# A PARADIGM SHIFT IN XSS-DOM MITIGATION VIA DYNAMIC CONTENT SECURITY POLICY

**A Project-1 Report**

***Submitted by***


| | |
|---|---|
| Krishnaa S | 2020506045 |
| Jawahar A S | 2020506035 |
| Thamizharasi M | 2020506102 |


***Under the supervision of***

Dr. P. Kola Sujatha

*In partial fulfilment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**MADRAS INSTITUTE OF TECHNOLOGY CAMPUS**

**ANNA UNIVERSITY, CHENNAI – 600044**


NOVEMBER 2023

# BONAFIDE CERTIFICATE

Certified that this mini-project report titled **"A PARADIGM SHIFT IN XSS-DOM MITIGATION VIA DYNAMIC CONTENT SECURITY POLICY"** is the bonafide work of Krishnaa S (2020506045), Jawahar A S (2020506035), Thamizharasi M (2020506102) who carried out the project work under my supervision.


**Signature**

**Dr. M. R. Sumalatha**

**HEAD OF THE DEPARTMENT**

Professor

Department of Information Technology

MIT Campus, Anna University

Chennai – 600044

**Signature**

**Dr. P. Kola Sujatha**

**SUPERVISOR**

Associate Professor

Department of Information Technology

MIT Campus, Anna University

Chennai – 600044

# ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement made us accomplish this project.

We express our thankfulness to our project supervisor **Dr. P. Kola Sujatha**, Associate Professor of the Department, Department of Information Technology, MIT Campus, for providing invaluable support and assistance with encouragement which aided to complete this project.

We are thankful to the panel members **Dr. P. AnandhaKumar,** and **Dr. M. Hemalatha,** Department of Information Technology, MIT Campus for their invaluable feedback in reviews.

Our sincere thanks to **Dr. M. R. Sumalatha**, Head of the Department of Information Technology, MIT Campus for catering all our needs giving out limitless support throughout the project phase.

We express our gratitude and sincere thanks to our respected Dean of MIT Campus, **Dr. J. Prakash**, for providing excellent computing facilities throughout the project.

**Krishnaa S** **2020506045**

**Jawahar A S** **2020506035**

**Thamizharasi M** **2020506102**

# ABSTRACT

In today's web application landscape, securing client-side interactions is crucial. This thesis addresses the menace of Cross-Site Scripting attacks, especially those targeting Document Object Model vulnerabilities. The approach presented involves detecting and preventing XSS-DOM attacks through dynamically generated Content Security Policy files and advanced pattern matching. The system uses a real-time adaptive CSP generation mechanism to tailor security policies based on the web application's context, offering robust protection. The integration of pattern matching algorithms enhances detection capabilities, identifying malicious patterns in user inputs. Objectives include building a flexible system for proactive XSS-DOM threat mitigation, with automatic CSP directive generation based on application content and efficient pattern matching for early detection. This research contributes to a paradigm shift in XSS-DOM security, emphasizing a dynamic, context-aware, and pattern-sensitive approach. Evaluation includes rigorous testing against diverse XSS-DOM attack scenarios, with comparative analysis against existing mitigation strategies to assess effectiveness. The study aims to advance understanding and provide practical insights for developers, security professionals, and policymakers, ultimately enhancing client-side security in web applications against evolving cyber threats.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| XSS | Cross- Site Scripting |
| DOM | Document Object Model |
| CSP | Content Security Policy |
| CSRF | Cross-Site Request Forgery |
| OWASP | Open Web Application Security Project |
| HTML | Hyper Text Markup Language |
| ML | Machine Learning |
| JSCSP | JavaScript based Content Security Policy |
| CVE | Common vulnerability exposure |
| WAF | Web application firewalls |
| HTTP | Hyper Text Transfer Protocol |

# CHAPTER 1
# INTRODUCTION

## 1.1 OVERVIEW

In the rapidly advancing landscape of web applications, the specter of security concerns looms large, with a particular emphasis on the imperative need to thwart XSS attacks meticulously targeting vulnerabilities within the Document Object Model. The jeopardy posed by XSS-DOM vulnerabilities extends beyond mere data compromise; it strikes at the very heart of system integrity, raising alarming concerns for both developers and end-users. As web applications evolve, fortifying the defences against XSS-DOM exploits becomes an increasingly intricate endeavour. The dynamic nature of online content, coupled with the nuanced interactions facilitated by modern web applications, adds layers of complexity to the already challenging task of mitigating these security threats. The profound challenge lies in discerning and rectifying XSS-DOM vulnerabilities in a manner that harmonizes with the seamless user experience expected in contemporary web usage. This narrative encapsulates the essence of a multifaceted issue – the pressing need to comprehend and rectify XSS-DOM vulnerabilities, navigating through the intricacies of web application security. Through this comprehensive exploration, we delve into the various dimensions of the problem, setting the stage for a thorough investigation into the multifaceted realm of XSS-DOM challenges within the intricate fabric of modern web application security.

## 1.2 RESEARCH CHALLENGES

A major difficulty in maintaining the security of constantly changing online applications is mitigating XSS-DOM assaults. The main challenges include recognising various attack patterns, responding quickly to threats, and striking a balance between strong security and user friendliness. Developing strong CSP

guidelines and putting in place a thorough assessment system for thorough testing are crucial components. It takes an interdisciplinary approach to solve these problems, combining knowledge from dynamic policy adaption, pattern recognition, and online security. These difficulties push research towards creative fixes to strengthen security against XSS-DOM attacks in the ever-changing world of contemporary online applications.

## 1.3 OBJECTIVE

This project's main goal is to create a proactive, adaptive system that can identify and stop XSS-DOM assaults in web applications. The main goal is to create a framework for generating dynamic Content Security Policies (CSPs) that can adapt to the changing threat landscape. The project also intends to use sophisticated pattern matching algorithms to improve the system's capacity to recognise and efficiently neutralise harmful patterns in user inputs. The main objective is to strengthen client-side security in online applications by offering a dynamic, context-aware, and pattern-sensitive approach that will ultimately lead to a paradigm change in XSS-DOM security. The study will include thorough testing against a range of XSS-DOM attack scenarios, a comparison with current mitigation techniques, and useful advice for security and web development experts.

## 1.4 SCOPE OF THE PROJECT

This project, which addresses XSS-DOM vulnerabilities, has great potential to advance online application security. Its scope includes creating a system to generate dynamic CSP and combining it with sophisticated pattern matching algorithms to identify and stop XSS-DOM assaults. The study will support a paradigm change in security tactics by highlighting pattern sensitivity and flexibility. The project's results have wide-ranging consequences, offering useful perspectives to web developers, security experts, and legislators. This

study is important in improving client-side security in web applications since it focuses on real-time threat adaption and efficient pattern matching.

## 1.5 CONTRIBUTION

Particularly in the area of XSS DOM mitigation, the project significantly advances online application security. Its main achievements are the incorporation of sophisticated pattern matching techniques and the creation of a dynamic system for generating Content Security Policies (CSPs). Through dynamic adaptation of CSP according to the environment of the application, the system improves defence against XSS-DOM assaults. Efficient pattern matching techniques are also incorporated to help discover harmful patterns in user inputs early on. Through their joint efforts, XSS-DOM security solutions undergo a paradigm change that promotes a more dynamic, context-aware, and pattern-sensitive approach to successfully combat developing threats.

## 1.6 WEB ATTACKS

Web applications have grown essential in the rapidly changing digital world, yet they are always at risk from different types of cyberattacks. Web assaults compromise the availability, confidentiality, and integrity of online systems. They can take many different forms, from simple vulnerabilities to complex strategies. By utilising strategies like injection assaults, XSS, CSRF and others, attackers take advantage of weaknesses in online applications. These assaults frequently target human input and prey on holes in security protocols. Understanding and preventing these various web assaults is crucial to maintaining a safe online environment for users and protecting sensitive data as web technologies become more intricate and linked.

## 1.7 INJECTION

Web applications are vulnerable to injection attacks, which target weaknesses that occur when processing untrusted input. Notable instances are SQL injection, operating system command injection, and the progressively widespread XSS. Attackers can insert malicious code into input fields with this type of attack, making it easier to execute commands without authorization, access databases without authorization, and possibly compromise whole systems. Injection vulnerabilities are recognised as serious security issues by the OWASP. Finding and fixing these issues requires extensive application security testing. The implementation of parameterized SQL queries and the sanitization of user input by eliminating special characters are two examples of remediation measures that are supported by OWASP.

## 1.8 CROSS - SITE SCRIPTING

XSS represents a critical security vulnerability enabling attackers to execute malicious code in a victim's web browser, evading the same-origin policy. Successful exploitation grants hackers the ability to impersonate users, accessing data and executing actions as legitimate users. In cases where the victim holds privileged access, adversaries may seize control of the application server, paving the way for additional attacks. Vulnerable websites allow attackers to manipulate client-side scripts, introducing tampered JavaScript to be executed within users' browsers. This interception of interactions between users and web/application servers poses severe risks. XSS manifests in various forms, including Reflected XSS, Stored XSS, and DOM-based XSS, with dynamic websites relying on JavaScript being particularly susceptible to such attacks.

## 1.9 DOM BASED XSS

A serious security risk is posed by DOM-based XSS, which enables hackers to alter the client's browser environment in order to insert harmful payloads into websites. By using the DOM this type of cross-site scripting may coordinate attacks against the client-side post-page load. Interestingly, both the HTML source code and the server response are left intact, which makes it difficult to use standard traffic analysis tools to identify the malicious input. Client-side scripting and runtime examination of the document object model are required for detection. Adversaries target apps that have an executable route that allows data to flow from a source to a sink in DOM-based cross-site scripting (XSS) assaults. JavaScript attributes such as document.URL, document.referrer, location.search, and location.hash are included in sources. These attributes might be places where malicious input could be introduced. Conversely, sinks, which include element.innerHTML, setTimeout, eval, and setInterval, are the places or methods in HTML rendering that carry out the harmful purpose. Remarkably, our project's primary goal is to mitigate cross-site scripting vulnerabilities that stem from DOM-based assaults, tackling the particular difficulties presented by this type of vulnerability.

## 1.10 PATTERN MATCHING

A computational method essential to many disciplines, including data analysis and computer science, is pattern matching. It entails locating recurrent structures or patterns in data to facilitate effective manipulation or retrieval. Pattern matching is an essential tool for cybersecurity threat detection and prevention. Systems are able to recognise and react to possible security threats by establishing certain patterns linked to malevolent actions. This method is commonly used in antivirus software, intrusion detection, and other security programmes to identify and block patterns suggestive of online dangers. All

things considered, pattern matching improves computing capacities for identifying, deciphering, and responding to structured data.

## 1.11 CONTENT SECURITY POLICY

Content Security Policy (CSP) is a web security standard designed to mitigate the risks of cross-site scripting (XSS) attacks. It enables web developers to declare the sources from which various types of content can be loaded, reducing the likelihood of unauthorized script execution. By defining a set of content policies, CSP enhances the protection of web applications against XSS vulnerabilities. These policies restrict the origins that browsers consider valid for specific types of content, enhancing the overall security posture of websites. CSP acts as a crucial layer of defence, providing a declarative approach to specifying and controlling content origins in a web page.

## 1.12 ORGANIZATION OF THE THESIS

The rest of the thesis is organized as follows. Chapter 2 presents the literature survey on XSS based DOM attack detection and prevention by various approaches. Chapter 3 models the architecture and system design, outlining the architecture and design of the proposed system. Chapter 4 explains about the implementation details, explaining the specifications and environment. The results achieved are presented in Chapter 5. Chapter 6 presents the conclusion and some possible avenues for future research on the topic.

# CHAPTER 2
# LITERATURE SURVEY

## 2.1 CONTENT SECURITY POLICY APPROACH

JSCSP proposes an innovative defence against XSS attacks (Guangquan Xu et al., 2022). While the W3C recommends CSP for XSS mitigation, its low adoption (less than 3.7% according to Google's survey) is attributed to deployment challenges. JSCSP addresses this by supporting real-world browsers and automatically generating security policies. It introduces a self-defined policy, confining JavaScript functions, DOM elements, and data access. The algorithm efficiently generates and enforces policies in a cascading manner, offering finer granularity than CSP. Implemented as a Chrome extension, JSCSP demonstrates compatibility with popular JavaScript libraries and outperforms other XSS defence solutions in performance.

## 2.2 SUBSEQUENCE MATCHING APPROACH

This research introduces a method for detecting XSS vulnerabilities, a significant threat to web security. The proposed technique employs a subsequence matching algorithm, emphasizing the identification of common subsequence between input parameters and generated data (Jingyu Zhang et al., 2021). By implementing a threshold to limit the common substring length, the approach successfully detects XSS vulnerabilities. The results demonstrate the effectiveness of the method in mitigating the growing threat of XSS attacks, showcasing its potential as a robust solution for enhancing web security against this prevalent threat.

## 2.3 RECORDED CVE DATABASE APPROACH

XSnare is a Firefox extension that offers client-side protection against XSS attacks (J. C. Pazos et al., 2021). It acts pre-emptively, safeguarding users before

developers release patches or server operators apply them. XSnare blocks XSS attacks by leveraging prior knowledge of web application HTML templates and rich DOM context. It utilizes an exploit database, crafted from recorded CVEs, to identify and sanitize injection points, preventing malicious code from appearing in the DOM. Even when under attack, XSnare displays a secured version of the site. It effectively defends against 93.8% of tested XSS exploits. This application-specific solution is unique and minimally impacts web page loading times.

## 2.4 ANTI-DOM XSS FRAMEWORK APPROACH

The prototype tool demonstrates the practical implementation of the anti-DOM XSS framework in a controlled environment (K. Ali et al., 2014). Through comprehensive testing and analysis, the framework exhibited a high success rate in detecting and mitigating various forms of XSS attacks while maintaining compatibility with existing web applications. Furthermore, the paper outlines best practices for developers to integrate the framework into their projects seamlessly. Overall, this framework presents a significant advancement in addressing the persistent threat of DOM-based XSS vulnerabilities, enhancing the security posture of web applications and ensuring a safer browsing experience for users.

## 2.5 PROTOTYPE APPROACH

Against XSS vulnerabilities, understanding the different types of XSS attacks, such as stored, reflected, and DOM-based, is crucial for implementing effective preventive measures. Employing input validation, output encoding, and secure coding practices can significantly mitigate the risk of XSS attacks. Furthermore, the adoption of Content Security Policy (CSP), which dictates the approved sources of content that a browser should execute, serves as an additional layer of defense. Regular security audits, code reviews, and employee training are essential for maintaining a robust security posture against evolving XSS threats.

As web applications continue to play an integral role in everyday activities, a proactive approach to XSS prevention is indispensable for ensuring the integrity and confidentiality of sensitive data.

## 2.6 COOKIE MONITOR APPROACH

The paper introduces XTrap, a novel approach to detect and prevent Cross-Site Scripting (XSS) attacks by intelligently monitoring cookies and sessions using distinct operators (Ankit Shrivatsava et al., 2016). It employs four smart agents—cookie monitor, session monitor, content monitor, and phishing monitor—to effectively scrutinize and counteract abuses on the client side. Additionally, two server-side agents—coordination specialist and data access specialist—are deployed to enhance overall security. The paper focuses on injection, detection, and prevention of stored-based XSS, reflected XSS, and DOM-oriented XSS, offering a comprehensive defense strategy against diverse XSS threats. The approach involves meticulous monitoring and response mechanisms for both client and server sides.

## 2.7 MACHINE LEARNING APPROACH

This study by Gulit Habibi and Nico Surantha (et al., 2020) addresses Cross-Site Scripting (XSS) attacks, a prevalent threat to websites. It evaluates machine learning methods, including Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Naïve Bayes (NB), enhanced with the n-gram technique for script feature analysis. XSS, executed by inserting malicious scripts into websites, exploits vulnerabilities in approximately 68% of websites. The simulation demonstrates that combining SVM with the n-gram method yields the highest accuracy, reaching 98%. The research contributes to advancing XSS attack detection through the integration of machine learning and feature analysis methodologies.

## 2.8 WHITELIST APPROACH

This paper addresses the vulnerability of web applications, specifically Cross-Site Scripting (XSS), focusing on Document Object Model (DOM) Based XSS (Khaled Ali et al., 2014). The proposed solution is an anti-DOM XSS framework, functioning solely on the client side to prevent the execution of malicious scripts embedded in HTML DOM tree sources. The framework introduces a whitelist approach, enhancing security. A prototype tool implementation validates the effectiveness and practicality of the proposed technique. By targeting client-side vulnerabilities, this approach contributes to safeguarding users from DOM XSS attacks, presenting a valuable advancement in web application security.

## 2.9 SCANNING TOOL APPROACH

This paper addresses the challenge of detecting DOM-based Cross-Site Scripting (XSS) vulnerabilities, a type of vulnerability affecting client-side script code execution in web browsers. Unlike traditional XSS vulnerabilities, DOM-based XSS is harder to detect as it resides within website script codes. The authors (Trong Kha Nguyen et al., 2016) introduce a distributed scanning tool designed for large-scale crawling of modern web applications and the detection and validation of DOM-based XSS vulnerabilities. The proposed tool addresses the difficulty of executing script codes without errors and monitoring their execution, making it a valuable contribution to identifying and mitigating this complex type of XSS vulnerability.

## 2.10 HYBRID ANALYSIS APPROACH

This study addresses the rising threat of DOM-sourced Cross-Site Scripting (XSS) in browser extensions, an underexplored vulnerability. With web applications increasingly shifting functionality to the client side, client-side vulnerabilities, especially in browser extensions, become more prevalent. Unlike

conventional DOM-based XSS, DOM-sourced XSS introduces a new attack surface, making the Document Object Model (DOM) itself a potential vulnerable source. The proposed detection framework utilizes hybrid analysis with two phases: lightweight static analysis and dynamic symbolic execution with a component named shadow DOM (Jinkun Pan et al., 2017). In a large-scale real-world experiment, the framework successfully discovered 58 previously unknown DOM-sourced XSS vulnerabilities in user scripts of the popular browser extension Greasemonkey.

## 2.11 SUMMARY OF THE LITERATURE SURVEY

The literature survey encompasses various innovative approaches to address Cross-Site Scripting (XSS) vulnerabilities, particularly focusing on the client-side variant known as DOM-Based XSS. Researchers propose novel frameworks and tools to detect and prevent these vulnerabilities, acknowledging the evolving threat landscape. Techniques include leveraging Content Security Policy (CSP), introducing application-specific solutions like XSnare, utilizing machine learning and n-gram methods for detection, and implementing white-list frameworks for prevention. Some studies concentrate on specific scenarios, such as DOM-sourced XSS in browser extensions, introducing hybrid analysis with static and dynamic components for effective detection. Overall, these contributions collectively contribute to the advancement of strategies to mitigate XSS threats, showcasing a diverse range of methodologies and applications.

# CHAPTER 3
# SYSTEM ARCHITECTURE AND DESIGN

## 3.1 SYSTEM ARCHITECTURE

In the proposed system architecture for web application security as shown in Figure 3.1, the user initiates interaction with the web application by providing input. This input is then subjected to analysis through a set of carefully crafted regular expression patterns, strategically designed to bolster the system's security measures. If the analysis reveals no signs of a potential attack, the user is granted seamless access to the web application.

However, in the event of a detected attack via pattern matching, a dynamic response mechanism is triggered. This involves updating the pattern library promptly to ensure the security system is equipped with the latest threat intelligence for effective mitigation. Following the pattern library update, a tailored Attack Prevention CSP is generated. This CSP serves as a proactive measure against identified threats, providing a robust shield for the web application.

The final step involves seamlessly integrating the generated Attack Prevention CSP into the web page, fortifying its defences against potential security breaches. This dynamic and responsive approach not only enhances the system's resilience but also ensures that the web application remains secure in the face of evolving cyber threats. The architecture prioritizes real-time threat detection, continuous pattern library updates, and the proactive implementation of security measures to safeguard the user experience and protect against potential vulnerabilities. The overall system design aims to strike a balance between user accessibility and robust security, contributing to a resilient and secure web environment.

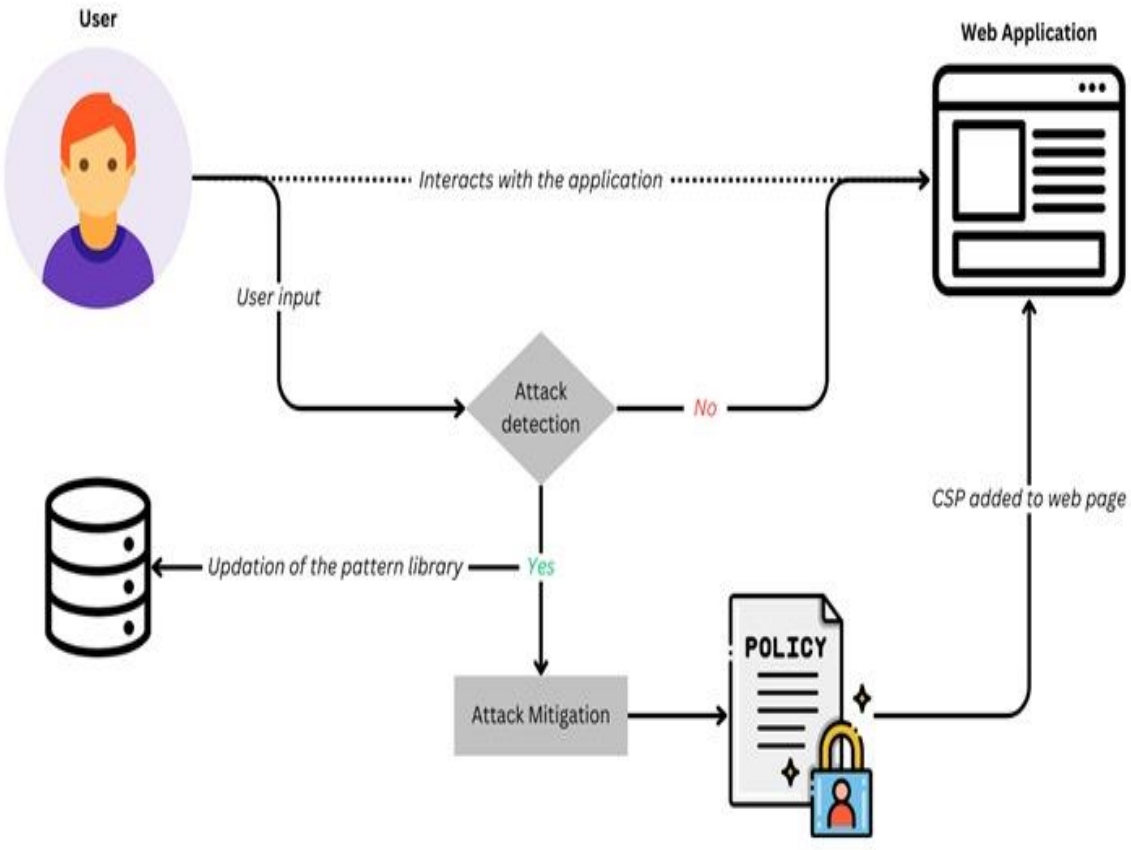


**Figure 3.1** – System Architecture

## 3.2 UNDERSTANDING XSS-DOM

Understanding XSS-DOM vulnerabilities is pivotal in fortifying web applications against potential security threats. XSS poses a significant risk by allowing malicious scripts to be injected into websites, compromising user data and privacy. DOM vulnerabilities exacerbate this risk, as attackers manipulate the structure and content of web pages.

To comprehend these vulnerabilities, one must delve into the intricacies of how attackers exploit scripting languages to execute unauthorized code within a user's browser. The nature of XSS-DOM attacks involves manipulating the DOM to alter page content dynamically. This manipulation can lead to unauthorized access, data

theft, and even the initiation of malicious transactions on behalf of unsuspecting users.

Detection of XSS-DOM vulnerabilities is a multifaceted challenge, often requiring advanced techniques such as pattern matching and heuristic analysis. Proactive defence measures, including input validation and robust CSP play a crucial role in preventing these vulnerabilities from being exploited. Real-world insights into notable attacks provide valuable lessons on the potential consequences and implications of XSS-DOM vulnerabilities, emphasizing the need for constant vigilance and adaptive security measures.

In the evolving landscape of web security, understanding XSS-DOM vulnerabilities becomes a cornerstone for developers, security professionals, and system architects alike. By comprehending the mechanics of these threats, the industry can develop and implement effective strategies to safeguard web applications and user data from the ever-present risk of XSS-DOM attacks.

## 3.3 DETECTION TECHNIQUES FOR XSS-DOM VULNERABILITIES

Detection techniques for XSS-DOM vulnerabilities involve a meticulous exploration of methods aimed at identifying and mitigating potential threats. One key approach is pattern matching, a method that enables the systematic identification of patterns associated with malicious scripts or manipulations within the Document Object Model (DOM). By employing pattern matching, security systems can proactively defend against XSS-DOM attacks by recognizing and intercepting malicious code patterns before they can compromise the web application.

Heuristic analysis is another powerful detection technique that involves the application of intelligent algorithms to identify abnormal patterns or behaviours indicative of XSS-DOM vulnerabilities. This method allows for the detection of previously unknown threats by analysing the dynamic behaviour of web content and user interactions.

Additionally, static code analysis can be employed to scrutinize the source code of web applications for potential vulnerabilities. This method involves a comprehensive examination of the codebase to identify and address any potential weaknesses that could be exploited by XSS-DOM attacks.

Regular security audits and penetration testing also play a crucial role in detecting XSS-DOM vulnerabilities. By systematically testing the web application for known attack vectors and employing simulated attacks, security professionals can uncover potential vulnerabilities and address them before they can be exploited maliciously.

Ultimately, a comprehensive approach to detection involves the integration of multiple techniques, including pattern matching, heuristic analysis, static code analysis, and regular security testing. This multi-faceted strategy enhances the web application's resilience against XSS-DOM vulnerabilities, providing a robust defense mechanism in the dynamic landscape of web security.

### 3.4 PREVENTIVE STRATEGIES AGAINST XSS-DOM ATTACKS

Preventive strategies against XSS-DOM attacks necessitate a proactive and multi-layered approach to fortify web applications against potential threats. One fundamental strategy is the implementation of rigorous input validation mechanisms, ensuring that user inputs are thoroughly examined and sanitized before being processed by the application. By validating and filtering user inputs, developers can mitigate the risk of malicious scripts being injected and executed within the application.

CSP serves as a pivotal defence mechanism against XSS-DOM attacks. By defining and enforcing a robust CSP, developers can specify the trusted sources for content and script execution, limiting the potential for unauthorized script injections. This strategy adds an additional layer of protection by explicitly defining the allowable domains and sources for loading scripts and resources.

WAFs are another critical component of preventive strategies. WAFs act as a protective barrier between the web application and the internet, filtering and monitoring HTTP traffic to detect and block malicious requests, including those indicative of XSS-DOM attacks. Regularly updating and configuring the WAF to recognize emerging threats enhances its efficacy in preventing potential attacks.

Additionally, employing secure coding practices, such as output encoding, ensures that user-generated content is appropriately encoded before being rendered in the browser. This minimizes the risk of attackers manipulating the DOM through injected scripts.

Educating developers and users on security best practices is integral to a comprehensive preventive strategy. By fostering a security-aware culture, organizations can empower both the development team and end-users to recognize and address potential vulnerabilities, contributing to a more resilient defence against XSS-DOM attacks.

### 3.5 DYNAMIC RESPONSE MECHANISM FOR PATTERN MATCHING

The dynamic response mechanism for pattern matching is a pivotal component in the arsenal of web security, offering an adaptive shield against potential threats. When a threat is detected through pattern matching algorithms, the system triggers dynamic responses to counteract and neutralize the identified attack vectors. This responsiveness involves swift actions, such as updating the pattern library in real-time, ensuring that the security infrastructure is armed with the latest threat intelligence.

Beyond pattern library updates, the dynamic response mechanism extends to the creation of a tailored Attack Prevention Content Security Policy (CSP). Crafted based on the specifics of the detected threat patterns, this CSP serves as a proactive measure against identified risks, adding an additional layer of defense to the web application.

The dynamic response doesn't stop there—it seamlessly integrates the generated Attack Prevention CSP into the web page, fortifying the application's defenses against potential security breaches. This adaptive approach not only addresses current threats but also lays the groundwork for anticipating and mitigating future security challenges, contributing to a resilient and secure web environment.

In summary, the dynamic response mechanism for pattern matching ensures that the security posture of the web application is not only reactive but also predictive, adapting in real-time to evolving cyber threats for comprehensive protection.

## 3.6 BALANCING ACCESSIBLITY AND SECURITY

Balancing accessibility and security is a delicate equilibrium that demands meticulous attention to both user experience and robust protective measures. Striking this balance is crucial to achieve a resilient, secure web environment that not only safeguards against potential threats but also ensures a seamless and user-friendly interaction.

This equilibrium involves implementing security measures that don't compromise the accessibility and functionality of the web application. User-friendly authentication methods, such as multi-factor authentication, can enhance security without burdening users with overly complex processes. Similarly, encryption protocols, when thoughtfully implemented, bolster security without sacrificing performance.

Accessibility considerations extend to users with diverse needs and preferences. Implementing clear and concise security communication, offering alternative authentication methods, and ensuring compatibility with assistive technologies are integral aspects of maintaining an inclusive and accessible digital environment.

Additionally, a well-designed user interface can contribute to a positive user experience without compromising security. Striking a balance in the design ensures that security features seamlessly integrate into the overall user journey, minimizing friction while upholding protection.

Regular user education and awareness programs play a vital role in this delicate equilibrium. By keeping users informed about security best practices, potential threats, and the importance of their role in maintaining a secure environment, organizations foster a shared responsibility for security without sacrificing user-friendliness.

In essence, achieving a balance between accessibility and security involves thoughtful design, user education, and the implementation of security measures that enhance rather than hinder the user experience. This approach contributes to a web environment where both accessibility and security are mutually reinforcing elements, creating a harmonious and protected digital space.

# CHAPTER 4
# ALGORITHM DEVELOPMENT AND IMPLEMENTATION

## 4.1 PATTERN MATCHING TECHNIQUES

Pattern matching techniques play a crucial role in identifying and mitigating XSS-DOM vulnerabilities within web applications. These techniques involve the systematic examination of input data, code, or patterns that may indicate potential threats associated with XSS and DOM attacks.

In the context of XSS-DOM vulnerabilities, pattern matching helps detect specific sequences or structures indicative of malicious script injections and manipulations within the DOM. By implementing sophisticated pattern matching algorithms, security systems can recognize and intercept patterns commonly associated with XSS-DOM attacks, allowing for proactive defence measures. These pattern matching techniques enable the identification of known attack vectors and aid in the prevention of unauthorized script executions. Regular updates to the pattern library ensure that the security system remains resilient against evolving XSS-DOM threats, contributing to a robust defence mechanism that safeguards web applications from potential vulnerabilities.

## 4.2 ALGORITHM

Input:

- User input from various sources in the web application.

Output:

- Detection of potential malicious scripts.
- Application of Content Security Policy for mitigation.

Steps:

1. Initialization:
   - Initialize the dynamic pattern library with predefined patterns.
2. User Input Processing:
   - Receive user input from different sources in the web application.
3. Dynamic Pattern Matching:
   - Iterate through the dynamic pattern library.
   - Check if the user input matches any predefined patterns.
   - If a match is found:
     - Log the potential malicious script.
     - Optionally, take further actions (e.g., alerting, blocking).
4. Taint Analysis
   - Optionally perform taint analysis on user input to detect potential vulnerabilities.
   - Identify and log any tainted input that may lead to script execution.
5. Dynamic Pattern Update:
   - Check if the user input is not in the dynamic pattern library.
   - If not:
     - Add the new input to the dynamic pattern library.
   - Log the addition of the new pattern.
6. Content Security Policy Implementation:
   - Apply the Content Security Policy rules to the web application.
   - Restrict the execution of JavaScript and CSS based on the defined policy.
   - Allowlist external scripts and stylesheets from trusted sources.

## 4.3 DYNAMIC PATTERN UPDATION

Our innovative Dynamic Pattern Updating algorithm represents a cutting-edge approach in web security, evolving in tandem with the application's

behaviour. Utilizing advanced taint analysis, the algorithm meticulously tracks the flow of data, enabling a comprehensive assessment of inputs for suspicious keywords and contextual cues. This dynamic nature ensures the algorithm stays attuned to the changing threat landscape, learning from real-time application interactions. By adapting to the evolving patterns of data flow and recognizing potential vulnerabilities through taint analysis, our algorithm fortifies the system against emerging security threats, particularly in the context of XSS-DOM attacks. This responsive mechanism, grounded in taint analysis, contributes to a proactive defence strategy, allowing the system to autonomously update its pattern library based on observed behaviours, ultimately enhancing the security posture of web applications.

## 4.4 RISK ASSESSMENT

Risk assessment in our system involves assigning scores to inputs, and those surpassing a predefined threshold are identified as potential new attack patterns. This approach enables a nuanced understanding of the security landscape, allowing us to quantify the risk associated with various input sources. By setting a threshold, the system can efficiently differentiate between normal and potentially harmful inputs, prioritizing proactive measures against high-risk patterns. This risk-based strategy enhances our ability to pre-emptively address emerging threats, contributing to a more robust and adaptive security framework. Regularly adjusting the threshold based on evolving threat scenarios ensures the system's agility in identifying and mitigating novel attack patterns, reinforcing its efficacy in safeguarding against potential vulnerabilities.

## 4.5 PATTERN ADDITION

The Pattern Addition mechanism is a pivotal component in our defense strategy against XSS-DOM attacks. This dynamic approach involves integrating newly identified attack patterns into the system's pattern library in real-time, ensuring that security measures remain current and responsive to emerging

threats. This continuous vigilance allows the system to adapt swiftly to novel XSS-DOM attack techniques, taking a proactive stance by anticipating potential vulnerabilities before they can be exploited.

The scalability of this mechanism is noteworthy, accommodating an expanding array of attack patterns and ensuring that the system remains resilient against a growing spectrum of XSS-DOM threats. As the threat landscape evolves, the Pattern Addition feature keeps the system adaptable to emerging trends and variations in XSS-DOM attack methods, reducing response times and minimizing the window of vulnerability.

This real-time pattern addition enhances the precision of threat detection, enabling the system to discern even subtle variations in XSS-DOM attack patterns for more accurate identification. The system's agility in pattern addition facilitates agile mitigation, ensuring it can effectively counteract the latest tactics employed by attackers targeting XSS-DOM vulnerabilities.

In summary, the Pattern Addition mechanism is instrumental in maintaining a proactive and adaptive defence against XSS-DOM attacks, providing a robust shield for web applications in the face of evolving security challenges.

## 4.6 PROPOSED SYSTEM IMPLEMENTATION

The proposed system architecture for web application security adopts a user-centric approach, initiating interaction when the user provides input to the web application. This user input undergoes thorough analysis through a set of meticulously crafted regular expression patterns, strategically designed to fortify the system's security measures. If the analysis indicates no signs of a potential attack, the user seamlessly gains access to the web application, ensuring a smooth user experience.

### 4.6.1 Detection

In this module, we have implemented a set of regular expression patterns designed to enhance the security system's ability to identify potential XSS-DOM vulnerabilities.

This pattern integration allows for more robust detection of security threats related to client-side scripting and DOM manipulation

### 4.6.2 Mitigation

Under this module, we explore the integration of CSP as a proactive measure to mitigate the risks associated with client-side scripting and DOM manipulation. We employ a specific whitelist framework to ensure no scripts or resources are loaded from a source that we are unaware of. Additionally, we receive alerts when such an attack is detected and blocked.

These alerts include detailed information about the detected threats and recommended actions for effective incident response. This combined approach ensures that not only are vulnerabilities identified, but also steps are taken to mitigate and respond to potential threats effectively.

# CHAPTER 5
# RESULTS AND DISCUSSIONS

## 5.1 IMPLEMENTATION ENVIRONMENT

The implementation environment for our web application security system utilizes a versatile tech stack to ensure platform independence and seamless functionality. The primary programming language employed is Python, leveraging its versatility and extensive libraries. The web framework Flask is utilized to facilitate rapid and efficient development, providing a robust foundation for handling HTTP requests and responses.

JavaScript, a fundamental scripting language for web development, plays a pivotal role in the implementation, enhancing interactivity and enabling dynamic content updates. Sqlite is chosen as the database engine, offering a lightweight and self-contained solution for efficient data management within the system.

For the development environment, Windows 10 and above are recommended, ensuring compatibility and optimal performance. Visual Studio Code, a lightweight and feature-rich code editor, serves as the preferred integrated development environment providing a streamlined coding experience.

In terms of system requirements, the web application is designed to run seamlessly on Windows 10 and above operating systems. Chrome browser compatibility is emphasized for optimal user experience. The chosen technology stack, comprising Python, Flask, JavaScript, and Sqlite, aligns with industry standards for web development, ensuring a robust and scalable foundation for our security-focused web application.

This implementation environment adheres to the principles of simplicity, flexibility, and compatibility, aiming to strike a balance between effective security measures and a user-friendly development experience.

## 5.2 INJECTING SCRIPT IN VARIOUS WAYS

The injection of malicious scripts poses a serious threat to web applications, and various avenues for exploitation were identified in our security assessment. In one instance, a potential vulnerability was discovered on an online shopping website's catalogue page. The search bar, designed to facilitate user-friendly searches by category, became a potential vector for exploitation. Through this interface, a malicious code was injected, exemplified by the script <script>alert(document.cookie)</script>. This injection aimed to trigger a pop-up alert, revealing the user's cookie information, thereby demonstrating the potential for unauthorized access to sensitive data. This was shown in Figure 5.1

Furthermore, a notes page within the application became another avenue for malicious script injection as shown in Figure 5.2. This underscores the importance of scrutinizing all user inputs, as even seemingly innocuous sections can become vectors for attacks. Additionally, the third avenue of attack highlighted the susceptibility of the web application to URL-based injection as shown in Figure 5.3. Attackers could potentially manipulate URLs to inject malicious code, emphasizing the need for robust input validation and secure coding practices to mitigate such threats. These findings underscore the significance of implementing a comprehensive security strategy, including regular audits and stringent input validation, to fortify web applications against diverse injection attacks.

Scripts have been injected in the following ways:

a. Injection via catalogue page – script embedded inside of product description.
b. Injection via notes page – script embedded inside of a particular note.
c. Injection via url – script embedded in url while trying to query for a particular product.

## 5.3 DETECTION OF XSS-DOM ATTACK

Detection of XSS-DOM attacks through pattern matching is a crucial component of our web application security system. By employing carefully crafted regular expression patterns, the system systematically analyses user inputs for any indications of potential XSS and DOM manipulation threats. These patterns are strategically designed to recognize known attack vectors and suspicious sequences within the input data that could be indicative of malicious scripts attempting to compromise the security of the web application. The detection of XSS-DOM is shown in the Figure 5.4

In the event of a detected attack, the system triggers an alert mechanism as shown in Figure 5.6. This is in response to an update to the pattern library to ensure that the security system is equipped with the latest threat intelligence for effective mitigation. Following the pattern library update, a tailored Attack Prevention CSP is generated, serving as a proactive measure against identified threats. This comprehensive approach to detection and response through pattern matching enhances the system's resilience, providing real-time protection against evolving XSS-DOM attack vectors.

### 5.3.1 Pattern updation

In the dynamic pattern updation process, the system employs an iterative approach to enhance its defence against potential malicious scripts. This involves continuous refinement of the pattern library, a repository of regular expressions used to identify and thwart suspicious input patterns. The system provides an administrative interface, accessible through an admin page, allowing authorized personnel to contribute to the pattern library by specifying keywords anticipated in malicious scripts as shown in Figure 5.5. The entered keywords undergo transformation into regular expressions, serving as dynamic patterns that adapt to emerging threats. This proactive strategy empowers the system to fortify its security measures by systematically integrating novel patterns, ensuring an agile

response to evolving attack vectors. This iterative pattern updation mechanism plays a pivotal role in the system's ability to detect and mitigate sophisticated script-based attacks effectively, contributing to an adaptive and resilient security posture.

## 5.4 MITIGATION OF XSS-DOM ATTACK VIA CSP

Mitigating risks associated with client-side scripting and DOM manipulation in our web application security system is achieved through the integration of a robust Content Security Policy as shown in Figure 5.7. CSP serves as a proactive measure to control the behaviour of web page content, significantly reducing the risk of XSS and DOM-based attacks. A specific whitelist framework is employed within the CSP, meticulously defining trusted sources from which scripts and resources can be loaded. This ensures that only authorized and known entities are allowed to execute scripts, mitigating the potential threat of malicious code injection from unknown sources.

In the event of an attack, the CSP framework not only prevents the execution of unauthorized scripts but also triggers alerts. These alerts provide detailed information about the detected threats, offering insights into the nature and specifics of the attack. This immediate notification allows for swift incident response, empowering administrators to take recommended actions to thwart the attack effectively. The combined approach of CSP implementation, whitelist enforcement, and alerting mechanisms ensures a comprehensive security strategy. Not only does it identify vulnerabilities, but it also takes proactive steps to mitigate and respond to potential threats, creating a resilient defence against client-side scripting and DOM manipulation attacks.

The dynamic pattern updation mechanism in our web application security system plays a pivotal role in enhancing its adaptive defences. This innovative approach involves the continuous learning and evolution of the system's pattern library based on the application's behaviour, employing a technique known as

taint analysis. Taint analysis is utilized to track the flow of data, assessing inputs for suspicious keywords and contextual cues indicative of potential security threats, particularly XSS-DOM attacks.

When an attack is detected, the dynamic pattern updation mechanism is promptly triggered. It involves the real-time update of the pattern library to ensure that the security system is armed with the latest threat intelligence. This adaptive response allows the system to autonomously recognize and adapt to emerging attack patterns, fortifying its resilience against evolving cyber threats. The dynamic pattern updation approach not only identifies vulnerabilities but actively contributes to a proactive defence strategy, making continuous improvements to the pattern library for robust and adaptive security.

## 5.5 EVALUATION METRICS

List of type of scripts detected

- <script>alert('XSS')</script>
- <img src="x" onerror="alert('XSS')">
- <a href="javascript:alert('XSS')">Click me</a>
- <input type="text" value="x" onfocus="alert('XSS')">
- <svg onload="alert('XSS')"></svg>
- Hover me
- <iframe src="javascript:alert('XSS')"></iframe>

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

## 6.1 CONCLUSION

In conclusion, this web application security project serves as a comprehensive and dynamic defence system against a spectrum of potential threats, especially focusing on the prevalent risks associated with XSS and DOM manipulation attacks. The implementation of carefully crafted regular expression patterns and heuristic analysis ensures a proactive stance in identifying vulnerabilities, showcasing a commitment to real-time threat detection. The incorporation of a robust CSP serves as a significant mitigation strategy, adding an additional layer of protection against client-side scripting and DOM manipulation. The dynamic pattern updation mechanism enhances the system's adaptability, allowing it to evolve and respond to emerging attack patterns, a critical feature in the ever-changing landscape of cyber threats. By fortifying the system's defence through continuous learning, responsive pattern updates, and proactive measures, this project contributes to creating a resilient and secure web environment.

Furthermore, the identified vulnerabilities, such as script injections through search bars, notes pages, and URL attacks on an online shopping website, highlight the necessity for heightened vigilance and a layered defence approach. These findings underscore the importance of ongoing security assessments, rigorous input validation, and user-awareness programs to bolster the resilience of web applications against diverse injection attacks. The insights gained from this project emphasize the need for a holistic security mindset, integrating both proactive and reactive measures, to safeguard against potential threats and foster a secure digital landscape for users and businesses alike.

## 6.2 FUTURE WORK

In the present research, we have successfully developed a comprehensive web application security framework that combines dynamic pattern matching and CSP implementation to detect and mitigate potential malicious scripts. The system has proven effective in identifying predefined patterns and adapting to emerging threats through dynamic pattern updates. To further enhance the capabilities of our security framework, future work could delve into the integration of ML techniques, particularly supervised learning algorithms, for the classification of malicious inputs. This approach could leverage labelled datasets to train models that learn intricate patterns indicative of security threats, providing a more adaptive and accurate detection mechanism. Additionally, feature engineering efforts could focus on extracting key characteristics from user inputs to improve the efficiency of ML models. Exploring unsupervised learning, such as anomaly detection, presents an opportunity to identify novel threats without relying on predefined patterns. Collaborating with threat intelligence feeds to incorporate the latest information on emerging threats could provide a proactive defence mechanism. Furthermore, user interaction and feedback mechanisms, including user-friendly interfaces and educational components, could empower end-users to actively contribute to the security of the application. This comprehensive approach aims to create a robust, adaptive, and user-centric security framework capable of addressing the evolving landscape of web-based security threats across diverse domains and functionalities.

# REFERENCES


1. A. F. Maskur and Y. Dwi Wardhana Asnar, "Static Code Analysis Tools with the Taint Analysis Method for Detecting Web Application Vulnerability," 2019 International Conference on Data and Software Engineering (ICoDSE), Pontianak, Indonesia, 2019, pp. 1-6, doi: 10.1109/ICoDSE48700.2019.9092614

2. A. Shrivastava, S. Choudhary and A. Kumar, "XSS vulnerability assessment and prevention in web application," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 850-853, doi: 10.1109/NGCT.2016.7877529.

3. A. Shrivastava, V. K. Verma and V. G. Shankar, "XTrap: Trapping client and server side XSS vulnerability," 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2016, pp. 394-398, doi: 10.1109/PDGC.2016.7913227.

4. G. Habibi and N. Surantha, "XSS Attack Detection With Machine Learning and n-Gram Methods," 2020 International Conference on Information Management and Technology (ICIMTech), Bandung, Indonesia, 2020, pp. 516-520, doi:10.1109/ICIMTech50083.2020.9210946.

5. G. Xu et al., "JSCSP: A Novel Policy-Based XSS Defense Mechanism for Browsers," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 2, pp. 862-878, 1 March-April 2022, doi: 10.1109/TDSC.2020.3009472.

6. I. Dolnák, "Content Security Policy (CSP) as countermeasure to Cross Site Scripting (XSS) attacks," 2017 15th International Conference on Emerging

eLearning Technologies and Applications (ICETA), Stary Smokovec, Slovakia, 2017, pp. 1-4, doi: 10.1109/ICETA.2017.8102476.

7. J. C. Pazos, J. -S. Légaré and I. Beschastnikh, "XSnare: Applicationspecific client-side cross-site scripting protection," 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 2021, pp. 154-165, doi: 10.1109/SANER50967.2021.00023.

8. J. Harish Kumar and J. J Godwin Ponsam, "Cross Site Scripting (XSS) vulnerability detection using Machine Learning and Statistical Analysis," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-9, doi: 10.1109/ICCCI56745.2023.10128470.

9. J. Pan and X. Mao, "Detecting DOM-Sourced Cross-Site Scripting in Browser Extensions," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 24-34, doi: 10.1109/ICSME.2017.11.

10. J. Pan and X. Mao, "DomXssMicro: A Micro Benchmark for Evaluating DOM-Based Cross-Site Scripting Detection," 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 2016, pp. 208-215, doi: 10.1109/TrustCom.2016.0065.

11. K. Ali, A. Abdel-Hamid and M. Kholief, "Prevention Of DOM Based XSS Attacks Using A White List Framework," 2014 24th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, 2014, pp. 68-75, doi: 10.1109/ICCTA35431.2014.9521633.

12. K. Santithanmanan, "The Detection Method for XSS Attacks on NFV by Using Machine Learning Models," 2022 International Conference on Decision Aid Sciences and Applications (DASA), Chiangrai, Thailand, 2022, pp. 620-623, doi: 10.1109/DASA54658.2022.9765122.

13. M. Obaidat, J. Brown and A. A. Hayajneh, "Web Browser Extension UserScript XSS Vulnerabilities," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Calgary, AB, Canada, 2020, pp. 316-321, doi: 10.1109/DASC-PICom-CBDComCyberSciTech49142.2020.00062.

14. P. Wang, B. Á. Guðmundsson and K. Kotowicz, "Adopting Trusted Types in ProductionWeb Frameworks to Prevent DOM-Based Cross-Site Scripting: A Case Study," 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Vienna, Austria, 2021, pp. 60-73, doi: 10.1109/EuroSPW54576.2021.00013.

15. P. Wang, J. Bangert and C. Kern, "If It's Not Secure, It Should Not Compile: Preventing DOM-Based XSS in Large-Scale Web Development with API Hardening," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, ES, 2021, pp. 1360-1372, doi: 10.1109/ICSE43902.2021.00123.

16. Samer Attallah Mhana, Jamilah Binti Din and Rodziah Binti Atan, "Automatic generation of Content Security Policy to mitigate cross site scripting," 2016 2nd International Conference on Science in Information

Technology (ICSITech), Balikpapan, Indonesia, 2016, pp. 324-328, doi: 10.1109/ICSITech.2016.7852656.

17. S. K. Mahmoud, M. Alfonse, M. I. Roushdy and A. -B. M. Salem, "A comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques," 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, 2017, pp. 36-42, doi: 10.1109/INTELCIS.2017.8260024.

18. T. K. Nguyen and S. O. Hwang, "Large-Scale Detection of DOM-Based XSS Based on Publisher and Subscriber Model," 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2016, pp. 975-980, doi: 10.1109/CSCI.2016.0187.

19. T. Takeuchi, K. Mouri and S. Saito, "Mocha: Automatically Applying Content Security Policy to HTML Hybrid Application on Android Device," 2017 Fifth International Symposium on Computing and Networking (CANDAR), Aomori, Japan, 2017, pp. 503-509, doi: 10.1109/CANDAR.2017.48.

20. T. Wang, D. Zhao and J. Qi, "Research on Cross-site Scripting Vulnerability of XSS Based on International Student Website," 2022 International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi'an, China, 2022, pp. 154-158, doi: 10.1109/ICCNEA57056.2022.00043.

21. V. K. Malviya, S. Saurav and A. Gupta, "On Security Issues in Web Applications through Cross Site Scripting (XSS)," 2013 20th Asia Pacific

Software Engineering Conference (APSEC), Bangkok, Thailand, 2013, pp. 583-588, doi: 10.1109/APSEC.2013.85.

22. Z. Jingyu, H. Hongchao, H. Shumin and L. Huanruo, "A XSS Attack Detection Method Based on Subsequence Matching Algorithm," 2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID), Guangzhou, China, 2021, pp. 83-86, doi: 10.1109/AIID51893.2021.9456515.

# APPENDIX

*Different ways to inject scripts*

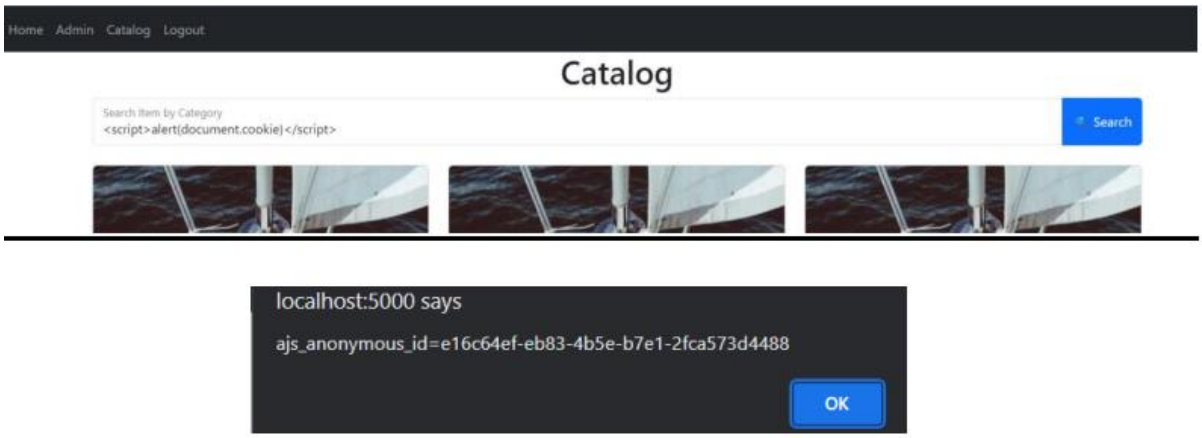


**Figure 5.1** – Via Catalogue Page

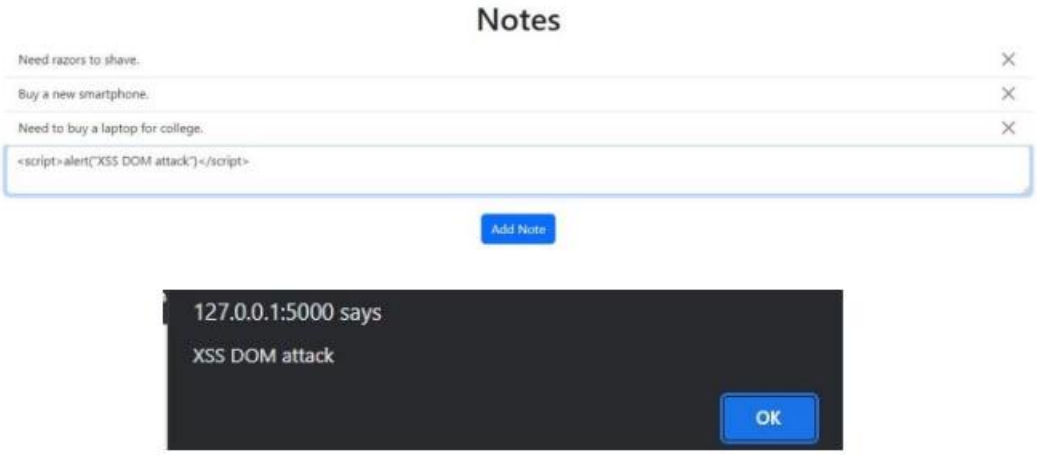


**Figure 5.2** – Via Notes page

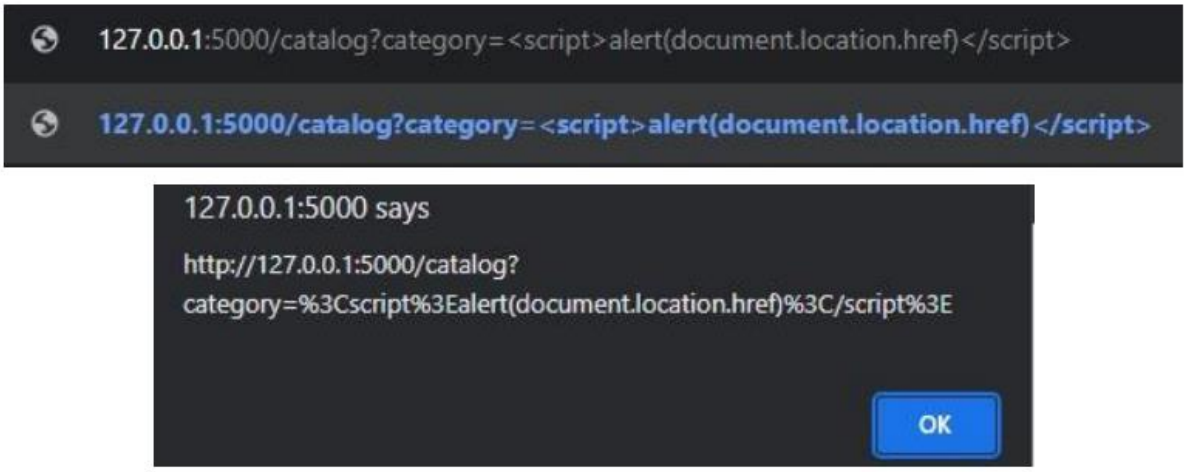


**Figure 5.3** – Via URL

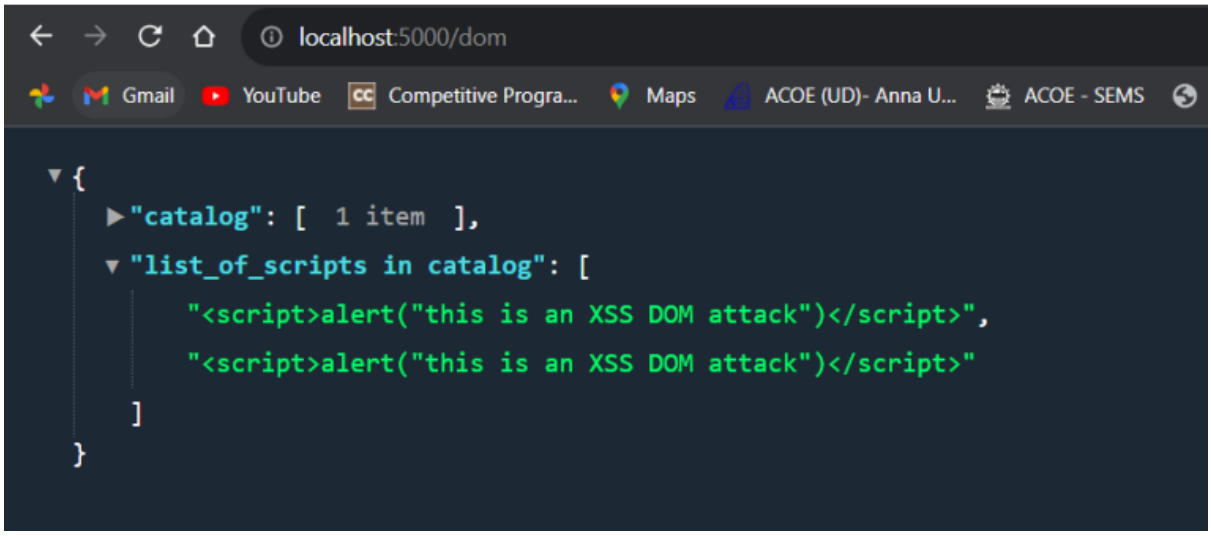


**Figure 5.4** – Pattern Matching

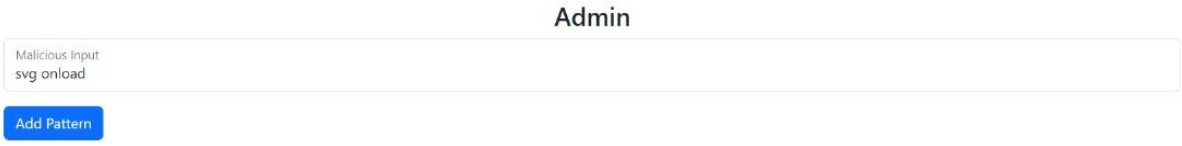


**Figure 5.5** – Pattern Updation

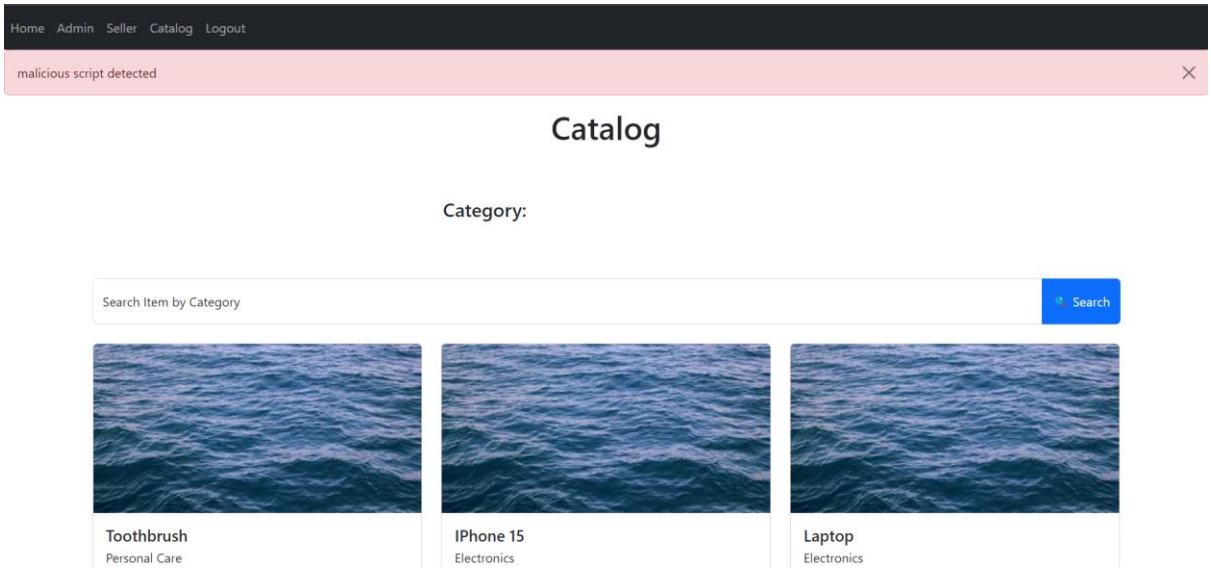


**Figure 5.6** - Detection of script (alert)


**Figure 5.7** – Warning on prevention of XSS DOM Attack