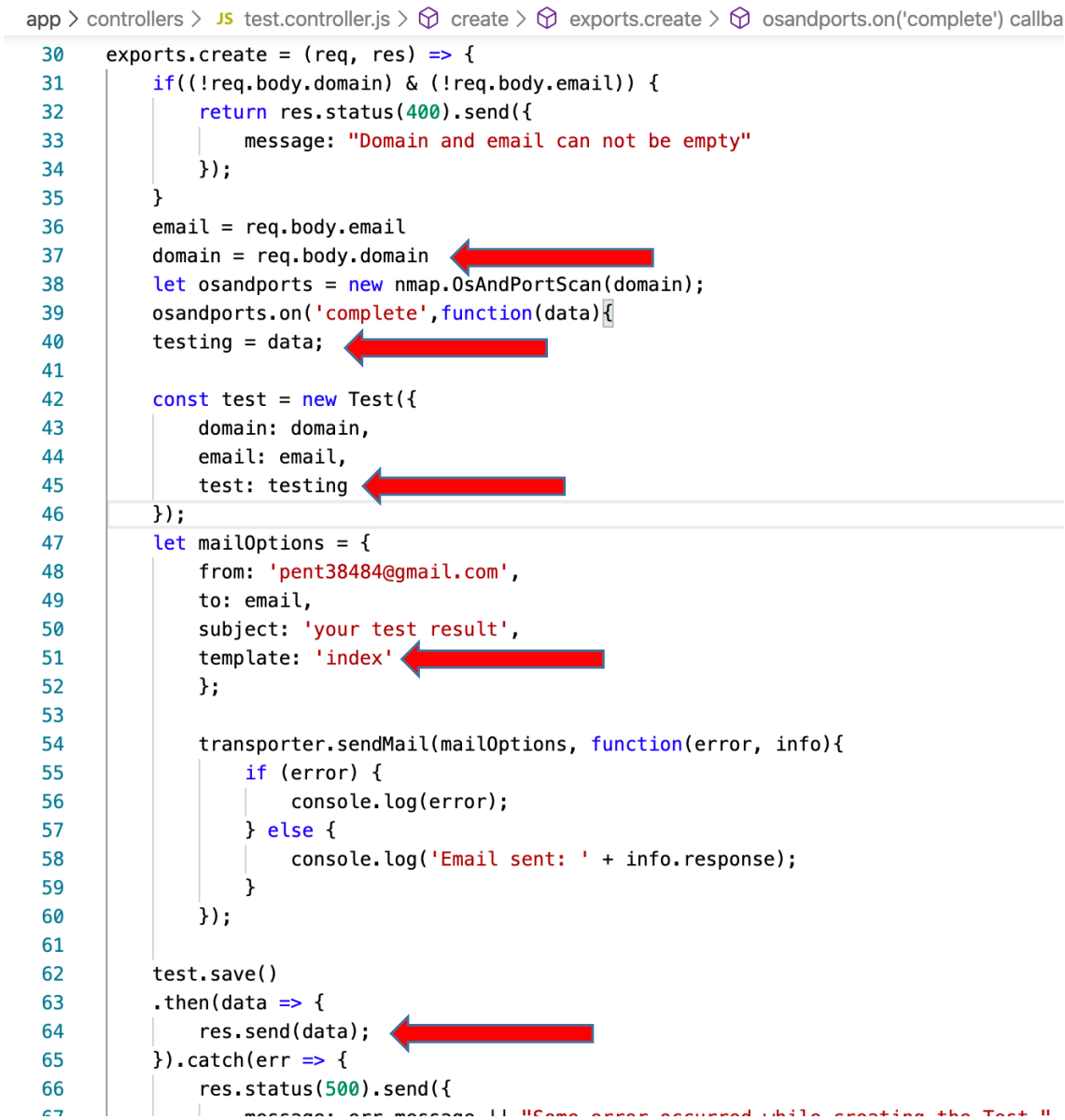

Figure 10 Running test and sending result

Domain = req.body.domain is where the domain value is being passed into a variable.

Testing = data is where the response from the test is being sent back and passed to data.

Test: testing represent the test result and its one of the objects being passed to the instance of the test function which was imported from the model.

Template: index is how the result will show in email of the client which is one of the objects being passed to the instance of the mail options for sending email using the node mailer module.

Res.send(data) is the point at which when all the previous stages have been passed successfully it then sends the JSON object to the database for storage.

4.2.5 File structure

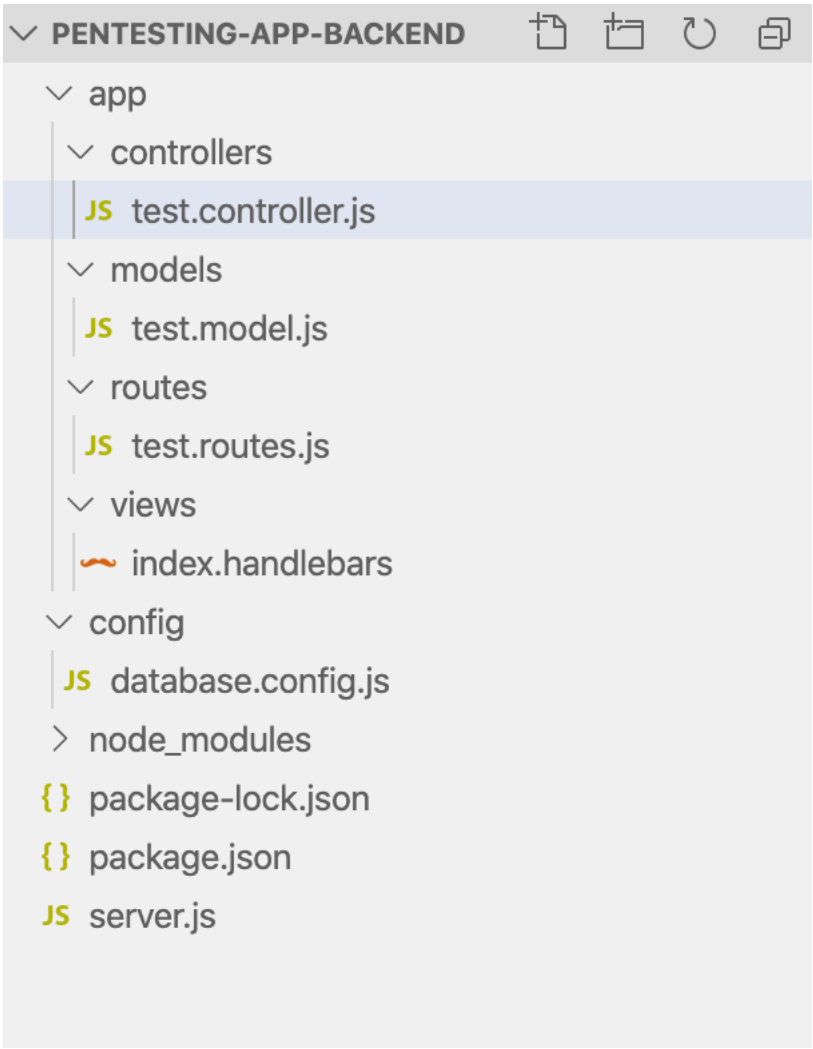

Figure 11 Server project structure

Server file structure (Figure 11) is divided into the following part. App folder where the controllers, models and routes file are being stored. The controller handles all the incoming HTTP request and send back appropriate response based on the request being made. Models are schemes for the for different collections of json objects that are stored in the MongoDB database. Routes are the API paths where users send request. The view file contains visible files but in this case the template for the email sent to users plus result.

Config are for modules exporting constants and in this case is exporting database URL. Node modules, package-lock.json and package.json files are generated automatically, they are needed for the server to run. The main file is server.js where all the server configurations are being made for example where the sever starts, what ports to listen to and etc.

4.3 Client-side

The client-side is implemented using React.js which also uses package.json file with similar explanation like in section 4.2.

4.3.1 Structure

React.js uses App.js as an entry point from which any component can be render but will still be inside the main entry component. In this application components are using Axios to perform request. Axios is a promised based HTTP client. Figure 12 gives more understanding of this process.

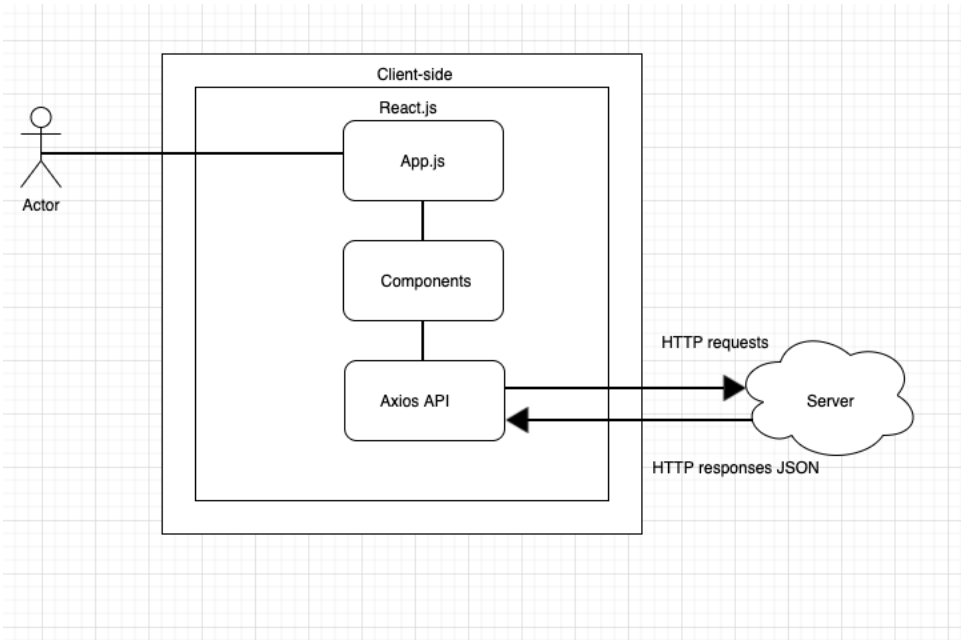

Figure 12 Client-side structure

4.3.2 Components

App component is the main component that user will see, the app component is displaying the application logo, search component and a footer tag.

src > JS App.js > App

```
import React from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import { MDBContainer } from "mdbreact";
import Search from './components/Search'
import './App.css';
import Logo from './Assets/PT.png';

function App() {
  return (
    <div className="App">
      <div>
      <img src={Logo} alt="website logo"/>
      </div>
      <div>
      <Search/>
      </div>
      <div className="footer-copyright footer">
        <MDBContainer fluid>
          &copy; {new Date().getFullYear()} Copyright: <a href="https://alapomeji.web.app/" target="blank"> Toyib Ahmed </a
        </MDBContainer>
      </div>
    </div>
  );
}

export default App;

```

Figure 13 App.js component

Search component is the component user will interact with, it's the component that ask users for inputs, validates the inputs and send the values to the server for the test to be made. Formik module is used to handle this form process because of its easy integration of error messages and validation.

src > components > JS Search.js > Search > render

```
class Search extends Component {

  constructor(props){
    super(props)
    this.state = {
      value: '',
      domain: '',
      email: '',
      message: null
    };
    this.onSubmit = this.onSubmit.bind(this);


  }

  componentDidMount(){
    Service.createTest(this.state.id)
    .then(response => this.setState({
      domain: response.data.domain,
      email: response.data.email
    }))
  }

  onSubmit(values){
    let test = {
      domain: values.domain,
      email: values.email
    }
    console.log(test);
    Service.createTest(test)
    .then(() =>
    this.setState({ message: `Domain sent for testing, Please check your email for the result` }),
    this.setState({values: ''}))
  }

  validate(entries) {
    let errors = {}

    if (!entries.domain){
        errors.domain = 'Please enter a domain'
    }
    if (!entries.email) {
        errors.email = 'Enter an email for result'
    }
    return errors
```

Figure 14 Search.js component

Service components is the component in charge of sending the test received in the search component but for simplicity sake author incorporate service component in the search component for easier implementation.

src > components > JS Service.js > ...

```
1   import axios from 'axios';
2
3   const TEST_API_URL = 'http://localhost:5000'
4
5   class Service{
6       createTest(test) {
7           return axios.post(`${TEST_API_URL}/tests`, test);
8       }
9       retrieveTest(id) {
10          return axios.get(`${TEST_API_URL}/tests`);
11      }
12  }
13
14  export default new Service();
```

Figure 15 Service.js component

4.3.3 User Interface

Achieving the present appearance of the application different component and modules were used for example the react-bootstrap which make gives the pleasant look of the web application. The figures below show the web application in action.

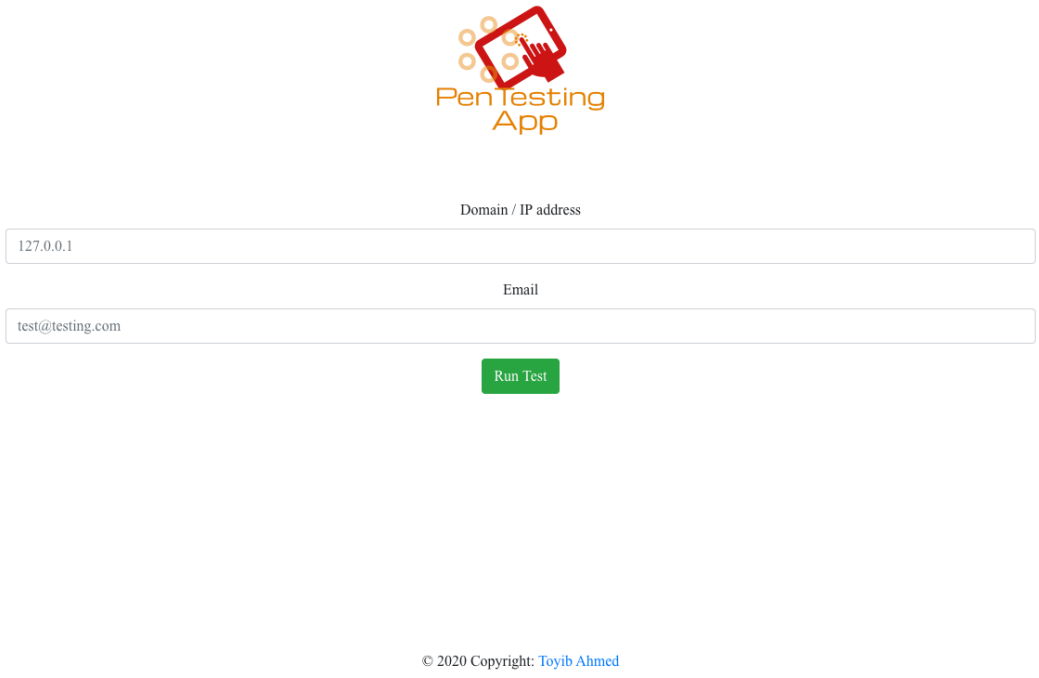

Figure 16 Web application homepage

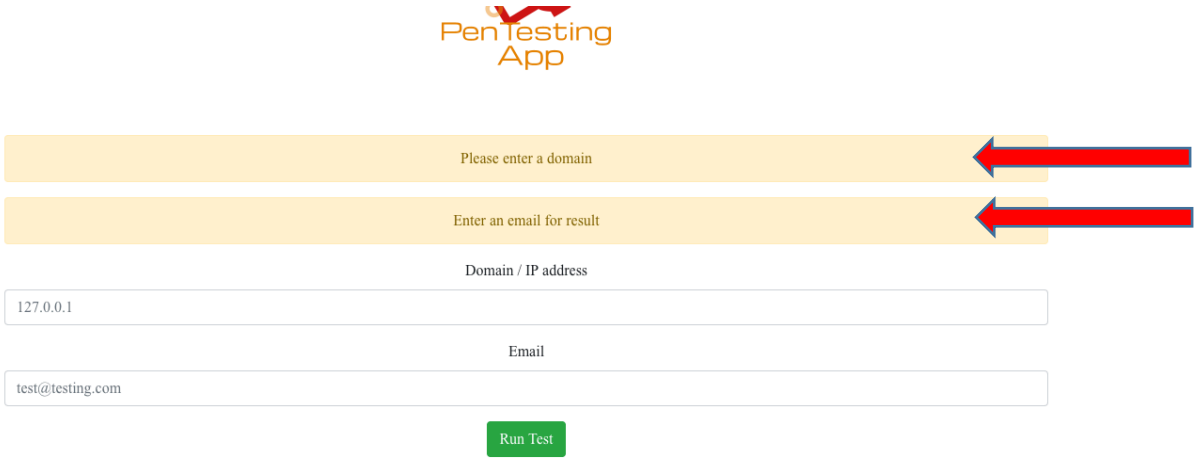

Figure 17 Error Messages when fields are left empty

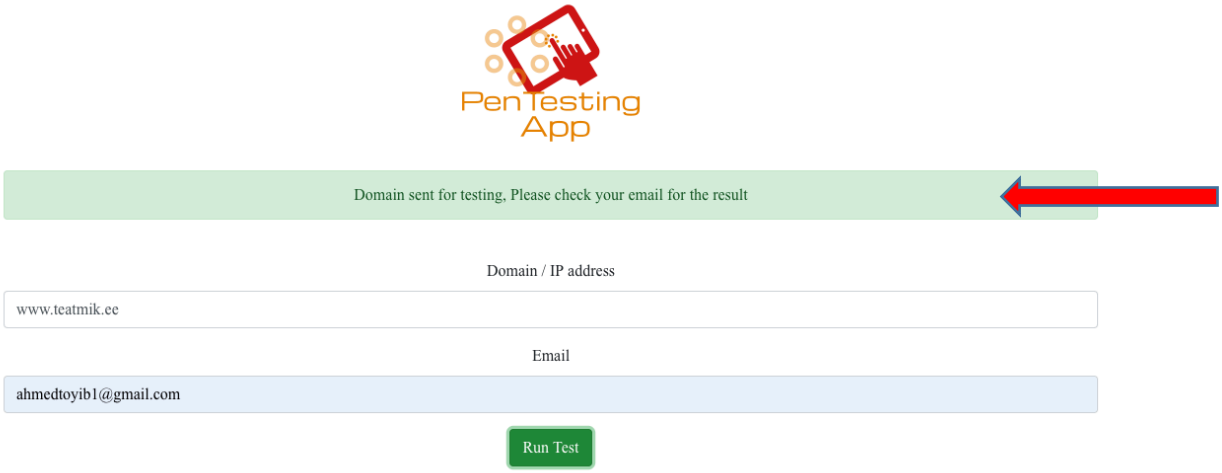

Figure 18 Successful message on test completion

4.3.4 File structure

Client-side file structure (Figure 19) is divided into the following parts. Static files like web page logo are being save in the assets folder. Components folder are for the components files which are described in section 4.3.2. Styles folder consist of the stylesheets files used to make the web application colorful. App.js is the main page for the application.

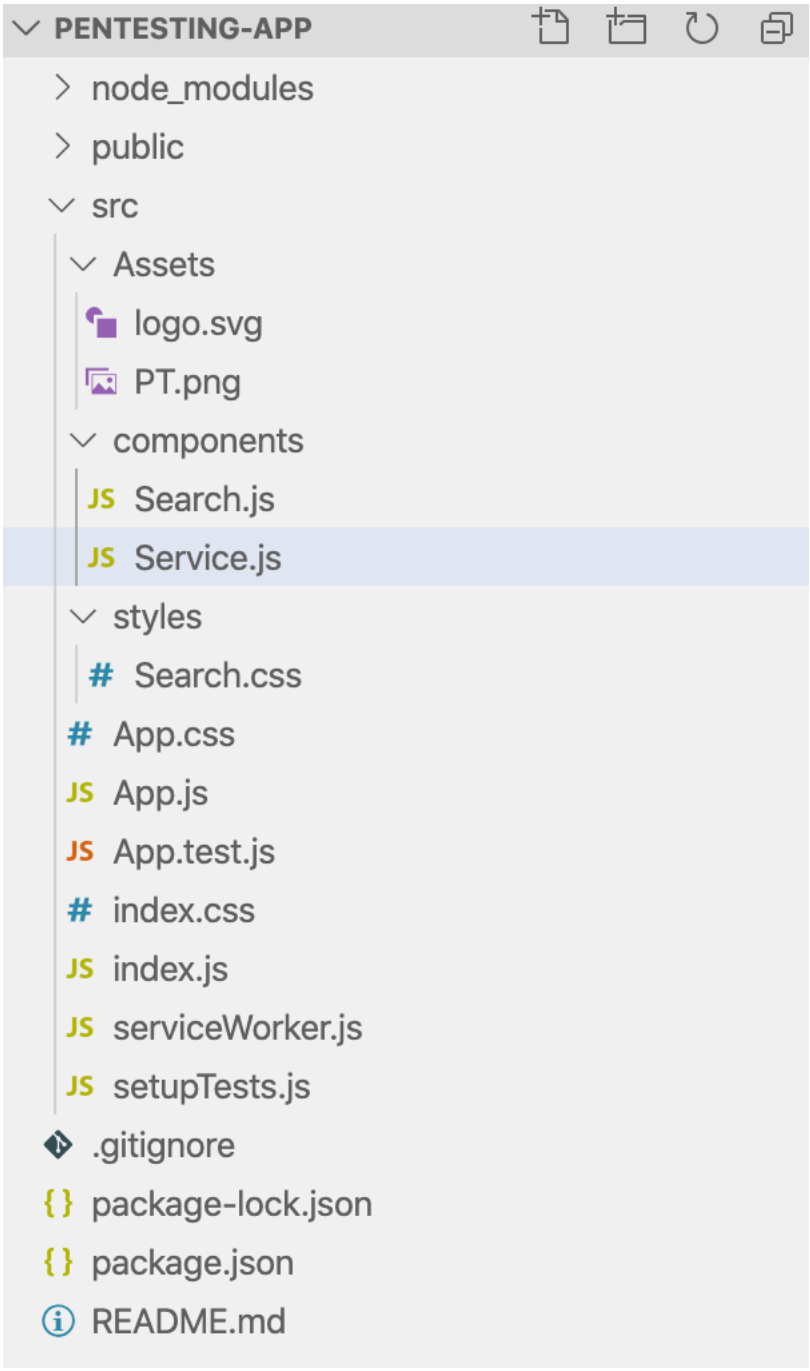

Figure 19 Client file structure

4.4 Future development

For future development first thing which is really important is to make the result visible to the user from the web page and also provide the possibility to download the result in pdf format.

Implementing all other steps of penetration testing that this thesis work did not cover.

Integrate a reusable header that contains menu and sub- menu item and give users more flexibility to choose from what kind of test they will like to run by using checkboxes to specify which tests to be conducted and which tests should be ignored.

5 Summary

The goal of this thesis work is to build a web application that's open source and suitable for a non-technical person to check the security flaws of a domain.

The result is a web application that accepts inputs from the user from the client-side of the application and send this to the server-side of the application as a JSON object, while the server-side run the specified vulnerability test on the specific domain submitted by the user and send the respective result from the testing to the user specified email address.

This solution will help small businesses to evaluate their domain on the basic penetration testing level and have an idea of what vulnerability they have or could possibly have.

The final product of this thesis work is a full-stack web application with the server (Backend) written in Node.js and client (Frontend) written in React.js. Date is saved in MongoDB and the application deployed to a different server on Heroku.

6 References

[1] Jason Andress and Ryan Linn. Coding for Penetration Testers: building better tools. Syngress, ISBN 978-1-59749-729-9, 2012

[2] Christopher Hadnagy. Social Engineering: The Art of Human Hacking. Wiley, ISBN-10 0470639539, December 2010

[3] David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. Metasploit: The Penetration Tester's Guide. No Starch Press, ISBN 978-1-59327-288-3, 2011.

[4] Rapid7, Inc. "Metasploit Framework" [Online]. Available: http://www.metasploit.com/download/. [Accessed 06 01 2021].

[5] "Insecure.Com LLC. Nmap" [Online]. Available: http://nmap.org/download.html. [Accessed 06 01 2021].

[6] Gordon "Fyodor" Lyon. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Nmap Project, ISBN 978-0-9799587-1-7, 2009.

[7] "Wireshark," [Online]. Available: https://www.wireshark.org/download.html. [Accessed 06 01 2021].

[8] "Ulf Lamping, Richard Sharpe, and Ed Warnicke. Wireshark User's Guide: for Wireshark 1.9," [Online]. Available: http://www.wireshark.org/docs/wsug_html_chunked [Accessed 07 01 2021].

[9] "BurpSuite," [Online]. Available http://www.portswigger.net/burp/download.html. [Accessed 07 01 2021].

[10] "Penetration Testing of Web Application in a Bug Bounty Program," [Online]. Available: http://www.diva-portal.org/smash/get/diva2:723516/FULLTEXT02.pdf [Accessed 06 01 2021].

[11] “Penetration Testing on Domain Name Service,” [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/155680/Famuwagun_Samuel.pdf?sequence=1&isAllowed=y. [Accessed 07 01 2021].

[12] “What Is a Single-Page Application?,” [Online]. Available: https://dzone.com/articles/what-is-a-single-page-application. [Accessed 07 01 2021].

[13] “ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET,” [Online]. Available: https://msdn.microsoft.com/enus/magazine/dn463786.aspx. [Accessed 07 01 2021].

[14] J. Gossman, “Introduction to Model/View/ViewModel pattern for building WPF apps,” 08 10 2005. [Online]. Available: https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern. [Accessed 07 01 2021].

[15] “Introduction,” [Online]. Available: https://reactjs.org. [Accessed 07 01 2021].

[16] “Getting Started,” [Online]. Available: https://reactjs.org/docs/getting-started.html. [Accessed 07 01 2021].

[17] “Node”, [Online]. Available: https://nodejs.org/en/. [Accessed 07 01 2021]

[18] “Express,” [Online]. Available: http://expressjs.com/. [Accessed 07 01 2021].

[19] “Cors,” [Online]. Available: https://www.npmjs.com/package/cors/. [Accessed 07 01 2021]

[20] “Mongoose,” [Online]. Available: https://www.npmjs.com/package/mongoose. [Accessed 07 01 2021].

[21] “MongoDB,” [Online]. Available: https://www.mongodb.com/. [Accessed 07 01 2021].

[22] “Nodemailer,” [Online]. Available: https://nodemailer.com/about/. [Accessed 07 01 2021].

[23] “Node-nmap,” [Online]. Available: https://www.npmjs.com/package/node-nmap/. [Accessed 07 01 2021].

[24] “DNS,” [Online]. Available: https://nodejs.org/api/dns.html - dns_class_dns_resolver. [Accessed 07 01 2021].

[25] “Heroku,” [Online]. Available: https://www.heroku.com/home/. [Accessed 07 01 2021].

[26] “MVVM,” [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel/. [Accessed 07 01 2021].

[27] “MVVM React,” [Online]. Available: https://medium.cobeisfresh.com/level-up-your-react-architecture-with-mvvm-a471979e3f21/. [Accessed 07 01 2021].

[28] “Single page Architecture,” [Online]. Available: https://www.goconqr.com/c/74455/course_modules/113576-single-page-vs-multi-page-architecture#/. [Accessed 07 01 2021].

[29] “MVC pattern,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/MVC/. [Accessed 07 01 2021].

Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Toyib Ahmed

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis Open-source Penetration Testing Tool for Beginners, supervised by Kaido Kikkas,

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

07/01/2021

---

[1] *The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.*