Nearsoft

Academy 2018B

Build Something From Scratch

"Vulnerability Scanner"

Team members:

Juan Daniel Amparan de la Garza
Hugo Fernando Licón Valenzuela
Guillermo Martínez Espina
Hector Manuel Robles Montes
Jose Manuel Ruiz Ruiz
Jose Francisco Saldana Ortega
Brian Harlán Samaniego Dávila
Andrés Sánchez Pérez
Javier Ivan Venegas Carrillo

nearsoft
an indecomm company

CDMX, October 16, 2018.

Contents

List of Figures

Chapter 1

Introduction

Users trust web services with sensitive information, like credit card information, birthdays, and addresses. This makes Web Applications a high-value target for bad intentioned folks also known as Black Hat Hackers.

It is the duty of the service provider to assure that the user information will be secure, this responsibility ends up in the hands of the developers.

This tool aims to help developers safeguard their Web Applications and Services providing a vulnerability scanner that can be used to find well known security holes.

It is designed to find various vulnerabilities using "black-box" method, that means it won't study the source code of web applications but will work using fuzz testing, scanning the pages of the deployed web application, extracting links and forms and attacking the scripts.

1.1 What?

The objective of this project is to deliver a web platform where users can check if a web page has ten of the Web Application Security Risks, by providing and scanning the url of the web page. The platform will contain a dashboard for the user, where it will show the scans results, represented by graphs and text, and also the user will have the option to export the report in .PDF, .XML and .TXT formats, or keep consulting it at the web platform by logging with his user account [1].

1.2 Why?

The importance of delivering a web platform that can scan the security risks of other web platforms, is to have a powerful tool that can be used for testing new websites and verify if a website is exposed to the most common security risks.

1.3 How?

The software will have several components, it will connect to Python REST API to perform certain tasks to test web vulnerabilities that could take a lot of time to finish.The application itself will be developed using Ruby On Rails, Python and PostgreSQL to manage data.

1.4 Limitations

Due to the time and the way some vulnerabilities checks have to be done (access to the website's source code and/or the server), we consider that it is not feasible to implement some vulnerabilities or to extend the functionality of the already implemented ones.

1.5 Vulnerabilities

The vulnerabilities tested by this application are the following ones:

- XSS - Cross Site Scripting
- DOM based XSS
- Blind SQL Injection
- SQL Injection
- HTTP Header SQL Injection
- PHP Injection
- OS Command Injection
- HTML Injection
- XML Injection
- LDAP Injection

Chapter 2

Design

In this chapter the approach for building the application will be shown with different diagrams and as well some diagrams which could help the understanding of the application's flow.

2.1 C4 Modeling

First of all for the sake of simplicity and understanding the design will be explained using the C4 model architecture.

As shown in the Figure 2.1, a System Context diagram is shown and it allows to step back and see the big picture of the application.

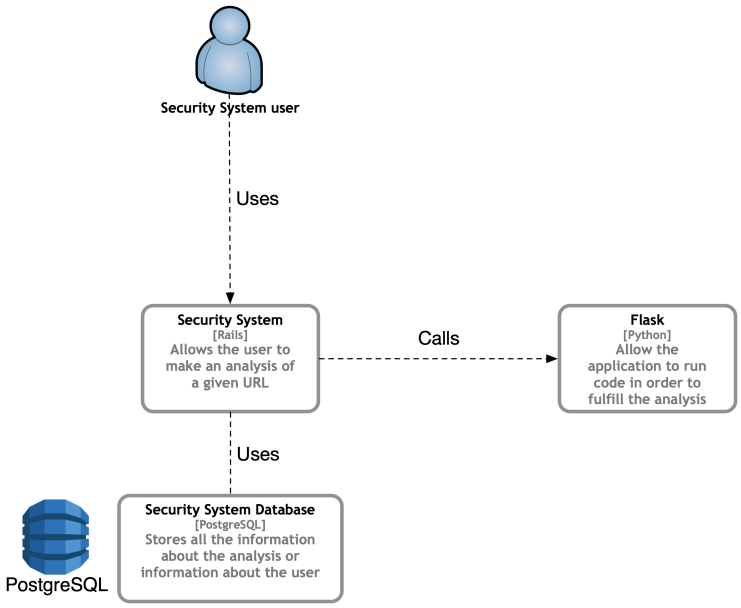

Figure 2.1. *System Context diagram*

In the Figure 2.3 the Container Diagram helps explaining the system in a deeper overview than the System Context Diagram. The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another [2].

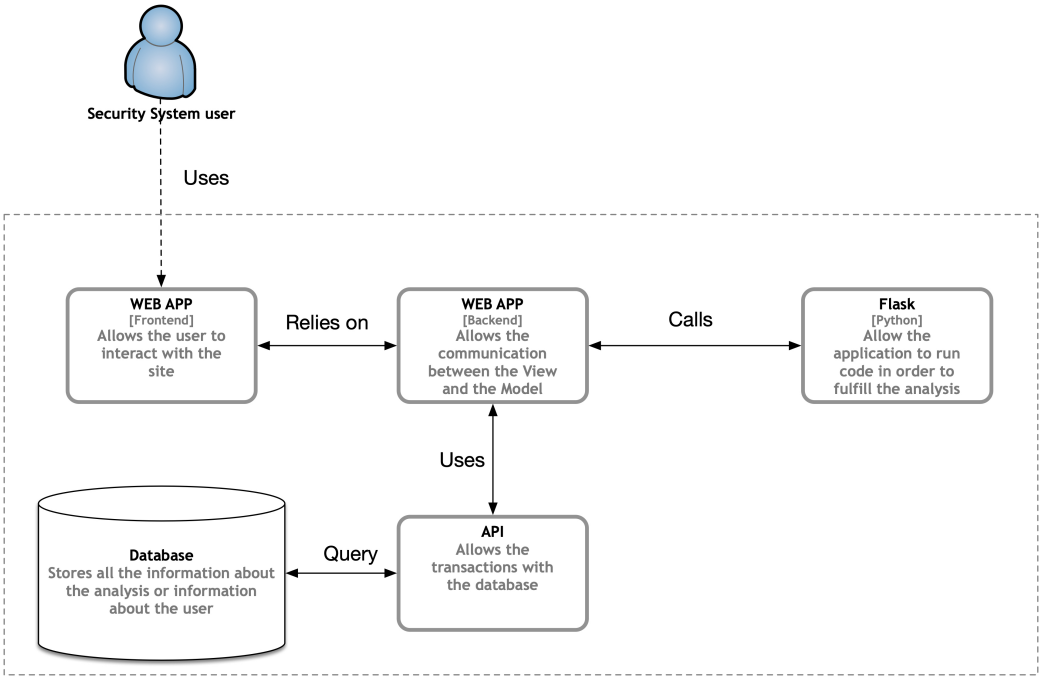

Figure 2.2. *Container diagram*

Finally on the Component diagram we identify the major structural building blocks and their interactions, it shows the different components of the application, their responsibilities and the technology/implementation details.

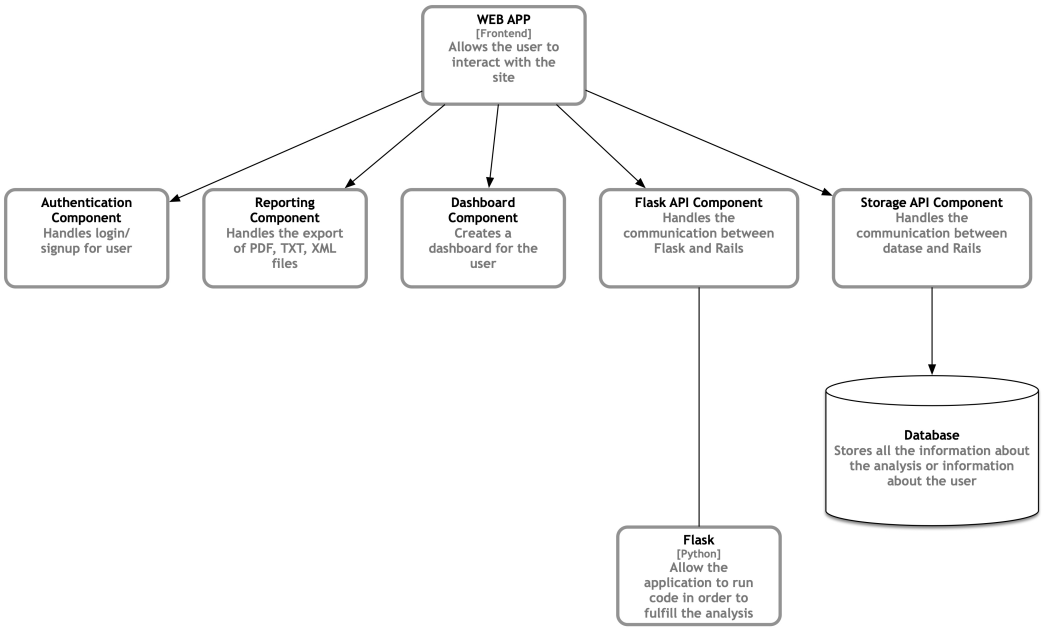

Figure 2.3. *Component diagram*

2.2 Feature model

As a second approach, the Fiure 2.4 shows the core features of the application, showing which of them are mandatory and which of them are optional.

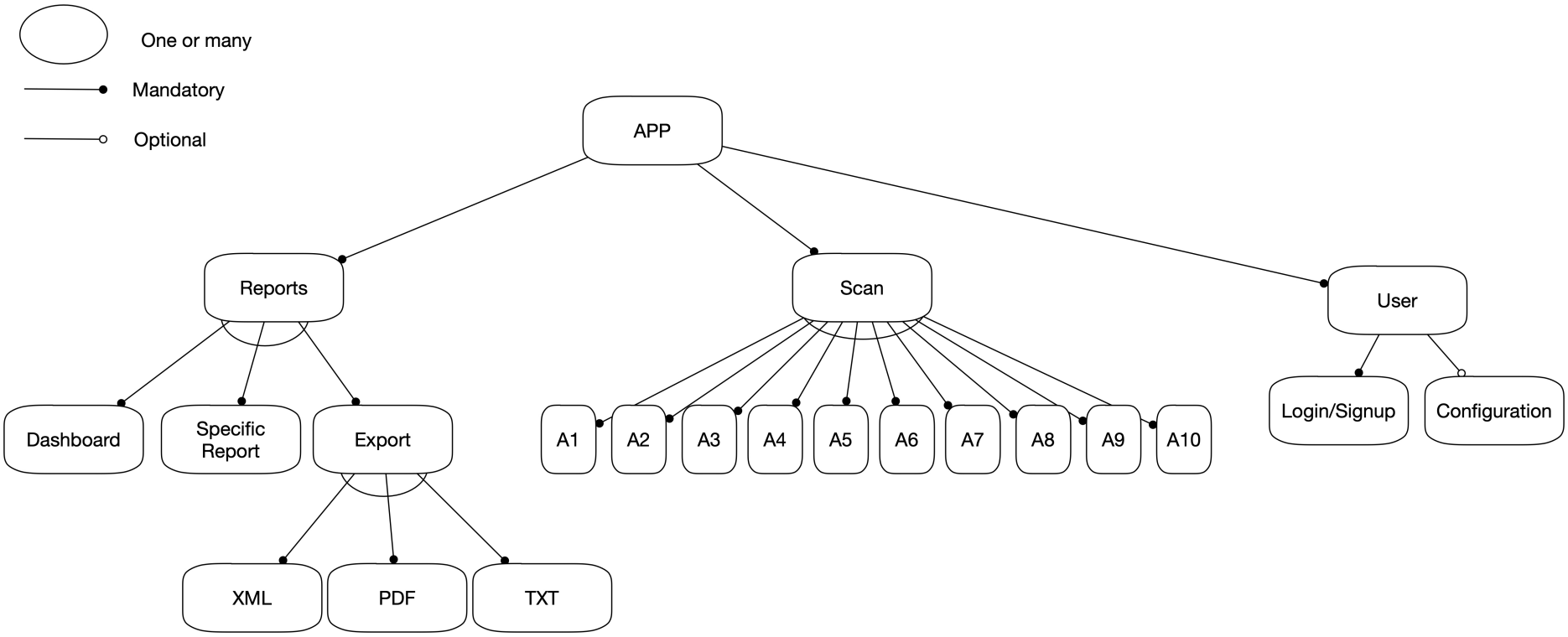

Figure 2.4. *Feature model*

2.3 Squence Diagrams

In the Figure 2.5 the sequence between the user's interaction with the web application and all the steps it has to achieve before displaying new information to the user is shown.

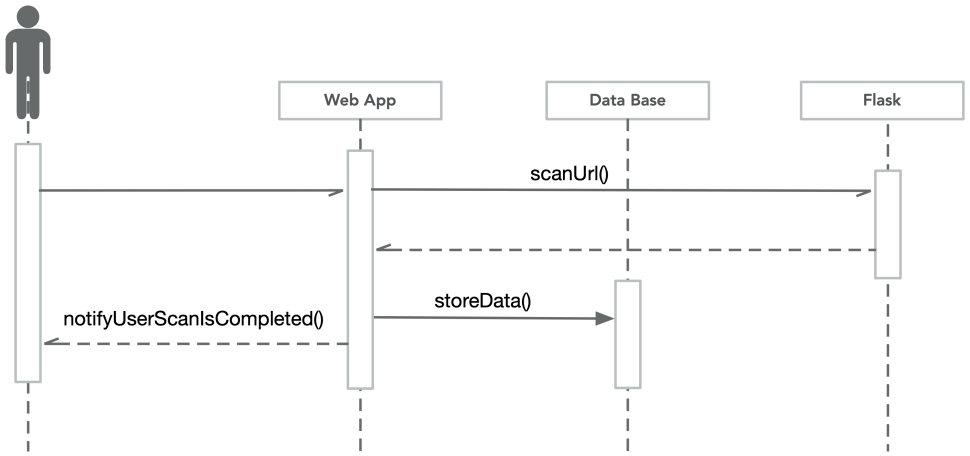

Figure 2.5. *Sequence diagram for scanning a URL*

Another important sequence of the application is how the data is displayed on the dashboard as shown on the Figure A.8

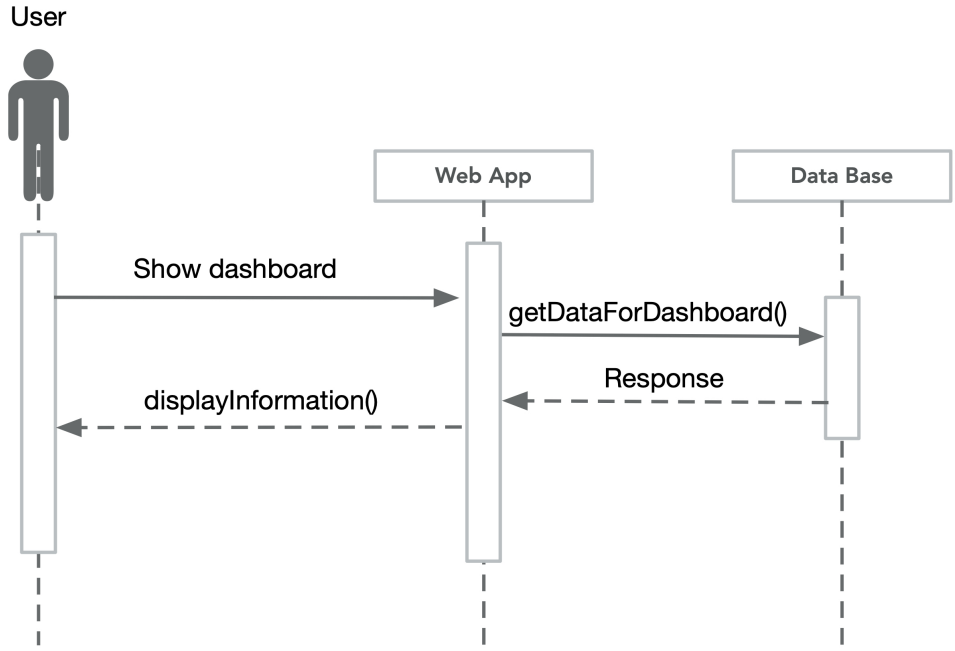

Figure 2.6. *Sequence diagram for the dashboard*

2.4 Database

As shown on the Figure 2.7 it can be appreciated that the database consists on 6 tables, each table has a different type of relationship from one another. Most of the relationships shown on the diagram are a one-to-many relationship, there is just one relationship many-to-many between Analyses and Vulnerabilities.

The tables have the following usage/meaning:

- Users - stores the user's information.
- Analyses - stores each analysis made by a user.
- Vulnerabilities - stores the information of the 10 vulnerabilities that can be scanned, each vulnerability is unique so it cannot be repeated.
- Analysis-has-Vulnerabilities - it is a relationship between an analysis and all the vulnerabilities that have to be scanned for that specific analysis.
- Security-Flaws - stores the data of an specific flaw found on an specific analysis.
- Notifications - stores the information of the notifications shown on the site related to a user.

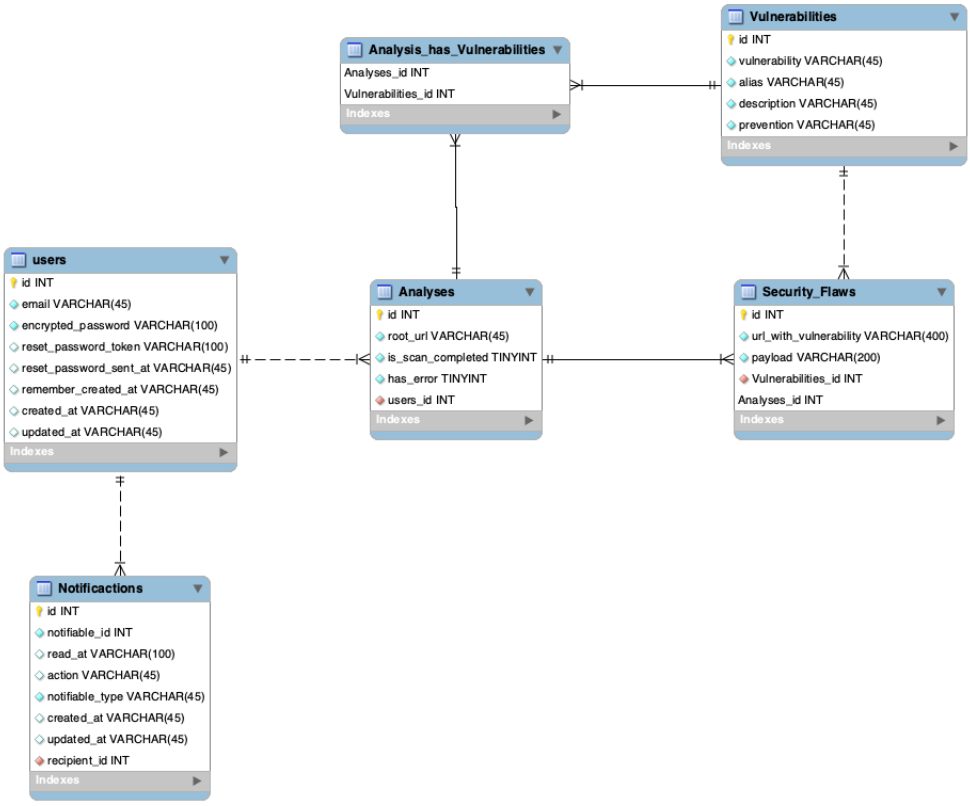

Figure 2.7. *Relational database's model*

Chapter 3

Possible enhancements for version 2.0

During the development of the application we noticed that some features could be improved, the following features were considered to be good features for a possible Version 2.0:

- Cancel an analysis
- Check analysis's status
- Show elapsed time of an analysis
- Show more statistics or graphics

Appendix A

Appendix

A.1 User Documentation

Test and manage your Web Applications security with a few clicks, The WASS software allows you to scan a Web Application for security vulnerabilities, save the results of the scan and generate reports in PDF, XML and TXT formats.

1. Create Account

   1.1. The first step in order to use WASS is enter to http://178.128.186.156:3000 in your favorite browser.

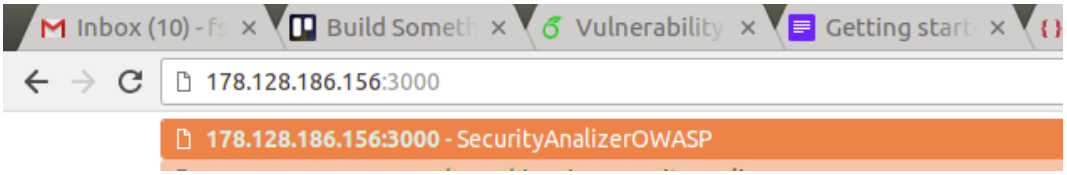

Figure A.1. *WAAS URL*

   1.2. The next step is to create an user account, navigate to the software and you will see a login page

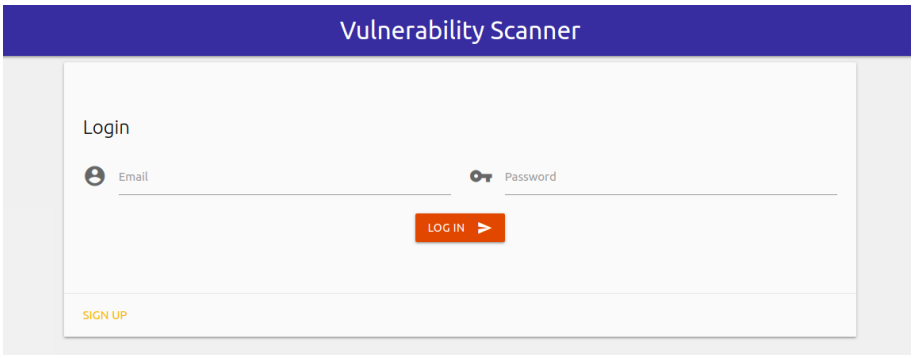

Figure A.2. *Login page*

1.3. Click on sign up

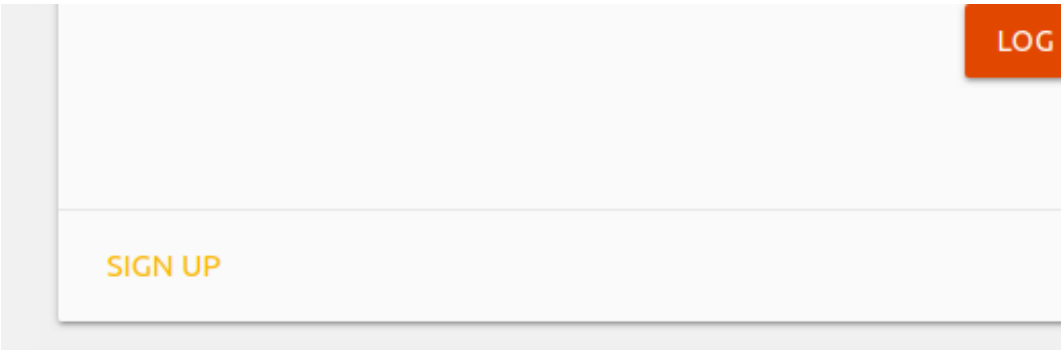

Figure A.3. *Sign up*

1.4. Insert your email, create a password and click on sign up.

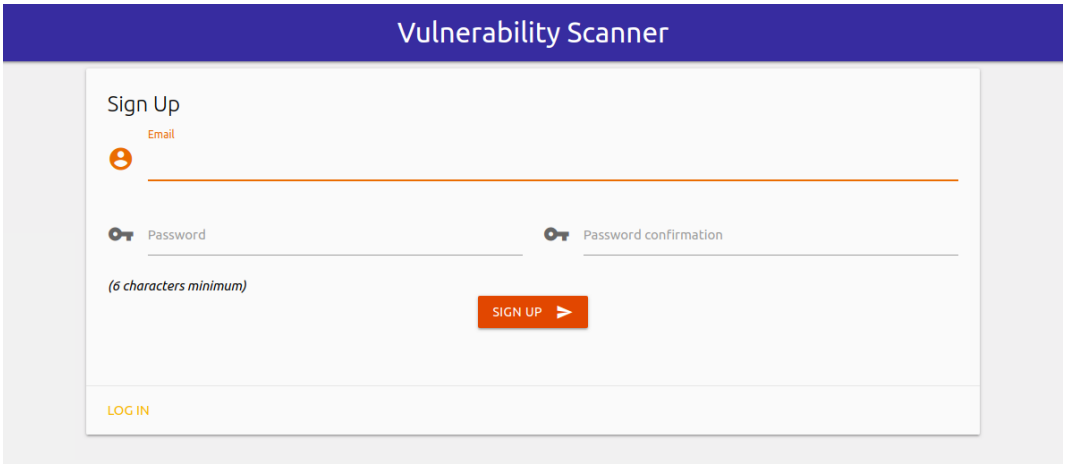

Figure A.4. *Sign up form*

2. Analysing a Web Application by URL

2.1. Once you have your user account, login with your email and password and click on log in button.

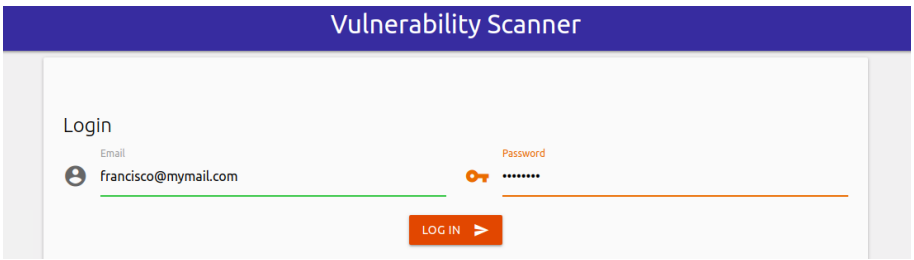

Figure A.5. *Log in form*

2.2. To analyze a web application, write or paste its URL into the URL field. If the web app you want to scan requires authentication provide the auth token.

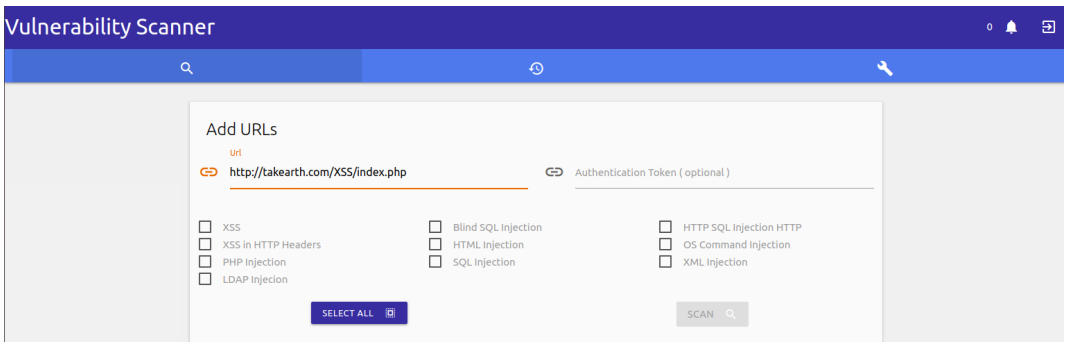

Figure A.6. *Scanner options*

2.3. Then select the vulnerabilities that you want to analyze and click SCAN

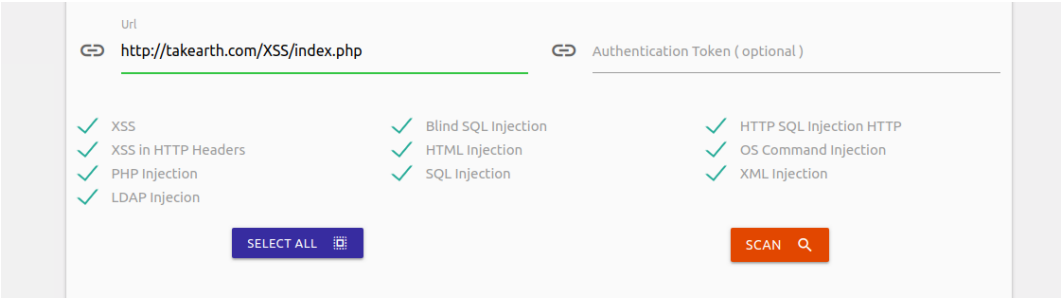

Figure A.7. *Vulnerabilities selected*

You can click on the select all button to enable all the possible scans

2.4. You will be redirected to the dashboard

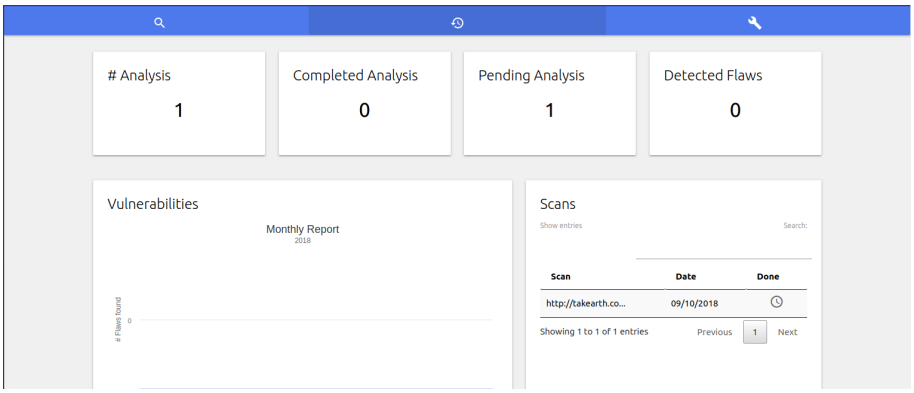

Figure A.8. *Dashboard*

3. Dashboard
   Here, you can see the information and results about previous scans.

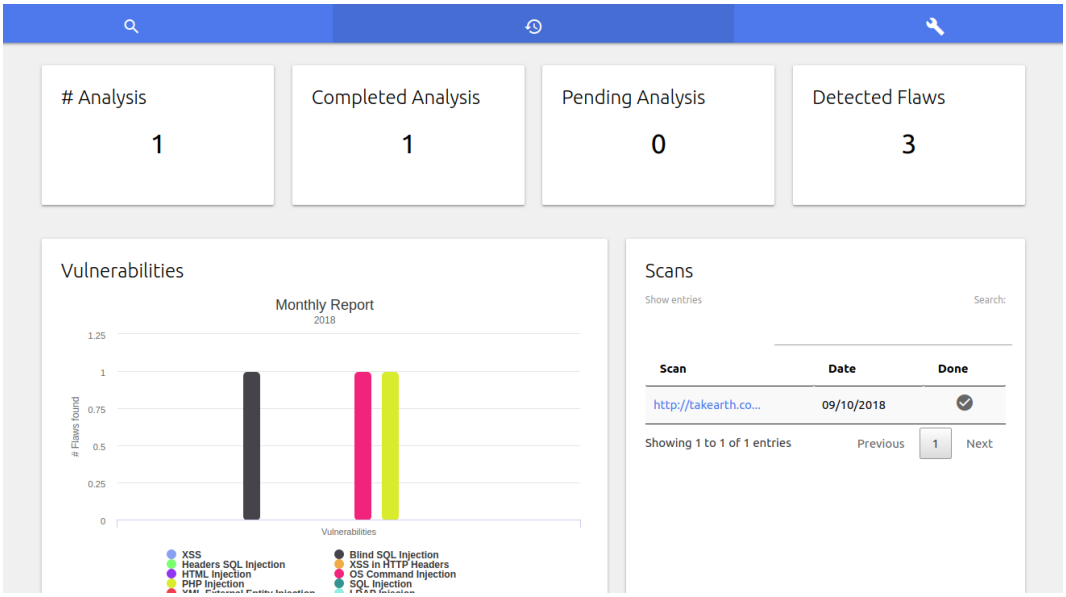

Figure A.9. *Dashboard*

In this page you can check the total number of analysis, the completed analysis, the pending analysis and the flaws detected.

In the right top corner you can see the notifications and the logout buttons

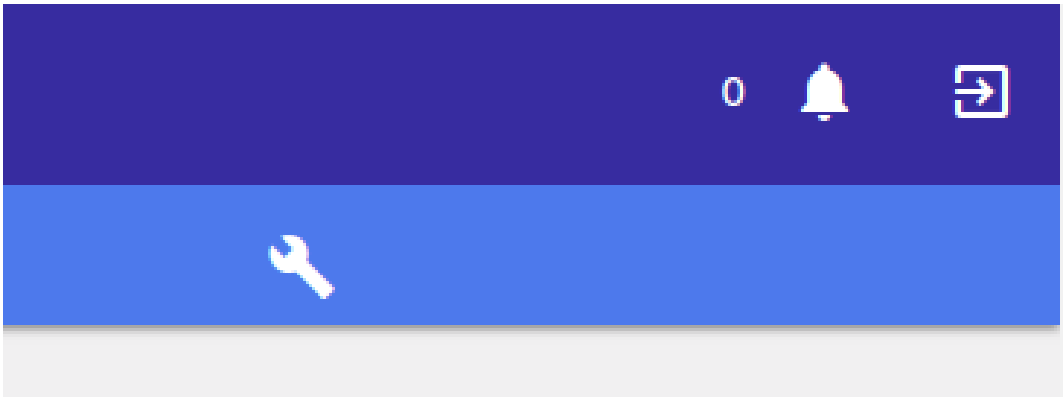

Figure A.10. *Notifications and log out options*

You can see detailed information of any specific scan by clicking the url in the Recent Scans section

Figure A.11. *Recent Scans*

Here you will see at the left the vulnerabilities found on the analyzed web page, on the right we see a detailed description of the vulnerability

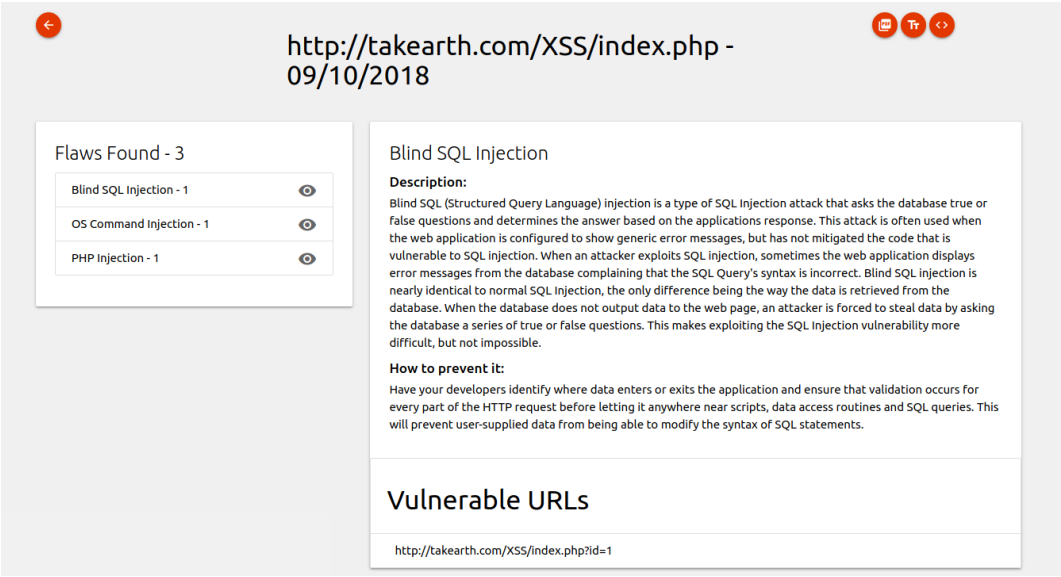

Figure A.12. *Vulnerabilities details*

The number at the right of the flaw represents the number of urls affected by this vulnerability inside the application

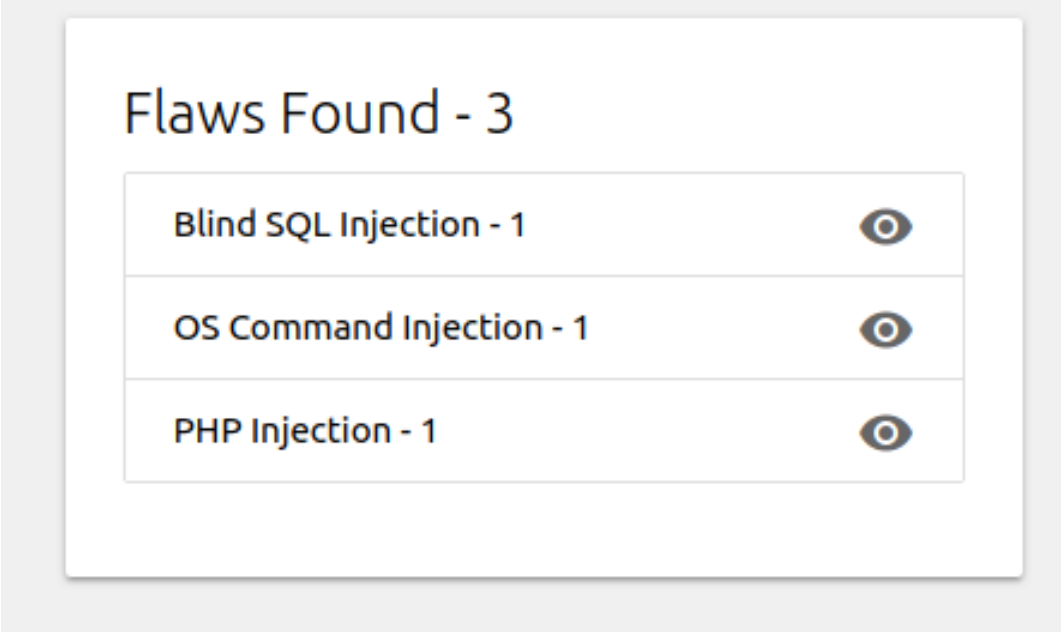

Figure A.13. *Flaws found*

4. Exporting reports
   You are able to export this analysis in PDF, XML and in plain text or TXT, in order to do this click on the icons on the top right

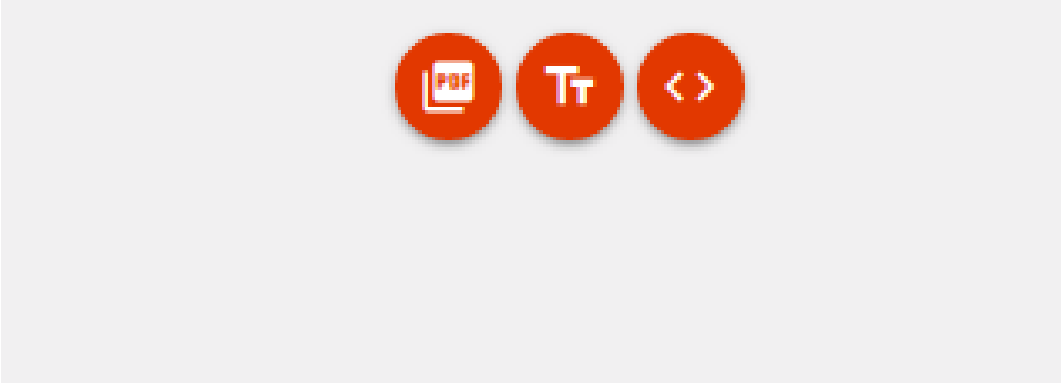

Figure A.14. *Export icons*

When you click on one of the icons a download will start

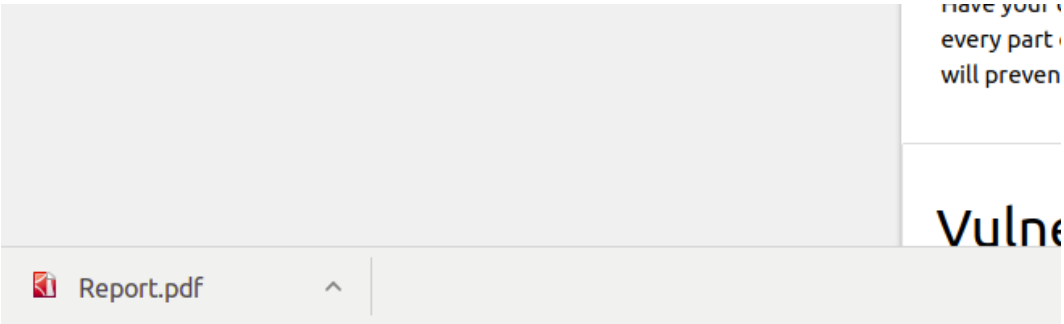

Figure A.15. *File Downloaded*

You should be able to open the file on your prefered reader.

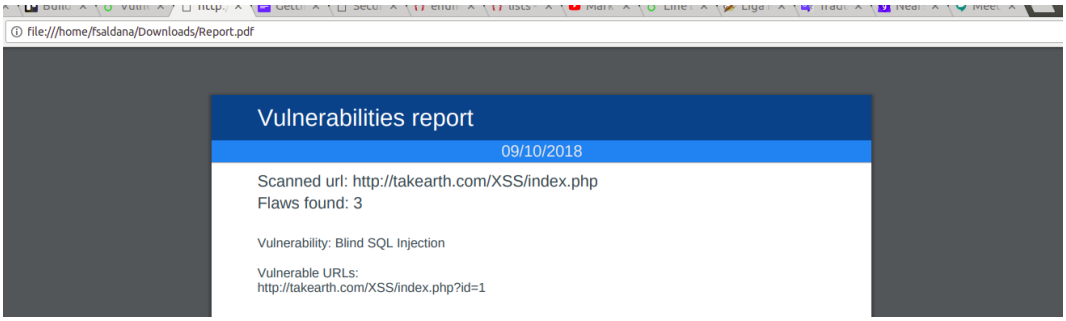

Figure A.16. *Report*

5. Managing your account
   You are able to modify your account email and password, if you want to change one just go to the wrench icon, it's the menu at the right.

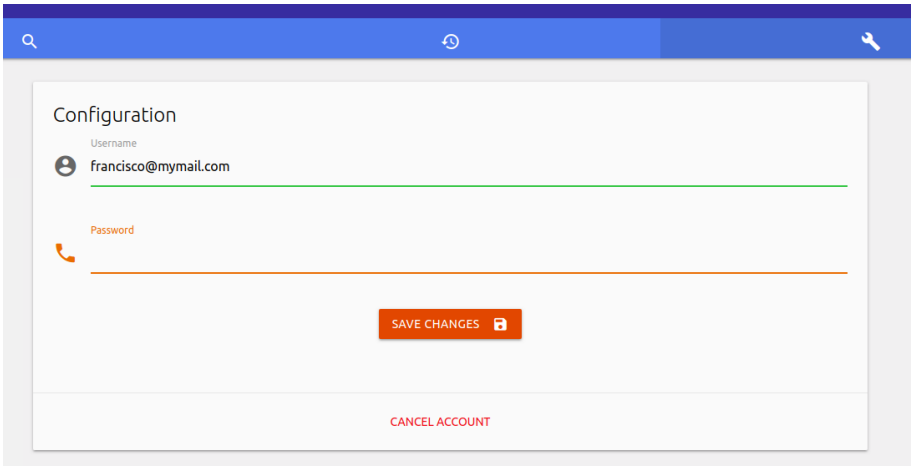

Figure A.17. *Account configuration*

Update your credentials and click on SAVE CHANGES in order to conserve the changes, you will be asked to sign up again in order to continue using the scanner.

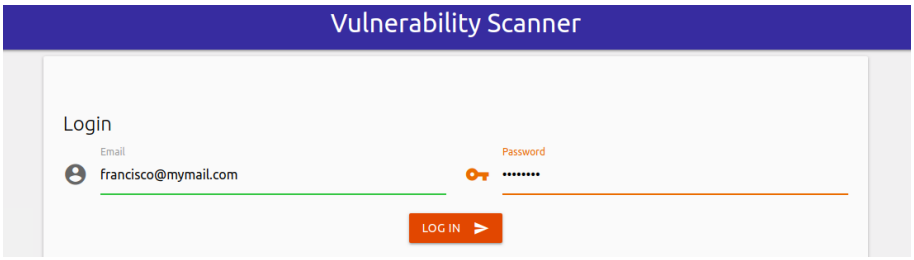

Figure A.18. *Log in*

A.2 Developer Documentation

A.2.1 Installing Docker

Update the apt package index:

```
$ wget http://tex.stackexchange.com
```

Install packages to allow apt to use repository over HTTPS:

```
$ sudo apt-get install apt-transport-https
ca-certificates curl software-properties-common
```

Add Docker's official GPG key

```
$ curl -fsSL https://download.docker.com/linux
/ubuntu/gpg | sudo apt-key add -
```

Set up the stable repository (copy and paste all line in one go)

```
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
   stable"
```

Update the apt package index.

```
$ sudo apt-get update
```

Install the latest version of Docker CE

```
$ sudo apt-get install docker-ce
```

Verify that Docker CE is installed correctly by running the hello-world image.

```
$ sudo docker run hello world
```

Create the docker group.

```
$ sudo groupadd docker
```

Add your user to the docker group.

```
$ sudo usermod -aG docker $USER
```

Restart the machine

```
$ sudo reboot
```

Verify that you can run docker commands without sudo

```
$ docker run hello-world
```

Run this command to download the latest version of Docker Compose:

```
 $ sudo curl -L "https://github.com/docker/compose/releases
/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Apply executable permissions to the binary

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Test the installation

```
$ docker-compose --version
```

A.2.2 Cloning the repositories

Create a new directory with any name you like

```
$ mkdir VulScanner & cd VulScanner
```

Cloning and building the python server

Clone the python server repository inside the folder

```
$ git clone https://github.com/gmotzespina/VulnerabilityScannerPython
```

Move inside the new folder

```
$ cd VulnerabilityScannerPython
```

Run the following command to build the docker image, this is a necessary step in order to run the the ruby client as it depends on this image

```
$ docker build -t python-scanner .
```

Note that is has a dot at the end

Cloning the rails app

Move one directory back and clone the ruby app

```
$ cd ..
$ git clone https://github.com/gmotzespina/VulnerabilityScanner
```

Move inside the new directory

```
$ cd VulnerabilityScanner
```

Build the docker image

```
$ docker-compose build web
```

Run the rails migrations and database seeding

```
$ docker-compose run --rm web rails db:create db:migrate
$ docker-compose run --rm web rake db:seed
```

Run the rails server, this will automatically start the db server and the python server

```
$ docker-compose up web
```

A.2.3 Utilities

Show mounted images

```
Docker ps -a
```

Bibliography

[1] The OWASP Foundation. 2018. URL: https://www.owasp.org/index.php.

[2] Simon Brown. *The C4 model for software architecture*. URL: https://c4model.com.