Advanced Browser Security Features for Mitigating Web-Based Attacks

Muhammad Iqbal Mohd Khairi 1, a), Uzma Jafar 1, 2, b) and Hafiz Adnan Hussain 1, 2, c)

*1Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia  
2Centre for Big Data and Blockchain Technologies, CoE for Advanced Cloud, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia*

b) Corresponding author: uzma.jafar@mmu.edu.my

a) muhdiqbal1000@gmail.com

c) hafiz.adnan@mmu.edu.my

**Abstract.** Web browsers serve as the primary interface between users and the internet, making them a major target for web-based attacks. As digital activities increasingly rely on browsers for sensitive tasks like banking, communication, and remote work, the need for robust browser security mechanisms has become urgent. This paper presents a comprehensive Literature Review and Systematic Literature Review (SLR) on advanced browser security features developed between 2021 and 2025. The study analyses peer-reviewed research and industry findings to evaluate script injection defences, plugin security, site isolation, HTTPS enforcement, profile/session protection, hardware-based mechanisms, and passwordless authentication. The novelty of this paper lies in its dual-layered review approach and identification of integration challenges across browser architectures. Our findings emphasize the need for behaviour-based detection models, enhanced user-interface signals, and cross-feature interoperability to mitigate threats such as XSS, phishing, clickjacking, and MitB attacks. This work contributes actionable insights and highlights future research opportunities in browser-side security design.

# Introduction

This document offers both a review of existing literature and a systematic review (SLR) concerning the effectiveness of the browser security features, as well as identifying gaps that need deeper investigation. There are still issues with browser security due to the ever-changing strategies employed by attackers, inconsistent cross-platform user implementation, and user vulnerabilities. The systematic assessment of these specific security mechanisms involves evaluating their existing benefits and prominent deficiencies, as well as gaps hindering their practical effectiveness, including surfacing gaps contrary to research. Addressing the key research questions mentioned below:

1. How effective are current browser-based defences against evolving web-based attacks?
2. What are the integration, usability, and deployment issues related to these defences?
3. What should browser vendors and researchers focus on to improve browser security ecosystems?

# Literature Review

## Script Injection Defences

Cross-site scripting (XSS) continues to pose significant threats to web browsers [1]. Analysed trends in XSS vulnerabilities and mitigation strategies. Content Security Policy (CSP) has become a standard defensive mechanism, yet its misconfiguration remains a common issue. In [2], ScriptShield is a runtime JavaScript behaviour profiler that dynamically detects anomalies during script execution. Their solution highlights the shift from static defences toward behaviour-based, adaptive protection mechanisms. Figure 1 illustrates how a malicious site can impact the system, similarly to how the browser security features and their objectives are presented in Table 1.

### Architecture diagram showing a sequence titled 'How browsers create open channels for web-based threats.' A user accesses an unmanaged device, which then uses an outdated browser to open an email. The user clicks a phishing email link, represented by an icon and a label reading 'User clicks phishing email link.' This triggers the browser to create a new session to a malicious site, depicted with an icon and the label 'Browser creates new session to malicious site.' Arrows connect each step to illustrate the flow from user to threat.

**FIGURE 1.** How a malicious site can affect the system

## Browser Extension and Plugin Security

Browser extensions, while enhancing functionality, can introduce serious security risks. Patel et al. [3] developed a machine learning model to detect overprivileged and malicious browser extensions by analysing access and permission patterns. Additionally, runtime DOM monitoring, such as that in ExtPrivWatch [4] has emerged as an effective technique to mitigate unauthorized data access by third-party extensions.

## Profile and Session Protection

Persistent user sessions and profile data, such as cookies and tokens, are increasingly targeted by attackers. Somé et al. [5] identified profile storage as a vulnerability in mainstream browsers. Liu and Abbas [6] introduced a biometric-secured session container, which employs sandboxing and real-time anomaly detection to protect session integrity and restrict unauthorized session hijacking.

## Site Isolation and Spectre Mitigations

Modern browsers like Chrome and Edge have adopted site isolation to mitigate speculative execution attacks such as Spectre. This technique separates each domain’s processes, minimizing cross-site data exposure. While memory overhead remains a challenge, adaptive isolation strategies show promise in balancing security and resource usage [7].

## HTTPS Enforcement

Transport Layer Security (TLS) and HTTPS are necessary infrastructural components for secure web browsing. Silva and Kumar [6], [7] focused on the effectiveness of HSTS and TLS 1.3 in mitigating man-in-the-middle attacks. Their CertEnforce framework reduces the exposure arising from policy misconfiguration by automating governance of secure transport policy, thus improving transport policy governance.

## Hardware-Based Protection

Work is being done on TEEs (Trusted Execution Environments) such as Intel SGX and ARM TrustZone as possible protective shields for browsers. Wang et al. [8] and Lin et al. [9] expressed that placing vital browser operations into TEEs protects against memory scraping and MitB attacks. These protective frameworks are certainly attractive, but they do have a performance cost.

## Passwordless Authentication Protocols

The most recent changes to browsers include enabling the use of passwordless authentication, FIDO2, and WebAuthn. These protocols not only strengthen defences against phishing attacks, but also improve overall usability as noted by Patel and Cheng [10]. Furthermore, underlying authenticators (biometric) enhance protective measures while improving ease of use and security.

**TABLE 1.** Browser security features and their objectives

|  |  |  |
| --- | --- | --- |
| **Security Feature** | **Primary Objective** | **Threats Addressed** |
| Script Injection Defenses | Prevent execution of unauthorized scripts | Cross-Site Scripting (XSS), Code Injection |
| Extension and Plugin Security | Mitigate risks from malicious or over-privileged extensions | Malicious Extensions, Over-Permission, Unauthorized Data Access |
| Profile and Session Protection | Safeguard user data and session integrity | Session Hijacking, Token Theft, Unauthorized Access to User Profiles |
| Site Isolation | Enforce separation between websites' data and processes | Spectre, Cross-Domain Data Leakage, Side-Channel Attacks |
| HTTPS Enforcement | Ensure secure communication between browser and web server | Man-in-the-Middle (MitM) Attacks, Spoofing |
| Hardware-Based Protection | Utilize hardware capabilities for enhanced security | Memory Scraping, Man-in-the-Browser (MitB) Attacks |
| Passwordless Authentication Protocols | Replace traditional passwords with stronger authentication methods | Phishing, Credential Theft |

# Systematic Literature Review (SLR)

## Methodology

This systematic literature review (SLR) scanned the literature published between 2021 and 2025 and used peer-reviewed articles from IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, and arXiv. The primary focus and search terms were: “browser security,” “WebAuthn,” “site isolation,” “XSS prevention,” and “certificate policy management.”

The inclusion criteria focused on:

* Browser-side mitigation techniques
* Empirical evaluations of browser security
* Articles published between 2021 and 2025

Exclusion criteria involved:

* Server-side security models
* Non-peer-reviewed material

A total of 14 articles were initially identified. After removing duplicates and applying inclusion/exclusion criteria, 10 articles were selected for in-depth analysis. The screening process followed PRISMA guidelines. Each selected study was reviewed for technical contribution, experimental rigor, and relevance to current browser security architectures.

## Analysis and Discussion

The following Table 2 summarizes the key browser security features, the threats they target, and the technologies proposed in the reviewed studies.

**TABLE 2.** Key browser security features and their mitigation strategies

|  |  |  |  |
| --- | --- | --- | --- |
| **Security Focus** | **Threats Mitigated** | **Technologies/Methods** | **Reference** |
| Script Injection Defense | XSS, code injection | CSP, ScriptShield, behavior profiling | [1] |
| Extension Security | Malicious extensions, over-permission | ML models, ExtPrivWatch | [2] |
| Profile and Session Security | Session hijacking, token theft | Biometric authentication, sandboxed containers | [3] |
| Site Isolation | Spectre, cross-domain data leakage | Adaptive isolation | [4, 5] |
| HTTPS and Transport Security | Man-in-the-middle, spoofing | CertEnforce, TLS 1.3, HSTS | [6] |
| Passwordless Authentication | Phishing, credential theft | FIDO2, WebAuthn | [10] |

Table 2 provides a comprehensive overview of the key browser security features and the corresponding threats they mitigate. While the table successfully categorizes core mechanisms such as HTTPS enforcement and script injection defenses, it omits newer threat vectors like adversarial machine learning and supply chain vulnerabilities related to browser updates and plugins. Additionally, the table does not fully address the interplay between features. For example, the effectiveness of passwordless authentication is amplified when combined with session anomaly detection, while site isolation benefits from integration with hardware-backed memory protection. The next revisions of this framework should address the hybrid and interrelated approach, which incorporates multifaceted strategies that depict the reality of the situation regarding browser defences.

### Script Injection Defence

The risks related to cross-site scripting (XSS) and code injection remain one of the most notorious threats attacking web browsers. An attacker who wants to do malicious things can execute different programs in the web browser of the targeted victim. As a result, account hijacking, data breaches, and full account takeovers are common. Though it is possible to defend them with Content Security Policy (CSP), its enforcement and configuration gaps, as pointed out in [1-5], are critical weaknesses. To address those gaps, newer solutions have been implemented, such as behavior-based profiling for executing script anomalies monitoring within [2]. Further enhancements through CSP Level 3 features like script nonces, script hashes, and subresource integrity (SRI) provide more control over third-party scripts. A static CSP framework combined with real-time detection provides more robust protection against reflected and stored XSS attacks.

### Browser Extension and Plugin Security

Though browser extensions may enhance user engagement with technology, they present notable security risks due to their deep-level access to a webpage's content, cookies, and application programming interfaces (APIs). Extensions of either bad intention or being too loose in their settings pose a risk. As an example, Patel and Cheng [10] applied machine learning algorithms to anomaly detection for extension permission request profiling. Runtime methods like ExtPrivWatch have been suggested for monitoring and controlling abuse of override interfaces with the Document Object Model (DOM). Apart from behavioral tracking, cryptographic techniques such as DOMtegrity provide means for verifying partial content integrity concerning extensions. Some recent studies are looking into using blockchain technology for assigning and unassigning extension permissions, enabling trustless, tamper-proof audit trails.

### Profile and Session Protection

User profiles often contain highly sensitive data such as session tokens, browsing history, and stored credentials, and have become a prime target for attackers. Somé et al. [1] revealed critical flaws in browser profile storage, particularly in how credentials are stored and accessed. Their study emphasized the need for encrypted, sandboxed user profiles and stronger file system isolation, and in [3] proposed a biometric-secured session container that uses sandboxing combined with anomaly detection to ensure session integrity and prevent hijacking. Future designs may adopt zero-trust principles, ensuring each browser tab operates with minimal privileges and requiring explicit re-authentication for sensitive actions.

### Site Isolation and Speculative Execution Defense

Site isolation used notably in Chrome and Edge ensures that content from different domains runs in separate processes, providing strong defenses against side-channel attacks such as Spectre. This architectural change limits cross-origin data leakage by isolating memory spaces [6]. Although site isolation increases memory usage, adaptive resource management strategies are being explored to balance performance with security. These approaches are particularly important given the persistent nature of hardware-level threats and speculative execution vulnerabilities.

### HTTPS Enforcement and Transport Security

Transport mechanisms such as HTTPS are critical for user privacy as they mitigate man-in-the-middle (MitM) attacks and noted issues with uniform policy enforcement and certificate management. With CertEnforce, configurations for TLS are automatically processed, which reduces the chances of security header misconfigurations. Moreover, improving browser user interface indicators such as padlocks and warning dialogs is essential to depict connection safety better and reduce user carelessness, as pointed out by other researchers.

### Hardware-Based Protection

Examples of Intel SGX and ARM TrustZone illustrate the execution environments that an operating system might use to do sensitive operations while keeping sensitive parts of the browser isolated. TEEs could help protect passwords, form fields, and cryptographic keys from malware and other processes that might compromise them. Previously, Eskandarian et al. [4] showed that TEEs could counteract MitB attacks by executing critical code within enclaves and isolating it from the rest of the system. While performance constraints and platform dependencies pose challenges, these are being addressed with multi-enclave architectures and cloud-based secure systems designed to offer more comprehensive hardware protections.

### Passwordless Authentication Protocols

Firefox Multi-Account Containers, along with risk assessment and real-time session activity evaluation, exemplify container-based isolation. Additionally, browsers are increasingly integrating behavioural biometrics such as rhythm, typing, mouse movements, and container-based isolation and risk assessment, like Firefox Multi-Account Containers. To mitigate risks related to session hijacking and impersonation attacks, some suggest combining continuous validation processes with zero-trust frameworks at the browser level.

## Comparative Evaluation

Reviewing prominent browsers like Chrome, Edge, Firefox, and Safari reveals how each one integrates security measures. Both Chrome and Edge integrate complete site isolation, advanced control on extensions, and full WebAuthn support. They pioneered complete site isolation and advanced control on extensions. Firefox provides reasonable privacy and customization and has useful extensions but does not provide adequate strong hardware-backed protections or secure credential handling for application credentials. While Safari is focused on user privacy, mitigates tracking, and limits extension usage, these features enhance security against certain attack vectors at the cost of extensibility. Users disregard HTTPS warning prompts, gleefully install extensions from questionable sources, or while developing, disable essential security measures like Content Security Policy (CSP). These actions showcase why education targeting security is vital. This provides insight into systems that focus on security over technical barriers and aim to simplify the user interface and experience.

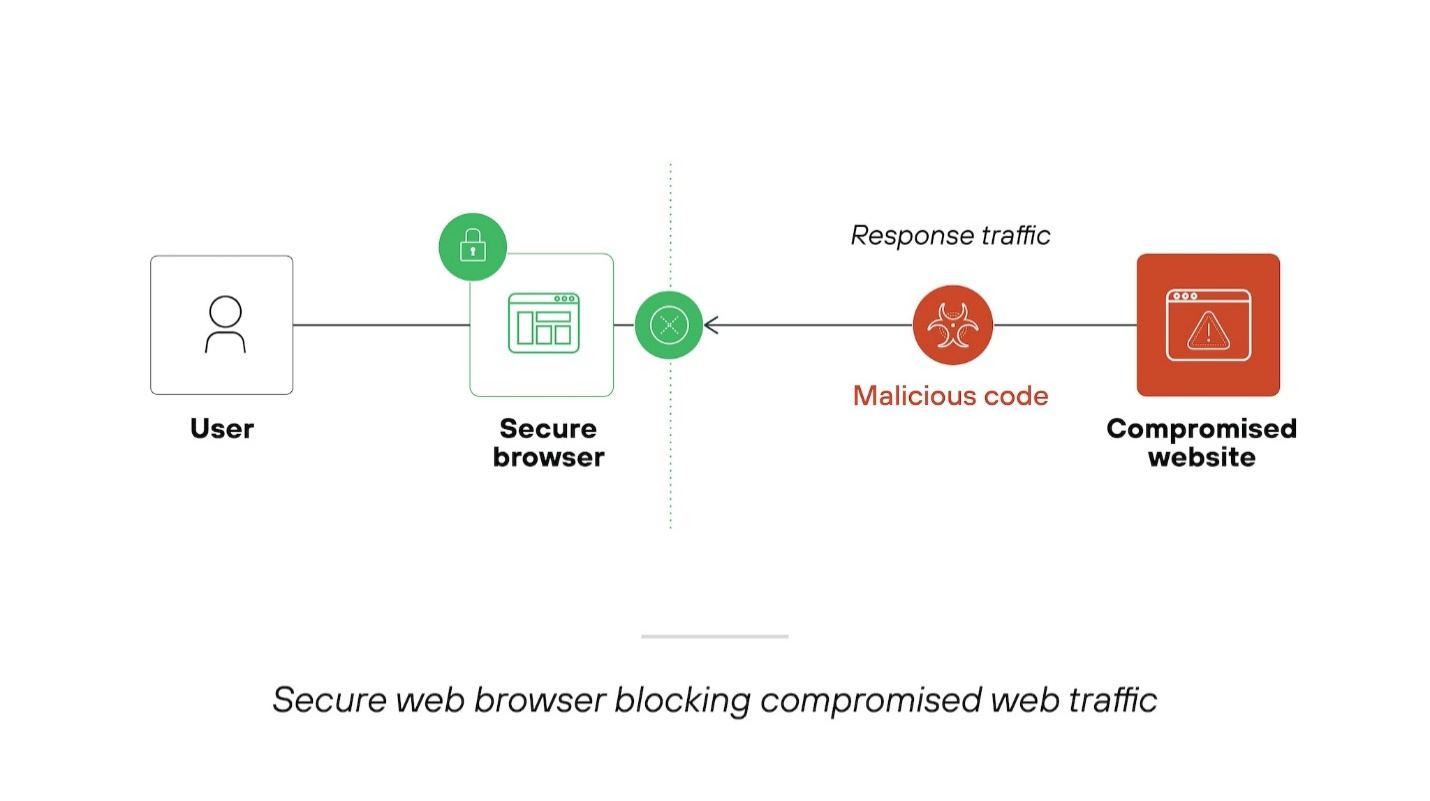
## Identified Research Gaps

The browser security literature review highlights several critical gaps in research. To begin with, there are no existing standardized benchmarks for testing and evaluating the security features of different browsers across vendors. Several gaps were identified through this review:

1. Benchmarking: A lack of standardized evaluation metrics for browser security hinders meaningful comparison across platforms.
2. User Behavior: The impact of users’ actions on the security of browsers is still not well studied, particularly with regards to security notifications and permission windows.
3. Integration Challenges: WebAuthn and similar technologies have issues with device agnostic cross sync.
4. Automation Risks: Built-in tools like autofill and password managers can introduce vulnerabilities if not implemented properly.

The lack of benchmarks makes it impossible to assess feature efficiency comparison and conduct meaningful longitudinal studies. Another gap remains in usability and user perception of security functions at the browser level, for example, the effectiveness and clarity of permission prompts, warning dialogs, or any other user interface elements. There are also some unresolved problems concerning WebAuthn and comparable protocols pertaining to the synchronization of secure credential sets across platforms and devices. Another critical gap under researched is the influence of automated browser functions such as password management and autofill interfaces, which could pose new security vulnerabilities if not properly designed. These gaps indicate that improving browser security will likely require more than just a technical solution; interdisciplinary approaches incorporating usability design and systems integration will also be needed.

The operation of a secure browser that prevents access to already hacked or harmful websites and actively filters such traffic is described in Figure 2. This proactive defence model utilizes a blend of static blacklists alongside machine learning-driven dynamic behavioural profiling. Browser adaptive profiling provides the capability to respond to new threats by identifying anomalies in user or site behaviour in real time. These approaches are efficient for known and pattern-based threats; however, they need to learn and adapt to be effective. The model shown in Figure 2 works together with а session protection, biometric authentication, and other models, which makes it possible to have a multi-layered defence strategy that significantly minimizes the potential attack avenues.



**FIGURE 2.** Secure web browser blocking compromised web traffic

# CONCLUSION

This review highlights the growing sophistication of browser security mechanisms and their importance in protecting users from a wide array of web-based attacks. Features such as site isolation, TLS enforcement, behavioural detection, and passwordless authentication provide strong individual defences. However, their real-world effectiveness is often compromised by poor user awareness, inconsistent implementation, and a lack of integration across security layers. Key findings include the efficacy of hybrid defence models, the critical role of session protection, and the untapped potential of biometric authentication and TEEs. To advance browser security, we recommend a multi-layered, user-aware approach that integrates static configurations with adaptive, real-time threat detection. Future work should prioritize usability, interoperability, and automation to ensure scalable and effective deployment in diverse user environments.

# References

1. A. Hannousse, S. Yahiouche, and M. C. Nait-Hamoud, “Twenty-two years since revealing cross-site scripting attacks: a systematic mapping and a comprehensive survey,” arXiv:2205.08425 (2022).
2. A. Kapravelos, Y. Shoshitaishvili, L. Cavedon, C. Kruegel, and G. Vigna, “Revolver: An automated approach to the detection of evasive web-based malware,” Proc. 23rd USENIX Security Symposium (2014).
3. D. F. Somé, M. Airan, Z. Durumeric, and C.-A. Staicu, “User profiles: The Achilles’ heel of web browsers,” arXiv:2504.17692 (2025).
4. S. Eskandarian, J. Cogan, S. Birnbaum, P. C. Wei, D. Franke, F. Fraser, … and D. Boneh, “Fidelius: Protecting user secrets from compromised browsers,” arXiv:1809.04774 (2018).
5. A. Greenberg, “With Spectre still lurking, Google looks to protect the web,” WIRED (2019).
6. A. Barth, C. Jackson, and J. C. Mitchell, “Robust defenses for cross-site request forgery,” Proc. 15th ACM Conference on Computer and Communications Security (2008).
7. D. Akhawe, A. Barth, P. E. Lam, J. C. Mitchell, and D. Song, “Towards a formal foundation of web security,” IEEE Computer Security Foundations Symposium (2013).
8. D. Fett, R. Küsters, and G. Schmitz, “A comprehensive formal security analysis of OAuth 2.0,” Proc. ACM SIGSAC Conference on Computer and Communications Security (2019).
9. H. Kim et al., “ScriptShield: A runtime JavaScript behavior profiler for detecting anomalies during script execution,” (2022).
10. M. Patel and L. Cheng, “Detecting overprivileged and malicious browser extensions using machine learning analysis of access and permission patterns,” (2023).