RESEARCH STATEMENT

Abdul Haddi Amjad
https://people.cs.vt.edu/~hadiamjad/

The vision of my research is to develop frameworks that enhance the privacy and security of online websites through advanced program analysis. My work integrates techniques from software debugging and testing into the domains of web security and privacy, bridging two traditionally distinct areas to make the web safer and more privacy-preserving. This interdisciplinary approach has resulted in my research appearing in top-tier venues across both software engineering (ICSE, FSE) and web security and privacy (IMC, PETS, CCS).

Overview. Modern websites often provide free services, yet nearly every online interaction now involves the collection and processing of personal user data. To facilitate this, websites rely extensively on JavaScript, whose size and complexity have expanded dramatically in recent years [4], extending far beyond its original purpose of enabling interactive functionality. This evolution has led to the widespread integration of third-party JavaScripts for analytics, advertising, and user tracking, embedding data collection code deeply into the operational fabric of the modern web. Such extensive tracking enables entities across websites to construct detailed user profiles and targeted content that can shape and even bias user behavior, raising fundamental concerns about privacy and individual autonomy in the modern web. These concerns are further amplified by the rise of browser-based intelligent agents, such as Atlas by OpenAI, which automate user interactions and expand the surface for data collection and inference.

Users today rely on tools such as ad-blockers and privacy-focused browsers, but these solutions are largely ineffective at addressing the root causes of data collection. Over a decade of research and industry effort has focused on superficial defenses that treat the symptoms of tracking rather than preventing it. Because these approaches are brittle, easily circumvented, and fail to address the underlying sources of tracking, users remain exposed to pervasive and opaque practices that are intrusive, difficult to detect, and increasingly hard to avoid. At the same time, protecting privacy often conflicts with the business models of free online websites that depend on collecting, analyzing, and sharing user data for revenue.

Contributions. The key insight of my research vision is to address privacy risks at their origin rather than through superficial solutions. This vision has grown increasingly relevant as privacy continues to gain national attention. The 2025 Mid-Year Review of U.S. State Comprehensive Data Privacy Law Updates, highlights expanding federal and state efforts to strengthen online privacy protections by expanding definitions of sensitive data, and enhanced safeguards for minors [22]. To realize this vision, my work applies program analysis techniques that combine dynamic analysis, semantic slicing, and lightweight JavaScript refactoring to treat websites as software systems where privacy-invasive behavior is localized and mitigated without breaking functionality. My research is organized around two central themes:

- The first theme focuses on using program analysis techniques to identify privacy-invasive code within modern web applications, where increasingly large and complex JavaScript code conceal data collection behaviors.
- The second theme examines how to refactor such code and validate the refactored scripts through testing to ensure that website functionality is preserved while privacy risks are effectively mitigated.

Theme I: Identifying privacy-invasive code through program analysis. Modern websites present significant challenges for identifying privacy-invasive behaviors due to their increasing JavaScript code obfuscation, minification, and dynamic nature. According to the Web Almanac, the median JavaScript payload on popular websites now exceeds several hundred kilobytes , with the widespread adoption of bundlers and build pipelines further blurring logical dependencies between scripts [4]. The growing adoption of minification removes meaningful variable names and structural cues, while obfuscation deliberately disguises control flow to hinder analysis [21, 3]. Additionally, the pervasive use of dynamic constructs such as `eval` [18], function constructors, and runtime script injection allows code to alter its own execution context, making static inspection alone inadequate for identifying privacy-invasive behaviors.

To address these challenges, my research applies program analysis techniques that combine static and dynamic analysis to uncover privacy-invasive behaviors within modern websites. Specifically, I perform cross-script call stack analysis, constructing timestamped, event-based execution graphs that capture interdependent script interactions. This approach collects complete call stacks, including the sequence of function calls leading to each event on a website, enabling precise differentiation between functions contributing to privacy-invasive behavior and those essential for legitimate functionality. By mapping these call relationships into structured execution graphs that incorporate DOM , network requests, JavaScript API calls, and inter-function dependencies, my analysis reveals fine-grained data flow patterns and exposes hidden privacy-invasive code paths that are otherwise difficult to detect.

Impact Summary

This research led to the discovery of previously unidentified privacy invasive JavaScript on websites, which we reported to the developers of ad-blockers, who acknowledged our findings [14, 12, 13, 15, 16]. Furthermore, our analysis uncovered a bug in one such ad-blocker (Ghostery [17]), demonstrating its inability to detect certain privacy invasive scripts that our approach successfully identified [8]. This line of work also resulted in my co-authorship of the *Privacy* [11] and *JavaScript* [10] chapters of the *Web Almanac 2024* edition, where I contributed empirical insights on web tracking and code transparency.

Theme II: Refactoring and testing privacy-invasive JavaScript. Once the privacy-invasive code has been identified, the next challenge is to refactor or replace it while maintaining the website's legitimate functionality. Three factors make this task particularly challenging. First, the dynamic nature of JavaScript, characterized by runtime code generation, event-driven execution, and cross-script dependencies, complicates both refactoring and replacement. Second, applying such modifications at runtime without introducing performance delays or functional issues requires the development of lightweight and adaptive mechanisms. Third, conducting a comprehensive website breakage analysis requires careful evaluation to ensure that the refactored code continues to operate correctly across different user interactions and execution contexts.

To address these challenges, my research introduces a multi-stage approach. First, we design a graph based machine learning localization technique that captures the complete execution flow of a website when privacy-invasive code is triggered. This approach handles JavaScript dynamism by collecting detailed metadata at runtime, allowing accurate localization of privacy-invasive code. The localized code is then refactored by diffusing one of the function calls in the privacy-invasive call sequence to break the execution chain and stop the unwanted behavior. Second, we develop a Chrome extension compatible with both Manifest Version 2 and Version 3 that enables fast, runtime replacement of refactored code without adding computational cost. Third, we define a rigorous manual evaluation process to assess client-side website functionality after refactoring, ensuring that essential features remain intact.

Impact Summary

This work opened a new line of research that inspired others to examine privacy-invasive JavaScript at function-level granularity, shifting the focus from superficial approaches to the core origins of data collection. [1, 19]. This work was recognized with the *Best Artifact Award* at CCS 2024 [20], underscoring its technical rigor, deployability, and impact in advancing practical privacy-enhancing technologies. Moreover, this line of work has also led to collaboration with Brave, where our refactoring and replacement strategy has been integrated into one of their products, Brave Talk [9].

Current Research

TrackerSift [IMC 2021 and PETS 2023]. My research on JavaScript privacy analysis unifies two complementary efforts, TrackerSift [6] and a follow up study on JavaScript blocking [7]. This line of work was the first to demonstrate that websites intentionally entangle privacy-invasive code within functional code, creating mixed scripts that obscure tracking behavior. TrackerSift addresses the challenge of localizing such mixed code by combining static dependency analysis with dynamic execution tracing to separate legitimate and privacy-invasive functionality across multiple granularities, including domain, hostname, script, and function levels. Building on these insights, the subsequent study introduced a client side breakage analysis framework to examine the trade off between privacy protection and website functionality across different blocking granularities. *Together, these studies revealed that nearly one third of all analyzed JavaScripts exhibit mixed behavior, and longitudinal analysis shows that this proportion continues to increase over time, underscoring the growing complexity of privacy-invasive JavaScript on the modern web.*

Not.js [CCS 2024]. My next work, Not.js [5] addresses the challenge of runtime localization and mitigation by detecting and refactoring privacy-invasive JavaScript during execution without disrupting legitimate functionality. I designed NoT.js as a Chrome-based privacy enhancing extension that deploys lightweight runtime scripts to intercept, analyze, and modify JavaScript behavior. It incorporates a graph-based machine learning model trained on execution traces to distinguish between interleaved privacy-invasive and functional logic. Once identified, Not.js automatically refactors the code by diffusing function calls, isolating the invasive behavior, and replacing it with safe alternatives while preserving overall site functionality. *Evaluated across real world websites, Not.js eliminated 92% of privacy-invasive activity while maintaining 97% of website functionality, reducing JavaScript heap-usage and improving load times with lightweight execution. This work received the Best Artifact Award at CCS 2024 [20], recognizing its real world impact on practical privacy protection.*

Other directions [ICSE 2025 and PETS 2026]. In my most recent work [2], I examined how privacy risks are unevenly distributed across users, particularly among vulnerable groups such as children, elderly users, and individuals who are vulnerable. My paper demonstrates that gestures used by screen readers, such as tabbing, focusing, or hovering, can inadvertently trigger privacy-invasive JavaScript. *We show that vulnerable users were exposed to 1.7x more privacy invasive activity, emphasizing that privacy is not uniform across users and that web design must account for interaction patterns that unintentionally increase exposure.* Another direction I am exploring involves the use of custom headers in collecting and transmitting user identifiers to tracking services. From a JavaScript code perspective, we examine how these identifiers are generated, embedded into custom headers, and subsequently shared with third party tracking entities. This work is currently under submission.

Industry collaborations. I have collaborated with industry partners on projects that offered practical perspectives on large-scale data integrity and privacy challenges. At Amazon Marketplace, I applied my research expertise to a different problem space, developing a framework to detect fraudulent sellers by analyzing behavioral and transactional patterns across large datasets. At Admiral, I worked on classifying and differentiating ad-blockers and on designing strategies to re-serve ads through targeted user engagement prompts.

Future Direction

The next stage of my research focuses on solving the emerging challenges in this field, understanding the barriers that have limited broader engagement, and highlighting why my expertise positions me to address them effectively. In the following section, I describe several projects that exemplify my recent research efforts and outline speculative directions that I aim to explore in the future.

Tracing the provenance of user identifiers on the web. A natural next step in my research focuses on uncovering the end-to-end provenance of user identifiers by tracing how they are generated,

transformed, and ultimately transmitted to tracking entities. Existing tools, including my own, can detect privacy-invasive scripts but lack visibility into the complete lifecycle of these identifiers. My goal is to understand this lifecycle comprehensively, i.e., from the point an identifier is created to when it is exfiltrated, while attributing each stage to the responsible JavaScript code and APIs. To achieve this, the framework constructs execution provenance graphs that integrate taint tracking, symbolic execution, and runtime instrumentation, revealing how identifiers flow through the code with fine grained precision. Building on this foundation, I plan to develop (1) a taxonomy of identifier generation techniques across a large web corpus, (2) a cross layer provenance analyzer that links JavaScript execution traces with network data to pinpoint where and how identifiers are exfiltrated, and (3) a developer assist tool that visualizes provenance chains. Together, these efforts aim to provide the first system level understanding of identifier lifecycles, enabling privacy transparency, accountability, and compliance auditing at scale.

Embedding privacy defenses into the JavaScript engine. A complementary research direction explores detecting and refactoring privacy-invasive JavaScript directly within the V8 engine rather than at the extension layer. Through my work with Brave, which leverages the Chromium-based V8 JavaScript engine, I have gained in depth experience working with its internals i.e., from parsing JavaScript source code into abstract syntax trees to generating bytecode and optimizing it into machine code through TurboFan. Building on this foundation, I plan to embed fine-grained instrumentation hooks at key stages of the V8 execution pipeline to identify privacy-invasive operations such as identifier construction, cross-context data sharing, and network bound serialization. By integrating lightweight static analysis during bytecode generation and applying policy guided transformations at runtime, the system can automatically refactor or neutralize invasive logic before it executes. This approach aims to move privacy protection closer to the engine level, enabling faster, more transparent, and tamper resistant enforcement of privacy rules within the browser itself.

Leveraging large language models for automated privacy repair. Another research direction is the use of large language models (LLMs) to automate the localization and refactoring of privacy invasive JavaScript in a more efficient and context-aware manner. While current approaches in NoT.js rely on symbolic analysis and graph-based learning, LLMs can better generalize across diverse obfuscation patterns and mixed functionalities. The goal of this direction is to train code-aware models such as CodeLlama or DeepSeek Coder on paired datasets of original and refactored JavaScript from NoT.js, enabling them to automatically generate privacy preserving code transformations while maintaining site functionality. This approach aims to achieve near real-time mitigation with minimal human oversight. Future extensions include developing a context guided prompting system using call stack and dataflow summaries, an automated patch verification framework for functional testing, and a feedback loop that refines the model based on runtime validation. Together, these efforts aim to evolve NoT.js into an adaptive, self improving privacy repair system.

References

[1] M. M. Ali, P. Snyder, C. Kanich, and H. Haddadi. Unbundle-rewrite-rebundle: Runtime detection and rewriting of privacy-harming code in javascript bundles. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2192–2206, 2024.

[2] A. H. Amjad, M. Danish, B. Jah, and M. A. Gulzar. Accessibility Issues in Ad-Driven Web Applications . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pages 2393–2405, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00171, doi:10.1109/ICSE55347.2025.00171.

[3] A. H. Amjad and N. Goel. Javascript — 2024 — the web almanac by http archive. https://almanac.httparchive.org/en/2024/javascript#minification, March 2024. Accessed: 2025-10-18.

[4] A. H. Amjad and N. Goel. Javascript — 2024 — the web almanac by http archive. https://almanac.httparchive.org/en/2024/javascript#how-much-javascript-do-we-load, Mar 2025. Accessed: 2025-10-18.

[5] A. H. Amjad, S. Munir, Z. Shafiq, and M. A. Gulzar. Blocking tracking javascript at the function granularity. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 2177–2191, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3658644.3670329.

[6] A. H. Amjad, D. Saleem, M. A. Gulzar, Z. Shafiq, and F. Zaffar. Trackersift: Untangling mixed tracking and functional web resources. In *Proceedings of the 21st ACM Internet Measurement Conference*. Association for Computing Machinery, 2021.

[7] A. H. Amjad, Z. Shafiq, and M. A. Gulzar. Blocking javascript without breaking the web: An empirical investigation. *arXiv preprint arXiv:2302.01182*, 2023.

[8] H. Amjad. Comment on issue #2346: Generic cosmetic filter not applied for document_start — ghostery/ghostery-extension. https://github.com/ghostery/ghostery-extension/issues/2346#issuecomment-2774889651, Apr 2025. Accessed: 2025-10-19.

[9] H. Amjad. Pull requests by hadi amjad on the brave talk repository (closed). https://github.com/brave/brave-talk/pulls?q=is%3Apr+is%3Aclosed+author%3Ahadiamjad, 2025. Accessed: 2025-10-19.

[10] H. Amjad et al. Javascript. In *The Web Almanac 2025*. HTTP Archive, 2025. Accessed: 2025-10-19.

[11] H. Amjad et al. Privacy. In *The Web Almanac 2025*. HTTP Archive, 2025. Accessed: 2025-10-19.

[12] E. authors. Issue #18185: [description missing; user to fill with specific title]. https://github.com/easylist/easylist/issues/18185, Jan 2024. Accessed: 2025-10-19.

[13] E. authors. Issue #18230: [description missing; user to fill with specific title]. https://github.com/easylist/easylist/issues/18230, Jan 2024. Accessed: 2025-10-19.

[14] E. authors. Issue #18242: Script (https://script1.kakaku.k-img.com/script/history/js...). https://github.com/easylist/easylist/issues/18242, Jan 2024. Accessed: 2025-10-19.

[15] E. authors. Issue #18243: [description missing; user to fill with specific title]. https://github.com/easylist/easylist/issues/18243, Jan 2024. Accessed: 2025-10-19.

[16] E. authors. Issue #18244: Script (https://www.npttech.com/advertising.js) on oe24.at is detecting ad-blockers and taking alternative and is not blocked. https://github.com/easylist/easylist/issues/18244, Jan 2024. Accessed: 2025-10-19.

[17] Ghostery, Inc. Ghostery browser extension. https://github.com/ghostery/ghostery-extension, 2025. Privacy-enhancing browser extension for tracking protection. Accessed: 2025-10-19.

[18] G. Richards, C. Hammer, B. Burg, and J. Vitek. The eval that men do: A large-scale study of the use of eval in javascript applications. In *European Conference on Object-Oriented Programming*, pages 52–78. Springer, 2011.

[19] H. Shuang, L. Zhao, and D. Lie. Duumviri: Detecting trackers and mixed trackers with a breakage detector. *arXiv preprint arXiv:2402.08031*, 2024.

[20] A. SIGSAC. Ccs 2024 awards. https://www.sigsac.org/ccs/CCS2024/program/awards.html, 2024. Accessed: 2025-10-19.

[21] P. Skolka, C.-A. Staicu, and M. Pradel. Anything to hide? studying minified and obfuscated code in the web. In *The world wide web conference*, pages 1735–1746, 2019.

[22] A. C. Thomson, L. Shen, H. W. Waltzman, A. Kourinian, M. P. V. Klein, and D. Garcia. 2025 mid-year review: Us state comprehensive data privacy law updates (part 1). https://www.mayerbrown.com/en/insights/publications/2025/09/2025-mid-year-review-us-state-comprehensive-data-privacy-law-updates-part-1, Sept. 2025. Accessed: 2025-10-27.