Project Deliverables

Light Pentesting Tool – API & Database Development

Written by LT Loo

Introduction

Interns in SecureStack were to co-operate with each other to develop a light pentesting tool. The software tool aims to scan for vulnerabilities in software applications. This tool is made up of five components, front-end User Interface (UI) for interaction between user and scanning tool, GraphQL middleware as communicator between front-end and back-end, web-based Application Programming Interface (API) for data management, PostgreSQL database for data storage, and scanning logics to scan for vulnerabilities. All components are then integrated and containerised using Docker in the final stage of project.

This document will be focusing on the API and database development of the application. The contents of the document include full descriptions and requirements of both components, final outcomes, and appropriate diagrams to provide better understanding.

Table of Contents

1 FastAPI

API is the acronym for Application Programming Interface, which is a software intermediary that allows communication between computers or between computer programs. An API is developed for the light pentesting tool using the FastAPI web framework to carry out multiple functions.

1.1 Objectives

The purposes of building an API in the scanning application are as below:

- Establish connection with database to store and manage data by carrying out operations such as data insertion, extraction, deletion, and modification.
- To receive request (input data) from middleware and carry out appropriate operations before returning output data as response.
- To trigger scanning functions when requested, and store or update scanning results in database.

1.2 Framework

The web framework used in developing API is FastAPI. It is a modern web framework for building APIs with Python programming language. The reason FastAPI is chosen to be implemented in the project is because the framework carries useful features that support the development of the API. The key features are given as following:

- **High performance**
- **Security and authentication**
  FastAPI library has its own built-in security scheme to provide support for security and authentication.
- **Data validation**
  With the implementation of Pydantic package library, FastAPI supports data validation and setting management.
- **Automatic documentation**
  FastAPI generates visual documentation, allowing developers to interact with API's resources. This makes it easier for back-end implementation and client-side consumption.
- **Easy to learn**
  FastAPI provides documentation that covers various topics with clear explanations and examples.

1.3 Implementation

1.3.1 Paths

Paths are created to manage data in database. The table below briefly describes the four main paths applied in the API.

Path	Description
POST	Create new data and insert data into database. Trigger scanning logics to scan targeted application.
GET	Return data from database.
PUT	Update data in database.
DELETE	Delete data from database.

Figure 1 API for light pentesting tool using FastAPI framework.

1.3.2 Data Validation

There will be a lot of data stored in the database, hence why it is important for API to check if the data is valid in terms of types and formats before it is being processed and stored into the database, so that the database remains organized. As mentioned in Section 1.2, Pydantic package library is implemented to achieve the goal.

1.3.3 Authentication

The authentication scheme is built using FastAPI's built-in security module. A valid access token will be sent to API for verification and API will only perform requested operation after verifying the token. The token used in the API is Bearer token. It is the predominant type of access token used with Auth0, a login platform. This implementation ensures data to be accessed and transferred securely.

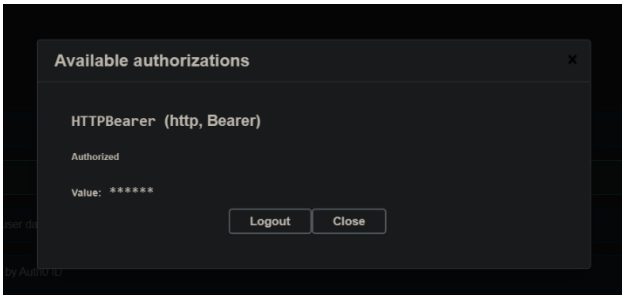

Figure 2 Implementation of token verification using Bearer.

1.3.4 Background Task

FastAPI provides a special feature that allows specific tasks to be run in the background without delaying the returning of a response. Scanning an application often takes time, and by implementing this feature in the API, we can then return the response straight away while scanning is carried out in the background.

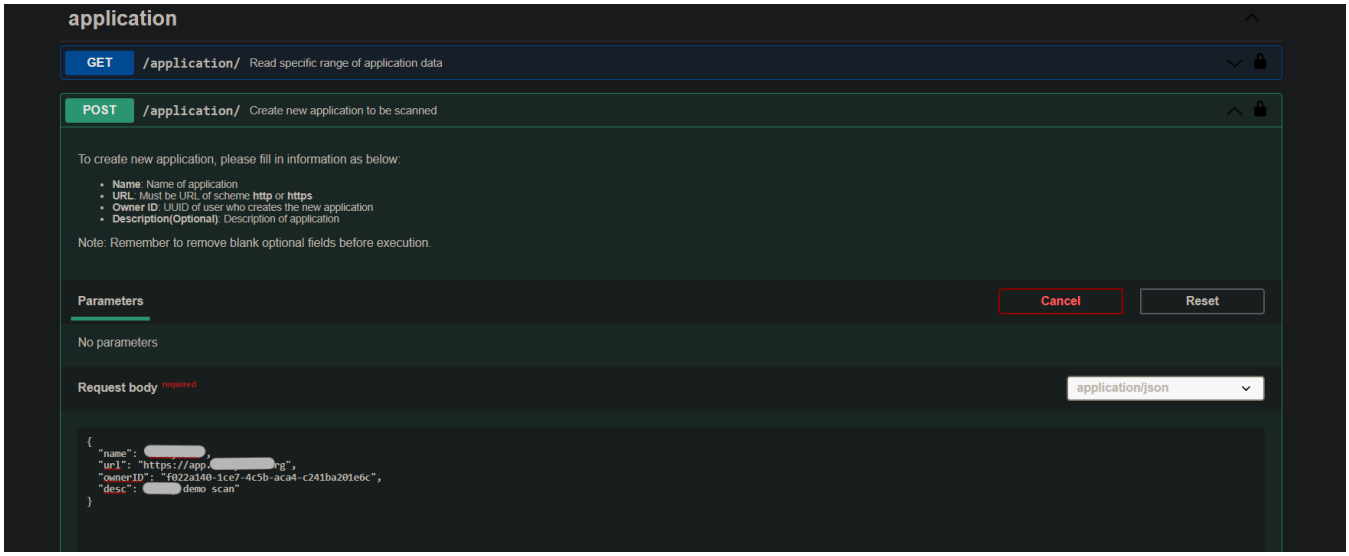

Figure 3 Creating new software application to be scanned through POST method.

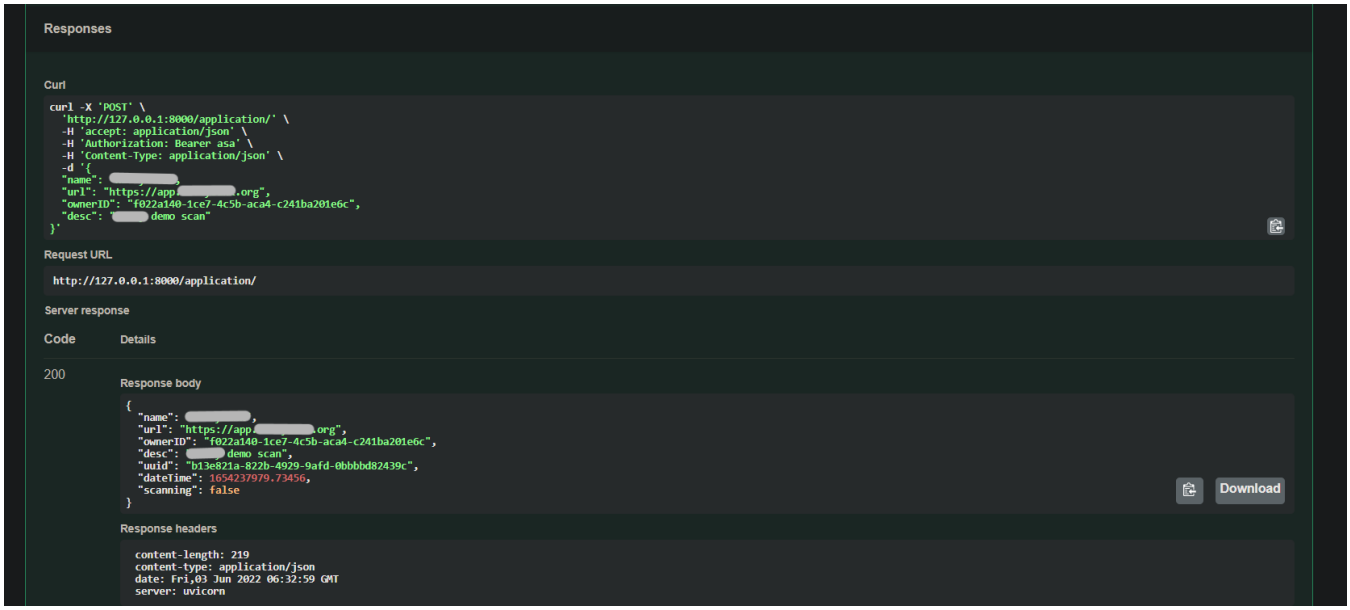

Figure 4 Response received after successfully creating new software application.

2 PostgreSQL Database

Database is an organized collection of structured information, or data, typically stored electronically in a computer system. PostgreSQL is used to build the database for the light pentesting tool to store various data.

2.1 Objectives

The purposes of building a database are as below:

- To store data in an organized manner for each access and simple management.
- Relationships established between tables to prevent redundant data and make sure data stays synchronized (information in one table matches information in another).

2.2 Database Management Software (DBMS)

The database system used is PostgreSQL. It is a powerful, open-source object-relational database system which is known for its reliability, feature robustness and performance. The reasons of us choosing to implement the system is as below:

- **Open source**
  Source code is freely available under an open-source license. This allows developers the freedom to use, modify and implement it as per one's business or personal needs.
- **Support for writing database functions using various programming languages**
  Here, Python language is used to build the database.
- **Scalability**
  There are multiple technical options for operating PostgreSQL at scale, which allows the database to grow upon developer's needs.
- **Diversified extension functions**
  PostgreSQL has flexible management system as it supports different kinds of technique for geographic data storage.

2.3 Implementation

2.3.1 Schema Diagram

A schema diagram is constructed to show the information stored in each table and the relationships between the tables. Relationships between tables allow us to retrieve related information in child table through its parent table or vice versa, saving time in accessing data.

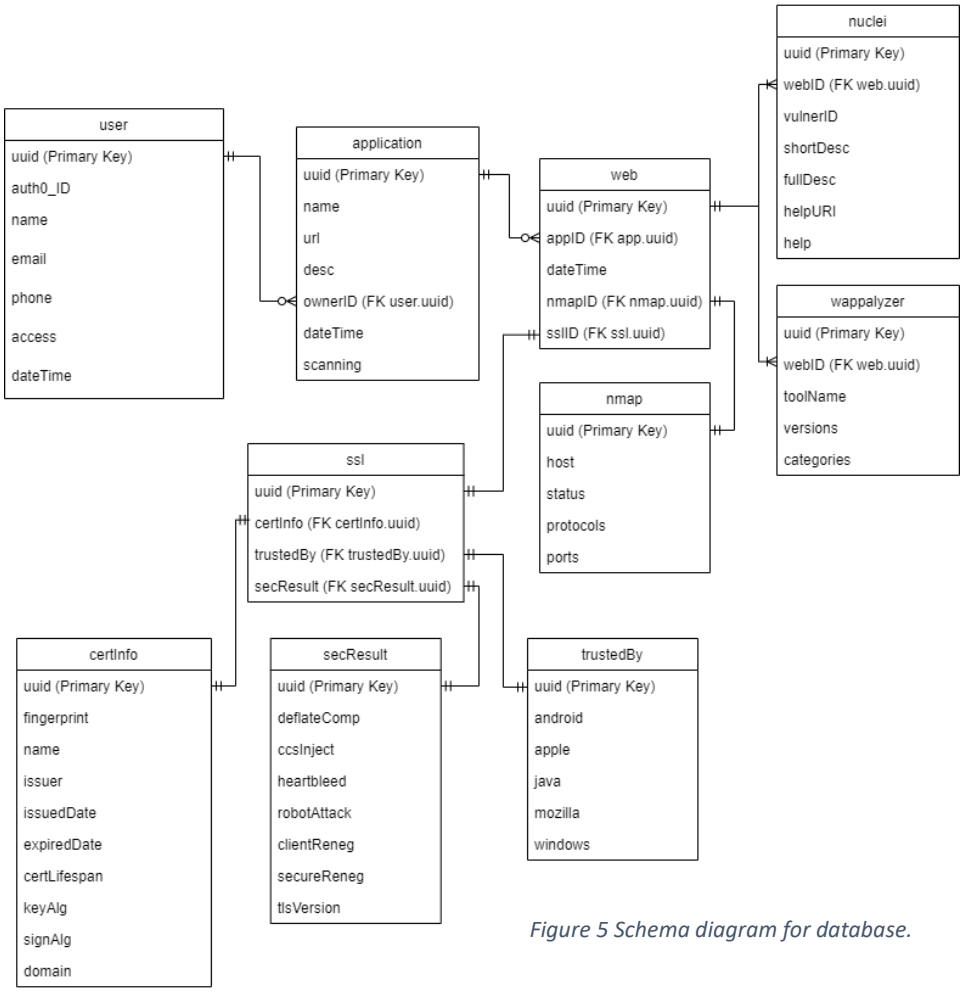

Figure 5 Schema diagram for database.

2.3.2 SQLAlchemy

SQLAlchemy is a Python library that facilitates the communication between Python programs and database. The Object Relational Mapper (ORM) feature allows translation from Python classes to tables on relational database. It also automatically converts function calls in Python script files to SQL statements.

```
postgres=# select * from "user";
                 uuid                 |  auth0_ID  |  name   |        email        |   phone    |  accessPermission  |     dateTime
--------------------------------------+------------+---------+---------------------+------------+--------------------+------------------
 f022a140-1ce7-4c5b-aca4-c241ba201e6c | example123 | example | example@example.com | 0123456789 | {"Access Profile"} | 1654237886.006102
(1개 행)


postgres=# select * from "application";
 uuid | name | url | description | ownerID | dateTime | scanning
------+------+-----+-------------+---------+----------+----------
(0개 행)


postgres=#
```

Figure 6 Database to store data of users and scan results of software applications.

3 Docker

Docker is an open-source containerization platform that enables developers to package applications into standardized units called containers that have everything the software needs to run including libraries, system tools and source code. All components of the light pentesting tool will be running in Docker containers.

3.1 Advantages

The reasons of us choosing to use Docker are given below:

- **Portability**
  Applications can be deployed to any environment or system. Application is guaranteed to perform exactly as it does when run locally. It allows developers that use different operating systems to work on the same project easily.
- **Container**
  Small containerized applications in Docker make it easy to identify issues and roll back for remediation.
- **Isolation**
  A Docker container that contains the application also includes the relevant versions of any supporting software that the application requires. If other Docker containers contain other applications that requires different version of the same supporting software, it wouldn't be a problem as different Docker containers are independent of each other.

3.2 Implementation

3.2.1 Dockerfile

Dockerfile is a text document that consists of instructions to build Docker images which will then be used to create Docker containers. Dockerfile is written to build the image of the API. And since there is already a PostgreSQL image provided by Docker, there is no need to write a Dockerfile for the database as all we need to do is extract the image from Docker's resource and build the container.

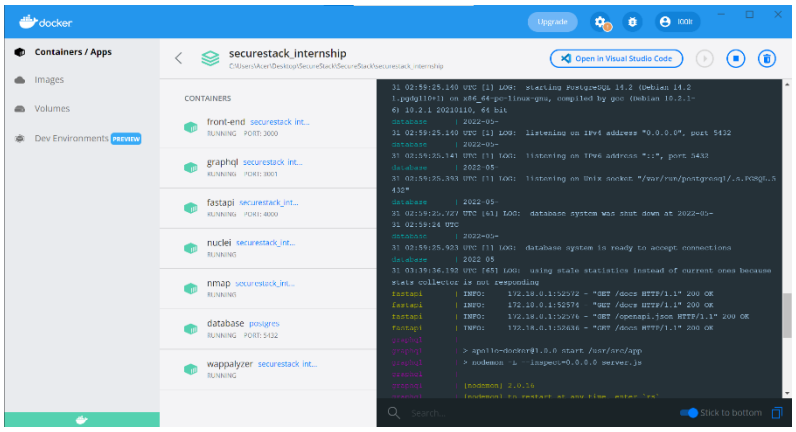

Figure 7 Components are containerised and integrated in Docker.

3.2.2 Docker Compose

Docker compose is a tool for defining and running multi-container Docker applications. The light pentesting tool will have multiple containers running in Docker. Instead of building the containers one by one, a YAML file is written to configure all application's services so that we are able to create and start all containers with just one single command:

```
docker-compose up --build –detach
```

4 Final Deliverables

The API and database are successfully built on time to be integrated with other components to produce the light pentesting tool.

- Both API and database run successfully in Docker containers along with other components. All operations are executed smoothly, and data is managed as requested in database successfully while running in Docker.
- Connection between API and database is established successfully. API is able to insert, retrieve, update or delete data into or from database upon requests. Data is stored in the correct tables and format.
- Relationships between tables in database is established successfully. API is able to access the data in child tables through their respective parent tables and vice versa. When deleting the data in parent table, the matching data in child data is successfully deleted as well without messing up the structures of the tables.
- API is able to trigger scanning functions and immediately return a response successfully. The implementation of background task functions smoothly.