UNIVERSITY OF BAHRAIN

College of Information Technology

A Recommendation System for Web Application Vulnerability Penetration Testing Tools

A Thesis Submitted in Partial Fulfilment of the Requirements for the M.Sc. Degree in Cyber Security

Submitted by
Shayma Ahmed Altayran
202000199

Supervised by
Dr. Abdullah Alasaadi
Assistant Professor
University of Bahrain

Kingdom of Bahrain

March 2023

UNIVERSITY OF BAHRAIN

College of Information Technology

A Recommendation System for Web Application Vulnerability Penetration Testing Tools

A Thesis Submitted in Partial Fulfilment of the Requirements for the M.Sc. Degree in Cyber Security

Submitted by
Shayma Ahmed Altayran
202000199

Supervised by
Dr. Abdullah Alasaadi
Assistant Professor
University of Bahrain

Kingdom of Bahrain
March 2023

University of Bahrain
Deanship of Graduate Studies
And scientific Research

جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Dissertation Deposit Form

Author Name:	Shayma Ahmed Altayran		
Email:	202000199@stu.uob.edu.bh	Contact No:	00966540593036
College:	**Information Technology**		
Department:	**Computer Science**		
Thesis Title:	A Recommendation System for Web Application Vulnerability Penetration Testing Tools.		
Year:	2022		

Author's Declaration: I agree to the following conditions:

The above thesis will be available (in the University of Bahrain library and externally) and reproduced as necessary at the discretion of the University of Bahrain. It may also be digitized by the University of Bahrain and made available via the university's repository or via other parties on the Internet.

This condition shall apply to ALL copies including electronic copies.

The above thesis has been provided on the understanding that it is copyright protected material and that no quotation from this thesis may be published without proper acknowledgement.

Signature of the Author:	Shayma Altayran	Date:	28-Sep-22

DECLARATION

I declare that this work is the result of my own investigation and that it has not already been accepted in substance for any degree, nor is it currently submitted for any degree.

Signed: [illegible] Date: 29/09/2022

Shayma Ahmed Mohammed ALTAYARAN (Candidate)

The Thesis defense committee considers the thesis titled:

Submitted by **Shayma Ahmed Mohammed ALTAYARAN** to the College of Information Technology is satisfactory and acceptable for the Master of Science in Cyber Security.

The Defense Committee:

[illegible]

Dr. Abdulla Ahmed Alasaadi

(Supervisor)

Department of

University of Bahrain

Ibrahim Kamel

(Dr.Ibrahim Kamel)
(External Examiner)

Sharjah University

RIYADH Ksantini

(Dr.RIYADH Ksantini)
(Internal Examiner)

Department of

University of Bahrain

ABSTRACT

The growing technology has shifted the business focus from manual to automated ways. This fact increased the development of software applications in every field of life. However, there are growing cyber threats to the applications by malicious users and attackers. The developers of software applications have to handle the vulnerabilities in source code during the development and penetration testers have to identify threats during or after deployment. A large number of tools exist for assessing threats and vulnerabilities in web applications. However, the selection of appropriate tools for code analysis and vulnerability detection remains a big challenge for software developers and penetration testers.

This research initially investigates different types of vulnerabilities and attacks that exist in software applications. Based on the requirements, this research proposes a mathematical representation using Tensor that considers different features of web applications, security tools, and deployment infrastructure.

The dataset is prepared by extracting the needed features manually from open-source web applications, popular security tools in the domain, and infrastructure modes, then used to train five different machine learning classifiers. Feature optimization is applied to the dataset to reduce the number of features while achieving higher prediction accuracy values. Random Forest (RF), Decision Tree (DT), Support vector machines (SVM), and Naive Bayes (NB) classifiers provided the highest accuracy values. The trained models are used to predict suitable tools for randomly selected open-source web applications. This research found that RF achieved 97% and DT showed 93% prediction accuracy, while SVM 89% and NB achieved 86% prediction accuracy. The selected tools are further validated manually and found the same suitable security tool. The results of the study are promising and provide a strong foundation for cyber security tools recommender systems.

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr.Abdulla Alasaadi for his great support, help, and motivation. His guidance helped me in all the time of my research and during the writing of this thesis.

I would like to thank my husband for his endless support. I would like also to thank my mother and father, my brothers and sisters.

DEDICATION

To my leader, my crown, my father Dr.Ahmed Altayran, who lighten this path for me and guided me with wisdom. To my mother for supporting me and my family.

To my great husband, without his patience, understanding, support, and most of all love, this work would have not been possible.

To my sons Azzam, Faris, Yaser and Sami, and my daughters Aseel and Somaya for their encouragements.

Lastly to my self, for holding on my dreams until they came true. Thank you.

PUBLICATIONS

During the years of this M.Sc. I published two conference papers, as following:

CONFERENCE PAPERS

1. Altayaran, S. A., Elmedany, W. (2021, October). Integrating Web Application Security Penetration Testing into the Software Development Life Cycle: A Systematic Literature Review. In 2021 International Conference on Data Analytics for Business and Industry (ICDABI) (pp. 671-676). IEEE.
2. Altayaran, S., Elmedany, W. (2021, November). Security threats of application programming interface (API's) in internet of things (IoT) communications. In 4th Smart Cities Symposium (SCS 2021) (Vol. 2021, pp. 552-557). IET.

LIST OF ABBREVIATIONS

CWE	Common Weakness Enumerations
OWASP	Open Web Application Security Project
WAPTT	Web Application Penetration Testing Tools
FP	False Positives
TP	True Positives
FN	False Negatives
XSS	Cross-Site Scripting
SAST	Static Application Security Testing
DAST	Dynamic Application Security Testing

Chapter 1

Introduction

1.1 Overview

The massive growth in the evolution of web applications inherently guides the consideration of security factors related to these applications. Testing web applications is essential to guarantee security, identify weaknesses, and ensure that the application performs as expected (Spadini, Palomba, Zaidman, Bruntink, & Bacchelli, 2018). Application testing is a vital and expensive activity in the Software Development Life Cycle (SDLC). Furthermore, insufficient application testing usually has considerable risks and outcomes (Thota, Shajin, Rajesh, et al., 2020). The application testing persists through the SDLC, unlike when the testing is performed at the end of application development. Moreover, the testing in the development of an application is shown in figure 1.1.

Enterprise applications often suffer from frequent flaws at design and implementation levels that fail to merge security throughout the development procedure. Unfortunately, a late life-cycle penetration testing discloses problems too late, for both time and budget that hardly restrain the possibilities for remedy. On the other hand, not fixing bugs and flows in this stage is costly.

Coding and programming are significant phases in SDLC; if there is

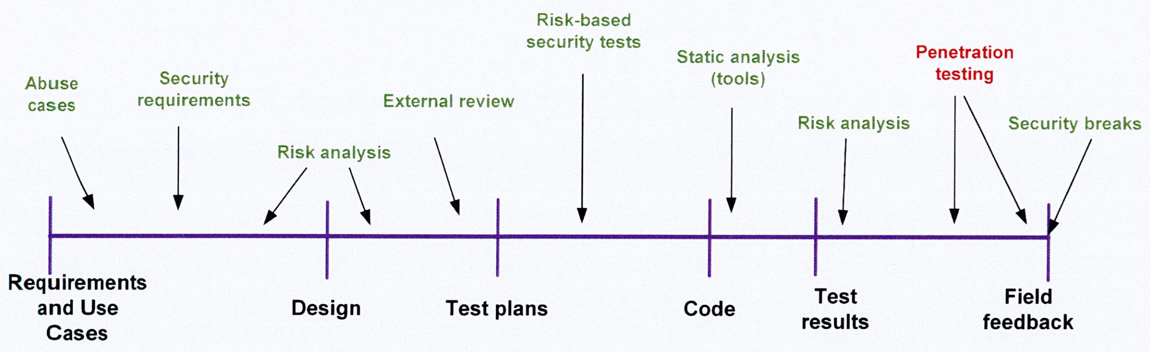


Figure 1.1: Security Throughout the SDLC (PCIDSS, 2022)

any weakness in the web application, attacks can happen. Nevertheless, understanding such vulnerabilities is challenging due to a lack of knowledge regarding the appropriate vulnerability assessment tools. Moreover, there is a long list of tools currently available that can identify these vulnerabilities. This thesis focuses on the top 10 vulnerabilities listed in Open Web Application Security (OWASP-SAST, 2022).

In addition, the tools that are used to detect these vulnerabilities will also be identified; based on input parameters from application developers, such as known vulnerabilities, programming language, the operating system of the web application, deployment infrastructure settings, appropriate vulnerability assessment, and penetration testing tools will be provided based on a proposed recommendation system.

The recommender system aims to provide the software developer or penetration tester with the best suitable security assessment tool in a given scenario. When the developer uses the same tools that the penetration tester use, it will make his developed web application more secure; however, when applying penetration testing while an application is in development and before launching, it may reveal critical vulnerabilities that need to be fixed so that the application can be released free of vulnerabilities (Neil Daswani, 2021).

1.2 Research Problem

Application development is a complex and time-consuming activity. Therefore, testing the web application during programming requires extra time and effort. This assessment is essential because, increasingly, people depend on online services such as e-commerce, logistics, communication, banking, and many other areas. Data reveals enormous financial losses due to cyber-attacks; statistics of these losses are shown in Table 1.1.

The description of these losses indicates the need for handling software vulnerabilities at the code level. However, there are many tools to find vulnerabilities targeting web applications at the code level. A list of vulnerability scanning tools is provided on (OWASP, 2022). OWASP project reports more than 85 Dynamic Application Security Testing (DAST) tools as well as 109 Static Application Security Testing (SAST) tools. However, SAST and DAST tools cannot be specified without the knowledge of the programming language for the particular application development, where the web application has to be deployed for the web server, the operating system where the web server is hosted, and the physical Information and Communication Technologies (ICT) infrastructure is available for deployment.

The data on ICT infrastructure is required because it presents the information needed for the availability of firewalls, encrypters, Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS). This information helps application developers concentrate more on coding than on particular security scenarios. Due to essential factors in choosing the right vulnerability assessment tool, it is necessary to determine the most accurate tool that helps programmers test their web applications. The application developers are doubtful about selecting these tools due to their vast number and similar functionalities. On the other hand, developers have to focus more on productivity than security. As a result, many developers neglect security to provide the required application at the project time, resulting in more vulnerable web applications being exposed to severe attacks. Furthermore, improving the safety of web applications demands providing the right testing tools for developers to test their web applications during development and after the release. These issues faced by the application developer signify the need for an automated way that provides an appropriate security assessment tool to the developer with accuracy values. This tool will take input parameters from the developer and recommend a suitable tool. Thus, the research question in the thesis can be formulated as follows:

- RQ1. What is the most accurate and suitable tool to test the web application code in the given programming language?

Table 1.1: Financial Losses (in USD million) with Location (Aldasoro et al., 2022)

	Asia	**Africa**	**Americas**	**Europe**	**Oceania**	**Total**
Frequency	3739	345	102459	7627	1024	115415
Total losses	4079.95	1793.37	28988.08	4299.89	362.41	39523.82
Mean loss	19.43	99.63	2.06	6.29	4.53	2.62
Std. Dev. of loss	89.34	399.53	47.41	44.36	19.64	79.97

1.3 Research Objectives

Information is one of the significant assets in today's industrial applications. This information is delivered through web applications to the stakeholders (Aliero, Qureshi, Pasha, Ahmad, & Jeon, 2020). Nevertheless, attackers desire to acquire non-legitimate credentials to the data by manipulating vulnerabilities in the application's source code.

The attacks on web applications are increasing day by day. Although assuring the security of web applications is similar to software development, static analysis of the source code and penetration testing are fundamental approaches. However, selecting an appropriate tool that detects the vulnerabilities in the source code is a complex process. This challenge evolves manifold in the presence of a large number of open-source and proprietary tools. Also, different tools can give different results when scanned on the same target, so it is essential to define the most accurate tool to be used in each penetration testing method.

To develop robust web applications against cyber attacks, this research aims to develop a recommendation system framework that provides information about a vulnerability assessment and the penetration testing tool based on input values. This framework is based on an $n - dimensional$ matrix of attributes where n represents a feature such as vulnerability, operating system, ICT infrastructure, web server, etc. This framework will be evaluated using machine learning-based implementation where various classifiers will be applied to the input dataset and deliver the accuracy values as output; results acquired from machine learning classifiers will be validated via penetration testing of available open-source web applications.

Overall, this recommendation system will assist application developers in testing the web application during the coding stage, and after the development, this system will allow the penetration tester to accomplish different types of attacks knowing the vulnerabilities recognized through the vulnerability assessment tool. This effort will help the developer minimize the developing time and ensure the web application's security attributes.

1.4 Contribution to the Field

The contribution of this thesis is to design a recommendation system with a machine learning model for selecting a suitable security penetration testing tool for the web application developer. In the web application penetration testing field, the thesis assembles the following key contributions:

- To help the developer in selecting the best security assessment tool from many available tools on the Internet to perform the security testing on the web application. A recommendation system model is introduced to help select the most suitable tool for testing the developed web application.

- The dataset was prepared by the researcher by extracting real features from open-source web applications and penetration testing tools.

- The mathematical representation is evaluated using five different classifiers for given metrics.

- This model is also validated by manual testing for the selected open-source web applications which proved the effectiveness of the tool selection by the recommendation system.

1.5 Thesis Outline

As a summary of this chapter, The thesis is addressed as follows:

- Chapter 1 Introduction: it delivers the research background, the objectives specified for the research, and the research question.

- Chapter 2 Literature Review: it provides details of related definitions of code security, static analysis, penetration testing, vulnerabilities, and attacks. It also includes statics on various vulnerabilities and attacks during the last few years. Reviewing existing vulnerability assessment and penetration testing tools is also part of the literature review. Moreover, this chapter aims to provide a background on the field of knowledge and the works done in previous studies and the importance of the research question.

- Chapter 3 Research Methodology: This chapter discusses different parallel research methodologies and selects suitable methodologies for current research. This chapter provides the methodology layout adopted in the following chapters.
- Chapter 4 Framework For Recommendation System: this is an important chapter that represents the input parameters of the proposed framework, the processes, and the results. The metrics used in the research will be defined and justified. Finally, the implementation of the framework will be provided, followed by validation via penetration testing of the web applications.
- Chapter 5 Results and Discussion: This chapter provides the results and discusses the research findings, research limitations, and future work will also be identified in this chapter.

Chapter 2

Literature Review

The security life cycle starts in parallel with SDLC (PCIDSS, 2022). Generally, security is ignored and thought to be ensured through testing after the deployment of the applications. Late testing may result in the identification of defects and bugs very late and impossible to r evert. Different security testing processes are adopted at different stages of software development. The research work focuses on risk analysis, source code security analysis, and penetration testing during and after software development. Different related definitions of tools and processes will also be provided.

2.1 Risk Analysis

Different types of risks are associated with the software. These risks are related to the performance, security, quality, and usability of the application to be developed. These are non-functional requirements and are referred to as implicit constraints imposed by the customer. Later the identification of risk in SDLC, higher the chances of failure of the application. Different tools, techniques, frameworks, and approaches have been proposed in the literature to cope with certain types of risk. A framework for quantitative risk assessment for web applications has been proposed in (De Gusmão, Silva, Poleto, e Silva, & Costa, 2018). The research shows that quantitative models are not sufficient to detect and avoid security risks in web applications. The results are drawn from the analytical model.

A requirement diversity and traceability mechanism for risk assessment in complex software development for medical devices have been proposed in (Regan, Mc Caffery, Mc Daid, & Flood, 2013). A risk control module is introduced at the requirement specification phase during the software development process. This model follows the standards define by regulatory bodies. A methodology based on the privacy-by-design principle is proposed in (Notario et al., 2015). The methodology operates throughout the SDLC with a particular focus on the requirement analysis phase. To provide a risk-based testing taxonomy framework to understand, reorganize, and compare risk testing methods to help select and tailor specific purposes, has been mentioned in (Dahiya, Solanki, & Dhankhar, 2020). This framework is generic and can be applied at any stage of SDLC. The major components of the framework are risk (drivers, assessment, and test process).

2.2 Static Analysis

The source code is a common artifact in all software projects. Code review from the perspective of security reasons and the architectural risk analysis is essentially a software best practice (Tahaei, Vaniea, Beznosov, & Wolters, 2021; Nunes et al., 2018). Coding mistakes are common among all developers, Most of the time, the compiler identifies the error and the programmer corrects the error. The development progress in the converse of most of the security vulnerabilities that exist in the code. The cost of recovery increases with an increase in delay in the detection of these vulnerabilities.

The role of programming language is vital and the languages are designed to serve specific purposes. For example, the major focus of the design of languages such as C or C++ is efficiency and portability while it is little or no handling of run-time errors (Rassokhin, 2020). An example of run time error is the bound checks of arrays in C and C++. Null pointer exception is also a common problem. To avoid these errors, programmers are required to place checks in code which is labor intensive and still not 100% safe. A robust programming language ensures that checks by a programmer are not required and error handling is done by the language itself.

Static analysis means automated analysis of run-time effects of code without running it (Siavvas, Gelenbe, Kehagias, & Tzovaras, 2018). These properties result in early ending or weak results of the application. These possessions do not contain address syntax errors or straightforward type errors. Static analysis can be used to inspect why program performance is aborted due to unexpected run time errors. The static analysis does not check for the functional requirements defined in the software requirement specification (SRS) document. Static analysis also varies from dynamic analysis, involves examining the application based on its execution, and contains properties such as testing, monitoring the performance, errors isolation, and debugging. Although the static analysis does not certify the lack of run-time errors; moreover, it can lower the requirement for testing/detecting errors that, in practice, cannot be discovered by testing. Therefore, static analysis cannot replace software testing. A variety of tools exist that perform Static Analysis without running the application. Theoretically, the source code or binaries of an application could be examined. However, not all tools can decode the binaries of a software program.

The static analysis of code is recommended to be part of the software development process and helps the developers to perform control flow and data flow analysis (Sherman & Dwyer, 2018), pattern matching, and other bug identification.

2.3 Penetration Testing

Permission of access to resources is the main difference between the attacker and the penetration tester. The penetration tester works with the permission of the owner of resources and is responsible for providing the report with the major goal of increasing the security of the application.

"Penetration testing is the process of discovering and exploiting the vulnerability found in the tested application in an authorized and systematic method." (Khera, Kumar, Garg, et al., 2019).

"Penetration test is the authorized, scheduled, and systematic process of using known vulnerabilities in an attempt to perform an intrusion into host, network, or application resources. The penetration test can be conducted on internal (a building access or host security system) or external (the company connected to the Internet) resources. It normally consists of using an automated or manual tool-set to test company resources." (Weidman, 2014).

Penetration testing is different from vulnerability scanning. The differences are shown in Table 2.3.

The penetration testing is conducted after the release of the application with the consent of the owner of the application. The penetration tester or ethical hacker is trustworthy to report the activity to the application owner.

2.3.1 Types of Penetration Testing

The researcher in (Weidman, 2014), addressed that the main types of penetration testing are:

- Physical Penetration Testing

Table 2.1: Vulnerability Scan Versus Penetration Test

	Vulnerability Scan	**Penetration Test**
Purpose	Identify, rank, and report the vulnerabilities in an application that, if exploited, may result in compromising the application.	Identify ways to exploit vulnerabilities to defeat the security of the application components.
When	At a significant release of the application or at least quarterly	annually and upon significant change
How	Automated tools with the support of manual verification and identification of issues	A manual process which may include the support of other automated tools and result in a comprehensive report.
Report	Probable risks for available vulnerabilities are classified according to CVSS/NVD standards. External scans are conducted outside the organization, while internal scans are performed internally.	Each confirmed vulnerability is explained, as well as each potential problem found. Furthermore, any detailed threats that vulnerability may pose, including typical techniques, how and to what the scope may be manipulated.
Duration	Moderately, it takes a short period, ordinarily several minutes per host.	Engagements can take weeks, relying on the test's scope and the tested environment's size.

- Social Engineering Testing
- Web Application Penetration Testing
- Network Penetration Testing

2.3.2 Penetration Testing Approaches

There are different approaches to penetration testing such as white box, black box, and grey box. In white-box testing, the client provides complete information about code, networks, and protocols. In black-box testing, no information is provided to the penetration tester. In grey-box testing, the tester may be provided with partial details. A detailed review of the penetration testing terminologies and guidelines can be found at (Baloch, 2017).

- Black box testing is testing the application without any given knowledge about it in any way; it is similar to what the real hacker does when he tries to exploit an application (Khera et al., 2019).
- The white box testing technique covers every design element and checks the related code with every possible way of execution; the tester knows everything about the application usually, it is done for unit testing by the developers (Mansour & Houri, 2017). Moreover, This method demands an understanding of the programming language.
- Grey box testing is when the tester has a piece of partial information about the network that will be tested, considered a combination of both black and white testing. Furthermore, these tools can scan both the white box / black box environments automatically. In addition, the code review in black box testing only scans the application behavior for anomaly detection.

Recent studies show black-box testing tools can deliver security outcomes at a certain status and specified qualifications, e.g., detecting Cross-Site Scripting (XSS) and processing a few components such as (flash, Java Applet) and identifying typical application defects (Khamdamovich & Aziz, 2021; X. Li & Xue, 2011). In contrast, another study added that black-box testing is a complex process (Tripp, Weisman, & Guy, 2013).

The authors in (Seng, Ithnin, & Mohd Said, 2018) conducted a test examining a vulnerable web application for both scanners, and the test provided significant results showing the performance between black and white box scanners. In addition, white-box scanners' code visibility delivered better coverage than black-box scanners.

To explain the white-box and black-box tool's performance, (Seng et al., 2018), an experiment is conducted on a vulnerable web application. As a result, white-box tools achieve better test outcomes because of code visibility. Hence, black-box tools deliver false negatives. However, white-box tools are tolerant of false positives.

2.4 Common Vulnerabilities in Web Applications

More and more business is trending towards Information Technology (IT), and the security of web applications is becoming more and more important. Many web-based solutions are being utilized in operating sensitive financial data and medical devices. The downtime of these applications can cause millions of dollars in damage, so it is important to protect these applications from illegal attempts from malicious users.

The web applications follow a client-server model for message passing and input handling. The client-side input is handled by a server-side program or database. Although the libraries offered by different languages provide built-in security, it is possible to make logical programming errors that lead to vulnerabilities such as SQL injections (Balasundaram & Ramaraj, 2012) and cross-site scripting attacks (Gupta & Gupta, 2017).

2.4.1 Causes of Vulnerabilities

The most common problem identified in web applications is unchecked input which causes attacks by malicious users. The attacker may achieve one or two goals by exploiting unchecked inputs.

2.4.1.1 Inject malicious data into Web applications

This can be done by using the following common methods:

- URL manipulation: temper the URL by passing fabricated input parameters to get desired response.
- Parameter tempering: pass specially designed input values to the HTML forms provided by the Web applications.
- Hidden field manipulation: set some hidden field values to achieve desired results.
- Cookie poisoning: the data in cookies at the client side is manipulated by injecting malicious code and sent to the server.
- HTTP header tempering: manipulate parts of the HTTP header sent to the application.

2.4.1.2 Manipulate applications using malicious data

Common methods used are:

- SQL injection: modify the SQL queries and send the query to the server to get desired results.
- Command injection: exploit user inputs to execute shell commands.
- HTTP response splitting: exploits the behavior of the user for a given input and expected output or Web cache poisoning attack.
- Cross-site scripting attack: due to unverified programming mistakes, the malicious scripts are injected into the web application.
- Path traversal: check the user inputs and exploit to access the files from the Web server.

2.5 OWASP Top 10 Vulnerabilities

The OWASP Project is the best starting point for testing web application security. As mentioned in (J. Li, 2020), the well-known ten risk-rated vulnerabilities are specified by the OWASP. It was updated in 2021. Table 2.2 shows the OWASP Top 10 vulnerabilities list in 2017. The names in the latest list are altered to concentrate on the core cause rather than the symptom. In a study (Qasaimeh, Shamlawi, & Khairallah, 2018), researchers recognized that most scanners illustrate vulnerabilities to these categories: Critical, High, Medium, Low, and Informational.

2.6 Web Applications Security Testing

To guarantee web applications' security, they must be tested for exploitation or any well-known vulnerabilities; a web application includes different data at various levels. For example, the data delivered in web applications can be from a personal user, such as private data, to extensive enterprises and companies that carry significant portions of sensitive information, such as critical infrastructure. Thus, security for web applications is vital. Static analysis of source code and penetration testing of the deployed web application is the most widely accepted practice to secure web applications.

2.6.1 Web Application Attacks

At the start of the World Wide Web (WWW) and the Internet, it only hosted websites with static content. The web browser's mechanism was designed to reveal those static contents; in other words, they were a one-way data stream from the server to the browser; if the websites were attacked, the attacker would not get any sensitive data; because the details held on the server are open for the public. However, nowadays, websites have devel-

Table 2.2: OWASP Top 10 Vulnerabilities (OWASP-SAST, 2022)

Rank (2021)	**Vulnerability Name (2021)**	**Rank (2017)**	**Vulnerability Name 2017**	**Description**
A01-2021	Broken Access Control	A05-2017	Broken Access Control	94% of applications were tested for some form of broken access control.
A02-2021	Cryptographic Failure	A03-2017	Sensitive Data Exposure	Focus is on failures related to cryptography which leads to data exposure or compromising the application.
A03-2021	Injection	A01-2017	Injection	94% of web applications were tested for some types of injection.
A04-2021	Insecure Design	-	(New)	Risks related to design flaws
A05-2021	Security Misconfiguration	A04-2017	XML External Entities	90% of web applications were tested for misconfiguration
A06-2021	Vulnerable and Outdated Components	A06-2017	Using Components with Known Vulnerabilities	It is the only category that does not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs
A07-2021	Identification and Authentication Failures	A02-2017	Broken Authentication	Includes CWEs that are more related to identification failures
A08-2021	Software and Data Integrity Failures	-	(NEW)	Making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity.
A09-2021	Security Logging and Monitoring Failures	A10-2017	Insufficient Logging and Monitoring	Can impact on visibility, incident alerts, and forensics
A10-2021	Server-Side Request Forgery	-	(NEW)	The data shows a relatively low incidence rate with above average testing coverage

oped into powerful platforms; websites on the internet are real applications. They are considered web applications that are highly operative and have communication between users via the browser and the server. They sustain trades, logins, and much more (Altayaran & Elmedany, 2021).

As the number of web applications grows, the attacks on those web applications also increase. In the era of COVID-19, the number of cyber-attacks has increased tremendously. These attacks targeted thousands of people working from home. Working from home has discovered levels of cyber security threats and challenges by enterprises and individuals never encountered before. Cybercriminals manipulate the chance to extend their attacks and target users at all levels, from employees to critical infrastructure such as health care services. As a result, web applications witnessed a rising number of threats and episodes.

Cyber-attacks increased by 600% by March 2020. companies and individuals faced challenges securing their data in a way never envisioned before (Lallie et al., 2021). In addition, attacks are extending significantly due to the evolution of knowledge, technology, and the existence of weaknesses and vulnerabilities in web applications as well as exploitation techniques. As a result, testing web applications is an essential and demanding mission (Futcher & Von Solms, 2008).

The Data Breach Investigations Report (Verizon, 2022) collected and analyzed in total over 914,547 incidents, 234,638 breaches, and 8.9 TeraBytes (TBs) of cyber security data since 2018 (15 years). In addition, this report mentioned that the top action vectors in incidents of data breaches are hacking web applications more than 60%, as shown in figure 2.1.

According to a recent comprehensive analysis of the data induced from testing, more than 15 million web application security in organizations throughout 2021 by NTT Application Security (Staff, 2022); they found that 50% of all web applications were vulnerable to at least one critical vulnerability.

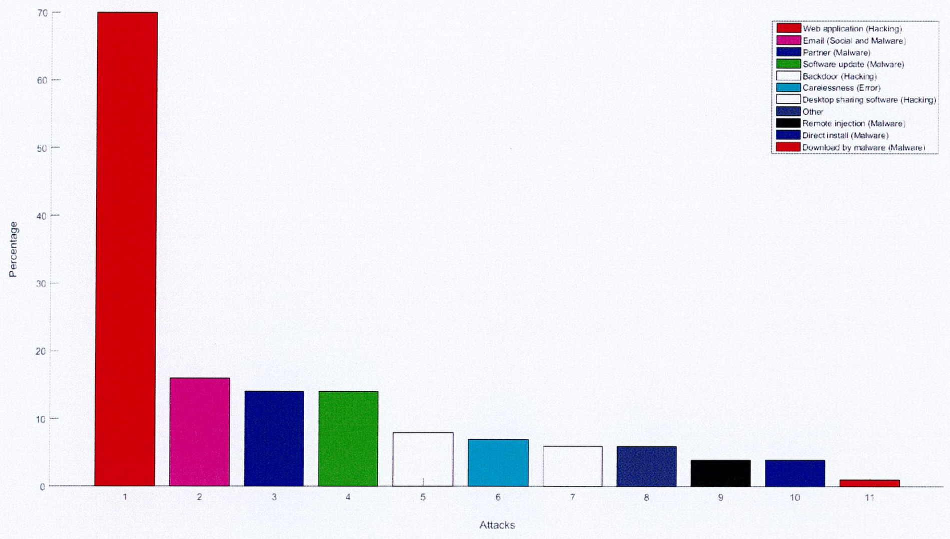


Figure 2.1: Top Action Vectors in Incidents and Breaches (Verizon, 2022)

In the Global Threat Analysis Report, (Radware, 2022), from 2020 to 2021, malicious web application requests increased by 88%. The blocked security violations by the 2017 OWASP Top 10 in 2021 are shown in figure 2.2, where injection attacks are the highest web application security risks based on the 2017 OWASP Top 10.

On the other hand, with Broken Access Control OWASP application security risk, more than 75% of web application attacks were characterized as injection attacks and broken access control. Furthermore, banking, finance, and SaaS providers are the most attacked enterprises in 2021, accounting for more than 28% web application attacks. Retail and high-tech enterprises rated third and fourth, each with nearly 12% of the web security occurrences observed by manufacturing (9%), government (6%), carriers (6%), and transportation (5%), as represented in figure 2.2.

As a reaction to the comprehensive scope of attacks associated with the pandemic, in 2020, the United Kingdom's National Cyber Security Centre (NCSC) and the United States Department of Homeland Security (DHS), Cyber security, and Infrastructure Security Agency (CISA) issued a collab-

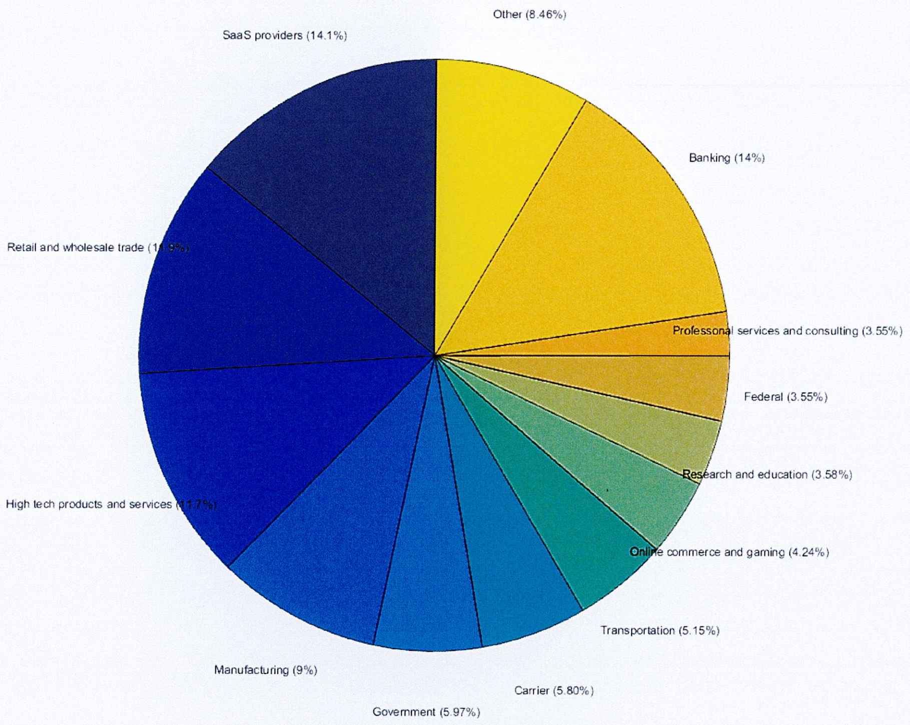


Figure 2.2: Web Application Attacks per 2017 OWASP Top 10 Category (Verizon, 2022)

orative consultative on how cyber-criminals are manipulating the present COVID-19 pandemic. These consultative documents examined malware, phishing, and web application attacks. Moreover, organizations are hugely challenged to develop suitable security measures for protection and response (Lallie et al., 2021).

2.6.2 Web Application Testing Tools

Testing web application security is a large field that needs attention from researchers because the community depends more and more on web applications. In addition, each testing tool is different in the kinds of attacks it can perform, for example, port scanning, application detection, and known vulnerability scanning (Suto, 2010).

Studies done by (Vassallo et al., 2018) confirmed that working on open-source software gives efficient results, such as RIPS (Re-Inforce PHP Security) and OWASP WAP (Web Application Protection Project). In addition, different methods are used to detect vulnerabilities, such as penetration testing (Ibrahim & Kant, 2018; Setiawan & Setiyadi, 2018), and static and dynamic code analysis (J. Li, 2020).

One of the best coding practices is using code review to detect vulnerabilities from the beginning of programming the web application. The researchers (Tripp et al., 2013) declared that the static code review analysis approach is used to secure applications. However, the researcher determined many concerns associated with testing input data validation in a web application because of the enormous number of payloads that can exploit vulnerabilities; detecting precise inputs utilized as an attack vector can be complicated and it is not easy to search for them manually. In (Apriani, 2019), the researcher indicates that the proper tools influence testing results due to their performance, which will depend on the level of vulnerability of the application, and the drawback of application penetration testing is that it takes ample time to inspect the whole application.

2.7 Security Testing Frameworks and Solutions

Different security testing frameworks that exist for software applications are discussed in the section. The results of a study show the trends of cyber threats from the Year 2012 to 2018 (Kettani & Wainwright, 2019). The frameworks proposed considered the threats described in Table 2.3.

A procedure for immediate impact and reversal testing is produced in (Amalfitano, Fasolino, & Tramontana, 2011). This procedure stands on a crawler that automatically assembles a benchmark of the application GUI and gets test cases that can be automatically conducted. A model to detect

Table 2.3: Annual Changes in Ranking of the Top fifteen threats (European Union Agency For Network and Information Security) (Kettani & Wainwright, 2019)

Year	2018	2017	2016	2015	2014	2013	2012
Malware	1	1	1	1	1	2	2
Web-Based Attacks	2	2	2	2	2	1	1
Web Application Attacks	3	3	3	3	3	3	3
Phishing	4	4	6	8	7	9	7
Denial of Service	5	6	4	5	5	8	6
Spam	6	5	7	9	6	10	10
Botnets	7	8	5	4	4	5	5
Data Breaches	8	11	12	11	9	12	8
Insider Threat	9	9	9	7	11	14	-
Physical Manipulation/Damage/Theft	10	10	10	6	10	6	12
Information Leakage	11	13	14	13	12	13	14
Identity Theft	12	12	13	12	13	7	13
Cryptojacking	13	-	-	-	-	-	-
Ransomware	14	7	8	14	15	11	9
Cyber Espionage	15	15	15	15	14	-	-
Exploit Kits	-	14	11	10	8	4	4

the conflicting behavior of different plugins and show different behaviors as expected is developed in (Nguyen, Kästner, & Nguyen, 2014). The author executed the test cases on many configurations and found that while plugin exchanges exist, a significant quantity of sharing allows a variability-aware implementation to rise to 250 configurations within seven minutes of running time.

A framework for testing single sign-on vulnerabilities in web applications has been proposed in (Zhou & Evans, 2014). The vulnerabilities associated with single sign-on based on third-party APIs are evaluated on twenty thousand top-ranked websites. The results show that over 20% of the websites are found to be suffered from at least one vulnerability. Furthermore, a black-box security testing framework for web applications based on HTTP protocol is proposed in (Aliero, Ghani, Qureshi, & Rohani, 2020). The proposed tool that executes attacks depending on the attack surface, such as HTTP requests referred to as Uniform Resource Identifiers (URIs), the web applications take input that can have payloads for testing attack vectors; such as buffer overflows, SQL injection, privilege escalation, and arbitrary code execution. Besides, the tool can analyze the existence or absence of vulnerabilities by considering HTTP responses from the web application. One of the disadvantages of this study is that the black-box testing performed at the end of the application development activity does not provide an understanding of the internal working of the application.

2.7.1 Cyber-Attack Recommendation System

Researchers adopted different approaches for the selection of tools for cyber assessment. A graph-based attack detection approach is used in (Polatidis, Pimenidis, Pavlidis, Papastergiou, & Mouratidis, 2020). The graph shows all possible paths that an attacker can adopt for launching an attack and gaining unauthorized access to the cyber system. A recommendation system is proposed for the detection of privilege escalation attacks. However, the effectiveness of this system is tested for privilege escalation attacks only. Another work emphasizes that the recommendation system cannot itself make decisions (Gadepally et al., 2016). It rather has other actors that participate in the decision-making process. This work is discussed in the context of data collection from sensors. The system collects information and combines it with domain knowledge, human intuitions, and goals. The recommender model is derived from data collections, model updates, model exploitation, and recommendations from other users or previous actions.

Smart contracts are used in blockchain technology that has the properties of inherent cryptographically secured decentralized architecture, immutability, and user anonymity. However, these smart contracts have vulnerabilities (J. S. Yadav, Yadav, & Sharma, 2022). The study presents twelve publicly available security analysis tools and vulnerabilities present in smart contracts. The recommender system adopts a two-step approach and uses the method of continuous improvement for selecting the suitable tool.

A comprehensive study (Husák & Čermák, 2022) on the automated and semi-automated incident handling and response recommendation systems reveal that there does not exist a full-scale recommender system that guides the user about the appropriate steps. Many of them aim at a particular problem.

To the best of our knowledge, this is the first work that did a general recommendation system framework to select the best suitable pentesting tool for web application vulnerabilities.

2.8 Problem Identification

This literature helped identify a diverse and vast range of vulnerabilities in web applications. At the same time, many open-source and proprietary tools exist for code scanning and penetration testing. Considering the implementation of security in parallel with software development, it becomes challenging for a software developer to select a suitable tool for code scanning during software development. Considering the strict timeline and project plan requirement, an automated recommender system for scanning tools is necessary that assist software developers to find a suitable tool with the best possible accuracy value.

2.9 Summary

This chapter provides background information on vulnerabilities that exist in web applications. Different security testing methods are discussed, including static analysis of source code and penetration testing. Statistics about the top 10 vulnerabilities according to the OWASP project are provided. In the end, the research gap is identified, covering vulnerabilities, existing security tools, and software developer needs in the form of a security testing recommender framework. The next chapter will provide the methodology adopted to develop the framework.

Chapter 3

Research Methodology

This chapter provides the research methodology adopted in this thesis. The major focus is to provide results that are comprehensive and analytical. The research process will define the ways to collect related data and its processing. The research method used in the research to develop the framework for the recommender system is provided. Moreover, the other different research processes involved during the research are explained along with the necessary components.

3.1 Framework Development Methodology

Development and implementation of a cyber security framework can be considered an art where there is no manual for testing the implementation of interconnected systems and the cyber security expert has to rely on his experience. It is a science as well which requires the identification of faults resulting from interconnected software and hardware components. The social aspects cannot be ignored where the actions of individuals are responsible for major security issues. Detail of different approaches and actors is provided in (Haley, Moffett, Laney, & Nuseibeh, 2006; Todorović & Trifunović, 2020; Whitman & Mattord, 2021).

The research methodology is derived from the literature review and experience in the practical field. Cyber Security Recommender System (CSRS) Framework components are reviewed and updated from the literature. An overarching view of the research methodology adopted to develop and evaluate the CSRS Framework is provided in figure 3.1. Different components of the adopted research method are briefly explained as follows:

- **Data Collection and Analysis:** The data is collected from literature including scholarly articles, books, white papers, and websites for standards and guidelines. The focus is on cyber security vulnerabilities, attacks, available tools, cyber security strategies, international security standards, and security implementation frameworks. This phase is covered in Chapter 2.

- **Identify Research Gaps:** The purpose of this phase is to identify the research gap based on evidence from the literature. This phase extracts common features and components necessary in the domain including tools, inputs, and outputs. The process provides high-level components in the framework rather than low-level technical details.

- **Generalized Framework Components:** the components to be used in the framework are generalized, duplicates are removed and inputs are identified. The research processes that will be involved are generalized and validation criteria are established.

- **Propose Recommendation System Framework:** All the components identified in the previous step are conceptually integrated to provide a generalized framework. The design of the framework is based on traditional processing systems that take input, process it, and generate output. The decision about the output is based on feedback.

- **Validate Framework:** one of the most important parts of framework design is its validation (Sabillon, Serra-Ruiz, Cavaller, & Cano, 2017). The researcher will validate it by verifying the output through real test bed implementation in real-life scenarios.

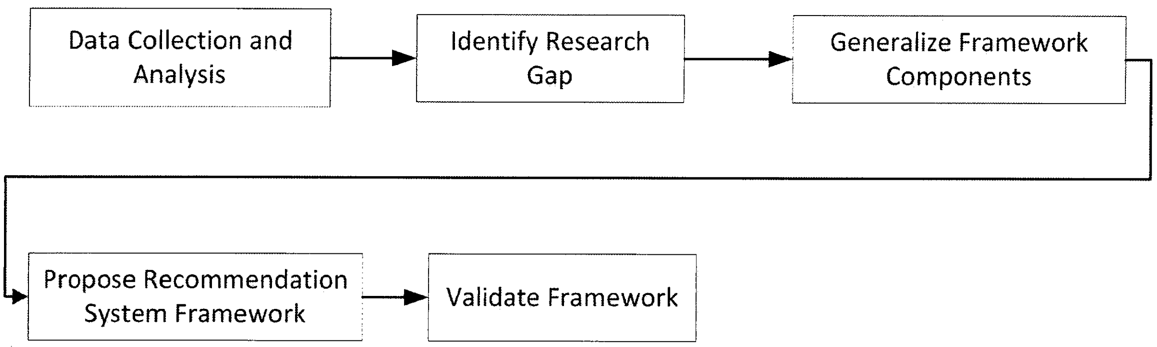

Data Collection and Analysis
Identify Research Gap
Generalize Framework Components
Propose Recommendation System Framework
Validate Framework

Figure 3.1: Research Methodology

3.2 Research Process

The research work is initiated for answering the research question established in chapter 1. A detailed review of different aspects involved in the research is provided in Chapter 1.5. This review includes different vulnerabilities that exist in the source code of web applications, tools to identify these vulnerabilities, and existing cyber security frameworks to address these vulnerabilities. It is identified the process of selecting the proper tool for security testing of the web application is time-consuming for the software developer and it may not provide results with the desired accuracy. Therefore, the design of a Cyber Security Recommender System (CSRS) framework is desired. Different steps that will be followed in the research process are as follows (figure 3.2).

3.2.1 Select Open-Source Tools and Vulnerabilities

There exist a very large number of proprietary and open-source tools for the security testing of a web application. For this work, security testing refers to source code analysis and penetration testing of the deployed web application. Also, the scope of security tools is limited to web applications only. We consider only open-source tools available online. The selection criteria of these tools are based on the number of positive reviews, number of branches, and number of downloads. OWASP's top 10 vulnerabilities are considered for the process. Initially, the list of vulnerabilities mentioned in 2021 is used. The purpose of this phase is to extract different features from tools and vulnerabilities that can be used in the research process.

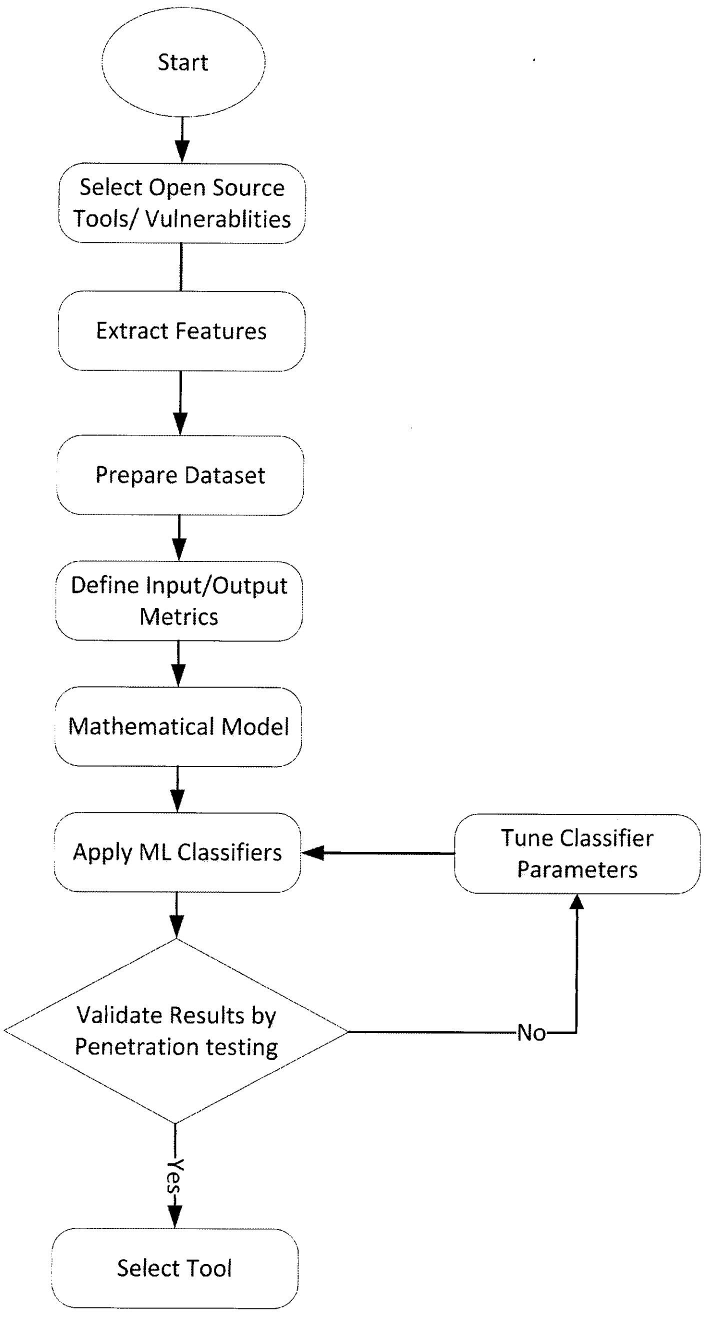

Start
Select Open Source Tools/ Vulnerablities
Extract Features
Prepare Dataset
Define Input/Output Metrics
Mathematical Model
Apply ML Classifiers
Tune Classifier Parameters
Validate Results by Penetration testing
No
Yes
Select Tool

Figure 3.2: Research Process

3.2.2 Prepare Dataset

Different tools and vulnerabilities follow different naming conventions and representations of a feature. The features extracted from the previous phase are generalized in this phase and added to data display formats such as CSV files.

3.2.3 Define Metrics

For any framework, metrics of measurement are important. These metrics are extracted from the properties of inputs to the CSRS framework. These inputs are generally derived from the created dataset. The desired output and units of measurement for the output are also defined. The metrics of measure are necessary for the evaluation and comparison of results.

3.2.4 Mathematical Representation

The proposed Cyber Security Recommendation System (CSRS) framework is based on different features of security assessment tools, ICT infrastructure and the web applications to be tested. These features are integrated using mathematical notations and concepts. The combination of features is used to calculate the accumulative weights of the tool to be selected. The selected security assessment tool is validated using machine learning classification and validated through practical examples (Todorović & Trifunović, 2020).

3.2.5 Apply Machine Learning Classification

The dataset is prepared in section 3.2.2. is classified using different machine learning classifiers such as K-Nearest Neighbor (KNN), Convolutional Neural Network (CNN), Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT) Classifier. These classifiers train a model based on the given dataset and test the performance of the trained model with new data through different metrics of measurement. The metrics of measurement are provided in 3.2.3. A detail of these classifiers is provided in (Kilincer, Ertam, & Sengur, 2021). The machine learning algorithms is explained as the following:

3.2.5.1 Supervised learning

In supervised machine learning algorithms, the prediction models are built on given input data sets and produce output in the form of discrete values (Garcia, Garcia, Villasenor-Pineda, & Mendoza-Montoya, 2021). These models are used for classification tasks for binary labeled data. These machine learning algorithms are known as supervised learning.

- Naive Bayes The Naïve Bayes classifier is based on the Bayes theorem and is mainly used for text categorization. The probability of an attribute x being in class c is calculated through a posterior probability of $p(c/x)$.

- Support Vector Machine (SVM) SMV classifier is used for the supervised machine learning process and is suitable for training and testing the data for classification and regression problems. An imaginary hyper-plane is created in multidimensional space and then data is classified into different classes to minimize errors. The larger number of planes, the better will be the classification.

- Decision Tree It is also a non-parametric, supervised machine learning method for classification and regression analysis. The classification is a two-step process comprising learning and prediction. It works on continuous and discrete variables.

- Random Forest Random forest is a supervised machine learning algorithm and is mainly based on the regression analysis technique. It divides the whole dataset into multiple decision trees where each tree selects a class based on regression analysis. The main steps for the random forest are, to select the random sample from the dataset and construct a decision tree to get a prediction from each tree. Perform a vote for the prediction tree and select the final prediction based on the most vote.

- K-Nearest Neighbor (KNN)

  KNN classification algorithm also applies to supervised machine learning data and is used in cases with data labeled discretely and continuously. KNN creates labels by calculating the similarity between input data and training instances to predict the label. Mainly the KNN algorithm finds the closest neighbor based on distance and the class is selected based on the voting mechanism.

3.2.5.2 Unsupervised Learning

In unsupervised learning, the data is automatically segmented into groups called clusters (Garcia et al., 2021). The main idea behind clustering is to group the elements with similar properties and place the other elements in another group having similar properties. The clusters could be disjoint or overlapping. It could be hierarchical or exhaustive. A member of a cluster joins the cluster based on a probabilistic or deterministic approach. For numeric data, proximity measures in multidimensional space based on Euclidean, Manhattan, or Minkoski are used. For non-numeric variables, measures such as binary, nominal, or non-linear scalars are used.

3.2.5.3 Reinforcement Learning

Reinforcement learning is based on learning from interaction with the environment (Garcia et al., 2021). It tells us the consequences of our actions that could be used to achieve goals. Reinforcement learning can be defined as a technique that guides how to behave in an environment based on interactions. The objective of reinforcement learning is to apply state to action for maximizing performance. For achieving rewards there could be many states. RL follows a trial and error process; there are delayed rewards, and there is a balanced between exploration and exploitation.

3.2.6 Results Validation

Machine learning classifiers provide a list of selected security assessment tools with certain accuracy values. These results require validation which is achieved by applying static analysis or penetration testing using the tools with the highest recommendation. The achieved results are cross-checked with the results of ML classifiers. If the results are comparable i.e. within an acceptable range then the tool is selected else the parameters of the classifiers are tuned and the classification process (Section 3.2.5) is applied again to get better output results.

It is important to note that the selected ML classifiers are trained with a limited number of records consisting of properties of security assessment tools, web applications, and infrastructure-related properties. In every iteration, the results are refined by changing the combination of input parameters for refining the output values of trained models. In this way, results with the best possible values are tried to achieve.

3.3 Research Techniques

3.3.1 Data Collection

Scholarly databases are the main source of research data available in the form of articles. The objective of data collection is as follows:

- **Literature review:** It consists of different scholarly articles available in different digital libraries (IEEE Xplor, ACM, Springer, Science Direct, and many others). Along with this, data related to tools, vulnerabilities, attacks, and defense techniques will be found in general web search engines.

- **Implementation:** The tools related to security testing and open-source web applications will be downloaded from the Internet

3.3.2 Analysis Techniques

The analysis will be done by collecting data related to open-source web applications and security assessment tools. The parameters will be extracted from these web applications and the tools, and then used to train the machine learning model. The testing web application properties will be fed into the classifier and the recommendation of the tool by the classifier is validated through either source code analysis or penetration testing.

3.3.3 Validation

The researchers will generate the results at different stages of the research process to validate the system (figure 3.2). The processed data will be analyzed through the numerical method and statistical analysis. Bugs and vulnerabilities reported in web applications by different security tools will be cross-checked for validity and then made part of the training dataset. The output of the classifiers is reproduced using different parameter values. The objective is to reproduce the results which can be validated. These results will provide a clear and valid answer to the research question about the selection of security tools for software applications.

3.4 Ethical Consideration

This study does not collect data through quantitative or qualitative methods such as surveys, interviews, or focus group studies. The data is collected from scholarly databases and websites. The issues related to the data such as plagiarism and data confidentiality are identified earlier to prevent the problems that could occur later in the research process. However, it is important to clarify that the research is to develop the tool recommendation system. A prototype implementation of the proposed framework will be provided and a commercial or proprietary implementation is beyond the scope of the research. Therefore, there do not exist any legal constraints on the prototype implementation. Moreover, only open-source tools will be considered for testing and evaluating the proposed security testing framework.

3.5 Summary

This chapter provides a methodology for the development of a cyber security framework for a recommendation system. All the phases of methodology are explained. The research process is illustrated to provide an in-depth picture of the flow of the process. Moreover, data collection, data analysis, and validation techniques are provided.

Chapter 4

Cyber Security Recommendation System (CSRS) Framework

4.1 Introduction

The chapter introduces the Cyber Security Recommendation System (CSRS) framework. Components of the framework will be identified and integrated in a way to provide collective output. Before this, a mathematical representation of these components will be given to achieve an analytical output. The model will provide the upper and lower bounds of the input parameters and results. Data will be finalized for websites and security assessment tools. This parameter data will be generalized to be used in programming tools. The data will be analyzed using machine learning classifiers using Python programming language. In the end, the results will be tested for source code assessment or penetration testing using a suitable environment.

4.2 Feature Selection for CSRS Framework

The CSRS framework is based on the input of multiple features for the selection of appropriate cyber security assessment tools. Initially, the framework is based on features of the cyber assessment tool, web applications, and ICT infrastructure.

4.2.1 Cyber Assessment Tools Features

There is a long list of features that exist for security assessment tools that could be a reason for the selection of a particular tool (Vu, Tippenhauer, Chen, Nicol, & Kalbarczyk, 2014). These features include functionality, extensibility, sustainability, and portability (Roldán-Molina, Almache-Cueva, Silva-Rabadão, Yevseyeva, & Basto-Fernandes, 2017). However, these are generic features of a tool. In this study, the metrics related to a tool such as cyber security metrics (confidentiality, integrity impact, etc.), standards (CVE, CVSS, etc.), number of downloads, number of development branches, rating, and number of academic citations. The features are described briefly as follows:

- **Operating Modes (OM):** The modes of assessment that are supported by the tool. For example, some tools only provide installers that install on the supported operating system. Some tools provide deployable web applications and are accessible via web interface and some tools provide command line support. The rating is, installer=1, deploy-able web app = 2, command-line = 3.

- **Test Type (TT):** it refers to the type of testing provided by the tool. It could be a black box, a white box, or both. The rating of this parameter is black-box=1, white-box=2, and both =3.

- **Data Confidentiality (DC):** it refers to the ability of the tool to check the data confidentiality. The value is zero if DC is not checked else it is 1.

- **Integrity Impact (II):** II can determine the impact if the integrity of data is compromised and the vulnerabilities that can be exploited to compromise data integrity. The value is zero if II is not checked and 1 if the tool provides a detailed impact of potential vulnerability.

- **Standards (Std):** The support of tool for Common Vulnerability Scoring System (CVSS) or CVS scores. The CVSS score is a computation of base metrics that reflect how much risk a vulnerability poses to network security. If one standard is followed, its rating is 1, and so on.

- **Number of Downloads(Down):** For open-source tools, the number of downloads is an indicator of the popularity of tools.

- **Number of branches(Branch):** it refers to the number of development groups working on the source code.

- **Rating (Rate):** The rating of the tool given by different users is based on different features provided by the tools. The rating is available on the online source code repository of the application.

- **Academic Citations (AC):** The number of articles that cited the tool in academic research. These citations can be collected from Google Scholar for any given academic resource.

4.2.2 Web Application Features

Web application features play important role in the selection of security analysis tools. The selection of a cyber assessment tool depends on the feature of the web application and a few of these features are given as follows:

- **Programming Language (PL):** The programming language of the web application in which the application is developed. It is one of the constraints in the selection of security assessment tools. This is because if the cyber assessment tool does not support the programming language of the under-test application, the decision based on other features cannot be correct.

- **Web Server (WS):** The web server on which the application is hosted.

- **Host Operating System (OS):** The operating system on which the web server is hosted. The tools provided within OS help to make the hosted application more securable. OS runs different authorization and authentication tasks that provide support to prevent unauthorized access (D. Yadav, Gupta, Singh, Kumar, & Sharma, 2018).

Another list of features reported in OWASP Application Security Verification Standard (ASVS) project (ASVS, 2016) is given as follows:

1. **Architecture (Arc):** There are three levels of architecture. In level 1, components of the app are defined in the code but not implemented; in level 2, architecture is defined and adheres to the code and in level 3, all design and code are integrated.

2. **Authentication (Auth):** The credentials of the user are secure during communication. There is a further checklist of 33 points for placing the rating in V1, V2, and V3.

3. **Session Management (SM):** The set of all controls governing statefull interaction between a user and the web-based application.

4. **Access Control (AC):** The mechanism to permit access to only authorized users.

5. **Malicious Input Handling (MIH):** Validate input coming from the client or the environment before using it

6. **Cryptography at rest (CRYPT):** It deals with the handling of cryptographic errors, suitable number generators, and secure access keys.

7. **Error Handling/Logging (EHL):** Proper error handling and logging are provided in the application for the support of the administrator or users of the application.

8. **Data Protection (DP):** Data protection is centered around three components that are Confidentiality, Integrity, and Availability (CIA). In the web application context, data protection is ensured by hardening the server on which the web application is deployed.

9. **Communications (COMM):** It refers to applying Transport Layer Security (TLS) when data is exchanged on the communication channel. One common approach to achieving communication security is by applying strong ciphers.

10. **HTTP Security Configuration (HSC):** It refers to a customized configuration of the application server instead of the default configuration. Hardening of the application server is also required.

11. **Malicious Controls (MC):** It refers to the handling of malicious controls in the code. These include time bombs, phone home to unauthorized destinations, back doors, and logic flaws.

12. **Business Logic (BL):** It means business logic flows are sequential and in order. The conditions in the code such as transferring funds continuously or adding millions of friends are checked and verified. Check business logic to prevent spoofing, tampering, repudiation, information disclosure, and elevation of privilege attacks.

13. **Files and Resources (FR):** It refers to the handling of untrusted data securely, saving it with limited permission and not in the web root directory.

14. **Mobile (Mob):** All the security controls are implemented on the mobile client.

15. **Web services (WS):** The authorized web applications have session management and authorization of all web services. It also requires input validation and parameter verification for consuming web services.

16. **Configuration (Conf):** The application has up-to-date libraries, operating platform(s) and secure configuration.

4.2.3 ICT Infrastructure Features

Generally, the web applications are hosted in data centers equipped with ICT equipment to ensure cyber security (Hunter & Weiss, 2021). This equipment includes but is not limited to firewalls, encrypters, an Intrusion Detection System (IDS), Intrusion Prevention System (IPS), and domain controllers for authentication and authorization of incoming requests. A brief description of basic hardware components installed/configured in a data center is given as follows:

- **Firewalls:** These are devices that pass network traffic following the rules defined in the device. These devices serve as filters to stop undesired communication (Sabur, 2018).

- **Encrypters:** These are hardware devices that encryptdecrypt all the incoming or outgoing traffic from a data center. The cryptography algorithms implemented in these device work based on publicprivate key or symmetric keys mechanism (Zhang, Yu, Ramani, Afanasyev, & Zhang, 2018).

- **Intrusion Detection System (IDS):** IDS systems have the general purpose of detecting unauthorized access to the system. These devices can be used with many features like signature-based IDS, Anomaly-based IDS, Host-Based IDS, Network-based IDS, Stack Based IDS, and so on. IDS and IPS are useful against cross-site scripting (XSS) attacks (Chen, Nshimiyimana, Damarjati, & Chang, 2021).

- **Domain Controllers:** A domain controller is a server that responds to authentication requests and verifies users on computer networks. For web applications that authenticate via domain controller, rely on tested authentication mechanism (Al-Anezi, 2019).

4.3 Mathematical Representation

The mathematical representation provides a high-level representation of different variables and the way they are integrated into a system. The mathematical representation of a cyber security recommendation system describes different input features and the way that are combined to generate the desired output. A detailed description of the mathematical representation is given as follows:

4.3.1 Parameter Generalization

Parameter generalization is the process to describe the existing features in a generic form. This process is important for developing a generic mathematical description of the proposed solution. The three feature sets given in section 4.2.1, 4.2.2, and 4.2.3 as F_T, F_A and F_I are defined in this section. In each of these feature sets, there exist two types of features, mandatory features, and optional features.

4.3.2 Assumptions

The mandatory features are the constraints on the tool selection process. A description of the constraints used in the model is as follows:

- **Programming Language Support (*L*):** For a tool to be selected, it must support the programming language of the web application that has to be tested. Mathematically, it can be described as $L \in F_T$ such that $L \neq 0$. The value of $L = 1$ if it supports one programming language and so on.

- **Operating System (*O*):** It is a support tool for working on a particular operating system. Mathematically, it can be described as $O \in F_T$ such that $O \neq 0$.

  For the tool to be considered, the above two constraints should be satisfied.

There is a list of optional variables provided in section 4.2.1, 4.2.2, and 4.2.3. For quantification, each optional feature is assigned a weight that will assist in the total weight computation of the feature set. The weights of different features are summarized in Table 4.2.

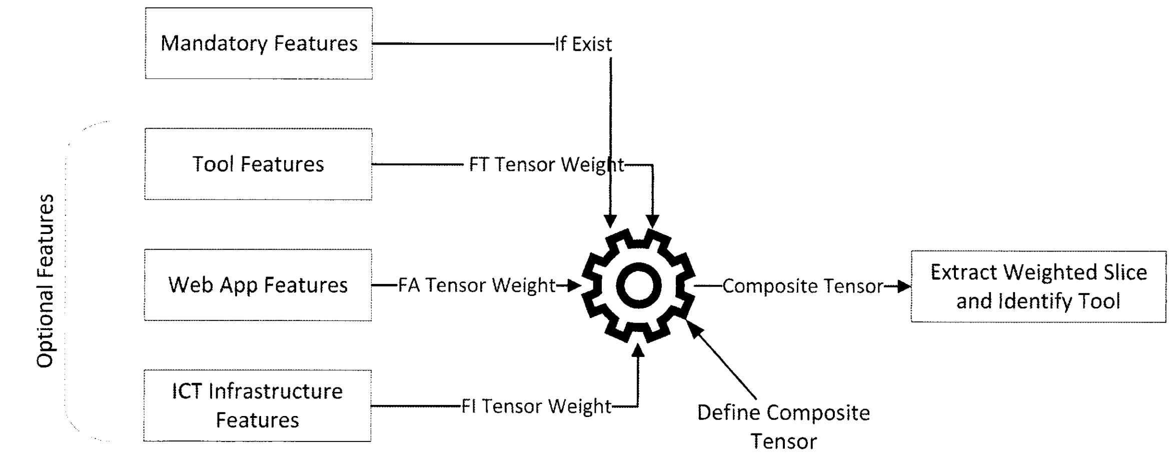


Figure 4.1: Framework Design

Table 4.1: Weight of Features in Different Feature Sets

Feature	Weight Description
Operating Mode	Number of supported operating modes. The minimum value is 1.
Data Confidentiality	The support of data confidentiality features in the tool. Its value is zero if no support is provided else it is 1.
Integrity Impact	Weight is 1 if the impact is only on OS, 2 if OS and databases, and so on.
Standards	Number of supported standards for vulnerability Assessment.
Web Server	If the application is to be deployed, the number of web servers that support the deployment of web applications. The minimum value is 1.
Hosting OS	The operating system which supports the deployment of web applications. The minimum value is 1.
Architecture	$Level1 = 1$, $Level2 = 2$ and $Level3 = 3$.
Authentication	Implemented = 1, Not implemented = 0
Session Management	Implemented = 1, Not implemented = 0
Access Control	Implemented = 1, Not implemented = 0
Cryptography	Implemented = 1, Not implemented = 0
Comm. Security	Implemented = 1, Not implemented = 0
Malicious Controls	1, 2, 3, ...
File and Resources	Implemented = 1, Not implemented = 0
Mobile	Implemented = 1, Not implemented = 0
Web services	Implemented = 1, Not implemented = 0
Configuration	Implemented = 1, Not implemented = 0
Firewall	Exist =1, Absent = 0
Encryptor	Exist =1, Absent = 0
Domain Controller	Exist =1, Absent = 0
IDS/IPS	Exist =1, Absent = 0

4.3.3 Framework with Mathematical Description

The CSRS framework integrates multiple related features in a single model (figure 4.1). Before explaining the framework, the definitions of different concepts are as follows:

- **Tensor:** A tensor can be defined as a a multidimensional array (Kolda & Bader, 2009). An Nth order tensor represents N dimensions. In this CSRS framework, each dimension represents a feature set. A tensor with one dimension is called a vector, with 2 dimensions is called a matrix, and with 3 or more dimensions, it is called a composite or higher-order tensor.

- **Order of Tensor:** The number of dimensions, also called modes, is the order of the tensor. For a single dimension (vector), it is represented by a lowercase boldface letter i.e. **a**. Two-dimensional tensor (matrix) is represented by uppercase boldface letter i.e. **A** and higher order tensors are represented by boldface Euler letter i.e. $\mathscr{A}$

- **Tensor Index:** The value of the feature inside a tensor is represented by the index i.e.a_i represents ith feature in a tensor. The feature in two-dimensional tensor is represented by A_{ij} and higher order indices can be represented as $\mathscr{A}_{\mathbf{ijk}}$

- **Slice:** Slices are two-dimensional sections of tensor that fix one index and the other two are variable. A horizontal tensor is shown in figure 4.2, denoted by $\mathscr{A}_{\mathbf{i}::}$. More compactly, the ith slice of a sensor can be represented by $\mathscr{A}_{\mathbf{i}}$.

Each feature set is defined as a tensor with a weighted sum of the weights of their features. The tensors (feature set) can be calculated as follows:

$$F_T = \{i_1, i_2, i_3....i_l\}; \quad l \subseteq n \quad and \quad min \leq i_l \leq max \tag{4.1}$$

$$F_A = \{j_1, j_2, i_3....j_l\}; \quad l \subseteq n \quad and \quad min \leq j_l \leq max \tag{4.2}$$

$$F_I = \{k_1, k_2, k_3....k_l\}; \quad l \subseteq n \quad and \quad min \leq i_l \leq max \tag{4.3}$$

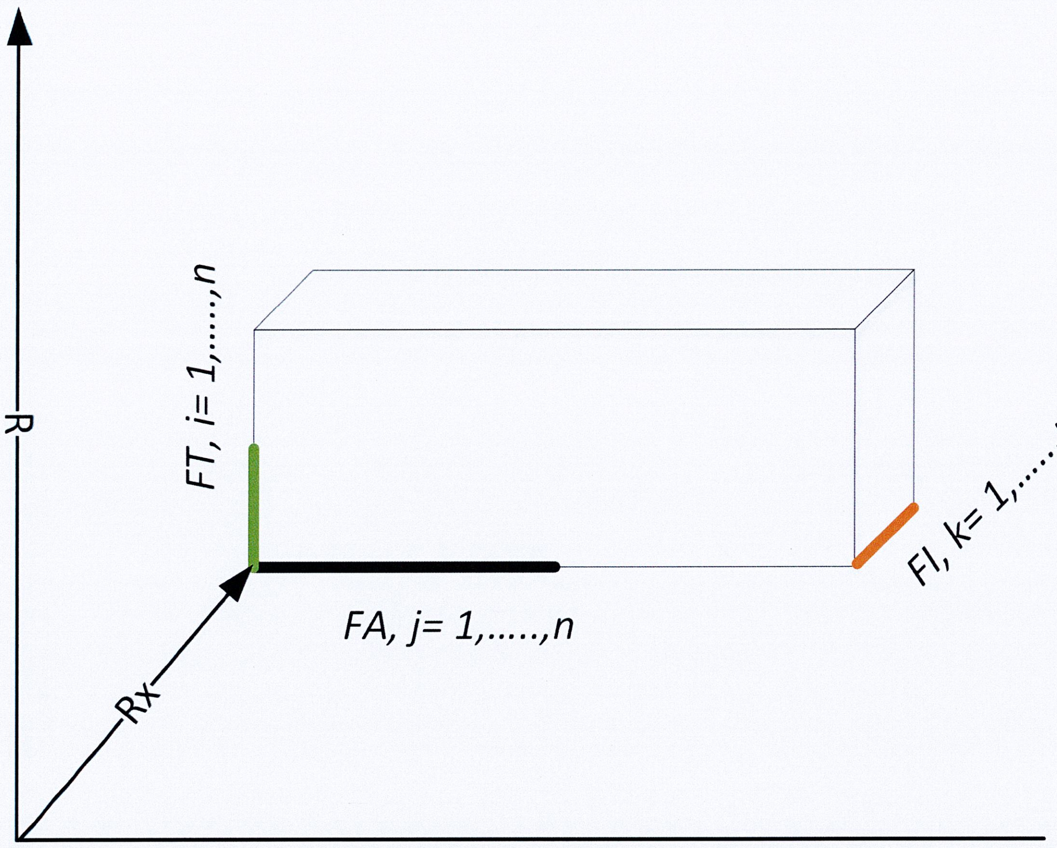


Figure 4.2: Composite Tensor, R_x is the resultant value

The resultant value of composite tensor, R_x can be represented as $R_{x:::}$ where the value from equations 4.1, 4.2 and 4.3 are calculated and fixed. The resultant value R_x lies between $R_{x_{min}}$ and $R_{x_{max}}$. The lower ($R_{x_{min}}$) and upper ($R_{x_{max}}$) limit of the accumulative weight of features is set earlier based on the defined tool. The graphical representation is provided in figure 4.2 and mathematically given in equation 4.4.

$$R_x = R_{x:::} \quad \text{where} \quad R_{x_{min}} < R_x < R_{x_{max}} \tag{4.4}$$

Where R_x is the calculated weight based on the weights of feature sets that are represented as slice ($R_{x:::}$) in the 4-dimensional space.

4.4 Implementation

The framework is implemented using open-source web applications. The selection of these web applications is based on features provided in section 4.2.2. These applications are as follows:

- WebGoat is an insecure web application provided by OWASP for testing server-side application vulnerabilities (OWASP-WEBGOAT, 2022).
- Mutillidae II is an open-source web application provided by OWASP. It can be hosted on Linux and Windows platforms and can be de-ployed on LAMP, WAMP, and XAMP web servers (OWASP-MULTIDAE, 2022).
- OWASP Juice Shop is one of the most used deliberately insecure applications developed by the OWASP community. It covers OWASP top ten vulnerabilities along with many other security flaws found in real-world applications (OWASP-JUICESHOP, 2022).
- Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is vulnerable and provides various levels of difficulty with a relatively simple user interface (OWASP-DVWA, 2022).
- Codelab is a small web application built around Gruyere and contains bugs such as cross-site scripting and cross-site request forgery, information disclosure, denial of service, and remote code execution (OWASP-CODELAB, 2022).
- CryptoPaste is a secure service developed to perform all encryption, decryption, and data handling in the user's browser (OWASP-CRYPTOPASTE, 2022).

- WackoPicko is a website that contains known vulnerabilities (OWASP-WACKOPICKO, 2022).

4.4.1 Data Preparation

The selection of cyber assessment tools for dataset preparation is based on operating modes, test type, data confidentiality, integrity impact, number of branches, rating, and academic citations (section 4.2.1). The weight of selected features of OWASP's top 10 tools is provided in Table 4.3. For ICT features, it is assumed that none of the infrastructures exist. The values are given in Table 4.5.

Data from different tools, sample web applications, and infrastructure is collected. The classification and labeling of data are performed based on the CSRS framework. For each category, the dataset with similar properties is assigned a similar class. An example of classification is given in Table 4.6. The labels of different classes from each category are combined according to the CSRS framework. A sample of a combination of the label of the aggregated class is given in Table 4.7.

A basic combination of the features of selected web applications, tools, and infrastructure is labeled. This labeled data is used to train and test the machine learning classifiers. The performance of classifiers is further improved by adding two more applications and increasing the number of records for training the classifiers. Overall, the results are generated using a combination of 54 features. However, a subset of features is also tested for demonstrating the performance gains achieved by a larger number of features.

Table 4.2: Web Application Features and Weights

Features	**WebGoat**	**Mutillidae II**	**Juice Shop**	**DVWA**	**Codelab**	**CryptoPaste**	**WackoPicko**
PL	JS	Ajax, JS, PHP	NodeJS	PHP, JS	HTML, Python	PHP	PHP
WS	Apache	LAMP, WAMP, XAMP	NodeJS	LAMP, WAMP, XAMP	IIS	LAMP, WAMP, XAMP	LAMP, WAMP, XAMP
OS	Linux	L,W,M	L,W	L,W,M	W	L,W	L,W
Arc	2	1	1	3	1	2	2
Auth	0	0	0	0	0	1	1
SM	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0
MIH	0	0	0	0	0	0	0
Cryp	0	0	0	0	0	1	1
EHL	0	0	0	0	0	0	0
DP	0	0	0	0	0	1	1
Comm	0	0	0	0	0	1	1
MC	0	0	0	0	0	0	0
HSC	0	0	0	0	0	0	0
BL	0	0	0	0	0	0	0
FR	0	0	0	0	0	0	0
Mob	0	0	0	0	0	0	0
WSvc	0	0	0	0	0	0	0
Conf	0	0	0	0	0	1	1

Table 4.3: Tool Features and Weights

Tools	**OM**	**TT**	**DC**	**II**	**Branch**	**Rate**	**AC**	**Attacks/Vul.**
ZAP Proxy (OWASP-ZAP, 2022)	1	3	1	1	1900	9600	384	AUTH,INT
XRCross (OWASP-XRCross, 2022)	1	3	1	0	56	237	2	XSS,SSRF,CORS,SSTI, IDOR, RCE,LFI,SQLI
NMap (OWASP-NMAP, 2022)	1	2	1	1	1900	6500	25400	MON
Crypto Detector (OWASP-CRYPTO, 2022)	1	2	1	1	23	111	16	CRYP
sqlmap (OWASP-SQLMAP, 2022)	1	1	1	1	4900	2390	2060	SQLI
SonarQube (OWASP-SONARQUBE, 2022)	2	2	1	1	1700	6900	3440	AUTH,INT,CRYP
W3af (OWASP-W3AF, 2022)	2	2	0	0	1200	4000	998	SQLI,XSS,OSC
Dependency check (OWASP-DependencyCheck, 2022)	3	2	0	1	959	4200	1350	OSD
thc-hydra (OWASP-THCHYDRA, 2022)	1	3	1	1	1500	6300	502	AUTH,CRYP
Commix (OWASP-COMMIX,2022)	1	1	1	1	704	3300	798	OSC

Table 4.4: Attack and Definitions

Attack	Definitions
AUTH	Check for the process of authentication e.g. password
INT	Check the data integrity e.g. hashing
XSS	Cross-Site Scripting
SSRF	Server-side request forgery
CORS	Cross-Origin Resource Sharing
SSTI	Server Side Template Injection
IDOR	Insecure direct object references
RCE	Remote Code Execution
LFI	Local File Inclusion
SQLI	SQL Injection
MON	Monitoring Network Traffic
CRYP	Presence of encryption techniques for data protection
OSC.	Operating System Commanding i.e. gain access to OS
OSD.	Operating System Dependencies i.e. updates or patches

Table 4.5: ICT Features and Weights

Feature	Type	Weight
Firewall (FW)	H/W	0
Domain Controller (DC)	S/W	0
Intrusion Detection& Prevention System (IDS)	H/W	0
Encryptor (ENC)	H/W	0

Table 4.6: Sample Labeling of Web Applications

APP	JS	PHP	XAMP	WAMP	LINUX	WIN	ARC	Label
WEBGOAT	1	0	1	0	0	1	2	B
MULTIDAE II	1	1	1	1	1	1	1	A
JUICE SHOP	1	0	1	0	0	1	2	B
DVWA	1	1	1	1	1	1	1	A
CODELAB	1	0	0	1	1	0	1	C

Table 4.7: Final Labeling of Each Class

Class	WebApp	Tool	Infra	Class Label
1	B	A	A	BAA
2	A	A	A	AAA
3	B	A	B	BAB
1	B	A	A	BAA
5	A	B	A	ABA

4.4.2 Tool Selection for Implementation

Python (version 3.8) programming language is used to evaluate the results generated from the data based on the given model. Python is selected as a tool for different reasons (Sarkar, Bali, & Sharma, 2018; Raschka, Patterson, & Nolet, 2020). One of the reasons is the availability of libraries for handling large amounts of data collected from different sources. Another reason is the ease of use which makes it the preferred choice of researchers. The libraries in Python are developed using lower-level static-type languages such as Fortran, C/C++, and CUDA that make it performance efficient for scientific computing and machine learning. Scikit-learn library (SciKit, 2022) is used for applying different machine learning models and calculating metrics values. It is an open-source library with the provision of simple and efficient tools for predictive data analysis.

4.4.3 Metrics Selection

Three different metrics are used to determine the performance of classifiers (Grandini, Bagli, & Visani, 2020). These metrics are as follows:

- **Accuracy:** It is defined as the number of correct predictions as compared with a total number of predictions.

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions} \tag{4.5}$$

- **Precision:** Precision is a measure of the ability of a classification model to identify only the relevant data points.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.6}$$

- **Recall:** Recall is a measure of the ability of a model to find all the relevant cases within a dataset.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.7}$$

- **F1-Score:** F1 score is the harmonic mean of precision and recall. Just as a caution, it's not the arithmetic mean. If precision is 0 and recall is 1, the f1 score will be 0, not 0.5.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.8}$$

4.5 Results

The data combined for feature values of web applications, tools, and infrastructure. Initially, basic values of tensors are considered resulting in 140 records. Results generated from 90/10 train/test split and 54 features for training data are provided in Table 4.8. Moreover, the highest accuracy classifier is the RF which shows 97% prediction accuracy. Also, It is found that DT showed 93% prediction accuracy, while SVM 89% and NB achieved 86% prediction accuracy, These results are reflected in the validation process of the model and explained in section 4.6.

Table 4.8: Initial Results with 54 Features

Classifier	Accuracy	Precision	Recall	F1-Score
Random Forest (RF)	97	97	97	97
Decision Tree (DT)	93	93	93	93
Naive Bayes (NB)	86	87	86	86
SVM(One-vs-Rest)	89	87	89	88
K-Nearest Neighbor (KNN)	60	57	60	52

4.5.1 Feature Optimization

Feature selection is the process of selecting of best available feature for predicting the output value. The presence of irrelevant features in the dataset can decrease the performance of the classifiers (Cai, Luo, Wang, & Yang, 2018). The benefits of performing the feature selection process before training the models are as follows:

- **Reduces Over-fitting:** Removing redundant data removes the noise from the data and improves classifier performance.
- **Improves Accuracy:** Removing misleading data improves the modeling accuracy.
- **Reduces Training Time:** The training data volume is usually very large. Reducing the number of features also decreases modeling time.

Statistical tests are used to find the most relevant features in the dataset. The univariate algorithm is one of the algorithms for selecting the best features. The Univariate algorithm is best suited for integer datasets and therefore, applicable to the dataset in the research. The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features. The other feature selection algorithms include Principle Component Analysis (PCA) (Karamizadeh, Abdullah, Manaf, Zamani, & Hooman, 2020) and Recursive Feature Elimination (RFE) (Ustebay, Turgut, & Aydin, 2018).

Applying the Univariate on the complete dataset resulted in figure 4.3. The number of features was reduced from 53 to 46 where the results show identical accuracy values with a lesser number of features as shown in Table 4.8.

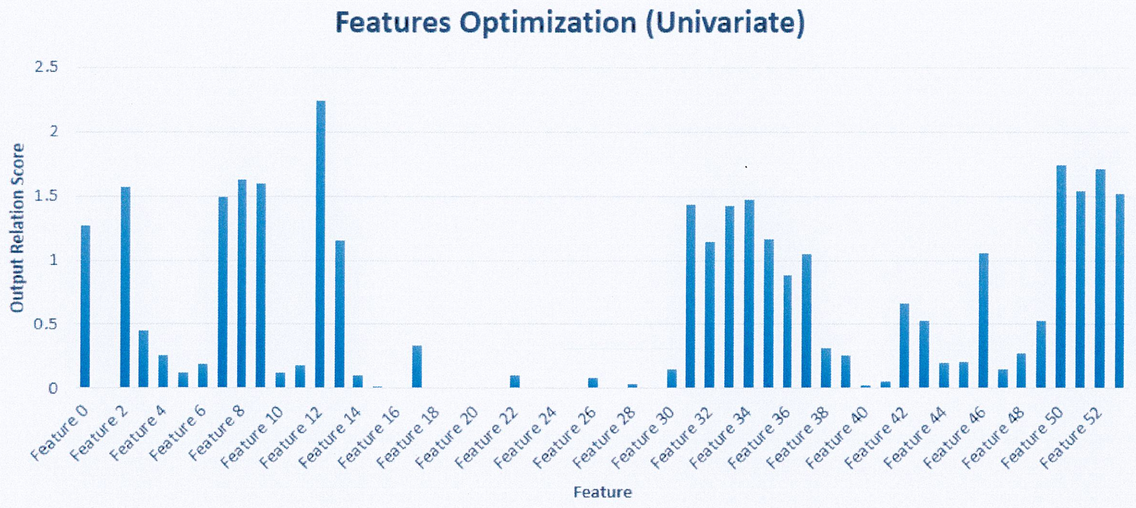


Figure 4.3: Feature Optimization

4.6 Tool Prediction and Data Analysis

For the prediction of the suitable tool for the new web applications, three different open-source web applications are selected. The features of these web applications are fed into the trained machine-learning models. The prediction accuracy of tools for all these applications (“Domain Mod”, 2022; “Online-Invoicing-System”, 2022; *OSPOS*, 2022) is provided in Table 4.9.

The results show a comparison of the actual versus predicted class using different machine learning classifiers. The features of the test applications are extracted similarly to the features of the training application data set. All the input features of test web applications are provided to the trained model

Table 4.9: Tools Prediction (Actual class/Predicted class)

Classifiers	Invoicing system	OSPOS	Domain Mod	Accuracy (%)
NB	6/6	18/18	4/4	100
RF	6/6	18/18	4/4	100
DT	6/6	18/18	4/4	100
SVM	6/6	18/18	4/24	67
KNN	6/4	18/16	4/4	33

and the most relevant class of these applications is predicted. Naive Bayes, Random Forest, and Decision Tree classifiers predicted the classes of input data with 100% accuracy while SVM with one-versus-rest predicted 2 out of 3 classes correctly. KNN could only predict 1 out of 3 classes correctly. The reason for the high scores of the three classifiers (RF, DT, and NB) is that the classifiers are designed to perform multi-class classification instead of binary-class classification. On the other hand, an SVM works by projecting the data into a higher dimensional space and separating it into different classes by using a single (or set of) hyper-planes. A single SVM does binary classification and can differentiate between two classes, this is a limitation in SVM classifiers and hence failed to detect any class correctly. Therefore, the SVM variant of one-versus-rest is used for training and testing the model. KNN classifier also could predict one class correctly while failing for the other two. KNN has the limitation of handling a small set of data records with imbalanced class distribution (Hasib et al., 2020).

4.7 Results Validation

The data from three different open-source web applications are used for tool prediction in section 4.6. The tools defined for classes 4, 6, and 18 are predicted by the classifiers. These tools for class 4 and 6 is XRCross (OWASP-

XRCross, 2022) and for class 18 Zap Proxy (OWASP-ZAP, 2022).

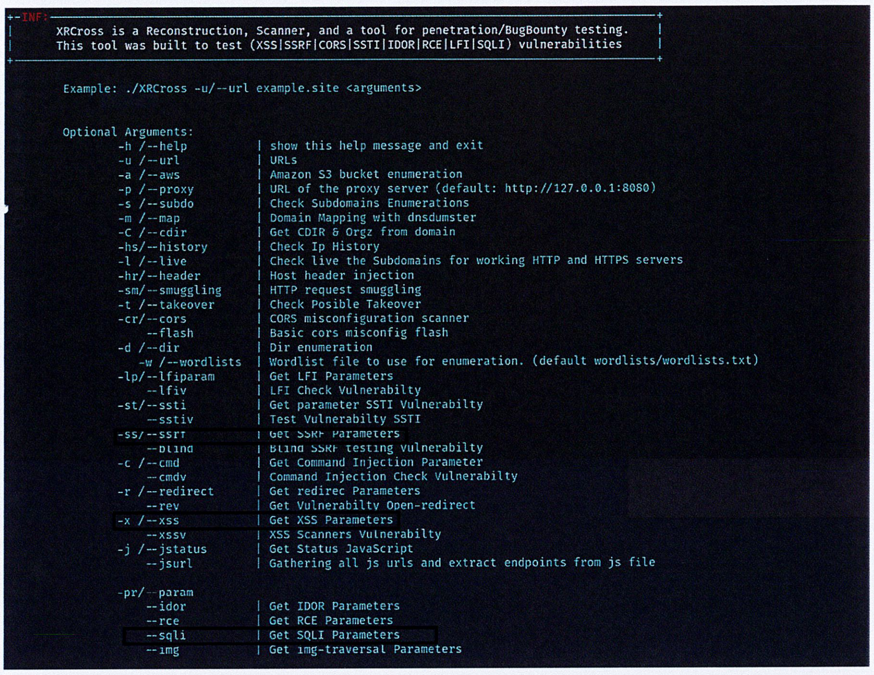
A complete environment for tool prediction is configured. This environment consists of a virtual machine hosted on the VMWare platform. Kali Linux (*Kali-Linux*, 2022) is hosted as the operating system on the virtual machine (VM) containing many pre-built tools for the security assessment of web applications. The VM consists of 4GB of memory and 60 GB of disk space. The quad-core processor is installed for better performance.

The network architecture was equivalent in all tests and completely unchanged during the scanning process. Firewall configurations were set to permit all network traffic. In addition, the testing tools were configured to scan for services on all TCP ports and allowed vulnerability signatures.

The validators desired to check the first two applications (Online invoicing system and Domain mod) for Cross-Site Scripting (XSS), Server Side Request Forgery (SSRF), and SQL Injection attack. Corresponding features of the web applications, desired attacks to test, and infrastructure mode are passed to the tool predictor. The models predicted XRCross with 100% accuracy with NB, RF, and DT algorithms. Both of the test applications are tested on the Kali platform using XRCross and the results are shown in figure 4.4.

The third testing application is OpenSourcePos (OSPOS) (*OSPOS*, 2022). The desired tool should have authentication and integrity testing support. The CSRF recommended ZAP Proxy (OWASP-ZAP, 2022) tool based on inputs from the validators. The report generated by the ZAP proxy tool is large enough. Therefore, a part of the report is shown in figure 4.5. The results of scanning OSPOS using ZAP show that different types of attacks that cover authentication and integrity check are reported with their number and the risk level.

For comparing the performance of the recommender system, some cyber assessment tools are randomly selected by the researcher. The selection



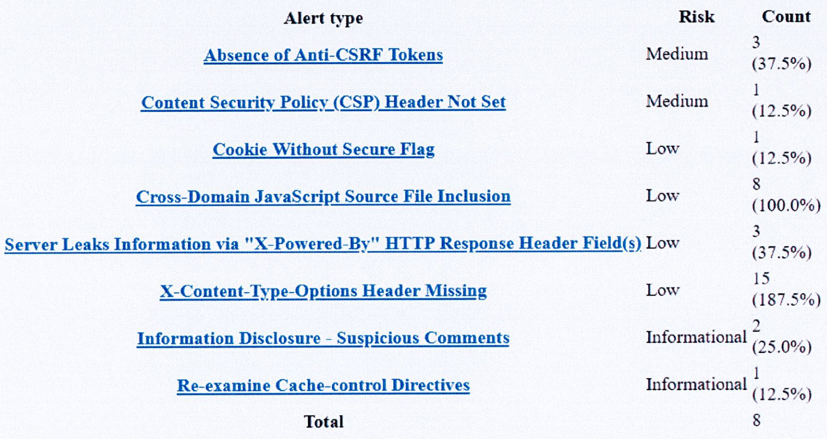
```
+-INF:-----------------------------------------------------------------------------+
|      XRCross is a Reconstruction, Scanner, and a tool for penetration/BugBounty testing.   |
|      This tool was built to test (XSS|SSRF|CORS|SSTI|IDOR|RCE|LFI|SQLI) vulnerabilities    |
+----------------------------------------------------------------------------------+

        Example: ./XRCross -u/--url example.site <arguments>

        Optional Arguments:
                -h /--help          | show this help message and exit
                -u /--url           | URLs
                -a /--aws           | Amazon S3 bucket enumeration
                -p /--proxy         | URL of the proxy server (default: http://127.0.0.1:8080)
                -s /--subdo         | Check Subdomains Enumerations
                -m /--map           | Domain Mapping with dnsdumster
                -C /--cdir          | Get CDIR & Orgz from domain
                -hs/--history       | Check Ip History
                -l /--live          | Check live the Subdomains for working HTTP and HTTPS servers
                -hr/--header        | Host header injection
                -sm/--smuggling     | HTTP request smuggling
                -t /--takeover      | Check Posible Takeover
                -cr/--cors          | CORS misconfiguration scanner
                    --flash         | Basic cors misconfig flash
                -d /--dir           | Dir enumeration
                   -w /--wordlists  | Wordlist file to use for enumeration. (default wordlists/wordlists.txt)
                -lp/--lfiparam      | Get LFI Parameters
                    --lfiv          | LFI Check Vulnerabilty
                -st/--ssti          | Get parameter SSTI Vulnerabilty
                    --sstiv         | Test Vulnerabilty SSTI
                -ss/--ssrf          | Get SSRF Parameters
                    --blind         | Blind SSRF testing vulnerabilty
                -c /--cmd           | Get Command Injection Parameter
                    --cmdv          | Command Injection Check Vulnerabilty
                -r /--redirect      | Get redirec Parameters
                    --rev           | Get Vulnerabiltv Open-redirect
                -x /--xss           | Get XSS Parameters
                    --xssv          | XSS Scanners Vulnerabilty
                -j /--jstatus       | Get Status JavaScript
                    --jsurl         | Gathering all js urls and extract endpoints from js file

                -pr/--param
                    --idor          | Get IDOR Parameters
                    --rce           | Get RCE Parameters
                    --sqli          | Get SQLI Parameters
                    --img           | Get img-traversal Parameters
```

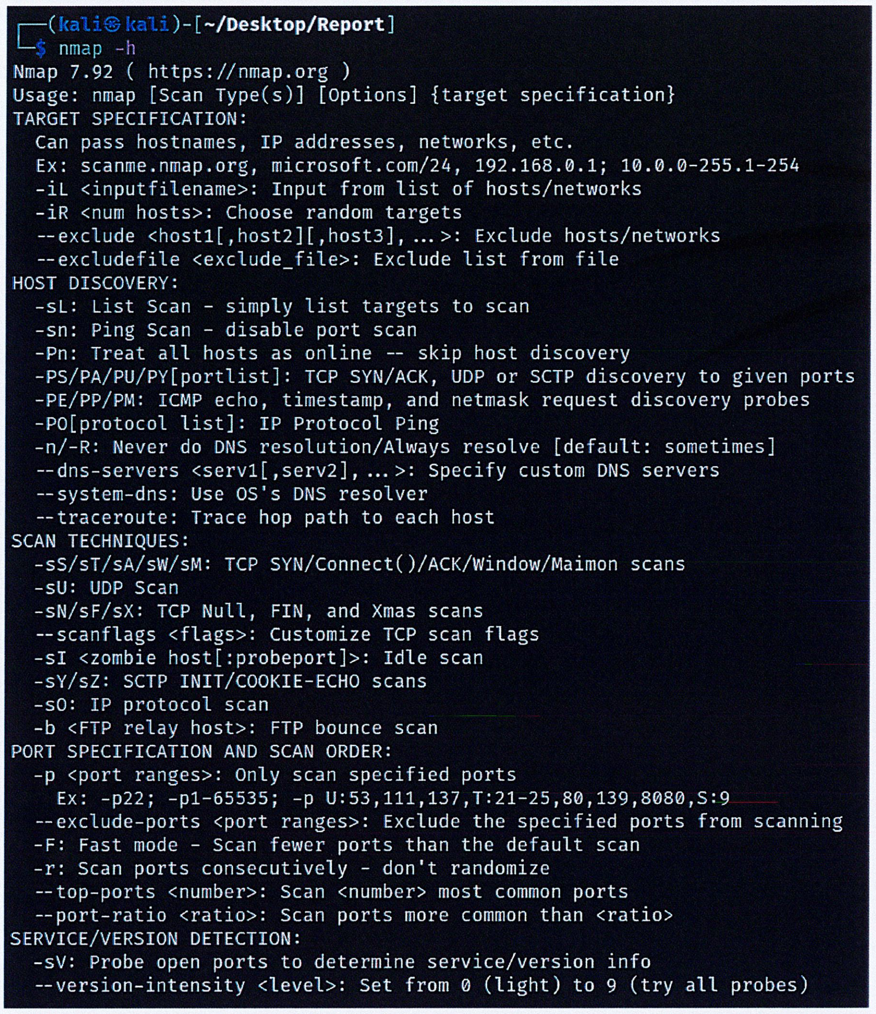
Figure 4.4: XRCross Providing Support for XSS, SSRF, and SQL Injection Attacks



Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	3 (37.5%)
Content Security Policy (CSP) Header Not Set	Medium	1 (12.5%)
Cookie Without Secure Flag	Low	1 (12.5%)
Cross-Domain JavaScript Source File Inclusion	Low	8 (100.0%)
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	3 (37.5%)
X-Content-Type-Options Header Missing	Low	15 (187.5%)
Information Disclosure - Suspicious Comments	Informational	2 (25.0%)
Re-examine Cache-control Directives	Informational	1 (12.5%)
Total		8

Figure 4.5: Part of Scanning Report using ZAP Proxy

of these tools is based on popularity among the community. Three different tools (nmap, nulcei, nikto) are configured. The output from these tools is shown in figures 4.6, 4.8 and 4.7. Since the three test web applications (Table 4.9) need authentication, authorization, XSS, SSRF, and SQL Injection attacks to be detected and reported, the three randomly selected tools failed to identify these vulnerabilities. This proves that the CSRS framework provides improved performance accuracy in tool selection as compared with manual selection.

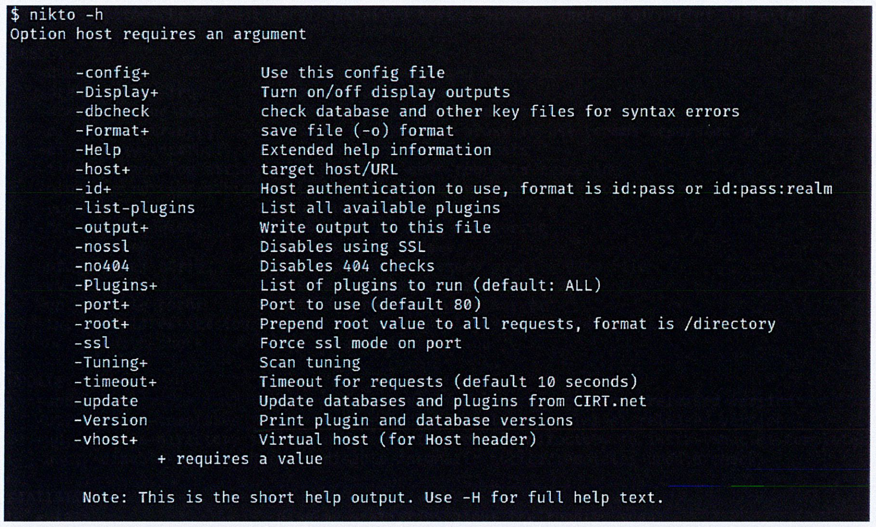


```
┌──(kali㉿kali)-[~/Desktop/Report]
└─$ nmap -h
Nmap 7.92 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
```

Figure 4.6: NMap Help

4.8 Summary

This chapter provides a comprehensive cyber assessment tool recommendation system. The decision of tool selection is based on the available input. These inputs are framed in the mathematical representation developed based on the idea of Tensor. A combination of inputs defines a unique slice of tensor that represents a unique tool. The data is gathered from multiple

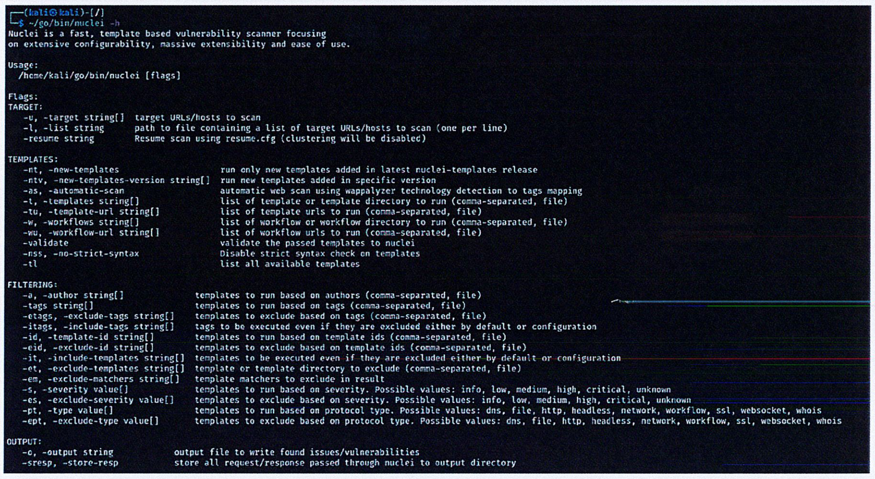


```
$ nikto -h
Option host requires an argument

       -config+            Use this config file
       -Display+           Turn on/off display outputs
       -dbcheck            check database and other key files for syntax errors
       -Format+            save file (-o) format
       -Help               Extended help information
       -host+              target host/URL
       -id+                Host authentication to use, format is id:pass or id:pass:realm
       -list-plugins       List all available plugins
       -output+            Write output to this file
       -nossl              Disables using SSL
       -no404              Disables 404 checks
       -Plugins+           List of plugins to run (default: ALL)
       -port+              Port to use (default 80)
       -root+              Prepend root value to all requests, format is /directory
       -ssl                Force ssl mode on port
       -Tuning+            Scan tuning
       -timeout+           Timeout for requests (default 10 seconds)
       -update             Update databases and plugins from CIRT.net
       -Version            Print plugin and database versions
       -vhost+             Virtual host (for Host header)
                + requires a value

        Note: This is the short help output. Use -H for full help text.
```

Figure 4.7: Nikto Help



```
┌──(kali㉿kali)-[/]
└─$ ~/go/bin/nuclei -h
Nuclei is a fast, template based vulnerability scanner focusing
on extensive configurability, massive extensibility and ease of use.

Usage:
  /home/kali/go/bin/nuclei [flags]

Flags:
TARGET:
   -u, -target string[]  target URLs/hosts to scan
   -l, -list string      path to file containing a list of target URLs/hosts to scan (one per line)
   -resume string        Resume scan using resume.cfg (clustering will be disabled)

TEMPLATES:
   -nt, -new-templates                    run only new templates added in latest nuclei-templates release
   -ntv, -new-templates-version string[]  run new templates added in specific version
   -as, -automatic-scan                   automatic web scan using wappalyzer technology detection to tags mapping
   -t, -templates string[]                list of template or template directory to run (comma-separated, file)
   -tu, -template-url string[]            list of template urls to run (comma-separated, file)
   -w, -workflows string[]                list of workflow or workflow directory to run (comma-separated, file)
   -wu, -workflow-url string[]            list of workflow urls to run (comma-separated, file)
   -validate                              validate the passed templates to nuclei
   -nss, -no-strict-syntax                Disable strict syntax check on templates
   -tl                                    list all available templates

FILTERING:
   -a, -author string[]               templates to run based on authors (comma-separated, file)
   -tags string[]                     templates to run based on tags (comma-separated, file)
   -etags, -exclude-tags string[]     templates to exclude based on tags (comma-separated, file)
   -itags, -include-tags string[]     tags to be executed even if they are excluded either by default or configuration
   -id, -template-id string[]         templates to run based on template ids (comma-separated, file)
   -eid, -exclude-id string[]         templates to exclude based on template ids (comma-separated, file)
   -it, -include-templates string[]   templates to be executed even if they are excluded either by default or configuration
   -et, -exclude-templates string[]   template or template directory to exclude (comma-separated, file)
   -em, -exclude-matchers string[]    template matchers to exclude in result
   -s, -severity value[]              templates to run based on severity. Possible values: info, low, medium, high, critical, unknown
   -es, -exclude-severity value[]     templates to exclude based on severity. Possible values: info, low, medium, high, critical, unknown
   -pt, -type value[]                 templates to run based on protocol type. Possible values: dns, file, http, headless, network, workflow, ssl, websocket, whois
   -ept, -exclude-type value[]        templates to exclude based on protocol type. Possible values: dns, file, http, headless, network, workflow, ssl, websocket, whois

OUTPUT:
   -o, -output string          output file to write found issues/vulnerabilities
   -sresp, -store-resp         store all request/response passed through nuclei to output directory
```

Figure 4.8: Nuclei Help

open-source web applications, tools, and infrastructure modes. This data is combined based on the mathematical representation and different machine learning models are trained on the dataset. Random Forest and Decision Tree models showed the highest accuracy values. Features optimization is applied to the dataset and the number of features is reduced from 54 to 46 while providing the same accuracy value as with a larger feature dataset. Three different open-source web applications are tested using the models are the most suitable tools are identified. The suitability of these tools for the cyber assessment of these applications is validated by hosting the application on a test platform and executing these tools for desired attacks and vulnerabilities. The results provided by these tools show 100% validation.

Chapter 5

Conclusion and Recommendations

5.1 Introduction

Security of software applications is one of the key requirements during and after software development. Due to the very large number of technologies and variety of programming languages, it is challenging for developers and security analysts to find suitable security assessment tools in the given context. The decision of tool selection is dependent on many variables that are difficult to handle manually. Thus, the research work attempts to answer the research question of finding a suitable security assessment tool.

5.2 Discussion

The research work is related to finding suitable security testing tools for software applications. Software developers face the challenge of finding the appropriate tool during software development and security analysis using the tool for statics analysis and dynamic analysis of the applications. In this research, a security tool recommendation system is developed. This system integrates different feature sets using the concepts of tensors from mathematics. The data is collected from different vulnerable web applications, security assessment tools, and infrastructure modes. This data is labeled and a unique class is assigned to the relevant combination.

This data is fed into different machine learning classifiers and the machine learning models are trained with the input dataset. After this, feature optimization is performed to find the best suitable and minimal set of related features that provide similar or better prediction accuracy with a more efficient execution time. This step reduced the number of features from 54 to 46. The data of three new and different web applications are used to predict the suitable tool for security testing of the web applications. It is identified that Dissuasion Tree, Random Forest, and Naïve Bayes classifiers provide 100 percent accuracy in tool prediction while other classifiers show relatively fewer accuracy values. The accuracy of prediction is dependent on multiple factors. One such factor is the size of the dataset used to train the model. Since the dataset is synthesized with a limited number of feature sets and only fewer combinations are applied to train the models, the performance of models can be improved with extended datasets.

5.3 Limitations and Future Work

The research work is initiated with the research question of finding a cyber assessment tool recommendation system. Based on a mathematical representation and machine learning classification, a solution is proposed and validated. However, there are certain limitations exist in this research that can be addressed in future works. The limitations are as follows:

- Three feature sets are given as input to the tensor model. Implementing a larger number of feature sets in a larger study would be helpful to analyse the generalization of the mathematical representation.

- The dataset used for training the machine learning classifiers is prepared with a limited number of open-source web applications, cyber assessment tools, and infrastructure modes. It is recommended to use a larger dataset containing a large number of records that will improve classification accuracy.
- Only five (05) machine learning classifiers are tested for training and testing of data for tool prediction. There is a long list of classifiers recently reported in the literature. Training and testing using these classifiers may improve the performance values.

5.4 Conclusion

This research covers the broad topic of cyber security assessment in existing web applications. The selection of suitable tools for cyber assessment remains challenging for software developers and penetration testers. This challenge becomes manifold when dealing with programs developed using many different languages and deployed on a variety of deployment platforms. The research provides a comprehensive literature review of different types of cyber attacks and vulnerabilities that exist in the source code and deployed applications. A mathematical representation based on the Tensors combination is proposed in which different features are used as input parameters. The data of different applications, tools, and infrastructure features is prepared and given as input to multiple machine learning models. The results from these models are optimized and then the suitable tools are predicted using these trained models. The prediction results generated by these models provided a promising accuracy value of 97%. However, future research is required to get better results with larger datasets, with many machine learning classifiers, and with multi-dimensional feature sets.

References

Al-Anezi, M. M. (2019). Analyzing security and performance of networks applying different network security architecture best practices.

Aldasoro, I., Gambacorta, L., Giudici, P., & Leach, T. (2022). The drivers of cyber risk. *Journal of Financial Stability*, *60*, 100989.

Aliero, M. S., Ghani, I., Qureshi, K. N., & Rohani, M. F. (2020). An algorithm for detecting sql injection vulnerability using black-box testing. *Journal of Ambient Intelligence and Humanized Computing*, *11*(1), 249–266.

Aliero, M. S., Qureshi, K. N., Pasha, M. F., Ahmad, A., & Jeon, G. (2020). Detection of structure query language injection vulnerability in web driven database application. *Concurrency and Computation: Practice and Experience*, e5936.

Altayaran, S. A., & Elmedany, W. (2021). Integrating web application security penetration testing into the software development life cycle: A systematic literature review. In *2021 international conference on data analytics for business and industry (icdabi)* (pp. 671–676).

Amalfitano, D., Fasolino, A. R., & Tramontana, P. (2011). A gui crawling-based technique for android mobile application testing. In *2011 ieee fourth international conference on software testing, verification and validation workshops* (pp. 252–261).

Apriani, W. (2019). Automatic Scanner Tools Analysis As A Website Penetration Testing. *Sistem Pendukung Keputusan Pemilihan Pimpinan*

Dengan Metode Multi Attribute Utility Theory (MAUT) di PT. Sagami Indonesia, *3*(2), 10–19.

ASVS, O. (2016). *Application security verification standard 3.0.1.*Retrieved from `https://owasp.org/www-pdf-archive/OWASP Application_Security_Verification`

Balasundaram, I., & Ramaraj, E. (2012). An efficient technique for detection and prevention of sql injection attack using ascii based string matching. *Procedia Engineering*, *30*, 183–190.

Baloch, R. (2017). *Ethical hacking and penetration testing guide.* Auerbach Publications.

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79.

Chen, H.-C., Nshimiyimana, A., Damarjati, C., & Chang, P.-H. (2021). Detection and prevention of cross-site scripting attack with combined approaches. In *2021 international conference on electronics, information, and communication (iceic)* (pp. 1–4).

Dahiya, O., Solanki, K., & Dhankhar, A. (2020). Risk-based testing: identifying, assessing, mitigating & managing risks efficiently in software testing. *International Journal of advanced research in engineering and technology (IJARET)*, *11*(3), 192–203.

De Gusmão, A. P. H., Silva, M. M., Poleto, T., e Silva, L. C., & Costa, A. P. C. S. (2018). Cybersecurity risk analysis model using fault tree analysis and fuzzy decision theory. *International Journal of Information Management*, *43*, 248–260.

Domain mod [Computer software manual]. (2022). Retrieved from `https://demo.domainmod.org`

Futcher, L., & Von Solms, R. (2008). Guidelines for secure software development. In *Proceedings of the 2008 annual research conference* *of the south african institute of computer scientists and information technologists on it research in developing countries: riding the wave of technology* (pp. 56–65).

Gadepally, V. N., Hancock, B. J., Greenfield, K. B., Campbell, J. P., Campbell, W. M., & Reuther, A. I. (2016). Recommender systems for the department of defense and intelligence community. *Lincoln Laboratory Journal, 22*(1), 74–89.

Garcia, A. A. T., Garcia, C. A. R., Villasenor-Pineda, L., & Mendoza-Montoya, O. (2021). *Biosignal processing and classification using computational learning and intelligence: Principles, algorithms, and applications.* Academic Press.

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756.*

Gupta, S., & Gupta, B. B. (2017). Cross-site scripting (xss) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management, 8*(1), 512–530.

Haley, C. B., Moffett, J. D., Laney, R., & Nuseibeh, B. (2006). A framework for security requirements engineering. In *Proceedings of the 2006 international workshop on software engineering for secure systems* (pp. 35–42).

Hasib, K. M., Iqbal, M., Shah, F. M., Mahmud, J. A., Popel, M. H., Showrov, M., ... others (2020). A survey of methods for managing the classification and solution of data imbalance problem. *arXiv preprint arXiv:2012.11870.*

Hunter, R., & Weiss, J. (2021). Cybersecurity and data centers. *Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center*, 349–358.

Husák, M., & Čermák, M. (2022). Sok: Applications and challenges of using recommender systems in cybersecurity incident handling and response. In *Proceedings of the 17th international conference on availability, reliability and security* (pp. 1–10).

Ibrahim, A. B., & Kant, S. (2018). Penetration testing using sql injection to recognize the vulnerable point on web pages. *International Journal of Applied Engineering Research, 13*(8), 5935–5942.

Kali-linux. (2022). Retrieved from `https://www.kali.org/`

Karamizadeh, S., Abdullah, S. M., Manaf, A. A., Zamani, M., & Hooman, A. (2020). An overview of principal component analysis. *Journal of Signal and Information Processing, 4.*

Kettani, H., & Wainwright, P. (2019). On the top threats to cyber systems. In *2019 ieee 2nd international conference on information and computer technologies (icict)* (pp. 175–179).

Khamdamovich, K. R., & Aziz, I. (2021). Techniques and methods of black box identifying vulnerabilities in web servers. In *2021 international conference on information science and communications technologies (icisct)* (pp. 1–4).

Khera, Y., Kumar, D., Garg, N., et al. (2019). Analysis and impact of vulnerability assessment and penetration testing. In *2019 international conference on machine learning, big data, cloud and parallel computing (comitcon)* (pp. 525–530).

Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks, 188*, 107840.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review, 51*(3), 455–500.

Lallie, H. S., Shepherd, L. A., Nurse, J. R., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021). Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers & Security*, *105*, 102248.

Li, J. (2020). Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). *Annals of Emerging Technologies in Computing*, *4*(3), 1–8.

Li, X., & Xue, Y. (2011). A survey on web application security. *Nashville, TN USA*, *3*, 10–19.

Mansour, N., & Houri, M. (2017). White box testing of web applications. *Journal of Systm and Software*, 1–9.

Neil Daswani, M. E. (2021). *Big breaches: Cybersecurity lessons for everyone 1st ed. edition.* Apress Media.

Nguyen, H. V., Kästner, C., & Nguyen, T. N. (2014). Exploring variability-aware execution for testing plugin-based web applications. In *Proceedings of the 36th international conference on software engineering* (pp. 907–918).

Notario, N., Crespo, A., Martín, Y.-S., Del Alamo, J. M., Le Métayer, D., Antignac, T., ... Wright, D. (2015). Pripare: integrating privacy best practices into a privacy engineering methodology. In *2015 ieee security and privacy workshops* (pp. 151–158).

Nunes, P., Medeiros, I., Fonseca, J. C., Neves, N., Correia, M., & Vieira, M. (2018). Benchmarking static analysis tools for web security. *IEEE Transactions on Reliability*, *67*(3), 1159–1175.

Online-invoicing-system [Computer software manual]. (2022). Retrieved from `https://github.com/bigprof-software/online-invoicing-system`

Ospos. (2022). Retrieved from `https://demo.opensourcepos.org/login`

OWASP. (2022). Source code analysis tools — owasp foundation [Computer software manual]. Retrieved from https://owasp.org/www-community/Source Code Analysis Tools

OWASP-CODELAB. (2022). Code lab [Computer software manual]. Retrieved from https://google-gruyere.appspot.com/

OWASP-COMMIX. (2022). Commix [Computer software manual]. Retrieved from (https://github.com/commixproject/commix)

OWASP-CRYPTO. (2022). Crypto detector [Computer software manual]. Retrieved from https://github.com/Wind-River/crypto-detector

OWASP-CRYPTOPASTE. (2022). Cryptopaste [Computer software manual]. Retrieved from https://github.com/HackThisSite/CryptoPaste

OWASP-DependencyCheck. (2022). Owasp dependency check [Computer software manual]. Retrieved from (https://jeremylong.github.io/DependencyCheck/

OWASP-DVWA. (2022). Dvwa [Computer software manual]. Retrieved from https://github.com/digininja/DVWA

OWASP-JUICESHOP. (2022). *Juice shop.* Retrieved from https://github.com/juice-shop/juice-shop

OWASP-MULTIDAE. (2022). *Mutillidae ii.* Retrieved from https://github.com/webpwnized/mutillidae

OWASP-NMAP. (2022). Nmaps [Computer software manual]. Retrieved from https://nmap.org/

OWASP-SAST. (2022). *Source code analysis tools.* Retrieved from https://owasp.org/www-community/Source Code Analysis Tools

OWASP-SONARQUBE. (2022). Sonarqube [Computer software manual]. Retrieved from https://github.com/SonarSource/sonarqube

OWASP-SQLMAP. (2022). Sqlmap [Computer software manual]. Retrieved from https://github.com/sqlmapproject/sqlmapr

OWASP-THCHYDRA. (2022). thc-hydra [Computer software manual]. Retrieved from (https://github.com/vanhauser-thc/thc-hydra)

OWASP-W3AF. (2022). W3af [Computer software manual]. Retrieved from https://github.com/andresriancho/w3af/

OWASP-WACKOPICKO. (2022). Wackopicko [Computer software manual]. Retrieved from https://github.com/adamdoupe/WackoPicko

OWASP-WEBGOAT. (2022). *Web goat.* Retrieved from https://github.com/WebGoat/WebGoat

OWASP-XRCross. (2022). Xrcross [Computer software manual]. Retrieved from https://github.com/pikpikcu/XRCross

OWASP-ZAP. (2022). Zap access control testing [Computer software manual]. Retrieved from https://github.com/zaproxy/zaproxy

PCIDSS. (2022). Pci data security standard (pci dss) [Computer software manual]. Retrieved from https://www.pcisecuritystandards.org

Polatidis, N., Pimenidis, E., Pavlidis, M., Papastergiou, S., & Mouratidis, H. (2020). From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks. *Evolving Systems*, *11*, 479–490.

Qasaimeh, M., Shamlawi, A., & Khairallah, T. (2018). Black box evaluation of web application scanners: Standards mapping approach. *Journal of Theoretical and Applied Information Technology*, *96*(14), 4584–4596.

Radware. (2022). *Radware 2021-2022 global threat analysis report.* Retrieved from https://www.radware.com/

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python:

Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, *11*(4), 193.

Rassokhin, D. (2020). The c++ programming language in cheminformatics and computational chemistry. *Journal of Cheminformatics*, *12*(1), 1–16.

Regan, G., Mc Caffery, F., Mc Daid, K., & Flood, D. (2013). Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model. *Computer Standards & Interfaces*, *36*(1), 3–9.

Roldán-Molina, G., Almache-Cueva, M., Silva-Rabadão, C., Yevseyeva, I., & Basto-Fernandes, V. (2017). A comparison of cybersecurity risk analysis tools. *Procedia computer science*, *121*, 568–575.

Sabillon, R., Serra-Ruiz, J., Cavaller, V., & Cano, J. (2017). A comprehensive cybersecurity audit model to improve cybersecurity assurance: The cybersecurity audit model (csam). In *2017 international conference on information systems and computer science (inciscos)* (pp. 253–259).

Sabur, A. (2018). *Analysis and management of security state for large-scale data center networks* (Unpublished doctoral dissertation). Arizona State University.

Sarkar, D., Bali, R., & Sharma, T. (2018). Practical machine learning with python. *A Problem-Solvers Guide To Building Real-World Intelligent Systems. Berkely: Apress.*

SciKit. (2022). Scikit-learn [Computer software manual]. Retrieved from `https://scikit-learn.org/stable/`

Seng, L. K., Ithnin, N., & Mohd Said, S. Z. (2018). The approaches to quantify web application security scanners quality: A review. *International Journal of Advanced Computer Research*, *8*(38), 285–312.

Setiawan, E. B., & Setiyadi, A. (2018). Web vulnerability analysis and implementation. In *Iop conference series: materials science and engineering* (Vol. 407, p. 012081).

Sherman, E., & Dwyer, M. B. (2018). Structurally defined conditional data-flow static analysis. In *International conference on tools and algorithms for the construction and analysis of systems* (pp. 249–265).

Siavvas, M., Gelenbe, E., Kehagias, D., & Tzovaras, D. (2018). Static analysis-based approaches for secure software development. In *International iscis security workshop* (pp. 142–157).

Spadini, D., Palomba, F., Zaidman, A., Bruntink, M., & Bacchelli, A. (2018). On the relation of test smells to software code quality. In *2018 ieee international conference on software maintenance and evolution (icsme)* (pp. 1–12).

Staff, V. (2022). *Report: 50% of all web applications were vulnerable to attacks in 2021.* Retrieved from `https://venturebeat.com/2022/02/21/report-50-of-all-web-applications-were-vulnerable-to-a`

Suto, L. (2010). Analyzing the accuracy and time costs of web application security scanners. *San Francisco, February*.

Tahaei, M., Vaniea, K., Beznosov, K., & Wolters, M. K. (2021). Security notifications in static analysis tools: Developers' attitudes, comprehension, and ability to act on them. In *Proceedings of the 2021 chi conference on human factors in computing systems* (pp. 1–17).

Thota, M. K., Shajin, F. H., Rajesh, P., et al. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, *17*(4), 331–344.

Todorović, B., & Trifunović, D. (2020). Security science as a scientific discipline-technological aspects. *Security Science Journal*, *1*(1), 9–20.

Tripp, O., Weisman, O., & Guy, L. (2013). Finding your way in the testing

ملخص الدراسة

حولت التكنولوجيا المتنامية تركيز الأعمال من الطرق اليدوية إلى الطرق الآلية. على سبيل المثال ، يتم تحويل التجارة إلى تجارة إلكترونية حيث يفضل الناس الشراء عبر الإنترنت. أدت هذه الحقيقة إلى زيادة تطوير تطبيقات البرمجيات في كل مجال من مجالات الحياة. ومع ذلك ، هناك تهديدات إلكترونية متزايدة للتطبيقات من قبل المستخدمين الضارين والمهاجمين. يتعين على مطوري تطبيقات البرامج التعامل مع الثغرات الأمنية في التعليمات البرمجية المصدر أثناء التطوير ويجب على مختبري الاختراق تحديد التهديدات أثناء النشر أو بعده. يوجد عدد كبير جدًا من الأدوات لتقييم التهديدات ونقاط الضعف في التطبيقات. ومع ذلك ، لا يزال اختيار الأدوات المناسبة لتحليل الكود واكتشاف الثغرات يمثل تحديًا كبيرًا لمطوري البرامج ومختبري الاختراق.

يقوم هذا البحث في البداية في بطرح أنواع مختلفة من نقاط الضعف والهجمات الموجودة في تطبيقات برامجيات الويب. بناءً على المتطلبات ، تم اقتراح نموذج رياضي باستخدام Tensors الذي يأخذ في الاعتبار الميزات المختلفة لتطبيقات الويب وأدوات الأمان والبنية التحتية للنشر. تم اعتبار تطبيقات وأدوات الويب مفتوحة المصدر في هذه الدراسة. يتم استخدام الأدوات المطروحة بواسطة مشروع الحماية المفتوح المصدر(OWASP).

تم إعداد مجموعة البيانات من الميزات المتاحة لتطبيقات الويب مفتوحة المصدر وأدوات الأمان وأنماط البنية التحتية. تُستخدم البيانات لتدريب خمسة مصنّفات مختلفة للتعلم الآلي. يتم تطبيق تحسين الميزة على مجموعة البيانات لتقليل عدد الميزات مع تحقيق قيم دقة تنبؤ أعلى. قدمت المصنفات (DT) و (RF) و (NB) أعلى قيم الدقة. تُستخدم النماذج المدربة للتنبؤ بالأدوات المناسبة لتطبيقات الويب مفتوحة المصدر المختارة عشوائيًا. لقد وجد أن RF و NB حققوا دقة تنبؤ بنسبة 100٪ بينما أظهرت DT دقة تنبؤ بنسبة 67٪. يتم التحقق من صحة الأدوات المختارة يدويًا وتجد أنها تدعم العثور على الثغرات الأمنية المطلوبة. نتائج الدراسة واعدة وتوفر أساسًا قويًا لأنظمة التوصية بأدوات الأمن السيبراني.

جـامعة البحـرين

كلية تقنية المعلومات

نظام توصية لأدوات اختبار ثغرات اختراق تطبيقات الويب

أطروحة مقدمة كجزء من متطلبات الحصول على درجة الماجستير
في الأمن السيبراني

إعـــداد

شيماء أحمد الطيران

٢٠٢٠٠٠١٩٩

إشـــراف

د. عبدالله الأسعدي

أستاذ مساعد

جامعة البحرين

مملكـــة البحـــرين

مارس ٢٠٢٣م