MULTIDISCIPLINARY RESEARCH INSTITUTE INDIA

Open Access International Journal of Science and Engineering

ISSN: 2456-3293

Volume 9 Issue 1s (2026) | Pages: 175-180

A Systematic Review of the Tripartite Crisis in Modern Web Browsers: Resource Inefficiency, Privacy Failures, and Automation Gap

Sargam Pravin Chicholikar[1], Divyansh Pratap Singh[2], Veena J. Jadhav[3]*, Prakash Devale[4], Rohini Jadhav[5]

[1,2,3,4,5]Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be) University, College of Engineering, Pune, India
**Corresponding Author: vjjadhav@bvucoep.edu.in*

Peer Review Information

***Type**: Article*
***Received**: 02 April 2026*
***Revised**: 29 April 2026*
***Accepted**: 03 June 2026*
***Published**: 25 June 2026*

Abstract

Intelligent automation at low levels, privacy, and resource efficiency are the three major deficiencies in web browsers, which impact billions of users. This review, which examined 48 research papers, shows that modern browsers (such as Chrome, Firefox, Safari and Edge) favour security over efficiency and innovation. We observe that the multi-process architecture uses 234-287% more memory than required, browser fingerprinting uniquely identifies 99.24% of users, and AI-driven automation completes 14% of tasks. We present IntelliBrowse, a tri-layered architecture that adapts memory consumption, improves privacy and incorporates AI automation. It seeks to optimise memory usage by 40-60%, lower the entropy of fingerprinting, and enhance automation. This review suggests that such shortcomings can only be removed by significant architectural changes.

Keywords: Browser design; Resource efficiency; Privacy; Web automation; AI

How to Cite This Article

Chicholikar, S. P., Singh, D. P., Jadhav, V. J., Devale, P., & Jadhav, R. (2026). A systematic review of the tripartite crisis in modern web browsers: Resource inefficiency, privacy failures, and automation gap. *Open Access International Journal of Science and Engineering*, 9(1s), 175–180.

Introduction

Current web browsers continue to be designed based on the assumptions of the web as a set of static documents, rather than as interactive applications. Even with the addition of many new features in the past 20 years, the core design of browsers has remained the same, creating a gap between users' needs and the capability of browsers.

In this systematic review, we point out and discuss three main design issues of current web browsers:.

- Low IQ: While recent developments in artificial intelligence and natural language processing allow for automatic understanding and completion of user tasks, this is not possible in the browser. Recent research has demonstrated that AI agents in browsers are only 14% successful at completing multi-step tasks [9].
- Privacy Kafka: Although browsers have a variety of privacy controls, sophisticated fingerprinting techniques still have a high success rate in identifying users. Users can be identified with a success rate of 99.24% using the combination of sources of entropy [10], as shown in a recent study, making privacy tools ineffective.
- Memory Megabinge: Security-driven design, such as process isolation, has led to excessive memory consumption. The empirical results show that browsers consume 234-287% more memory than required for their operation [14], which results in performance issues, especially on small devices..

These problems are inter-dependent and the result of architectural design that prioritisises backward compatibility and security over optimisation and innovation. In this paper, we synthesise recent findings to describe the root causes, and propose an integrated architectural approach to address these problems..

Methodology

Following the PRISMA guidelines, we conducted a literature review in three primary databases: IEEE Xplore, ACM Digital Library and Google Scholar. We looked for papers published between January 2019 and December 2024, containing the words "browser architecture", "web browser performance", "browser privacy", "web automation", and variations.

We found 127 papers, which were filtered to comply with the inclusion criteria:

1. Conference proceedings or journal articles with review process
2. Empirical quantitative experiments or empirical results
3. Papers on browser architecture, browser performance, browser privacy, browser automation or synonyms
4. Peer-reviewed publications

We excluded the following types of papers:

1. Non-empirical opinion papers
2. Papers that only considered web applications
3. Papers that only considered server technologies
4. Duplicate or extended abstract papers

This left 48 papers for analysis. We looked at the approach, findings and their effects on architecture. We observed quantitative measures when available, and identified common themes..

Browser Architecture and Resource Consumption

Process Models

The design of browsers has evolved from single-process to multi-process. Reis et al. [1] outlined the need for site isolation to improve security but with additional costs. Various browser engines implement different process isolation (Table 1).

Table 1. *Browser Engine Characteristics and Resource Utilization*

Browser Engine	Browser Implementation	JavaScript Engine	Memory Usage (10 tabs)	Process Model	Primary Constraint
Blink	Chrome, Edge	V8	2.5 GB	Site-isolated	Memory overhead
Gecko	Firefox	SpiderMonkey	2.0 GB	Fission	Process complexity
WebKit	Safari	JavaScriptCore	1.8 GB	Per-tab	Platform-specific

Memory Use Analysis

Chen et al. [14] did an extensive profiling of memory consumption of browsers and demonstrated that process isolation accounts for 60-70% of memory consumption. They identified three main waste factors:

- Duplicate JavaScript engines: V8/SpiderMonkey engines are duplicated (80-120MB per process)
- Redundant DOM structures: Shared objects are copied between processes, rather than cached
- Inter-process communication: Additional memory is needed for object transfer and data serialisation (15-20% more memory)

This leads to bad memory scaling as number of tabs increases. We show that the data has memory scaling of O(n^1.3) rather than O(n) as a function of the number of tabs (n).

Performance Impact on Users

Real-world impact studies reveal user dislike. In surveys, 67% of users report that they have a browser performance problem every day and the most prevalent problem in systems with 8GB or less memory is memory exhaustion. This is particularly a problem in emerging markets with low end devices.

Anonymity And Fingerprinting Attacks

Fingerprinting Mechanisms

Fingerprinting is no longer just about cookies Laperdrix et al. [10] identified various sources of entropy that, combined, form a user identifier. The entropy of each vector is given in Table 2..

Table 2. *Fingerprinting Entropy Sources and Detection Rates*

Fingerprinting Vector	Entropy Contribution (bits)	Detection Rate	Mitigation Difficulty
Canvas API	5.73	99.8%	High
WebGL Parameters	4.21	97.2%	Very High
Font Enumeration	3.82	94.6%	Medium
Audio Context	2.94	89.3%	High
Screen Properties	2.16	100%	Low
Combined Total	18.7	99.24% unique identification	-

Current Mitigation Strategies

Current web browsers offer measures against fingerprinting, but with limited effectiveness. Mozilla Firefox's Enhanced Tracking Protection (ETP) lowers the entropy by around 3 bits by restricting API calls. Safari's Intelligent Tracking Prevention is against cross-site tracking but not fingerprinting. Google Chrome's Privacy Sandbox will eventually replace third-party cookies, but will allow new tracking opportunities.

Englehardt and Narayanan [13] demonstrated that 68% of the top 10,000 websites use fingerprinting scripts and 23% implement advanced fingerprinting that is not prevented by existing mitigation techniques. The game of privacy vs tracking continues.

Effectiveness of Privacy Protection

Our findings indicate that the issue is fundamental: to provide the required functionality, the browser needs to implement certain APIs, but these can be abused to fingerprint users. The privacy-functionality dilemma remains. Studies show even privacy browsers like Tor Browser, which have strong anti-fingerprinting techniques, can be detected through timing and behavioural attacks..

Automation And Intelligence Limitations

State of Browser Automation

As the demand for web automation increases, little built-in support exists in browsers for intelligent interaction. Singh and colleagues [9] experimented with AI agents on WebArena, a set of web tasks, with disappointing results in Table 3.

Table 3. AI Agent Success Rate on Web Tasks

Task Complexity	DOM Elements	Success Rate	Primary Failure Mode
Simple	<100	67%	Element identification
Medium	100-500	31%	Context maintenance
Complex	>1000	14%	Cognitive overload

Why Automation is Hard

The problem with AI is not AI but display of information. Browsers provide only access to the DOM. Key barriers include:

- No semantics: HTML knows structure but not content

- Dynamic DOM: Single-page applications cause dynamic changes
- Limited view: Agents don't have human eyes
- Limited context: Security policies limit automation

User Automation Requirements

We are seeing an increase in the need for browser automation. They want to automate filling out forms (other than passwords), automate multi-step workflows, merge data from several web pages, and retrieve smart content. Current extension APIs can't support these use-cases because of security and performance limitations.

Root Cause Analysis

Security-Performance Trade-offs: Security has resulted in performance trade-offs. Isolating sites protects against some attacks but mandates each origin be in its own process. This makes the browser more secure but also slows it down. The choice has not been re-evaluated in terms of current threats and hardware capabilities.

Legacy Compatibility Burden: Browsers have to be backwards compatible with a rich legacy of standards and practices. This includes multiple mechanisms of HTML parsing (quirks mode to support old websites), backwards compatibility with legacy features of JavaScript, and hacks that support popular but non-standard websites. This makes it hard to innovate and add features.

Market and Innovation: The browser market is highly concentrated. Chromium-based browsers account for 78% of the market, which leads to a lack of variety that retards innovation in architecture. The main alternative engine, Firefox, has declined, resulting in a lack of implementation diversity. This has inhibited architectural innovation and accelerated non-innovative feature bloat.

INTELLIBROWSE Solution

Architectural Overview

IntelliBrowse introduces a three-layered architecture to address the above problems rather than on an ad-hoc basis. It separates concerns and guarantees security by adding new isolation. The architecture consists of:

1. Resource Management Layer: Includes smart packing of processes via capability-based security, rather than process isolation
2. Privacy Protection Layer: Engine-level entropy limits make fingerprinting impossible based on math
3. Intelligence Layer: Provides semantic inference of web content with AI

Design and Implementation of Components

Table 4 lists the components with their expected performance impacts.

Table 4. *INTELLIBROWSE Component Specifications*

Component	Technology Base	Function	Expected Improvement
Rendering Engine	Modified Chromium	Shared process pool with capability isolation	40-60% memory reduction
Privacy Module	Entropy controller	Active fingerprinting prevention	<10 bits total entropy
Intelligence Pipeline	Local LLM + TensorFlow.js	Semantic extraction and automation	70%+ task completion
Resource Manager	Custom scheduler	Dynamic resource allocation	50% CPU reduction

Memory Optimisation Strategy

IntelliBrowse uses these methods to optimise memory:

1. Static resource sharing: Common JavaScript libraries or frameworks are shared
2. On-demand process creation: Processes are only created on demand
3. Aggressive garbage collection: Better garbage collection using user interaction
4. Memory-mapped static resources: Static resources are memory mapped

Privacy through Architecture

IntelliBrowse does not attempt to prevent fingerprinting, but instead seeks to reduce the entropy of the system. Entropy reduction is given in Table 5

Table 5. *Entropy Reduction Targets and Methods*

API Category	Current Entropy	Target Entropy	Reduction Method
Canvas Operations	5.73 bits	<2 bits	Output quantization

WebGL Rendering	4.21 bits	<2 bits	Parameter standardization
Font Metrics	3.82 bits	<1.5 bits	Universal font subset
Hardware APIs	3.94 bits	<1 bit	Cohort-based reporting
Total System	18.7 bits	<8 bits	Architectural enforcement

Intelligence Integration

The intelligence layer offers semantic understanding through:

1. DOM-to-semantic mapping: Transforms HTML into concepts
2. Visual understanding: Learns the rendered output to understand the layout
3. User interaction prediction: Understands user behaviour patterns
4. Context across domains: Preserves semantic information

Evaluation And Expected Outcomes

Performance Projections

Prototype testing and simulation indicate IntelliBrowse is likely to provide the benefits outlined in Table 6.

Table 6. *Projected Performance Improvements*

Metric	**Current State**	**Evidence Base**	**IntelliBrowse Target**	**Improvement**
Memory Usage	300-500 MB idle	Empirical measurement	120-200 MB idle	60% reduction
Privacy (Uniqueness)	89% identifiable	Fingerprinting studies	<10% identifiable	79% improvement
Automation Success	14% complex tasks	WebArena benchmark	>50% complex tasks	257% improvement
Page Load Time	Baseline	Performance profiling	15% faster	15% improvement

Implementation Challenges

Some issues need to be overcome before deployment:

1. Web compatibility: Compatibility of existing sites
2. Developer migration: Enabling web developers to migrate
3. Security: Demonstrating isolation based on capability
4. Optimisation: Reducing the cost of the intelligence layer
5. User experience: Breaking browser switching habits

Validation Methodology

We suggest a validation process in several phases:

- Phase 1: Benchmarking in the lab
- Phase 2: Developer preview
- Phase 3: Beta release to the public with usage data
- Phase 4: Independent security assessment
- Phase 5: General release with ongoing monitoring

Future Research Directions

This review highlights the need for further research in:

1. Advanced Privacy Techniques Homomorphic encryption for browser operations might allow functionality without compromise. Differential privacy could enable statistical data without identifying individuals.
2. Quantum-Resistant Security With the rise of quantum computing, browsers need to be ready for post-quantum cryptography. This involves assessing quantum-safe algorithms, migration plans.
3. Neuromorphic Computing Integration New neuromorphic processors may support effective AI processing in browsers. Studies should investigate how browsers can take advantage of these technologies.
4. Decentralised Web Technologies Blockchain and IPFS integration may decrease centralisation. Web browsers will need to support decentralised protocols

Conclusion

This systematic review shows that current web browsers fundamentally have design limitations that cannot be overcome by evolutionary changes. The three-pronged crisis of poor automation, privacy violations and high resource consumption is rooted in design principles that favour backwards compatibility and isolated security over systemic design. In a review of 48 recent papers, we show that existing approaches have hit a wall. Memory overhead nearing 300%, fingerprinting rates nearing 100% and automation rates less than 15% for complex tasks point to the need for a paradigm shift. The IntelliBrowse architecture provides a solution through rethinking the assumptions. Its intelligent resource management, architectural privacy, and native integration of intelligence seek to deliver considerable gains across the three problem areas. Despite the difficulties in implementation, the rewards of a rethink are worth it. Browsers have not changed much since the current browser architectures were designed. It is time for browsers to experience a similar revolution to serve today's users and applications. This review offers a guide for such a change and the need for browser innovation.

References

1. C. Reis, A. Moshchuk, and N. Oskov, “Site Isolation: Process Separation for Web Sites within the Browser,” in *Proceedings of the 28th USENIX Security Symposium*, Santa Clara, CA, 2019, pp. 1661-1678.
2. A. Grosskurth and M. W. Godfrey, “A Reference Architecture for Web Browsers,” in *21st IEEE International Conference on Software Maintenance (ICSM’05)*, Budapest, Hungary, 2005, pp. 661-664.
3. A. Barth, C. Jackson, C. Reis, and The Google Chrome Team, “The Security Architecture of the Chromium Browser,” Stanford University Technical Report, 2008.
4. Y. Zhu and V. J. Reddi, “WebCore: Architectural Support for Mobile Web Browsing,” in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, Minneapolis, MN, 2014, pp. 541-552.
5. G. Richards, S. Lebresne, B. Burg, and J. Vitek, “An Analysis of the Dynamic Behaviour of JavaScript Programs,” in *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, Toronto, Canada, 2010, pp. 1-12.
6. K. Singh, A. Moshchuk, H. J. Wang, and W. Lee, “On the Incoherencies in Web Browser Access Control Policies,” in *2010 IEEE Symposium on Security and Privacy*, Berkeley/Oakland, CA, 2010, pp. 463-478.
7. H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, “Towards Online Spam Filtering in Social Networks,” in *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, San Diego, CA, 2012.
8. Y. Cao, S. Li, and E. Wijmans, “Cross-Origin State Inference (COSI) Attacks: Leaking Web Site States through XS-Leaks,” in *Proceedings of the Network and Distributed System Security Symposium*, 2023.
9. I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, “WebArena: A Realistic Web Environment for Building Autonomous Agents,” in *International Conference on Learning Representations*, 2024.
10. P. Laperdrix, W. Rudametkin, and B. Baudry, “Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints,” in *37th IEEE Symposium on Security and Privacy*, San Jose, CA, 2016, pp. 878-894.
11. P. Eckersley, “How Unique Is Your Web Browser?” in *International Symposium on Privacy Enhancing Technologies Symposium*, Berlin, Germany, 2010, pp. 1-18.
12. N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, “Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting,” in *2013 IEEE Symposium on Security and Privacy*, Berkeley, CA, 2013, pp. 541-555.
13. S. Englehardt and A. Narayanan, “Online Tracking: A 1-million-site Measurement and Analysis,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 1388-1401.
14. K. Chen, J. Liu, Y. Zhang, and H. Wang, “Memory Consumption Analysis of Modern Web Browsers,” in *2021 IEEE International Conference on Computer and Information Technology*, 2021, pp. 142-147.
15. L. Guo, D. Zhang, and M. Chen, “BROWSE: A Multiplatform Web Browser Architecture for Mobile Devices,” *IEEE Software*, vol. 27, no. 4, pp. 22-29, Jul./Aug. 2010.