A dissertation submitted to the **University of Greenwich** in partial fulfilment of the requirements for the Degree of

Master of Science

*in*

**Big Data & Business Intelligence**

**Automated Cyber Vulnerability Assessment and Penetration Testing Using the Detect, Implement, Predict, and React (DIPR) Framework**

**Name:** XYZ

**Student ID:** 000

**Supervisor:** XYZ

**Submission Date:** August, 2024

**Word Count:** 10,019

AUTOMATED CYBER VULNERABILITY ASSESSMENT AND PENETRATION TESTING USING THE DETECT, IMPLEMENT, PREDICT, AND REACT (DIPR) FRAMEWORK

XXXXX

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK

*(Submitted 10 August 2024)*

# Abstract

We created the DIPR system, which is a cutting-edge cybersecurity solution, with the intention of simplifying the process of discovering and fixing vulnerabilities in online networks and applications. In order to do comprehensive vulnerability scans, it combines the capabilities of conventional tools like OWASP Zap and Nmap to evaluate vulnerabilities. This results in a platform that is both more efficient and easier to use.  By using automated procedures, the system reduces the amount of human involvement required while simultaneously improving the accuracy and speed of security assessments. The study provides an explanation of the creation, deployment, and testing of the DIPR tool. The combination of static and dynamic analysis, a user-friendly interface, and the possibility to create detailed reports are some of the major qualities of the tool that are highlighted in this article. The Coursera website's deployment of the system proves its usefulness in identifying and resolving vulnerabilities. In further development, the incorporation of machine learning techniques will be the primary emphasis in order to improve its predictive skills in the identification of vulnerabilities. Enhancing its effectiveness in protecting digital assets is another important consideration. The DIPR system represents a significant advancement in cybersecurity techniques because it provides a robust solution to the problems caused by the ever-evolving environment of network and online vulnerabilities.

**Keywords:** Nmap, OWASP Zap, Static and dynamic analysis, Network security, Web application security, Vulnerability detection, and Cybersecurity.

# Acknowledgements

I would especially like to thank XXXXXXXXXX for agreeing to be my supervisor and for his consistent advice, feedback, guidance and support throughout the lifecycle of this Msc data warehouse project.

I want to thank both XXXXXXXXXXXXX and XXXXXXXXX for agreeing to have the project demonstration on the schedule day.

# Table of Contents

[Abstract i](#_Toc173422528)

[Acknowledgements ii](#_Toc173422529)

[Table of Contents iii](#_Toc173422530)

[1 Introduction 1](#_Toc173422531)

[1.1 Existing Problem 1](#_Toc173422532)

[1.2 Proposed Solution 2](#_Toc173422533)

[1.3 Aim and Objectives 3](#_Toc173422534)

[1.3.1 Primary Aim of the Project 3](#_Toc173422535)

[1.3.2 Objectives 3](#_Toc173422536)

[1.4 Project Scope 3](#_Toc173422537)

[1.5 Report Structure 4](#_Toc173422538)

[1.5.1 Introduction 4](#_Toc173422539)

[1.5.2 Literature Review 4](#_Toc173422540)

[1.5.3 Methodology 4](#_Toc173422541)

[1.5.4 Design & Implementation 4](#_Toc173422542)

[1.5.5 Evaluation and Results 5](#_Toc173422543)

[1.5.6 Discussion 5](#_Toc173422544)

[1.5.7 Conclusion and Future Work 5](#_Toc173422545)

[2 Literature Review 6](#_Toc173422546)

[2.1 Background 6](#_Toc173422547)

[2.2 LR Methodology 7](#_Toc173422548)

[2.3 Related Work Review 9](#_Toc173422549)

[2.3.1 Artificial Intelligence and Machine Learning based Solutions 9](#_Toc173422550)

[2.3.2 Solutions based on Crawlers 11](#_Toc173422551)

[2.3.3 Methods for Static Database-Based Threat Detection 12](#_Toc173422552)

[2.4 Identified Research Gap 13](#_Toc173422553)

[3 Methodology 15](#_Toc173422554)

[3.1 Static Analysis 15](#_Toc173422555)

[3.2 Dynamic Analysis 16](#_Toc173422556)

[3.3 DIPR Framework Operations 18](#_Toc173422557)

[3.3.1 URL Submission for the Purpose of Scanning 18](#_Toc173422558)

[3.3.2 Starting Website and Network Vulnerability Scans 18](#_Toc173422559)

[3.3.3 Analysis of Scan Results 19](#_Toc173422560)

[3.3.4 Comprehensive Report Collection and Retrieval 19](#_Toc173422561)

[4 Designing & Implementation 20](#_Toc173422562)

[4.1 Tools and Technologies 20](#_Toc173422563)

[4.2 Hardware Requirements 20](#_Toc173422564)

[4.3 Source Code Description 20](#_Toc173422565)

[4.4 Static Folder Contents 22](#_Toc173422566)

[4.4.1 CSS Files 22](#_Toc173422567)

[4.4.2 JS Files 23](#_Toc173422568)

[4.4.3 ICO File 23](#_Toc173422569)

[4.5 Templates Folder Contents 24](#_Toc173422570)

[4.5.1 Index File 24](#_Toc173422571)

[4.5.2 Login File 25](#_Toc173422572)

[4.5.3 Signup File 26](#_Toc173422573)

[4.5.4 Scan File 27](#_Toc173422574)

[4.6 Design of the Database 28](#_Toc173422575)

[4.7 Code Files Segregation 29](#_Toc173422576)

[5 Evaluation and Results 31](#_Toc173422577)

[6 Discussion 39](#_Toc173422578)

[7 Conclusion and Future Work 41](#_Toc173422579)

[References 42](#_Toc173422580)

List of Figures

[Figure 1: Types of Security Attacks Against Applications Experienced by Organizations in Past 12 Months (Statista, 2024) 2](#_Toc173402743)

[Figure 2: Extraction of Relevant Literature using Search String 8](#_Toc173402744)

[Figure 3: Classification of the Extracted Literature According to the Methods for Vulnerabilities Detection 9](#_Toc173402745)

[Figure 4: Static Analysis Procedure Implemented in the DIPR Framework 16](#_Toc173402746)

[Figure 5: Dynamic Analysis Procedure Implemented in the DIPR Framework 17](#_Toc173402747)

[Figure 6: A Framework for the Operational Process of the DIPR 18](#_Toc173402748)

[Figure 7: File Hierarchy of DIPR Framework 21](#_Toc173402749)

[Figure 8: Static Folder Contents 21](#_Toc173402750)

[Figure 9: HTML Files in Templates Folder 24](#_Toc173402751)

[Figure 10: Index File Code 25](#_Toc173402752)

[Figure 11: Login File Code 26](#_Toc173402753)

[Figure 12: Signup File Code 27](#_Toc173402754)

[Figure 13: Scan File Code 28](#_Toc173402755)

[Figure 14: MySQL Database Creation 28](#_Toc173402756)

[Figure 15: SQL Database Users Table Creation 29](#_Toc173402757)

[Figure 16: Login Form Creation 29](#_Toc173402758)

[Figure 17: DIPR System Signup Interface 31](#_Toc173402759)

[Figure 18: DIPR System Login Interface 32](#_Toc173402760)

[Figure 19: DIPR System Vulnerability Assessment Module 33](#_Toc173402761)

[Figure 20: DIPR System Scanning Interface 34](#_Toc173402762)

[Figure 21: DIPR System Vulnerability Scan Results of Targeted Website 35](#_Toc173402763)

[Figure 22: Detailed Scan Results with Vulnerability Names and Instances 36](#_Toc173402764)

[Figure 23: Target Network Vulnerabilities Detected 37](#_Toc173402765)

[Figure 24: DIPR System Comprehensive Report Overview 38](#_Toc173402766)

List of Tables

[Table 1: Inclusion and Exclusion Criteria Specified for the Selection of Relevant Literature 8](#_Toc173403455)

# Introduction

## Existing Problem

Because of the linked nature of today's digital ecosystem, the danger posed by cyberattacks has grown to be both wide and sophisticated. Threats posed by the internet pose enormous dangers to both enterprises and governments globally. The extent of these dangers ranges from simple efforts to get access via a backdoor to intricate assaults on vital infrastructure. As a result of the never-ending development of hacking methods and the fast advancement of technology, there is an increased need for cybersecurity measures that are both resilient and dynamic. Consequently, in order to expedite the process of vulnerability identification while reducing the amount of interaction from human resources, it is necessary to have vulnerability identification and management systems that are both user-friendly and automated. Taking this method will not only save time, but it will also reduce mistakes that are caused by human error.

A new analysis from Statista (2024) indicates that the number of security incidents and data breaches has been significantly growing over the last several years. Because of this tendency, there is a pressing need for vulnerability detection tools that are both more sophisticated and more rapid. It is common for experts in the field of cybersecurity to feel overwhelmed by the enormous number of possible vulnerabilities that they are required to address. The manual detection and evaluation of these vulnerabilities calls for the use of a variety of specific tools and instructions. As a result, it is a strategy that requires a significant amount of time and resources. Not only might it lengthen the amount of time it takes to respond, but it also makes it more likely that someone will miss something (Statista 2024).

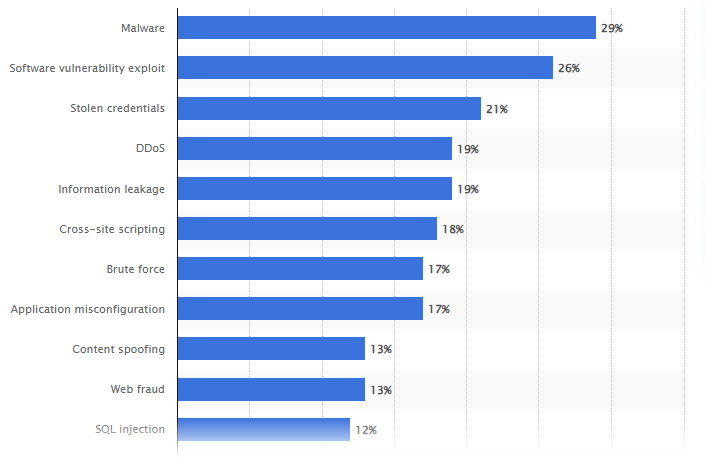


Figure 1: Types of Security Attacks Against Applications Experienced by Organizations in Past 12 Months (Statista, 2024)

In addition, the complexity of today's information technology infrastructures has made vulnerability management a job that is far more difficult to do. A wide array of linked devices, apps, and networks make up the modern digital systems that we use today. It's possible that every component has its own unique set of vulnerabilities. In addition, the interdependencies that exist between these components might provide further hurdles to the security architecture. Because of the dynamic nature of these systems, it is possible for new vulnerabilities to suddenly appear at any moment. Because of this, it is necessary to do ongoing monitoring and evaluation. Consequently, it is not just advantageous but absolutely necessary to have an automated tool that is capable of effectively identifying and managing vulnerabilities in the context that is considered.

## Proposed Solution

The Detect, Identify, Predict, and React (DIPR) framework is proposed to automate the process of vulnerability identification and management. This framework aims to address the above-mentioned challenges by enhancing the efficiency and effectiveness of vulnerability identification in digital systems. The proposed DIPR framework will offer a streamlined and user-friendly solution for cybersecurity professionals and penetration testers by integrating multiple scanning tools and processes into a single application.

## Aim and Objectives

### ****Primary Aim**** of ****the Project****

The main objective of the proposed project is to provide cybersecurity experts and penetration testers with a straightforward and user-friendly solution that integrates many scanning tools and procedures into a unified and efficient application.

### ****Objectives****

The following is the list of objectives required to complete in order to achieve the primary aim of the proposed project:

* To create an insightful and accessible application that simplifies the vulnerability assessment process for both experts and non-experts.
* To combine various open-source scanning tools into a single framework to provide a comprehensive vulnerability assessment.
* To automate the process of identifying and managing vulnerabilities.
* To utilize open-source tools that are frequently updated by their communities to maintain the relevance and effectiveness of the framework.
* To automatically incorporate updates and enhancements from the integrated open-source tools.

## Project Scope

The proposed project aims to develop an automated and user-friendly solution for vulnerability detection and management in various digital systems, such as online applications and networks. We intend to implement this solution across all these digital systems. This strategy places a major emphasis on the incorporation of a number of open-source technologies with the goal of ensuring continuous development and adaptation to new risk factors. The framework's design prioritizes scalability and flexibility as two of its defining features. As a result, it will be able to accommodate a wide range of security needs and will make it feasible to put preventive security measures into place in order to anticipate and eradicate any potential threats.

## Report Structure

A number of primary chapters make up the framework of the project report, and each of these chapters focuses on a distinct facet of the project. The following is a concise explanation of each of them:

### Introduction

Chapter 1 provides an explanation of the existing problem along with its potential remedy. The project's key goals and objectives are briefly explained, and the scope of the project is discussed.

### Literature Review

The background information about the topic is presented in this chapter. This chapter provides an overview of the history of cybersecurity and vulnerability detection, as well as the present level of knowledge in this topic. Technical concepts are explained in language that is easy to understand, and it is designed for readers who are not specialists in the area. Several different strategies are discussed in this chapter, with an emphasis placed on the advantages and disadvantages of each one. When it comes to the creation of the proposed system, this chapter was also responsible for identifying possibilities and best practices that were used.

### Methodology

The methodology of the project, including the stages of design, development, and testing, is described in depth in this chapter. Furthermore, it covered both prospective tactics that may be effective as well as problems that were experienced during the voyage of the project. Alternative strategies that were taken into consideration but ultimately not adopted were also highlighted, along with the reasoning behind the choices that were made.

### Design & Implementation

The technological design and development of the automated system are the primary topics that are covered in this section of the report. The system architecture, the procedure for implementing the system, and the tools and technologies that were chosen were all outlined in this document. In this way, it offers insights on the issues that were faced throughout the development process and how they were resolved.

### Evaluation and Results

This chapter provides an overview of the testing procedures that were used to verify the system, as well as the findings that were achieved. In it, the results are discussed in relation to the goals of the project, and an evaluation of the system's efficiency in automating vulnerability detection is carried out. Additionally, in order to produce a coherent narrative, the discussion was connected to the methodology and the literature review.

### Discussion

Chapter 6 comprehensively analyzes the thesis and the project's results in cybersecurity. It highlights the DIPR system's capacity to integrate static and dynamic analysis to find and fix vulnerabilities in online applications and web networks. The chapter describes the substantial system implementation preparation, including file processing and specialized software and technologies. It explains using Python and Flask for the backend, HTML, CSS, and JavaScript for the frontend, and OWASP ZAP and Nmap APIs for scanning and vulnerability detection.

### Conclusion and Future Work

In the last chapter, a summary of the most important results from the project was presented, and a connection was made between those findings and the original goals that were defined in the introductory chapter. In addition to providing a qualitative and quantitative analysis of the project's results, it also provided their evaluation. In addition to that, it provided suggestions for potential areas of study and development in the future. This last chapter also includes suggestions for enhancing the system and expanding the scope of the project. There are other recommendations included.

# Literature Review

## Background

Cybersecurity has become more important as a result of the widespread use of digital technology in a variety of facets of daily life and company operations. The proliferation of online applications and network systems has resulted in an increase in the number of possible vulnerabilities already present. Professionals in the field of cybersecurity faced complex hurdles as a result of this predicament.

Since the beginning of time, the majority of cybersecurity efforts have been reactive (Chen 2017). It refers to the actions taken by organizations in response to vulnerabilities and threats after such areas have been exploited. Firewalls and antivirus software were among the fundamental security measures that were essential in the first stages of cybersecurity (Dastres 2021). The primary objective of these first cybersecurity measures was to safeguard systems against recognized dangers. On the other hand, as digital infrastructures grew increasingly complicated and crucial to the operations of organizations, the need of taking preventative security measures became more apparent.

According to Yaacoub (2023), the notion of penetration testing, also known as ethical hacking, has arisen as a proactive way to find and repair vulnerabilities before they may be exploited by bad actors. For the purpose of locating vulnerabilities, penetration testers mimic cyberattacks on computer systems, applications, or whole networks. This method began as manual testing carried out by a small group of knowledgeable individuals and has now expanded to include a variety of automated tools that are meant to scan systems for vulnerabilities that are already known. There has been a tremendous improvement in the effectiveness of vulnerability detection as a result of the creation and use of a variety of scanning techniques. OWASP ZAP and NMAP are examples of tools that are meant to provide protection against certain categories of vulnerabilities. According to Riadi (2019), OWASP ZAP was developed specifically for the purpose of ensuring the security of online applications. It has capabilities that allow for both automatic and human testing of vulnerabilities in web applications. According to Bagyalakshmi (2018), Network Management Accounting (NMAP) is a tool that focuses on network security and offers capabilities for network discovery as well as security audits.

Although considerable breakthroughs have been made, the process of vulnerability detection continues to be riddled with difficulties. A huge strain has already been imposed on cybersecurity specialists as a result of the growing number of possible vulnerabilities. An additional degree of complexity was added to the process of vulnerability detection by the need to make use of various tools, each of which had its own particular emphasis and capabilities. In order to do this, a significant number of resources and a high degree of experience are required. This results in the process being less efficient and may leave systems vulnerable to vulnerabilities that have not been addressed.

In the realm of vulnerability identification, the current state of the art is to integrate and automate the process to the greatest extent possible. The concept of an automated system, capable of intelligently selecting and deploying the essential tools for a specific situation, represents the next step in the development of cybersecurity approaches. This is the next stage in the development process. We recommend incorporating this concept into the project. This solution would not only enhance the effectiveness of vulnerability identification, but also increase its accessibility to a wider audience. The scenario will ultimately lead to a general enhancement in the security posture of digital infrastructures. We are conducting this literature review to identify gaps and opportunities that the proposed project can address, thereby contributing to the advancement of cybersecurity practices. We will accomplish this by acquiring knowledge of both the historical context and the existing practices.

## LR Methodology

We came to the conclusion that the best way to identify data that was pertinent was to use a strategy that was both thorough and orderly. A specific search phrase was devised. We did this to simplify the process of conducting a literature review. We are referring to a search phrase that combines significant terms associated with vulnerability identification, security mechanisms, and cybersecurity approaches. The generated search string takes the following form:

*("Vulnerability identification" OR "security scanning" OR "penetration testing") AND ("automation" OR "automated tools") AND ("web applications" OR "network systems")*

The primary goal of reviewing related works is to gather insights into the latest developments and challenges in automating vulnerability detection and management within digital infrastructures. Google Scholar was chosen as the primary digital research library for this attempt due to its comprehensive database of scholarly articles.

As illustrated in Figure 2, the application of search string in Google Scholar enabled the extraction of relevant literature.

A screenshot of a computer

Description automatically generated

Figure 2: Extraction of Relevant Literature using Search String

In order to maintain the quality and reliability of the literature review, strict inclusion and exclusion criteria was specified as summarized in Table 1 below:

Table 1: Inclusion and Exclusion Criteria Specified for the Selection of Relevant Literature

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria Type** | **Publication Year** | **Publication Language** | **Publication Type** |
| Inclusion | 2018-2024 | English | High Impact Factor i.e. Q1/Q2 journals and well-known authentic conferences |
| Exclusion | Before 2018 | Non-English | Books, Book Chapters, Magazines, Websites, Thesis, and so on. |

After gathering the literature, we systematically organized it into three primary categories. We included solutions based on machine learning, threat detection approaches supported by static databases, and customized crawlers designed to identify vulnerable web applications in these categories. We successfully implemented the inclusion and exclusion criteria outlined in the previous paragraph. The figure below presents an illustration of the authors' classification of the extracted literature in percentage terms.

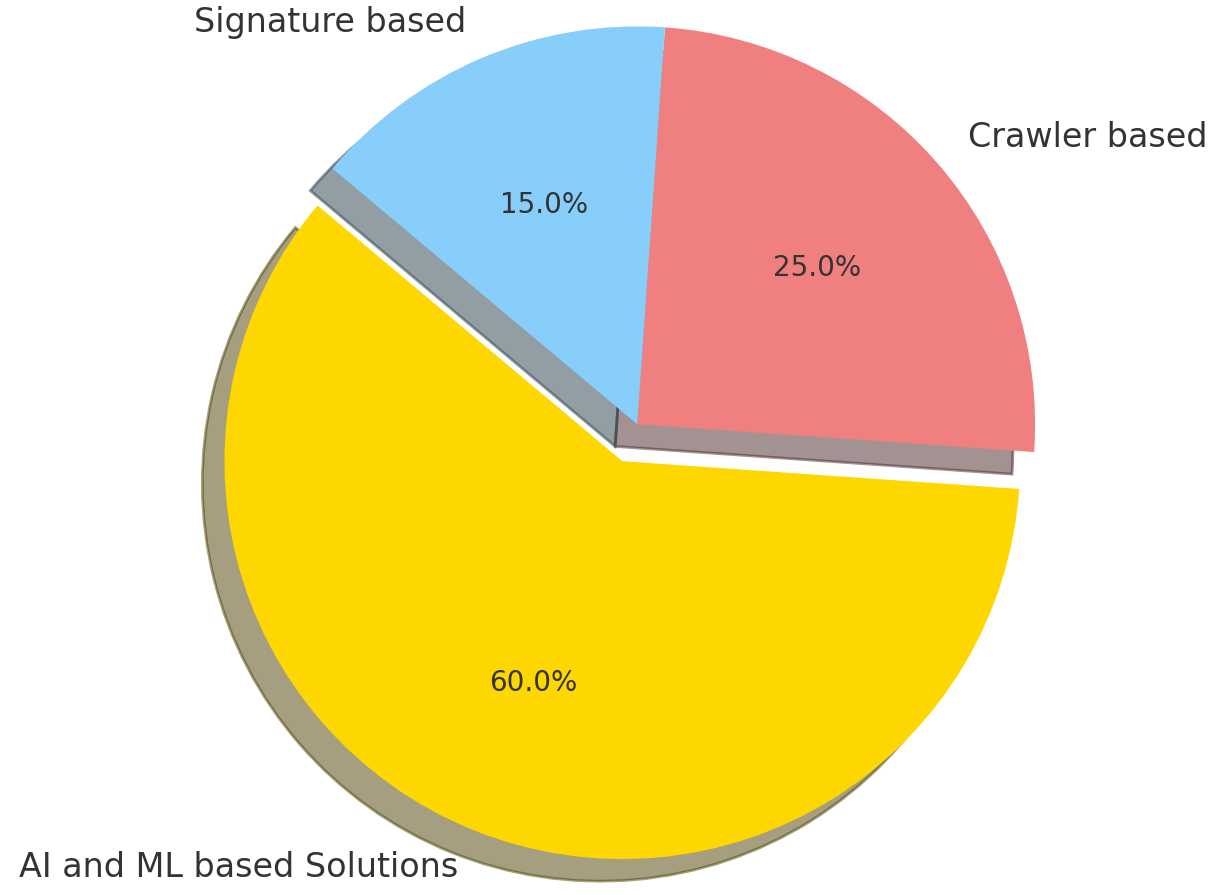


Figure 3: Classification of the Extracted Literature According to the Methods for Vulnerabilities Detection

## Related Work Review

### ****Artificial Intelligence and Machine Learning based Solutions****

A unique system known as "Vulnerability Hunter" was provided by the researchers as part of their research contributions (Guo 2020). This method was designed to find software vulnerabilities with a reduced reliance. It is typically necessary for professionals to manually identify characteristics when using conventional approaches, which might result in vulnerabilities being overlooked, as stated by Guo (2020). Therefore, by leveraging bytecode, their suggested method eliminates the possibility of security flaws. Using bytecode to describe vulnerabilities is a novel technique, and this particular method is one of the first to do so. The bytecode of a target application is compared with preset templates of vulnerabilities in order for the system to function properly. Therefore, it is able to identify whether or not PHP software has vulnerabilities such as SQL injection and Cross-Site Scripting (XSS) (Guo 2020). Through a series of tests, the efficacy of VulHunter was evaluated, and the results were rather outstanding. It achieved an F1-Score of over 88% for SQL injection, over 95% for cross-site scripting, and more than 90% for mixed sorts of vulnerabilities when identifying specific vulnerability types. Additionally, in comparison to other approaches currently in use, VulHunter exhibits a less percentage of both false positives and false negatives.

According to Backes (2017), PHP is the most popular programming language for use in online applications. Over eighty percent of the most popular websites are built on PHP, as stated by Backes (2017). However, security was not a primary consideration throughout the design process of these products. As a consequence, this often leads to unforeseeable security concerns as a consequence of a succession of fixes and mixed functionality. As a result, PHP is susceptible to vulnerabilities such as SQL Injection and Cross-Site Scripting. An innovative approach to analyze PHP programs for vulnerabilities in terms of security was presented by the authors. This approach makes use of code property graphs. The prototype that they produced was based on the most recent capabilities of PHP 7, and it included a graph database that was used to store the code graphs. A program's syntax, control flow, and data links are all included into a single graph. This is what code property graphs do. The ability to effectively search for and identify code errors is made possible by storing these graphs in a database (Backes 2017). They tested their approach on 1,854 open-source projects, and it was successful in identifying a variety of vulnerabilities that are present in web applications. There are some of them that include about 80 million lines of written PHP code. The procedure is mostly automated, beginning with the creation of the graphs and ending with their storage. In light of this, it makes it possible for analysts to look for and recognize a variety of weaknesses with ease (Backes 2017).

In the year 2020, Calzavara clarified how the Cross-Site Request Forgery (CSRF) attack is still an effective online attack that has the potential to inflict big problems such as economic harm and account captures. Many websites continue to be susceptible to this kind of attack, despite the fact that it is rather straightforward. Calzavara (2020) asserts that the procedures that were previously used to identify these vulnerabilities were technically impossible to implement. In order to do professional manual evaluation, these approaches often need access to the source code of the website. Without requiring access to the source code, the authors developed a machine learning technique called "Mitch," which is aimed to uncover vulnerabilities related to cross-site request forgery (CSRF). Over 5,800 HTTP requests from well-known websites were used to train Mitch, which is able to automatically recognize vulnerable HTTP requests that call for CSRF protection. Additionally, this dataset was distributed to the security community by the authors. Detecting 35 new cross-site request forgery vulnerabilities across 20 big domains is an impressive accomplishment for Mitch. In addition to this, Mitch discovered three CSRF problems in the production program that were overlooked by the other premier tools. This proves that Mitch has the ability to improve the security of the web (Calzavara 2020).

### ****Solutions based on Crawlers****

According to Azshwanth (2022), in the rapidly changing technological world of today, each new breakthrough comes with it the possibility of new security dangers. XSS is one example of such a widespread problem. One sort of attack is known as cross-site scripting (XSS), and it exploits vulnerabilities that are present in web pages (Azshwanth 2022). The emergence of this security issue occurs when a website permits the addition of custom JavaScript to its URL or directly into the text of the webpage. The end result is that if this malicious code is executed in a user's browser, it has the potential to infect their machine. As a result, the OWASP has identified cross-site scripting (XSS) as one of the top 10 vulnerabilities. Accordingly, the authors focus their attention on determining whether websites are vulnerable to cross-site scripting attacks by assessing the risks and severity levels connected with these attacks. In order for the researchers to do this, they make use of a specific crawler that is able to analyze both POST and GET queries. For the purpose of improving the identification of cross-site scripting vulnerabilities, this crawler was developed to allow multi-threading while also handling failures and exceptions in an effective manner (Azshwanth 2022).

Liu (2018) conducted a research study in which the authors investigated the intricate topic of second-order vulnerabilities in the context of online security. Liu (2018) asserts that second-order vulnerabilities are less evident than first-order vulnerabilities, but they have the potential to do greater damage. The authors propose a novel method for determining the existence of second-order security flaws in websites, which they have developed as a result of their research into online penetration testing and first-order attacks. The method that they use requires two steps of web crawling. A preliminary examination of the website is carried out in order to identify possible areas of interest. After this, a second scan is performed, which focuses on the URLs that are associated with these locations in order to identify vulnerabilities of the second order. This approach addresses a gap in the current strategies for identifying second-order security concerns on the web. It also greatly reduces the amount of time that is required to uncover vulnerabilities of this kind (Liu 2018).

Through the use of a web crawler, the researchers developed an automated approach for identifying vulnerabilities related to cross-site scripting (Zhou 2020). Scanning URLs using a crawler is the first step throughout the process. URLs were then simplified in order to eliminate duplicates by using a tree-based HASN technique. In order to uncover possible vulnerabilities, the system engages in a simulation of an XSS attack. According to the findings of the tests, URL deduplication took just 0.05 seconds to complete, attaining an accuracy rate of 89% while maintaining a low resource consumption (Zhou 2020). When compared to an existing tool known as XSSer, this alternative technique demonstrated a lower number of false positives and negatives and needed a shorter amount of time for detection.

According to Wang (2022), attacks on networks present a substantial risk to the safety of important information systems and essential network infrastructure. In light of this, it is of the utmost importance to provide a platform that allows for the collection, verification, and dissemination of software and hardware security vulnerabilities. Currently available vulnerability sharing systems, both local and foreign, often suffer from low degrees of automation and high human costs. In order to address these concerns, Wang (2022) suggested the development and deployment of a platform for the sharing of security vulnerabilities that makes use of web crawler technology. This technology makes it possible to automate the collecting of vulnerability data, which in turn makes it possible to query and share vulnerability information in a manner that is both more thorough and more easily accessible. According to Wang (2022), a platform of this kind may be of great assistance in the prompt warning and correction of security flaws, both of which are necessary to improve overall cybersecurity efforts.

### Methods for Static Database-Based Threat Detection

When it comes to cyberattacks and security breaches, Open-Source Software (OSS) vulnerabilities are a key source, as stated by Wan (2019). Software teams often make use of open-source software (OSS) in order to reduce the amount of work that has to be redone and to accelerate their development cycles. On the other hand, a significant number of people who utilize open-source software are unaware of the risks that exist in the components that they use (Wan 2019). It was said by the researcher that the National Vulnerability Database (NVD) often does not provide thorough coverage and may be somewhat inconsistent. It may take many weeks or even months for a Common Vulnerabilities and Exposures (CVE) to be discovered and fixed. This may be a very lengthy process. An extensive examination of Git commits from well-known open-source repositories, such as Linux, was carried out by the author as a result. Given that there was no existing dataset for Git commits that were significant to security, a web-based system was developed to enable security experts to manually categorize commits according to the degree to which they were useful to maintaining security. A deep neural network was constructed by Wan (2019) to automatically recognize commits those correct vulnerabilities based on the messages included inside the commits. This was done after the commits were cleaned and labeled. His technique demonstrated a substantially higher level of accuracy compared to the ways that were previously used, and it also boosted the recall rate by 16.8%.

AlYousef (2019) suggested a sophisticated approach for automatically updating attack signatures in Intrusion Detection Systems (IDS) in order to meet the difficulty of reacting to new forms of network attacks without the need for human intervention. The model that they have developed incorporates a filtering engine that functions as an auxiliary intrusion detection system (IDS) to improve the system's capability of properly detecting and responding to actions that are not permitted. In addition, the approach that was provided by AlYousef (2019) simplifies the process of incorporating new attack data into the intrusion detection system (IDS) by leveraging a blacklist of IP variables and adopting a mechanism that helps identify new attack signatures based on how similar they are to those that are already there. Because of this automation, the dependency on manually performed updates is considerably reduced. According to AlYousef (2019), this results in an increase in the overall efficacy of intrusion detection systems (IDS) in protecting network security.

## Identified Research Gap

As shown in Figure 3, most of the solutions proposed by the authors in existing literature i.e. 60% are backed with Machine Learning and Artificial Intelligence based solutions. 25% of the solutions employ crawler-based methods which used crawlers or spiders to detect potential security threats. The rest 15% of solutions adhere to a more conventional method. They are relying on a comprehensive database of known threat signatures to recognize and thwart attacks.

Most security solutions for networks and web applications now rely on AI and ML. Those solutions require manual updates due to the evolving nature of cybersecurity threats. These systems rely on training data to learn patterns and detect anomalies. The training data must be updated to reflect these changes as new types of attacks and vulnerabilities emerge. This process typically involves collecting new data, labeling it appropriately, and retraining the models to ensure they can recognize and respond to the latest threats. Additionally, model parameters and algorithms may need adjustments to improve accuracy and reduce false positives. Some aspects of this process can be automated. However, human administration is often necessary to validate the quality of data and to ensure the relevance of update to fine-tune the models. Additionally, there is a lack of research on simplifying complex security tools, which are often too difficult for non-experts to use. There is a research gap in developing self-updating security systems that are easy for everyone to use and potentially make network protection quicker and more user-friendly.

Tools that are open-source and freeware will support the proposed DIPR framework on the backend. These open-source applications are accessible to the communities that use them, and they frequently receive updates with new versions. Any modifications or enhancements made to these tools immediately propagate to the framework for use. This approach ensures the constant safeguarding and strengthening of the framework moving forward. This technique provides justification for the self-updating nature of the framework. This is due to the fact that it is reliant on the open-source tools that are used, which need active maintenance and regular upgrades.

# Methodology

The DIPR framework is characterized by the fact that it essentially combines both static and dynamic analysis as important operational aspects.

## Static Analysis

In the DIPR framework, static analysis is an essential component that performs an important purpose. An inspection of the source code was the primary emphasis of static analysis, with the goal of locating possible vulnerabilities inside the code. In this procedure, each file that is included inside the source code of the web application that is being targeted is subjected to a comprehensive examination. According to Rafique (2015), the use of static analysis necessitates the upkeep of a dataset that contains known vulnerabilities in the backend system. The vulnerability scanning tool begins a comparison between the source code of the website that has been specified and the vulnerability database that has been saved in advance whenever a user inserts a URL for vulnerability scanning. The purpose of this comparison is to ascertain whether or not the coding of the website conceals any of the vulnerabilities that have been known about. Because of this, it is possible to discover and eliminate any security issues at an earlier stage.

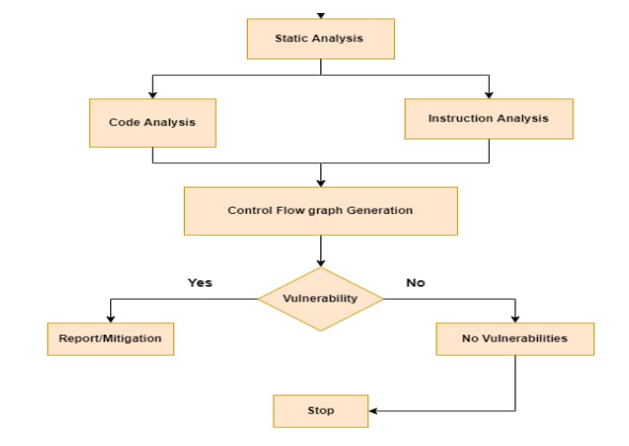


Figure 4: Static Analysis Procedure Implemented in the DIPR Framework

The process of static analysis is shown inside a flow diagram, which is shown in Figure 4. According to what has been shown, it may be broken down into two independent methodologies: instruction analysis and code analysis. Static analysis is a technique that allows users to uncover possible vulnerabilities in an application by examining the code of the program. Users are provided with a methodical approach to static analysis that allows them to investigate the underlying code structure and logic of a program in order to discover any potential security problems that may be concealed inside the application.

## Dynamic Analysis

In the process of dynamic analysis, there is a significant pre-testing step that takes place before the actual execution of the program. In order to assess whether or not the code is ready and secure, this approach is dependent on the inputs provided by the user (Rafique 2015). Its primary function is to ascertain whether or not the code is vulnerable to flaws that occur during runtime. As an example, dynamic analysis may entail a user entering a malicious SQL query into a login form in order to determine whether or not the website has enough safeguards, such as input validation, to mitigate the impact of assaults of this kind. It is useful to detect vulnerabilities that only become apparent during the operating phase of a program, and this kind of analysis is beneficial in doing so. It is for this reason that dynamic analysis offers a realistic evaluation of the security posture of the targeted system in situations that are based on the real world.

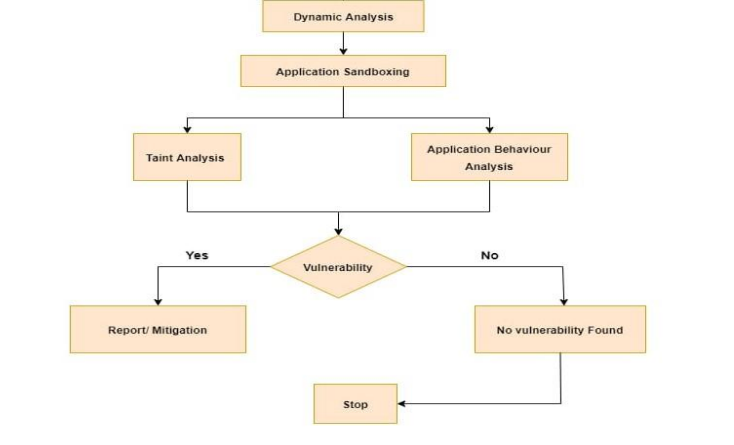


Figure 5: Dynamic Analysis Procedure Implemented in the DIPR Framework

Diagrammatic representation of the flow diagram for dynamic analysis is shown in Figure 5. It draws attention to the fact that users are required to install their application inside a sandbox environment. The implementation of this safety measure guarantees that the analysis will not interfere with or cause damage to any other system operations (Rafique 2015). It is essential to isolate the application that is being analyzed in order to ensure that the integrity of the system is preserved. This is because dynamic analysis has the ability to affect a number of different components of the system. In order to track the activity of the program and provide a variety of outputs in real time, this analysis approach requires the user to provide inputs. In order to assess the application's robustness and locate any operational vulnerabilities, it is necessary to carry out this procedure by determining if the program performs as intended under a variety of settings and inputs.

## DIPR Framework Operations

Figure 6 depicts a specific operational component of the DIPR Framework in further detail.

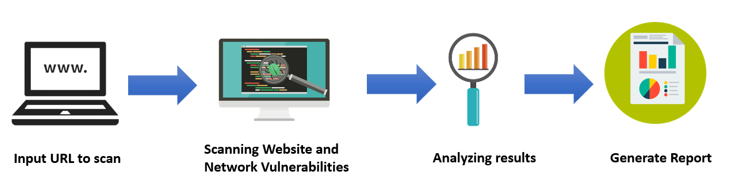


Figure 6: A Framework for the Operational Process of the DIPR

To guarantee that a comprehensive analysis of possible security risks is carried out, the procedure for scanning web application vulnerabilities that has been established in this project is meticulously detailed in a series of phases, as can be seen in Figure 6.

### URL Submission for the Purpose of Scanning

Users are required to provide the URL of the website that they desire to analyse for vulnerabilities in order to proceed with this first step. Users allow the framework to guide its scanning capabilities towards a particular online domain by supplying the targeted URL. This further puts the framework in a position to conduct a vulnerability evaluation that is more narrowly focused.

### Starting Website and Network Vulnerability Scans

Upon receiving the URL that was supplied by the user, the DIPR framework immediately begins a comprehensive scanning operation with the goal of discovering any vulnerabilities that may already exist. Through the application programming interfaces (APIs) of well-known security tools like OWASP ZAP and Nmap, it makes use of the capabilities of these tools. An initial spider scan is carried out with the assistance of OWASP ZAP as the first step in the procedure. This scan does a comprehensive crawl over the website in order to find vulnerabilities by simulating a variety of interactions with the website. Following the spider scan, the program will extract the domain name from the URL that has been supplied. This domain name will then be employed by Nmap in order to carry out a comprehensive network vulnerability scan via the network. By taking into account vulnerabilities connected to both the web and the network, this dual method guarantees a thorough security evaluation.

### Analysis of Scan Results

As a consequence of the scanning procedure, users are provided with a consolidated view of the results that are shown on the DIPR platform. During this step, it is possible to conduct a comprehensive study of the vulnerabilities that have been detected by displaying particular information on the severity levels and characteristics of the vulnerabilities. Users have the ability to examine the data in order to acquire insights into possible security holes that may exist within the architecture of the website and the network infrastructure that lies behind it.

### Comprehensive Report Collection and Retrieval

The final product is a unified report. The report compiles and presents the findings of vulnerability checks conducted on both the website and the network. We have made this single document available to consumers on the DIPR platform, enabling them to download and analyze the full study of vulnerabilities discovered during the event. We take this action with the intention of giving consumers the choice to meet their needs. This report serves as a valuable tool, providing insight into the security posture of the targeted website and its associated network. In relation to this matter, it is of great assistance. Furthermore, it makes it much simpler to arrive at well-informed judgments about the activities that will be necessary in the not-too-distant future. This is an additional benefit to the previous point.

# Designing & Implementation

## Tools and Technologies

In order to guarantee that the DIPR Tool functions in an effective manner, it is necessary to fulfill certain software requirements before it can be deployed. It is possible to use the tool with a wide range of operating systems since it was developed with adaptability in mind. In addition to the most recent versions of Visual Studio Code and Python, which are the basic programming environment and language, respectively, essential components include these softwares. OWASP ZAP and Nmap, in their most recent forms, are the engines propelling security evaluations. The extensive vulnerability scanning capabilities of each of these programs have earned them a lot of widespread recognition. Additionally, in order to fully support a wide range of functions, the environment must have the most recent version of Git bash for version control and a Java Environment that is up to date. In order to guarantee the best possible performance and compatibility of the DIPR Tool, it is very necessary to correctly align the installation of these software components with the particular needs of the operating system that is now being used.

## Hardware Requirements

In order for the suggested system to function without any hiccups, it is necessary to have certain hardware specs. In order to handle and analyze data in an effective manner, the tool necessitates the presence of a computer system that is outfitted with a 64-bit quad-core processor, as well as a minimum clock speed of 1.8 GHz. A minimum of four gigabytes of random access memory (RAM) should be included in the system in order to guarantee smooth multitasking and the ability to manage data. In addition, it is essential to have a reliable internet connection in order to ensure that users are able to access online resources without interruption and to carry out web-based vulnerability checks.

## Source Code Description

Each and every file associated with the project is stored in the "WEB\_VUL" folder, as shown in Figure 6. A Flask-based organization system is used to arrange the folder structure. Since this is the default configuration for Flask, when the code is executed, it will automatically look for HTML files in the "templates" folder. According to the same logic, Flask anticipates discovering CSS, JavaScript, and image files under the "static" subdirectory. The Flask application will be able to simply access all of the files since this configuration guarantees that they are all appropriately ordered.

A screenshot of a computer

Description automatically generated

Figure 7: File Hierarchy of DIPR Framework

Flask anticipates discovering graphics, CSS, and JavaScript files in the "static" folder, as was previously explained. The following is a depiction of the Static folder that is associated with the computer system that is being suggested. As can be seen in Figure 8, it includes files including CSS and JavaScript as well as graphics that are associated with the system.

A screenshot of a computer program

Description automatically generated

Figure 8: Static Folder Contents

The description of each file that is included inside the Static Folder is explained below.

## Static Folder Contents

### CSS Files

The Login\_style.css file includes the custom CSS stylings that were developed with the express purpose of improving the user experience of the Login.html page. The most important role of this file is to make the process of logging in visually appealing and interesting to the user.

The customized CSS rules that control the aesthetic layout of the Signup.html page are included inside Signup\_style.css file. This file is specifically designed to guarantee that the sign-up form is not only functional but also aesthetically beautiful.

The CSS stylings that are applied to the administrative areas of the website are included inside Sb-admin2.css file. In particular, the user interface of the Scan.html page is enhanced with a contemporary and expert appearance. In particular, the user interface of the Scan.html page is enhanced with a more polished and organized appearance thanks to the CSS stylings that are included inside this file. These stylings are applied to the administrative areas of the website seems quite lean.

During the process of vulnerability assessment, the Scan.css file includes extra CSS stylings that complement the Scan.html page and are focused on the scanning interface. The goal of this file is to guarantee that the users have a clear understanding of the process and that it is easy for them to navigate.

The stylings for the formatHTML.html page are included inside formatHTML.css file. With these stylings, the presentation of the final report that was created by the tool is designed to be more aesthetically pleasing.

The Successfully.css file provides the CSS definitions for the successfully.html page, which is used to communicate successful operations or activities inside the tool by providing clear visual feedback that increases the level of satisfaction experienced by the user.

The Index\_style.css file provides the basic CSS stylings that may be used to establish the visual tone of the front-end web page and to create an atmosphere that is engaging and user-friendly on the homepage of the website.

### JS Files

Enhanced with responsive features and interactive components that raise the overall user experience, the Bootstrap.bundle.js file comprises the bundled JavaScript functions that are offered by Bootstrap. Additionally, it enriches the design of the website.

JavaScript functions that are directly related to the HTML.html file format are included inside the formatHTML.js file. These functionalities allow dynamic interactions and the display of the report in a way that is more interactive and stimulating.

The Index2\_script.js file provides essential JavaScript functions that improve the Scan.html page by adding interactive features that include support for user actions and feedback in real time while the scanning process is being carried out.

The Login\_script.js file includes JavaScript code that adds interaction to the login.html page. The goal of this code is to guarantee that the user experience throughout the login process is seamless and responsive, with validations and user feedback.

All of the interactive aspects of the administrative dashboard, particularly those found on the Scan.html page, are brought to life by the JavaScript code that is included inside Sb-admin-2.min.js. More efficient administration and operation of the scanning process is made possible as a result of this tool.

For the purpose of providing users with real-time feedback and interactive controls while the vulnerability scanning process is being carried out, the Scan.js file is directly integrated with the scan.html page and includes essential JavaScript functions that increase the functionality of the detection process.

The Signup.html page is powered by JavaScript logic included in the Signup\_script.js file. This file also supports dynamic interactions, such as form validations and interactive feedback, which are designed to make the sign-up process comfortable for users and secure.

### ICO File

The particular file, Favicon.ico, includes the recognizable picture that is shown in the browser tab. This image acts as a visual identifier for the website, which helps in branding and makes it easier to identify the website among the many tabs that are open.

When the code is executed, Flask will do a search for HTML files in the "templates" folder, as was previously mentioned. This is because Flask's default settings allow it to do so. The files that occupy the template folder of the source code are shown in Figure 9.

A screenshot of a computer program

Description automatically generated

Figure 9: HTML Files in Templates Folder

The following is a description of each file that can be found in the Templates folder.

## Templates Folder Contents

### Index File

A representation of the source code for the index.html file, which is the homepage of the website, may be seen in Figure 10. Users are able to access their current account or establish a new one by using the navigation choices that are located on this primary page. These options include buttons for logging in and signing up new accounts. In addition, the homepage provides a summary of the website, including the logo, the name of the website, and a short introduction or description to welcome visitors.

A screenshot of a computer program

Description automatically generated

Figure 10: Index File Code

### Login File

The source code for the Login.html file, which is responsible for constructing the front-end interface of the login page, is available for viewing in Figure 11. By inputting their login credentials on this page, users are able to get access to the website where the DIPR is located. On the screen, there are boxes for users to enter their login and password, as well as a button that users may click to submit their information and go on with the process of verifying their credentials.

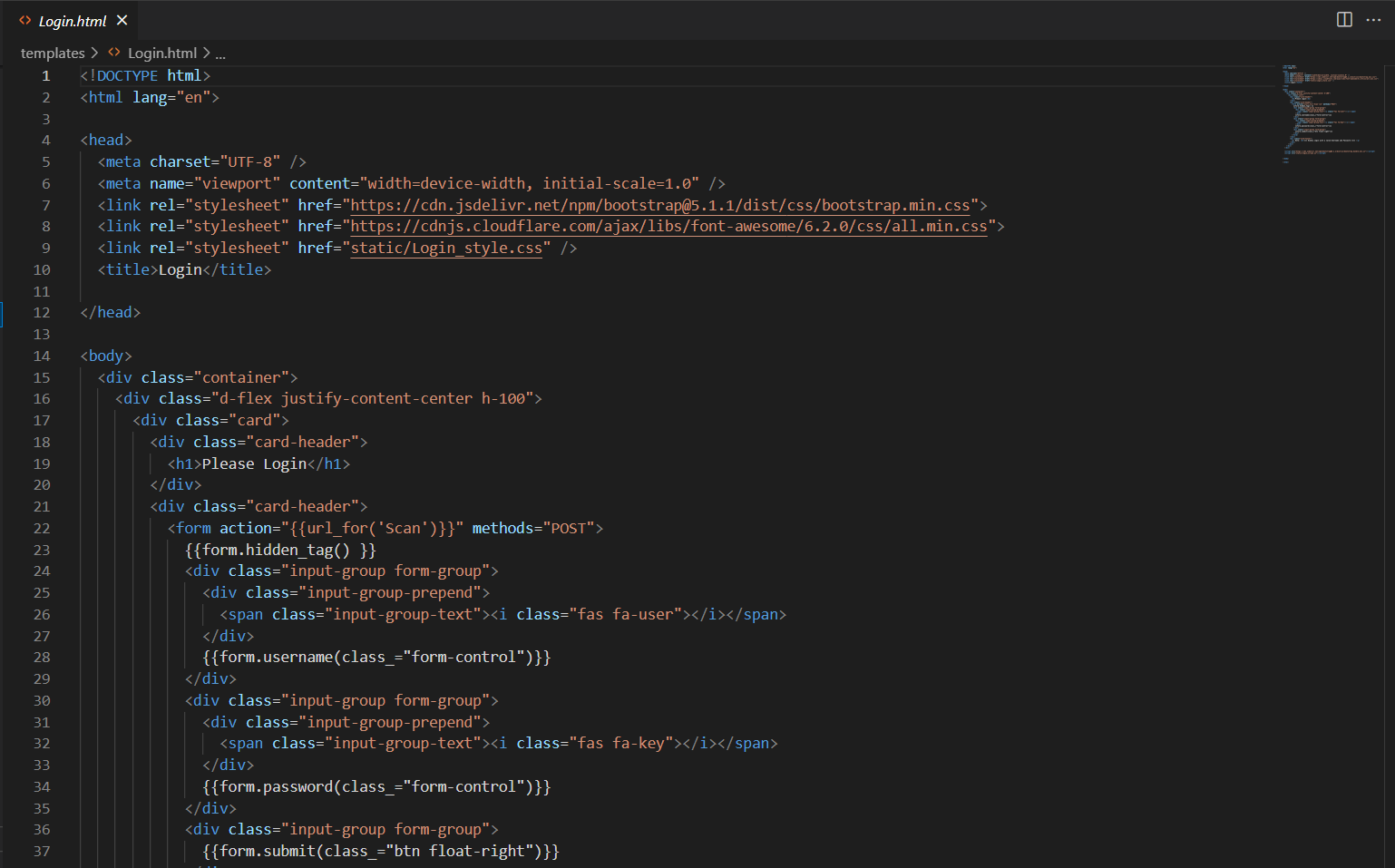


Figure 11: Login File Code

### Signup File

The source code for the Web page known as Signup.html is seen in Figure 12. This file is organized in such a way that it contains a registration form, which makes it possible for new users to sign up and establish an account on the website. In addition to directing people through the registration process, it is the component of the sign-up page that is considered to be the front-end.

A screenshot of a computer program

Description automatically generated

Figure 12: Signup File Code

### Scan File

A representation of the Scan.html file's source code can be seen in Figure 13. The DIPR Framework's scan capability is defined by this, along with the layout and style of the functionality. Additionally, it has semantic components and classes that imply an organized and stylized page layout, maybe for a dashboard or control panel where users can start or evaluate scans for vulnerabilities. This includes the possibility of a dashboard or control panel. The fact that the file contains a link to a font-awesome stylesheet for icons is another evidence of a well-designed and user-friendly interface.

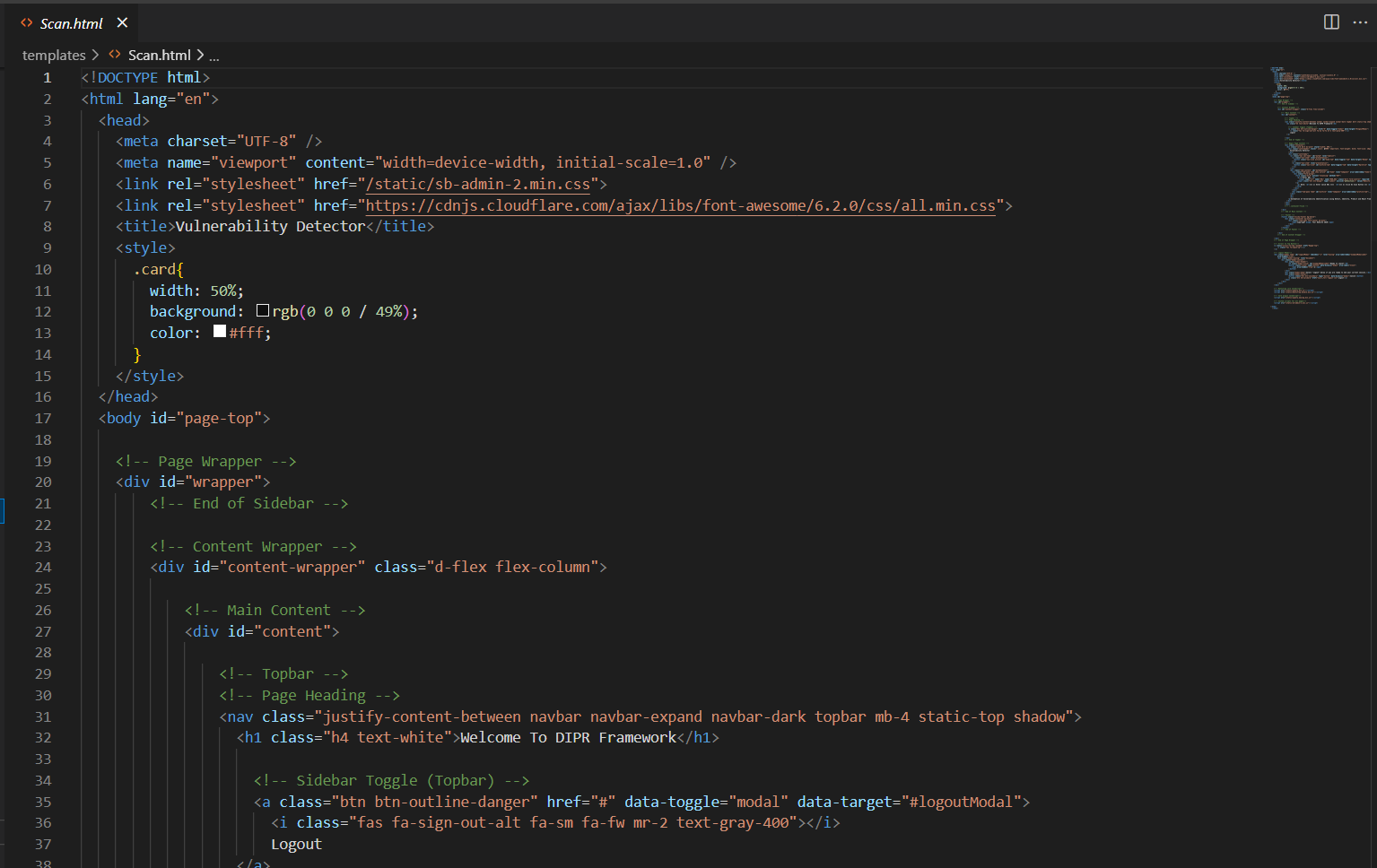


Figure 13: Scan File Code

## Design of the Database

As shown in Figure 14, we are now working toward the establishment and addition of a database that will contain the information of users who have signed up via the Signup process. The initial setting of the SQLALCHEMY DATABASE URL or SECRET KEY is the principal function that it is responsible for doing. The DIPR program makes use of that MySQL database.

A computer screen with text on it

Description automatically generated

Figure 14: MySQL Database Creation

The program is seen in Figure 15, which is an integrated MySQL database setup that is started in order to store the information used for user registration. The SQLALCHEMY DATABASE URL and a SECRET KEY are two of the main parameters that are included in it to guarantee that connections are kept safe. Within the confines of this database, a user table is constructed with the intention of storing essential user information, such as a username, a password, and a unique identifier (ID1).

A screen shot of a computer code

Description automatically generated

Figure 15: SQL Database Users Table Creation

After that, a user registration interface is created in order to direct new users through the process of signing up for the vulnerability scanning services. This is done before the new users are allowed access to the services. This particular interface consists of a registration class that has input fields for the username and password, which is then followed by a submit action that allows the data to be entered into the overall system. A validation method that is part of the registration class verifies that each username is different in order to ensure that the database maintains the exclusive status of each user.

There are fields on the login form that have specified properties, such as length constraints and obligatory input requisites. This is another essential component of the interface with the database. Users are able to verify their identities and give lawful access due to the presence of username and password fields, which allow them to input their credentials. These credentials are subsequently confirmed against the data that has been saved.

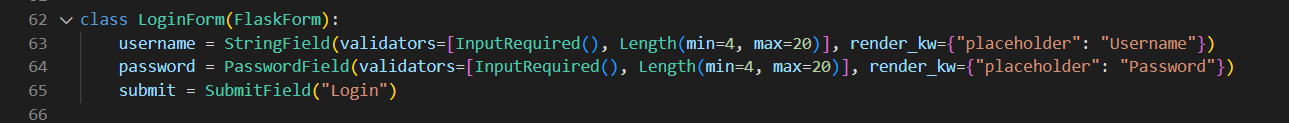


Figure 16: Login Form Creation

## Code Files Segregation

There are a total of 43 files included in the project. It consists of 15 HTML files for the front-end design of the website and four Python files that are responsible for the backend and database activities of the website. There are also eight CSS files that are used to style the design of the website, in addition to twelve JavaScript files that are used for interactive frontend components. Files with the name "database.db" are used to store user login credentials. In order to facilitate the process of setting up, a "Readme.txt" file is used to offer detailed instructions, and a "requirements.txt" file is used to list all of the critical Python packages that are required for the project to work successfully, along with the version numbers of those packages.

# Evaluation and Results

It was first hosted on a local server environment so that the efficacy of the DIPR architecture could be evaluated. A controlled testing phase was able to be carried out as a result of this configuration, which ensured that all of the system's components interacted as they were supposed to in order to produce accurate vulnerability assessments. The first step for users to do before beginning to work with the system is to register and then log in. Both the signup and login interfaces are shown in Figures 17 and 18, to be specific. Because the major focus is on the capabilities of the system that are connected to cybersecurity and vulnerability assessments, the registration procedure has been made as simple as possible and just needs a user ID and password in the first edition of the system.

A screenshot of a computer

Description automatically generated

Figure 17: DIPR System Signup Interface

A screenshot of a computer

Description automatically generated

Figure 18: DIPR System Login Interface

Figure 18 is an illustration of the primary interface of the system, which enables users to perform vulnerability checks on websites. This screen is shown when the user has successfully registered and logged into the system. Before beginning a scan, the user must first enter the URL of the website that they want to scan (also known as the target website), and then they must hit the button labeled "Search Website." Because an input validation mechanism is established in the system, it is vital for the user to submit a valid URL. This mechanism evaluates the website address to determine whether or not it is accurate. An error notice will be shown by the system in the event that the user inputs an incorrect address, which is defined as an address that does not follow the typical format of a website address. The user will be required to wait for a period of time for the completion of the scan once the website URL has been input properly. To demonstrate our capabilities, we carried out a vulnerability scan on the website located at <https://www.coursera.org/> .

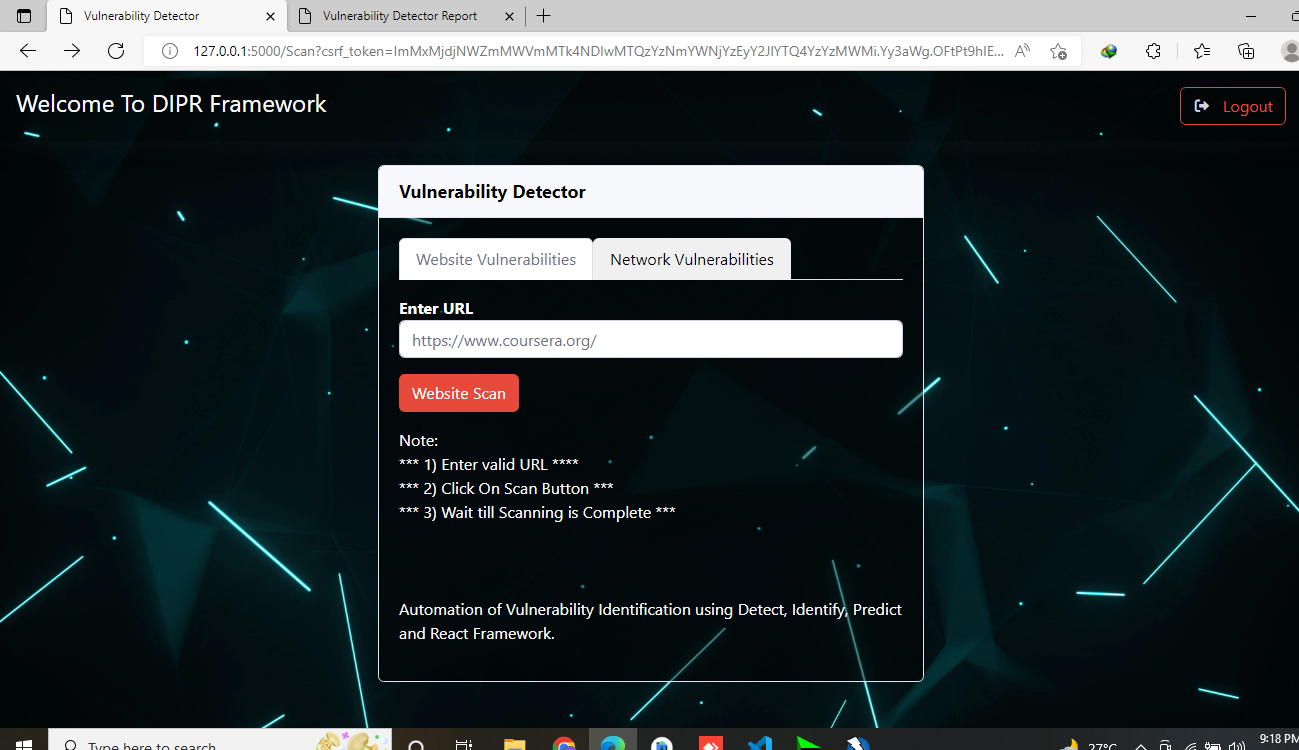


Figure 19: DIPR System Vulnerability Assessment Module

Figure 19 demonstrates that the DIPR system provides a user interface that is far less complicated and more user-friendly in comparison to OWASP Zap. Despite the fact that it performs and processes the same operations as OWASP Zap, the DIPR system is more user-friendly. An interface similar to the one shown in Figure 20 will be shown by the system whenever the user hits the button labeled "website scan." In order to complete the scan, some time will be required. Because of this, the DIPR tool is more multipurpose and provides a variety of functionalities with a single button, as seen in the previous paragraph. In order to utilize OWASP Zap for vulnerability screening, you will need to first get familiar with its complex capabilities. Additionally, the command line interface version of Nmap requires the drafting of complicated instructions in order to do several scans each time. Users are no longer need to commit complicated instructions to memory thanks to the DIPR system that has been presented, which integrates all of these functions into a single button.

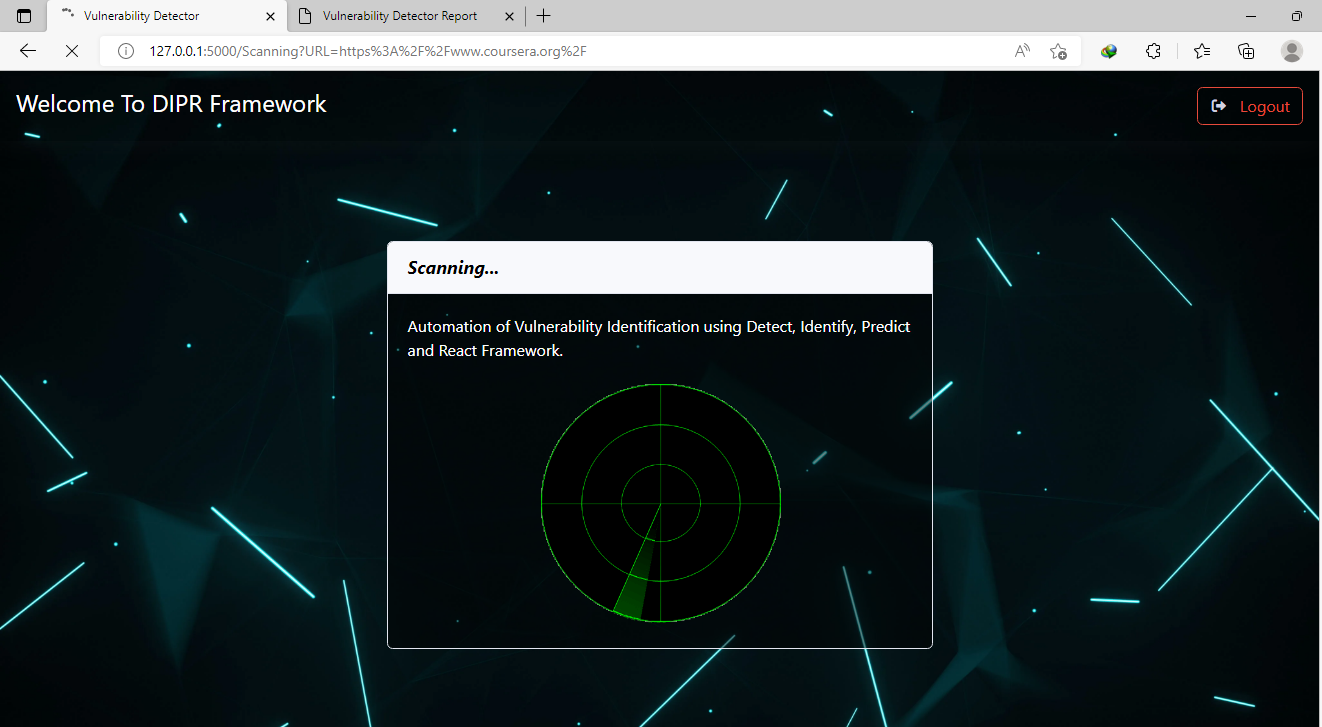


Figure 20: DIPR System Scanning Interface

The results of a vulnerability scan that was performed using the DIPR system on the website located at <https://www.coursera.org/> are shown in Figure 21. The page is comprised of three horizontal bars. A risk score is shown on the first bar, while vulnerabilities in web applications are displayed on the second bar. The third demonstrates flaws in the network.

In preparation for a more in-depth explanation, we have grabbed screenshots of each bar. Users are able to examine risk ratings that are classified as high, medium, low, informative, and false positive inside of the first bar. We found one vulnerability at the medium level, eight vulnerabilities at the low level, and two vulnerabilities at the informational level as a result of our scan. It is not possible to get a unified report of vulnerabilities in both web applications and networks using OWASP Zap and Nmap. The DIPR system, on the other hand, showed a comprehensive report that included both kinds of vulnerabilities.

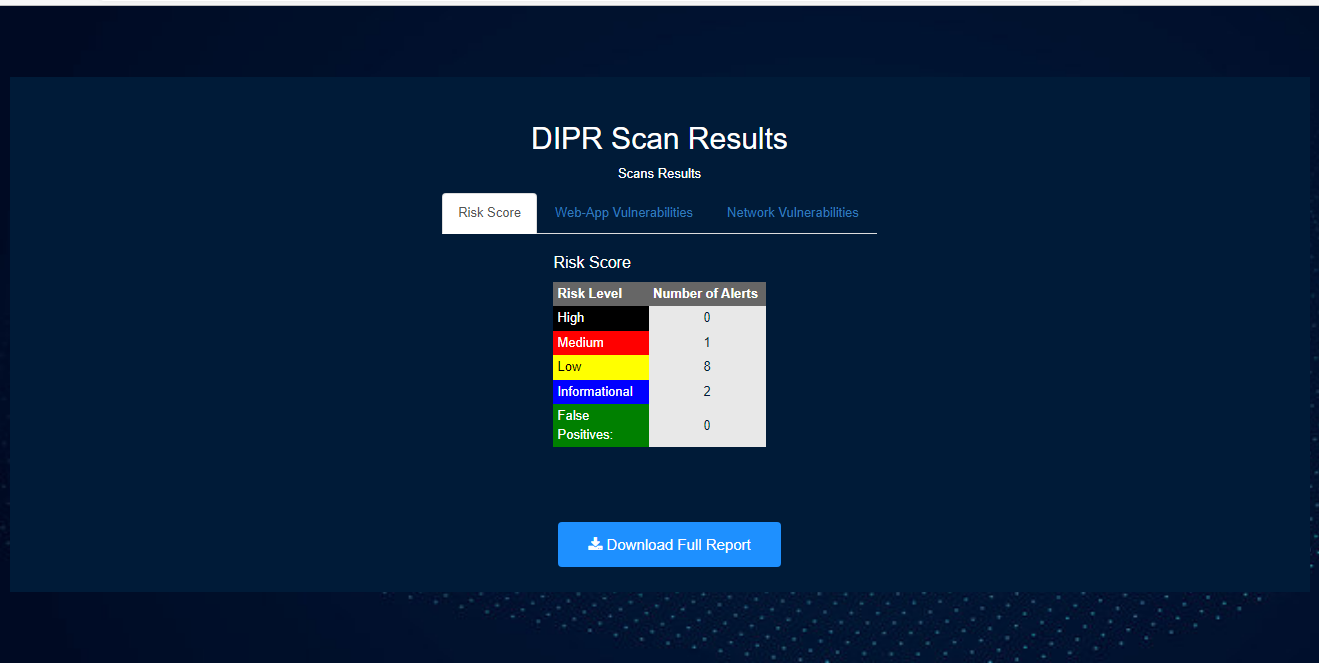


Figure 21: DIPR System Vulnerability Scan Results of Targeted Website

A list of the vulnerabilities that were found in the web application during the scan is shown in Figure 22. The fact that each sort of vulnerability that has been found has a specific name is something that may be seen. Within the findings of the scan, there are three instances of cross-domain misconfiguration that are of a medium degree. As a result, it brings to light possible security vulnerabilities associated with the sharing of resources across several domains. Furthermore, there are 69 low-level occurrences of the lack of anti-CSRF tokens, which indicates a vulnerability that might enable cross-site request forgery attacks. These instances are indicative of a vulnerability. In addition, the research identifies four low-level occurrences of each of the following vulnerabilities connected to cookies, which are as follows:

* There are cookies that lack the HTTP Only flag, which might potentially expose them to client-side scripts.
* Additionally, there are cookies that lack secure flags, which can make them vulnerable to interception over connections that are not secure.
* Cookies that do not have the same site attribute, which helps prevent cross-site request context from occurring.

A further 97 low-level instances of cross-domain JavaScript source file inclusion were discovered as a result of the scan. It recommends the incorporation of script files from many domains, some of which may not be trusted due to their origin. The "X-Powered-By" HTTP response header is responsible for 42 instances of low-level server information leakage. There is also one case of a cache-control header set that is either missing or not present at all, which may result in problems with caching or privacy concerns. One of the most important discoveries was that there were 1331 instances of low-level Timestamp leakage in UNIX. This demonstrates the possibility of system timestamps being exposed, which poses a threat to the security of the system. There are also 99 informative instances of suspicious comments in the code, which might disclose internal information that an attacker could use to their advantage. Additionally, there are two informational instances of cookies with a loose scope, which indicates that there may be possible security flaws in the definition of cookie scope.

The DIPR system not only discovers these vulnerabilities, as seen in Figures 21 and 22, but it also displays them in a way that is clear and easy to understand, with a variety of buttons that allow users to navigate through the findings. Having these capabilities not only makes the tool more useful for users, but it also gives them the ability to comprehend and resolve the vulnerabilities that have been found in a thorough manner. In addition, consumers are able to get the report of the scan formatted in HTML via the system.

A screenshot of a computer

Description automatically generated

Figure 22: Detailed Scan Results with Vulnerability Names and Instances

Detailed information on the vulnerabilities of the network is shown in Figure 22, which includes information about open ports, services, and versions. During the scan, it was discovered that port number 80 is open, and that an HTTP service is now operating on it. Sync-ack TTL 237 is connected with the service, and it makes use of Amazon CloudFront HTTPD, which may be recognized as CloudFront using the HTTP-server-header. A network vulnerability scan was performed on the website <https://www.coursera.org/>, and these results are a part of that survey. As can be seen in Figure 22, there is a button located at the bottom of this page that gives users the chance to download the whole report of the vulnerability scans. This enables users to get a comprehensive study of the vulnerabilities that are present in the network and online applications.

A screenshot of a computer screen

Description automatically generated

Figure 23: Target Network Vulnerabilities Detected

Following the completion of the report's download, customers will be able to see the comprehensive reports of the scans in the manner shown in Figure 24. They are able to scroll down to examine the results of the DIPR system's testing, which are extensive and comprehensive. There is evidence that the DIPR system is more effective. It is capable of doing numerous scans at the same time. On the other hand, in order to run independent web application and network vulnerability assessments, tools such as OWASP Zap and Nmap need a greater amount of time that is not available. While simultaneously doing extensive scans of websites and networks, the major objective of this project was to lessen the number of resources that were required to accomplish this task. Users are able to avoid utilizing several tools for distinct web application and network vulnerability scans by selecting the DIPR system as their tool of choice.

A screenshot of a computer error

Description automatically generated

Figure 24: DIPR System Comprehensive Report Overview

# Discussion

In order to achieve higher levels of efficiency, businesses have become more reliant on sophisticated computer systems. It is important to note, however, that increasing dependence on digital technologies also brought substantial security vulnerabilities. Considering the severity of these dangers, it is necessary to conduct a comprehensive vulnerability assessment prior to the implementation of any applications or networks. Through the use of an automated procedure that encompasses vulnerability detection, identification, and penetration testing, the DIPR system served to fulfill these requirements. The extensive study that has been conducted in this field, which was examined in the section under "Literature review," emphasizes how important it is to comprehend and deal with vulnerabilities. One of the distinguishing features of the DIPR system is its ability to combine static and dynamic analyzes in order to effectively identify and eliminate vulnerabilities in online applications and web networks. This integration simplifies the process by breaking it down into four primary steps: entering the URL, scanning for vulnerabilities, assessing the findings, and producing a report.

In order to successfully implement the DIPR system, extensive preparation was necessary. The handling of a wide variety of files, as well as specialized software and technology, were all necessities. During the development of the system's backend, we used Python and Flask. HTML, CSS, and JavaScript are examples of frontend technologies that are used by the system. In order to improve the system's scanning and vulnerability detection capabilities, the Application Programming Interfaces of the OWASP ZAP and Nmap tools have been included into the system. Coursera's website was used for testing the system, which was finished successfully. According to the findings, the system continues to efficiently scan for vulnerabilities in the target and has delivered a comprehensive report on the flaws it has found. When compared to other tools such as OWASP Zap and Nmap, the system stands out because to its efficiency and its ease of use. To conduct independent scans of web apps and networks, OWASP and Nmap demand more effort and knowledge than other tools.

The system that has been suggested makes a significant contribution to the area of cybersecurity by offering a method that is both user-friendly and effective for scanning web applications and networks for vulnerabilities. Users are able to analyze and plan mitigations with the assistance of thorough reports that are generated by it. A shift toward automated and enhanced security analysis is reflected in the architecture of the system. The DIPR system may be used by DevSec Ops teams in real applications for the purpose of performing proactive security checks. It may be used by administrators of networks for the purpose of conducting penetration testing in order to improve network security. The capability of the system to carry out both static and dynamic analysis guarantees that a comprehensive investigation of possible vulnerabilities will be carried out. The security of organizational assets, as well as the preservation of information's confidentiality, integrity, and availability, are thus supported by this.

This makes the DIPR system a significant accomplishment within the realm of security analysis. As a result, it provides a comprehensive solution for identifying vulnerabilities in online applications and networks, reducing the impact of these imperfections. In light of this, the DIPR system is a significant achievement for those who work in the field of security monitoring. The development of this technology and its successful deployment have highlighted the importance of establishing integrated security processes within the information technology environment. It has come to light as a result of the fact that this technology has been effectively deployed, which has brought it to light.

# Conclusion and Future Work

One of the most significant achievements in the field of cybersecurity is the DIPR system, which provides an integrated solution for identifying and mitigating vulnerabilities in both online applications and networks. It is precisely because of this capability that the DIPR system is essential. It provides a response that considers both of these qualities in relation to your query. It distinguishes itself from more popular tools like OWASP Zap and Nmap with its user-friendly interface and ability to conduct comprehensive exams. In addition, it is unique in that it can do extensive examinations. The automated vulnerability detection and analysis process in the presented system effectively reduces the need for human interaction. This, in turn, enhances the system's effectiveness. Consequently, the implementation of changes led to significant progress in the vulnerability management process. The system's successful installation and multiple tests have proven its ability to significantly enhance the security posture of enterprises. Multiple tests have proven this. Moreover, it holds the potential to serve as a valuable tool in the continuous battle to safeguard digital assets against emerging threats. This is a serious issue that is always evolving.

According to the developed plans, the next iteration of the system will prioritize the integration of machine learning. The use of machine learning algorithms will be applied via the utilization of this technology in order to find odd patterns in the behavior of applications or network traffic, which will signal the presence of possible security vulnerabilities or concerns. This will be accomplished through the utilization of this technology. The system will incorporate certain algorithms to automatically prioritize vulnerabilities based on their potential impact, ease of exploitation, and the criticality of the compromised system. This will be accomplished by using the system. This action aims to exacerbate the situation by adding another layer of complexity. This capability will be beneficial to businesses since it will allow them to concentrate, especially in high-risk regions.

# References

AlYousef, M.Y. and Abdelmajeed, N.T., 2019. Dynamically detecting security threats and updating a signature-based intrusion detection system’s database. *Procedia Computer Science*, *159*, pp.1507-1516.

Azshwanth, D. and Sujatha, G., 2022, January. A novel automated method to detect XSS vulnerability in webpages. In *2022 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-4). IEEE.

Backes, M., Rieck, K., Skoruppa, M., Stock, B. and Yamaguchi, F., 2017, April. Efficient and flexible discovery of php application vulnerabilities. In *2017 IEEE European symposium on security and privacy (EuroS&P)* (pp. 334-349). IEEE.

Bagyalakshmi, G., Rajkumar, G., Arunkumar, N., Easwaran, M., Narasimhan, K., Elamaran, V., Solarte, M., Hernández, I. and Ramirez-Gonzalez, G., 2018. Network vulnerability analysis on brain signal/image databases using Nmap and Wireshark tools. *Ieee Access*, *6*, pp.57144-57151.

Calzavara, S., Conti, M., Focardi, R., Rabitti, A. and Tolomei, G., 2019, June. Mitch: A machine learning approach to the black-box detection of CSRF vulnerabilities. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 528-543). IEEE.

Chen, H.M., Kazman, R., Monarch, I. and Wang, P., 2017. Can cybersecurity be proactive? A big data approach and challenges.

Dastres, R. and Soori, M., 2021. A review in recent development of network threats and security measures. *International Journal of Information Sciences and Computer Engineering*.

Guo, N., Li, X., Yin, H. and Gao, Y., 2020. VulHunter: An automated vulnerability detection system based on deep learning and bytecode. In *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers 21* (pp. 199-218). Springer International Publishing.

Liu, M. and Wang, B., 2018. A web second-order vulnerabilities detection method. *IEEE Access*, *6*, pp.70983-70988.

Rafique, S., Humayun, M., Hamid, B., Abbas, A., Akhtar, M. and Iqbal, K., 2015, June. Web application security vulnerabilities detection approaches: A systematic mapping study. In *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (pp. 1-6). IEEE.

Riadi, I. and Raharja, P.A., 2019. Vulnerability analysis of E-voting application using open web application security project (OWASP) framework. *International Journal of Advanced Computer Science and Applications*, *10*(11).

Wan, L., 2019. *Automated vulnerability detection system based on commit messages* (Doctoral dissertation).

Wang, Z., Wang, Z., Wang, Z., Zhang, Z. and Yang, T., 2022. Design and implementation of security vulnerability sharing platform based on web crawler. In *Proceedings of the 11th International Conference on Computer Engineering and Networks* (pp. 678-687). Springer Singapore.

Yaacoub, J.P.A., Noura, H.N., Salman, O. and Chehab, A., 2023. Ethical hacking for IoT: Security issues, challenges, solutions and recommendations. *Internet of Things and Cyber-Physical Systems*, *3*, pp.280-308.

Zhou, Z.J., 2020. Automatic Detection and Security Defense of Network Vulnerabilities by Using Crawlers. *Telecommunications and Radio Engineering*, *79*(14).