|  |  |
| --- | --- |
| Name: | Arpan Nyati |
| Lab User ID: | 23SEK3324\_U04 |
| Date: | 09-01-24 |
| Application Name: | Damn-Vulnerable-Bank |

**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)

A white line on an orange background

Description automatically generated 

AWS EC2 Instance SNYK Vuln Scanner

Backend Server

Android Application

Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

This application is a financial management or banking application providing a secure method for users to manage their financial transactions and beneficiary information.

1. Sign Up: Allows users to create a new account.

2. Login: Existing users can access their accounts.

3. My Profile Interface: It provides users an interface to view and edit their profile, including personal details.

4. Change Password: A feature enabling users to update or change their password to maintain account security.

5. Settings Interface to Update Backend URL: This feature might be used primarily by admins/developers for configuration purposes.

6. Add Fingerprint Check: It adds an extra layer of security by requiring fingerprint authentication before transferring or viewing funds.

7. Add Pin Check: Another security feature that requires entering a pin before moving or viewing funds.

8. View Balance: Users can check the current balance in their account.

9. Transfer Money: Allows users to transfer money from their account to another. They have two options:

- Via Manual Entry: Users can manually input the recipient's account details.

- Via QR Scan: Users can simply scan the QR code of recipient's account details for transfer.

10. Add Beneficiary: This feature allows users to add and save recipient's account details for future transactions.

11. Delete Beneficiary: Allows users to remove a saved beneficiary.

12. View Beneficiary: Users can see a list or details of their added beneficiaries.

13. View Transactions History: Provides a history of all past transactions for tracking and auditing purposes.

14. Download Transactions History: Allows users to download and maintain a personal record of their transaction history.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**During peak usage hours, the app will not show any unexpected behavior or undisclosed vulnerabilities.**

**Known**

**In the event of a DDoS attack, the app will continue to function normally without any significant increase in errors or decrease in performance.**

942236

**Unknown**

**Unexpected API failures or wrong responses.**

**If API services fail randomly or give unexpected responses, the application will correctly handle such anomalies without causing any customer-facing issues**

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Preparation**

1. Establish an AWS account if you don’t already have one. Log in to the AWS Management Console and access the EC2 Dashboard.

2. Create an AWS t2.medium instance. Choose your desired OS (e.g. Ubuntu Server 20.04 LTS).

3. After the instance is successfully launched, connect to your instance via SSH through the command line or use an SSH client.

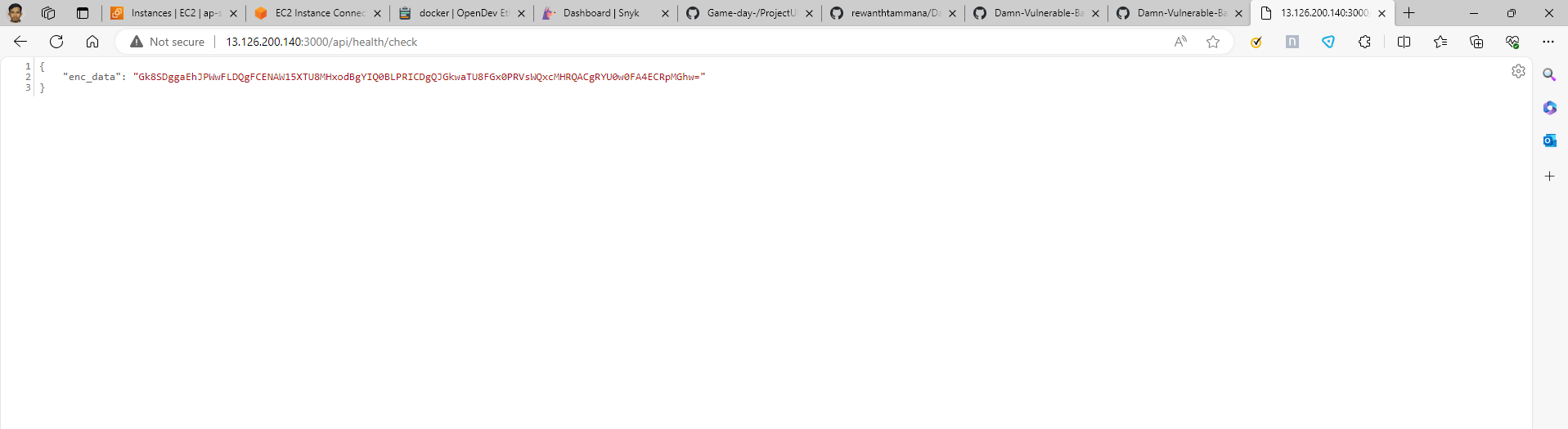
**Implementation**

1. Update the system: Execute the command `sudo apt update && sudo apt upgrade -y` to ensure all the system packages are up to date.

2. Install Docker: Run the installation commands for Docker as follows:

3. Confirm Docker has been installed correctly by running: `sudo docker run hello-world`. This command downloads a test image and runs it in a container.

4. Build and Run your server: Build your Docker image and run it accordingly.



Checking the working of the Backend API

A screenshot of a computer

Description automatically generated

Live Server on AWS EC2

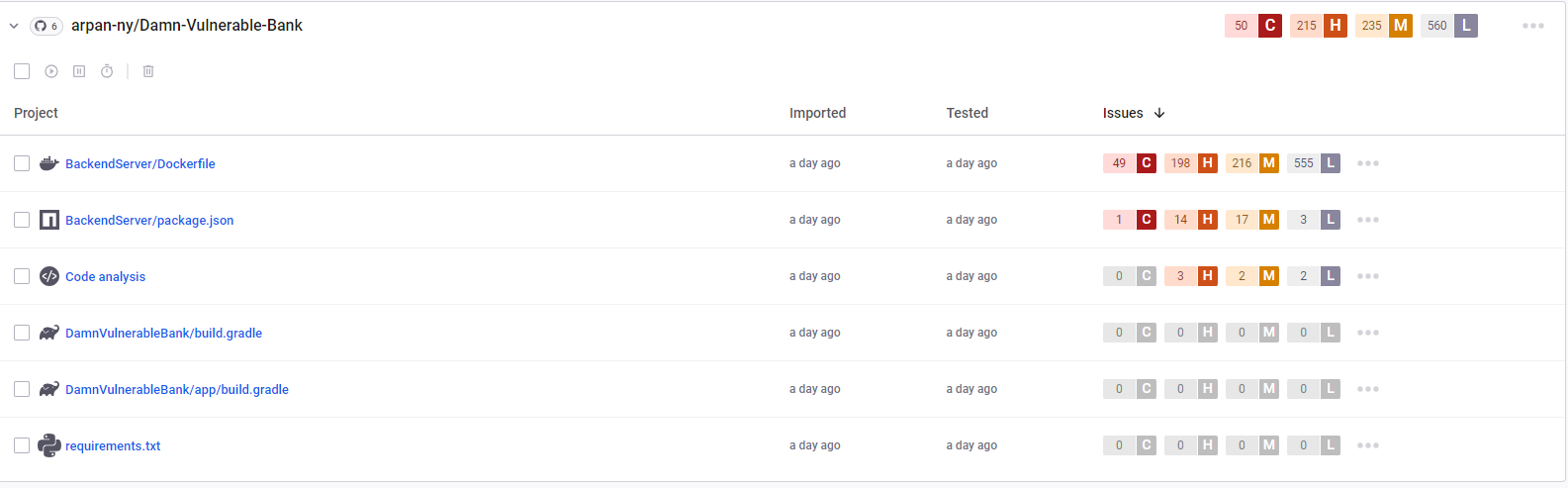
**Observation**

1. Install Snyk: Install the Snyk CLI tool, which you can use to run a security vulnerability test on your machine. Here, you can install it using npm by running `npm install -g snyk`.

2. Authenticate your session: Use `snyk auth` command, it will guide you to the Snyk website to obtain a token. After that Snyk CLI will be connected to your account.

3. Run a Test: Use the `snyk test` command to start the vulnerability scan. This will analyze your project’s dependencies and report any known vulnerabilities.

4. Interpret the Results: The snyk test command will provide a report of vulnerabilities it encountered, their severity level, and even possible remediation methods.



**Analysis**

Damn Vulnerable Bank has 50 Critical, 215 High, 235 Medium and 560 Low vulnerabilities. The backend server Dockerfile has 49 critical vulnerabilities. Package.json has 1 critical vulnerability.

Following are the 4 critical vulnerabilities in the Damn Vulnerable Bank.

1. CVE-2022-1664

Status: Reserved

This CVE ID has been reserved by MITRE, the organization that manages the CVE catalogue, and currently, there isn't any specific vulnerability associated with it. The reasons behind reserving a CVE might include anticipation of a future vulnerability disclosure, a vulnerability that’s not yet fully analyzed, or it might have been reserved for use during the CVE Board's processes.

2. CVE-2018-16402

This is a vulnerability in the "Script Security Plugin" 1.49 and earlier in Jenkins. This flaw allows a person with Overall/Read access to download an arbitrary 'allowedSignatures.txt' file from the administrative monitor. Once an attacker acquires this file, they can use it to escalate privileges. This vulnerability was fixed in version 2.0 of the "Script Security Plugin". Users are advised to update to the latest version to avoid falling prey to the vulnerability.

3. CVE-2022-25235

Status: Reserved

Just like CVE-2022-1664, this CVE ID has been registered but not yet used or associated with any vulnerability. More details on this vulnerability will be made available once it has been classified. It's recommended to regularly check updates related to this vulnerability for timely mitigation.

4. CVE-2017-14626

This vulnerability exists in SaltStack Salt before 2017.7.2 and 2016.11.6. It's an arbitrary file read vulnerability in minion's master. This allows remote unauthorized actors to access random files on the hosting system using salt modules via Salt's minion from the master. An attacker could exploit this to read all the system's files accessible to the Salt master process. The company released a patch for this vulnerability in later versions (2017.7.2 and 2016.11.6).