RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Address Bar Spoofing in Contemporary Web Browsers: A Taxonomy, Exploitation Study, and Mitigation Guidelines

Renua Hiwa Ismael, Jaza Mahmood Abdullah

Department of Information Technology, College of Commerce, University of Sulaimani, Sulaymaniyah, Iraq

ABSTRACT

The browser address bar is the cornerstone of user trust and web security. Despite advancements, address bar spoofing remains a persistent threat, enabling attackers to make malicious URLs appear legitimate. This paper presents an extensive investigation into address bar spoofing vulnerabilities across modern desktop and mobile browsers. We introduce a comprehensive taxonomy classifying over 15 distinct spoofing techniques, many of which are novel. Across systematic testing, over 70 vulnerabilities were identified and responsibly disclosed, resulting in patches across more than 15 browsers. These findings are enumerated in this paper for verification. This research analyzes the root causes of these vulnerabilities, highlighting common pitfalls in URL parsing, display logic, and UI state management. Based on our findings, we propose a robust mitigation framework and best practices for browser developers, alongside actionable advice for users. Our findings underscore the ongoing challenge of maintaining address bar integrity and the critical need for continuous vigilance in browser security. A public repository documents these findings to aid further research.

Index Terms: Address Bar Spoofing, Web Browser Security, Phishing, URL Spoofing, UI Security, Vulnerability Taxonomy, Cyber Security

1. INTRODUCTION

Phishing attacks continue to be one of the most prevalent and damaging cyber threats, costing individuals and organizations billions of dollars annually [1]. Attackers employ various social engineering tactics, often relying on deceptive URLs to trick victims. While URL masking or cloaking techniques are common, a more insidious form of deception is address bar spoofing, where a vulnerability within the web browser itself is exploited to display a trusted origin (e.g., bank.com) while the user is actually interacting with a malicious site. This undermines the fundamental trust users place in the address bar as an authenticator of web identity [2].

Despite significant efforts by browser vendors to harden this critical UI component, our research demonstrates that address bar spoofing is not a solved problem. Modern browsers, with their complex feature sets and support for diverse URI schemes and UI states, introduce new attack surfaces. This paper details an extensive, multi-year investigation into these vulnerabilities across prevalent desktop (Windows, macOS) and mobile (iOS, Android) browsers, including Chrome, Firefox, Safari, Edge, Opera, Brave, Arc, and others.

A comprehensive dataset, including detailed descriptions, Proof-of-Concept (PoC) code, and replication videos for all 74 vulnerabilities discussed, is publicly available in our

Access this article online

DOI:10.21928/uhdjst.v9n2y2025.pp335-345

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: Renua Hiwa Ismael, Department of Information Technology, College of Commerce, University of Sulaimani, Sulaymaniyah, Iraq. E-mail: renwax23@gmail.com

Received: 12-07-2025 Accepted: 25-09-2025 Published: 30-11-2025

repository [3]. For direct verifiability within this paper, a summary of these findings is provided in Appendix A.

The primary contributions of this paper are:

1. Demonstrated Persistence and Novelty: We uncovered over 70 distinct address bar spoofing vulnerabilities, many leveraging novel or refined exploitation techniques, leading to direct security improvements in over 15 browsers.
2. Comprehensive Vulnerability Taxonomy: We developed a structured taxonomy classifying diverse spoofing techniques, moving beyond isolated bug reports to provide a systematic understanding of the attack surface.
3. Actionable Mitigation Framework: Based on identified root causes, we propose concrete mitigation strategies for browser developers and user education guidelines.

2. THE EVOLVING THREAT OF ADDRESS BAR SPOOFING

Address bar spoofing is not a new phenomenon. Chromium's guidelines for URL display [4] acknowledge many such scenarios. Early instances date back to vulnerabilities in browsers, such as Internet Explorer 6, exploiting dialog boxes or character encoding quirks [5]. Over time, attacks evolved to leverage weaknesses in handling HTTPS indicators with delayed responses [6], Unicode homograph attacks [7], race conditions in mobile browsers [8], manual spoofing tests [9], and flawed server-side redirect handling [10]. However, as browsers add new features (e.g., Picture-in-Picture, varied URI scheme support) and UIs evolve, new vectors for spoofing continually emerge, often due to subtle implementation errors or incomplete parsing and validation logic. Our research found that even browsers built on mature, security-focused engines, such as Chromium and Gecko are susceptible to these attacks when specific UI choices or feature integrations are made.

3. RESEARCH METHODOLOGY

Our research methodology involved a multi-staged approach conducted over a multi-year period (2019–2024). This section details our testing environment and the systematic process used for vulnerability discovery, analysis, and classification.

3.1. Threat Model and Assumptions

For this research, we assume a standard web-based threat model. The attacker is capable of hosting a malicious website and can persuade a victim to click a link leading to it (e.g., through a phishing email or social media message). However, the attacker cannot install malware on the victim's machine or compromise the underlying operating system. The victim is assumed to be a non-expert user who relies on the browser's UI, particularly the address bar, as a primary indicator of a website's identity. The goal of the attacker is to exploit a browser vulnerability to make their malicious site's address appear as a trusted, legitimate domain.

3.2. Testing Environment and Scope

Our investigation targeted a wide range of modern web browsers across major desktop and mobile platforms. We primarily focused on the latest stable and beta versions available during the testing periods to ensure relevance. The scope of our test environment is summarized in Table 1.

3.3. Analysis Stages

Our process consisted of five distinct stages:

1. Systematic Testing of Browser Features: We systematically tested browser features known to be complex or historically problematic. For each feature, we developed a test plan to probe edge cases. For instance, to test data: URI handling, over 50 permutations were crafted, varying the encoding, position of hostname-like strings, and special characters to observe parsing and display logic. Features tested included search engine integration, Address bar freezing, internal URI schemes, full screen/PiP APIs, blob/data/javascript URI schemes, RTL character display, and Unicode parsing.
2. Reverse-Engineering Browser Behavior: For identified anomalies, we analyzed browser behavior to determine the root cause. This sometimes-involved decompiling Android applications or examining open-source code. For example, to diagnose a search query spoof in Opera Mini (ID: 11), we used jadx-gui [11] to decompile the Android application package (APK). By analyzing the Java class responsible for parsing search provider URLs,

TABLE 1: Overview of the testing environment and scope

Browser (engine)	Versions tested	Operating systems (OS)
Google Chrome (Chromium)	130–139	Windows 10, macOS 15, Android 14, iOS 15-26
Mozilla Firefox (Gecko)	135–144	Windows 10, macOS 15, Android 14, iOS 15-26
Apple Safari (WebKit)	15–26	macOS 15-26, iOS 15-26
Microsoft Edge (Chromium)	130–139	Windows 10, macOS 12-14, Android 14, iOS 15-26
Opera/Opera GX (Chromium)	120–130	Windows 10, Android 14
Arc Browser (Chromium/WebKit)	1.0–1.10.1	Windows 10, macOS 15–26, iOS 15–26

we identified a flawed regular expression that failed to correctly validate the origin.
3. Exploitation and PoC Development: Once a potential vulnerability was identified, we developed a minimal PoC to confirm its exploitability and demonstrate its security impact.
4. Responsible Disclosure: All vulnerabilities were reported to the respective browser vendors through bug bounty programs (e.g., HackerOne, Bugcrowd) or direct security contacts. Public disclosure of details was withheld until patches were made available or a reasonable timeframe had passed.
5. Taxonomy Creation: Based on the diverse attack vectors and root causes, we categorized all confirmed vulnerabilities into the comprehensive taxonomy presented in Section 4.

4. A COMPREHENSIVE TAXONOMY OF ADDRESS BAR SPOOFING TECHNIQUES

Our research identified numerous ways attackers can manipulate the browser's address bar. The following taxonomy categorizes the primary techniques observed. For brevity, only a selection of categories and illustrative examples are presented.

To provide a more structured overview, Table 2 summarizes the key characteristics of the primary spoofing classes identified in our research, including their typical root cause and user interaction requirements.

4.1. UI State Manipulation

Exploiting browser states like Full-screen mode or features, such as Picture-in-Picture (PiP) to obscure or replace the authentic address bar.

- Full-screen Spoofing: An attacker's site requests full-screen mode upon user interaction. The browser UI, including the address bar, disappears. The attacker then renders a fake address bar image within the web content, tricking the user (Fig. 1). While some browsers warn upon entering full screen, many do not, or the warning is easily missed.
- Window without Address Bar (e.g., via PiP): Certain APIs, such as Picture-in-Picture, can create new windows or views without a native address bar. Attackers can then draw a fake address bar within this content area. We found Microsoft Edge on Mac vulnerable to this via PiP.

4.2. Non-Standard Scheme Handling

Abusing the browser's often less-rigorous parsing, validation, and display logic for URIs beyond standard http and https.

- Data: URI Spoofing: data: URIs embed content directly. We found that by crafting a data: URI with a spoofed hostname-like string before the actual comma-separated data (e.g., data://google.com/,<malicious_content>), some mobile browsers (such as Arc Mac or Opera GX Android) would misleadingly display google.com in the address bar. See Fig. 2 for an example in Arc Mac.
- Blob: URI Spoofing: blob: URIs represent raw data. Some browsers, when navigating to a blob: URI, display the full blob URL. By appending a hash fragment containing the spoofed URL (e.g., blob: https://real-origin.com/GUID#https://spoofed.com), browsers, such as Safari would prioritize displaying the fragment, effectively hiding the true blob: origin.
- JavaScript: URI Spoofing: Redirecting to or opening a JavaScript: URI can lead to confusion. In PlayStation 5's browser, redirecting to a legitimate site, then immediately opening a new window to a JavaScript: URI caused the address bar to retain the legitimate site's URL while executing an attacker-controlled script that wrote to the new window.
- Internal URI Schemes: Browsers use internal schemes (e.g., internal://, arcmobile-internal://) for error pages or special functions. Redirecting to these schemes, often with parameters controlling the displayed URL, allowed spoofing. For example, Firefox for iOS allowed redirection to internal://local/errorpage?url=https://

TABLE 2: Characteristics of spoofing technique classes

Taxonomy class	Primary root cause	Typical user interaction	Common dependency
UI State Manipulation	Hiding/replacing browser chrome	One click (e.g., to enter full screen)	Browser/OS specific APIs (PiP, Fullscreen)
Non-Standard Scheme Handling	Lax parsing of non-HTTP (S) URIs	None (on link click)	Browser-specific scheme implementation
Navigation Timing	Premature address bar update	None (on link click)	General browser navigation logic
URL Parsing and Display	Flawed truncation or display logic	None	Logic, especially on mobile UI
Unicode & RTL	Incorrect text rendering	None	Unicode/BiDi rendering engine
Internal Component Interaction	Privilege boundary violation	None (on link click)	Browser-specific internal pages/APIs

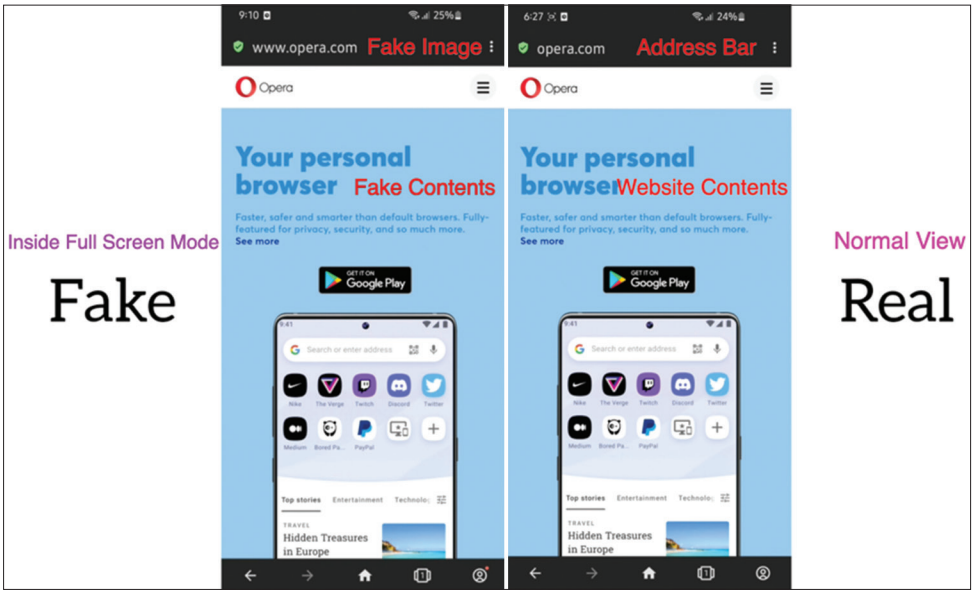

Fig. 1. Demonstration of Fullscreen Spoofing in Opera Mobile. (Left) The attacker's page (attacker.com) enters fullscreen, hides the real UI, and renders a fake address bar image showing opera.com. (Right) The legitimate opera.com site shown in the standard browser view for comparison.

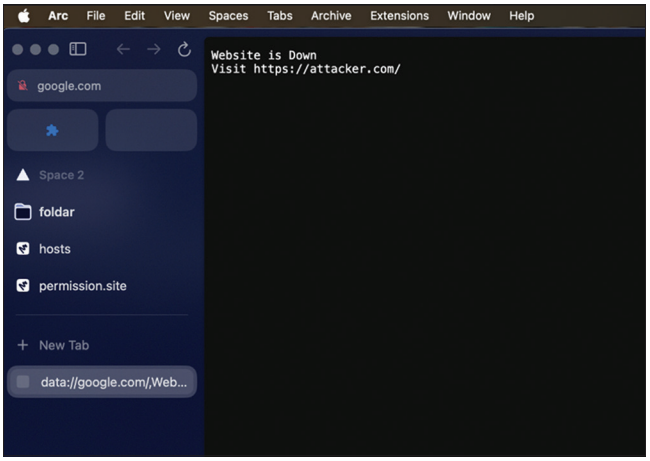

Fig. 2. Exploitation of data: URI parsing in Arc for macOS. The address bar misleadingly displays google.com, a string parsed from the URI path while the page content is fully controlled by the attacker via the data: URI itself.

google.com/., which would display google.com while allowing content.

4.3. Navigation Timing and Race Conditions

Capitalizing on the temporal gap between initiating a navigation request and receiving a definitive server response.

- URL Freeze (Change Before Response): The most common vulnerability. An attacker redirects to an unreachable port on a legitimate domain (e.g., google.com:8081). Many browsers immediately update the address bar to google.com *before* realizing the port is unreachable. The page content remains attacker-controlled, leading to a spoof (Fig. 3). This was observed in numerous browsers, including Arc Android and Safari.
- HTTP 204 No Content: This technique is distinct from URL Freeze: while URL Freeze exploits an unreachable destination, the HTTP 204 attack uses a valid but empty server response. The browser correctly updates the address bar to the target URL, but because no new content is sent, the attacker's original page content remains visible. This was observed in Arc for Android.

4.4. URL Parsing and Display Logic Flaws

Leveraging ambiguities in how browsers parse, truncate, and display complex URLs.

- Long Domain/Subdomain to Hide Origin: On small screens (especially mobile), long subdomains (e.g., very.long.subdomain.target.com.attacker.com) can cause the browser to truncate the URL, potentially only showing very.long.subdomain.target.com and hiding attacker.com. Firefox for Android/iOS was vulnerable, prioritizing the beginning of the URL, PoC in Fig. 4.
- Search Engine Integration Quirks: Some mobile browsers display the search query in the address bar instead of the search engine's URL. Flaws in validating the search engine's origin or path allowed spoofing. For example, a malformed regex in Opera GX allowed google.pwn.wtf to be treated as a Google search, displaying the query from ?q= as the URL. The Arc browser for iOS was similarly vulnerable.

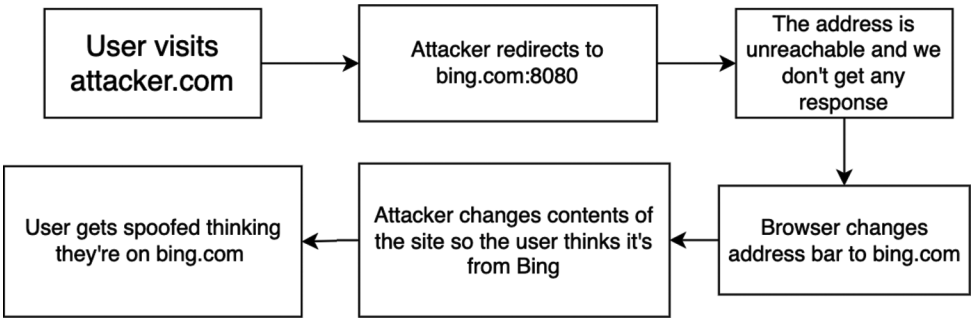

Fig. 3. Flowchart of a "URL Freeze" timing attack. The browser pre-maturely updates the address bar to a legitimate domain (bing.com) upon redirection, but the destination port is unreachable, causing the navigation to fail while leaving the spoofed address and attacker-controlled content visible.

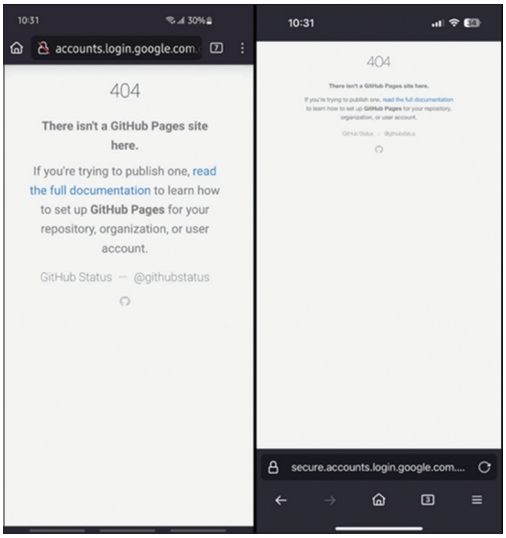

Fig. 4. Long Subdomain Spoof in Firefox Mobile. The browser truncates the long URL, prioritizing the display of the fake subdomain (accounts.login.google.com…) while completely hiding the true, malicious origin (…github.io).

4.5. Unicode and Right-to-Left (RTL) Manipulation

Utilizing visually confusable Unicode characters or exploiting complex rendering rules for mixed RTL/LTR text.

- Unicode Whitespace-like Characters: Characters like U+08FC can appear as whitespace. Bentkowski [12] showed this could be used in domains, such as important-domain.google.com<U+08FC><U+08FC>evil.com, where Chrome/Firefox would display only the important-domain.google.com part.
- RTL/LTR Confusion: Mixing RTL (e.g., Arabic, Kurdish) and LTR (e.g., English) scripts in a domain can confuse the browser's display logic, causing parts of the URL to be reordered or hidden, leading to spoofing. Brave for iOS was vulnerable when displaying domains like https://xn--llb.login.accounts.google.com.xn--llb.attacker.com, misrendering the true origin.

4.6. Browser Internal Component Interaction

Exploiting vulnerabilities within or between privileged browser components.

- Internal Pages XSS: An XSS in a privileged internal browser page (e.g., arcmobile-internal://in Arc iOS) responsible for rendering UI elements or handling errors could be used to manipulate the address bar display by controlling parameters passed to this internal URI.

5. RESULTS AND DISCUSSION

Our process involved identifying and classifying each vulnerability, reporting it to the vendor for patching, documenting the entire procedure along with the root causes that are shared among them, and then assigning severity ratings of each spoof.

5.1. Statistical Overview

Our research initiated in 2019 and intensified recently, uncovered over 70 unique address bar spoofing vulnerabilities across 15+ different web browsers and browser-based applications on various platforms. This led to over $80,000 in bug bounty reward. Table 3 summarizes the number of unique vulnerabilities found in selected major browsers (mobile versions included).

5.2. Key Root Causes

Several recurring themes emerged as root causes:

1. Premature Address Bar Update: Changing the address bar value *before* navigation is complete and a response is received (most common), URL freezing and navigation timing.
2. Window/View Without Address Bar: Features creating content views (PiP, certain extension APIs) without mandating a secure, native address bar.

TABLE 3: Number of vulnerabilities found in major web browser vendors

Desktop+Mobile	Number of unique vulnerabilities found
Chrome	4
Firefox	9
Safari	9
Brave	5
Edge	6
Opera/Opera GX	12
Yandex	4
Arc	10

A "unique vulnerability" is defined as a distinct root cause, even if it manifested in multiple browsers based on the same engine (e.g., a single WebKit bug affecting both Safari and Arc for iOS)

3. Flawed URL Truncation/Display Logic: Incorrectly prioritizing the beginning/end of long URLs or failing to clearly indicate the true registrable domain.
4. Insufficient Prompts/Notifications: Lack of clear user notification when entering states, such as full screen, where the address bar is hidden.
5. Insecure Handling of Search Engine Queries: Weak validation of search engine origins/paths when displaying queries in the address bar.
6. Lax Parsing of Non-HTTP(S) Schemes: Treating schemes, such as data:, blob:, or internal URIs, with less rigor than http(s), leading to misinterpretation of origin.
7. Incorrect RTL/Unicode Character Handling: Failure to correctly parse and display mixed-direction text or confusable Unicode characters in domain names.

5.3. Exploitability and Severity

Vulnerabilities ranged in severity:

To move beyond qualitative descriptions, we developed a bespoke scoring rubric to consistently assess the severity of each vulnerability. This framework, summarized in Table 4, was inspired by CVSS but adapted for UI spoofing, scores vulnerabilities on two primary factors: Visual Fidelity and Interaction Complexity.

Severity is assigned based on the sum of the two scores (ranging from 2 to 6):

- High severity (Score 5–6): These vulnerabilities achieve near-perfect mimicry with minimal or no user interaction. They pose the most significant threat. Example: The Safari about: blank spoof (Fig. 5) was pixel-perfect (Score 3) and required a single click (Score 2), for a total score of 5.
- Medium severity (Score 3–4): These spoofs are convincing but have minor visual flaws or require more obvious user interaction. Example: The Brave iOS blob: URI spoof required one click (Score 2) but had minor visual inconsistencies (Score 2), for a total score of 4.
- Low severity (Score 2): These attacks are less realistic, rely on uncommon browser states, or require significant user interaction, making them less likely to succeed. Example: The Opera GX Unicode freeze required specific text selection (Score 1) and was visually flawed (Score 1), for a total score of 2.

Browsers, such as Opera and Arc, which implement many new features and adjust the address bar more toward UX and esthetic look, are more prone to spoofing attacks, even "imperfect" spoofs pose a real risk to unsuspecting users. Responsible disclosure timelines varied, with some vendors patching within days and others taking several months.

6. MITIGATION STRATEGIES

A multi-layered approach is necessary to effectively mitigate address bar spoofing, along with the browser vendor and user knowledge in the area.

6.1. Proposed Fixes and Recommendations for Browsers

1. Always Show Top-Level Domain Clearly:
   - If a URL is too long, prioritize displaying the TLD and the end of the URL path. Do not hide the TLD.
   - For multiple subdomains, ensure the registrable domain (e.g., example.com in sub.example.com) is always visible and clearly distinguished (Fig. 6).
2. Don't Change Address Value Before Response: Only update the address bar *after* a successful response from the destination server is received and content begins to render. For unreachable destinations or HTTP 204, either revert to the previous URL or clear content and display the new URL with an empty/error page (Fig. 7).
3. Show Prompts for UI State Changes: Clearly notify users when entering states, such as full-screen or landscape mode if the address bar is hidden.
4. Address Bar is for URLs, Not Search Queries: Ideally, always display the search engine's URL. If queries must be shown, use robust origin/path validation for the search engine.
5. Don't Expose Internal URIs to Web Pages: Prevent web content from redirecting to or interacting with privileged internal browser URIs.
6. Every Window Must Have a Secure Address Bar: All windows/tabs/popups/PiP displaying web content must have a non-spoofable, browser-controlled address bar

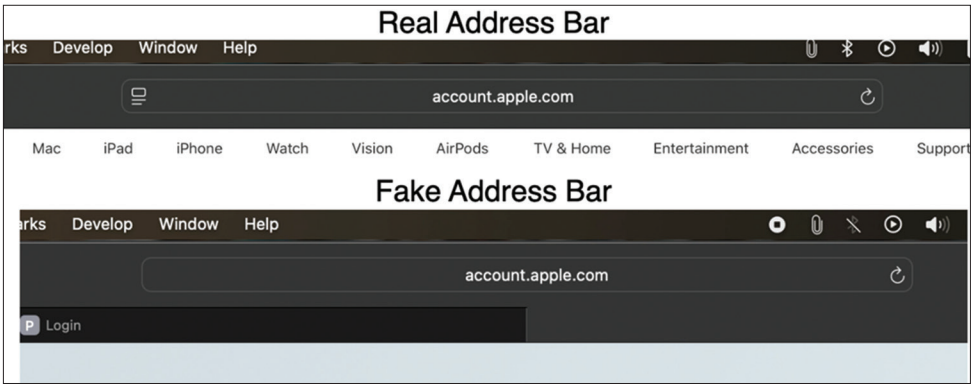

Fig. 5. High-Severity Spoof in Safari. A comparison shows an attacker-controlled page rendering a fake address bar (bottom) that is pixel-perfect and visually indistinguishable from the legitimate browser UI (top), demonstrating a highly convincing attack.

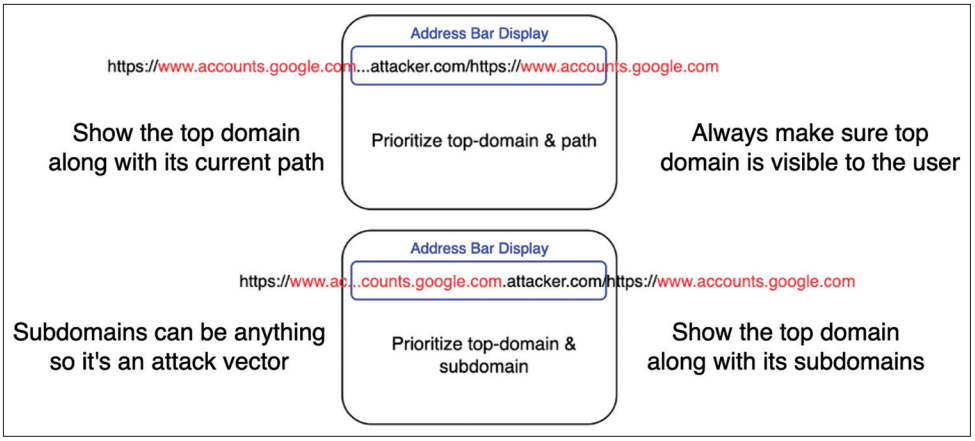

Fig. 6. Diagram illustrating the correct principle for displaying long URLs. The browser UI must always prioritize displaying the true registrable domain (attacker.com) rather than potentially deceptive subdomains, which can be arbitrarily long.

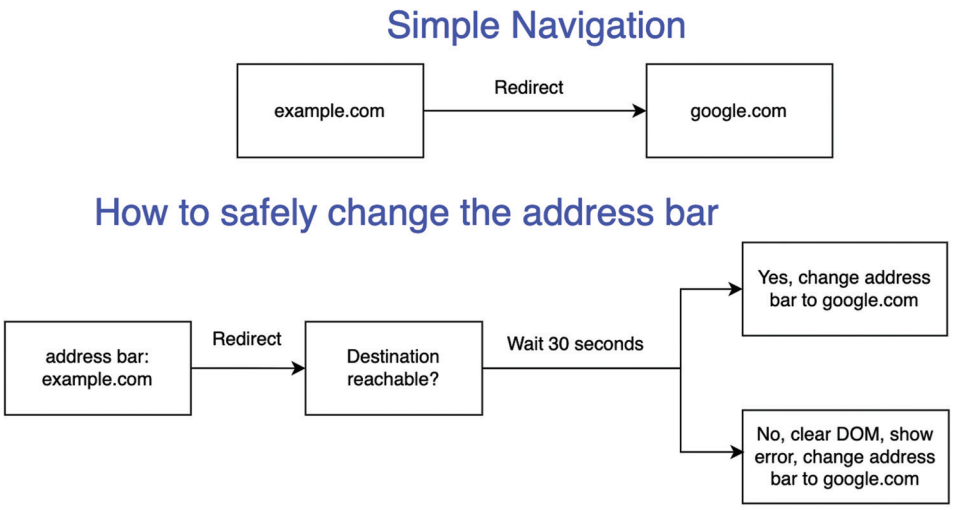

Fig. 7. Flowchart for a secure address bar update process. The UI should only change the displayed origin after receiving a successful response from the destination server, preventing timing attacks where the navigation fails.

TABLE 4: Severity Scoring Rubric for Address Bar Spoofing

Score	Visual Fidelity (How convincing is the spoof?)	Interaction Complexity (How much user action is needed?)
3	Pixel-Perfect: Indistinguishable from the legitimate browser UI to a typical user.	Zero-Click: Occurs automatically upon navigation or page load.
2	Convincing: Minor visual flaws or artifacts that an attentive user might notice.	Single-Click: Requires one simple, non-obvious user click (e.g., "Enter Fullscreen").
1	Flawed: Obvious visual defects, incorrect fonts, or missing UI elements.	Complex Interaction: Requires multiple clicks, text entry, or non-standard actions (e.g., copy-paste).

displaying the present origin.

7. Correctly Parse and Display Non-HTTP Schemes: For about: blank, blob:, data:, etc., clearly indicate the scheme and its nature (e.g., "insecure," "local data:", "blob:" URI). Do not allow hash fragments to obscure the scheme itself.
8. Correctly Display RTL Domains: Implement strict parsing and display rules for mixed RTL/LTR domains to prevent visual reordering that hides the true origin.
9. Maintain the "Line of Death": Strictly enforce separation between browser chrome (trusted UI) and web content. Prevent web content from overlaying or interacting with the address bar area, a principle core to Browsers and Applications' security UI design [13].

6.2. User Education

1. Type Sensitive Addresses Manually or Use Trusted Bookmarks: Especially for banking or critical services.
2. Use Developer Tools to Verify Origin: In case of suspicion, console.log (location.origin) reveals the true origin or Inspect site identity via browser padlock icon/certificate details.
3. Keep Browsers Updated: Apply security patches promptly.
4. Be Wary of Unexpected Behavior: If a site looks slightly off or behaves unusually after clicking a link, exercise caution.

7. DISCUSSION AND CONCLUSION

This research has systematically demonstrated that address bar spoofing remains a significant and evolving threat in contemporary web browsers. By developing a comprehensive taxonomy, discovering and disclosing over 70 vulnerabilities, and analyzing their root causes, we have provided a deep insight into the present state of address bar security.

Our analysis of the findings in Table 3 reveals notable trends. Chromium-based browsers that implement heavy UI customizations, such as Opera, Opera GX, and Arc, accounted for a disproportionately high number of vulnerabilities. These often stemmed from unique features, such as custom search engine integrations or non-standard UI state handling, suggesting that deviations from the core Chromium security model introduce new attack surfaces. In contrast, vulnerabilities in WebKit (Safari) were more frequently tied to the nuanced handling of non-standard URI schemes, such as blob:, while Gecko (Firefox) showed susceptibility to URL truncation on mobile displays. This highlights that while engine-level security is mature, browser-specific feature implementations remain a primary source of spoofing vulnerabilities.

The findings underscore a persistent between usability and security. Features designed to create a cleaner UX – such as fast address bar updates, hiding URL components, and novel UI states, such as Picture-in-Picture – are frequently the source of the security flaws we identified. While this research was extensive, it could not cover every browser or operating system, and niche browsers may harbor similar flaws.

In conclusion, maintaining the integrity of the address bar requires continuous, meticulous attention to detail across navigation logic, protocol handling, and UI rendering. Our proposed mitigation strategies offer a multi-layered defense to help vendors fortify this critical UI element. For future work, we recommend two key directions: (1) Expanding testing to include embedded web views and browser extensions, which represent growing attack surfaces, and (2) developing standardized, automated testing tools to systematically probe for these complex UI-based vulnerabilities at scale. The insights from this research aim to significantly advance the understanding and fortification of this crucial element of web security, ultimately working toward a safer online experience.

REFERENCES

[1] *StationX, "Phishing Statistics"*, 2024. Available from: https://www.stationx.net/phishing-statistics [Last accessed on 2025 Jul 10].

[2] "*Trend Micro. Address Bar Spoofing.*" Available from: https://www.trendmicro.com/vinfo/us/security/definition/address-bar-spoofing [Last accessed on 2025 Jul 10].

[3] R. Ismael. "*Address Bar Spoofing. GitHub Repository*," 2025. Available from: https://github.com/renwax23/address_bar_

spoofing [Last accessed on 2025 Jul 10].

[4] *Chromium. "Guidelines for URL Display*." Available from: https://chromium.googlesource.com/chromium/src/+/HEAD/docs/security/url_display_guidelines/url_display_guidelines.md [Last accessed on 2025 Jul 10].

[5] Y. Koster. "*Address Bar Spoofing flaw in Internet Explorer,"* 2004. Available from: https://www.akitasecurity.nl/advisory/AK20040801/address_bar_spoofing_flaw_in_internet_explorer.html [Last accessed on 2025 Jul 10].

[6] L. Treiber. "*Google Chrome HTTPS Address Bar Spoofing*," 2012. Available from: https://blog.acrossecurity.com/2012/01/google-chrome-https-address-bar.html [Last accessed on 2025 Jul 10].

[7] C. Weber. "*Unicode Security Guide - Visual Spoofing*," 2014. Available from: https://cweb.github.io/unicode-security-guide/visual-spoofing

[8] R. Baloch. "*Bypassing Mobile Browser Security for Fun and Profit. In: Presented at BlackHat Asia*," 2016.

[9] *"Chromium. Chromium Trickuri GitHub Repo*," 2019. Available from: https://github.com/chromium/trickuri [Last accessed on 2025 Jul 10].

[10] R. Baloch. "*Multiple Address Bar Spoofing Vulnerabilities in Mobile Browsers,"* 2020. Available from: https://www.rafaybaloch.com/2020/10/multiple-address-bar-spoofing-vulnerabilities.html

[11] *Skylot, "Jadx GitHub Repo*." Available from: https://github.com/skylot/jadx [Last accessed on 2025 Jul 10].

[12] M. Bentkowski. *Address Bar Spoofing in Chrome and Firefox*, 2017. Available from: https://research.securitum.com/address-bar-spoofing-in-chrome-and-firefox-description-of-cve-2017-5089-and-cve-2017-7763 [Last accessed on 2025 Jul 10].

[13] E. Law. "*The Line of Death*," 2017. Available from: https://textslashplain.com/2017/01/14/the-line-of-death [Last accessed on 2025 Jul 10].

[14] A. N. Joinson, U. D. Reips, T. Buchanan and C. B. P. Schofield. "Privacy, trust, and self-disclosure online." *Human-Computer Interaction*, vol. 25, no. 1, pp. 1-24, 2010.

[15] K. Yee. "Aligning security and usability." *IEEE Security and Privacy*, vol. 2, no. 5, pp. 85-88, 2004.

APPENDIX A: SUMMARY OF DISCLOSED VULNERABILITIES

The following table summarizes the 74 unique vulnerabilities identified and disclosed during this research. The full dataset, including PoC codes, image/video PoCs, bounty amount, timeline and full bug description, is available at the repository cited in Ismael [3]. The "Taxonomy Class" refers to the categories defined in Section 4. Note: Not all browsers assign CVEs, which is why some are missing.

ID	Title	Technique	Browser	CVE
1	Bypassing CVE-2019-10875 or Xiaomi's Mint Browser's URL Spoofing patch	Search-Engine	Xiaomi Mint	0
2	0-day Alert: URL Spoofing Bypassed for the latest Mint Browser 1.6.4	Search-Engine	Xiaomi Mint	0
3	Opera Mini Browser Address Bar Spoof	Internal-URI	Opera Mini	0
4	Full Address Bar Spoofing in Opera Touch and Opera GX Mobile	Search-Engine	Opera GX Mobile	0
5	Opera GX For Android URL Spoof/Freeze	URL-Freeze	Opera GX Mobile	0
6	XSS in play.gx.games to RCE, Full Address Bar Spoof in Opera GX	Window-Without-Address-Bar	Opera GX	0
7	Opera for Android Address Bar Spoof	Fullscreen	Opera Android	0
8	Opera Mini for Android Address Bar Spoof	Fullscreen	Opera Mini	0
9	Full Address Bar Spoof and Browser Takeover Inside Opera GX	Window-Without-Address-Bar	Opera GX	0
10	Opera GX Android - Address Bar Spoof with intent://vtp.operagx.gg/?url=	Internal-URI	Opera GX Android	0
11	Opera Mini - Address Bar Spoof with opera-mini://open?url=data:	Internal-URI	Opera Mini	0
12	Iframe Injection in GX Corner to Address Bar Spoofing in Opera GX	Window-Without-Address-Bar	Opera GX	0
13	Opera Mini - Full Address Bar Spoof	End-of-URL-Shown	Opera Mini	0
14	Opera Mac/Windows - New Window Address Bar Spoof using Picture-in-Picture	Window-Without-Address-Bar	Opera	0
15	Brave iOS Address Bar Spoofing Using blob: URI	blob:-URI	Brave	0
16	Brave iOS Address Bar Spoofing Using about: blank URI	about: blank-URI	Brave	0
17	Brave iOS Address Bar Spoof Using 2 RTL (Arabic Characters) Subdomains	RTL-Character	Brave	0
18	Brave iOS Address Bar Spoof Using RTL Domain with RTL+LTR Subdomains	RTL-Character	Brave	0
19	Firefox Desktop Address Bar Spoofing Using Arabic Punycode and a long URL	RTL-Character	Firefox	CVE-2024-11695
20	Edge - Mac - Address Bar Spoofing using Picture-in-Picture	Window-Withour-Address-Bar	Edge Mac	CVE-2024-38093
21	Edge Mobile (Android-iOS) Address Bar Spoof Using Bing Search	Search-Engine	Edge Mobile	CVE-2024-38083
22	Zerion Wallet App For iOS Address Bar and Wallet Confirmation Origin Spoof	Long-Subdomain	Zerion Wallet	0
23	Coinbase Wallet App For iOS Address Bar Origin Spoof	Long-Subdomain	Coinbase Wallet	0
24	Address Bar Spoofing Using about: blank URI	about: blank-URI	Arc Mac	0
25	Address Bar Spoofing using data: URI	data:-URI	Arc Mac	0
26	Address Bar Spoofing using documentPictureInPicture	Window-Without-Address-Bar	Arc Mac	0
27	Address Bar Spoof Using Fullscreen	Fullscreen	Arc Mac	0
28	Address Bar Spoof in Arc Search iOS	Invalid-Scheme	Arc iOS	0
29	Arc Search iOS Full Address Bar Spoof Using Blob: URI	blob:-URI	Arc iOS	0
30	Arc Search iOS Address Bar Spoof Using Search Functionality	Search-Engine	Arc iOS	0
31	Arc Search Android Full Address Bar Spoof Using Custom Port Website Hanging	URL-Freeze	Arc Android	0
32	Arc Searc Android Address Bar Spoof Using Long Subdomain	Long-Subdomain	Arc Android	0
33	Arc Search Android Address Bar Spoof Using Fullscreen	Fullscreen	Arc Android	0
34	Arc Search Android Full Address Bar Spoof by Redirecting to a 204 No Content Page	URL-Freeze	Arc Android	0

ID	Title	Technique	Browser	CVE
35	Arc Browser address bar spoof using view-source:	view-source:-URI	Arc Mac	0
36	Address Bar Spoofing Using blob: URI	blob:-URI	Arc Mac	0
37	Arc Browser address bar spoof using file://URI	file:-URI	Arc Mac	0
38	XSS in arcmobile-internal://using Server-Side Redirect to JavaScript: URI to Full Address Bar Spoof in Arc Search iOS	Internal-XSS	Arc iOS	0
39	Stored XSS in search.arc.net to Address Bar Spoof in Arc iOS and Potential UXSS	Search-Engine	Arc iOS	0
40	Firefox Mobile Address Bar Spoof Using Long Subdomain	Long-Subdomain	Firefox Mobile	0
41	Firefox Mobile iOS Full Address Bar Spoof Using Open in New Tab and JavaScript: URI	JavaScript:-URI	Firefox iOS	CVE-2025-23108
42	Firefox Mobile iOS Full Address Bar Spoof Using Server-Side Redirect to internal://local/errorpage	Internal-URI	Firefox iOS	CVE-2025-27426
43	Firefox Mobile iOS Address Bar Spoof Using Server-Side Redirect to non-http Scheme	Invalid-Scheme	Firefox iOS	CVE-2025-27424
44	Yandex iOS Address Bar Spoofing Using blob: URI	blob:-URI	Yandex iOS	0
45	Yandex Browser iOS Address Bar Spoofing Using about: blank UR	about: blank-URI	Yandex iOS	0
46	Yandex Browser iOS Address Bar Spoof Using 2 RTL (Arabic Characters) Subdomains	RTL-Character	Yandex iOS	0
47	Google App iOS Full Address Bar Spoof	about: blank-URI	Google iOS	0
48	Brave iOS Address Bar Spoof and Potential UXSS Using brave://open-url?url=	Internal-URI	Brave iOS	0
49	OKX App iOS-Android dApp Address Bar Origin Soof	URL-Freeze	OKX	0
50	Coinbase Wallet App For iOS Address Bar Origin Spoof	Long-Subdomain	Coinbase Wallet iOS	0
51	PlayStation 5 Browser Full Address Bar Spoofing	JavaScript:-URI	PlayStation 5 Browser	0
52	Safari New Window Address Bar Spoof With a Subdomain or Long Domain	Long-Subdomain	Safari	0
53	Address Bar Spoof Using Invalid Protocol	Invalid-Scheme	Naver WHALE	0
54	Safari iOS Full Address Bar Spoof Using Tab Hanging	URL-Freeze	Safari	CVE-2025-24128
55	Safari Mac/iOS Full Address Bar Spoof Using Tab Hanging and Invalid URLS	URL-Freeze	Safari	CVE-2025-24128
56	Safari Mac and iOS Address Bar Spoof	about: blank-URI	Safari	CVE-2025-24113
57	Safari Mac/iOS Browser Address Bar Spoof Using Search Engines	Search-Engine	Safari	CVE-2025-30467
58	Safari Mac Address Bar Spoof Using Cursor Overlap	overlap	Safari	
59	Safari Address Bar Spoof Using blob: URI	blob:-URI	Safari	CVE-2025-24113
60	Safari New Window Address Bar Spoof With a Subdomain or Long Domain	Long-Subdomain	Safari	CVE-2025-31266
61	Safari Browser Address Bar Spoof Using Redirect	URL-Freeze	Safari	CVE-2025-43327
62	Firefox Desktop Address Bar Spoof Using Search Query (bypass fix in bug 1970997)	Search-Engine	Firefox	CVE-2025-9183
63	Rabby Wallet iOS Browser Address Bar Origin Spoof to Account Takeovers	Tab-Hanging	Rabby Wallet	0
64	Arc iOS Full Address Bar Spoof Using Timing Attack	URL-Freeze	Arc iOS	0
65	Arc iOS Full Address Bar Spoof Using Search Functionality and Race Condition	Search-Engine	Arc iOS	0
66	MetaMask iOS Address Bar Spoof Using Drag-Drop	URL-Freeze	MetaMask iOS	0
67	Brave iOS Address Bar Spoof Using about: blank	about: blank-URI	Brave iOS	0
68	MetaMask Wallet iOS - Browser Full Address Bar Spoof	Tab-Hanging	MetaMask iOS	0
69	Brave iOS Address Bar Spoof with open-url and data: URI	data:-URI	Brave iOS	0
70	Brave iOS Drag-Drop Address Bar Spoof	URL-Freeze	Brave iOS	0
71	Address Bar Spoof and UXSS Using Shortcuts Chrome iOS	End-of-URL-Shown	Chrome iOS	0
72	Firefox Address Bar Spoof with Long Subdomain and Popup	Long-Subdomain	Firefox	0
73	Edge Mobile (Android-iOS) Full Address Bar Spoof Using Bing and Path Traversal	Search-Engine	Edge Mobile	0
74	Edge iOS Address Bar Spoof Using 2 RTL (Arabic Characters) Subdomains	RTL-Character	Edge iOS	0