

UNIVERSITY OF BUCHAREST

FACULTY OF MATHEMATICS AND INFORMATICS

SECURITY AND APPLIED LOGIC

DISSERTATION PAPER

Python Web Application Scanner

SCIENTIFIC COORDINATOR: GRADUATE:

LECT.DR. SILVIU LAURENȚ VASILE NĂIBOIU TEODOR

BUCHAREST, SEPTEMBER 2024

Contents

[I. Introduction 5](#_Toc173874652)

[I.1 Motivation 5](#_Toc173874653)

[I.2 Personal contribution 6](#_Toc173874654)

[I.3 Structure 7](#_Toc173874655)

[II. Technologies 8](#_Toc173874656)

[II.1 Python 8](#_Toc173874657)

[II.2 Requests 8](#_Toc173874658)

[II.3 Urllib 9](#_Toc173874659)

[II.3.1 Urllib.parse 9](#_Toc173874660)

[II.3.2 Urllib.requests 9](#_Toc173874661)

[II.4 BeautifulSoup 10](#_Toc173874662)

[II.5 Threading 10](#_Toc173874663)

[II.6 OS 10](#_Toc173874664)

[II.7 ConfigParser 10](#_Toc173874665)

[II.8 RE 11](#_Toc173874666)

[II.9 ArgParse 11](#_Toc173874667)

[II.10 HTML 11](#_Toc173874668)

[II.11 JINJA2 12](#_Toc173874669)

[II.12 DVWA 12](#_Toc173874670)

[II.13 Bee-Box 12](#_Toc173874671)

[II.14 Other technologies 13](#_Toc173874672)

[III. Scanner Vulnerabilities and Misconfigurations Detections 14](#_Toc173874673)

[III.1 Injections 14](#_Toc173874674)

[III.1.1 SQL Injections 15](#_Toc173874675)

[III.1.2 XSS Injections 15](#_Toc173874676)

[III.1.3 HTML Injections 16](#_Toc173874677)

[III.1.4 Command Execution Injections 16](#_Toc173874678)

[III.1.5 iFrame Injections 16](#_Toc173874679)

[III.1.6 Javascript Injections 17](#_Toc173874680)

[III.1.7 IDOR Injections 17](#_Toc173874681)

[III.1.8 LFI Injections 17](#_Toc173874682)

**Abstract**

The following project represents a scanner software application written in Python programming language, designed to aid developers, white hat hackers (security researchers, red-teamers, bug-bounty hunters) and other cybersecurity enthusiast in detecting vulnerabilities and assessing the attack surface and risk exposure of a Web Application. This new implementation is a redesigned and enhanced version of my previous Web Application Vulnerability Scanner software presented in my under-graduate thesis [1]. This new implementation is designed for Linux ***O****perating* ***S****ystem* (OS) usage only. Users can interact with the application through the ***C****ommand* ***L****ine* ***I****nterface* (CLI) and by providing the scanning arguments such as the target website, scan type and login details.

The new scanner functionality has been enhanced by refining more accurate detections and by adding new features such as rogue comment identification, hierarchical site map graph, external ***U****niform* ***R****esource* ***L****ocators* (URLs) references, hidden URL paths and ***C****ommon* ***V****ulnerabilities and* ***E****xposures* (CVE) detection. The reporting method has also been changed to a ***H****yper****T****ext* ***M****arkup* ***L****anguage* (HTML) report. The new version has been built for scalability and ease of access, as the number of prerequisite configurations has been decreased compared to the previous version. Additionally, the majority of ‘future enhancements’ mentioned in the previous thesis were implemented in the new version along a completely new method of scanner-website interaction logic that increases reliability and decreases the time of execution.

# Introduction

*“*Cybersecurity is the art of protecting networks, devices, and data from unauthorized access or criminal use*”* [2]. This formal definition of cybersecurity can be visualized as a tug-of-war battle between two opposing forces whose common purpose is to dominate each other’s’ territory. Over the last years, the need of understanding and educating people on cyber security has been increasing. Nowadays, not only people working in the **I**nformation **T**echnology (IT) industry are impacted by digital data polluted with malicious additions, but any other person with a smart device connected to the internet can a victim of the attackers’ bad intentions either directly or, as most of the time, coincidentally.

Currently, more and more solutions are transitioning to web applications designed to provide the users with a fast resolution for everyday activities (for example: buying a bus ticket, ordering a meal from the local restaurant, tracking your parcels or accessing your bank balance). “On average every day, approximately 30000 new websites fall to hacking attacks” [3]. As the numbers of websites is increasing due to demand and digitalization, so are the attacks aimed at stealing data by exploiting misconfigurations, vulnerabilities or by simple social engineering attacks facilitated by lack of access restrictions. Due to this reason, numerous websites are storing sensitive user data that can be worth lots of money in the hands of the wrong people. So, protecting the web applications and their data has become one of the most important security issues of our times, as today, compromising such an application can provide valuable data for further elaborated attacks.

## Motivation

I have decided to continue my previous implementation of the web application vulnerability scanner and to enhance its functionality due to my ongoing passion for web application security and for my personal projects. “A web application is a computer program stored on a remote server and run by its users via a web browser” [4]. “A web browser is a computer program used for accessing sites or information on a network (such as the World Wide Web)”[5]. As the web application is designed to be publicly accessible, attackers have an easy target available to lock on their scanners to and wait for immediate and easy to reach security weaknesses. Due to this reason, I have aimed to develop a scanner tool that simulates attackers’ tactics to proactively identify the latest emerging threats.

As information is constantly changing, so are software solutions and their configurations. Due to the ongoing modifications in web application service configurations and their increased complexity, researchers and red-teamers can have difficulties in detecting vulnerabilities in a timely manner. This also causes automated vulnerability detection tools to become deprecated really fast. As my project is an automated detection tools as well, I have decided to further continue its development and update the detection capabilities to keep up the latest security risks and recommendations while also keeping track of scalability and performance. In the following pages, I will present the new features of this tool, as well as the implementation methods I have chosen for this new and enhanced web applications scanner.

## Personal contribution

While the previous version was much more focused on the ***O****pen* ***W****orldwide* ***A****pplication* ***S****ecurity* ***P****roject* (OWASP) Top 10 Standards [6], the new version is based on my work experience as a Security Analyst and on ***C****apture* ***T****he* ***F****lag* (CTF) competitions experience and observations, over the last 3 years.

“CTFs are gamified competitive cybersecurity events that are based on different challenges or aspects of information security” [7]. The focus was shifted from a OWASP Top 10 approach to a more subjective means of detections based on my past observations from live models of vulnerable web applications. Almost all of the detection methods and capabilities were implemented based on my best judgement instead of generic approaches. This was done to further increase the accuracy of detections and to decrease the false-negative findings. Due to this approach, there might be a significant number of false-positive findings as well, hence the scanner results should be confirmed by manual replication.

Furthermore, I have decided to bring this scanner up to a competitive level by lowering the number of prerequisite configurations needed, and by providing a human readable report with dynamic and intuitive ***U****ser* ***I****nterface* (UI).

## Structure

TBD after chapters

# Technologies

Compared to the previous scanner version which was running Python version 3.2, the new version was implemented in Python version 3.9. Python is one of the best programming languages to use for automated sequential or parallel instructions, since it is easy to learn, to read and to understand [8]. Additionally, what makes Python great for automations is the big number of modules and libraries available and easily compatible with numerous software applicability (web and desktop applications, data structures and analysis, scripts, etc.).

For this project, the most important modules and components are the ones used to communicate with the website, such as *beautifulsoup* [9], *requests* [10], *urllib* [11]. In the following pages I will be describing the modules among other technologies and solutions used to build, run and test the new version of the scanner.

## Python

“Python is an interpreted, object-oriented, high-level programming language with dynamic semantics” [12]. Due to the great, easy to read and write syntax, Python is widely used for complex automated actions among cybersecurity enthusiasts. In projects similar with this one, Python is the best choice of any individual interested in ease of use and versatility. Web applications are constantly changing their structure and code, due to this reason, automated scanning tools need to be implemented in easy to maintain solutions that can handle large amount of data and also be compatible with other tools and technologies [13].

## Requests

The *requests* module is used for transferring data from a client (*Web Browser*) to a *Web Server* using the ***H****yper****t****ext* ***T****ransfer* ***P****rotocol* (HTTP) protocol and ***U****niform* ***R****esource* ***L****ocators* (URLs) [10]. There are a total of nine requests type (*GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH*). During the development of this project, I have used four types of requests:

**GET** – This is one of the principal types of requests as it represents the action of requesting data from the server. This request type is used for receiving data according to the provided parameters.

**POST** – Another principal request type method used to send data to the server. This method can be used for creating new data on the server and for updating already existing data.

**PUT** – Method used for creating or changing the representation of a resource according to the data provided in the request payload.

**TRACE** – Method generally used for debugging purposes. This method provides information on the path the requests take to the target destination.

The *requests* module is not only used for exchanging data with the server, but is also used for creating the session to maintain persistent communication with the Web Server during the testing procedures. This module also provides custom request creation functions and HTTP error handling methods.

## Urllib

Urllib module is used generally for URL manipulation [11]. From this module, I have used two functionalities:

### Urllib.parse

The *.parse* module from *Urllib* is used for combining strings back into URL format. In this project, the module has been used for combining actions, extracting certain strings of interest and fragment parts of URLs into variables [14].

### Urllib.requests

The .requests module from Urllib is similar with the *requests* module. The difference between the two is that this module provides some more advanced HTTP actions. In this project, this module was used for opening and reading URLs [15].

## BeautifulSoup

One of the most popular modules used for interaction with ***H****yper****t****ext* ***M****arkup* ***L****angua*ge (HTML), ***E****xtensible* ***M****arkup* ***L****anguage* (XML) components and ***D****ocument* ***O****bject* ***M****odel* (DOM) of a page. It provides a wide array of functions that help with pulling data from web pages, navigation, modifying and parsing different values available on a website [9]. This module is reponsable for any interaction between the scanner and the targeted web page content.

## Threading

Threading is used for changing the flow of execution from a sequential form to a parallel form by creating a separate flow of execution that can run simultaneously. Threading is useful for applications that handle large number of independent instructions and in which execution time is important. The CPU is used for allocating divisions of the physical core into virtual components or codes, these virtual components are called *threads* [16]. Threading was implemented for decreasing the execution time of the project by running multiple tests for multiple URLs in parallel.

## OS

The ***O****perating* ***S****ystem* (OS) module is used any interaction with the operating system. Actions such as creating, reading, modifying, opening and closing files and directories are performed through the usage of this module [17]. This module can also be used for more advanced interactions with the OS such as accessing environments variables, process information, managing registries and other system related functions. In this project, the module is responsible for file interaction.

## ConfigParser

Configuration files are a necessary part of an application that interacts with multiple pieces of data and complex logic. Configuration files are code structures with the *.ini* file extension that can be easily modified by experienced and beginner users without the need of any application architecture knowledge. A configuration file contains custom variables that are usually depended on the execution environment or purpose of the application [18].

The *ConfigParser* module is responsible with the interaction between the configuration file and the application Python code. This module is used to access and import the configuration values in the code and execution logic.

## RE

***R****egular* ***E****xpressions* (RE or Regex) are recognition rules based on symbols that form search patterns to detect any certain characters or string of characters inside other strings based on that searching rule [19]. Once a pattern is defined, a string can be manipulated in any way the user wants, those include edits, deletions, word substitutions, character(s) extractions from any string of characters that contains that pattern.

The RE module is the Python implementation of Regex functionality. Regex has been extensively used across this project for vulnerability detection, result confirmation, URL manipulation and others.

## ArgParse

ArgParse is a module that is used for Python applications that require user interaction through ***C****ommand* ***L****ine* ***I****nterface* (CLI). ArgParse is the link between *sys.argv* (the list of commands passed to the script using the CLI) and the Python code [20]. The advantage of using ArgParse is that it provides a user-friendly CLI while also automatically generating the *–help* menu used for displaying all the available arguments. From a programming point of view, this module also provides an easier method to manipulate the arguments passed via CLI into the code logic. Since this new scanner version is based on CLI interaction, this module is needed for parsing the user arguments to the application logic.

## HTML

This is a simple module used for manipulating HTML entities. The submodules used from HTML are the modules that *escape* and *unescape* characters. An *escaped* *character* is a character that uses an alternative interpretation of the same character [21]. For Web Application scanning, certain characters need to be escaped when a set of data is sent to the website and others need to be unescaped when data is received. Across this project, data sent and received to, respectively from the application, needs to be escaped for further processing down the test logic.

## JINJA2

The Jinja2 module is responsible for HTML template creation and manipulation through Python code [22]. This module is used for creating the final HTML report file based on an already created template which serves as a starting point for the report output data and format. All scanning results are parsed through Jinja2 to the templating functions that handle the data. The data is then added to the report, through Jinja2 methods, along with the HTML tags required for the HTML page to be correctly interpreted and displayed.

## DVWA

***D****amn* ***V****ulnerable* ***W****eb* ***A****pplication* (DVWA) is a PHP/MySQL web application that is left intentionally vulnerable with the main goal of helping security enthusiasts, web developers, students and teachers to use test their tools, practice their cybersecurity skills and understand web application security better, in a controlled, isolated and legal environment [23].

This application is one of the two web servers used in this project for testing the scanner and simulating real life scenarios. The web server contains numerous different vulnerabilities and misconfigurations which allow a comprehensive coverage of the most frequently found vulnerabilities in the wild. This application was hosted in a controlled virtual environment using the Linux Distribution – Kali Linux.

## Bee-Box

Bee-Box or BWapp is also a PHP/MySQL web application left intentionally vulnerable that follows the same purpose as DVWA. Bee-Box contains over one hundred web vulnerabilities and it covers all the major known web weaknesses and bugs along with the ones from OWASP Top 10 project [24].

This is the second web server used for testing the scanner progress and detection rate in this project. This web server was hosted in a controlled virtual environment using the Linux Distribution – Ubuntu.

## Other technologies

Below are some of the less used but still important modules that were used for implementing the web application scanner.

The *random* module is used for generating pseudo-random numbers or choices for various purposes [25]. In this project, the module was used for choosing a pseudo-random web user agent from a list. A web user agent is a characteristic string of data that helps servers identify the application, operating system, vendor of the requesting source [26].

The *datetime* module is used for accessing the calendar’s date and time. Using this module, I have provided date and time details in the resulting report with the format *YYYY:MM:DD HH:MM:SS* [27]*.*

The *math* module provides access to mathematical operations and functions [28]. In this project, the module was used to perform rounding on list length divisions used to split the data for threading purposes.

The *colorama* module is used to color the output text to terminal [29]. In this project the module was used to color the output text when the error file is not available due to any reason.

The *json* module is used to import **J**ava**S**cript **O**bject **N**otation (JSON) data interpretation. JSON is a standardized format for data that can be easily interpreted and read both by humans and computers [30]. In this project, the *json* module was used for creating and manipulating the Website Map URLs Graph.

The *warning* module provides warning control functions. Warnings are issued in situations in which the program does not raise a condition that terminates the program, but the user should be aware of the event. In this project, the *warning* module was used only to suppress character encoding warnings for HTML received data.

# Scanner Vulnerabilities and Misconfigurations Detections

The following sections are necessary for better understanding vulnerabilities and common misconfigurations a web application usually has. This chapter covers the theoretical implementation of the scanner. The below pages present all the information necessary for understanding the detection capabilities of the scanner and its final report.

“A vulnerability, is a flaw in code or design that creates a potential point of security compromise for an endpoint or network”[31]*.* Usually, thereason a vulnerability exists is due to a misconfiguration. These misconfigurations are usually incorrect setups of data in the system or environment. Misconfigurations can be caused by lots of variables, both from a technical point of view to business logic.

Vulnerabilities represent attack opportunities for adversaries. There are multiple methods of abusing vulnerabilities for gaining access to the weakened system. Depending on the type and motives, attackers can leverage them for unauthorized access in the environment. Both *white-hat hackers* and *black-hat hackers* are constantly searching for these vulnerabilities as they represent the initial key of access to any system. The notion of *white-hat hacker* is used to refer to a person motived by legal and ethical reasons to find and report such misconfigurations. The *black-hat hacker* term is used to describe people motivated by financial gain or by seeking reputation from their community while disregarding the legal and moral constrictions [32].

In the following pages I will be describing some of the vulnerabilities and misconfigurations that can be detected using this scanner, with the mention that the new scanner version is no longer following the OSWASP Top 10 Standards, hence the vulnerabilities will be more biased towards personal criteria of exploitability, severity and importance.

## Injections

“Injections based vulnerabilities are still one of the most abused and widely spread ones, due to this reason, injections are considered a critical threat to cybersecurity” [33]. Injections rely on user input fields. The provided data always needs to be interpreted by the application; hence the application is susceptible to any malicious instruction if the input field is not properly sanitized. There are multiple types of Injection attacks, with the majority of them depending on the software solutions running on the application server.

The following types of injections are the most spread injections in the wild and also the ones detectable by this new scanner project.

### SQL Injections

***“S****tructured* ***Q****uery* ***L****anguage* (SQL) is the most common language for interacting with data stored in a relational database” [34]. These injections rely on misconfigurations in the input fields connected to the SQL Database running in the back-end of the web application. Attackers can leverage these vulnerabilities to arbitrarily tamper with any piece of data stored in the application. SQL Injections can be blind or reflected. With reflected SQL Injections, feedback is provided back to the client according to the provided set of SQL data. Blind injections do not provide any feedback back to the client; hence more difficult to detect.

For mitigating such vulnerabilities developers needs to ensure that the inputs are properly sanitized and that the input fields are not directly referencing the SQL database query interpreter.

### XSS Injections

***“Cross****-****S****ite* ***S****cripting* (XSS) injections refer to the practice of sending malicious scripts, in the form of a browser side script to a target victim using a web application” [35]. The victim’s web browser will execute the code as part of the legitimate website. XSS attackers are used when attackers want to execute code on behalf of another user in their browser. The executed code will have all the privileges and access rights the victim has at the moment the malicious script is executed. There are multiple types of XSS attacks.

#### Reflected XSS

Reflected XSS attacks are the most common XSS attack in the wild. This type of attack relies on single execution of the malicious script and requires user interaction. The malicious code is not stored in the browser and can only be executed once, this applies to both requests and response transmitted malicious code [36].

#### Stored XSS

Stored XSS is one of the most dangerous XSS attack as it does not require any user interaction. This attack is based on injecting XSS code directly into the database through the vulnerable inputs. The malicious code gets stored on the affected page and gets executed each time the page is accessed, hence affecting all users visiting the website [37].

### HTML Injections

This injection is a type of attack in which HTML tags are injected directly in the DOM of the website. This attack is different than others as it does not exploit the back-end of the server, but the structure and content of web page. Similar with the XSS, this type of Injection also has single time execution (Reflected HTML Injection) and persistent injections (Stored HTML Injection) [38].

### Command Execution Injections

Also called ***R****emote* ***C****ode* ***E****xecution* (RCE), code injection is a method that allows attackers to remotely execute unauthorized code on targeted systems with the end goal of executing commands. The targeted server executes OS commands received through the web application inputs. Similar with the SQL Injection, Command Injection can also be reflected or blind.

### iFrame Injections

“iFrame is short for inline frame and is used to embed content from another website into the current page” [39]. iFrames injection is the practice of injecting malicious code into a webpage in the form of an iFrame element in the HTML page. Once an iFrame is injected, attackers can add their own controlled domain snippet to the targeted app, this can cause numerous security issues as iFrames provide full view of the embedded domain to the body of the compromised webpage.

### Javascript Injections

“JavaScript is the most powerful and versatile web programming language used for building interactive websites through implementation of animations, interactive forms and dynamic content to web pages” [40]. Javascript injections are a type of injections in which attackers inject malicious Javascript code into the client side of the application, usually a web browser. Through this attack, adversaries can steal sensitive information such as personal user data, credentials, sessions, cookie settings and any other data available in the target’s browser, but also manipulate the targeted website, redirect the users to other attacker-controlled domains or installing malicious extensions in the victim’s browser.

### IDOR Injections

“***I****nsecure* ***D****irect* ***O****bject* ***R****eference* (IDOR) is a vulnerability that arises when attackers can access or modify objects by manipulating parameters used in a web application” [41]. This type of attack allows adversaries to access unauthorized data through URL tampering or DOM body manipulation by modifying the vulnerable fields with custom values that link different objects in the back-end or database. This vulnerability is most commonly abused for horizontal privilege escalation, but it can be used for vertical privilege escalation as well. The term of horizontal privilege escalation refers to the practice of tampering with data of users with the same access level as the attacker’s. Vertical privilege escalation refers to the practice of accessing a higher level of privilege than the one they initially possess.

### LFI Injections

***“L****ocal* ***F****ile* ***I****nclusion* (LFI) allows an attacker to include files on a server through the web browser” [42]. Local files are files that are already present on the server that is running the web application. Accessing such files can lead to information disclosure, RCE or XSS. This vulnerability can be easily abused by providing filenames or file paths as inputs in the URL parameters or in the body of a website. Using this vulnerability, an attacker can perform a Directory Transversal attack and access any file from the server. Path or Directory transversal is a type of vulnerability in which the attacker is navigating the file hierarchy on the targeted system through CLI compatible commands such as *‘../../etc/passwd*’ for navigating 2 folders back and accessing the account passwords in UNIX based systems.

**References**

|  |  |
| --- | --- |
| [1] | T. Naiboiu, Web Application Vulnerability Scanner, Bucharest: University Of Bucharest, 2021. |
| [2] | CISA, 1 February 2021. [Online]. Available: https://www.cisa.gov/news-events/news/what-cybersecurity. [Accessed 1 August 2024]. |
| [3] | BUGTI, "Website Hacking Statistics 2024," BUGTI, 07 05 2024. [Online]. Available: https://www.bugtiwallet.com/website-hacking-statistics-2024/. [Accessed 1 August 2024]. |
| [4] | A. Volle, "Web application," [Online]. Available: https://www.britannica.com/topic/Web-application. [Accessed 1 August 2024]. |
| [5] | W. I. a. W. Browser?. [Online]. Available: https://www.lifewire.com/what-is-a-browser-446234. [Accessed 1 August 2024]. |
| [6] | "OWASP," [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed 1 August 2024]. |
| [7] | K. Crawley, 10 06 2021. [Online]. Available: https://www.hackthebox.com/blog/what-is-ctf. [Accessed 1 August 2024]. |
| [8] | "What is Python?," [Online]. Available: https://www.qamadness.com/knowledge-base/python-for-automation-why-is-it-great-and-how-to-use-it/. [Accessed 1 August 2024]. |
| [9] | "Beautiful Soup Documentation," [Online]. Available: https://beautiful-soup-4.readthedocs.io/en/latest/. [Accessed 1 August 2024]. |
| [10] | "Requests: HTTP for Humans," [Online]. Available: https://docs.python-requests.org/en/latest/index.html. [Accessed 1 August 2024]. |
| [11] | "URL handling modules," [Online]. Available: https://docs.python.org/3/library/urllib.html. [Accessed 1 August 2024]. |
| [12] | "What is Python?," [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed 1 August 2024]. |
| [13] | "Why is Python Important for Cybersecurity?," [Online]. Available: https://www.scaler.com/topics/cyber-security/cybersecurity-importance-for-python/. [Accessed 1 August 2024]. |
| [14] | "urllib.parse — Parse URLs into components," [Online]. Available: https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse. [Accessed 1 August 2024]. |
| [15] | "urllib.request — Extensible library for opening URLs," [Online]. Available: https://docs.python.org/3/library/urllib.request.html#module-urllib.request. [Accessed 1 August 2024]. |
| [16] | "What are Threads in Computer Processor or CPU?," [Online]. Available: https://www.geeksforgeeks.org/what-are-threads-in-computer-processor-or-cpu/. [Accessed 1 August 2024]. |
| [17] | "os - Miscellaneous operating system interfaces," [Online]. Available: https://docs.python.org/3/library/os.html. [Accessed 1 August 2024]. |
| [18] | "configparser - Configuration file parser," [Online]. Available: https://docs.python.org/3/library/configparser.html. [Accessed 1 August 2024]. |
| [19] | "A Practical Guide to Regular Expressions," [Online]. Available: https://www.freecodecamp.org/news/practical-regex-guide-with-real-life-examples/. [Accessed 1 August 2024]. |
| [20] | "argparse — Parser for command-line options, arguments and sub-commands," [Online]. Available: https://docs.python.org/3/library/argparse.html. [Accessed 1 August 2024]. |
| [21] | "Escape sequences," [Online]. Available: https://en.cppreference.com/w/cpp/language/escape. [Accessed 1 August 2024]. |
| [22] | "Jinja," [Online]. Available: https://jinja.palletsprojects.com/en/2.10.x/. [Accessed 1 August 2024]. |
| [23] | "DVWA," [Online]. Available: https://www.kali.org/tools/dvwa/. [Accessed 1 August 2024]. |
| [24] | "BWapp - An extremely buggy web app!," [Online]. Available: http://itsecgames.com/. [Accessed 1 August 2024]. |
| [25] | "random - Generate pseudo-random numbers," [Online]. Available: https://docs.python.org/3/library/random.html. [Accessed 1 August 2024]. |
| [26] | "User-Agent," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent. [Accessed 1 August 2024]. |
| [27] | "datetime — Basic date and time types¶," [Online]. Available: https://docs.python.org/3/library/datetime.html. [Accessed 1 August 2024]. |
| [28] | "math — Mathematical functions," [Online]. Available: https://docs.python.org/3/library/math.html. [Accessed 1 August 2024]. |
| [29] | "colorama," [Online]. Available: https://pypi.org/project/colorama/. [Accessed 1 August 2024]. |
| [30] | "What Is JSON and How Do You Use It?," [Online]. Available: https://www.howtogeek.com/devops/what-is-json-and-how-do-you-use-it/. [Accessed 1 August 2024]. |
| [31] | TechTarget Contributor, "vulnerability (information technology)," [Online]. Available: https://www.techtarget.com/whatis/definition/vulnerability. [Accessed 03 August 2024]. |
| [32] | A. Ot, "What's the Difference Between Black-Hat and White-Hat Hackers?," [Online]. Available: https://www.makeuseof.com/what-are-black-white-hat-hackers/. [Accessed 03 August 2024]. |
| [33] | B. Lenaerts-Bergmans, "INJECTION ATTACKS," [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/cyberattacks/injection-attack/. [Accessed 03 August 2024]. |
| [34] | S. Gupta, "What Is SQL & How Does It Work? A Guide to Structured Query Language," [Online]. Available: https://www.springboard.com/blog/data-analytics/what-is-sql/. [Accessed 03 August 2024]. |
| [35] | KirstenS, "Cross Site Scripting (XSS)," [Online]. Available: https://owasp.org/www-community/attacks/xss/. [Accessed 03 August 2024]. |
| [36] | A. Dizdar, "Reflected XSS: Examples, Testing, and Prevention," [Online]. Available: https://brightsec.com/blog/reflected-xss/. [Accessed 03 August 2024]. |
| [37] | A. Dizdar, "Stored XSS: Impact, Examples, and Prevention," [Online]. Available: https://brightsec.com/blog/stored-xss/. [Accessed 03 August 2024]. |
| [38] | "HTML Injection," [Online]. Available: https://www.imperva.com/learn/application-security/html-injection/. [Accessed 03 August 2024]. |
| [39] | J. Kent, "What Are iframe," [Online]. Available: https://www.cequence.ai/blog/api-security/what-are-iframe-injection-attacks/. [Accessed 04 August 2024]. |
| [40] | "JavaScript Tutorial," [Online]. Available: https://www.geeksforgeeks.org/javascript/. [Accessed 04 August 2024]. |
| [41] | "Insecure Direct Object Reference Prevention Cheat Sheet," [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Insecure\_Direct\_Object\_Reference\_Prevention\_Cheat\_Sheet.html. [Accessed 04 August 2024]. |
| [42] | Aptive, "Local File Inclusion (LFI) — Web Application Penetration Testing," [Online]. Available: https://medium.com/@Aptive/local-file-inclusion-lfi-web-application-penetration-testing-cc9dc8dd3601. [Accessed 04 August 2024]. |