What Load-Time Accessibility Checks Miss on Ad-Supported News Sites

Muhammad Hassnain
Department of Computer Science
University of California, Davis
Davis, CA, USA
mhassnain@ucdavis.edu

***Abstract*—Visually impaired users rely on screen readers to browse the web. These screen readers need web pages, including the third-party ads they carry, to be accessible. To meet accessibility guidelines such as WCAG, industry tests pages with automated checkers. These checkers usually audit a page once, at load, and only its top-level document. On ad-supported pages, these two choices miss failures: ads keep loading as the user scrolls, and they render in cross-origin frames that such checks do not reach. We built a test harness that audits each page before and after a minute of scripted use, repeats every visit with ad vendors blocked, and audits inside every cross-origin frame. On 2,298 news sites, we find that load-time checks miss real failures: after a minute of use, 40.7% of sites fail rules that did not fail at load, often only inside frames. Ads drive much of this: blocking them reduces new failures on far more sites than it increases them, and the effect grows with the number of ad vendors a site loads. Matching failing markup against ad patterns, a cheaper alternative to blocking, works for missing names but misattributes failures of page structure. On 36.0% of sites, a visible cross-origin frame from an ad vendor or an embed fails a WCAG rule that the publisher cannot fix. We discuss what this means for testing practice, failure attribution and WCAG conformance claims.**

***Index Terms*—web accessibility, accessibility testing, WCAG, dynamic analysis, third-party components, online advertising**

I. Introduction

People with disabilities use the web for news, work and public services, often through a screen reader. Whether they can do so depends on how developers build each page. The Web Content Accessibility Guidelines (WCAG) [1] define what an accessible page must do. They underlie Section 508 [2] in the United States and EN 301 549 [3], the standard on which the European Accessibility Act [4] relies.

Much of the web is paid for by advertising, and ads are a known source of accessibility failures. Amjad et al. [5] audited about 5,000 websites with and without an ad blocker and found that ads added violations on 67% of them. He et al. [6] found issues in 84.4% of Android ad screens. These failures are hard for publishers to fix, because third-party vendors write and serve the ads.

Developers usually check accessibility with automated tools such as axe-core [7]. Teams run them in their test suites, and large studies run them over thousands of sites [8]. These tools cover at most half of WCAG's success criteria [9]. Even within that coverage, they have two blind spots that matter on ad-supported pages: *when* they look, and *where*. Most audits run once, on the loaded page, before any interaction [8], [10], although what a checker finds depends on when it runs [11], [12]. Ad-supported pages keep changing after load: ad slots fill as the user scrolls, creatives refresh, and overlays appear. Ads also render inside cross-origin frames, which a checker injected into the publisher's top-level document cannot read. The ad studies above share the first blind spot. They audit the page once, without interaction, or the ad's markup alone, and one of them notes that its pipeline does "not capture changes in accessibility violations triggered by user interactions" [5].

This paper measures what these blind spots hide, and who is responsible for it. We built a test harness that combines three techniques. It audits each page before and after a scripted *exercise* of scrolling, waiting, tabbing and hovering. It repeats every visit with requests to ad vendors blocked, so that each site is its own *counterfactual*. And it audits *in place* inside every cross-origin frame, recording who serves it. Each technique has precedents [5], [12], [13]. Combined, they separate what a load-time audit misses from what the ads cause. We apply the harness to 2,298 sites drawn from news corpora and ask:

RQ1 Missed failures. What does an audit after a minute of use find that the load-time audit does not?

RQ2 Attribution. Which of these new failures do ads cause, and does matching failing markup against ad patterns find them correctly?

RQ3 Inside frames. What violations sit inside cross-origin frames, and who serves those frames?

A load-time audit rarely misjudges *whether* a page fails: on 99.2% of sites, it already does. It misjudges *what* fails. On 40.7% of sites, a rule that did not fail at load fails after a minute of use, and on half of these only inside frames (RQ1). With ads, far more sites gain extra violations during use than gain fewer, and the gap grows with the number of ad vendors a site loads (RQ2). Markup-based attribution agrees with blocking on rules about accessible names, but on rules about page structure it misses native-ad feeds on a few sites and over-credits ads elsewhere. On 36.0% of sites, a visible cross-origin frame fails a WCAG rule, and on 60% of these a vendor on our ad block list serves it (RQ3).

This paper contributes:

- a test harness that exercises each page, re-runs it with ad vendors blocked, and audits inside frames (Section III

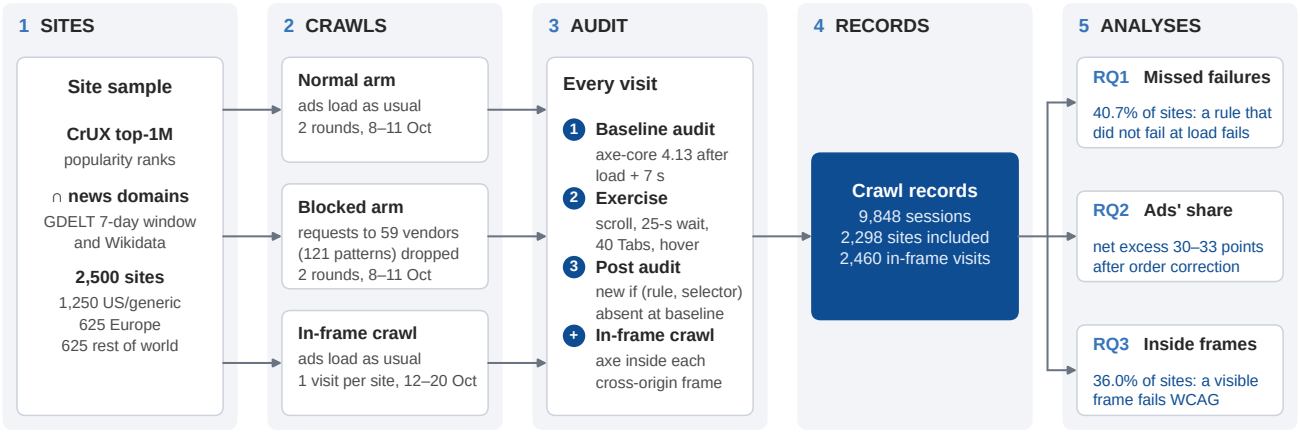


Fig. 1. The harness and the study. (1) We sample 2,500 news sites. (2) We crawl each site with ads (*normal* arm) and with ad vendors blocked (*blocked* arm); a separate in-frame crawl audits inside frames. (3) axe-core audits every visit before and after an exercise of scrolling, waiting, tabbing and hovering. (4) The records answer (5) the three research questions, shown with their headline results. The difference between the arms estimates the ads' share.

and Fig. 1), released with our data and scripts;
- a study of 2,298 sites answering RQ1–RQ3 (Section IV);
- implications for testing, failure attribution, AI repair tools and WCAG conformance claims (Section V).

II. Related Work

Accessibility of ads. Amjad et al. [5] compared pages with and without an ad blocker, auditing each once, without interaction. Yeung et al. [13] extracted ads from 90 websites, including ads in nested frames, and audited them against a subset of WCAG. Their blind participants described pop-up ads that are hard to close. He et al. [6] built AdMole to find accessibility issues in Android ad screens. The gestures that screen-reader users perform on ads also trigger tracking [14]. These studies audit the page once, or the ad on its own. We audit the whole page in place, before and after use, and test how well markup-based attribution works.

Automated accessibility testing. Vigo et al. [9] measured what automated checkers miss. Later tools infer a page's semantic structure from its visual layout [15], detect keyboard failures by modeling keyboard navigation [16], [17], drive a real screen reader [18], or check temporal-logic properties of navigation [19]. These tools improve *what* a checker can judge. Our concern is *when* and *where* it looks.

Dynamic accessibility evaluation. Crawljax [20] showed that a tester must reach a web application's states through interaction before testing them. Fernandes et al. [12] audited pages before and after the browser processed them, and after triggering interaction events, and found that the results differed considerably. QualState [21] and Demodocus [22] crawl an application's states by triggering its interactive elements, and Flow-A11y [23] runs user-defined scenarios to test WCAG criteria that only interaction reveals. These tools test the publisher's own interface; we measure what third-party code adds to it during use.

III. Study Design

Fig. 1 summarizes the study. We sample sites from news corpora (Section III-A), audit each visit before and after an exercise (Section III-B), visit each site with and without ads (Section III-D), and audit inside cross-origin frames in a separate crawl (Section III-F). We built the harness on Playwright 1.63 [24]. It drives headless Chromium 153 from a US vantage point, with an `en-US` locale.

A. Sites

We sample from the Chrome UX Report (CrUX) top-1M list [25], restricted to domains that appear as news sources in a seven-day GDELT window [26] or in Wikidata. We pin both snapshots. We take 2,500 sites in order of CrUX rank bucket, with three regional quotas by top-level domain: 1,250 US or generic sites, 625 European sites, and 625 from the rest of the world. We then exclude 35 platform and government domains (e.g., `amazon.com`, `cdc.gov`), which leaves 2,465.

Not every site in a news corpus is a news publisher. Automatic labeling with domain rules, page-feature rules and an LLM proved unreliable (pairwise Cohen's $\kappa \leq 0.27$). The author therefore labeled a random sample of 50 sites by hand. Of these, 41 (82.0%, 95% CI 69.2–90.2) were news publishers. "News sites" is thus shorthand for sites from news corpora.

B. Audit, exercise, audit

Each visit opens the site in a fresh browser context, with a 1366×900 viewport, and has three steps.

Baseline audit. Seven seconds after the `load` event, the harness runs axe-core 4.13 [7] with its default rules. It uses axe-core's Playwright integration, which injects the checker into every frame it can reach, cross-origin frames included. It also forces closed shadow roots open, so that axe-core can audit them. The default rules include best-practice rules that go beyond WCAG, such as `region`. We therefore say *violation* for any rule failure, say that a page *fails* if it has at least one violation, and report WCAG-tagged rules separately.

Exercise. The harness scrolls with the mouse wheel in 540-pixel steps, for up to 20 steps. It pauses 0.9 seconds at each step, so that lazy-loaded ad slots can fill, and then scrolls inner containers to their end. It waits 25 seconds, presses Tab 40 times, and hovers over up to 15 ad-like elements to trigger expand-on-hover creatives. Finally, for each ad-like overlay inserted after the DOM was ready, it focuses the overlay's first focusable element and presses Tab up to ten more times. The exercise takes about a minute.

Post audit. After a 2.5-second pause, the harness runs axe-core again.

C. Measures

New violations. A violating node is *new* if its `(rule, selector)` pair is absent from the baseline audit. For a node inside a frame, the selector is that of the frame element. Before comparing, we canonicalize selectors. We normalize hash-like tokens (four or more hex digits) in ids and class names and drop `:nth-child` indices, so that an ad re-rendered with a fresh id does not count as new. Canonicalization misses some re-renders: for 42.7% of new nodes, the same rule already failed at baseline on a node with the same HTML, or on a selector that differs only in digits. Some of these are genuinely new copies of an element, so we report results with and without them.

Newly failing rules. A new node may repeat a failure that the load-time audit already reports. A stricter measure therefore counts a rule as *newly failing* if it fails after the exercise but failed nowhere on the page at load, either because it passed or because nothing on the page was subject to it yet.

Inclusion. A visit is *included* if the harness detected no bot wall or hard consent wall at load, the HTTP status was below 400, both audits ran, and the harness pressed Tab at least ten times. A site counts for an outcome if the outcome occurs on at least one of its included visits. We also report the stricter criterion of a majority of its included visits, which for the 2,006 sites with two included visits means both. Proportions carry Wilson 95% confidence intervals, and other estimates carry bootstrap 95% intervals.

D. Visiting with and without ads

To estimate the ads' share, we visit each site in two arms. In the *normal* arm, ads load as usual. In the *blocked* arm, the browser drops every subresource request to one of 59 vendors on a hand-curated *block list*. We call these *listed vendors*. The list targets the advertising ecosystem as a whole. Besides ad servers and exchanges, it includes analytics and tracking vendors, and six services that draw their own interface, such as a paywall and a video player. We crawled each arm in three rounds between 8 and 11 October 2026, alternating between the arms. A DNS outage on the crawl machine disrupted one normal-arm round, so we analyze two rounds per arm.

For each site, we pair the normal and blocked visits of the same round and compute

$$\Delta = \mathrm{median}_k \left(\mathrm{new}_{\mathrm{normal},k} - \mathrm{new}_{\mathrm{blocked},k} \right)$$

over the rounds k in which both visits are included. Here, new is the number of new violating nodes. $\Delta > 0$ means that the site gained more new violations with ads than without. A few very large differences would dominate an average, so our main measure is the sign of Δ. We summarize its size with the Hodges–Lehmann estimator, the median of all pairwise averages of the per-site values.

Δ could differ from zero for reasons other than the ads, so we run three checks. The first is chance: a page can change between two visits even when nothing else does. On sites with two included normal visits, we compare these two identical visits to see how much Δ can vary by chance. The second is visit order: the normal visit came first in all but 3 of 4,113 pairs, a median of 15.3 hours before the blocked visit. To see whether going first matters, we also pair each site's first blocked visit with its later normal visit, so that the blocked visit comes first. The third is a placebo test. On sites that request no listed vendor, blocking removes nothing, so if the ads cause Δ, it should be balanced there.

E. Markup-based attribution

Blocking needs a second visit to every site. A cheaper way to find failures caused by ads is to look at the failing element and check whether its markup looks like an ad. We call this *selector matching*, and RQ2 compares it with blocking. It tries three patterns in order, first on the element's CSS selector and then, if none matches, on its HTML. The first matches cookie banners, so that they are never counted as ads. The second, the *ad-token* pattern, marks the element as an ad if it names an ad network or an ad slot, such as `div-gpt-ad`. The third, the *sponsored-content* pattern, catches *native ads*: recommendation widgets that label their items as sponsored rather than as ads.

F. Auditing inside cross-origin frames

The main audit reaches into cross-origin frames, but it reports an in-frame failure only as a chain of selectors, without the frame's domain or size. A separate *in-frame crawl* records both (12–20 October 2026, one normal-arm visit per site). We set Chromium's site-isolation flags so that every ad frame runs in a separate process [27], and run axe-core with WCAG level A and AA rules inside each cross-origin frame, nested frames included. We exclude four rules that apply only to a whole document, such as requiring a page title.

Many ad-tech frames are tiny or hidden, and some exist only to synchronize user identifiers between vendors [28]. We therefore count a frame only if its body covers more than $2{,}500\,\mathrm{px}^2$, and call such a frame *visible*. We give each frame 5 seconds and audit at most 25 frames per site, in page order. The harness hit this limit on 461 sites and drops failed audits, so these results are a lower bound.

G. Consent and ethics

The harness does not click cookie banners. Instead, in both arms, it imitates the consent interfaces that ad code queries, such as IAB Europe's Transparency and Consent Framework (TCF) [29], and reports consent to all purposes. Publishers that

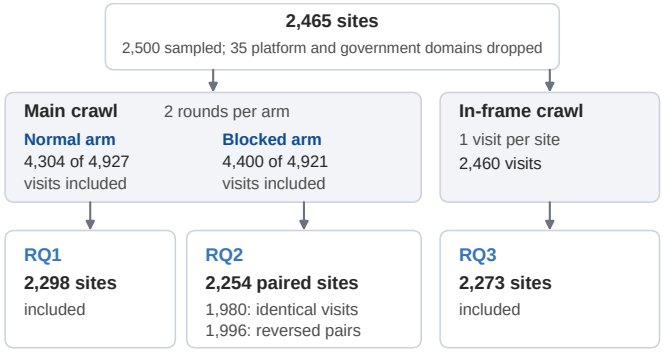


Fig. 2. From the sample to the sites behind each research question. A visit is included if it meets the inclusion rule (Section III-C).

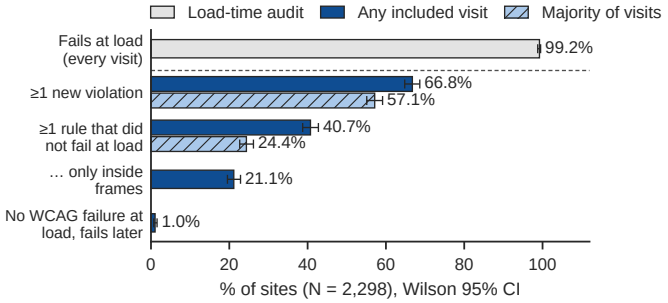


Fig. 3. RQ1. Nearly every site fails at load. After a minute of use, many fail a rule that did not fail at load, often only inside frames. Few go from no WCAG failure to failing. Share of 2,298 sites, on any included visit (dark) or a majority of visits (light, hatched). Wilson 95% CIs.

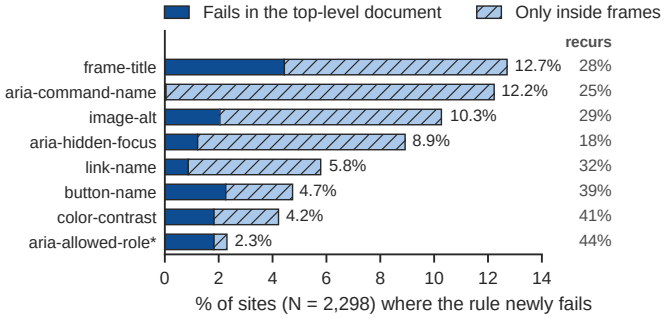


Fig. 4. RQ1. Which rules newly fail after a minute of use. A rule newly fails on a site if it fails after the exercise but failed nowhere on the page at load. Bars: share of 2,298 sites where each of the eight most common such rules newly fails, either in the top-level document (dark) or only inside frames (light, hatched). Recurs: of the sites with two visits where the rule newly fails on at least one, the share where it newly fails on both. *Best-practice rule, not part of WCAG.

hold back ads until the user consents therefore load them in our visits. The consent record it passes to vendors, however, lists no vendors (Section VI).

The study involved no human subjects. Each site received seven homepage loads, plus retries: three per arm, including the round we do not analyze, and one in the in-frame crawl. We made no clicks, logins or form submissions. Our loads rendered ads that advertisers may pay for, and our Tab and hover gestures triggered third-party tracking [14], about 12.5 requests to listed vendors per site on average (median 0). The consent our harness reported to vendors was automated. We accepted these costs because measuring the page as users see it requires loading its ads.

IV. Results

Fig. 2 shows how many visits and sites each analysis uses. We included 87% of normal-arm visits; most of the others ended before ten Tab presses, met a bot or consent wall, or returned an HTTP error.

A. RQ1: What does the load-time audit miss?

A load-time audit rarely misjudges *whether* a page fails, that is, has at least one violation. Nearly every page already does: on 99.2% of sites, every included visit fails at load, with a median of 61 failing nodes per visit. Use seldom changes this verdict. Only 24 sites (1.0%) have no WCAG failure at load and fail a WCAG rule after the exercise.

What the load-time audit misjudges is *which* failures a page has. After the exercise, 66.8% of sites have at least one new violation, or 59.8% if we ignore elements that look re-rendered. But many of these new violations only repeat a rule that already failed at load. We therefore also count *newly failing rules*: rules that fail after the exercise but did not fail anywhere on the page at load. At least one rule newly fails on **40.7%** of sites (38.7–42.8), or 37.6% for WCAG rules (Fig. 3). On half of these sites, the newly failing rules fail only inside frames. Our main audit reaches into frames, but a checker that runs only in the top-level document cannot see these failures. Section IV-C examines the frames themselves.

Fig. 4 shows which rules newly fail. Most are WCAG rules about names, failing inside frames: `frame-title` on 12.7% of sites, `aria-command-name` on 12.2% and `image-alt` on 10.3%. The best-practice rule `region` adds many new nodes but newly fails on only 1.5% of sites, because where it gains nodes, it had already failed at load.

Newly failing rules do not always recur. Of all 2,298 sites, 40.7% have a newly failing rule on at least one visit, but only 24.4% have one on every visit. Individual rules often do not recur either. Among sites with two visits where a common rule newly fails, that rule newly fails on both visits on only 18–41% of sites, depending on the rule (Fig. 4). Finally, the exercise mixes about a minute of waiting with the gestures, so we cannot tell whether the gestures or the waiting caused these failures.

RQ1 finding: Nearly every page already fails at load; a later audit changes which rules fail. On 40.7% of sites, a rule that did not fail at load fails after a minute of use, on half of these sites only inside frames. Only 24.4% of sites see this on every visit. **Implication:** audit again after a scripted interaction, more than once, and report the rules that newly fail.

B. RQ2: Which new failures do the ads cause?

About half of sites gain more new violations with ads than without, and few gain fewer. Of the 2,254 sites with an included

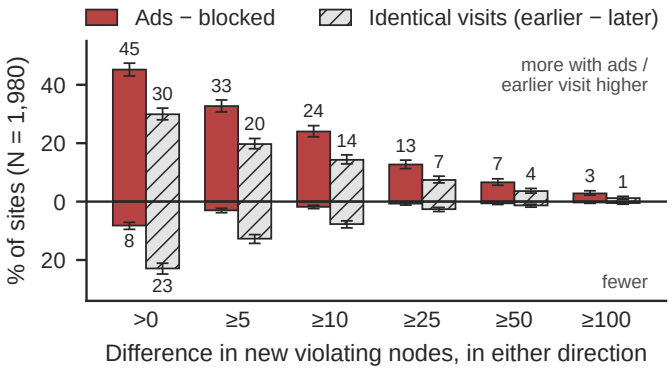


Fig. 5. RQ2. How much the number of new violations changes. Red: with ads minus with ads blocked. Gray, hatched: the earlier minus the later of two identical visits, which shows changes due to chance. Bars above zero: share of sites whose count rises by at least the amount on the axis; below zero: falls by at least that amount. 1,980 sites with two included normal visits and a paired visit. Wilson 95% CIs.

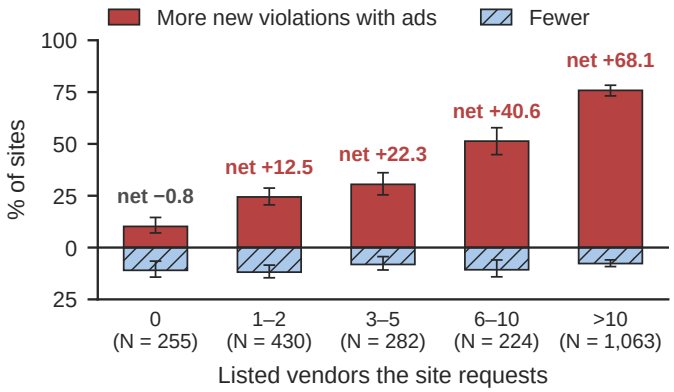


Fig. 6. RQ2. Share of paired sites with more (up) and fewer (down) new violations with ads than with ads blocked, by how many listed vendors the site requests. With none, blocking removes nothing: a placebo. Net: the difference between the two shares, in points. Wilson 95% CIs.

pair, $\Delta > 0$ on **50.5%** (48.4–52.5) and $\Delta < 0$ on only **9.2%**; the rest show no difference. A typical site gains 3.5 more new violating nodes with ads (Hodges–Lehmann estimate, 95% CI 3.0–4.0). The result also holds when we count differently. We try four variants: (a) WCAG rules only; (b) no nodes that look re-rendered; (c) only pairs whose visits scrolled the same distance; and (d) the change in all violating nodes. Across these, $\Delta > 0$ on 44–50% of sites and $\Delta < 0$ on 7–16%. So however we count, ads more often add new violations than reduce them.

Is it chance or visit order? Pages change between visits even when nothing else does. Two identical normal visits of a site give different counts of new violations on about half of sites (52.8%), so Δ says little about any single site. This noise has little direction: the earlier visit is higher only 7.1 percentage points more often than the later one. The ads' effect is much larger. On the same 1,980 sites, $\Delta > 0$ exceeds $\Delta < 0$ by 37.0 points, which leaves 29.9 points (95% CI 26.2–33.7) after we subtract the 7.1 points of chance and order. Putting the blocked visit first gives 32.8 points. Visit order therefore explains little: ads add new violations on 30–33 percentage points more sites than they reduce them. Large differences tell the same story (Fig. 5). With ads, 24.0% of sites gain at least ten more new violating nodes and only 1.8% gain ten fewer; between identical visits, these shares are 14.3% and 7.7%. So by chance, large changes go both ways, but with ads they almost always go up.

Is it the ads? If the ads cause Δ, blocking should change nothing on sites that load no listed vendor. It does not: on these 255 sites, Δ is positive and negative about equally often (10.2% and 11.0%), so visit order and time do not create the effect by themselves. The effect also grows with the number of listed vendors a site loads (Fig. 6). The gap between $\Delta > 0$ and $\Delta < 0$ rises from 12.5 points on sites with one or two vendors to 68.1 points on sites with more than ten. Both results point to the ads, not to how we visited the sites. Our block list also blocks six services that are not ads, such as a paywall. On the 1,267 sites that load none of them, Δ is still positive on 45.1% of sites and negative on 9.8%, close to the 50.5% and 9.2% on all sites, so these services do not explain the effect.

TABLE I

RQ2. SHARE OF EACH RULE'S NEW VIOLATIONS THAT EACH METHOD ASSIGNS TO ADS. SELECTOR: MATCHING WITH THE AD-TOKEN PATTERN (AD), AND WITH THE SPONSORED-CONTENT PATTERN ADDED (+SP.). COUNTERFACTUAL: THE DROP IN NEW VIOLATIONS WHEN ADS ARE BLOCKED, AS A SHARE OF NEW VIOLATIONS WITH ADS, WITH THE NORMAL VISIT FIRST (N FIRST) OR THE BLOCKED VISIT FIRST (B FIRST). NAMING: 17 RULES ON ACCESSIBLE NAMES. STRUCTURE: 23 RULES ON PAGE STRUCTURE AND PRESENTATION.

	Selector		Counterfactual	
Rule	**Ad**	**+Sp.**	**N first**	**B first**
Naming				
`role-img-alt`	83.1%	83.1%	94.5%	92.9%
`frame-title`	70.3%	70.3%	87.0%	80.4%
`aria-command-name`	74.4%	74.4%	89.5%	86.1%
`image-alt`	61.8%	62.1%	67.8%	55.2%
`link-name`	53.7%	55.4%	56.7%	57.6%
`button-name`	17.5%	17.5%	18.2%	7.5%
Structure				
`region`	12.7%	40.2%	43.8%	39.3%
`color-contrast`	5.3%	33.0%	33.7%	23.4%
Naming family	**59.5%**	**60.1%**	**67.1%**	**59.7%**
Structure family	**11.0%**	**37.6%**	**40.1%**	**34.3%**

Does selector matching agree? Table I compares, rule by rule, the share of new violations that each method assigns to ads. We group rules into *naming* rules, which require an accessible name such as alternative text, and *structure* rules, which concern page structure and presentation. On naming rules, the two methods roughly agree: selector matching assigns 59.5% of new violations to ads, and blocking 59.7–67.1%. This is expected, because here the failing element is usually the ad itself, and its markup names the ad network.

On structure rules, the methods disagree. The ad-token pattern assigns 11.0% of new violations to ads, but blocking assigns 34–40%. Adding the sponsored-content pattern raises

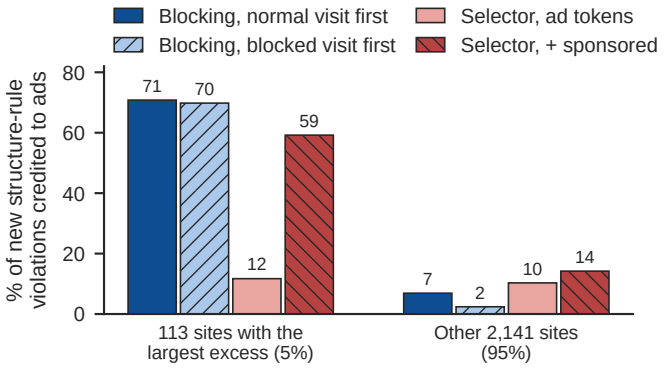


Fig. 7. RQ2. Share of new structure-rule violations that each method assigns to ads, on the 5% of sites where blocking shows the largest ad effect (left) and on the other 95% (right). Blocking finds a large effect on the few sites and almost none elsewhere; the ad-token pattern finds about 10% on both.

selector matching to 37.6%, which looks like agreement but is not (Fig. 7). Nearly all of the ads' effect on structure rules (91.7%) comes from 5% of sites, two thirds of which carry native-ad widgets. On these sites, blocking assigns about 70% of new structure violations to ads, and the ad-token pattern only 11.7%. On the other 95% of sites, blocking finds almost no ad effect, yet the ad-token pattern still assigns 10.3% to ads. Selector matching thus agrees with blocking only in aggregate: it misses native ads on a few sites and over-attributes failures to ads on the rest.

RQ2 finding: Ads more often add new violations than reduce them, by 30–33 percentage points of sites after correcting for visit order. The gap disappears on sites without listed vendors and grows with their number. Selector matching agrees with blocking on naming rules but misattributes structure failures. **Implication:** test with third-party code loaded, and attribute failures with repeated blocked runs, not with a markup pattern.

C. RQ3: What fails in frames, and who serves them?

How many sites have a failing frame? Our main audit already shows that frames matter: after the exercise, 13.7% of violating nodes lie inside frames, two thirds of them in frames that look like ads, and a checker that runs only in the top-level document misses all of them. On 818 of the 2,273 sites in the in-frame crawl (**36.0%**, 95% CI 34.0–38.0), at least one visible cross-origin frame fails a WCAG rule. Such a frame comes from a listed vendor on 21.6% of sites and from another host on 20.9%, such as a video embed, a social plugin, a consent dialog, or an ad host our list misses, like Google's `2mdn.net`. Stricter definitions of a visible frame give similar results (33.8–37.9%). So about a third of sites carry a failing frame that the publisher did not write, and not all are ads.

What fails inside them? Most in-frame failures are missing names (Fig. 8). The most common rules are `link-name` and `image-alt`: links and images with no accessible name, which a screen reader cannot describe. They are few, about 3% of the violating nodes that the main audit reports on the same visits, but the publisher cannot fix them itself.

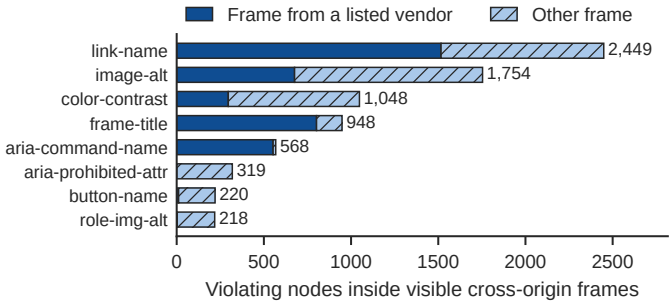


Fig. 8. RQ3. Which rules fail inside visible cross-origin frames: violating nodes for the eight most common rules, in frames from a listed vendor (dark) or from another host (light, hatched). A frame is visible if its body covers more than 2,500 px^2. Most are missing names.

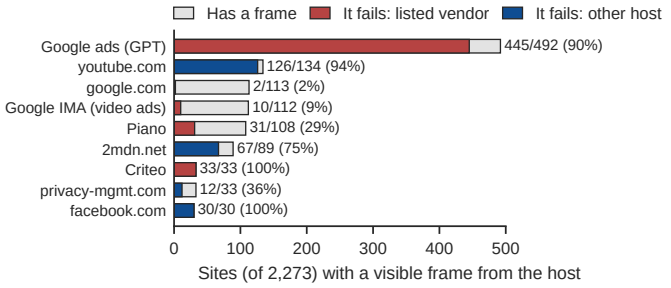


Fig. 9. RQ3. Who serves failing frames. For each host that serves visible cross-origin frames on at least 30 of 2,273 sites: the number of sites with its frame (gray) and the number where that frame fails a WCAG rule (red: a listed vendor; navy: another host). Labels give the failure rate. `youtube.com`, `google.com` and `2mdn.net` are Google hosts that are not on our list.

Who serves them? Google's ad frames hold 43.5% of in-frame violating nodes. But 84.3% of sites request Google's ad and tag hosts, so this share reflects how widely Google is used as much as how well its frames are built. Fig. 9 separates the two, and failure rates differ widely. Google's ad frames fail on 90% of the 492 sites that show them, and YouTube embeds on 94%. Piano's frames fail on 29% of sites, and Google's video-ad frames (IMA) on 9%. So whether a frame fails depends on who serves it.

RQ3 finding: On 36.0% of sites, a visible cross-origin frame fails a WCAG rule, mostly through missing names. On 60% of these sites, a listed vendor serves the frame, and failure rates differ widely between vendors. **Implication:** the publisher did not write this code and can act on it only by choosing its vendors and embeds.

V. Discussion

Testing practice. Teams that audit a page once at load, on a build without ads, miss failures that users encounter. Teams should audit again after a scripted interaction, with third-party code loaded, and repeat the run, because such failures do not always recur, much like flaky tests [30]. Dynamic crawlers [20] and Lighthouse user flows [10] already support audits after interaction. A test environment that stubs or blocks ads sees only what our blocked arm sees.

Attributing failures. Markup patterns are an unreliable way to decide who caused a failure (RQ2). Re-running the test with one component removed, a coarse form of delta debugging [31], needs no knowledge of the component's markup. Because pages vary between visits, it needs several runs in both orders.

Fixing in-frame failures. The publisher cannot edit a cross-origin frame, so the vendor must fix it. Most in-frame failures are missing names, which a vendor can often fix once, in a shared template, and a few large companies serve most ads [13], so their fixes would reach many sites. Publishers still have a lever: including third-party code means trusting its provider [32], and vendors' frames fail at very different rates (Fig. 9), so accessibility could inform which vendors a site uses.

AI repair tools. Tools that generate missing labels [33] inherit their checker's blind spots. A tool that audits at load, in the top-level document, misses the failures in RQ1 and RQ3, and may edit the publisher's markup for failures that ads cause. Repeated blocked runs could route each failure to the party that can fix it.

Conformance claims. WCAG allows a statement of partial conformance when failures come only from content "from uncontrolled sources" [1]. On ad-supported pages, this is not a corner case: on 21.6% of sites, a visible vendor frame fails a WCAG rule. No conformance claim thus covers many failures that users encounter: the publisher's claim may exclude them, and the vendors that serve them make none. Laws based on WCAG inherit this gap.

VI. Threats to Validity

Construct validity. Automated checkers cover only part of WCAG [9], and axe-core also reports best-practice rules, so we report WCAG-only results alongside. The harness models gestures but not a screen reader, and it does not trace focus order inside frames. Canonical selectors miss some re-renders, so new-violation counts include some old failures; without them, RQ1 and RQ2 shrink but keep their direction.

Internal validity. Blocking is imperfect. The block list misses unlisted and first-party ad code, removes some non-ad services, and leaks: a Google ad frame still appears on 2.8% of blocked visits, against 58.7% of normal ones. Leaks bias Δ toward zero, while the hover step, which targets ad-like elements and so runs more often with ads, may bias it upward. The arms were not counterbalanced, and they ran at different times of day: normal visits mostly at night, blocked visits mostly after noon. Ad demand varies over the day [34]. The reversed pairs and the placebo address order but not time of day, and our order correction uses identical visits that are further apart than a pair. The size of the effect is therefore specific to our crawl schedule, although its direction held in every check. Without a wait-only control, we cannot separate the gestures from the waiting. In RQ2, we picked the 5% of sites with the largest structure-rule effect using one visit order; the other order gives almost the same share (69.8% vs. 70.8%). On two identical visits, the same calculation assigns 25.0% of naming and 7.4% of structure violations to ads, well below our estimates. Our consent record lists no vendors, so vendors that validate it may have served fewer or non-personalized ads in both arms, and ad servers may treat an automated browser differently than a person [35], [36].

External validity. We crawl homepages only, from a US vantage point, in Chromium, and our inclusion rule removes 13% of normal-arm visits. Pooled rates are unweighted and hide regional differences: a new violation appears on 74.6% of US or generic sites but 45.8% of European ones, and $\Delta > 0$ on 58.6% and 34.2%. European sites also differ in popularity, consent state and distance from our vantage point, so we do not interpret this gap. About one in five sampled sites is not a news publisher.

VII. Conclusion

Accessibility checkers that run once at page load, in the top-level document, give an incomplete picture of ad-supported pages. On 2,298 sites from news corpora, a rule that did not fail at load failed after a minute of use on 40.7% of sites. Sites that gained more new violations with ads outnumbered those that gained fewer by 30–33 points, and markup patterns misattributed structure failures. On 36.0% of sites, a visible cross-origin frame had violations of its own. The remedies are not exotic: audit after interaction, test with third-party code loaded, attribute failures with blocked runs, and look inside frames. The harder question is one of responsibility: who must fix failures in code that the publisher serves but does not write.

Availability

The harness, the records of both crawls (including the round we do not analyze) and all analysis scripts are available at https://github.com/muhammad-hassnain/ad-a11y-after-load. One script, `scripts/replicate.sh`, regenerates the numbers and figures in this paper from the frozen crawl data. It takes about four minutes and checks each result against the value reported here. It also reports the crawler version behind each crawl.

Acknowledgment

Claude (Anthropic), via Claude Code [37], helped write the code and draft and polish the text. The author verified all of it and takes full responsibility. An LLM was also one of the three annotators in Section III-A.

References

[1] W3C, "Web content accessibility guidelines (WCAG) 2.2," W3C Recommendation, https://www.w3.org/TR/WCAG22/, 2023.

[2] U.S. Access Board, "Information and communication technology (ICT) standards and guidelines (revised Section 508 standards), 36 CFR part 1194," 2017.

[3] ETSI, "EN 301 549 V3.2.1: Accessibility requirements for ICT products and services," 2021.

[4] European Parliament and Council of the European Union, "Directive (EU) 2019/882 on the accessibility requirements for products and services (European Accessibility Act)," Official Journal of the European Union, L 151, 2019.

[5] A. H. Amjad, M. Danish, B. Jah, and M. A. Gulzar, "Accessibility issues in ad-driven web applications," in *Proc. IEEE/ACM 47th Int. Conf. on Software Engineering (ICSE)*, 2025, pp. 2393–2405.

[6] Z. He, S. F. Huq, and S. Malek, ""I tend to view ads almost like a pestilence": On the accessibility implications of mobile ads for blind users," in *Proc. IEEE/ACM 46th Int. Conf. on Software Engineering (ICSE)*, 2024.

[7] Deque Systems, "axe-core: Accessibility engine for automated web UI testing (version 4.13)," https://github.com/dequelabs/axe-core.

[8] WebAIM, "The WebAIM million: The 2026 report on the accessibility of the top 1,000,000 home pages," https://webaim.org/projects/million/, 2026.

[9] M. Vigo, J. Brown, and V. Conway, "Benchmarking web accessibility evaluation tools: Measuring the harm of sole reliance on automated tests," in *Proc. 10th Int. Cross-Disciplinary Conf. on Web Accessibility (W4A)*, 2013, pp. 1–10.

[10] Google, "Lighthouse user flows," https://github.com/GoogleChrome/lighthouse/blob/main/docs/user-flows.md, 2022, accessed: 2026-11-12.

[11] N. Fernandes, R. Lopes, and L. Carriço, "Evaluating web accessibility at different processing phases," *New Review of Hypermedia and Multimedia*, vol. 18, no. 3, pp. 159–181, 2012.

[12] N. Fernandes, A. S. Batista, D. Costa, C. Duarte, and L. Carriço, "Three web accessibility evaluation perspectives for RIA," in *Proc. 10th Int. Cross-Disciplinary Conf. on Web Accessibility (W4A)*, 2013, pp. 1–9.

[13] C. Yeung, T. Kohno, and F. Roesner, "Analyzing the (in)accessibility of online advertisements," in *Proc. ACM Internet Measurement Conf. (IMC)*, 2024, pp. 92–106.

[14] A. H. Amjad, B. Jah, and M. A. Gulzar, "Unintended privacy risks of using assistive technology on web applications," in *Proc. ACM Asia Conf. on Computer and Communications Security (ASIA CCS)*, 2026, pp. 1684–1696.

[15] M. Bajammal and A. Mesbah, "Semantic web accessibility testing via hierarchical visual analysis," in *Proc. IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE)*, 2021, pp. 1610–1621.

[16] P. T. Chiou, A. S. Alotaibi, and W. G. J. Halfond, "Detecting and localizing keyboard accessibility failures in web applications," in *Proc. 29th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE)*, 2021, pp. 855–867.

[17] R. Winn, P. T. Chiou, and W. G. J. Halfond, "Lost in navigation: Detecting keyboard navigation accessibility issues in web pages," in *Proc. IEEE Int. Conf. on Software Testing, Verification and Validation (ICST)*, 2026, pp. 682–693.

[18] S. Jain, S. F. Huq, Z. He, and S. Malek, "Automated detection of web application navigation barriers for screen reader users," in *Proc. 40th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*, 2025, pp. 1906–1918.

[19] C. Ge, K. Peng, C. Shi, B. Greenman, and Y. Jiang, "A11yLTLNav: Automatic detection of accessibility navigation failures," arXiv:2609.17959, 2026.

[20] A. Mesbah, A. van Deursen, and S. Lenselink, "Crawling Ajax-based web applications through dynamic analysis of user interface state changes," *ACM Transactions on the Web*, vol. 6, no. 1, pp. 1–30, 2012.

[21] F. Rosa Martins, L. Seixas Pereira, and C. Duarte, "QualState: Finding website states for accessibility evaluation," in *Proc. 21st Int. Web for All Conf. (W4A)*, 2024, pp. 96–105.

[22] T. Bostic, J. Stanley, J. Higgins, D. Chudnov, J. Brunelle, and B. Tracy, "Automated evaluation of web site accessibility using a dynamic accessibility measurement crawler," arXiv:2110.14097, 2021.

[23] N. E. Fliti, L. Kokorina, F. Tambon, and M. Papadakis, "Flow-A11y: Flow-aware accessibility testing," arXiv:2607.03100, 2026.

[24] Microsoft, "Playwright: Browser automation framework," https://playwright.dev.

[25] Google, "Chrome UX report (CrUX)," https://developer.chrome.com/docs/crux.

[26] The GDELT Project, "GDELT: Global database of events, language, and tone," https://www.gdeltproject.org.

[27] C. Reis, A. Moshchuk, and N. Oskov, "Site isolation: Process separation for web sites within the browser," in *Proc. 28th USENIX Security Symposium*, 2019, pp. 1661–1678.

[28] P. Papadopoulos, N. Kourtellis, and E. Markatos, "Cookie synchronization: Everything you always wanted to know but were afraid to ask," in *Proc. The World Wide Web Conference (WWW)*, 2019, pp. 1432–1442.

[29] IAB Europe, "Transparency and consent framework (TCF)," https://iabeurope.eu/transparency-consent-framework/, 2025.

[30] Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, "An empirical analysis of flaky tests," in *Proc. 22nd ACM SIGSOFT Int. Symp. on Foundations of Software Engineering (FSE)*, 2014, pp. 643–653.

[31] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 183–200, 2002.

[32] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You are what you include: Large-scale evaluation of remote JavaScript inclusions," in *Proc. ACM Conf. on Computer and Communications Security (CCS)*, 2012, pp. 736–747.

[33] J. Chen, C. Chen, Z. Xing, X. Xu, L. Zhu, G. Li, and J. Wang, "Unblind your apps: Predicting natural-language labels for mobile GUI components by deep learning," in *Proc. IEEE/ACM 42nd Int. Conf. on Software Engineering (ICSE)*, 2020, pp. 322–334.

[34] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: Measurement and analysis," in *Proc. 7th Int. Workshop on Data Mining for Online Advertising (ADKDD)*, 2013, pp. 1–8.

[35] S. S. Ahmad, M. D. Dar, M. F. Zaffar, N. Vallina-Rodriguez, and R. Nithyanand, "Apophanies or epiphanies? How crawlers impact our understanding of the web," in *Proc. The Web Conference (WWW)*, 2020, pp. 271–280.

[36] D. Zeber, S. Bird, C. Oliveira, W. Rudametkin, I. Segall, F. Wollsén, and M. Lopatka, "The representativeness of automated web crawls as a surrogate for human browsing," in *Proc. The Web Conference (WWW)*, 2020, pp. 167–178.

[37] Anthropic, "Claude Code," https://claude.com/product/claude-code, 2026, accessed: 2026-11-11.