

SIMULATING WEB-BASED VULNERABILITIES

Radhika Soni

B.E CSE IS

Chandigarh University

Gharuan, Punjab, India

sonyradhika07@gmail.com

Simran Kaur

B.E CSE IS

Chandigarh University

Gharuan, Punjab, India

kaursimran17jan@gmail.com

Krishma

B.E CSE IS

Chandigarh University

Gharuan, Punjab, India

krishma0803@gmail.com

Sheetal Laroiya

Assistance Professor

Chandigarh University

Gharuan, Punjab, India

sheetal.e15433@cumail.in

Abstract—The project titled "Simulating web-based vulnerabilities" addresses the escalating concern of unauthorized and potentially malicious activities performed on websites by finding and exploiting vulnerabilities in web pages by hackers or cyber criminals. The goal is to develop a comprehensive understanding of various types of vulnerabilities, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and insecure direct object references (IDOR), among others. The project will involve studying the underlying principles of these vulnerabilities, exploring real-world examples, and implementing practical solutions to address them. By employing penetration testing techniques, vulnerabilities in web pages and networks are identified, offering a glimpse into potential security breaches. Key modules explore hacker methodologies, programming languages, web servers, and testing tools like Burp Suite, OWASP ZAP, Nmap, and Metasploit. The ultimate goal is to detect and prevent unauthorized access, thus bolstering the overall security of web applications. This project is all about understanding and preventing security vulnerabilities in web applications. We're creating a simulation that mimics real-world scenarios where hackers might try to break into websites. Through techniques like penetration testing, we'll uncover weaknesses in web pages and networks that could be exploited by attackers. Our project covers various aspects of web security, including how hackers think and operate, the programming languages commonly

used in web development, different web servers and databases, and tools like Burp Suite, OWASP ZAP, Nmap, and Metasploit that help us test for vulnerabilities. Finding and fixing these vulnerabilities before unauthorized users can take advantage of them is our primary goal. By doing so, we're making web applications more secure and protecting them from potential cyber-attacks.

Keywords— *Web-based vulnerabilities, penetration testing, web application security, hacker methodologies, programming languages, web servers, testing tools, Burp Suite, OWASP ZAP, Nmap, Metasploit.*

INTRODUCTION

In the connected digital world of today, web application security is critical. Because cyber-attacks are becoming more sophisticated, developers, security experts, and system administrators need to understand and mitigate vulnerabilities within their web applications.

Vulnerabilities in websites can have a wide range of impacts, depending on the nature of the vulnerability and the intentions of the attacker. Due to these vulnerabilities, there are many impacts on the sensitive information of an organization like data breaches through SQL injection, insecure direct object references, and security misconfiguration allowing attackers to obtain unapproved access to private information such as bank records, personal data, and other sensitive data. Attacks on websites can lead to financial theft of payment information,

fraudulent transactions, or theft of money from victims. Some vulnerabilities can be exploited to disrupt the normal operation of websites or online. Vulnerabilities such as cross-site scripting (XSS) and cross-site request forgery (CSRF) can be used to compromise user accounts, leading to unauthorized access and misuse of user data. A successful online attack can harm an organization's reputation and brand, costing it clients and business possibilities.

The Project Simulating Web-based Vulnerabilities provides a clear insight into secure web application development and deployment by simulating various vulnerabilities based on the web and network. The goal of the project is to simulate numerous typical vulnerabilities present in web-based systems, thereby offering an immersive and practical learning experience in the field of web security. It helps in gaining practical experience in identifying, exploiting, and mitigating web security issues thereby enhancing their ability to secure web applications through best practices and defensive strategies.

The key objective of the project is researching and understanding common web vulnerabilities and their impact on web applications. Implementing security measures and recommended practices, like input validation, safe coding techniques, and the use of web application firewalls (WAFs), to guard against these vulnerabilities and Conducting penetration testing and vulnerability assessments to identify and address potential security weaknesses. Creating educational resources and documentation to raise awareness about web security best practices among developers and stakeholders.

PROJECT OVERVIEW

A. Website Vulnerability

A software code error, system configuration error, or other weakness in the website, or any of its parts and operations, is termed a vulnerability. Attackers can obtain unauthorized access to the organization's systems, procedures, and confidential information or

services. This can result in downtime. Loss of revenue, and inconvenience to users.

assets through web application flaws. With such unauthorized access, attackers can plan attacks, commandeer apps, utilize privilege escalation to steal data, interrupt critical services on a massive scale, and more. Vulnerabilities such as SQL injection, insecure direct references (IDOR), and security misconfiguration can allow attackers to obtain sensitive data—such as financial records, personal information, and other private information—without authorization. Attackers may exploit vulnerabilities to steal intellectual property, trade secrets, or proprietary information from websites, leading to competitive disadvantage and financial losses. Some vulnerabilities can be exploited to disrupt the normal operation of a website or online service. This can result in downtime, loss of revenue, and inconvenience to users. Many attacks lead to financial losses for individuals and conduct fraudulent transactions, or extort money from victims. A successful website hack or data breach might harm the standing of the company or person in charge of the website. Losing the trust of stakeholders, partners, and customers may result from this.

B. Current Study and Problem Description

There is a fair lot of enterprise and small business software available right now, but the issue with all of them is that they get overly technical highly intricate, as well as very costly to buy particularly affecting startups and small Indian businesses that cannot afford them. Although software like Acunetix and Burp offers a variety of plans, their initial fees can range from thousands to millions of dollars, and they continue to rise annually. They offer features that are superfluous, unnecessary for many businesses, and challenging to utilize. However, it gets more difficult for consumers to choose inexpensive options since there aren't many alternatives and because most people don't know much about the programs themselves. Current software solutions are primarily

designed for multinational companies with large staff bases and high revenue. The costs. Few software offers affordable and user-friendly vulnerability scanning solutions.

scanners are crucial. The scanner crawls a domain's web pages, scanning each URL with different payloads.

C. Types of vulnerabilities

SQL injection (SQLi)- This type of attack is performed by injecting malicious SQL code into input fields by an attacker to modify the SQL query on a website. Attackers may be able to see, alter, or remove sensitive data from the database as a result.

Cross-Site Request Forgery (CSRF)- When a person on a website where they have authenticated is tricked into performing unwanted actions, this is known as a cross-site request forgery (CSRF) vulnerability. This may result in activities like adjusting account preferences or carrying out fraudulent transactions.

Cross-site Scripting (XSS)- Attackers insert harmful scripts onto web pages by using XSS vulnerabilities. This can be used to deface the website, divert visitors to malicious domains, or steal session cookies.

Insecure Direct Object References (IDOR)- When an application discloses internal implementation objects—like files or database records—without conducting the required authorization checks, IDOR vulnerabilities occur. This can allow attackers to access sensitive data or perform unauthorized actions.

Security Misconfiguration- It occurs when a website is not properly configured to secure its resources. This can include default settings, unnecessary services enabled, or updated software with known vulnerabilities.

There is a lack of automated scanners for Detecting vulnerabilities in web apps. Some scanners are very pricey. These actions can result in server defacement, hijacking, and data theft. This poses a security risk for both businesses and government employees. Creating Inexpensive

Remote Code Execution- Attackers can take complete control of a web server and its resources by using RCE vulnerabilities to execute arbitrary code on the server.

XML External Entity (XXE)- Attackers can take complete control of a web server and its resources by using XXE vulnerabilities to execute arbitrary code on the server.

D. Finding Vulnerabilities

It involves a combination of manual testing, automated scanning, and vulnerability assessment tools. the basic concept is to create software that does all the major tasks as done by much more expensively priced software used by companies, as well as keeping the costs down. The software will scan full websites and find vulnerabilities as well as give information and solutions on how to fix it. There are some methods and tools used to find vulnerabilities.

Manual testing- This involves reviewing the website's code, configuration, and functionality to identify potential vulnerabilities. This can include analyzing input fields for SQL injection, checking for cross-site scripting (XSS) vulnerabilities, and reviewing access controls and authentication mechanisms.

Automated Scanning- These tools can help identify common vulnerabilities in websites quickly and efficiently. The websites are scanned by these programs for known vulnerabilities like SQL injection. OWASP ZAP, XSS, CSRF, Burp Suite, and acunetix.

Vulnerabilities Assessment Tools- The tools can help identify vulnerabilities in websites by scanning for security weaknesses in the website's code, configuration, and infrastructure. These assessments provide of the website's security posture and recommend measures to mitigate vulnerabilities. Tools like Nessus, Qualys, and OpenVAS.

Penetration testing- Pen testing involves simulating a cyberattack on a website to identify and exploit vulnerabilities. This can assist in finding security flaws that automated tolls might miss. Penetration

Bug Bounty Programs- Through these initiatives, ethical hackers and security researchers can expose vulnerabilities on websites in exchange for cash or recognition. By taking part in bug bounty programs, you can find vulnerabilities and fix them before bad actors take use of them.

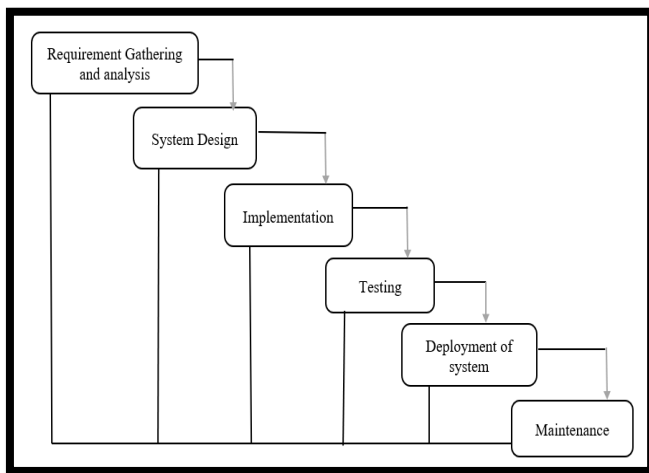


Fig.1 Finding Vulnerabilities

E. Feasibility

Operational feasibility- It refers to the ability to utilize support and fulfill the system's vital tasks. The proposed system is user-friendly and customizable, requiring minimal prior understanding of the technology involved. The application's interactive UI design allows users to quickly navigate and familiarise themselves with it.

testing can offer important insights into the security posture of a website and is usually carried out by skilled security specialists.

Code review- Reviewing the website's code for security vulnerabilities is an important step in identifying potential weaknesses. This can involve manual code review by security experts or the use of automated code analysis tools to find common weaknesses such as insecure coding practices, hardcoded passwords, and other issues.

Technical feasibility- The recommended software and hardware requirements for our application are technically feasible and align with existing technologies. The project aims to provide affordable and practicable software. Our application will be designed to function on any computer starting this decade.

Economic feasibility: The project's scope and advantages outweigh its cost. The primary agenda of the software is to provide identical vulnerability detection techniques as high-end enterprise solutions, but at a lower cost and with a more user-friendly interface.

RELATED WORK

A Framework for Web Application Vulnerability Detection Author name by Asra Kalim, C K Jha, Deepak Singh Tomar, Divya Rishi Sahu (2017)

The Internet has an impact on almost every aspect of human existence because of its constant growth in terms of features, usage, speed, ease of use, accessibility from anywhere in the world, etc. On the other hand, hackers are also using new strategies and methods to attack the digital world by taking advantage of online application weaknesses. It is critical to analyze these vulnerabilities to create a secure digital social environment. Two approaches can be used. First, manual analysis, is susceptible to errors because of humanitarian leniency, the way that technology is always changing, and fraudsters'

methods of attack. Secondly, by using the current web application vulnerability scanners, which occasionally have a false alarm rate. As a result, it's critical to offer a model that can recognize many categories of website weaknesses, including client-side, server-side, and communication-side vulnerabilities. This report reviewed the literature to identify novel attack vectors, vulnerabilities, detection techniques, and new areas of application in the same field. A model can be easily improved over time.

Web Vulnerability Scanners Author name by Angel Rajan, Emre Erturk (2015)

Acunetix's AcuSensor and AcuMonitor technologies contribute to the production of more precise and potentially vulnerable outcomes. This paper's goal is to familiarise current computer security students with vulnerability scanners. Second, a review of the literature on security vulnerability scanners is included in this work. Finally, the browser and mobile device viewpoints are used to discuss online vulnerabilities.

A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners by Suliman Alazmi (Member, IEEE), and Daniel Conte de Leon (Member, IEEE) (2017)

In recent years, successful security breaches have increasingly targeted web applications. Currently, the main way they are secured is by using specialized tools called Web Application Vulnerability Scanners (WVSs) to look for their flaws. Even though these dynamic testing methodologies have some benefits, there is still little research that examines their features and detection abilities methodically. This article presents the findings of a Systematic Literature Review (SLR) that examined the features and effectiveness of the most popular WVSs. A total of ninety research publications were thoroughly examined. Just twelve WVSs (out of thirty) that were

For many businesses, one of their main worries is cloud security. The increase in both the quantity and the need for improved website security grows with the scale of those websites. Web vulnerability identification and testing by hand might take a long time. Web application vulnerabilities can be found with the use of Automated Web Vulnerability Scanners (WVS). One of the most popular vulnerability scanners is Acunetix. Furthermore, Acunetix is simple to use and put into practice. The scan results include information on how to repair the vulnerabilities in addition to specifics on the issues themselves.

collected and reported had at least one quantitative efficacy assessment. On these twelve WVS, fifteen separate evaluation studies were carried out. We found that only two of the Top Ten vulnerability categories identified by the Open Web Application Security Project (OWASP)—cross-site scripting (XSS) (8/15) and SQL injection (13/15)—were mainly looked at in these evaluations.

Analysis of Automated Web Application Security Vulnerabilities Testing by parish touseef , Khubai Amjad Alam, abid jaml, hamza tauseef, Sahar Ajmal, bisma rehman, Sumaira Mustafa (2019)

The proliferation of online threats around the world in recent years has shown an urgent need for security models and preventative measures. This research's early results examine a thorough examination of the literature on security testing for web application vulnerabilities. After two research topics were guaranteed, thirty of the original 237 investigations were eventually included as Primary Research investigations (PRS). The results show that the eight most common online application threats are SQL injection, XSS, and sensitive data disclosure. Similarly to this, the research community hasn't given Invalidated Redirects and Forwards/Under Protected APIs much thought. This study's scope has

been expanded to include testing for web application vulnerabilities and identifying pertinent data sets.

Common Requirements for Web Application Vulnerability Scanner for the Internet of Things (2020)

The paper presents three conventional needs for website vulnerability scanners for Internet of Things devices: support for browser rendering engines, false positive minimizing, and device setting change minimization. These specifications were derived from the prior project's experience with security flaws in home gateways. By removing unusual vulnerabilities from IoT devices, an I-WAVS should reduce false positives. It should also have the ability to restrict access to specific links, buttons, or pages to limit device settings modifications.

FRAMEWORK

Designing a web vulnerability scanner involves several key components and considerations to ensure its effectiveness and efficiency. Simulating web-based vulnerabilities entails adhering to a systematic methodology to recognize, exploit, and address vulnerabilities within web applications. The following provides a general approach that can be employed for simulating web-based vulnerabilities:

A. Needs assessments and analysis

In this step, we will identify the target web application and gather information about its architecture, technologies used, and potential entry points. Make sure there is authorization from the website owner or the relevant authority before conducting any scans. Unauthorized scanning is both unethical and potentially illegal.

B. System architecture

This involves identifying input points in the application such as forms, URLs, and cookies where user input is accepted. Use tools like Burp Suite or

A network-based vulnerability scanner for detecting SQLI attacks in web application (2012)

To address this, vulnerability scanners have been offered, but other alternatives that can detect SQLI completely. The tools that are now in use have very low accuracy ratios and a high rate of false positives. In addition, all of these tools require a significant amount of time to scan. Their scanners have a very slow scanning speed. By concentrating on a single vulnerability at a time, additional flaws go undetected. Hence the paper presents a network-based vulnerability scanner approach that provides better coverage and with no false positives within a short period.

OWASP ZAP to map the application and identify all accessible pages, parameters, and functionalities. A crawling mechanism to discover and map the structure of the target website, identifying accessible pages, forms, and parameters.

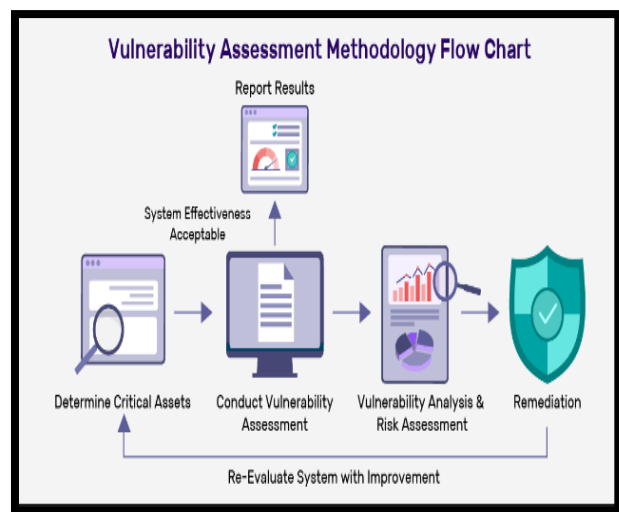


Fig.2 Framework

C. Inspection and exploitation

Finding a website's vulnerabilities entails scanning it with a vulnerability scanner. Using Owasp Zap, Burp Suit, and Nessus tools, scan details. To determine which ports are open on the target machine, use a port scanning program such as Nmap. Open ports may be a sign that a system's services are operational and available. After determining which open ports are in use and which services are utilizing them, look into any known vulnerabilities related to those services. Information about known vulnerabilities can be found on websites such as the National Vulnerability Database (NVD) or the Common Vulnerabilities and Exposures (CVE) database. Make use of an exploit or tool designed to specifically target the vulnerability you have found. To exploit the vulnerability, can entail sending malicious payloads to the service that is operating on the open port. If the exploit is successful, we may gain unauthorized access to the target system.

D. Countermeasures

Very common website attacks include malware attacks (ransomware, Botnets), Phishing attacks, Code injection attacks (Cross-site scripting, Malvertising), SQL injection attacks, and Denial of service (DoS) attacks.

Malware Attacks (Ransomware, Botnets):

- Use up-to-date antivirus software to detect and remove malware.
- Regularly scan your website for malware and vulnerabilities.
- Keep software and plugins up to date to prevent exploitation of known vulnerabilities.

Code Injection Attacks (Cross-site Scripting, Malvertising):

- Verify and Clean user input to stop attacks using code injection.

Depending on the nature of the vulnerability, this could allow you to execute commands, access files, or perform other actions on the target system.

Here is a flowchart describing the steps taken if we found the vulnerability in a website.

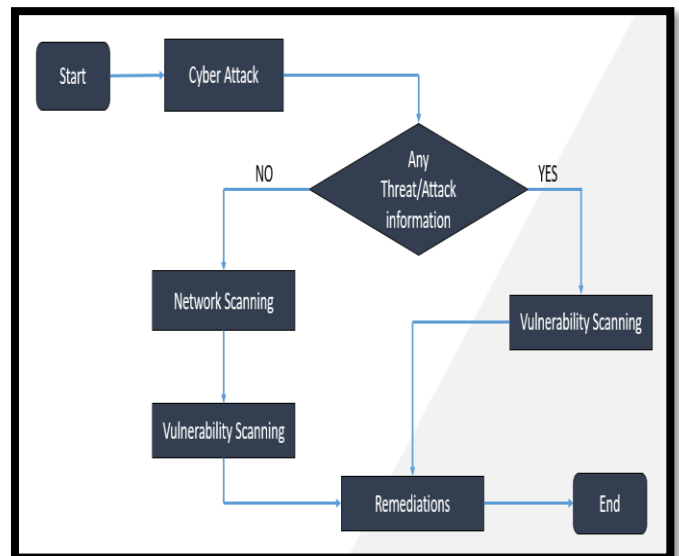


Fig.3 Flowchart of scanning process

- To limit the origin of executable scripts on the website, use a Content Security Policy(CSP).
- To guard against vulnerabilities. Patch and upgrade your web server and application regularly.

Phishing Attacks:

- Educate users about phishing techniques and how to identify phishing attempts.
- Use email filtering to detect and block phishing emails.
- Use multi-factor authentication (MFA) to protect accounts from unauthorized access.

SQL Injection Attacks:

- To stop SQL injection attacks, use prepared statements and parameterized queries.
- To defend against malicious input, provide input validation and sanitization.

- Use least privilege principles to restrict database access to only necessary operations.

Denial of Service (DoS) Attacks:

- Filter and block malicious traffic via a Web Application Firewall (WAF).
- Use IP blocking and rate limitation to lessen the effect of Dos assaults
- Use a content delivery network (CDN) to distribute traffic and absorb DDoS attacks.

FINDINGS



Fig.4 Website Interface

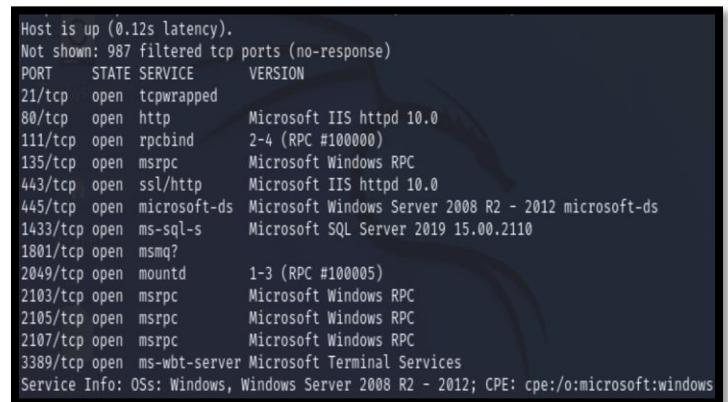


Fig.5 Information Gathering using NMAP

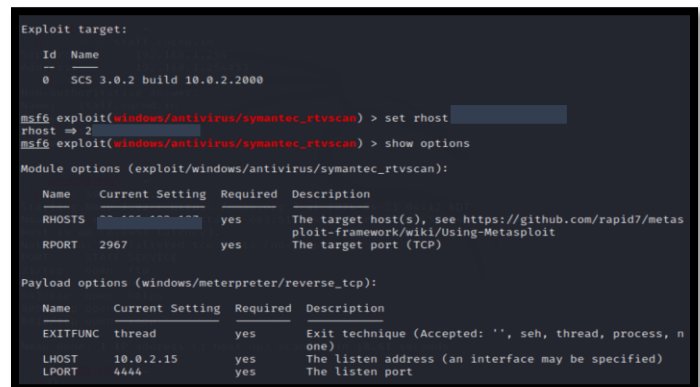


Fig.6 exploit using Metasploit

SUMMATION

The project's main goal was to simulate web-based vulnerabilities to understand their traits, effects, and defenses. We learned about the importance of web application security by looking into flaws like SQL injection, Cross-Site Scripting, Local and Remote File Inclusion, and Command Injection. Through this project, we were able to gain a deeper understanding of the threats such vulnerabilities present to user privacy and data security. We also explored various techniques and tools for identifying and resolving these weaknesses to enhance the security posture of web applications. Because of how important the internet is to the

modern economy, security risks are more common than ever. Securing our resources, data, and user privacy is paramount. Despite advancements in security strategies, tools, models, and methods, hackers remain a persistent threat. The proposed system seeks to identify vulnerabilities, including SQL injection, Cross-Site Scripting, Local and Remote File Inclusion, and Command Injection, in web applications. It also offers guidance on how to address vulnerable URLs and their associated vulnerabilities. Our approach addresses these diverse vulnerabilities effectively and is designed to be cost-effective, specifically targeting large commercial applications. The results have been satisfactory. Looking ahead, it is imperative to persist in

researching and implementing robust security measures to counter evolving threats. By remaining abreast of the latest security trends and best practices, we can effectively protect web applications and the sensitive data they handle.

REFERENCES

- [1] Ping (2017) proposed a method for detecting second-order SQL injection, presented at the 2017 IEEE 2nd ITNEC conference.
- [2] Tajbakhsh and Bagherzadeh (2015) presented a framework for dynamically preventing Local File Inclusion, discussed at the 2015 IKT conference.
- [3] Sadaphule et al. surveyed preventing website attacks based on Remote File Inclusion, published in the International Journal of Advance Engineering and Research Development.
- [4] Chen and Wu (2010) developed an automated vulnerability scanner for injection attacks based on the injection point, presented at the 2010 ICS conference.
- [5] Patil, Marathe, and Padiya (2016) designed an efficient web vulnerability scanner, presented at the 2016 ICICT conference.
- [6] Lee and Park (2017) discussed common requirements for web application vulnerability scanners for the Internet of Things, presented at the 2017 ICSSA conference.
- [7] Singh and Roy (2012) developed a network-based vulnerability scanner for detecting SQL injection attacks in web applications, presented at the 2012 RAIT conference.
- [8] Fonseca, Vieira, and Madeira (2007) tested and compared web vulnerability scanning tools for SQL injection and XSS attacks, presented at the 2007 PRDC conference.
- [9] Qianqian and Xiangjun (2014) researched and designed a web application vulnerability scanning service, presented at the 2014 ICSESS conference.
- [10] Huang et al. (year not provided) discussed non-detrimental web application security scanning, presented at the 15th ISSRE conference.
- [11] Christos Xenakis, Anastasios Stasinopoulos, and Christoforos Ntantogian developed Commix, a tool for detecting and exploiting command injection flaws, in 2015.
- [12] HackerOne published a list of the top 10 most impactful and rewarded vulnerability types on their website, accessed on October 22, 2019.
- [13] Abdalla Wasef Marashdih and Zarul Fitri Zaaba researched detection approaches for cross-site scripting in web applications in 2016.
- [14] Dr. Parminder Kaur and Ms. Daljit Kaur studied cross-site scripting attacks and methods for preventing them during development in 2017.
- [15] OWASP published the OWASP Top 10 - 2017, listing the ten most critical web application security risks, in 2017.
- [16] OWASP created an XSS filter evasion cheat sheet, last updated in 2019, to help developers protect against cross-site scripting attacks.
- [17] Alvise Rabitti, Gabriele Tolomei, Riccardo Focardi, Mauro Conti, and Stefano Calzavara. Using machine learning to discover online vulnerabilities: The example of cross-site request forgery. IEEE Security & Privacy (March 2020), 18(3):8–16.