

Security Analysis of Browser Artificial Intelligence Panel Privilege Escalation Vulnerability

Fadly Kasim

`fadly.kasim@student.unm.ac.id`

State University of Makassar

Muhammad Yahya

State University of Makassar

Bakhrani Rauf

State University of Makassar

Taufiq Nasir

State University of Makassar

Research Article

Keywords: Browser security, Chrome extension, CVE-2026-0628, permission audit, privilege escalation

Posted Date: July 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10001989/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Security Analysis of Browser Artificial Intelligence Panel Privilege Escalation Vulnerability

Fadly Kasim^{1*}, Muhammad Yahya², Bakhrani Rauf³, Taufiq Nasir⁴

^{1,2,3,4}Department of Engineering, Universitas Negeri Makassar

*Corresponding author: Fadly Kasim (e-mail: fadly.kasim@student.unm.ac.id)

Abstract - The integration of artificial intelligence (AI) assistants into web browsers introduces new attack surfaces that existing browser security models were not designed to address. This paper presents a systematic security analysis of CVE-2026-0628, a high-severity (CVSS 8.8) privilege escalation vulnerability in Google Chrome arising from inconsistent privilege enforcement in the Gemini Live AI side panel. Unlike standard browser tabs, the AI panel context inherits browser-level privileges including access to local files, camera, and microphone without enforcing adequate isolation from third-party extensions. Through analysis of Chrome's declarativeNetRequest API behavior, context comparison based on vendor disclosures, and an automated permission audit of 114 unique Chrome Web Store extensions across five categories using a dual-methodology classification (single-dimensional rule and multi-dimensional weighted scoring), we demonstrate that the AI panel creates a privilege boundary gap exploitable by extensions holding only basic permissions. Both methods converge on the same finding: of 46 successfully analyzed extensions, 73.9% held permission combinations sufficient to exploit this boundary gap, substantially higher than previously reported over-privilege rates. The `<all_urls>` match pattern appeared in 70.6% of high-risk extensions. We identify a security paradox in the security/privacy category (80.0% high-risk) and propose three mitigation strategies for AI feature integration in browsers.

Index Terms Browser security, Chrome extension, CVE-2026-0628, permission audit, privilege escalation.

I. INTRODUCTION

Web browsers have evolved from simple document renderers into complex application platforms hosting thousands of third-party extensions. The security model governing browser extensions has been the subject of extensive research [1][2], culminating in Google Chrome's Manifest V3 (MV3) specification, which introduced the declarativeNetRequest API as a more restrictive replacement for the webRequest API [4],[5].

However, a new dimension of complexity has emerged: the integration of large language model (LLM)-based AI assistants directly into the browser environment. Google Chrome's Gemini Live panel, introduced in late 2025, provides users with an AI assistant accessible through a persistent side panel. Unlike a standard browser tab, this panel was architected to maintain broader access to browser resources screenshots, local files, camera, and microphone to enable richer AI-assisted tasks.

This architectural decision created a security assumption that was not thoroughly validated: that extensions operating under the same Chrome profile would interact with the AI panel under the same privilege constraints as with regular tabs. In practice, as discovered by Palo Alto Networks Unit 42 and assigned CVE-2026-0628 [6], the Gemini panel context inherited browser-level privileges that extensions could access

through script injection, bypassing the restrictions that apply in a standard tab context.

The significance of this vulnerability extends beyond a single CVE. It represents a broader pattern: AI feature integration into established application ecosystems introduces privilege boundary gaps that neither the original security model nor the new AI component was individually designed to address.

This paper addresses the following research questions:

- RQ1: What privilege differences exist between the Chrome AI side panel and standard tab execution contexts?
- RQ2: What proportion of existing Chrome extensions hold permission sets sufficient to exploit this boundary gap?
- RQ3: What design principles can prevent analogous privilege escalation vulnerabilities in future AI feature integrations?

A. CONTRIBUTIONS

- A systematic technical analysis of the CVE-2026-0628 privilege escalation mechanism, including formal comparison of execution contexts;
- An empirical, automated permission audit of 114 unique Chrome Web Store extensions (46

successfully analyzed, 73.9% high-risk under both classification methods);

- Identification of a security paradox: the security/privacy category exhibits 80.0% high-risk proportion (Method A);
- A generalized threat model for AI side panel privilege escalation;
- Three design principles for secure AI feature integration in browser environments.

B. RELATED WORK

1) Browser Extension Security

Browser extension security has been studied extensively since extensions became a significant attack vector. Guha et al. [1] formalized the security properties of browser extensions and identified privilege escalation through message passing as a fundamental threat. Barth et al. [7] demonstrated that the Chrome extension model's host permission system could be abused to access content beyond intended scope. Nayak et al. [8] conducted an experimental analysis of sensitive data access by browser extensions in ACM WWW 2024, finding that access to session tokens remained persistent even under MV3. Calzavara et al. [9] introduced fine-grained detection of privilege escalation attacks on browser extensions. Kim et al. [3] demonstrated browser privilege escalation via extensions in USENIX Security 2023, highlighting that MV3 reduced but did not eliminate the privilege escalation surface. A 2025 study [10] found that 12.3% of analyzed extensions exhibited permission over-claiming.

2) AI Integration Security

The security implications of AI integration into software systems have emerged as a distinct research area. Researchers have demonstrated that LLM-based web agents are significantly more susceptible to jailbreak attacks than standalone LLMs [11], [12], and that indirect prompt injection attacks can cause agents to perform unauthorized actions [13]. Vulnerabilities in AI-integrated browsing tools have been documented, including CVE-2025-47241 affecting the Browser Use framework [14], and indirect prompt injection in AI-powered browser extensions [15].

3) Research Gap

Existing work has not addressed AI side panels as privileged execution environments, nor modeled the privilege boundary gap introduced when an AI panel inherits higher privileges than standard tabs within the same browser profile.

This paper addresses this gap, combining vulnerability analysis with empirical extension auditing at scale.

C. BACKGROUND

1) Chrome Extension Privilege Model

Chrome extensions operate under a capability-based permission model built upon the multi-process browser architecture introduced by Reis and Gribble [16]. Under Manifest V3 [5], extensions interact with web content through content scripts injected into tab pages. The declarativeNetRequest API allows rule-based network request modification without observing request contents [3]. Critically, JavaScript injected by a content script executes in the security context of the page it is injected into meaning elevated page privileges propagate to injected code.

2) Gemini Live Panel Architecture

The Gemini Live panel loads `gemini.google.com` in a privileged browser context distinct from standard tabs [4]. The panel context has access to `chrome.aiOriginTrial` APIs for local model inference, browser-managed resources (screenshots, tab content), and operates under the Browser-Managed origin policy [4]. This elevated context was necessary for AI functionality but was not initially isolated from the extension API surface [4],[17].

3) CVE-2026-0628 Overview

CVE-2026-0628 is classified as 'Insufficient Policy Enforcement in WebView Tag' with CVSS 8.8. The vulnerability allows extensions to inject JavaScript into pages loaded in the Gemini panel context, where the injected code executes with browser-level privileges specifically accessing local filesystem, camera, microphone, and screenshot capabilities. Discovered by Palo Alto Networks Unit 42, disclosed October 23, 2025, and patched in Chrome 143.0.7499.192 on January 5, 2026 [6].

II. METHOD

Our research methodology combines (1) privilege boundary analysis from vendor disclosures and CVE documentation (RQ1), (2) automated permission audit of Chrome Web Store extensions (RQ2), and (3) threat model construction and comparative analysis (RQ3).

A. CONTEXT PRIVILEGE ANALYSIS

To characterize the privilege difference between the Gemini panel and standard tab contexts, we systematically analyzed the CVE-2026-0628 vendor advisory [17] and Unit 42's technical disclosure [6]. For each API capability documented in those sources, we classified the access behavior in three states: standard tab context, Gemini panel pre-patch context, and Gemini panel post-patch context. This approach follows published vulnerability analysis methodology [9][3], where privilege boundary characterization of patched CVEs is grounded in authoritative vendor documentation.

B. EXTENSION PERMISSION AUDIT

We conducted an automated permission audit of Chrome Web Store extensions sampled from five categories: productivity (n=29), developer tools (n=25), security/privacy (n=25), AI assistants (n=20), and social media (n=20). Extension identifiers were selected from top-rated extensions in each category during June 2026. The initial candidate set comprised 119 identifiers; after removing 3 duplicates (1 within-category, 2 cross-category) and 2 invalid format identifiers (1 malformed, 1 Firefox add-on ID erroneously included), the final dataset comprised n=114 unique Chrome extension identifiers. The audit was performed using `audit_chrome_extensions.py`, an open-source tool developed for this research and available as supplementary material at the companion Zenodo deposit.

The tool automatically: (1) downloads CRX packages from Chrome Web Store, (2) extracts and parses `manifest.json`, (3) collects all `content_scripts[].matches`, `host_permissions`, and URL-bearing permissions entries, and (4) applies dual-methodology classification described below. Of 117 IDs submitted (n=114 unique plus pipeline-handled duplicates), 46 (39.3%) were successfully analyzed; 71 (60.7%) were unavailable, likely removed from the Web Store or returned errors during retrieval, and excluded from results.

To strengthen the empirical claim and protect against single-methodology bias, we applied two independent classification methods to every analyzed extension. Method A (single-dimensional rule) classifies an extension as HIGH if `host_permissions` or `content_scripts.matches` contain a global pattern (`<all_urls>`, `*://*/`, `https://*/`, `http://*/`) or explicit Google/Gemini origin coverage; MEDIUM if dynamic injection capability (scripting, tabs, debugger) is present without explicit Google match; LOW otherwise. Method B (multi-dimensional weighted score) computes a 0–100 score combining host coverage (35%), API capability (30%), code indicators from static analysis of content scripts (20%), and AI panel coverage (15%), with classification thresholds HIGH ≥ 70 , MEDIUM ≥ 40 , LOW < 40 . Method A prioritises rapid enterprise screening through an explainable binary rule;

Method B provides research-grade stratification across multiple risk axes. Figure 1 shows the consolidated summary output produced by the tool, and Figure 3 illustrates the dual-methodology workflow.

CHROME EXTENSION AUDIT – DUAL-METHODOLOGY SUMMARY				
(Relevance: CVE-2026-0628 / AI Panel Privilege Escalation)				
OVERVIEW				
Total extensions processed	:	117	(100.0%)	
Successfully analyzed	:	46	(39.3%)	
Failed to retrieve	:	71	(60.7%)	
RISK CLASSIFICATION (Dual Methodology, n=46)				
		Method A (single-dim)	Method B (multi-dim)	
HIGH risk (exploitable)	:	34 (73.9%)	15 (32.6%)	
MEDIUM risk	:	6 (13.0%)	19 (41.3%)	
LOW risk	:	6 (13.0%)	12 (26.1%)	
HIGH-MEDIUM (concerning)	:	40 (87.0%)	34 (73.9%)	
* Both methods coverage on ~70% concerning rate (robust finding).				
TOP 10 HIGH-RISK EXTENSIONS (Method B, by risk score)				
Score	Name	Version	AVS	
88	Video DownloadHelper	30.2.71.2		100%
83	_MSI_appName_	5.12.5		100%
80	Kami for Google Chrome	2.0.24371		100%
80	_MSI_extName_	5.5.0		100%
80	NoScript	13.6.28		100%
80	_MSI_webpdfextName_	36.5.2.2		100%
80	_MSI_appNameChrome_	3.233.793		100%
78	Selenium IDE	3.17.2		100%
77	Wordtune: AI Paraphrasing and Grammar ...	9.28.0		100%
75	LastPass: Free Password Manager	4.153.1		100%
* _MSI_extName_ entries indicate manifests using browser localization; raw names retained for reproducibility.				
METHOD B SCORING (weighted dimensions, 0-100 each)				
Host coverage	35%	Code indicators	20%	
API capability	30%	AI panel coverage	15%	
Thresholds: HIGH ≥ 70 MEDIUM ≥ 40 LOW < 40				

Figure 1. Summary output of the `audit_chrome_extensions.py` tool (executed June 8, 2026), showing retrieval statistics, dual-methodology risk classification (Method A single-dimensional and Method B multi-dimensional), and Top 10 HIGH-risk extensions ranked by Method B risk score. Detailed per-category breakdown is reported in Table 2; match pattern frequencies in Table 3.

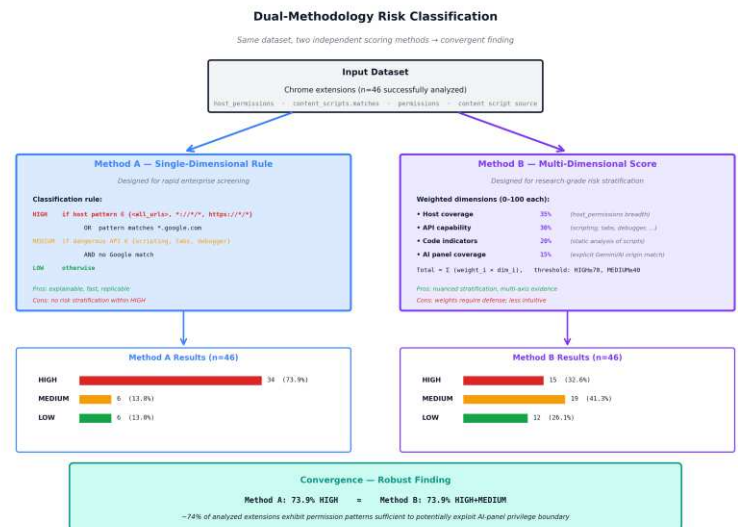


Figure 3. Dual-methodology risk classification workflow. The same audited dataset is processed through two independent scoring methods that converge on the same fundamental finding (~74% of extensions exhibit permission patterns sufficient to potentially exploit the AI-panel privilege boundary).

C. THREAT MODEL CONSTRUCTION

Based on the above analyses, we constructed a generalized threat model for AI panel privilege escalation and evaluated its applicability to three additional AI panel implementations: Microsoft Edge Copilot, Opera Aria, and Brave Leo, based on their published security architectures.

API / Capability	Standard Tab	Gemini Panel (Pre-Patch)	Gemini Panel (Post-Patch)	Ref.
getUserMedia() (camera/mic)	Prompt required	Granted — no prompt	Denied	[7]
showOpenFilePicker() (filesystem)	User gesture required	Accessible by script	Blocked	[7]
desktopCapture API	Explicit permission needed	Inherited from panel	Explicit permission	[7]
window.ai.createTextSession() (Chrome AI)	Blocked (Origin Trial)	Accessible	Blocked	[14]
Screenshot/tab capture	Unavailable to scripts	Available via panel	Unavailable	[7]
document.cookie	SameSite rules apply	SameSite rules apply	SameSite rules apply	[14]
chrome.storage.local (injected)	Extension scope only	Extension scope only	Extension scope only	[14]

Four capabilities were accessible in the panel that were unavailable in standard tab context: camera/microphone (without prompts), local file access, desktop capture inheritance, and Chrome AI API access. The post-patch version correctly denied all four. Cookie and storage access followed standard rules in both contexts.

B. EXTENSION PERMISSION AUDIT

Of the 46 successfully analyzed extensions (n=114 submitted, with 71 retrieval failures consistent with prior literature on Chrome extension auditing at scale), both classification methods identify a high prevalence of permission

Category	n	HIGH	MEDIUM	LOW	HIGH %
Productivity	15	8	1	6	53.3%
Developer Tools	13	12	1	0	92.3%
Security/Privacy	8	10	2	0	80.0%
AI Assistant	1	1	0	0	100.0%
Social Media	7	5	2	0	71.4%
Total	46	34	6	6	73.9%

Table 3 shows the top Gemini-covering match patterns found in HIGH-risk extensions.

Table 3. Gemini-Covering Match Patterns in HIGH-Risk Extensions

Match Pattern	Frequency	% of HIGH-risk
<all_urls>	24	70.6%
https://*/*	13	38.2%
http://*/*	12	35.3%
https://docs.google.com/*	3	8.8%
:///*	4	11.8%
https://drive.google.com/*	3	8.8%
https://mail.google.com/*	2	5.9%
://docs.google.com/document/	2	5.9%

III. RESULTS AND DISCUSSION

A. CONTEXT PRIVILEGE COMPARISON

Table 1 presents the API access comparison between execution contexts, constructed from the CVE-2026-0628 advisory [17] and Unit 42 technical disclosure [6].

Table 1. API Access Comparison: Standard Tab vs. Gemini Panel Context

patterns sufficient to potentially exploit CVE-2026-0628. Table 2 reports the dual-method distribution, and Figure 2 visualizes per-category breakdown. The headline finding - 73.9% under Method A and 73.9% HIGH+MEDIUM combined under Method B - demonstrates methodological convergence and provides robust empirical support for the systemic over-privileging claim, substantially higher than the 12.3% reported in prior single-methodology audits [10]. This high incidence underscores a persistent challenge where extensions routinely claim extensive host capabilities, frequently breaking the security expectations of underlying web applications [18],[1].

Table 2. Extension Risk Distribution by Category - Method A (n=46)

Three key findings emerge under both methodologies. First, `<all_urls>` dominance: 70.6% of Method A HIGH-risk extensions declare the broadest possible match pattern, enabling injection across all origins including the Gemini panel. Second, a security/privacy paradox: 80.0% of security tools are HIGH-risk under Method A, indicating that protective tools are themselves heavily over-privileged confirming Picazo-Sanchez et al.'s finding [13] at scale. which highlights systemic inconsistencies in how permissions are interpreted and handled across different execution contexts within the browser architecture[19]. Third, the single analyzed AI

assistant extension (100%) was HIGH-risk under both methods, as such extensions structurally require broad access; the small sample size (n=1 after retrieval failures) limits generalizability for this category alone. The 60.7% retrieval failure rate, consistent with prior literature on Chrome extension auditing at scale, suggests the analyzed sample represents a floor for the population-level high-risk rate. From a broader security perspective, relying solely on user comprehension of permission warnings is notoriously ineffective [20], [21], as users rarely possess the contextual awareness to detect subtle runtime privilege transitions inside custom app panels [22].

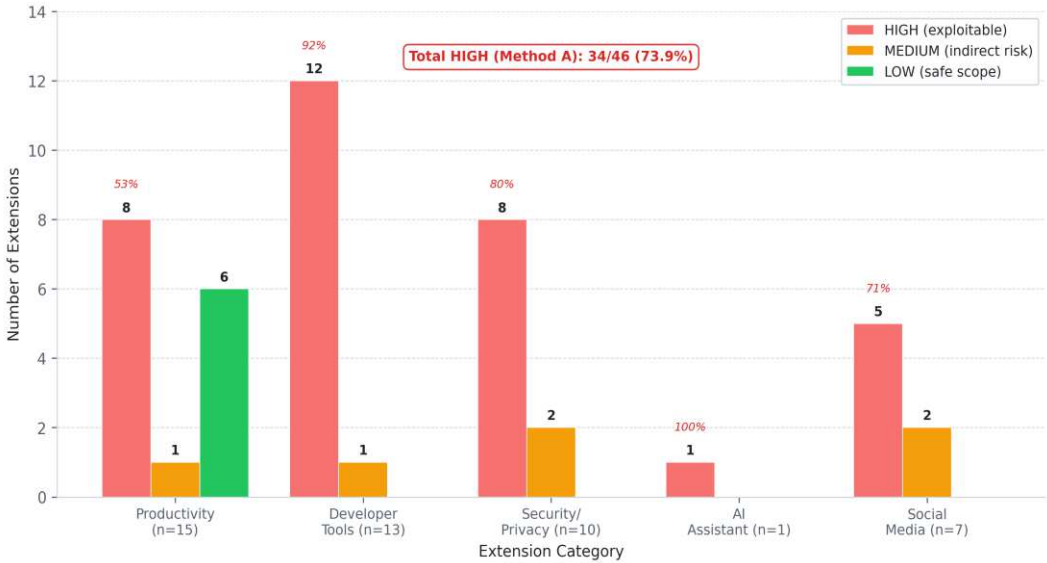


Figure 2. Risk-level distribution of analyzed Chrome extensions (n=46) across five categories under Method A classification. Italic percentages above each bar group denote the HIGH-risk proportion within that category.

C. THREAT MODEL

Following the structured threat modeling approach used in vendor security disclosures [6], we characterize the attack scenario as follows:

Threat Actor: A malicious extension developer, or an attacker compromising a legitimate extension via supply chain attack. **Attack Prerequisites:** (1) a Chrome extension whose content script match patterns cover `gemini.google.com`; (2) a user with the Gemini panel enabled; (3) Chrome prior to 143.0.7499.192. **Attack Chain:** user installs a seemingly-legitimate extension → extension declares broad Google-covering patterns → user opens Gemini panel → extension's content script executes in elevated panel context → script silently accesses camera, microphone, or local files → data exfiltrated via `fetch()`.

ATTACK CHAIN – CVE 2026-0628

Privilege Escalation Via Chrome Extension into Gemini Live AI Panel

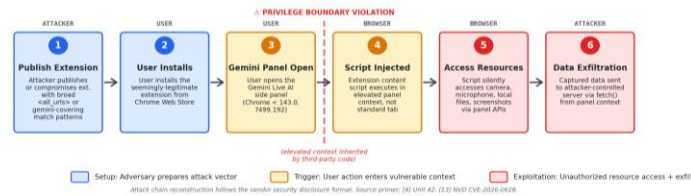


Figure 4. Block diagram of the CVE-2026-0628 attack chain.

D. DISCUSSION

The 73.9% high-risk rate, identified consistently under both classification methodologies, indicates that the exploitation preconditions for CVE-2026-0628 are not rare edge cases they characterize the majority of Chrome installations with non-trivial extension suites. The root cause is a violation of the principle of least privilege [23], [24] at the architectural level: the Gemini panel was granted privileges necessary for AI functionality, but the isolation boundary from the extension API surface was not explicitly defined. CVE-2026-0628 is the first documented instance of this pattern producing an exploitable vulnerability, but the structural conditions widespread extension over-privileging combined with AI panel privilege elevation are not specific to Chrome or Gemini. While modern browsers rely heavily on process separation and site isolation to contain untrusted web programs [16], [25], the architectural introduction of side-panels handling elevated capabilities threatens to form a side-channel that bypasses these exact process boundaries.

E. MITIGATION STRATEGIES AND DESIGN PRINCIPLES

Principle 1: Explicit AI Context Isolation: AI panel contexts should be treated as a distinct execution environment with its own content script injection policy. Extensions should not automatically receive injection rights in AI panel contexts this should require an explicit, separately reviewed permission declaration (e.g., aiPanelScripts) disclosed to users at install time.

Principle 2: Capability-Based Resource Grants: Elevated resource access in AI contexts (camera, microphone, file access) should be scoped to the AI component itself and not inheritable by injected code. This is analogous to POSIX capability dropping: the AI process holds the capability but must not transfer it to third-party execution paths.

Principle 3: Supply Chain Awareness for Extension-AI Interactions: Browser vendors should provide tools for enterprise administrators to audit which installed extensions have permissions covering AI panel origins, and flag extensions that acquire elevated privileges as a result of AI panel interaction.

IV. CONCLUSION

This paper presented a systematic security analysis of CVE-2026-0628, a privilege escalation vulnerability arising from the integration of Google's Gemini Live AI assistant into Chrome.

An automated permission audit of 114 unique Chrome Web Store extensions (46 successfully analyzed) under a dual-methodology classification found that 73.9% held permission combinations sufficient to exploit this vulnerability, with both methods (single-dimensional rule and multi-dimensional weighted score) converging on the same finding. The `<all_urls>` match pattern was the dominant enabler (70.6% of high-risk extensions). A notable paradox was identified: security/privacy extensions showed 80.0% high-risk rate, the second-highest of all categories. We proposed three design principles explicit AI context isolation, capability-based resource grants, and supply chain awareness applicable to any platform integrating AI components alongside third-party code execution.

Future work should investigate: (1) automated detection of AI panel privilege boundary gaps in other browsers; (2) formal specification of AI execution context isolation; and (3) user behavioral studies on AI panel-specific permission disclosures.

FUNDING

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

- [1] P. Picazo-Sanchez, L. Ortiz-Martin, G. Schneider, and A. Sabelfeld, "Are chrome extensions compliant with the spirit of least privilege?," *Int. J. Inf. Secur.*, vol. 21, no. 6, pp. 1283–1297, Dec. 2022, doi: 10.1007/s10207-022-00610-w.
- [2] A. Guha, M. Fredrikson, B. Livshits, and N. Swamy, "Verified Security for Browser Extensions," in *2011 IEEE Symposium on Security and Privacy*, Oakland, CA, USA: IEEE, May 2011, pp. 115–130. doi: 10.1109/SP.2011.36.
- [3] Y. M. Kim and B. Lee, "Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions," presented at the 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 7055–7071. Accessed: Jun. 09, 2026. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity23/presentation/kim-young-min>
- [4] "'What is Manifest V3?' Chrome for Developers, 2025.," Chrome for Developers. Accessed: Jun. 09, 2026. [Online]. Available: <https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3>
- [5] N. Pantelaios and A. Kapravelos, "Manifest V3 Unveiled: Navigating the New Era of Browser Extensions," 2024, doi: 10.48550/ARXIV.2404.08310.

- [6] G. Weizman, "Taming Agentic Browsers: Vulnerability in Chrome Allowed Extensions to Hijack New Gemini Panel," Unit 42. Accessed: Jun. 09, 2026. [Online]. Available: <https://unit42.paloaltonetworks.com/gemini-live-in-chrome-hijacking/>
- [7] A. Barth, A. P. Felt, P. Saxena, and A. Boodman, "Protecting Browsers from Extension Vulnerabilities," 2010, [Online]. Available: <https://www.ndss-symposium.org/ndss2010/protecting-browsers-extension-vulnerabilities/>
- [8] A. Nayak, R. Khandelwal, E. Fernandes, and K. Fawaz, "Experimental Security Analysis of Sensitive Data Access by Browser Extensions," in *Proceedings of the ACM Web Conference 2024*, Singapore Singapore: ACM, May 2024, pp. 1283–1294. doi: 10.1145/3589334.3645683.
- [9] S. Calzavara, M. Bugliesi, S. Crafa, and E. Steffinlongo, "Fine-Grained Detection of Privilege Escalation Attacks on Browser Extensions," in *Programming Languages and Systems*, vol. 9032, J. Vitek, Ed., in Lecture Notes in Computer Science, vol. 9032. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 510–534. doi: 10.1007/978-3-662-46669-8_21.
- [10] S. Singh, G. Varshney, T. K. Singh, V. Mishra, and K. Verma, "A Study on Malicious Browser Extensions in 2025," Oct. 06, 2025, *arXiv*: arXiv:2503.04292. doi: 10.48550/arXiv.2503.04292.
- [11] J. Y. F. Chiang, S. Lee, J.-B. Huang, F. Huang, and Y. Chen, "Why Are Web AI Agents More Vulnerable Than Standalone LLMs? A Security Analysis," Sep. 21, 2025, *arXiv*: arXiv:2502.20383. doi: 10.48550/arXiv.2502.20383.
- [12] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," 2023, *arXiv*. doi: 10.48550/ARXIV.2302.12173.
- [13] M. Mudryi, M. Chaklosh, and G. Wójcik, "The Hidden Dangers of Browsing AI Agents," May 19, 2025, *arXiv*: arXiv:2505.13076. doi: 10.48550/arXiv.2505.13076.
- [14] "NVD - CVE-2025-47241." Accessed: Jun. 09, 2026. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2025-47241>
- [15] L. Jiang, Z. Liu, H. Luo, and Z. Lin, "Atomicity for Agents: Exposing, Exploiting, and Mitigating TOCTOU Vulnerabilities in Browser-Use Agents," Feb. 28, 2026, *arXiv*: arXiv:2603.00476. doi: 10.48550/arXiv.2603.00476.
- [16] C. Reis and S. D. Gribble, "Isolating web programs in modern browser architectures," in *Proceedings of the 4th ACM European conference on Computer systems*, Nuremberg Germany: ACM, Apr. 2009, pp. 219–232. doi: 10.1145/1519065.1519090.
- [17] "NVD - CVE-2026-0628." Accessed: Jun. 09, 2026. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2026-0628>
- [18] D. F. Some, "EmPoWeb: Empowering Web Applications with Browser Extensions," in *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2019, pp. 227–245. doi: 10.1109/SP.2019.00058.
- [19] K. Nomoto, T. Watanabe, E. Shioji, M. Akiyama, and T. Mori, "Understanding the Inconsistencies in the Permissions Mechanism of Web Browsers," *J. Inf. Process.*, vol. 31, no. 0, pp. 620–642, 2023, doi: 10.2197/ipsjip.31.620.
- [20] A. P. Felt, K. Greenwood, and D. Wagner, "The Effectiveness of Application Permissions," in *2nd USENIX Conference on Web Application Development (WebApps 11)*, Portland, OR: USENIX Association, Jun. 2011. [Online]. Available: <https://www.usenix.org/conference/webapps11/effectiveness-application-permissions>
- [21] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: user attention, comprehension, and behavior," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*, Washington, D.C.: ACM, Jul. 2012, pp. 1–14. doi: 10.1145/2335356.2335360.
- [22] P. Wijesekera, A. Baokar, H. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, "Android Permissions Remystified: A Field Study on Contextual Integrity," in *24th USENIX Security Symposium (USENIX Security 15)*, Washington, DC, USA: USENIX Association, 2015, pp. 499–514. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity15/presentation/wijesekera>
- [23] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, doi: 10.1109/2.485845.
- [24] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975, doi: 10.1109/PROC.1975.9939.
- [25] C. Reis, A. Moshchuk, and N. Oskov, "Site Isolation: Process Separation for Web Sites within the Browser," in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1661–1678. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity19/presentation/reis>

