ELRO-Security

Eliran Maman & Royi Hamo

By:
Eliran Maman & Royi Hamo

Supervisor:
Nethanel Gelernter

Git:
WAF (Main System): https://github.com/eliranmaman/ELRO-Security-Project
Web Application (GUI): https://github.com/royiHamo/ElroWebApp

Table of Contents

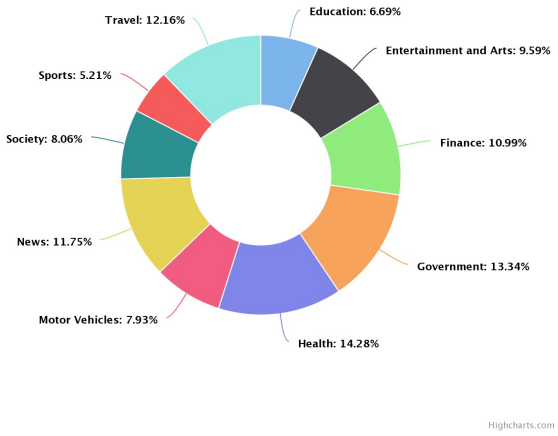
1. Project description

When we visit a website, there are some concerns about the reliability of the website,
In fact, we are exposed to a several number of vulnerabilities that we are not aware of.

In addition to that the server side may be exposed as well to some vulnerabilities,
such as SQL injections, Brute Force, XSS attacks, etc.
Resulting in a situation where all the website clients are now at a potential risk to future attacks and leakage of private information.

In our world, the web privacy and security are a very important matter, and it seems that not all the web developers are aware of the consequences of unsecured development,
And even worse there are cases where the developers think that their code is as secured as possible, and not aware that they are using unsecured functions, for example using scanf instead of scanf_s.

In fact, on average 30,000 new websites are hacked every day. For example, SQL Injection Attacks: So Old, but Still So Relevant. Here's Why



The Goal: Increase the web privacy and security among clients and service providers, that cannot afford highly expensive products, and are still aware of the risks online. We would like to prevent common server and client attacks.

2. Related Work

Our project is all about improving the web security and privacy experience for both clients and service providers. Today in most websites there are a lot of vulnerabilities, from SQL Injection to leak of information. Most of these vulnerabilities can be easily blocked or at least can be hardened to exploit.

This project will affect a wide range of websites worldwide, in one hand it will make the client experience safer and on the other hand it will make the service provider more secured.

3. Functional Description / Requirements

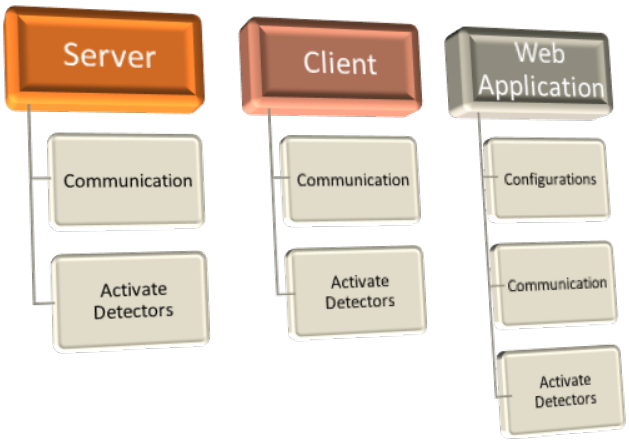
Our project will include a web application (UI), that will provide screens and functionality to control the **core system** service, and to inspect the statistics. In addition, the project will include the core system that contains the Database, Proxy server and
all the detectors modules.

3.1 Core-System

3.1.1 Core-System Users

The core system will provide services for several kind of users as follows:

a. **Web Application:** The Project includes web application, and we infer as a user of the core system, for example ask for data, update configurations, or ask to scan website's address.
b. **Client:** one of the guests that asks to communicate with one of our protected servers, every request will initiate the core system logic that will determine the eligibility of the request, and decide whether to transfer the response to the server or not.
c. **Server:** one of our customers' servers that we provide service to. That will communicate with the core system and we will test its responses eligibility and decide whether to transfer the response to the client or not.



3.1.2 Database

We will store the data in a SQL based database, which will include the following tables:

a. **Detectors Requests Data:** will store all the request that passes through the system, with several flags to help identify the destination, and filter the data. if response was blocked it will include the detector that blocked it.
b. **Detectors Data Responses:** will store all the responses that passes through the system, if response was blocked it will include the detector that blocked it.
c. **Cookies Token:** will store the cookies token's that are used by the cookie poisoning detector.
d. **Users (Web Application):** will store all the credentials of the registered web application users, permissions, etc.
e. **Servers:** will store all the users' servers' configurations such as IP, DNS, etc.
f. **Services:** will store all the configurations of users for each of their servers, such as which detectors will be on and off.
g. **Brute Force Data:** This table will contain all the info needed to detected brute-force attempt.
h. **Blacklist:** will store all the banned IPs that we immediately block requests from.
i. **Whitelist:** will store all the certified IPs that we immediately approve requests from.

3.1.3 Modules

3.1.3.1 Detectors

i. **SQL Injection:** will be responsible to detect SQL injections attacks.
ii. **XSS Injection:** will be responsible to detect XSS injections attacks.
iii. **XML Injection:** will be responsible to detect X injections attacks.
iv. **Brute Force:** will be responsible to detect brute force attacks.
v. **CSRF:** will be responsible to detect CSRF attacks.
vi. **User Protector:** will be responsible to protect the user from server attacks.
vii. **Bots:** will be responsible to detect if the user is legitimate.
viii. **Proxy Detector:** will be responsible to detect requests that arrive via proxy.

3.1.3.2 Controller

The controller will decide whether to transfer the request
to its destination or to block the request in according to findings of his research (applying all the detectors + logic).
(*graphic description below)

3.1.3.3 DB Agent

The DB Agent will be responsible for communication tasks with the database, in both directions whether to ask for data or to store new data in.

3.1.3.4 API Agent

Will be responsible to listen to requests from the web application, to do actions and provide data as response.

3.1.3.5 Parser

Will be responsible to parse the data and translate the request to data that we can work with.

3.1.3.6 HTTPS Enforcer

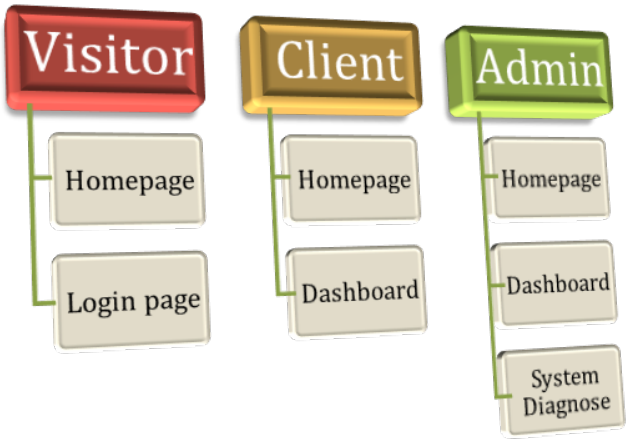
Will be responsible to check if the request is secured (HTTPS).

3.2 Web Application

3.2.1 Users & Permissions

Everyone has access to the Home Page, even unknown visitors. And in addition to that there are registered clients who are known to the system.

a. **Visitor (not logged in user):** The first page that everyone see is the home page (this is the only screen the not-login user has access to).
b. **Client (logged in user):** Once a Client logs in, in addition to the visitor's permissions he has access to his Dashboard, in which he can get more information from, such as Statistics, Control system features, edit information.
c. **Admin:** Has access to the Admin's Dashboard page, and he has all the permissions.



3.2.2 Home Page

The Home Page is the only screen that available to everyone (all “types” of users have access to it). Therefore, this screen is very general and does not include any personal data.

The Home page include general information about the system, contact information, information about awareness to web-application security and privacy, and a tool to rate a website's security.

3.2.3 Client Dashboard Page

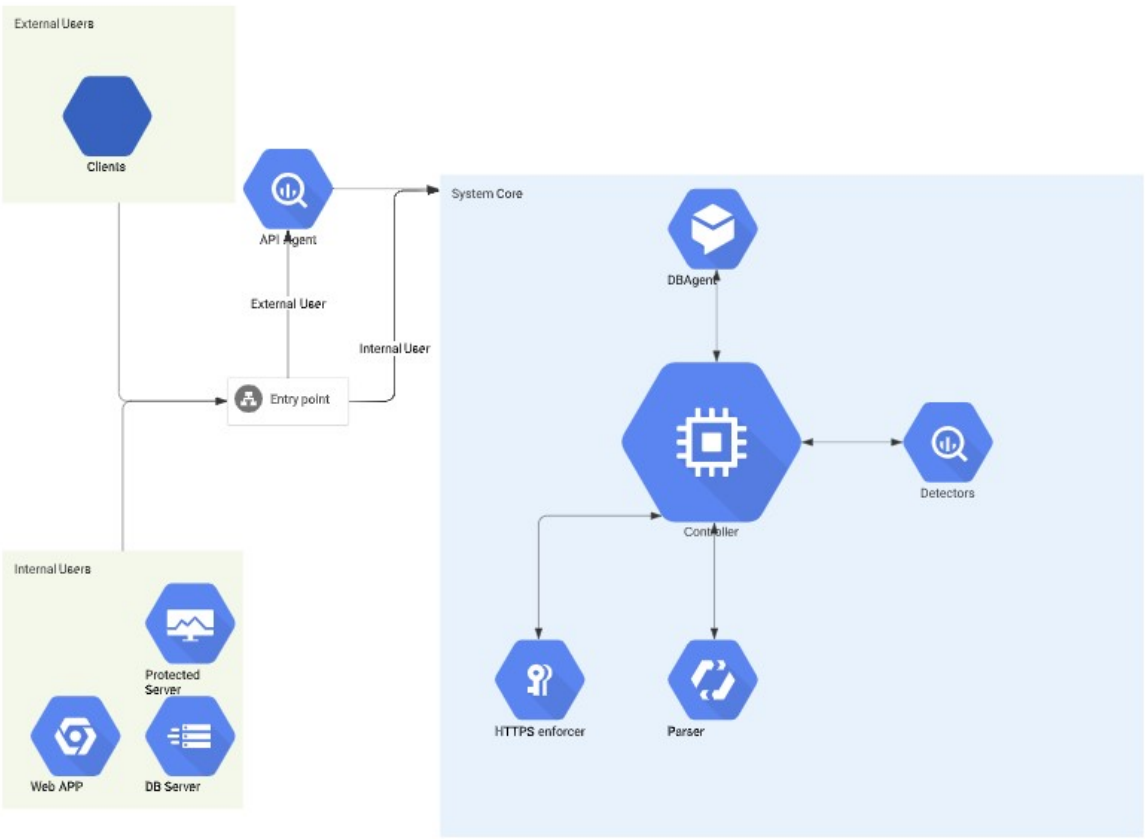
The Client Dashboard Page is where the client will see all the Information about his account, such as statistics, the features he would like to enable/disable, and option to edit his personal information.

3.2.4 Admin Dashboard Page

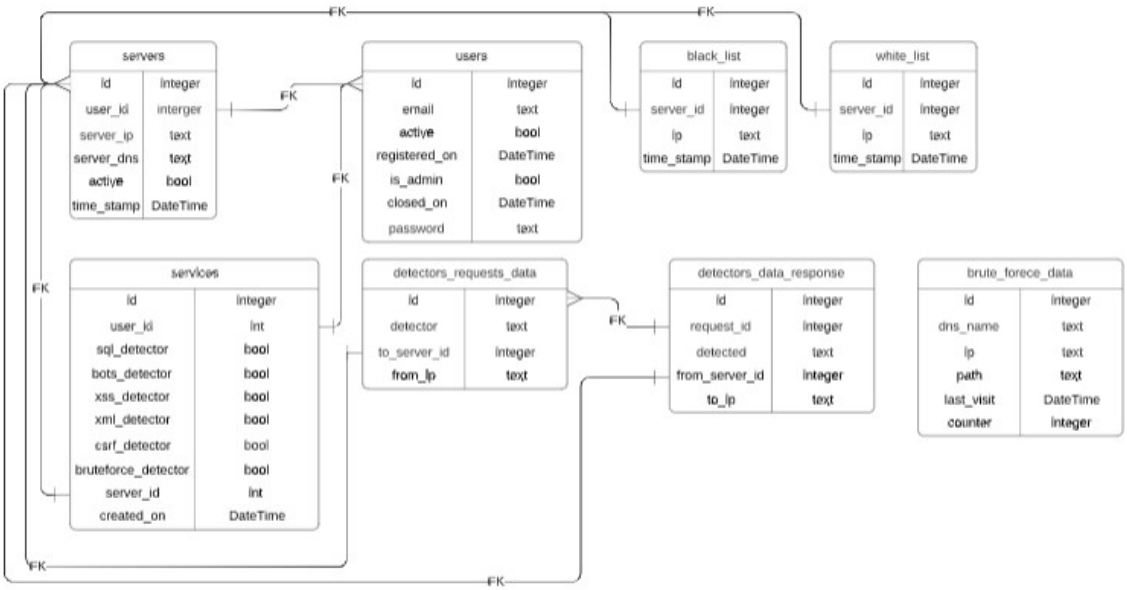
The Admin Dashboard Page is where we will see all the Information about the system and the clients, such as statistics, the features we would like to enable/disable, and manage clients.

4. Architecture

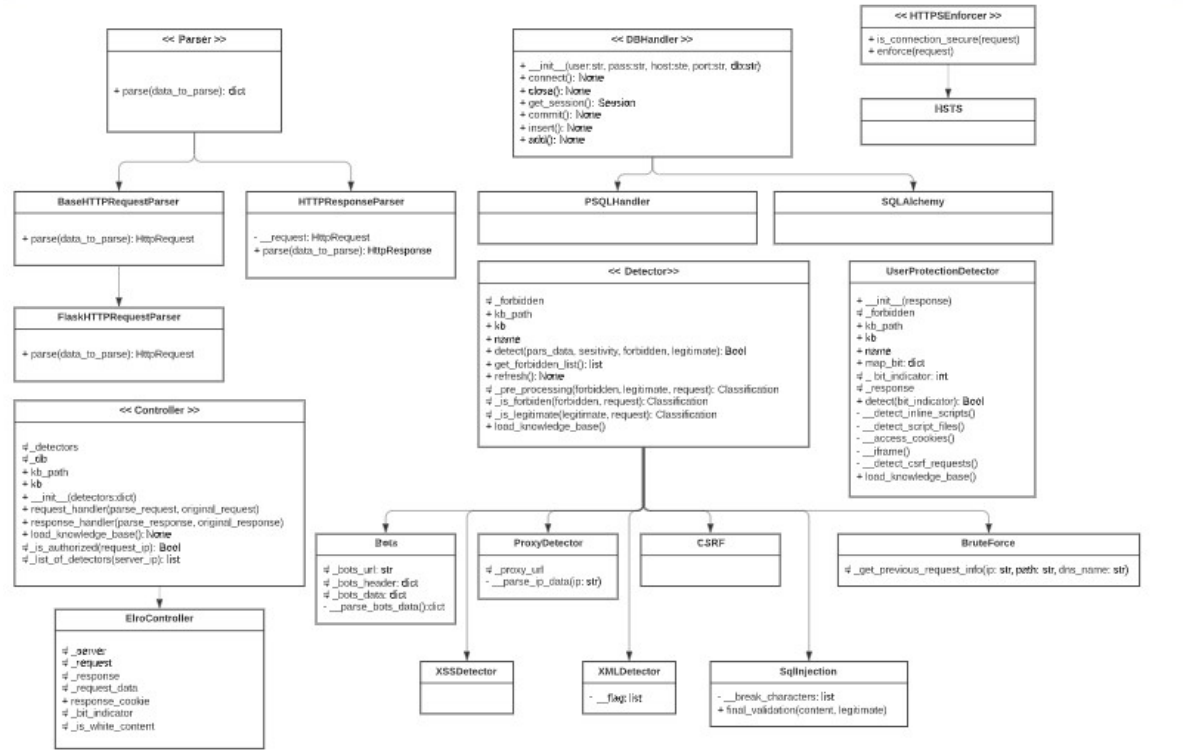
4.1. Overall System Architecture



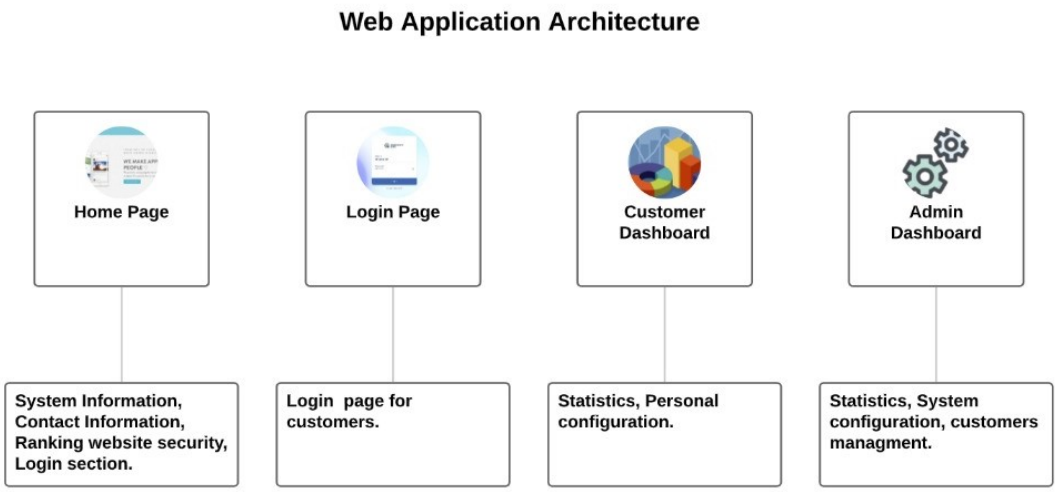
4.2. Database Architecture



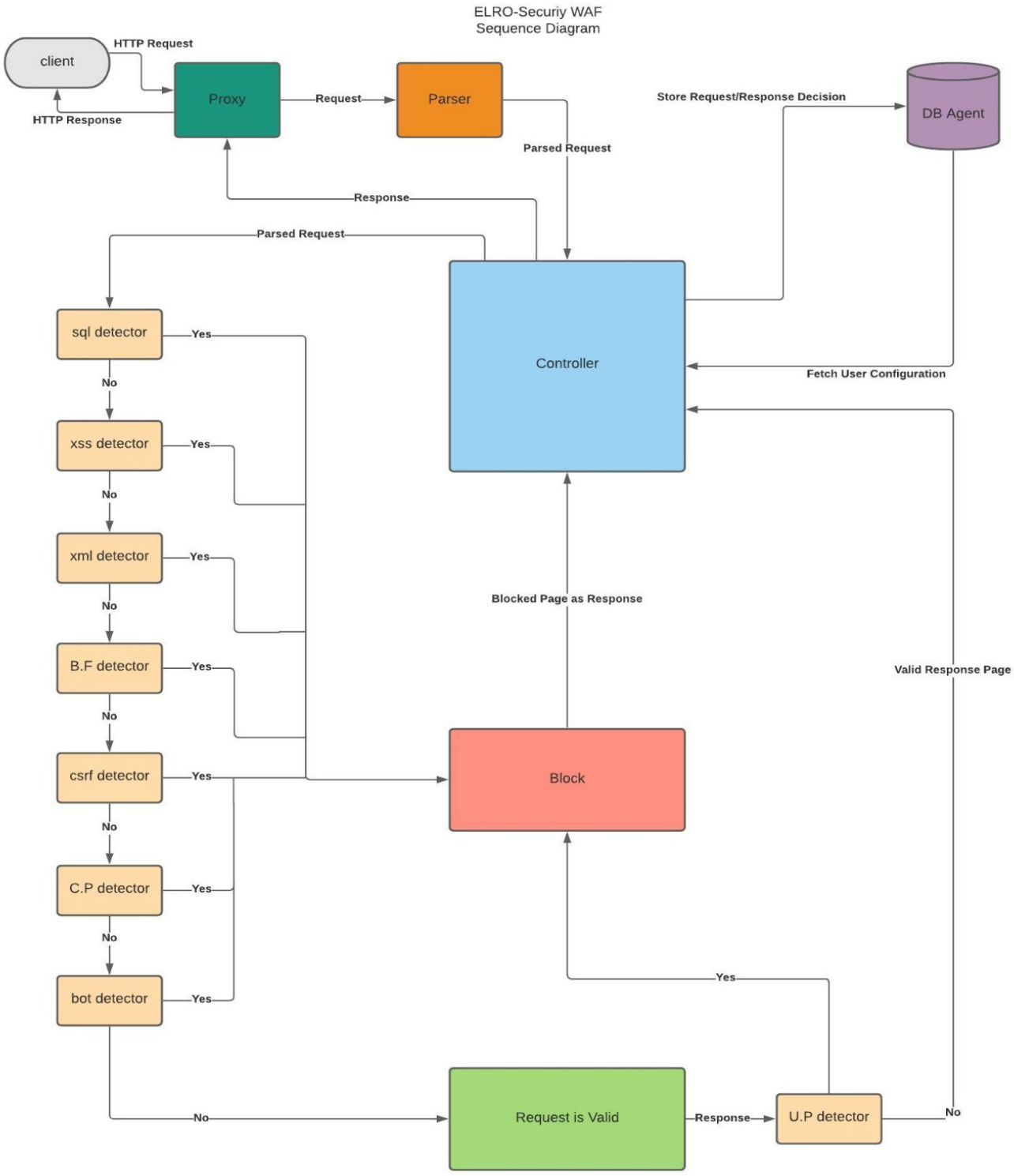
4.3. Core System Architecture



4.4. Web Application Architecture



4.5. Controller Sequence Diagram



5. Work plan

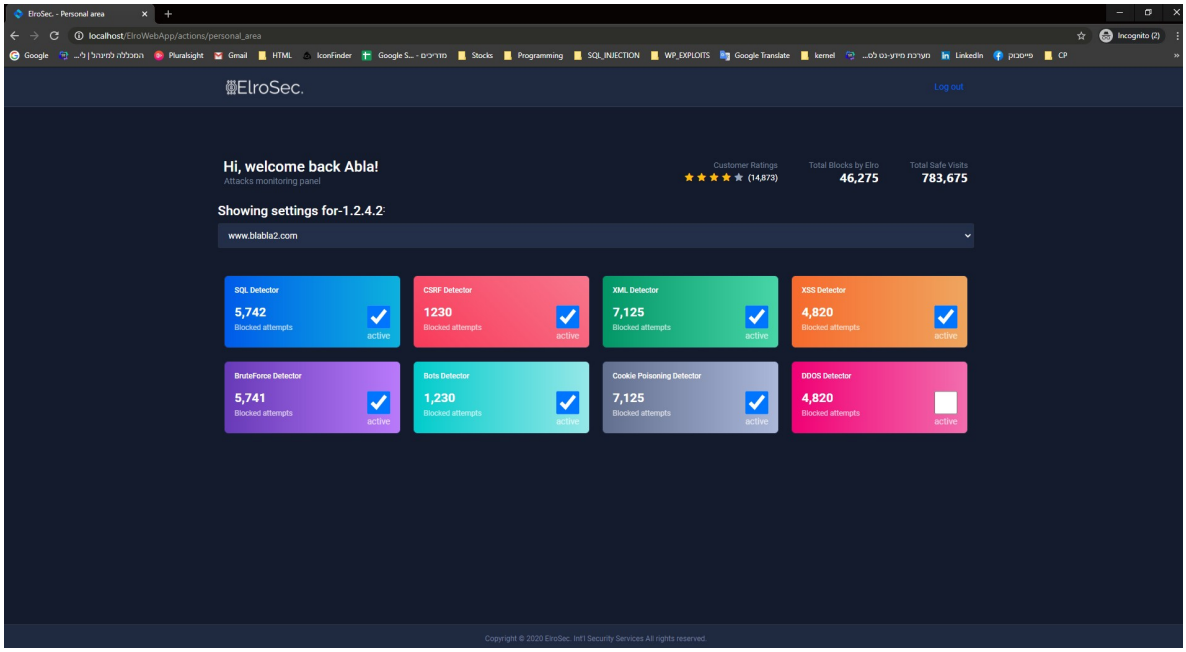
Module	Time	Responsible	Status
Database	70 Hours	Royi	Done
Server Set-up	70 Hours	Eliran	Done
Web Application	150 Hours	Royi	Done
User Filtering + BF Modules	90 Hours	Eliran	Done
Injections Modules	150 Hours	Royi	Done
CSRF + Proxy Detector	70 Hours	Eliran	Done
User Protection + HSTS	60 Hours	Eliran	Done
Controller	100 Hours	Eliran	Done
API Agent	150 Hours	Royi	Done
Final Adjustments	50 Hours	Royi + Eliran	Done
QA	150 Hours	Royi + Eliran	Done
Presentation	30 Hours	Royi + Eliran	Done
Production	20 Hours	Royi + Eliran	Done

6. Client side

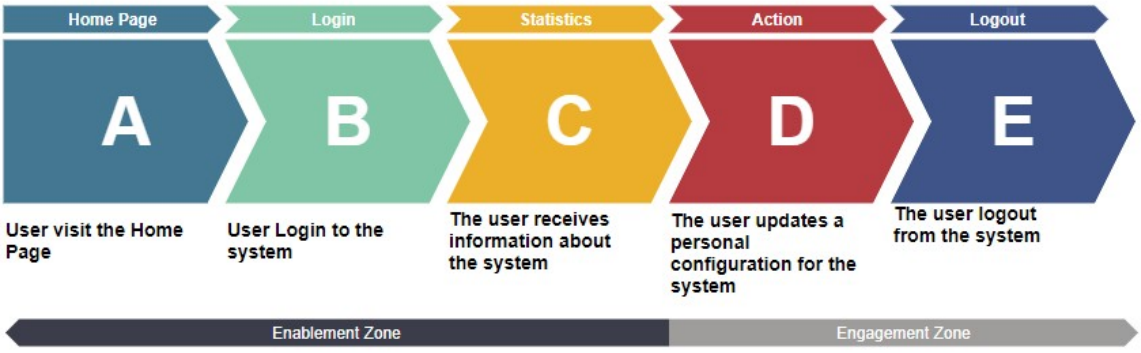
This section describes the looks and the operation of the client UI:

6.1. Graphic illustration for the home page

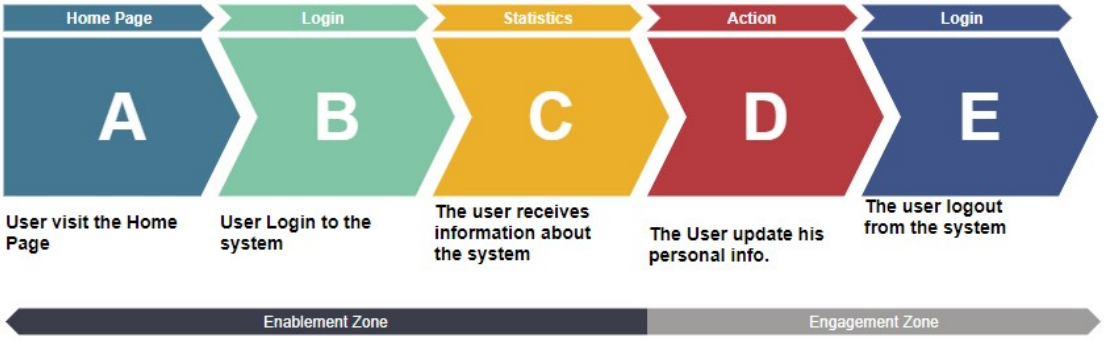
6.2. Graphic illustration for the Dashboard



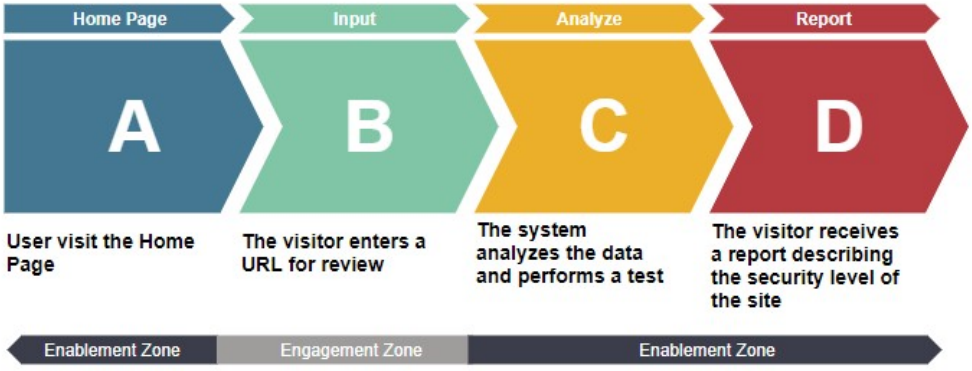
6.3. User System Configuration



6.4. User Personal Configuration



6.5. Security Rank for websites



7. Server Side

7.1. API

API	Method	Usage	Description
/getRank=url	GET	External	Get website security rank of url
/login	POST	Internal	Login user
/register	POST	Internal	Register new user
/getActiveServices	POST	Internal	Get server active services
/getUserData	POST	Internal	Get user information
/updateServicesStatus	POST	Internal	Update server services status.
/adminUpdateServicesStatus	POST	Internal	Admin update services for all the servers
/addNewWebsite	POST	Internal	Add new server
/userProtection	POST	Internal	Check website with user protection detector