A Paradigm Shift in XSS-DOM Mitigation via Dynamic Content Security Policy

Under the guidance of
Dr. P. Kola Sujatha

Krishnaa S (2020506045)
Jawahar A S (2020506035)
Thamizharasi M (2020506102)

PROBLEM STATEMENT

Cross-Site Scripting (XSS) vulnerabilities persist as a **major threat** in web applications, jeopardizing **user data** and **privacy**. This highlights the urgent need for innovative and adaptive security measures to enhance protection for evolving web applications.

OBJECTIVES

- **Understanding Vulnerabilities:** Gain insight into the nature and mechanics of Cross-Site Scripting (XSS) and DOM-based attacks.
- **Detection Techniques:** Investigate methods (Pattern matching) for identifying XSS-DOM vulnerabilities, enabling proactive defense against potential attacks.
- **Preventive Strategies:** Explore strategies such as input validation and Content Security Policy (CSP) to mitigate the risks of XSS-DOM attacks.
- **Real-world Insights:** Analyze real-world cases of notable attacks to grasp the potential consequences and implications of these vulnerabilities.

LITERATURE SURVEY

SNo	Description	Pros and Cons
1	**JSCSP: A Novel Policy-Based XSS Defense Mechanism for Browsers** * JavaScript based Content Security Policy (JSCSP) to mitigate XSS attacks. * Offers efficient algorithm to automatically generate the policy directives. * Implemented on a Chrome extension and delivers better performance compared to other XSS defense solutions.	**Advantages:** * Scalability: JSCSP is able to support for most browsers. * Automated generation of CSP. **Disadvantages:** * No detection of attack. * Only works for static pages.
2	**Prevention Of DOM Based XSS Attacks Using A White List Framework** * proposes an anti-DOM XSS framework designed to protect clients by blocking malicious scripts in the HTML DOM tree source. * Effectively prevents DOM XSS attacks, and a prototype tool has been developed to validate its effectiveness.	**Advantages:** * White list frameworks have a lower false positive rate * Provide granular control over the sources and types of input that are allowed, allowing for precise mitigation of specific attack vectors. **Disadvantages:** * Maintaining a white list can be challenging and time-consuming. * They only allow what is explicitly permitted on the list, which can limit the flexibility of web applications.

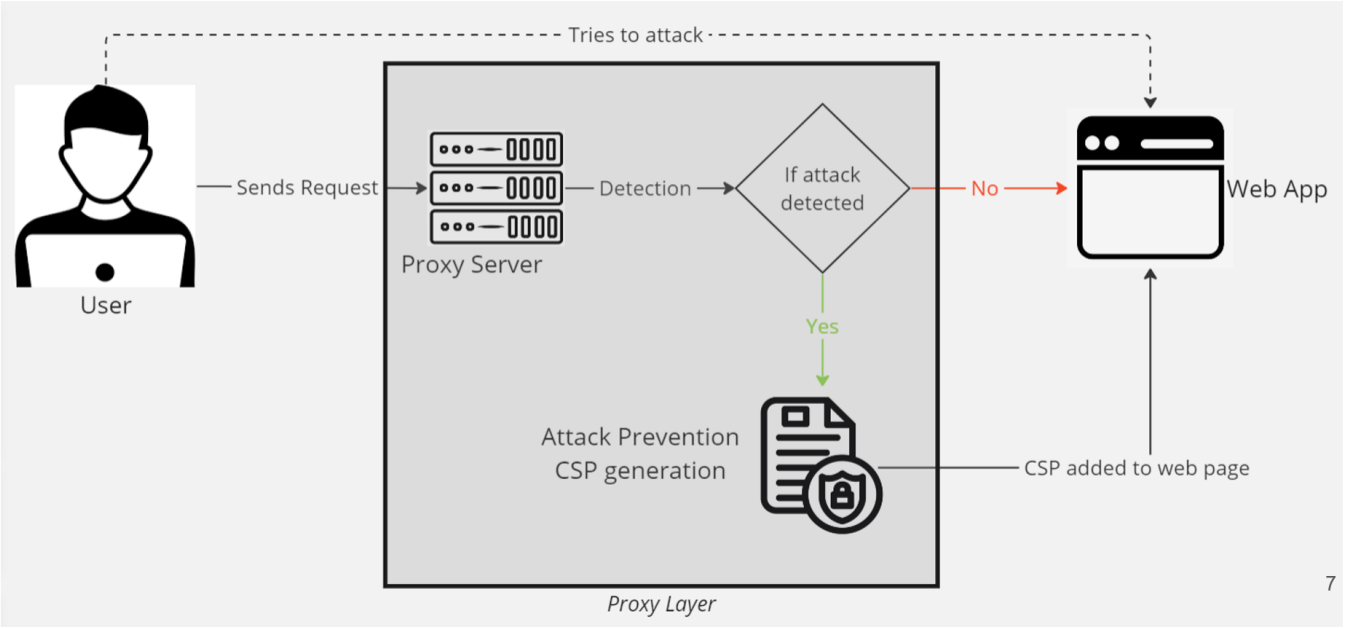
LITERATURE SURVEY

SNo	Description	Pros and Cons
3	**XSnare: Application-specific client-side cross-site scripting protection** * Firefox extension that offers client-side protection against XSS attacks. * Preemptively blocks XSS attacks by leveraging prior knowledge of web app's HTML templates and rich DOM context. * Utilizes an exploit database, crafted from recorded CVEs.	**Advantages:** * XSnare is designed to be application-specific. * Offers preemptive protection to users. **Disadvantages:** * Maintaining the exploit database with up-to-date CVE information * XSnare is implemented as a Firefox extension, hence it may not be compatible with other web browsers.
4	**A XSS Attack Detection Method Based on Subsequence Matching Algorithm** * Detection technique using a subsequence matching algorithm (b/w user input and generated data). * sets a threshold to limit the length of the common subsequence and blocks XSS attacks if the threshold is exceeded	**Advantages:** * Proposes a new method for detecting XSS vulnerabilities using a subsequence matching algorithm. * Incorporates a threshold to limit the length of common substrings. **Disadvantages:** * The proposed method may not be scalable for large-scale web applications. * may not be compatible with all web application, frameworks and technologies.

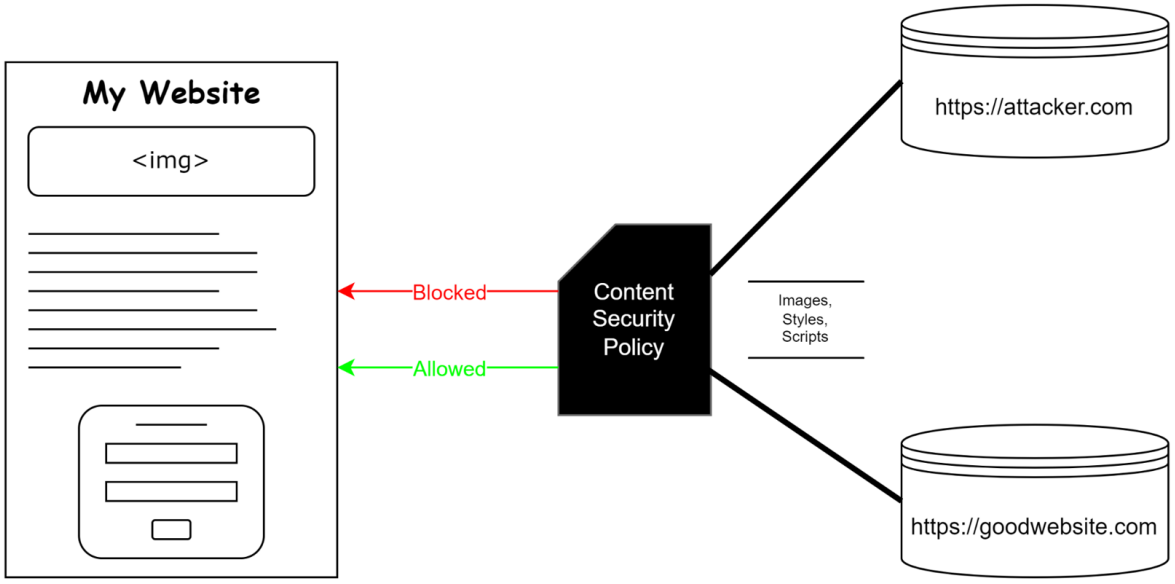
LITERATURE SURVEY

SNo	Description	Pros and Cons
5	**A comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques** * This paper highlights the critical threat of XSS attacks, which can compromise web application security by injecting malicious JavaScript code into either the client-side or server-side. * The study explores XSS attack taxonomy, incidence, and mechanisms for detection and prevention, emphasizing the imperative need for safeguarding against this vulnerability.	**Advantages:** * The hybrid analysis approach in this framework combines static analysis and dynamic symbolic execution, providing a more precise identification of DOM-sourced XSS vulnerabilities. * Incorporating shadow DOM in the dynamic analysis phase enhances the framework's accuracy. **Disadvantages:** * Implementing a framework with multiple phases of analysis, including static and dynamic components, demands a complex setup. * The dynamic symbolic execution phase, particularly when using shadow DOM, can be resource-intensive.

ARCHITECTURE DIAGRAM



ARCHITECTURE DIAGRAM



NOVELTY

- Traditionally XSS DOM attacks are detected using **manual techniques**.
- source code of the application is inspected for certain sources and sinks where there is a possibility of an XSS DOM attack.

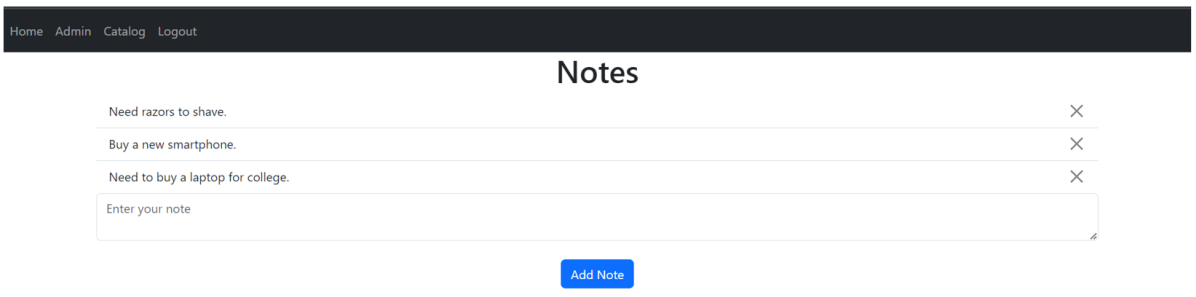
- We have implemented **pattern matching techniques** in our application to detect the vulnerable scripts that have been injected.

- We also propose a **dynamic pattern updation** algorithm where we perform taint analysis to detect a vulnerable input and update it to our pattern library.

IMPLEMENTATION

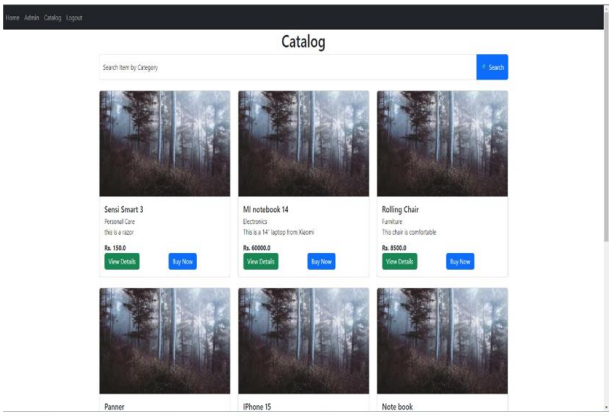
Created a E-Commerce Website to test XSS-DOM Web attack

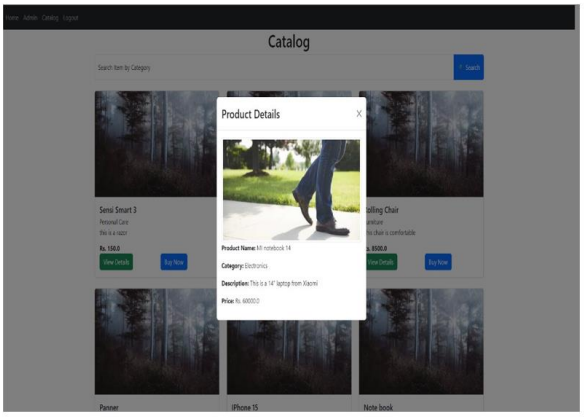
- In this page we can add notes like shopping list in Amazon



IMPLEMENTATION

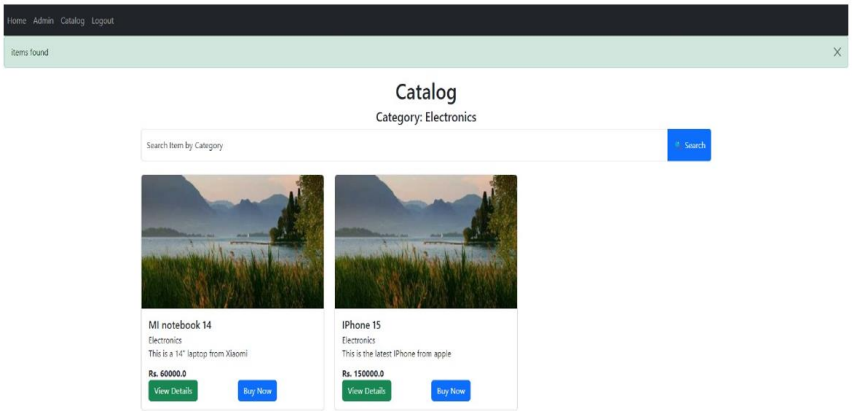
- And we created few more pages with feed option where we can inject the script code to attack the page.





IMPLEMENTATION

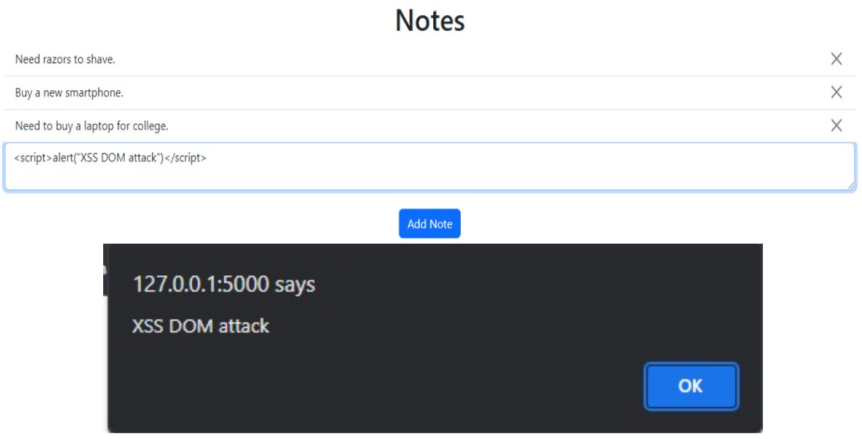
- Search by category page



IMPLEMENTATION - DETECTION

Detection of XSS – DOM attack

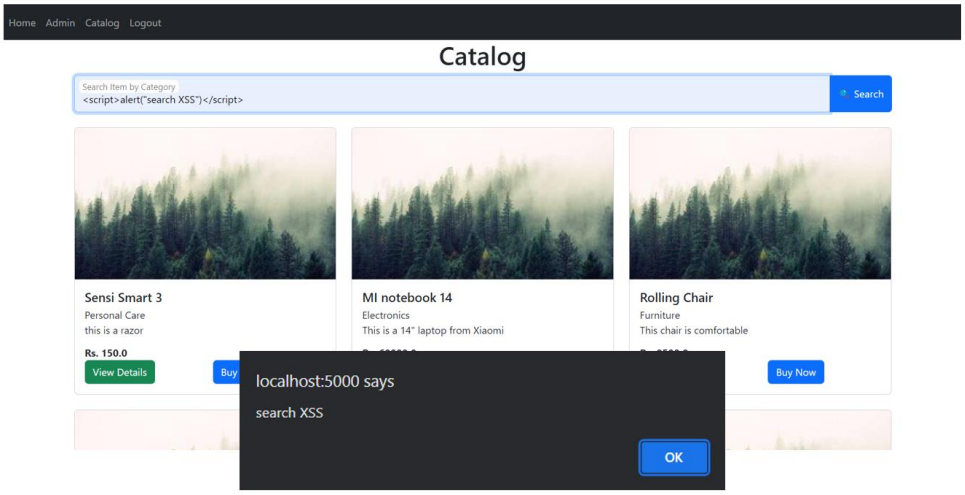
- In notes page, by feeding the script through the input bar.



IMPLEMENTATION

Detection of XSS – DOM attack

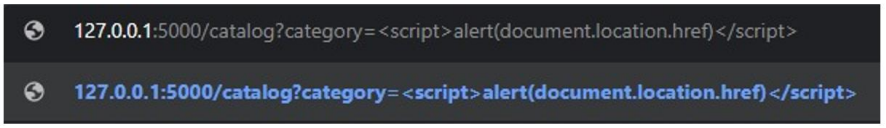
- In User page , by feeding the script through the category search bar. So that will lead to the missing information in the result page.

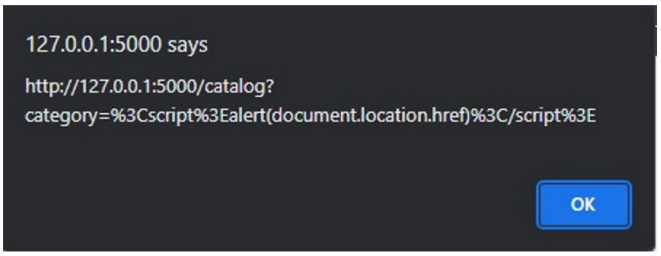


IMPLEMENTATION

Detection of XSS – DOM attack

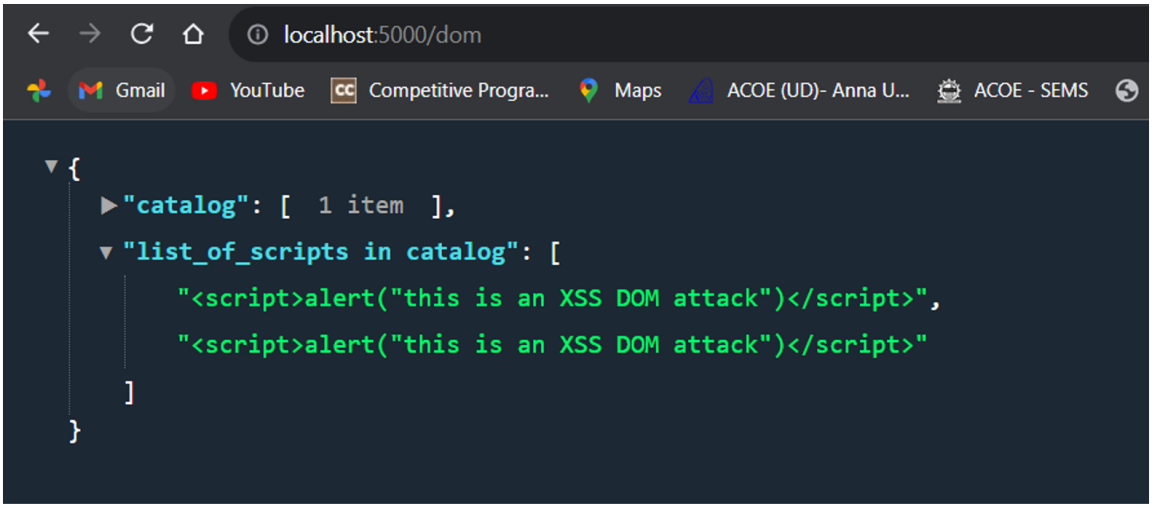
- By feeding through the URL.





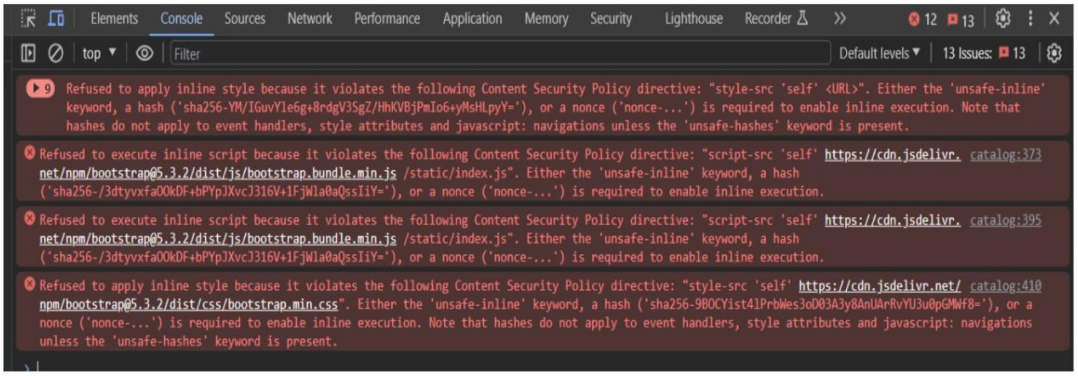
IMPLEMENTATION

- Detection of XSS – DOM attack using Regular expression and pattern matching



IMPLEMENTATION - MITIGATION

- Implemented CSP Policy to Prevent XSS DOM Attacks
- Illustration of warnings raised after prevention of the attacks.



EVALUATION METRICS

Types of attacks detected

- Injecting a script tag with malicious script inside or a link to the script
  - Eg: <script>alert(document.cookie)</script>

- Injecting a particular html tag with the script hidden inside of a HTML tag attribute
  - Eg: <img src=1 onerror=alert("XSS DOM attack") >

BENEFITS

- **Uncompromised User Confidence:** Mitigating XSS-DOM vulnerabilities fosters user trust in application's security and reliability.
- **Preserved Data Integrity:** By neutralizing these threats, you maintain the integrity of user data, preventing unauthorized access and tampering.
- **Enhanced Brand Reputation:** A secure application reflects positively on a brand, positioning the owner as a responsible and security-conscious provider.
- **Reduced Legal and Financial Risk:** Preventing attacks helps you avoid potential legal liabilities and financial losses that can arise from data breaches or compromised user information.

CONTRIBUTION

Krishnaa S	• Developed admin and catalog pages • Configured SQLite database • Implemented backend logic with Flask • Integrated attack detection using regex • Implemented mitigation via CSP
Jawahar A S	• Designed front-end of Notes page
Thamizharasi M	• Designed front-end of Login and Sign-up page

THANK YOU