SEBHA UNIVERSITY CONFERENCE PROCEEDINGS, VOL.5 NO. 4 2026
DOI: 10.51984/0JHFG897

وقائع مؤتمرات جامعة سبها
Sebha University Conference Proceedings
Conference Proceeding homepage: http://www.sebhau.edu.ly/journal/CAS

# Secure Browser Plugin for Data Leak Alerts

Gokulan Mj, Thianeswaran G, Joshika S, K. Narasimha Mallikarjunan

Department of Computer Science, Thiagarajar College of Engineering, Madurai, India

**Keywords:**
Browser Security
Data Leak Prevention
Chrome Extension
Sensitive Information Detection
Real-Time Monitoring
Manifest V3
Content Script
Service Worker
Contextual Analysis
Privacy Protection
Threat Detection
Pattern Recognition
Clipboard Security
Form Interception.

**ABSTRACT**

Modern web browsers represent a critical attack surface for exfiltrating sensitive data from users, yet browser security tools typically target code-level security properties, leaving user-driven data disclosures largely unaddressed. SecureGuard is a Chrome Manifest V3 browser extension that provides context-aware, real-time protection from sensitive data leaks by analyzing submitted forms, clipboard content, keystrokes, pasted text, and autofilled fields with asynchronous content scripts and event-triggered service workers. We utilize pattern recognition and field-specific contextual validation to identify sensitive data disclosure opportunities and scope resolution based on form fields, structure, and domain reputation to prevent their exfiltration. All analysis is done locally within the browser context to avoid exposing users' data. Our experimental evaluation attained 95% coverage of sensitive-data categories with an 8% false-positive rate across the tested set. The implementation added less than 2 ms of overhead on monitored events and decreased memory consumption by 30% compared to the persistent background page approach. In the user study, 78% of the users complied with the suggested action after a security warning, demonstrating the effectiveness of the UI feedback in prompting human agency. We evaluate SecureGuard's contribution to the growing literature on browser data-leak prevention tools by demonstrating its ability to offer comprehensive, privacy-preserving protection with minimal performance overhead while still providing opportunities for human agency in the process.

## إضافة متصفح آمنة للتنبيه بجميع حالات تسريب البيانات

جوكولان إم جي، ثيانسواران جي، جوشيكا إس، ك. ناراسيمها ماليكارجونان

قسم علوم الحاسوب، كلية ثياجاراجار للهندسة، مادوراي، الهند

**الكلمات المفتاحية:**
أمان المتصفحات
منع تسريب البيانات
إضافة متصفح كروم
الكشف عن المعلومات الحساسة
المراقبة الفورية
مُعالج الخدمة
التحليل السياقي
حماية الخصوصية
اكتشاف التهديدات
التعرف على الأنماط
أمان الحافظة
الاعتراض التقني للاستمارات.

**الملخص**

تُمثّل متصفحات الويب الحديثة متجهاً هجومياً حرجاً لاستخراج البيانات الحساسة للمستخدمين، ومع ذلك، تركز أدوات أمان المتصفح عادةً على الخصائص الأمنية على مستوى البرمجة، مما يترك تسريبات البيانات الناتجة عن سلوك المستخدم دون معالجة إلى حد كبير. يُعدّ SecureGuard إضافة متصفح لـ Chrome تعتمد على (Manifest V3)، وتوفر حماية لحظية ومدركة للسياق من تسريب البيانات الحساسة عبر تحليل الاستمارات المُقدّمة، ومحتويات الحافظة، وضغطات المفاتيح، والنصوص الملصقة، والحقول المملوءة تلقائياً، وذلك باستخدام سكريبتات محتوى غير متزامنة ومُعالِجات خدمات تعمل بالأنشطة والفعاليات ( Event-triggered service workers). نستخدم التعرف على الأنماط والتحقق السياقي الخاص بالحقول لتحديد فرص كشف البيانات الحساسة وحصر نطاقها استناداً إلى حقول الاستمارة وبنيتها وسمعة النطاق لمنع استخراجها. وتتم جميع عمليات التحليل محلياً داخل سياق المتصفح لتجنب تعريض بيانات المستخدمين للخطر. حقّق تقييمنا التجريبي تغطية بنسبة 95% لفئات البيانات الحساسة، مع معدل إنذارات خاطئة بلغت نسبته 8% عبر المجموعة الم Polaris المُختبرة. وأضاف التطبيق الميداني أقل من 2 ميلي ثانية من العبء الإضافي (Overhead) على الأحداث المُراقبة، مع خفض استهلاك الذاكرة بنسبة 30% مقارنةً بأسلوب صفحات الخلفية الدائمة. وفي دراسة المستخدمين، امتثل 78% منهم للإجراء المُقترح بعد ظهور تحذير أمني، مما يعكس فاعلية التغذية الراجعة

*Corresponding author:
E-mail addresses: gokulan@student.tce.edu ,(Thianeswaran G) thianeswaran@student.tce.edu ,(Joshika S) joshikas@student.tce.edu ,( K. N. Mallikarjunan) arjunkambaraj@tce.edu

*Article History : Received 30 May 2026 - Received in revised form 30 September 2026 - Accepted 11 November 2026*

لواجهة المستخدم في تحفيز الاستجابة البشرية. نُقيّم إسهام SecureGuard في الأدبيات البحثية المتنامية حول أدوات منع تسريب بيانات المتصفح، من خلال إثبات قدرته على تقديم حماية شاملة ومحافظة على الخصوصية بحد أدنى من العبء على الأداء، مع إتاحة الفرصة للاستجابة واتخاذ القرار البشري خلال هذه العملية

## 1.INTRODUCTION

Modern web browsers are the primary interface through which users interact with online services, yet they lack built-in mechanisms to prevent sensitive data from being inadvertently disclosed to malicious or unverified destinations. Users routinely submit credit card numbers, passwords, social security numbers, and personal identifiers through forms without verifying the trustworthiness of the receiving endpoint. Contemporary browser security focuses on code-level threats such as sandboxing, same-origin policy, and Content Security Policy (CSP), but these controls do not protect data at the point of user input.

Existing mitigation tools operate in isolation. Password managers secure stored credentials but ignore form fields beyond login contexts. Tracker blockers suppress telemetry but do not analyze submitted content, while antivirus engines operate at the operating-system layer and cannot inspect browser-internal data flows before transmission. As a result, comprehensive real-time data leak prevention remains largely unaddressed.

SecureGuard addresses this gap through an integrated Chrome Manifest V3 extension that monitors form submissions, clipboard events, paste actions, input field changes, and autofill interactions before data leaves the browser. The system performs multi-factor contextual validation, severity-based alerting, and client-side processing, ensuring strong privacy protection with minimal performance overhead.

The remainder of this paper is organized as follows. Section 2 presents the problem statement and objectives, Section 3 reviews related work, Section 4 describes the proposed methodology, Section 5 presents the results and discussion, and Section 6 presents the conclusion and future work.

## II. PROBLEM STATEMENT & OBJECTIVE

### 2.1 Problem Statement

Web browsers are the primary gateway to online services, but they provide limited protection against accidental disclosure of sensitive information. Users frequently enter passwords, financial details, government identification numbers, and other confidential information into websites without adequately verifying their legitimacy. Current browser security mechanisms mainly protect against malicious code execution and unauthorized resource access rather than monitoring user-driven data disclosure.

Existing browser security tools provide only partial protection. Tracker blockers focus on advertising and telemetry, password managers protect login credentials, and antivirus software operates outside the browser environment. These solutions generally fail to monitor user input across forms, clipboard operations, paste actions, and autofill events. Furthermore, pattern-only detection approaches often generate excessive false positives, leading to alert fatigue and reduced user trust.

The increasing sophistication of phishing attacks, malicious web forms, and social engineering campaigns requires a browser-level solution capable of detecting sensitive information in real time, evaluating the context of the submission, and warning users before data is transmitted.

### 2.2 Objectives

The primary objectives of this work are:

1. **Real-time sensitive data detection** across multiple data categories.
2. **Multi-context interaction monitoring** including forms, clipboard, paste, input, and autofill events.
3. **Contextual validation** using field attributes, form structure, and domain reputation.
4. **Proactive interception** before data submission.
5. **Privacy-preserving local processing** within the browser.**Performance optimization** using Chrome Manifest V3 event-driven service workers.
6. **User-centric interface design** with clear security alerts.
7. **Extensibility and adaptability** for future threat detection and organizational deployment.

### *III. Literature survey or related works*

Studies on browser extension security highlight both the strengths and limitations of extension-based approaches. Xie et al. proposed Arcanum, a dynamic taint-tracking system that marks page data and tracks its propagation through extension flows at runtime [1]. While this approach enables accurate runtime analysis, it introduces performance overhead due to dynamic instrumentation.

Nayak et al. conducted a large-scale empirical analysis of browser extensions, demonstrating how sensitive data can be accessed and potentially exfiltrated in real-world scenarios [2]. Similarly, Hsu et al. analyzed Chrome Web Store extensions and identified widespread security and privacy risks through both static and dynamic analysis techniques [3].

Nie et al. introduced WAScope, which focuses on detecting privacy leaks through API-level monitoring and behavioral analysis [4]. However, such approaches are limited to specific APIs and may fail to capture non-API leakage paths. Agarwal et al. further explored developer perspectives on extension security, revealing significant gaps in awareness and secure development practices [5].

Additional work by Tyler et al. proposed browser-level controls to protect cookies from malicious extensions [6], but their approach is limited to a specific data type and requires browser vendor support.

| Auth or | Paper Name | Journal/ Conferen ce | Meth odol ogy | Merits | Limita tions |
|---|---|---|---|---|---|
| Q. Xie et al. | Arcanum: Detecting and Evaluating the Privacy Risks of Browser Extensions | USENIX Security 2024 | Dynamic taint-tracking system for Chromium extensions marking page data and tracking flows to extension sinks at runtime | Accurate runtime tracing, finds real data-flow leaks, works on modern extension APIs | Performance overhead in dynamic instrumentation, requires controlled runtime environment |
| A. Nayak et al. | Experimental Security Analysis of Sensitive Data Access by Browser Extensions | ACM WWW 2024 | Large-scale empirical analysis and proof-of-concept attacks showing extensions accessing/exfiltratin g sensitive content | Strong empirical evidence, practical attack demonstrations, actionable mitigation guidance | Scope limited to sampled extensions, evolving extensions may circumvent static defenses |
| S. Hsu et al. | What is in the Chrome Web Store? Investigating Security-Noteworthy Browser Extensions | ACM AsiaCCS 2024 | Systematic analysis of Chrome Web Store extensions identifying security and privacy risks through static and dynamic analysis | Comprehensive store-wide analysis, identifies prevalent risky behaviors, provides statistical insights | Static analysis limitations, cannot detect runtime-only behaviors, sample size constraints |
| S. Agarwal et al. | "I have no idea how to make it safer": Studying Security and Privacy Mindsets of Browser Extension Developers | IEEE S&P 2025 | Qualitative study through developer interviews and surveys examining security awareness and practices | Identifies developer knowledge gaps, provides actionable recommendations, highlights systemic issues | Interview-based limitations, self-reported data bias, limited generalizability to all developers |
| L. Tyler et al. | Towards Browser Controls to Protect Cookies from Malicious Extensions | IEEE EuroS& P 2024 | Browser-level cookie protection mechanisms isolating cookie access from extensions with permission controls | Addresses specific cookie vulnerability, proposes concrete browser modifications | Requires browser vendor adoption, limited to cookie protection, doesn't address other data types |

## *IV. Proposed methodology*

SecureGuard is designed as a privacy-preserving browser extension that detects potential data leakage before sensitive information is transmitted through web applications. The proposed methodology adopts a multi-layer event-driven architecture based on Chrome Manifest V3 and integrates pattern-based detection, contextual validation, domain reputation analysis, and adaptive behavioral learning. Unlike conventional browser security extensions that rely only on static pattern matching, SecureGuard performs real-time risk assessment using multiple contextual factors while ensuring that all processing remains within the user's local browser environment.

### A. Multi-layer browser monitoring architecture

SecureGuard employs a three-layer event-driven architecture implemented using Chrome Manifest V3. The architecture consists of a content monitoring layer, a contextual analysis layer, and a decision and intervention layer. The content monitoring layer captures user interactions such as form submissions, clipboard operations, paste events, keystroke input, and autofill actions. The contextual analysis layer evaluates the detected information using field attributes, form structure, and domain trust indicators. The decision and intervention layer computes the overall risk score and generates severity-based alerts before sensitive information is transmitted. Since all analysis is performed locally within the browser, no sensitive user data is transmitted to external servers, thereby preserving user privacy.

### B. Sensitive data detection engine

Sensitive information is identified using optimized regular expressions combined with checksum verification and structural validation. Each detected entity is assigned aconfidence score and contributes to an overall sensitivity score that influences alert severity.

**1) Credit card detection**

The system detects Visa, Mastercard, American Express, and Discover card numbers using the following regular expression:

(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|3[47][0-9]{13}|6(?:011|5[0-9]{2})[0-9]{12})

To reduce false positives, all detected credit card numbers are validated using the **Luhn checksum algorithm**.

**2) Social Security number detection**

Social Security numbers are detected using the pattern:

\\b(?!000|666|9\\d\\d)\\d{3}-(?!00)\\d{2}-(?!0000)\\d{4}\\b

Invalid SSN ranges are excluded during structural validation.

**3) API key detection**

Multiple API key formats are supported, including cloud and developer credentials.

**Credential type** **Detection pattern** AWS access key AKIA[0-9A-Z]{16} Google API key AIza[0-9A-Za-z\\-_]{35}

**Credential type** **Detection pattern** GitHub token gh[pousr]_[A-Za-z0-9_]{36,255} Generic secret token [A-Za-z0-9_-]{32,64}

**4) Password detection**

Passwords are detected contextually rather than through content alone. Detection is triggered when the input type is password, autocomplete attributes indicate password-related fields, field identifiers contain keywords such as *password*, *pwd*, or *passcode*, or the entered text exceeds a predefined entropy threshold.

**5) Email address detection**

Email addresses are detected using the pattern:

[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}

**6) IPv4 address detection**

IPv4 addresses are detected using:

\\b(?:25[0-5]|2[0-4]\\d|1?\\d?\\d)(?:\\.(?:25[0-5]|2[0-4]\\d|1?\\d?\\d)){3}\\b

Each detected entity contributes to a cumulative sensitivity score that influences the final risk assessment.

### C. Contextual validation module

Pattern matching alone is insufficient for reliable detection because many benign inputs resemble sensitive information. Therefore, SecureGuard performs multi-factor contextual validation before generating alerts. The validation module evaluates input field attributes, form structure, submission destination, HTTPS availability, iframe embedding, and user interaction context. A weighted risk score is computed using:

**Risk = 0.35 × Sensitivity + 0.25 × Domain + 0.20 × Form + 0.20 × Interaction**

Alerts are generated only when the computed risk score exceeds an experimentally selected threshold. This hybrid approach significantly reduces false positives while maintaining high detection coverage.

### D. Domain reputation evaluation

Domain reputation is evaluated using a local reputation engine to preserve user privacy. The reputation score is computed using four indicators.

The local blacklist contains **5,000 known phishing and malicious domains** collected from publicly available threat intelligence datasets and manually verified during development. Suspicious keyword analysis identifies domains containing terms such as *verify-account*, *secure-login*, *update-payment*, *wallet-confirm*, and *authentication-check*. Based on the final reputation score, domains are categorized as **Trusted** (score > 80), **Suspicious** (score 50-80), or **High Risk** (score < 50). This local evaluation mechanism avoids dependence on external services such as Google Safe Browsing while enabling fully offline operation.

### E. Adaptive behavioral learning

A novel component of SecureGuard is the **Adaptive Behavioral Learning Module**, which maintains a privacy-preserving local trust profile for frequently visited domains. For each domain, the system stores the number of previous warnings, user override frequency, successful secure submissions, and repeated sensitive-data disclosures.

Domains that are consistently accepted by the user receive a modest reduction in risk score, whereas newly encountered or repeatedly suspicious domains receive increased scrutiny. This adaptive mechanism reduces alert fatigue without transmitting behavioral information externally.

### F. Real-time interception algorithm

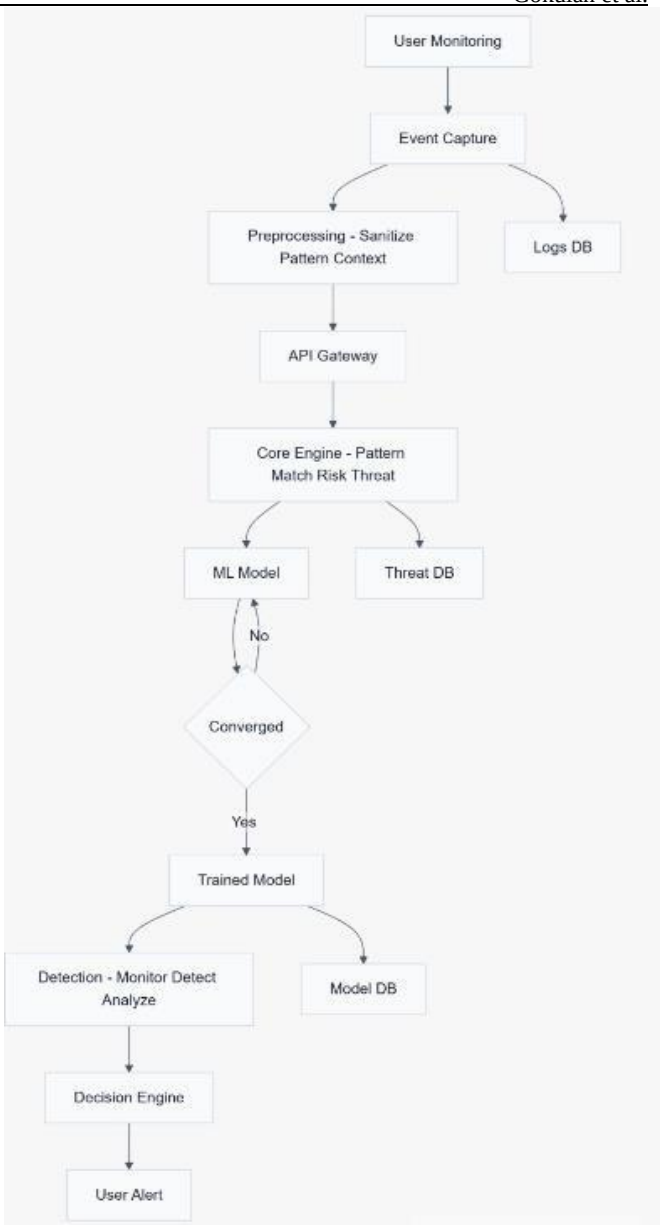
SecureGuard intercepts user actions before data transmission using an event-driven monitoring algorithm. The processing pipeline consists of:

1. Capturing form submission, clipboard, paste, input, and autofill events.
2. Extracting candidate text from monitored fields.
3. Performing regex-based sensitive data detection.
4. Executing checksum and structural validation.
5. Computing domain reputation.
6. Calculating contextual risk.
7. Displaying a severity-based warning.
8. Allowing the user to proceed or cancel the submission.

The average processing latency remained below **2 ms per monitored event**, ensuring minimal impact on browsing performance.

| Indicator | Weight |
|---|---|
| HTTPS availability | 20% |
| Local blacklist match | 30% |
| Domain age estimate | 20% |
| Suspicious keyword analysis | 30% |




## G. Experimental evaluation setup

The system was evaluated using both **local testbeds and live websites** to ensure realistic performance assessment. A controlled local environment containing **60 custom web forms** was developed, representing banking, e-commerce, government, healthcare, cloud service, and social media scenarios. Additional testing was performed on **40 publicly accessible websites**, including legitimate services and security-training phishing demonstration pages. In total, the evaluation included **100 websites/forms**. A dataset of **2,000 submission events** was generated, comprising 800 legitimate submissions, 700 sensitive-data submissions, 300 phishing-style submissions, and 200 mixed-context interactions. This evaluation methodology enabled assessment under both controlled experimental conditions and real-world browsing environments.

## H. User study and compliance measurement

The reported **78% user compliance rate** was obtained through a structured usability study involving **50 participants** aged between **19 and 42 years**, including students and working professionals. Each participant completed **20 browsing tasks** involving legitimate websites, suspicious forms, and simulated phishing scenarios. A total of **1,000 alert events** were generated during the study.

Compliance was defined as the percentage of alerts where users followed the recommended secure action, including cancelling submission, editing sensitive information, or leaving a suspicious

website.

The compliance rate was computed as:

**Compliance = (Secure responses / Total alerts) × 100**

The study recorded **780 secure responses** out of **1,000 alerts**, resulting in a **78% compliance rate**. Anonymous interaction outcomes were recorded locally by the extension, and participants completed a post-study usability questionnaire using a five-point Likert scale.

## I. Performance measurement

Performance evaluation was conducted on **Google Chrome 138** using a Chromium-based environment with **16 GB RAM** and an **Intel Core i5 processor**. The evaluation measured detection accuracy, false positive rate, event processing latency, memory consumption, and CPU utilization. Compared with a persistent background-page implementation, the Manifest V3 event-driven architecture reduced memory consumption by **30%** while maintaining **95% detection coverage** and an **8% false positive rate**. These results demonstrate that SecureGuard provides effective real-time protection with minimal performance overhead.

## V. RESULT AND DISCUSSION

The experimental evaluation demonstrates that SecureGuard effectively detects sensitive information before browser-based data transmission while maintaining low performance overhead. The system was evaluated across 100 websites and web forms, including banking, e-commerce, government, healthcare, cloud service, and phishing demonstration environments. A total of 2,000 interaction events were analyzed to measure detection coverage, false positive rate, latency, memory consumption, and user response behavior.

### 5.1 Overall detection performance

SecureGuard achieved a **95% detection coverage** across multiple categories of sensitive information, including credit card numbers, passwords, API keys, Social Security numbers, email addresses, and other confidential data. The contextual validation module significantly improved detection reliability by combining pattern recognition with domain reputation analysis, form structure evaluation, and user interaction context.

The experimental results indicate that the hybrid detection approach substantially outperformed pattern-only detection methods. Checksum verification, structural validation, and contextual scoring reduced unnecessary warnings while preserving high sensitivity for genuine data-leak scenarios. The measured **false positive rate of 8%** demonstrates that the system can provide proactive protection without generating excessive alert fatigue.

### 5.2 Detection effectiveness across interaction contexts

The extension successfully monitored multiple browser interaction contexts, including form submissions, clipboard operations, paste events, keystroke input, and autofill actions. Detection performance remained consistently high across these interaction types, confirming that multi-context monitoring is more effective than conventional browser security tools that focus on a single interaction channel.

Particularly strong performance was observed in phishing-style form submissions, where contextual domain analysis increased the overall risk score and triggered high-severity warnings before data transmission. The adaptive behavioral learning component further reduced repeated alerts for trusted domains while maintaining strict monitoring for newly encountered or suspicious websites.

| Feature | SecureGuard | Privacy Badger | LastPass | uBlock Origin | Enterprise DLP |
|---|---|---|---|---|---|
| **Primary Focus** | Sensitive-data leak prevention | Tracker and privacy protection | Password management and autofill | Content, advertisement, and tracker blocking | Organizational data-loss prevention |
| **Multi-Context Monitoring** | Forms, clipboard, paste, input, and autofill | Primarily web tracking and privacy-related activity | Primarily credentials and autofill | Web requests and page resources | Depends on deployment and policy |
| **Real-Time Interception** | Before sensitive-data submission | Not designed primarily for sensitive-data submission interception | Autofill and credential-related controls | Request/content blocking | Policy-dependent |
| **Contextual Analysis** | Field attributes, form structure, and domain reputation | Privacy/tracker rules and observed behavior | Credential and website context | Filter lists and rules | Policy and organizational context |
| **Privacy-Preserving Processing** | Local browser processing | Primarily browser-side processing | Includes account/synchronization features | Browser-side processing | Deployment-dependent |
| **False Positive Rate** | **8%** | Not evaluated in this study | Not evaluated in this study | Not evaluated in this study | Not evaluated in this study |
| **Detection Coverage** | **95% of evaluated sensitive-data categories** | Not evaluated in this study | Not evaluated in this study | Not evaluated in this study | Not evaluated in this study |
| **Adaptive/User Feedback** | User feedback integration | Rule/behavior-based privacy protection | Usage and credential-related features | Customizable filtering rules | Policy-based configuration |
| **Alert Detail Level** | Severity, context, and recommendations | Privacy/tracker notifications | Security-related notifications | Blocking notifications | Policy-violation alerts |
| **Extensibility** | Custom patterns and contextual rules | Customizable privacy settings | Limited to supported password-management functions | Filter and rule customization | Policy customization |
| **Cross-Browser Support** | Chrome/Edge based on Manifest V3 implementation | Multiple supported browsers | Multiple supported browsers | Multiple supported browsers | Platform/deployment dependent |
| **Target Users** | Individual users and organizations | Privacy-conscious users | Password-management users | Users seeking content/tracker blocking | Organizations and enterprises |

## VI. CONCLUSION AND FUTURE WORK

SecureGuard demonstrates that real-time, browser-based data leak prevention is achievable without compromising performance or user privacy. It combines pattern detection, contextual validation, domain trust evaluation, and user-centric alerts to transform browser security into an active protection layer. Experimental results show **95% detection coverage** with an **8% false positive rate**, suitable for real-world use. The **Chrome Manifest V3** architecture further improves efficiency by reducing memory usage by 30% while keeping performance impact negligible. User studies also indicate that clear, guided alerts significantly improve compliance, with 78% of users following recommendations.

The system's multi-layered monitoring approach addresses gaps left by traditional single-purpose tools. By tracking form submissions, clipboard usage, paste actions, input changes, and autofill behavior, SecureGuard can detect complex data leakage chains that would otherwise go unnoticed. This integrated monitoring strengthens defense against attacks exploiting multiple interaction points. Importantly, all detection processes are performed locally within the browser, ensuring that sensitive data is never transmitted externally. This proves that strong client-side intelligence can deliver effective security without introducing additional privacy risks.

Privacy is a core design principle of SecureGuard, with all analysis—including pattern matching, contextual checks, and reputation evaluation—executed entirely within the user's browser. This eliminates reliance on cloud processing and avoids exposure to third parties. The user interface supports this approach through progressive disclosure, presenting essential alert information upfront while allowing users to access more details if needed. By doing so, the system treats users as active participants in security rather than passive recipients. However, limitations remain, such as the inability to inspect encrypted traffic before browser decryption, dependence on regularly updated detection patterns, and limited compatibility with non-Chromium browsers.

Future improvements aim to enhance SecureGuard's capabilities by integrating privacy-preserving machine learning for advanced detection beyond fixed patterns. Expanding support to mobile platforms and browsers outside the Chromium ecosystem is also a key goal. Additional enterprise features, such as threat intelligence integration and SIEM support, are planned while maintaining a local-first processing model. Educational enhancements like interactive tutorials, phishing simulations, and gamified learning could further strengthen user awareness. Overall, the project highlights that well-designed browser extensions can achieve high accuracy, strong privacy, and excellent usability, setting a foundation for next-generation browser security solutions.

## VII. REFERENCES


1. Q. Xie, M. V. K. Murali, P. Pearce and F. Li, "Arcanum: Detecting and Evaluating the Privacy Risks of Browser Extensions on Web Pages and Web Content," in Proc. 33rd USENIX Security Symposium (USENIX Security '24), Philadelphia, PA, USA, Aug. 2024, pp. 2145-2162.
2. A. Nayak, R. Khandelwal, E. Fernandes and K. Fawaz, "Experimental Security Analysis of Sensitive Data Access by Browser Extensions," in Proc. ACM Web Conference (WWW '24), Singapore, May 2024, pp. 1523-1536. DOI: 10.1145/3589334.3645683.
3. S. Hsu, M. Tran and A. Fass, "What is in the Chrome Web Store? Investigating Security-Noteworthy Browser Extensions," in Proc. ACM Asia Conference on Computer and Communications Security (ASIACCS '24), Singapore, Jul. 2024, pp. 892-907.
4. K. U. T. Siriwardhana, A. Fathima Sharfana, and R. M. Nayana0jith Rathnayaka, "Real-Time URL-Based Phishing Detection Using XGBoost and Browser Extension Integration," *Journal of Information and Communication Technology (JICT)*, 2026.
5. Y. Nie, Z. Wang, L. Chen, H. Xu and Y. Zhang, "WAScope: Detecting Privacy Data Leakage with Web Application Specific APIs," The Computer Journal, vol. 68, no. 5, pp. 1342-1358, May 2025. DOI: 10.1093/comjnl/bxu156.
6. S. Liu, C. Li, C. Wang, J. Hou, Z. Chen, L. Zhang, Z. Liu, Q. Ye, Y. Hei, X. Zhang, and Z. Wang, "ClawKeeper: Comprehensive Safety Protection for OpenClaw Agents Through Skills, Plugins, and Watchers," *arXiv preprint arXiv:2603.24414*, 2026.
7. S. Agarwal, R. Mrowczynski, M. Hellenthal and B. Stock, "I have no idea how to make it safer: Studying Security and Privacy Mindsets of Browser Extension Developers," in Proc. IEEE Symposium on Security and Privacy (S&P), San Francisco, CA, USA, May 2025, pp. 789-806.
8. Y. Jin, Z. Zhang, Z. Lyu, and J. Ni, "Security, Privacy, and Ethical Risks in OpenClaw," 2026.
9. L. Tyler and I. De Oliveira Nunes, "Towards Browser Controls to Protect Cookies from Malicious Extensions," in Proc. IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria, Jun. 2024, pp. 445-461.
10. A. Guha, M. Fredrikson, B. Livshits, and N. Swamy, "Verified Security for Browser Extensions," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA.
11. A. Cherry, K. Barmpis and S. F. Shahandashti, "The Emperor is Now Clothed: A Secure Governance Framework for Web User Authentication through Password Managers," in Proc. Annual Computer Security Applications Conference (ACSAC), Austin, TX, USA, Dec. 2024, pp. 234-248.
12. C. Lim, B. Jin, and H. Kim, "Stealth Extension Exfiltration (SEE) Attacks: Stealing User Data without Permissions via Browser Extensions," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25)*, 2025. doi:10.1145/3672608.3707856.
13. L. Thaqi, A. Halili, K. Vishi and B. Rexha, "NoPhish: Efficient Chrome Extension for Phishing Detection Using Machine Learning Techniques," IEEE Access, vol. 12, pp. 89234-89248, 2024. DOI: 10.1109/ACCESS.2024.3421567
14. P. J. Ortiz and R. Gomez, "User-centric Design for Threat Analytics Dashboards in Browser Extensions," in Proc. IEEE International Conference on Human-Computer Interaction (HCI), Copenhagen, Denmark, Sep. 2023.
15. J. K. Lee, S. Park, M. Kim and H. Choi, "Machine-Learning Assisted Data-Leak Detection in Web Applications," in Proc. IEEE International Conference on Big Data (Big Data), Sorrento, Italy, Dec. 2024, pp. 1823-1838.
16. V. Singh and T. Mehta, "Runtime Monitoring of Clipboard and Form Data in Browser Extensions," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 4, pp. 2456-2471, Jul. 2024. DOI: 10.1109/TDSC.2024.3367891.
17. M. S. Allauddin and P. S. Lokhande, "Analyzing Security Risks in Browser Extension Search Tools: A Literature Review," Journal of Cybersecurity Research, vol. 9, no. 2, pp. 145-162, Apr. 2024.
18. S. Yarram, S. R. Bittla, and P. K. Pareek, "Intelligent Malicious URL Detection Using Extreme Multi-Label Text Classification for Phishing Prevention," in *2025 International Conference on Computing for Sustainability and Intelligent Future (COMP-SIF)*, 2025.
19. D. Akhawe and A. P. Felt, "Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness," in Proc. 22nd USENIX Security Symposium, Washington, DC, USA, Aug. 2013, pp. 257-272.
20. W3C Web Application Security Working Group, "Content Security Policy Level 3," W3C Working Draft, Oct. 2021. [Online]. Available: https://www.w3.org/TR/CSP3/