AdHere: Automated Detection and Repair of Intrusive Ads

Yutian Yan
University of Southern California
Los Angeles, CA, USA
yutianya@usc.edu

Yunhui Zheng
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
zheng16.cs@gmail.com

Xinyue Liu
University at Buffalo, SUNY
Buffalo, NY, USA
xliu234@buffalo.edu

Nenad Medvidovic
University of Southern California
Los Angeles, CA, USA
neno@usc.edu

Weihang Wang
University of Southern California
Los Angeles, CA, USA
weihangw@usc.edu

***Abstract*—Today, more than 3 million websites rely on online advertising revenue. Despite the monetary incentives, ads often frustrate users by disrupting their experience, interrupting content, and slowing browsing. To improve ad experiences, leading media associations define *Better Ads Standards* for ads that are below user expectations. However, little is known about how well websites comply with these standards and whether existing approaches are sufficient for developers to quickly resolve such issues. In this paper, we propose AdHere, a technique that can detect intrusive ads that do not comply with Better Ads Standards and suggest repair proposals. AdHere works by first parsing the initial web page to a DOM tree to search for potential static ads, and then using mutation observers to monitor and detect intrusive (dynamic/static) ads on the fly. To handle ads' volatile nature, AdHere includes two detection algorithms for desktop and mobile ads to identify different ad violations during three phases of page load events. It recursively applies the detection algorithms to resolve nested layers of DOM elements inserted by ad delegations. We evaluate AdHere on Alexa Top 1 Million Websites. The results show that AdHere is effective in detecting violating ads and suggesting repair proposals. Comparing to the current available alternative, AdHere detected intrusive ads on 4,656 more mobile websites and 3,911 more desktop websites, and improved recall by 16.6% and accuracy by 4.2%.**

***Index Terms*—ad experience, advertising practice, Better Ads Standards**

I. Introduction

Online advertising has been the most critical revenue stream for over 3 million websites [1]. Websites are eager to host more ads and attract more visitors to maximize ad revenue. However, ads often fail to meet user expectations and easily alienate visitors [2]. For example, ads that disrupt the user experience, interrupt content, or slow browsing, can be frustrating and eventually damage websites' reputations and financial interests [3], [4]. While distracting ads can bring in 0.10-0.80 USD per thousand impressions, they actually cost websites 1.53 USD due to the downgraded user experience [5].

However, website visitors do not hate all ads. In fact, 83% of visitors agree, "Not all ads are bad, but I want to filter out the really obnoxious ones," and 68% can live with ads as long as they are not annoying [6]. Fundamentally, undesired third-party ads exploit the lack of access control across multiple parties involved in the online advertising system [7]. These dynamic third-party ads may not expose their behaviors until fully rendered in browsers. Thus, it's usually challenging for web developers to analyze and proactively filter out the bad ones.

Because this issue affects all stakeholders, significant efforts from both academia and industry have been made to suppress or regulate undesired ads [4], [7]–[13]. Among them, the *Better Ads Standards* [8] are the first attempt to define undesired ads at scale. To enhance ad experiences, leading media associations and corporations, e.g., Google and Interactive Advertising Bureau, develop the Better Ads Standards that define the formats of unacceptable ads. Google testers conduct ads audit manually and release the compliance results in a list named Ad Experience Report [14]. When violating ads are detected, the website owners need to repair the issue within 30 days. Otherwise, Google will block all ads on the entire domain in the Chrome browser.

The manual audit results from Google have three major drawbacks. First, the results are limited to only 6.2% among popular websites (Section III-B). Second, the results do not pinpoint the code locations of the violating ads or provide guidance on how to repair them. Instead, developers have to manually locate and repair the violations. Third, the compliance audit is manually done by Google testers who conduct ad reviews periodically. Thus, the audit results may be out of date. Given the limited coverage, lack of repair support, and delayed feedback, it's unclear how well websites comply with these standards and whether existing approaches are sufficient for developers to quickly resolve such issues.

In this paper, we investigate the pervasiveness of *websites' compliance with the Better Ads Standards*. To the best of our knowledge, this is the first study to characterize the impact of intrusive ads on real-world websites. Due to the limited tooling support, we propose an automatic detection technique that addresses the following challenges: (1) Advertisements can

TABLE I: Key findings and implications.

	Findings	Implications
1	Google only monitored 6.2% (61,727) of Alexa Top 1 Million Websites on desktop and 5.4% (54,237) on mobile, with the remaining not reviewed (Section III-B).	The low website coverage calls for measurement tools that enable detection of violating ads on any website.
2	Since the Better Ads Standards impose more strict requirements for mobile ads, there are more mobile websites containing violating ads than desktop websites (Section III-C).	This suggests that different detection algorithms should be designed for detecting violating ads on mobile vs. desktop.
3	Due to failing to fix violations on time, Google Chrome has blocked ads on 228 websites for more than three months, and 77 have been blocked for over one year (Section III-D1).	Since majority of these websites are actively maintained, such slow reactions may indicate a lack of knowledge on how to fix them.
4	Compared to the Google audit tool, AdHere detected 4,656 more mobile websites and 3,911 more desktop websites with violating ads on the top one million websites (AdHere: 10,141, Google: 1,574) (Section V-A1).	This large discrepancy is caused by the facts that (1) AdHere is not limited to selected websites and (2) AdHere has a better recall (Section V-A2).
5	AdHere achieved good recalls in identifying the top three popular types. Pop-up: 87.1%/80.0% (D/M), Large Sticky: 88.0%/82.5% (D/M), and Density Higher Than 30%: 88.5% (M) (Section V-A2).	These violations can be the result of bad coding practices or incorrect configurations, suggesting that more testing efforts should be spent on these types.
6	Violations on 97 websites have been fixed. Among them, 93 completely removed the ad container, while only four modified the ad container' attributes to comply with the standards (Section V-A3).	AdHere suggested that all violations on the 97 websites can be fixed by modifying the ad attributes and the ads do not need to be removed.

Availability	Google Ad Exp Report	AdHere
Who Initiate the Scan	Google auditors	Website developers
Website Coverage	Selective (< 7% of top 1M)	Any
Result Availability	Unknown until next scan	Immediately
Developer Assistance		
Auditing Method	Manual	Automated
Assisting Violation Identification	1) URL 2) Screenshots/Videos (optional)	1) URL 2) Screenshots 3) Lines of code of the ads
Fixing Assistance	None	Fix Suggestions
Ad-network to Avoid	None	Provided
Impact of Violations		
Violation Found	All ads on the same website will be blocked in Chrome	No impact
Developer Actions	Manually locate and fix violations; wait for next audit	Fix violations before all ads on the website got blocked

Fig. 1: Google Ad Experience Report vs. AdHere.

be fully dynamic where the structure of an ad is unknown until runtime, making it difficult to identify an ad and differentiate it from other elements. (2) Dynamically loaded ads can be highly volatile (e.g., first appear and then disappear at any time), which makes it difficult to localize dynamic ads in the source code. (3) Advertisements can be initially preloaded and later have their properties modified during runtime, thus a pure static/dynamic analysis alone does not suffice.

To this end, we propose AdHere, a technique that can automatically detect violating ads and suggest repair proposals. The design of AdHere is based on a combination of static and dynamic analyses. It works by first parsing the initial web page to a DOM tree to search for potential static ads, and then using mutation observers to monitor and detect intrusive (dynamic/static) ads on the fly. To handle ads' volatile nature, AdHere includes two detection algorithms for desktop and mobile ads to identify different ad violations during three phases of page load events. Our approach recursively applies the detection algorithms to resolve nested layers of DOM elements inserted by ad delegations. We evaluate AdHere on Alexa Top 1 Million Websites to detect their compliance with the Better Ads Standards. AdHere detected violating ads on 5,540 mobile websites and 4,601 desktop websites. Comparing to the currently available alternative, AdHere detected violations on 4,656 more mobile websites and 3,911 more desktop websites and improved recall (by 16.6%) and accuracy (by 4.2%). Since AdHere shares some similar functionalities with Google Ad Experience Report, we summarize its main advantages in Fig. 1.

In summary, this paper makes the following contributions:

- We conduct a preliminary study using the manual audit results from Google on 1 million websites to understand the landscape of violating ads on real-world websites. This study motivates a technique that can help developers of any website detect and repair violating ads. The key results are summarized in Table I (Rows 1-3).
- We propose AdHere for automatically detecting violating ads and suggesting repair proposals. We address the challenges entailed by no prior knowledge of ads' DOM structures, the volatile nature of dynamic ads, and nested layers of DOM elements inserted by multiple layers of ad delegations.
- We evaluate AdHere on Alexa Top 1 Million Websites. AdHere is effective in detecting violating ads and suggesting repair proposals. Comparing to the manual audit results from Google, AdHere detected violating ads on 4,656 more mobile websites and 3,911 more desktop websites, and improved recall by 16.6% and accuracy by 4.2%. It also has an acceptable overhead of 44%. We highlight the key results in Table I (Rows 4-6).

II. Background

A. The Online Advertising Ecosystem

Most ads rendered on websites are actually not owned or hosted by the websites. Instead, multiple parties including websites, advertisers, and website visitors are connected by multiple gigantic ad networks. They work together to deliver ads in a dynamic and targeted way. Fig. 2 shows the entities in the online advertising ecosystem and the procedure of how an ad is delivered. ❶ Websites preallocate slots for advertisements by including bootstrapping JavaScript ad libraries provided by ad networks. When a user visits a website, these snippets

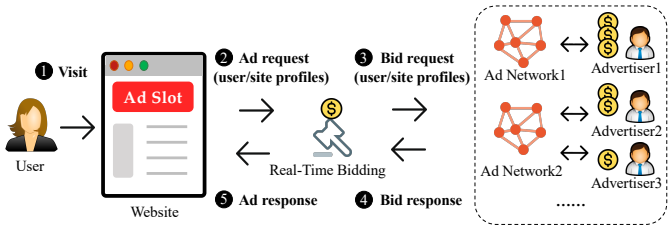


Fig. 2: Online ad ecosystem.

are executed in the user's browser. ❷ These ad libraries collect and send the user/website profiles to the real-time bidding platform. ❸ User profiles are further shared with participating ad networks and advertisers before a real-time auction is conducted. ❹ Advertisers evaluate its value and bid for a particular impression. ❺ Additional JavaScript snippets are returned to the end user and eventually load the actual ad content (e.g., videos, images or texts) from the auction winner.

B. The Better Ads Standards

To enhance user experiences with online ads, leading media associations and corporations (including Google, Microsoft, Interactive Advertising Bureau, etc.) conducted a study with 66,000 users to identify the least favorable ad experiences and developed the Better Ads Standards (as shown in Fig. 3), which define four unacceptable desktop ads and eight unacceptable mobile ads:

Pop-up Ads (D-1 and M-1) usually appear to deliberately block the main content of a web page. They may include a forced countdown that requires users to wait.

Auto-playing Video Ads with Sound (D-2 and M-2) automatically play disruptive videos and music. However, ads requiring users to activate the sound are deemed acceptable.

Large Sticky Ads (D-3 and M-3) remain constant at the bottom or the side edge. Such ads can continuously obstruct over 30% of the screen regardless of where the user is on the page.

Prestitial Ads (D-4 and M-4) appear before the main content is loaded. Users are required to wait before ads can be closed. It disrupts the navigation hence is rated as distracting ads.

Ad Density Higher than 30% (M-5) of a page are considered intrusive, where the density is measured by dividing the sum of all ad heights by the total height of the main content of the page.

Full-screen Scrollover Ads (M-6) cover the entire page and force users to scroll through the ad before viewing the main content, which can be disorienting to users.

Flashing Animated Ads (M-7) rapidly flash with alternating backgrounds, text, or colors, which are highly distracting and aggravating. Animated ads that do not flash are considered acceptable.

Postitial Ads with Countdown (M-8) appear after a link click and force a user to wait before redirecting to another page, which disrupt users' navigation flow.

III. Preliminary Compliance Study

The manual audit results from Google have three major limitations: (1) the results are limited to selective websites; (2) the results do not pinpoint violating ads location nor provide repair support; (3) the audit was performed manually and thus it cannot provide real-time feedback.

To investigate the necessity of an automated detection and repair technique, we perform a preliminary study using the Google manual audit results and analyze the audit results' website coverage, prevalence of violating ads, and common repair practice:

- **RQ1 – Website Coverage:** How many websites are manually reviewed by Google testers?
- **RQ2 – Violation Significance:** How many websites contain ads that fail to meet the Better Ads Standards?
- **RQ3 – Repair Time:** How long does it take to repair violating ads before blocking all ads on the entire domain?

Summary Results. Google's manual audit results are only available for a small number of websites. The Google audit tool reviewed fewer websites among lower-ranked websites where there are more websites failed to comply with the standards. These results motivate the design of a detection technique that can be used by developers of any website. Moreover, there were 32.5% of websites failed to react to the audit result and got punished to be blocked for more than a year. Slow response in reacting to ad filtering may indicate the audit is out of date or developers lack knowledge on how to repair them. This calls for the need of an automated detector that can provide real-time repair support.

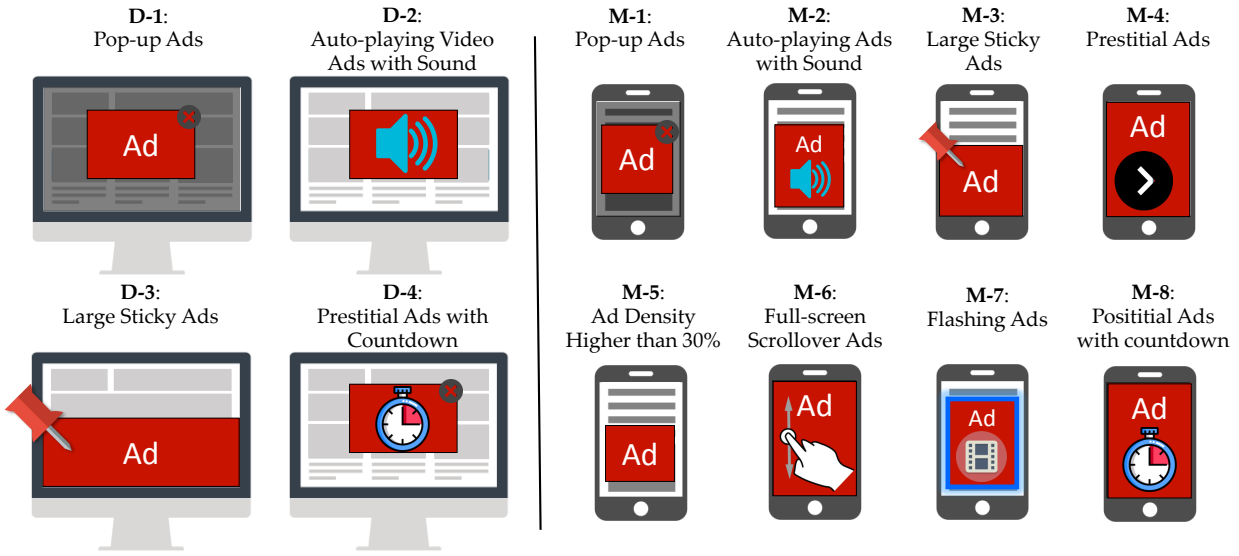
A. Methodology

By running Google Ad Experience Report, we can obtain the *ad experience ratings* of a website for both desktop web and mobile web. Fig. 4 shows the details of the ad experience ratings of an example website, *autopartspro.co.uk*. Specifically, the ad experience ratings contain the following information: (1) the Better Ads status (`PASSING`, `FAILING`, or `UNKNOWN`) (2) the timestamp the website was last reviewed; (3) the ad filtering status (`ON`/`OFF`); (4) the timestamp the website's ad filtering began, if the filter is `ON`; and (5) a link to the ad experience report.

We collect ad experience ratings for the Alexa Top 1 Million Websites every day for over four months. We focus on "the Better Ad status", "the ad filtering status", and "the timestamp the website was reviewed" to measure the website coverage and analyze the filtering enforcement time when violations are not resolved within 30 days. Note that the ad experience report does not include details about the location of the violations in source code, how to fix them, or which ad networks are involved.

B. RQ1: Website Coverage

Among Alexa Top 1 Million Websites, Google only monitored 61,727 websites (6.2%) for their desktop versions and 54,237 (5.4%) for the mobile versions. The Better Ad status



(a) Desktop Ad Experience **(b) Mobile Ad Experience**

Fig. 3: The Better Ads Standards. Four desktop violating ad types and eight mobile ad types are defined.

```
{ "reviewedSite": "autopartspro.co.uk",
  "mobileSummary": {
    "betterAdsStatus": "FAILING",
    "lastChangeTime":
        "2021-06-19T11:02:13.326467Z",
    "filterStatus": "ON",
    "enforcementTime":
        "2021-06-19T11:02:13.135624Z",
    "reportUrl":
        "https://www.google.com/webmasters/
        tools/ad-experience-mobile?siteUrl=
        autopartspro.co.uk"},
  "desktopSummary": {
    "betterAdsStatus": "PASSING",
    "lastChangeTime":
        "2021-05-26T04:32:24.552326Z",
    "filterStatus": "OFF",
    "reportUrl":
        "https://www.google.com/webmasters/
        tools/ad-experience-desktop?siteUrl=
        autopartspro.co.uk"}}
```

Fig. 4: Ad experience report for *autopartspro.co.uk*. The full report ("reportUrl") is not publicly accessible.

of the remaining sites are UNKNOWN, indicating that they are not reviewed.

Takeaway 1: The low coverage of Google's manual audit results calls for a detection technique that can be used by developers of any website for detecting violating ads on their websites.

C. RQ2: Violation Significance

Among the websites reviewed by Google, the average number of mobile websites failed daily is 884 while the desktop version is 690. There were 1,112 mobile websites and 874 desktop websites having the FAILING status at least once during the

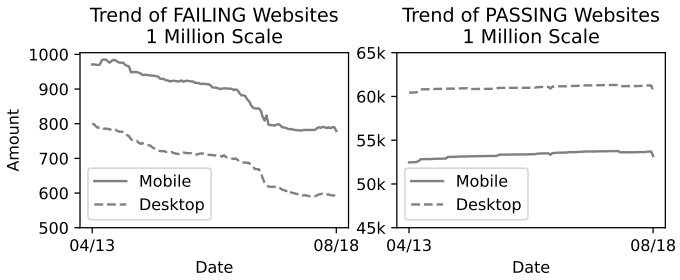


Fig. 5: FAILING and PASSING websites count. The average number of FAILING websites: 884 on mobile, 690 on desktop; PASSING websites: 53,353 on mobile, 61,025 on desktop.

study. Moreover, the average numbers of websites passed daily are 53,353 on mobile and 61,025 on desktop. Fig. 5 shows the number of sites with PASSING status and FAILING status from April 13, 2019 to August 18, 2019.

1) Rank Distribution of Websites Displayed Violating Ads: To further investigate the popularity of the websites that fail the test, we study the PASSING and FAILING websites based on their Alexa rankings and present the distribution and failing ratio in Fig. 6. The failing ratio is the ratio of the number of FAILING websites to the number of websites being monitored (i.e., the sum of FAILING and PASSING websites).

The rank distribution shows that more websites in the top 100K have been monitored and classified as FAILING than those in the other 100K segments. However, if we look at the failing ratios, the percentage of FAILING websites in websites ranked between 100K and 400K is actually higher than that in top 100K websites.

Takeaway 2: The higher failing ratio in lower-ranked websites and the fact that Google reviewed fewer websites among lower-ranked segments further motivate a detection

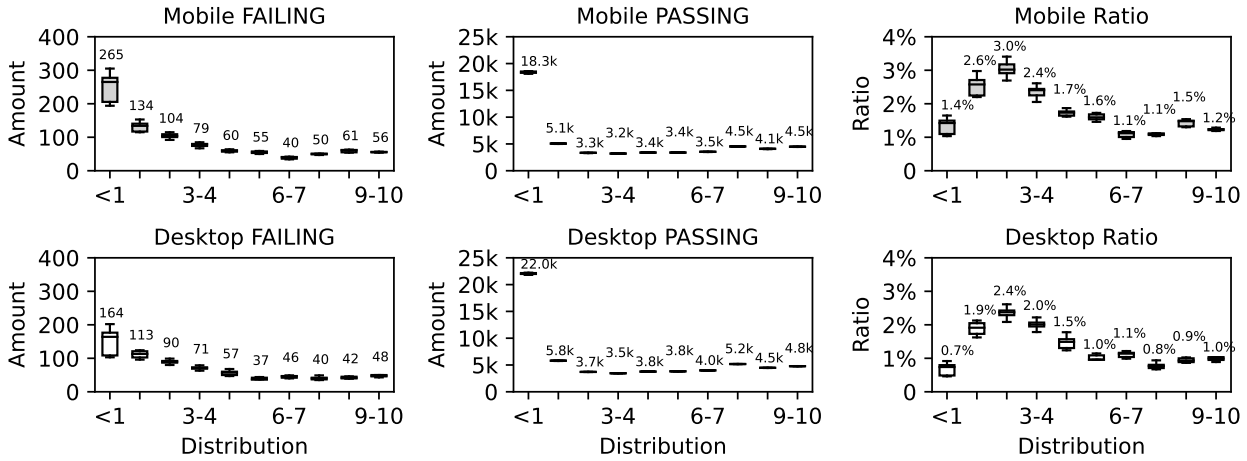


Fig. 6: Ranking distribution of FAILING and PASSING websites and the failing ratios. The x-axis is Alexa ranking 100K, 200K, ..., 1,000K. Each box and its whiskers represent the five-number summary of the website counts within each ranking range: the minimum, first quartile, median (shown above each bar), third quartile, and maximum.

approach that can be used by websites of any ranking.

Takeaway 3: In general, we observed more violations on mobile than desktop. As the Better Ads Standards impose more strict requirements for mobile platforms, different detection algorithms should be designed for mobile vs. desktop.

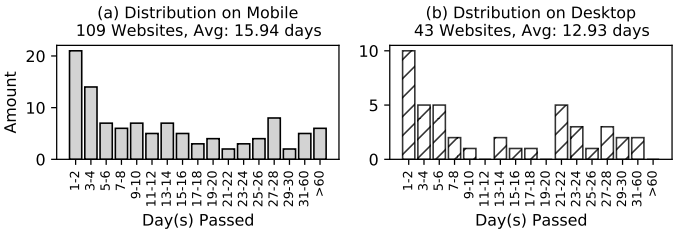


Fig. 7: Violating ads repair time. The x-axis represents the number of days to repair violations. The y-axis represents the number of websites having the same repair time.

D. RQ3: Repair Time

During the four-month study, 109 mobile websites and 43 desktop websites successfully repaired the violating ads. Among these fixes, 11 websites have been fixed for both their desktop and mobile versions. Without loss of generality, we approximate the repair time as the duration between the first time a website failed the daily test (i.e., rated as FAILING) and the first subsequent day the website passed the daily test (i.e., rated as PASSING). Fig. 7 shows the fix time on these FAILING websites. The result shows that on average, it took 15.94 days to repair violating ads on mobile and 12.9 days to repair on desktop, respectively. Note that the numbers reported in Fig. 7 exclude websites that were initially marked as FAILING since the repair time is unknown if a website failed the audit before our study.

1) Ad Filtering Enforcement Time: Once a website's Better Ad status is marked as FAILING by Google, the website owner will be asked to repair all violating ads within 30 days. Otherwise, Google Chrome will regardless filter out all ads on this website's entire domain. This is when the ad filtering enforcement time starts ticking [15].

Fig. 9 presents the result of the ad filtering enforcement time. The figure shows that 156 mobile websites and 81 desktop websites have their ads been filtered by the Chrome browser. 228 of the 237 websites (96.2%) have been filtered for more than 3 months. 77 websites (32.5%) were being filtered for over one year. On average, the ad filtering enforcement time for both mobile and desktop platforms is around 7 months. We manually inspected these 77 websites and found that 58 websites (75.3%) were still actively maintained after Google blocked their ads.

Takeaway 4: Slow response in reacting to ad filtering may indicate the audit is out of date or developers lack techniques on how to repair them, which calls for the need of an automated detector that can provide real-time repair support.

IV. Design of AdHere

In order to achieve automated detection and repair support, AdHere needs to handle several technical challenges:

- Advertisements can be either partially preloaded or fully dynamic that only initiates at runtime, thus the detection should include both static and dynamic analyses.
- Dynamically loaded ads can be highly volatile, so the detection algorithm should capture ads-related elements at different stages of page load events.
- Advertisements may include multiple layers of delegation that creates nested layers of DOM elements, making it difficult to locate the ad in the source code.

Fig. 8 presents the overview of AdHere, which includes three steps:

❶ **Collecting Website Status**. We collect the Better Ads Standards compliance status for one million websites each day. The status results will be used to identify developers' fixes (Sec. IV-A).

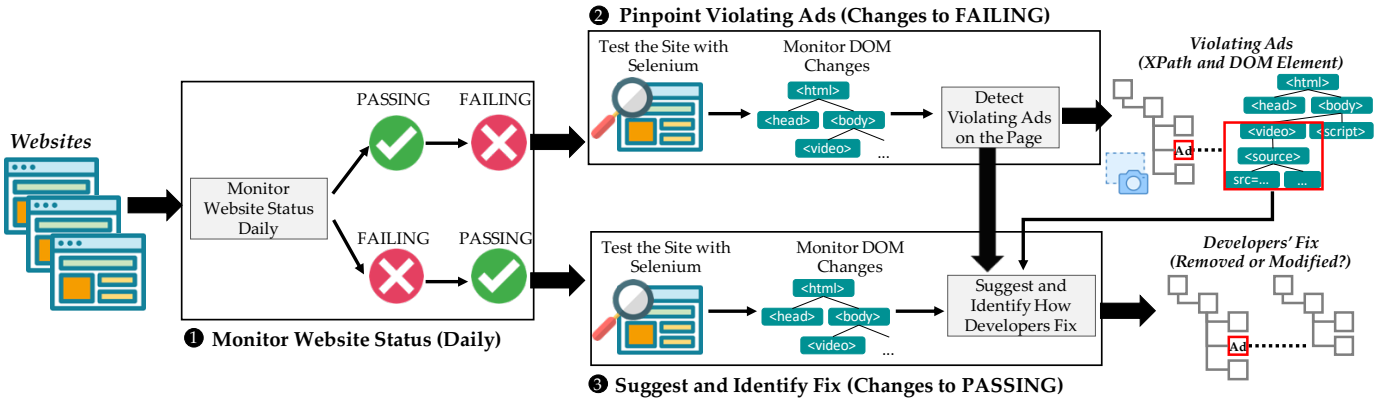


Fig. 8: Overview of AdHere.

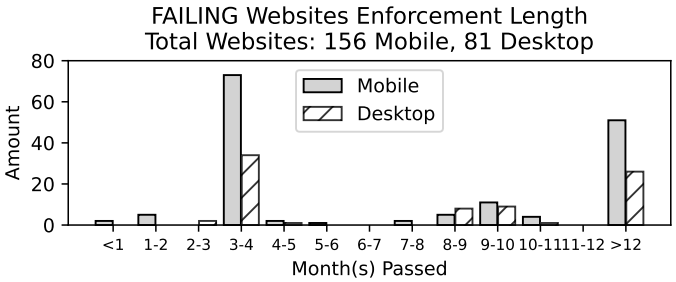


Fig. 9: Ad filtering enforcement time on websites marked FAILING longer than 30 days. The average enforcement time is 210.38/214.70 (mobile/desktop) days.

❷ **Pinpointing Violating Ads.** When the ad status changes from PASSING to FAILING, AdHere analyzes the pages and looks for violating ads. AdHere detects ads exhibiting distracting behaviors per the Standards (Sec. IV-B). This heuristic-based approach is needed to differentiate intrusive from non-intrusive ads.

❸ **Suggesting Repair Proposals.** When the ad status restores from FAILING to PASSING, we assume the violations are fixed. AdHere compares the PASSING DOM tree with and the FAILING DOM tree to learn fixes from developers. After aggregating the fixes, for newly detected violations, AdHere suggests either modifying ad attributes or removing them completely (Sec. IV-C).

We elaborate on each of the steps below.

A. Collecting Website Status

The goal of the ad status collection is to detect new violations and discover fixes that AdHere can learn from. We follow the Better Ads Standards and scan websites every day.

When new violating ads are found, AdHere tries to locate them using the algorithm presented in Sec. IV-B. If previously FAILING ads pass the compliance test, AdHere infers how the developers fixed them and records the patterns (Sec. IV-C). To compare the effectiveness of AdHere with Google Ad Experience Report, we also collected the audit results from Google and manually collected the ground truth of 1,000 websites.

B. Pinpointing Violating Ads

Automatically detecting violating ads on a website is challenging because the modern websites are highly dynamic. Some ads may only appear for a few seconds and then disappear. For example, a prestitial ad with countdown may appear before the page is loaded and close itself after a few seconds. AdHere may miss such ads if analyzing the page after all contents are fully loaded. On the other hand, if AdHere retrieves the page before fully loaded, it may get a different DOM tree and thus won't see the dynamically created structures. Thus, we propose a run-time testing approach and devise two algorithms for the desktop and mobile platforms.

1) Detecting Desktop Violating Ad: Given a desktop website, we visit the homepage and all sub pages marked as FAILING, identify violating ads, and download the subtrees of the violating ads. As violating ads may load, render and disappear in different phases, our detection algorithms are based on three critical events:

1) **DOMContentLoaded.** This event fires when the DOM tree is fully loaded. Note that when this event occurs, external resources like images, stylesheets and subframes may be not yet loaded. Hence, it is an important event to detect violations (e.g. prestitial ads) that occur prior to the main content.
2) **Load.** This event is fired when the page is fully loaded, including all dependent external resources such as stylesheets and images. Several types of violating ads can be detected after the load event is fired, such as pop-up ads and large sticky ads.
3) **Beforeunload.** The beforeunload event occurs when the user is leaving the page and the page is about to be unloaded. This event can be used to identify postitial ads that appear after a link click and postpone the redirection.

Fig. 10 outlines the procedure for desktop violating ads detection, which looks for four types of violating ads in three phases:

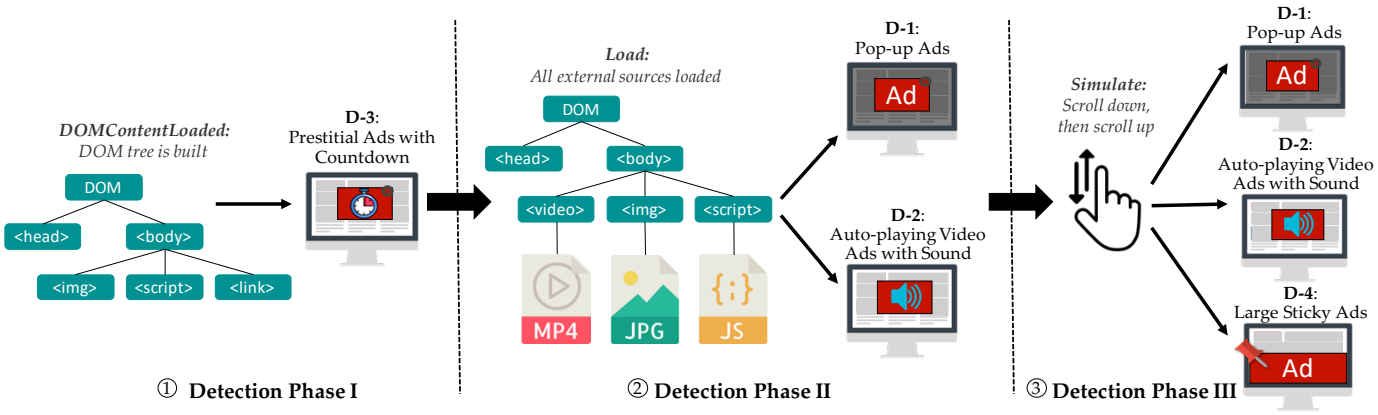


Fig. 10: Desktop violating ad detection.

- **Phase I.** We wait until the DOM tree is loaded. Once the `DOMContentLoaded` event occurs, the detection algorithm looks for the *Prestitial Ads with Countdown (D-4)*.
- **Phase II.** When the `load` event fires, the page including all of the external resources is fully loaded. On this event, we look for *Pop-up Ads (D-1)* and *Auto-playing Video Ads with Sound (D-2)*.
- **Phase III.** As some ads appear only when users scroll on the page, we programmatically scroll to the end and then scroll back to the top. By doing so, we detect *Pop-up Ads (D-1)*, *Auto-playing Video Ads with Sound (D-2)*, and *Large Sticky Ads (D-3)*.

In the following paragraphs, we will discuss details about detecting each type of violating ads.

- **D-1: Pop-up Ads.** Pop-up ads are implemented by setting CSS '`position`' attribute to '`absolute`' or '`fixed`', and assigning an unsigned integer or '`auto`' to '`z-index`'. The ad is considered violating only if the ad covers the web page.
- **D-2: Auto-playing Video Ads with Sound.** Video ads can be shown on a web page by using `<video>`, `<embed>`, or `<object>` tags. A `<video>` tag with attribute '`autoplay`' set but without setting '`muted`' attribute will make the ad disturbing. Similarly, tags like `<embed ... autoplay='true' volume='X'>` (X is a positive integer) can also introduce violating ads. Moreover, an `<object>` tag without `'play=false'` will make the ad violating.
- **D-3: Large Sticky Ads.** Large ads can be detected by measuring the actual heights of DOM elements. The ad is considered a *Large* ad if the '`height`' exceeds 30% of the actual page. Ads are *Sticky* when CSS attribute '`position`' is set to '`absolute`' or '`fixed`', to make ads won't move when scrolling.
- **D-4: Prestitial Ads with Countdown.** These ads behave like Pop-up Ads but appear before the page load. Hence, we detect them before page load in a similar way to **D-1**. We use Mutation Observer APIs [16] to detect the *countdown*.

2) Detecting Mobile Violating Ad: The mobile platform has more types of violating ads than the desktop platform. Three out of four desktop types (D-1, D-2, and D-3) exist on the mobile platform (M-1, M-2, and M-3). Note that the M-4 is simply the D-4 without a *countdown*. Additionally, the mobile platform has 4 new types: M-5, M-6, M-7, and M-8. For the types that also exist for desktop (M-1, M-2, M-3, and M-4), the detection algorithm for mobile is similar to the one for desktop. For the remaining four types (M-5, M-6, M-7, and M-8), they are handled as follows.

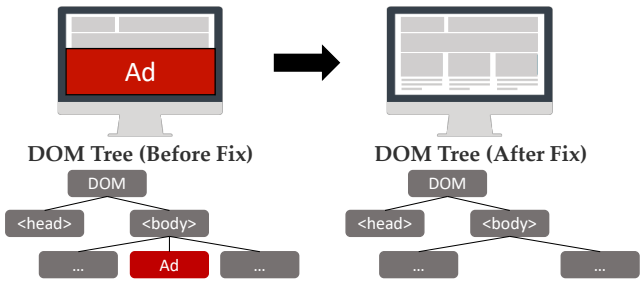
- **M-5: Ad Density Higher Than 30%** exhibits similar behaviors like *Large* ads. Checking if the total height of all ads exceeds 30% of actual page can find them.
- **M-6: Full-screen Scrollover Ads** have two features, *Full-screen* and *Scrollover*. *Full-screen* means the ad covers the entire screen, which can be detected by checking the '`height`' attribute. *Scrollover* property is measured by simulating page scroll down and then up. When a full-screen-size DOM element first exhibits *Sticky* feature and then disappears, the element is considered to be *Scrollover*.
- **M-7: Flashing Animated Ads** can be detected with the help of a Python3 library named Pillow [17]. We extract the frames from a GIF file. If it has more than one frame and loops the frames, it is considered *Animated*. For an *Animated* GIF, if each frame lasts for a very short time, we conclude this GIF is *Flashing*.
- **M-8: Postitial Ads with Countdown** appear after a link click. Before the redirection, such ads can be detected by identifying features *Countdown* and *Postitial*. Similar to D-4, we use Mutation Observers to detect the *Countdown* feature. We identify *Postitial* feature after the `beforeunload` event is fired.

Once violations are found, we use their XPaths to infer participating ad networks. In particular, we check the ads and their ancestors to extract links that containing ad network information, which can be obtained from attribute '`href`' or '`src`'. For Pop-up Ads and Large Sticky Ads that can have complex DOM structures with distinctive attributes in

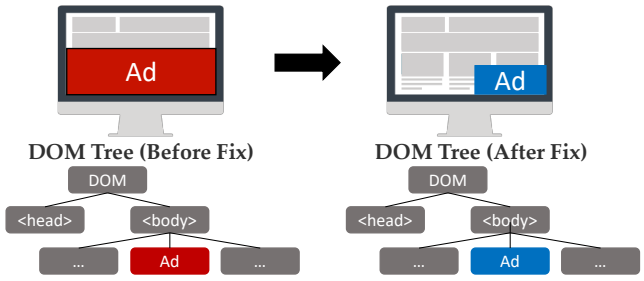
their siblings, we also extract links from the siblings and siblings' ancestors. Additionally, we capture the screenshots of the violating page to further assist developers in locating violations.

C. Suggesting Repair Proposals

1) Identifying How Developers Repair: When a previously `FAILING` website passes the compliance test, AdHere tests the website again to find out the changes developers applied. Specifically, AdHere locates the ads using the XPath obtained in the previous phase. Then, it compares the DOM structures of the `PASSING` and `FAILING` versions. As illustrated in Fig. 11, we observed developers either modified the attributes of the violating ads or simply removed them.



(a) Fix 1: Removed



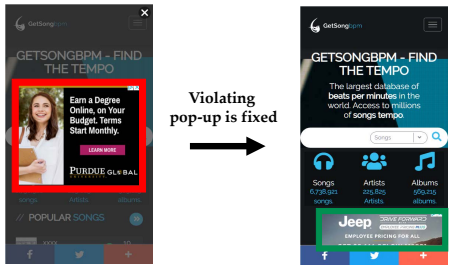
(b) Fix 2: Modified

Fig. 11: Two types of fixes: (a) violating ads are removed; (b) ad container's attributes are modified to comply with standards.

For example, Fig. 12 shows a repair found on the mobile version of *getsongbpm.com*. The screenshots in Fig. 12(a) shows that a pop-up ad on the page is replaced with an image ad. Fig. 12(b) and Fig. 12(c) are the code snippets of the pop-up ad and the image ad, respectively. At line 5695, the `z-index` of the ad container is a positive integer, indicating the content inside is pop-up content. After the fix, this pop-up ad is replaced with an image ad using the default `z-index` (line 4772). So, this image ad stays in the default layer and does not cover other contents.

By replacing the ad containers, modifying HTML document structures, and loading different scripts from the ad networks, this fix introduced 1,191 lines of code changes. The complete source code of this example can be found in [18].

2) Suggesting Repair: For violations, AdHere suggests either modifying the ad container's attributes or removing the ad container. Because removing ads completely will lower the



(a) Screenshots: the pop-up violating ad is replaced with an image ad

```
5695  <div id="aic-root-container-250" style="... z-index: 2147483647; ">...
5697    <iframe id="aic-frame-66"><html><body>...
5930      <div id="ad_unit"><div class="GoogleActiveViewElement">...
5941        <a src= "https://s0.2mdn.net/simgad/14894959027405314142" ...>...
```

(b) HTML code snippet of the pop-up ad (observed on 06/20/2019)

```
4754  <div class="leadboard addmt">...
4772  <div id="google_ads_iframe…_300x250_1_container_">...
4774    <iframe id="google_ads_iframe..._300x250_1" width="300" height="250">...
5587      <div id="google_image_div"><a href="https://www.googleadservice...">...
5594        <img src="https://tpc.googlesyndication.com/.../simgad/955...">
```

(c) HTML code snippet of the fixed image ad (observed on 07/21/2019)

Fig. 12: Fix example on *getsongbpm.com*.

ad income, it's the fallback option and thus is omitted. Since developers commonly convert websites from one platform to another, AdHere defines fix suggestion rules that work for both desktop and mobile. Our fix suggestion rules are derived from the Better Ads Standards, and we list them in Table II. Although we collect real-world fix attempts, the repair suggestion rules are not derived from previous fixes. The goal of collecting existing fixes is to determine if a repair proposal corresponds to the actual repair that makes the website compliant. Moreover, AdHere provides the repair suggestions for developers but does not repair ad violations automatically.

TABLE II: Fix suggestion rules.

Ads Type	Fix Suggestion
Pop-up Ads	• (D/M) Set 'position' attribute to 'relative' or 'static'. • (D/M) Set 'z-index' attribute to 0 or lower. • (D/M) Remove the countdown that forces users to stay on the page.
Prestitial Ads	• (M) Set the 'z-index' attribute to 0 or lower to avoid blocking users from viewing the main content. • (D/M) Remove the countdown so users can immediately close the ad.
Ad Density Higher Than 30%	• (M) Increase the height of the main content portion of the web page to reduce the density. • (M) Remove parts of the ads to reduce the total height of these ads to less than 30% of the main content.
Flashing Animated Ads	• (M) Modify ads to avoid rapidly changing background, highly aggravating texts, or colors.
Auto-playing Video Ads with Sound	• (D/M) Add 'muted' attribute, or remove 'autoplay' attribute.
Postitial Ads with Countdown	• (M) Remove the countdown.
Full-screen Scrollover Ad	• (M) Set 'z-index' attribute to 0 or lower. • (M) Set 'position' attribute to 'relative' or 'static'.
Large Sticky Ads	• (D/M) Set 'position' attribute to 'relative' or 'static'. • (D/M) Shrink the ad size until the ad is less than 30% of the screen's real estate.

• "**D**"/"**M**": the suggestion is for **D**esktop and **M**obile platform respectively

V. Evaluation

AdHere is built atop Selenium [19] and ChromeDriver [20]. For each website, we visited the desktop version using Headless Chrome [21] with all ad blocking features disabled, and used ChromeDriver to visit the mobile version by simulating a Samsung Galaxy S5 phone. In case a legitimate page behaves

similarly as one with violating ads, we used EasyList [22], a blacklist maintained and used by a set of popular ad blockers, to double check if it really contains the ads and eliminated false positives. The experiments were done on a computer with Intel i7 CPU and 16 GB memory.

A. Effectiveness

1) Prevalence: We run ADHERE on Alexa Top 1 Million Websites to detect the prevalence of violating ads. The result shows that ADHERE detected violating ads on 5,540 mobile websites and 4,601 desktop websites. Compared to Google Ad Experience Report, ADHERE detected 4,656 more mobile websites and 3,911 more desktop websites containing violating ads. Moreover, violating ads on mobile websites were more frequently observed. This result is consistent with the preliminary study.

TABLE III: ADHERE per-ad-type results on 10,533 ads.

Ad Type	Mobile	Desktop
Pop-up Ads	740 (12.6%)	1,685 (36.2%)
Auto-playing Video Ads with Sound	20 (0.3%)	23 (0.5%)
Prestitial Ads (w/ Countdown for Desktop)	82 (1.4%)	63 (1.4%)
Large Sticky Ads	1,647 (28.0%)	2,818 (61.9%)
Postitial Ads w/ Countdown	0 (0%)	N/A
Ad Density Higher Than 30%	2,358 (40.1%)	N/A
Full-screen Scrollover Ads	34 (0.6%)	N/A
Flashing Animated Ads	1,003 (17.0%)	N/A
Total*	5,884	4,649

*: Some websites contain multiple types of ads, so the number of total websites is larger than websites containing violating ads in Section V-A1.

Table III shows the number of websites that contain ADHERE-found violating ads per the Better Ads Standards. ADHERE considers the existence of a violation type if ads of that kind appeared on a website at least once. As the table shows, Large Sticky Ads, Ad Density Higher Than 30%, and Flashing Animated Ads are the most prevalent violating types. ADHERE detected a few instances of Auto-playing Video Ads with Sound, Prestitial Ads, and Full-screen Scrollover Ads. We did not observe the Postitial Ads with Countdown.

2) Comparisons with the Google Ad Experience Report: Similar to other defect detection works, due to the lack of the oracle, it's infeasible to automatically collect the ground truth for all 1 million websites used in the preliminary study (Section III). Therefore, we follow the Random Under-Sampling (RUS) approach [23] and selected 1,000 websites from the ones audited by Google from July 5th, 2021 to July 14th, 2021. We constructed a balanced set of samples according to Google's audit outcomes: 250 `FAILING` desktop websites, 250 `PASSING` desktop websites, 250 `FAILING` mobile websites, and 250 `PASSING` mobile websites. We manually analyzed them, obtained the ground truth, and then computed performance metrics [24] to compare ADHERE and the Google Ad Experience Report.

For each website, we run two rounds of independent inspections: (1) We open a fresh Chrome in incognito mode to avoid side effects. (2) We visit the homepage and look for Prestial Ads before the page is loaded. (3) When fully loaded, we verify if violating ads exist. (4) We scroll to the bottom and back to the top to check violations that may appear while scrolling (e.g., Large Sticky Ads). Additionally, we inspect the CSS to verify compliance with numerical constraints (e.g., ads covering 30% of the screen). (5) We click 10 links (if any) on the homepage from left to right, top to bottom. Then, we repeat (2)-(4) for these pages. A website is considered `FAILING` if violations were found in both rounds.

Based on the collected ground truth, we compute the confusion matrix and the precision, recall, accuracy, and F1-score metrics of the two detectors, ADHERE and the Google audit tool. Specifically, positive means that a website contains at least one intrusive ad; negative represents a website with no intrusive ads. True positive (TP) is a test result where the detector correctly detects the presence of violating ads; true negative (TN) is a test result that the detector correctly indicates the absence of violating ads; false positive (FP) means that the detector wrongly identifies that violating ads are present; false negative (FN) means that the detector wrongly indicates that violating ads are absent. Precision, or positive predictive value (PPV), is the number of true positives (correctly identified websites containing intrusive ads) divided by the number of all raised alarms (correct or not); negative predictive value (NPV) is the number of true negatives divided by the number of websites that no violating ads were found. Recall, or true positive rate (TPR), is the number of true positives divided by the number of all websites with violations in the dataset; true negative rate (TNR) is the number of true negatives divided by the number of all websites with no violations. Accuracy is the number of true predictions (i.e., the sum of true positives and true negatives) divided by the number of all websites in the dataset. F1-score is the harmonic mean of precision and recall. That is, $Prec=\frac{TP}{TP+FP}$, $NPV=\frac{TN}{TN+FN}$, $Rec=\frac{TP}{TP+FN}$, $TNR=\frac{TN}{TN+FP}$, $Acc=\frac{TP+TN}{TP+TN+FP+FN}$, $F1=\frac{2*Prec*Rec}{Prec+Rec}$.

TABLE IV: Confusion matrices.

Google		Predicted		ADHERE		Predicted	
		Negative	Positive			Negative	Positive
Actual	Negative	344	130	Actual	Negative	299	175
	Positive	156	370		Positive	69	457
Precision		74.0%		Precision		72.3%	
NPV		68.8%		NPV		81.3%	
Recall		70.3%		Recall		86.9%	
TNR		72.6%		TNR		63.1%	
Accuracy		71.4%		Accuracy		75.6%	
F1-score		72.1%		F1-score		78.9%	

Table IV shows the confusion matrices and the performance metrics result. ADHERE achieves 72.3% precision, 81.3% NPV, 86.9% recall, 63.1% TNR, 75.6% accuracy, and 78.9% F1-score, while the Google audit tool achieves 74.0% precision, 68.8% NPV, 70.3% recall, 72.6% TNR, 71.4% accuracy, and 72.1% F1-score. Our inspection of the 1,000 websites finds that ADHERE has a better recall and NPV, outperforming the Google audit tool by 16.6% and 12.5% respectively, i.e., AdHere missed fewer websites with intrusive ads. While AdHere has slightly lower precision and a lower true negative rate, the recall and NPV are more relevant as these rates measure how often our technique detects intrusive ads. ADHERE also achieves higher accuracy and F1-score compared to Google's tool. These

metrics show that ADHERE can more accurately identify more violations and help developers prevent Google's site-wide ad blocking.

We are also interested in comparing per-ad-type results. However, such information is not publicly accessible in the Google audit results and is only visible to website owners. We were not able to present a head-to-head breakdown comparison. Instead, we only discuss the per-type performance of ADHERE. Different from the confusion matrices discussed above, if a type of violation occurred on the web page during the manual inspection process is the same type of violation detected by ADHERE, we count it as a true positive for this ad type and this website. If the violation type doesn't match, we say it's a false positive.

TABLE V: ADHERE per-ad-type results on 1,000 websites.

Ad Type	TP	FP	TN	FN	Precision	Recall	Accuracy	F1-score
D-1: Pop	101	45	0	15	69.2%	87.1%	62.7%	77.1%
D-2: Auto	7	0	0	1	100%	87.5%	87.5%	93.3%
D-3: Sticky	146	61	0	20	70.5%	88.0%	64.3%	78.3%
D-4: Prestitial	0	0	0	5	N/A	0%	0%	N/A
M-1: Pop	80	34	0	20	70.2%	80.0%	59.7%	74.8%
M-2: Auto	5	0	0	1	100%	83.3%	83.3%	90.9%
M-3: Sticky	104	61	0	22	63.0%	82.5%	55.6%	71.4%
M-4: Prestitial	1	0	0	4	100%	20.0%	20.0%	33.3%
M-5: 30%	115	42	0	15	73.2%	88.5%	66.9%	80.1%
M-6: Full-screen	2	0	0	3	100%	40.0%	40.0%	57.1%
M-7: Flashing	31	0	0	22	100%	58.5%	58.5%	73.8%
M-8: Posititial	0	0	0	2	N/A	0%	0%	N/A

According to the per-type result shown in Table V, ADHERE performed well at identifying the top three common ad types (according to Table III). The recall for Pop-up Ads is 87.1% (desktop) / 80.0% (mobile). The recall of detecting Large Sticky Ads is 88.0% (desktop) / 82.5% (mobile). For Ads Density Higher Than 30%, which are treated as violations on the mobile platform, the recall is 88.5%.

ADHERE missed the detection of some ads, e.g., the Flashing Animated Ads. As shown in Table V, our approach detected 31 websites that truly included Flashing Animated Ads but missed 22 websites with such ads. Flashing Animated Ads animate and flash with quickly changing backgrounds, text, or colors. ADHERE identifies such ads by looking for GIFs with rapid frame alternation (<0.25s). However, ADHERE missed animated ads that were implemented using JavaScript, and we plan to support JavaScript animations in the future.

To ensure that the dataset on which recall was manually computed is available for replication by other researchers, we provide the raw data and the confusion matrices for each label and each ad type in our dataset [18].

3) Repair Proposal Result: Among the websites that have been fixed (109 on mobile and 43 on desktop) during the study, we observed that 55 sites were either not accessible or did not show violating ads (36 for mobile and 19 for desktop). ADHERE removed these sites and analyzed the fix approaches for the remaining sites. In total, only 4 websites (2 on mobile and 2 on desktop) were fixed by changing the ad containers' attributes to comply with the Standards, while the other 93 (71 on mobile and 22 on desktop) were fixed by completely removing the ad containers. The 2 mobile websites being fixed by modifying ad attributes were *getsongbpm.com* and *arabpornsex.com*. The 2 desktop websites were *portalpopline.com.br* and *simply-hentai.com*.

All 97 fixes are consistent with the fix suggested by ADHERE. Specifically, 5 websites fixed Prestitial Ads by removing them. Of the 14 websites having Pop-up Ads, 11 websites removed the ads, and 3 websites modified the '`position`' and '`z-index`' attributes which is consistent with the fix solutions recommended by ADHERE. 43 websites fixed Ad Density Higher Than 30% by removing it. Finally, 30 websites removed Large Sticky Ads, and 1 website fixed such ads by reducing the ad size. According to fixes recommended by ADHERE, all 97 ads can be fixed by modifying the ad attributes and they do not need to be removed.

B. Efficiency

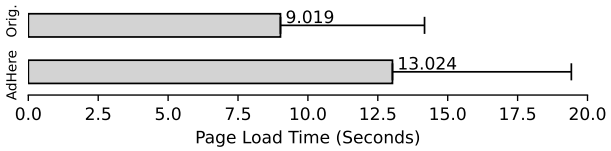


Fig. 13: Web page load time before (avg: 9.019, st: 5.152) and after (avg: 13.024, st: 6.398) ADHERE is applied.

In order to explore the overhead of ADHERE, we measure the page load time with and without ADHERE on 1,000 websites using the same list mentioned in Sec. V-A2. Fig. 13 shows the average page load time before and after ADHERE is applied. As shown in the figure, when ADHERE is applied, the browser takes longer to load the web page because (1) ADHERE introduces additional JavaScript and the browser needs extra time to parse the code; (2) ADHERE detects violations before the page is fully loaded (see IV-B) so the page loading process is slowing down. On average, the overhead of loading a web page is 44%, which is acceptable.

VI. Limitations and Future Work

While the evaluation of ADHERE on the top one million websites shows that it is effective, it has three potential limitations.

First, ADHERE is subject to web page crawling effectiveness. To identify where violating ads are and the types of violations, we built web crawlers to visit each website's homepage and all the sub-pages. After a page is fully loaded, the crawler simulates page scroll to detect certain types of ads that can only appear during/after scroll actions. ADHERE may miss other violating ads that only can be triggered under certain user navigation patterns or with particular idle time. In future work, we will continue to enhance our technique and improve its performance, especially for ad types ADHERE doesn't perform well. For example, we can sample intrusive ads and use machine learning techniques to train an identification model to increase accuracy. Moreover, ADHERE could be further improved if being incorporated with more effective web crawling techniques like [25]–[29].

Second, due to the dynamic nature inherent to the web, the set of violating ads identified by ADHERE may be different from those observed by Google. Since the full report from Google is not available, we are not able to verify the details of their dataset. To reduce the impact of randomness in ad serving, we performed the comparison experiment during the same 10-day period. Because an ad's halflife, or the time an ad loses 50% of its initial impact, spans between one and four weeks [30], it is highly likely that the majority of the ads observed by Google and ADHERE throughout the 10-day period are the same.

Third, ADHERE cannot recognize the fix practice that a violating ad is modified and moved to another location on the same page or other pages. In such cases, it is very challenging to determine whether two ads are identical. To identify such fix, ADHERE may be improved by leveraging machine learning techniques to model ads features and recognize similarities among ads. We leave this for future work.

VII. Related Work

Online Ad Experiences. Much research effort has been made to study the impact of ads on website performance and quality of web experience [4], [10], [31]–[38]. However, none of the existing work focuses on compliance with the Better Ads Standards. For example, [31], [32] conducted studies that measure impact of ads and other contributing factors on the performance of web pages. A study [4] of 21 Android apps concluded that in-app advertisements depleted significantly more computational resources, network bandwidth, and energy. [39] investigated the effectiveness of various video ad-choice formats. [40] studied ads that overpass ethical limits, [37] researched on how mobile apps violated the behavioral policy of advertisement libraries, and [36] measured smart TV advertising and tracking. [10] proposed a tool that allows web developers to specify resource constraints on third-party ad events to enhance user experience. AdGraph [38] presented a novel graph-based machine learning approach for detecting advertising and tracking resources on the web.

Ad Blockers. Ad blockers (such as AdBlock Plus, uBlock Origin, and AdGuard [41]–[43]) are tools designed to conceal advertisements on a web page from the user. While one perspective maintains that advertisements are intrusive [44] and users should possess complete autonomy over what ad is considered acceptable and privacy-preserving [42], [45], others believe that online advertising is a necessary revenue stream for free websites. Several methods designed to circumvent ad blockers or whitelist acceptable ads have been proposed [13], [46]–[48]. For example, WebRanz [13] used randomization that continuously mutates HTML elements and their attributes, rendering ad blockers unable to recognize ad patterns. On the other hand, anti-circumvention studies and methods were investigated to strengthen ad blockers [49]–[53]. ShadowBlock [49] is a Chromium-based adblocking browser that can hide traces of adblocking activities from anti-adblockers. CV-Inspector [50] proposed a machine learning-based approach to automatically detect ad blocker circumvention using differential analyses.

Privacy and Security Implications. Our work is also related to privacy and security implications of online advertising [12], [54]–[60]. FraudDroid [60] detected 335 ad frauds in mobile Android apps that were confirmed to be true positive results. MadDroid [59] performed a large-scale study on Android apps and found that 6% of apps deliver devious ad contents. MadTracer [61] detected malvertising through dynamic rule generation. [62] conducted a study of the origins of malvertising and showed that websites without explicit contracts with advertisers are prone to delivering malicious ads. The authors of [56] developed a classifier-based framework for identifying malicious advertisements. AdJail [63] proposed a framework that enables web developers to mitigate potentially harmful ad practices that jeopardize user confidentiality and integrity. AdSentry [64] protected users from untrusted ads by isolating their exposure to the main page via customizable access control policies.

VIII. Conclusion

We propose ADHERE, an automated technique to detect intrusive ads and suggest repair proposals. ADHERE addressed the challenges entailed by no prior knowledge of ads' DOM structures, the volatile nature of dynamic ads, and nested layers of DOM elements inserted by ad delegations. Our evaluation on Alexa Top 1 Million Websites shows that ADHERE is effective in identifying intrusive ads and suggesting repair proposals. Comparing to the current available alternative, ADHERE detected violations on more websites and improved recall and accuracy.

IX. Data Availability

We make our data publicly available at osf.io [18], including the collected dataset, ADHERE, scripts developed for the preliminary study, and setup instructions for the programs.

X. Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This research was partially supported by the U.S. National Science Foundation under grant no. CCF-2047980 and CCF-2106871. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

[1] W3Techs, "Usage of advertising networks for websites," aug 2019, https://w3techs.com/technologies/overview/advertising/all.

[2] M. An, "Why people block ads (and what it means for marketers and advertisers)," *Hubspot Research*, 2016.

[3] J. Marshall, "Growth of ad blocking adds to publishers' worries," aug 2015, https://blogs.wsj.com/cmo/2015/04/09/growth-of-ad-blocking-adds-to-publishers-worries/.

[4] J. Gui, S. Mcilroy, M. Nagappan, and W. G. Halfond, "Truth in advertising: The hidden cost of mobile ads for software developers," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015, pp. 100–110.

[5] D. G. Goldstein, S. Suri, R. P. McAfee, M. Ekstrand-Abueg, and F. Diaz, "The economic and cognitive costs of annoying display advertisements," *Journal of Marketing Research*, vol. 51, no. 6, pp. 742–752, 2014.

[6] M. Willner, "New data on why people hate ads: Too many, too intrusive, too creepy," sep 2016, https://www.vieodesign.com/blog/new-data-why-people-hate-ads.

[7] I. L. Kim, Y. Zheng, H. Park, W. Wang, W. You, Y. Aafer, and X. Zhang, "Finding client-side business flow tampering vulnerabilities," in *Proceedings of the 42nd International Conference on Software Engineering*, 2020.

[8] C. for Better Ads, "Better ads standards: Least preferred ad experiences for desktop web and mobile web," aug 2019, https://www.betterads.org/standards/.

[9] S. Lee and S. Ryu, "Adlib: Analyzer for mobile ad platform libraries," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 262–272. [Online]. Available: https://doi.org/10.1145/3293882.3330562

[10] W. Wang, I. L. Kim, and Y. Zheng, "Adjust: runtime mitigation of resource abusing third-party online ads," in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019, pp. 1005–1015.

[11] I. L. Kim, W. Wang, Y. Kwon, Y. Zheng, Y. Aafer, W. Meng, and X. Zhang, "Adbudgetkiller: Online advertising budget draining attack," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 297–307. [Online]. Available: \url{https://doi.org/10.1145/3178876.3186096}

[12] W. Wang, Y. Kwon, Y. Zheng, Y. Aafer, I. Kim, W.-C. Lee, Y. Liu, W. Meng, X. Zhang, P. Eugster *et al.*, "Pad: Programming third-party web advertisement censorship," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 240–251.

[13] W. Wang, Y. Zheng, X. Xing, Y. Kwon, X. Zhang, and P. Eugster, "Webranz: web page randomization for better advertisement delivery and web-bot prevention," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 205–216.

[14] Google_Ad_Experience_Report_API, "Rest resource: sites," 2019, https://developers.google.com/ad-experience-report/v1/reference/rest/v1/sites.

[15] Google, "Chrome ad filtering - web tools help: 2020," 2020, https://support.google.com/webtools/answer/7308033?hl=en&ref_topic=7566613.

[16] Mozilla and individual contributors., "Web apis — mdn," 2021, https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver/observe.

[17] A. Clark and Contributors, "Python imaging library," aug 2019. [Online]. Available: https://pillow.readthedocs.io/en/stable/

[18] AdHere, "Adhere data," 2022, https://osf.io/s8mhw/?view_only=42a1f52903964e68836faa76f84a180f.

[19] Software_Freedom_Conservancy, "Selenium," 2019, https://www.seleniumhq.org/.

[20] Google, "Chromedriver – webdriver for chrome," 2019, https://chromedriver.chromium.org/.

[21] ——, "Headless chromium," 2019, https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md.

[22] EasyList, "Easylist," aug 2019, https://easylist.to/.

[23] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[24] Wiki, "Precision and recall," 2021, https://en.wikipedia.org/wiki/Precision_and_recall.

[25] A. Mesbah, E. Bozdag, and A. Van Deursen, "Crawling ajax by inferring user interface state changes," in *2008 Eighth International Conference on Web Engineering*. IEEE, 2008, pp. 122–134.

[26] A. Mesbah and A. Van Deursen, "Invariant-based automatic testing of ajax user interfaces," in *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 2009, pp. 210–220.

[27] S. Artzi, J. Dolby, S. H. Jensen, A. Møller, and F. Tip, "A framework for automated testing of javascript web applications," in *Proceedings of the 33rd International Conference on Software Engineering*, 2011, pp. 571–580.

[28] S. R. Choudhary, M. R. Prasad, and A. Orso, "X-pert: Accurate identification of cross-browser issues in web applications," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 702–711.

[29] S. Mahajan, A. Alameer, P. McMinn, and W. G. Halfond, "Xfix: an automated tool for the repair of layout cross browser issues," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2017, pp. 368–371.

[30] S. Kerho, "How long does your ad have an impact? - fast company," 2021, https://www.fastcompany.com/16650084/how-long-does-your-ad-have-impact.

[31] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: measurements, metrics, and implications," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 313–328.

[32] M. Varela, L. Skorin-Kapov, T. Mäki, and T. Hoßfeld, "Qoe in the web: A dance of design and performance," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2015, pp. 1–7.

[33] J. Gui, M. Nagappan, and W. G. Halfond, "What aspects of mobile ads do users care about? an empirical study of mobile in-app ad reviews," *arXiv preprint arXiv:1702.07681*, 2017.

[34] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the quality of experience of web users," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 4, pp. 8–13, 2016.

[35] D. G. Goldstein, R. P. McAfee, and S. Suri, "The effects of exposure time on memory of display advertisements," in *Proceedings of the 12th ACM conference on Electronic commerce*. ACM, 2011, pp. 49–58.

[36] J. Varmarken, H. Le, A. Shuba, A. Markopoulou, and Z. Shafiq, "The tv is smart and full of trackers: Measuring smart tv advertising and tracking," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 129–154, 2020.

[37] F. Dong, H. Wang, L. Li, Y. Guo, G. Xu, and S. Zhang, "How do mobile apps violate the behavioral policy of advertisement libraries?" in *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, 2018, pp. 75–80.

[38] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, "Adgraph: A graph-based approach to ad and tracker blocking," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 763–776.

[39] S. Bellman, R. F. Potter, J. A. Robinson, and D. Varan, "The effectiveness of various video ad-choice formats," *Journal of Marketing Communications*, pp. 1–20, 2020.

[40] D. Belanche, "Ethical limits to the intrusiveness of online advertising formats: A critical review of better ads standards," *Journal of marketing communications*, vol. 25, no. 7, pp. 685–701, 2019.

[41] A. Plus, "Adblock plus," https://adblockplus.org/, aug 2019.

[42] R. Hill, "ublock origin," aug 2019, https://github.com/gorhill/uBlock.

[43] AdGuard, "Adguard: The world's most advanced ad blocker!" aug 2019, https://adguard.com/.

[44] E. Pujol, O. Hohlfeld, and A. Feldmann, "Annoyed users: Ads and ad-block usage in the wild," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 93–106.

[45] K. Garimella, O. Kostakis, and M. Mathioudakis, "Ad-blocking: A study on performance, privacy and counter-measures," in *Proceedings of the 2017 ACM on Web Science Conference*. ACM, 2017, pp. 259–262.

[46] Antiblock, "Anti adblock script," aug 2019, https://antiblock.org/.

[47] Acceptable_Ads, "Acceptable ads standards," 2019, https://acceptableads.com/en/.

[48] R. J. Walls, E. D. Kilmer, N. Lageman, and P. D. McDaniel, "Measuring the impact and perception of acceptable advertisements," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 107–120.

[49] S. Zhu, U. Iqbal, Z. Wang, Z. Qian, Z. Shafiq, and W. Chen, "Shadowblock: A lightweight and stealthy adblocking browser," in *The World Wide Web Conference*, 2019, pp. 2483–2493.

[50] L. Hieu, M. Athina, and S. Zubair, "Cv-inspector: Towards automating detection of adblock circumvention," in *Network and Distributed System Security Symposium (NDSS)*, 2021.

[51] S. Zhu, X. Hu, Z. Qian, Z. Shafiq, and H. Yin, "Measuring and disrupting anti-adblockers using differential execution analysis," in *The Network and Distributed System Security Symposium (NDSS)*, 2018.

[52] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: retrospective measurement and analysis of anti-adblock filter lists," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 171–183.

[53] S. Zhu, Z. Wang, X. Chen, S. Li, K. Man, U. Iqbal, Z. Qian, K. S. Chan, S. V. Krishnamurthy, Z. Shafiq *et al.*, "Eluding ml-based adblockers with actionable adversarial examples," in *Annual Computer Security Applications Conference*, 2021, pp. 541–553.

[54] S. Hussein, P. Meredith, and G. Roşlu, "Security-policy monitoring and enforcement with javamop," in *Proceedings of the 7th Workshop on Programming Languages and Analysis for Security*. ACM, 2012, p. 3.

[55] D. Samarasinghe, "Malvertising," aug 2019, https://www.cisecurity.org/blog/malvertising/.

[56] P. Poornachandran, N. Balagopal, S. Pal, A. Nair, P. a U, and M. R. Krishnan, *Demalvertising: A Kernel Approach for Detecting Malwares in Advertising Networks*, 11 2017, pp. 215–224.

[57] G. Chen, W. Meng, and J. Copeland, "Revisiting mobile advertising threats with madlife," in *The World Wide Web Conference*, 2019, pp. 207–217.

[58] S. Son, D. Kim, and V. Shmatikov, "What mobile ads know about mobile users." in *Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2016.

[59] T. Liu, H. Wang, L. Li, X. Luo, F. Dong, Y. Guo, L. Wang, T. Bissyandé, and J. Klein, "Maddroid: Characterizing and detecting devious ad contents for android apps," in *Proceedings of The Web Conference 2020*, 2020, pp. 1715–1726.

[60] F. Dong, H. Wang, L. Li, Y. Guo, T. F. Bissyandé, T. Liu, G. Xu, and J. Klein, "Frauddroid: Automated ad fraud detection for android apps," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 257–268.

[61] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: understanding and detecting malicious web advertising," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 674–686.

[62] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, "The dark alleys of madison avenue: Understanding malicious advertisements," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 373–380.

[63] M. Ter Louw, K. T. Ganesh, and V. Venkatakrishnan, "Adjail: Practical enforcement of confidentiality and integrity policies on web advertisements." in *USENIX Security Symposium*, 2010, pp. 371–388.

[64] X. Dong, M. Tran, Z. Liang, and X. Jiang, "Adsentry: comprehensive and flexible confinement of javascript-based advertisements," in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011, pp. 297–306.