Team Name
Bug Slayers

Project Proposal
CS416M: Foundations of Network Security and Cryptography

Due Date
10/02/21

Amol Shah	**Saavi Yadav**	**Shreya Laddha**	**Tezan Sahu**
18D070005	170020003	180070054	170100035

WAVS: Web App Vulnerability Scanner

A Virtual Server to Assess Vulnerabilities & Security Features of Web Apps

1 Problem Statement

The aim here is to develop **WAVS (Web App Vulnerability Scanner)**, a tool to scan & test URLs for certain vulnerabilities & security issues by simply inspecting the corresponding client-side website. The overall system would include a virtual server with modules for detecting the different vulnerabilities, along with a proxy server, to direct requests from a browser to the virtual server first while visiting a website. The proxy could warn the user before redirecting to the website if some vulnerabilities are found during the scan done by our virtual server.

We intend to identify & assess the following classes of vulnerabilities that a website may possess:

- Absence of Valid TLS Certificates
- Cross-Site Scripting (XSS)
- Potential Phishing Attempts
- Open Redirection

If time permits, we would also create a cross-platform app to test URLs for security features through our virtual server so as to keep the user's device out of harm.

Connection with the Course Content

- This project lies majorly in the domain of Application Security, which is an integral part of the course
- By building WAVS, we try to explore the various web app vulnerabilities and create modules to identify them to protect the user
- We also intend to gain some practical experience with current network security tools like Metasploit along with theoretical background

2 System Architecture

The proposed system architecture of WAVS, with all the necessary components, can be found in Figure 1. Below, we also discuss the approaches that we intend to follow to develop the various components of WAVS.

Proposed Implementation

Virtual Server

We plan to implement each of the vulnerability identification components of WAVS virtual server as a separate microservice & integrate them with the virtual RESTful API server. Each of these microservices can be enabled or disabled based on the preferences indicated by the user in the HTTP request. *(default being a **Full Scan**)*.

We suggest the following approaches to be taken for identifying each of the different security issues:

- **Absence of Valid TLS Certificates:** To detect the presence & validity of TLS certificates for a given URL, we propose to use necessary libraries to obtain the chain/hierarchy of certificates for the website & validate each of them while checking for their expiry.

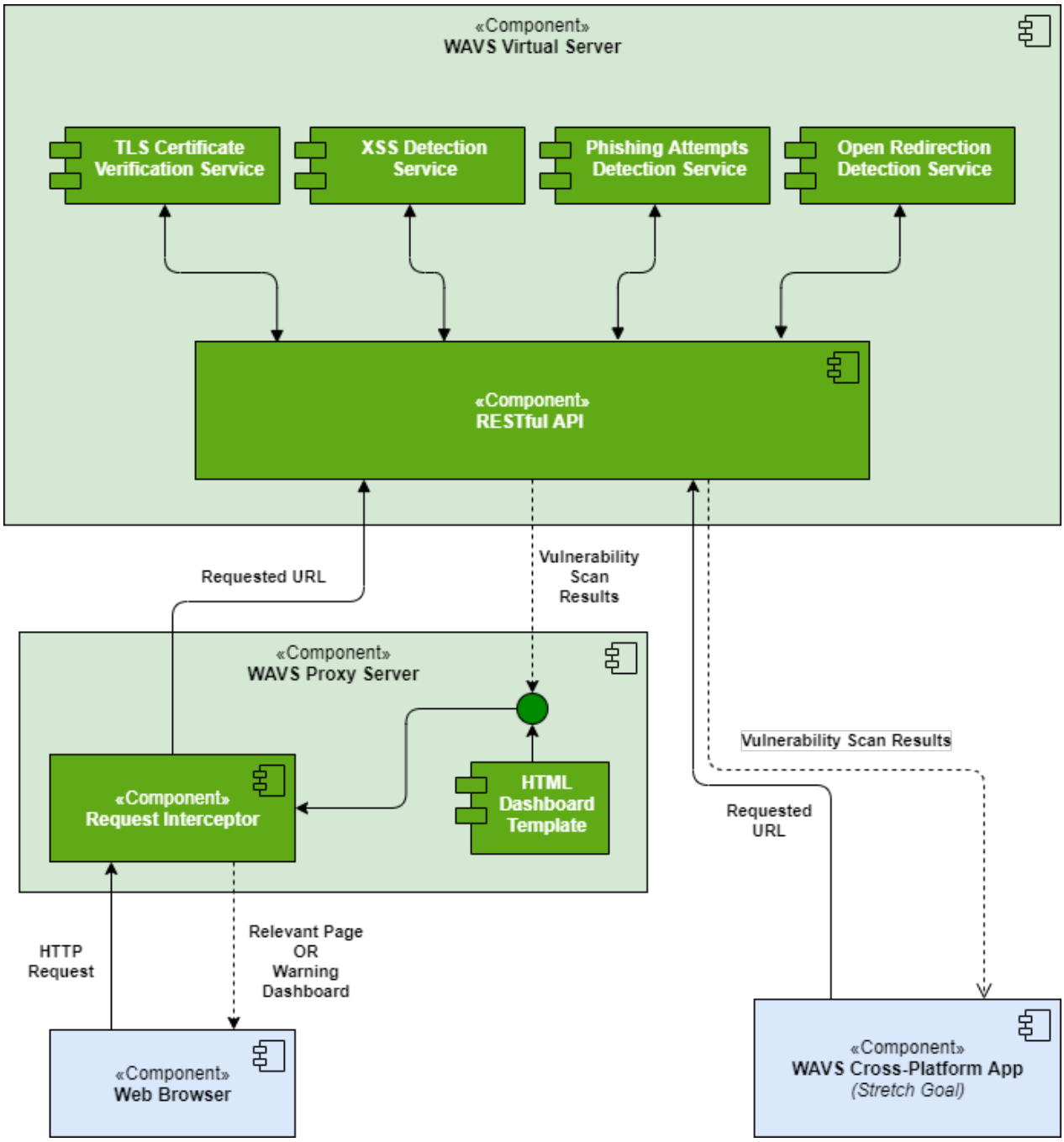

Figure 1: Overall System Architecture of WAVS

- **Cross-Site Scripting (XSS):** We propose to test XSS vulnerabilities by calling various functions to inject XSS payloads into DOM elements like forms & detecting whether the webpage is vulnerable or not.
- **Potential Phishing Attempts:** We intend to create a lightweight **rule-based** (or traditional ML-based) **classifier** using relevant *page keywords*, *address bar*-based, *HTML/JS*-based & *domain*-based features from the website (inspired from the technique developed in this paper. Later, when a URL is sent for scanning, all the required features for the webpage will be extracted & run through our classifier to identify the website as *phishing* or not.
- **Open Redirection:** To address open redirection issues in websites, we propose to work on the lines similar to Oralyser & ORtester. This involves infusing predefined payload URLs into the given website URL & trying to figure out *Header-based*, *JS-based* & *Meta tag-based* open redirect vulnerabilities.

Proxy Server

The proxy would be created to intercept requests made from web browsers (by the user), test them for potential vulnerabilities by sending the URL to our virtual server, redirect the requests if the URL is secure & warn the user accordingly in case some security issues are detected. The warning will be issued in the form of a dashboard created by injecting the relevant information about the detected vulnerabilities (from the virtual server) into an HTML template, which will be served to the user.

Libraries/Packages To Be Used

Following is a list of major libraries that we would leverage to implement the various components of WAVS:

WAVS Component	Language	Major Modules/Packages Involved
TLS Certificate Verification Service	Python	`check-tls-certs`, `OpenSSL`
XSS Detection Service	Python	`fuzzywuzzy`, `BeautifulSoup`, `splinter`
Phishing Attempts Detection Service	Python	tldextract, `BeautifulSoup`, `gensim`, `scikit-learn`
Open Redirection Detection Service	Python	`requests`, `BeautifulSoup`
Proxy Server	NodeJS	`node-http-proxy`

3 Choice of Platform & User Interface

- **Virtual Server:** We plan to use **FastAPI** (a python framework for building RESTful APIs) for developing the virtual server, along with other **python packages** (mentioned previously) for a variety of sub-tasks. The python code for assessing each of the different vulnerabilities will be suitably modularized into microservices & integrated into the server. This server is expected to be deployed on **Heroku**.
- **Proxy Server:** The proxy server would be created using **NodeJS**, to redirect web traffic from a user's browser to the virtual server for scanning the URL for potential threats.
- **App (Stretch Goal):** The cross-platform app for scanning URLs & displaying the vulnerability analysis of these URLs would be created using **Flutter SDK**.

4 Deliverables of the Project

Minimum Guaranteed

- A **virtual server** capable of identifying the following **client-side security issues**, given website URLs:
  - Absence of Valid TLS Certificates
  - Cross-Site Scripting (XSS)
  - Potential Phishing Attempts
  - Open Redirection
- A **proxy server** to redirect web traffic from a user's browser to our virtual server for scanning the URL for potential threats. This would automatically redirect to the website if no vulnerabilities are found. Otherwise, it will present suitable warnings on the user's browser.
- A **full-fledged working demo** of website vulnerability detection using WAVS virtual server & proxy
- A **presentation** about the nature, sources, impacts & detection techniques of the security issues targeted

Additional Desirable Goals

- Understand the working of **Metasploit** framework for vulnerability assessment, thereby comparing & contrasting their implementations with our vulnerability detection modules.
- Integrate modules that can identify other web vulnerabilities like **Cross-Site Resource Forgery (XSRF)**

Bonus/Ambitious Features

- Design a cross-platform app to allow users to test URLs for security features & vulnerabilities through our virtual server & display a comprehensive analytics dashboard for the various vulnerabilities detected.

5 Methodology

Learning Needed & Effort Estimate

TLS Certificate Verification Service

- Gain background knowledge about TLS certificates, the security they offer & ways to detect their absence
- Familiarize with the working of available CLI python tools like `check-tls-certs` & use a similar approach to develop the service for checking the existence & validity of a website's TLS certificate.
- **2-2.5 weeks** of work expected to create the standalone TLS certificate verification microservice

XSS Detection Service

- Study about XSS in more detail and review typical categories of XSS attacks
- Implement the XSS payloads & inject them into DOM elements that are susceptible to Client XSS attacks
- **3-4 weeks** of work expected to develop the microservice for detecting XSS

Phishing Attempts Detection Service

- Delve deeper into phishing attacks & study the **paper** *Intelligent rule-based phishing websites classification* to understand the **relevant features** to identify phishing websites
- Use the UCI Phishing Websites Dataset to train & test some **traditional ML classifiers** (DT, SVM & RF)
- Identify the **most important features** indicating phishing & build **functions to extract them**
- **3-4 weeks** of work expected to come up with final deployable feature extractors & a classifier

Open Redirection Detection Service

- Understand the **fundamentals** of Open Redirection vulnerabilities in websites & their potential impact
- Explore **existing tools** to detect these & emulate the **most convincing techniques** to build the service
- **2-2.5 weeks** of work expected to create this service to detect open redirections using a list of payloads

Virtual Server

- Familiarize with **FastAPI** & **Heroku** for developing & deploying the virtual server
- **1-1.5 weeks** of work expected to set up the server & integrate the various microservices

Proxy Server

- Familiarize with `node-http-proxy` for developing the proxy server
- Learn how to inject relevant information from JSON objects into DOM elements of an HTML template
- **1-2 weeks** of work expected to develop the proxy

Comparison with Metasploit

- Learn to use Metasploit and compare its working with the various modules that we would create
- **2-3 weeks** of work expected to gain hands-on experience with Metasploit

Roles

Following is the division of roles among the team members for accomplishing the objectives of this project:

Name	Tentative Task Allocation
Amol Shah	XSS Detection Service, Comparision with Metasploit
Saavi Yadav	TLS Certificate Validation Service, Virtual Server
Shreya Laddha	Proxy Server, Open Redirection Detection Service
Tezan Sahu	Phishing Attempts Detection Service, Virtual Server

Note: *This allocation is only tentative & modifications could be made during the course of the project as deemed necessary, based on the respective workloads.*

6 Summary

What will our team learn by doing this project?

- Through this project, we will be able to understand and analyze some of the **OWASP** top 10 vulnerabilities. The vulnerabilities which we will be tackling in depth include Absence of Valid TLS Certificates, Cross-Site Scripting (XSS), Potential Phishing Attempts and Open Redirection.
- Analyzing vulnerabilities will help us gain meaningful insight into client-side security issues. Also, implementing the modules from scratch will allow us to get in the head-space to understand root issues.
- By the end of this project, we will be able to build an **end-to-end deployable pipeline** which, given a URL as input, can output the security analysis performed by our services.
- We will also explore **Metasploit** and gain hands-on experience on tools for extensive penetration testing.

What will you be able to teach your classmates when you present this project?

While presenting the project, we will be able to teach our fellow classmates about the different client-side security issues and web vulnerabilities, and how to detect them. Learning to detect common vulnerabilities and misunderstandings might help many avoid deliberate security attacks.

We will also be delivering a small tutorial on Metasploit to the class and demonstrate some of its basic features.