Received: 28 May 2025 | Revised: 14 October 2025 | Accepted: 3 November 2025

DOI: 10.1002/asi.70039

RESEARCH ARTICLE

JASIST WILEY

Problems with archiving and replaying current web advertisements

Travis Reid[1] | **Alex H. Poole**[2] | **Hyung Wook Choi**[2] | **Christopher Rauch**[2] | **Mat Kelly**[2] | **Michael L. Nelson**[1] | **Michele C. Weigle**[1]

[1]Department of Computer Science, Old Dominion University, Norfolk, Virginia, USA

[2]Department of Information Science, Drexel University, Philadelphia, Pennsylvania, USA

Correspondence
Travis Reid, Department of Computer Science, Old Dominion University, Norfolk, VA, USA.
Email: treid003@odu.edu

Funding information
Institute of Museum and Library Services, Grant/Award Numbers: LG-252362-OLS-22, LG-256695-OLS-24

Abstract

Advertisements have always been a part of our cultural heritage, and this also applies to online web advertisements. Unlike print ads, there are serious technical challenges involved in archiving and successfully replaying ads displayed in web pages. To explore these challenges, we created a small dataset of 250 web ads. Ultimately, we collected and archived 279 ads, which we classified into 5 different categories: combination ads, image ads, embedded web page ads, video ads, and text-only ads. In our sample, combination ads were the most prevalent and, not surprisingly, text-only ads were the easiest to archive and replay. During the course of this study, we encountered five major problems in archiving and replaying the web ads. We detail the issues uncovered and provide suggestions for ameliorating the replay of some web ads, in addition to other dynamically loaded embedded web resources.

1 | INTRODUCTION

The founder of the Internet Archive, Brewster Kahle, noted as early as 1997 that the web constituted "a storehouse of valuable scientific, cultural and historical information" (Kahle, 1997, p. 82). Almost a quarter century later, Webster (2020) characterized the web in similar terms, but also stressed its potential as "a vast but underused scholarly resource for the study of almost every possible aspect of the last two decades" (p. 1). Despite calls to action from numerous scholars, however, web content has been hemorrhaged (D. J. Cohen, 2010; Nelson, 2012; Pennock, 2013).

Whether impelled by legal obligation, business purposes, social or cultural interest, or scholarly and/or historical research, web archiving involves collecting, storing, preserving, and providing long-term access to content (Ball, 2010; Cook, 2018; Niu, 2012; Pennock, 2013). Web archives provide evidence about the activities of their creator(s), their users, or about the period in which the archived content was created or modified.

Because the web depends upon advertising revenue (Einstein, 2017; Hwang, 2020), web ads constitute a particularly important type of dynamic content. Just as physical ephemera in libraries, archives, and museums undergird compelling research, so do online ads illuminate not only the contemporary objectives of advertisers, but also social norms, values, and ideals in ways that just news stories cannot. As a result, advertisements provide foundational source material for political, social, cultural, and business scholarship, especially in unpacking research questions concerning race, ethnicity, gender, and socioeconomic class (L. Cohen, 2003; Ewen, 2001; Leach, 1994; Marchand, 1985; Packard, 2007). According to historian Jackson Lears (1995), advertisements validate certain worldviews and structures and marginalize others. Advertising, he contends, promotes "the dominant aspirations, anxieties, even notions of personal identity, in the modern United States" (p. 2).

Despite web ads' importance, little has been done to build collections of them (Beard, 2018). In large measure, this gap results from both the novel and complex

technical work it demands, as well as the common attitude that ads are a nuisance to be avoided or blocked. Our research therefore addresses the following question: what are the key obstacles users face in archiving and replaying web ads?

First, we define and situate the core concepts that underpin our work. Next, we review the relevant literature on archiving dynamic content. Then, we explain our methods for archiving, replaying, and identifying resources associated with the 279 ads from our dataset, as well as the tool we developed to find difficult-to-replay ads. Afterward, we describe the five technical challenges we encountered in archiving and replaying web ads. Following this, we discuss the implications of our work for practice and research. Finally, we conclude and offer directions for future research.

2 | BACKGROUND

To contextualize the archiving and replay problems we identified, we first describe terms and concepts related to web archive crawlers, replay systems, and loading web advertisements on the live web.

2.1 | Web archive crawlers and replay systems

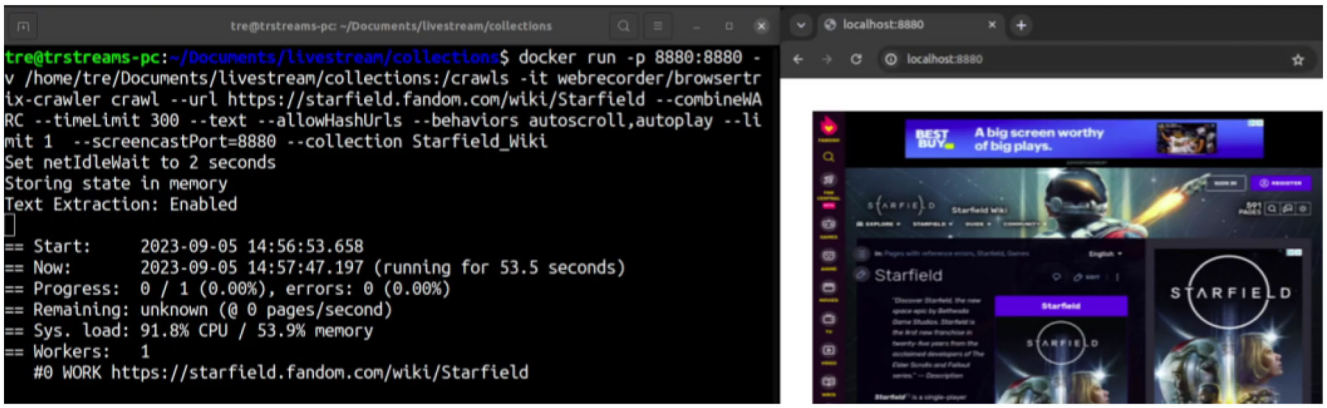
Web archiving involves using a crawler to collect content from the World Wide Web and preserve it in an archival format such as WARC (Web ARChive) (International Internet Preservation Consortium, 2015) or WACZ (Web Archive Collection Zipped) (Kreymer & Summers, 2021). An example crawling session is shown in Figure 1a. After the content is archived, a web archive replay system can display the archived version of the resource(s) in a browser. Some web archive crawlers use a graphical user interface (GUI)–based web browser, such as Chrome, or a headless (non-GUI) browser.

To replay an archived web page, a web archive replay system loads the archived content, or *memento*, in a browser, allowing users to view a previously stored version. An example is shown in Figure 1b. Using Memento (Van de Sompel et al., 2013) terminology, a URI-R is the URI of an Original Resource, which denotes the state of a web resource on the live web at the time it was archived. A URI-M is the URI for a memento, which is a previous state of an Original Resource. Replay systems use a URL rewriting system such as Wombat (Webrecorder, 2018) to modify URLs referenced by an archived web page's HTML, CSS, and JavaScript files. This rewriting changes a URI-R to a URI-M. In doing so, it prevents live web resources from loading during replay. An example URI-M from the Internet Archive's Wayback Machine is https://web.archive.org/web/20221220095518/https://www.google.com/. This URI-M includes the datetime indicating when the web page was archived (the *Memento-Datetime*): "20221220095518" (2022-12-20T09:55:18Z) and the URI-R for the archived web page: "https://www.google.com/." Some web archives, such as Perma.cc (Dulin & Ziegler, 2017) and Archive.today (Nelson, 2013), do not include the datetime or the URI-R in their URI-Ms (an example URI-M for https://www.google.com/ from Perma.cc is https://perma.cc/V2KT-MYA6 and from Archive.today is https://archive.is/SSsjK).

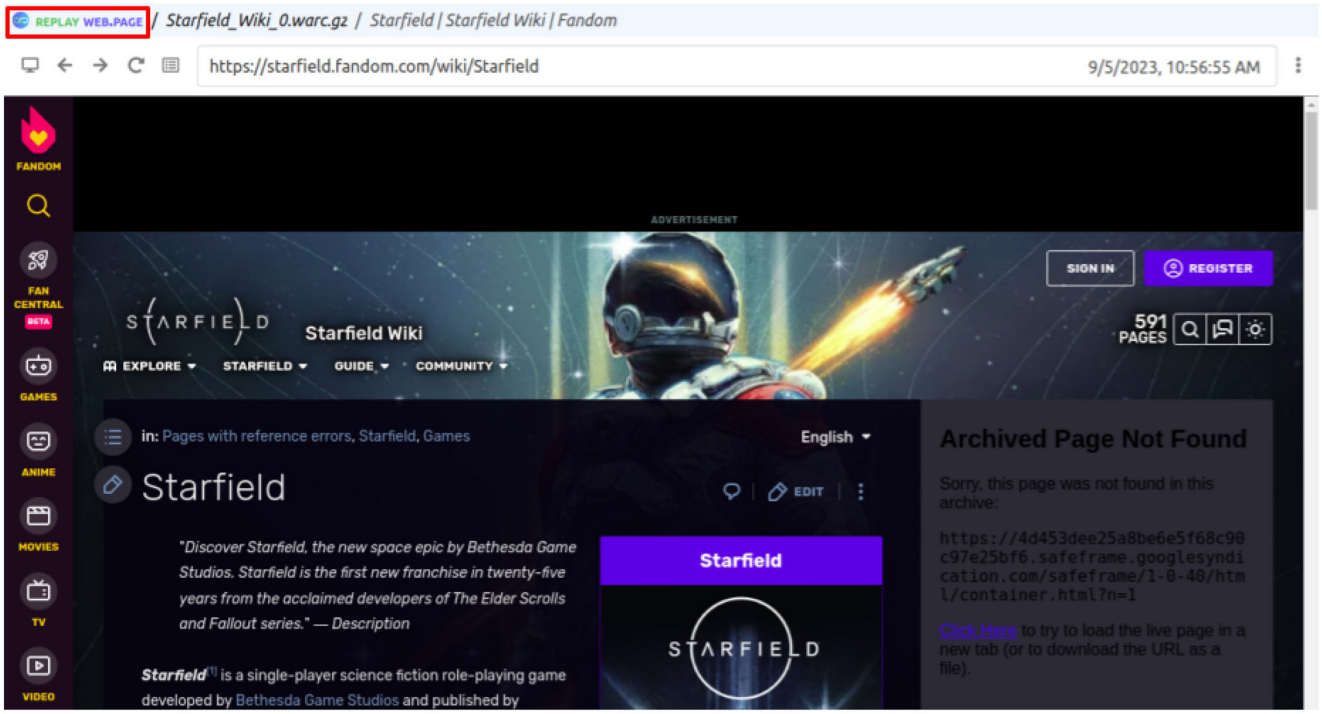
Some replay systems, such as ReplayWeb.page, use service workers (Archibald et al., 2022) to perform client-side URL rewriting. Service workers intercept HTTP requests made by archived web pages during replay (Alam et al., 2017; Berlin et al., 2023). In contrast, server-side URL rewriting modifies the URLs in archived HTML, CSS, and JavaScript files before it sends the archived resources to the user. This type of rewriting is useful for rewriting URLs in HTML and CSS files. On the other hand, in client-side URL rewriting, a service worker or a client-side rewriting JavaScript library like Wombat changes the URL (Berlin, 2018). Wombat, for example, performs the same URL rewriting that is done server-side and overrides the JavaScript Application Programming Interface (API) (Berlin, 2018). Notably, client-side URL rewriting outperforms server-side URL rewriting for URLs dynamically generated by JavaScript.

2.2 | Loading web advertisements

The process of loading web advertisements involves three steps, as illustrated in Figure 2. First, the publisher (website owner) adds advertisement code, which employs HTML elements such as div and iframe, to create ad spaces[1] on their website (Google, 2015). Example code for creating an ad slot is shown at the top row of Figure 2. During this step, the JavaScript files needed to use the ad service's API, like Google's gpt.js[2] and pubads_impl.js,[3] are loaded. Second, the publisher selects the ad(s) to load either by holding an auction (Amazon, 2024; Google, 2015) (example code for requesting auction bids is shown on the left side of the second row of Figure 2) or agreeing to host a sponsored ad (Google, 2020a) (the Ashoka-sponsored ad selected by IGN is shown on the right side of the second row of Figure 2). The ad service selects an auction's winning bid based on metrics such as cost per mille (CPM) (the cost per 1000 ad impressions; Amazon Ads, 2024; Google, 2019b). In the final step, the



(a) Browsertrix Crawler archiving a web page



(b) Replaying the archived web page with ReplayWeb.page

FIGURE 1 Example crawling and replay sessions.

ad script dynamically retrieves and renders the selected ad(s) into designated ad space(s) on the web page, ensuring proper display and interaction capabilities. In Figure 2, the Intel and Fortnite ads (outlined in red) were selected through auctions, while the website publisher (IGN) selected the Ashoka-sponsored ad (outlined in purple).

Some Google ads, such as embedded web page ads, use SafeFrame (Google, 2020c)—an iframe based on Interactive Advertising Bureau (IAB) specifications (IAB, 2016)—to enhance communication and security between the ad and the hosting web page. Unlike regular iframes, SafeFrames enable controlled communication between the publisher's web page and the ad, while limiting potentially harmful interactions (Interactive Advertising Bureau, 2016). One issue affecting web archiving of ads, to be explored further in Section 5.3.1, is that the current type of SafeFrame includes a random value in the URL's subdomain to sequester SafeFrame content, thereby strengthening security (Google, 2020a). Figure 3 shows an example of a Google SafeFrame URL with a dynamically generated random subdomain. This type of SafeFrame is difficult to replay because the random value generated during replay differs from the random value that was generated at crawl time.

Like Google's SafeFrame, Amazon's ad iframe uses a random value in the iframe's URL, albeit one located in

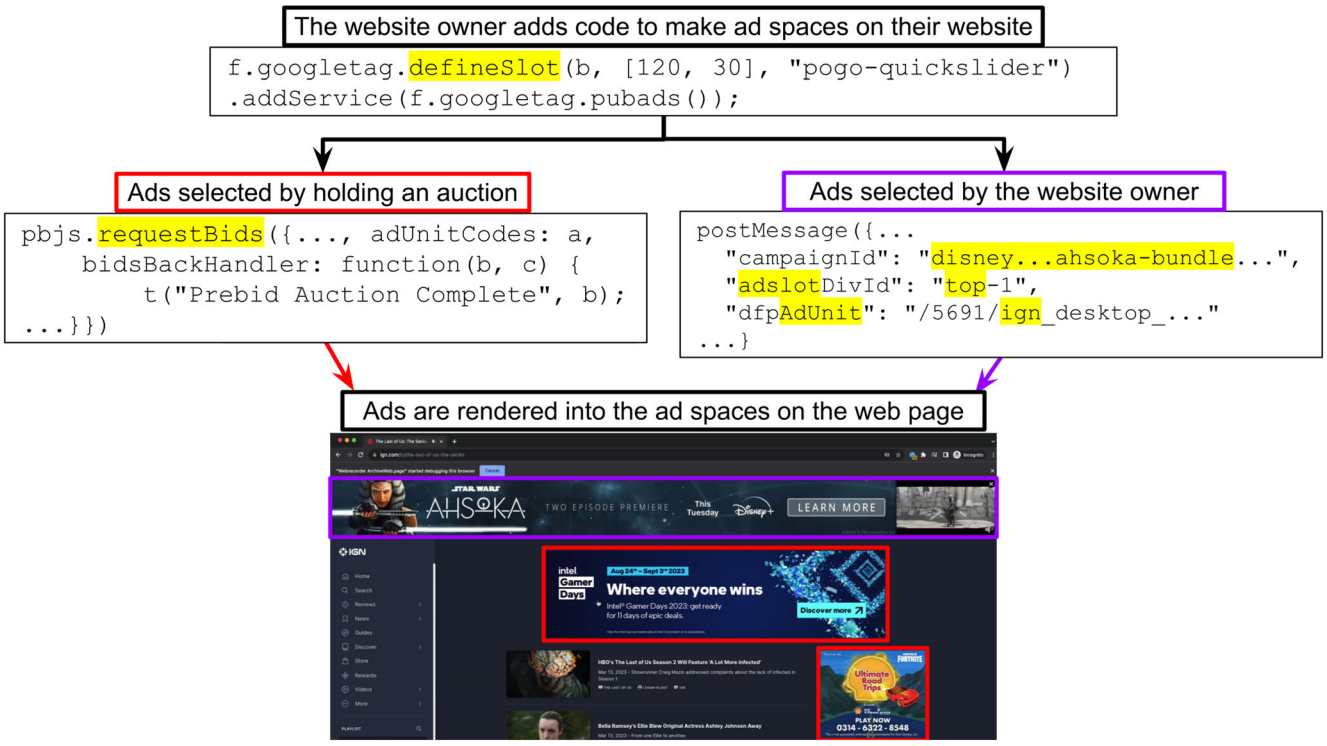


FIGURE 2 Loading advertisements into a web page.

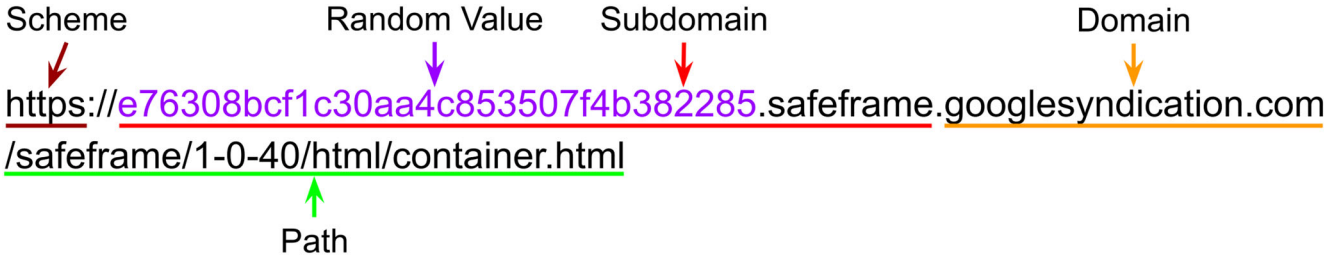


FIGURE 3 Google SafeFrame URI.

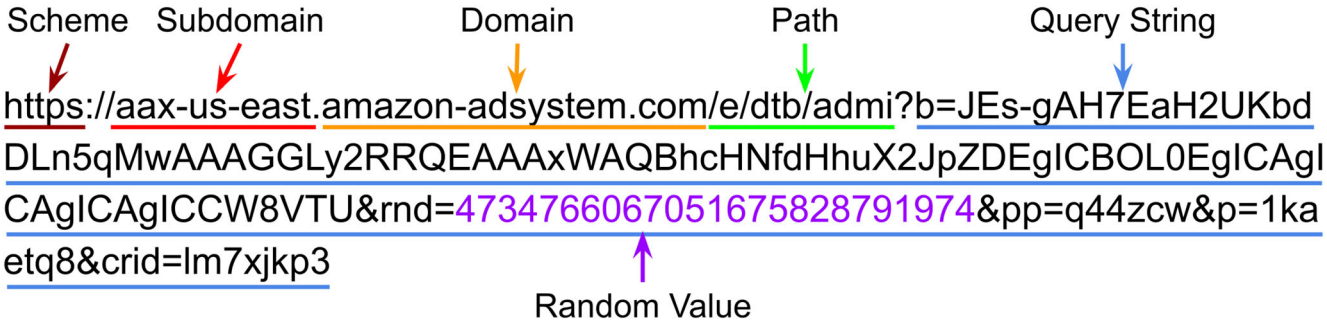


FIGURE 4 Amazon ad iframe URI.

the query string (Figure 4) instead of the subdomain. A random value's presence in the query string results in the replay system generating an unarchived Amazon ad iframe URL, which can prevent an Amazon ad from loading during replay.

3 | LITERATURE REVIEW

This research engages studies detailing the problems with archiving and replaying dynamic content, transparency challenges with web archiving services, browser-based

tools for archiving dynamic content, and other web ad datasets.

3.1 | Archiving and replay challenges

Kelly et al. (2013) investigated how the archivability of websites, or the ease with which they can be archived and preserved for future access, has changed over time. They attributed non-persistent URIs as the cause of incomplete fetch results from embedded JavaScript. Since web ads are embedded web resources typically loaded with JavaScript, poor website archivability hinders ad archiving. Reyes Ayala (2022) also created metrics for measuring the quality of archived web pages, which include visual correspondence, interactional correspondence, and completeness. Visual correspondence measures the similarity in appearance between archived and the live web pages. Interactional correspondence refers to the ability to perform the same interactions on the archived and the live web pages. Completeness is determined by how much content from the live web was archived successfully. These metrics are essential for determining the successful replayability of a web ad, as they assess the extent to which the archived ad replicates the visual and interactive experience of the original ad during the crawling session.

Goel et al. (2022) identified several sources of non-determinism that caused problems with replay including server-side state, client side state, client characteristics, and "`Date`," "`Random`," and "`Performance`" APIs. They also employed URL matching algorithms (query-strip and fuzzy matching) to handle the variance in dynamically generated URLs during replay to match the requested URL with a crawled URL. Kiesel, de Vries, et al. (2018) observed that JavaScript would generate different random values during replay than it did during crawl time. These problems with random values impact web ads, because ad services use random functions when generating some ad URLs. Brunelle et al. (2017) focused on the challenges posed by dynamically generated website content that is customized for each user session or exposed through user interactions. They highlighted examples where archives missed capturing important context, including embedded multimedia, third-party widgets, and personalized recommendations due to their focus on static objectives. Brunelle advocated for advancing web archiving methods such as scripted browsing and browser instrumentation to create web archives that more comprehensively preserve dynamic, interactive web experiences. This research is related to our work since some web ads are customized based on a user's session and require user interactions to load the resources for the ad.

3.2 | Transparency problems with web archiving services

Maemura et al. (2018) explored the challenges of ensuring transparency in the web archiving process by focusing on the importance of documenting provenance, or the origins and context of web archives. The authors highlight that the lack of detailed metadata about how web crawls are conducted, including crawl parameters, exclusions, technical settings, and the intentions of archivists, creates significant transparency problems.

Internet Archive's Save Page Now (SPN) (Graham, 2019) was one of the tools we used when archiving web ads, and Ogden et al. (2024) highlighted challenges in examining and reverse-engineering SPN's archiving processes. They identified three key challenges: examining web archive data at scale, the lack of transparency in the SPN data pipeline, and that SPN is not a fixed entity. This research is relevant due to the lack of transparency in SPN's ad archiving practices, stemming from Internet Archive's undocumented web advertisement exclusion policy.

3.3 | Browser-based web archiving tools

Brunelle et al. (2015) argued that traditional web archiving methods often fail to capture deferred representations, representations that are not immediately accessible upon page load but that are loaded after user interactions or other events. These researchers proposed a two-tiered crawling approach to archive deferred representations of web pages. Since most ads are deferred representations, it was necessary for us to use browser-based web archiving tools when archiving web pages with ads. Bhatt et al. (2015) also utilized an automated headless browser when archiving web pages to render and capture the full state of web pages, including any content loaded dynamically through user interactions or scripting. Notably, their tool harnessed Ajax, Flash, Silverlight, and complex JavaScript to recreate the user browsing experience that allowed it to capture dynamic content missed by traditional crawlers. Besser (2017) and the Internet Archive developed Brozzler,[4] a browser-based crawler that differs from tools by Brunelle et al. (2015) and Bhatt et al. (2015), because Brozzler can use a regular web browser (with a graphical user interface) when archiving web pages. This feature allowed us to view the live web page during the crawling session, which was useful when we compared the visual appearance of the live and archived version of an ad.

After archiving ads with tools like Brozzler, we replayed the ads with ReplayWeb.page, which is a replay

system that employs service workers to load embedded resources. Alam et al.'s (2017) work is relevant, as they were early adopters of the service worker API for web archive replay, demonstrating its capability to intercept and reroute requests for embedded resources, instead of using URL rewriting.

3.4 | Other web ad datasets

Examples of studies on web advertisements include the New York University Ad Observatory project (New York University Cybersecurity for Democracy, 2025), the Australian Ad Observatory project (Angus et al., 2024; Burgess et al., 2022), and the Bad Ads project (Yoshikawa & Roesner, 2025; Zeng et al., 2021). Both Ad Observatory projects concentrated on archiving ads from social media platforms. In contrast, the Bad Ads project focused on archiving web ads outside of social media platforms.

The New York University Ad Observatory project collected and analyzed the political and issue-based web ads that users see on Facebook and YouTube platforms (New York University Cybersecurity for Democracy, 2025). Rather than scraping all user activities, Ad Observer, a browser extension tool, allows users to voluntarily donate anonymized information. The donated data is aggregated into a public database to show who is paying for the ads, how audiences are targeted, and what messages are being amplified. Ad Observer creates a structured archive of web ads directly from end users' browsing experiences, providing a resource to explore how algorithms shape targeting practices, the responsibility of actors in digital campaigning, and the wider social effects of personalized advertising.

The Australian Ad Observatory at the ARC Centre of Excellence for Automated Decision-Making and Society monitors and studies digital political advertising (Angus et al., 2024; Burgess et al., 2022). The project is primarily based on data from the Facebook Ad Library and Google Transparency Reports. In addition, they created a browser extension, which scrapes sponsored web ads in users' news feeds and automatically extracts web ads. The goal is to provide accountability for and transparency of advertising targeted to Australian users on social media.

Zeng et al. (2021) investigated user perceptions of problematic online advertising, emphasizing how users actually experience and perceive ads. To create their dataset of 1838 unique ads, they used a browser-based web crawler called Puppeteer to automatically scrape ads identified using EasyList, an ad-blocking list. Through surveys they identified a range of factors that lead users to classify ads as "bad," including misleading or manipulative content, excessive intrusiveness, disruption of browsing experiences, and concerns about privacy or tracking.

A more recent paper from the Bad Ads project scrutinized how ads circulated on news and media websites during the 2024 US elections (Yoshikawa & Roesner, 2025). By analyzing the prevalence of political ads and differences based on geolocation, the study identified specific features of ads, including misleading content as clickbait, intrusive formats, and manipulative targeting. This work indicated that problematic ads are not only a matter of more rigorous scrutiny and transparency but also of user experience and content perception.

4 | METHODS

We created a dataset (Reid, 2024b) of 279 unique recent (January to June 2023) advertisements culled from the live web. We archived them with appropriate tools and identified problems with both their archiving and their replay. The archived ads in our dataset can be viewed at https://savingads.github.io/themed_ad_collections.html.

4.1 | Creating a dataset of recently archived ads

To construct our dataset, we randomly selected websites from SimilarWeb's top websites worldwide (SimilarWeb, 2023) (including all categories except "Adult"[5]), rendered a web page from each website, and if the page loaded ads, archived it. We repeated this process until we had collected at least 250 ads.

Ultimately, we selected 17 web pages and 279 ads to archive (Table 1). We used Internet Archive's Save Page Now, Arquivo.pt., archive.today, and Conifer because these web archiving services permit the archiving of an unlimited number of web pages cost-free. We also used three browser-based tools (ArchiveWeb.page, Webrecorder, 2020, Browsertrix Crawler, and Brozzler) that facilitate archiving dynamically loaded web resources. We used ArchiveWeb.page and Browsertrix Crawler to archive four web pages each, Brozzler to archive one web page, and four web archiving services (Save Page Now,[6] Arquivo.pt., archive.today, and Conifer) to archive two web pages each. We did not archive four web pages with each web archiving tool because we had reached our goal of 250 advertisements. We successfully archived (captured all the resources needed) nearly all (273) of these ads.

Six ads were not fully archived. One required a specific user interaction (clicking on a play button) to load all of the ad resources. Three ads requested unarchived JavaScript and HTML files during replay. We used ReplayWeb.

TABLE 1 The number of archived ads from each web page that we archived. All URLs are hyperlinked to the live web page, including the truncated URLs.

Web page	Number of ads archived	Web archiving tool	Replay system
https://canalturf.com	66	Save Page Now	Wayback Machine
https://www.lequipe.fr/Tous-sports/.../1389820	37	ArchiveWeb.page	ReplayWeb.page
https://www.leroymerlin.com.br/	31	Arquivo.pt	Arquivo.pt
https://www.ign.com/articles/the-last-...-review	24	ArchiveWeb.page	ReplayWeb.page
https://www.facebook.com/	24	ArchiveWeb.page	ReplayWeb.page
https://www.cnn.com/	23	ArchiveWeb.page	ReplayWeb.page
https://www.marketwatch.com/	18	Brozzler	ReplayWeb.page
https://www.diy.com/	13	Conifer	Conifer
https://www.realtor.com/news/.../frank-...7-9m/	11	Browsertrix Crawler	ReplayWeb.page
https://mortalkombat.fandom.com/wiki/Tag_Team_Ladder	8	Browsertrix Crawler	ReplayWeb.page
https://tokopedia.com	6	Arquivo.pt	Arquivo.pt
https://www.deviantart.com/kvacm/art/Hellstone-...274	5	Browsertrix Crawler	ReplayWeb.page
https://unsplash.com/t/wallpapers	3	archive.today	archive.today
https://sports.yahoo.com	3	Conifer	Conifer
https://www.vidal.ru/novosti/kak-potreblenie-...744	2	Save Page Now	Wayback Machine
https://canalturf.com	2	ArchiveBot	Wayback Machine
https://canalturf.com	1	Perma.cc	Wayback Machine
https://www.youtube.com/watch?v=PZShwWiepeY	1	Browsertrix Crawler	ReplayWeb.page
https://www.tripadvisor.it/Tourism-...-Vacations.html	1	archive.today	archive.today

page's URL prefix search and removed the query string from the requested URI to see if a resource with a similar URI was archived, but did not find these JavaScript and HTML files in ArchiveWeb.page's output file (WACZ). For one Flashtalking ad, it was impossible to determine if the ad was successfully archived because this type of ad[7] cannot be replayed outside of its ad iframe. This prevented us from comparing the ad resources that loaded on the live web and (presumably) would load during replay. The last partially archived ad's dynamically generated URL (during crawl time) included an `e` query string parameter that prevented[8,9] some of the images from loading. There were seven more images on the live version of the ad, but the live web ad was checked months after crawl time, so we cannot compare it to the archived version.

4.1.1 | Finding archived ads' web resources

By using ReplayWeb.page's URL search feature (Webrecorder, 2024) and our own bespoke Display Archived Ads tool (Reid, 2024a), we found that 55 out of 279 advertisements were not replayable when loading the archived *containing web page*, which is the page that loaded the ad during a crawling session. We used ReplayWeb.page's URL search feature to identify ads in our WARC and WACZ files. This feature allows users to specify the MIME type, which facilitates searching for a specific ad type like image ads. It also allows for a prefix search, which we used to identify resources associated with services like Flashtalking, Innovid, Amazon display ads, and Google AdSense.

We built a Display Archived Ads tool[10] (Figure 5) to display most of the HTML, image, and video files inside of a given WARC file. Our tool depends on the warcio (Kreymer, 2024), pywb (Webrecorder, 2013), and Selenium (Selenium, 2024) software packages. Warcio retrieves the URLs for the web resources from the WARC file. pywb replays the archived ads within an iframe,[11] and Selenium opens a web browser and executes the JavaScript necessary to display the ads. Our tool offered two affordances. It enabled us to filter out some of the known ad resources that remain invisible during replay,[12] thereby speeding up the review. Further, by allowing us to display an ad's live version beside its archived version, the tool showed problems with replay.

4.1.2 | Replaying archived ads

To replay the archived advertisements, we used four web archiving services (Internet Archive's Wayback Machine, Arquivo.pt., archive.today, and Conifer). We replayed the

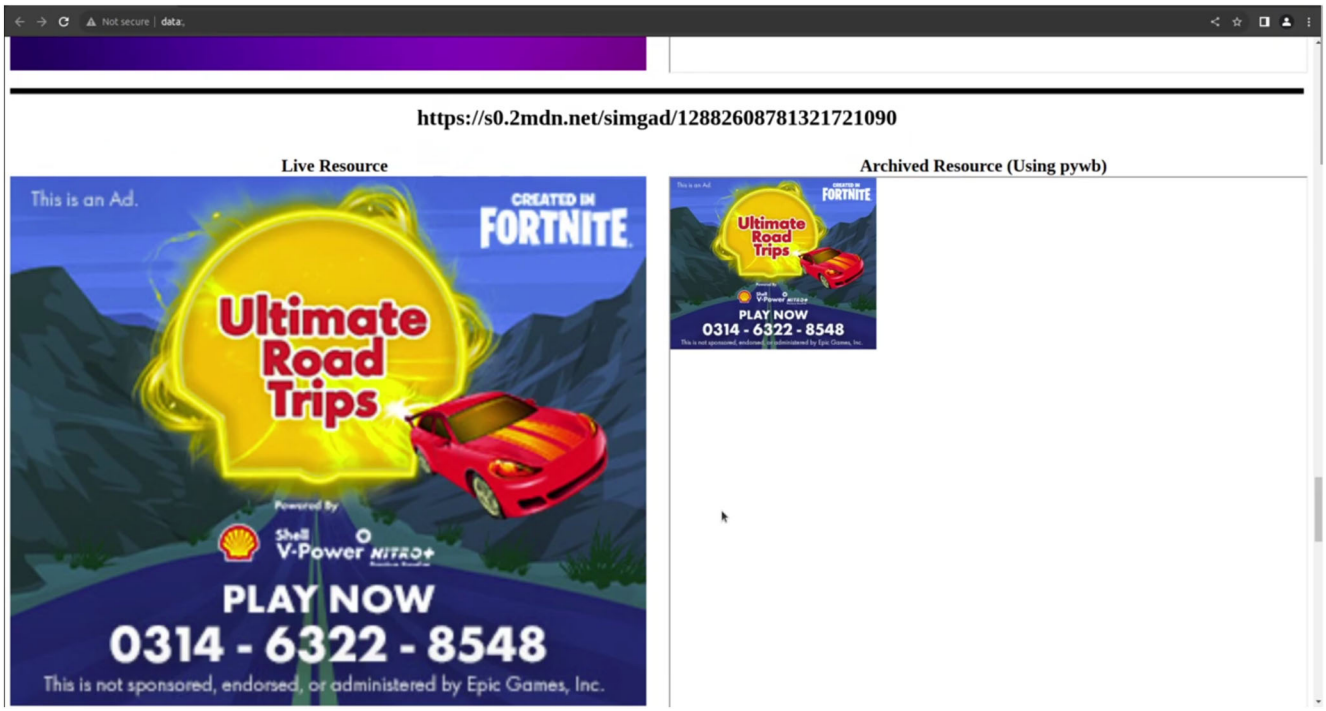


FIGURE 5 Our tool for displaying potential ads loads the live web ad beside the archived version of the ad.

web ads that we archived with a web archiving service with a service from the same web archive. We attempted to use three other replay systems (ReplayWeb.page, pywb, and OpenWayback; International Internet Preservation Consortium, 2012). In the end, we only used ReplayWeb.page to replay the archived ads from our WARC and WACZ files, because at the beginning of 2023, ReplayWeb.page could replay more dynamically loaded web resources than pywb and OpenWayback. pywb (version 2.7.3) failed to replay archived web ads that relied upon an Amazon ad iframe. During replay, OpenWayback version 2.4.0 (Ruest, 2019) loaded live web advertisements instead of the archived ads.[13]

4.2 | Categorizing advertisements

We categorized our 279 ads into image, video, embedded web page, text-only, or a combination. The first three types are associated with one web resource that is viewable outside of the containing web page (provided the user knows the URI associated with the resource). Figure 6 (image ad), Figure 7 (video ad), and Figure 8 (embedded web page ad) show examples.[14]

Text-only ads (Figure 9) cannot be viewed outside of the containing web page because the web page loads the text. The combination category comprises ads (Figure 10) that rely upon multiple resources and are constructed inside of the containing web page or ad iframe. Like text ads, combination ads cannot be viewed outside of the containing web page. Next, we coded each ad topically. Table 2 shows the 24 themes and the corresponding number of ads for each. Most themes (17 out of 24) aligned with SimilarWeb's website categories. The other themes included "Internet and Mobile Service Provider," "Politics," "Funeral Services," "Charity," "Military," "Sponsored Brand," and "Unknown" (ads that we were not able to replay and could not view on the live web).

5 | FINDINGS

In this section, we first describe the results of replaying the advertisements in our dataset and the replay problems that occurred for each ad type. Second, we discuss two archiving problems that involved the Internet Archive's Save Page Now excluding ads and recent versions of Chrome being incompatible with Brozzler. Third, we lay out three replay problems caused by random values in URLs, by a non-existent URL being requested, and by a Chromium bug that prevented service workers from accessing resources in an "`about:blank`" iframe.

5.1 | Replaying archived ads in the containing web page

Figure 11 shows the number of successfully archived ads, those ads with all of their web resources successfully captured by a web crawler, that we replayed in the

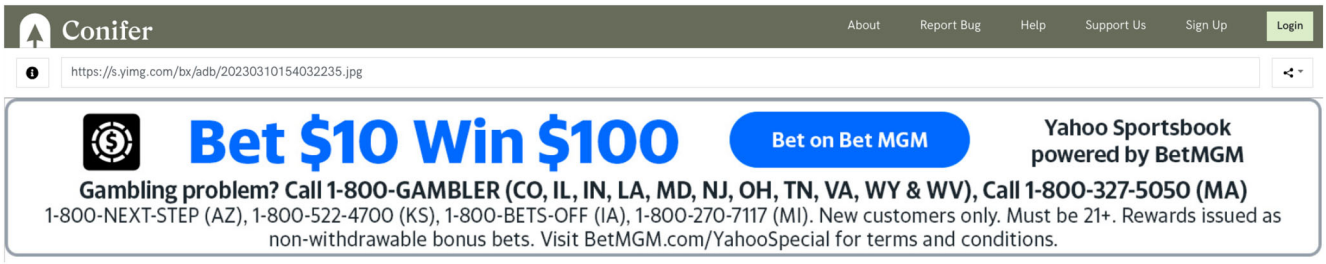


FIGURE 6 Image ad.

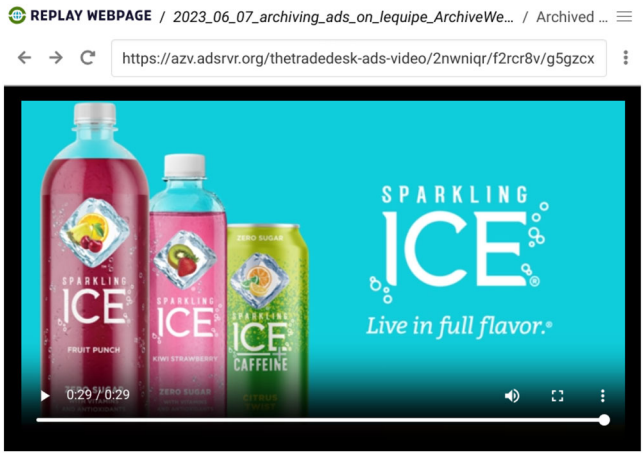


FIGURE 7 Video ad.

FIGURE 8 Web page ad.

containing web page. The number of ads that always replayed in the containing web page was 111 (40.66%). Another 113 (41.39%) *sometimes* failed to replay in the containing web page. For the "sometimes replay successfully" category, the ads failed to load depending upon the version of the replay system used (an error that sometimes occurs when replaying a web page, for example when the web page takes too long to dynamically load all the resources), or a different set of ads being replayed. There were 49 ads (17.95%) that never replayed in the containing web page (no requests were made for these ads during replay). Approximately 26 ads were not requested because the ad iframe (Google SafeFrame or Amazon's ad iframe) failed to replay successfully, which prevented the ad from being loaded.

The number of ads for each ad category is shown in Figure 12. Combination ads are the most common ad type (156 ads, or 57.14% of the ads in the dataset). Image ads were also common (80, or 29.30% of the ads in the dataset). Figure 13 shows the number of archived ads per category for each web archiving tool. Figure 14 shows the percentage of ads that successfully replayed in the containing web page for each ad category. First, the text-only ads always replayed successfully. Second, over half the image ads always replayed successfully. Image ads did not replay for three reasons: the ads were not selected to be replayed (or no request was made for the image during replay time), the web page sometimes failed to dynamically load all the necessary resources, or the ad used an ad iframe that failed to replay. Third, most of the video ads failed to replay (80% always failed). The video ads failed to replay because during replay time either the HTTP request for the ad had a 404 status code or no request was made to load the video into the ad slot. Fourth, around 90% of the embedded web page ads did not consistently load during replay because embedded web page ads usually require an ad iframe. Google SafeFrames always failed to replay. Depending on the version of ReplayWeb.page used, Amazon ad iframes sometimes replayed. Moreover, the web archives we used always failed to replay the Amazon ad iframe. Fifth, nearly three-fifths (58.97%) of the combination ads sometimes replayed. The combination ads did not load either because the set of ads selected was different each time the web page was loaded or because the web page did not dynamically load all of the necessary resources.

FIGURE 9 Text-only ad for a sponsored news article.

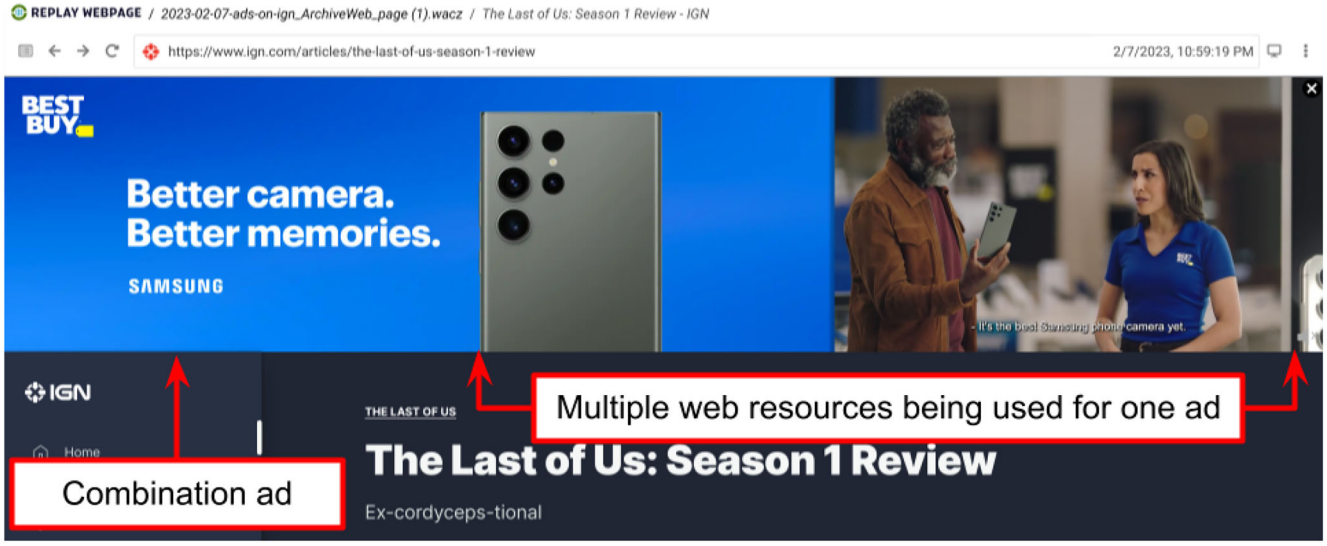


FIGURE 10 Example combination ad that uses three images and one video.

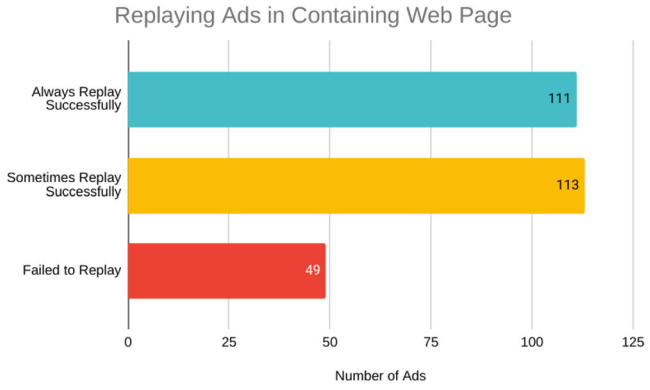


FIGURE 11 The number of successfully archived ads replayed in the web page that loaded the ad during the crawling session.

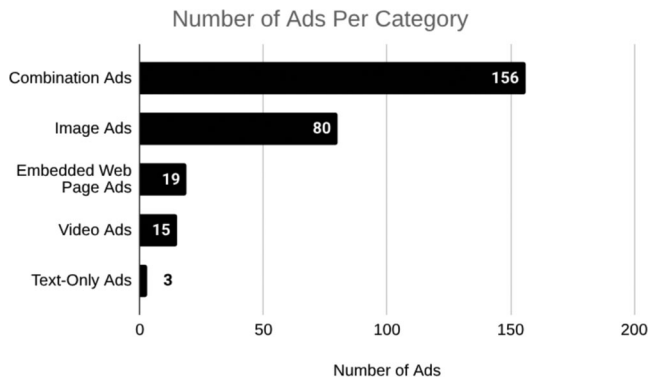


FIGURE 12 The number of successfully archived ads for each ad category.

5.2 | Archiving problems

Two problems prevented web ads from being archived: Internet Archive's Save Page Now intentionally excluding ads, and Brozzler being incompatible with recent versions of Google Chrome. We inspected ads available on Wayback Machine using three URL searches with the prefixes "https://s0.2mdn.net/,"[15] "https://s-static.innovid.com/,"[16] and "https://cdn.flashtalking.com."[17] Perma.cc, Archive Team, and Archive-It users archived or uploaded most of the recent (before August 2023) ads, but Internet Archive prevented Save Page Now users from archiving image, video, and web page ads, as well as the Google SafeFrames and other ad iframes needed to load them. After August 2023, Save Page Now allowed the archiving of more ads, provided the user directly archived the URI-R associated with the ad, rather than just the containing page. However, in October 2023, we archived a web page that loaded Google ads and found that Save Page Now still blocked ads loaded by a web page from being archived (Figure 15)—the image ad in Figure 15a was not archived (Figure 15b). In April 2025, we retested whether Save Page Now allows users to directly archive ad URLs. Save Page Now blocked the ad URLs by default, but a new option "Disable ad blocker" for logged-in users enabled archiving them. Using Save Page Now anonymously will result in archived web pages without ads.

Save Page Now also blocked URLs that included an ad-related file or directory name in the URL's path. One example was a social media account on Twitter named "displayads" (Figure 16a). Posts from this account were blocked by Save Page Now, because the social media platform used the ad-related username as a directory in the URL's path. We communicated with Wayback Machine staff about this issue (in August 2023) and they made this account archivable (Figure 16b). Our technical report (Reid et al., 2025) includes more examples of web pages that were blocked by Save Page Now.

Brozzler became incompatible[18] with versions of Google Chrome released after March 2023. When we attempted to archive a web page using the `brozzle-page` command, Brozzler failed to load the web page (Figure 17), which prevented it from being archived (Figure 18). When we employed the same command in early 2023, however, Brozzler loaded the desired web page (Figure 19). After testing Brozzler on a range of web pages, we concluded that browser incompatibility is to blame. We reported Brozzler's incompatibility problem, which was resolved during 2024. Brozzler is now compatible with Chrome version 130[19] and later.

TABLE 2 List of themes used for the ads in our dataset.

Theme	Number of ads
Shopping	85
Finance	27
Vehicle and automotive	23
Business services	21
Travel	19
Entertainment	16
Health	15
Real estate	15
News	14
Unknown	6
Internet and mobile service provider	5
Art	4
Beauty and cosmetics	4
Gaming	4
Military	4
Food and drink	3
Gambling and fantasy sports	3
Computer security	2
Fitness and sports	2
Pets and animals	2
Sponsored brand	2
Charity	1
Funeral services	1
Politics	1

5.3 | Replay problems

When we sought to replay ads from our dataset, we encountered three obstacles. First, the JavaScript for Google and Amazon ad services dynamically generated URLs with random values. Second, the JavaScript for Flashtalking ad service prevented the replay of an embedded web page ad outside of an ad iframe. Finally, some ads did not load during replay depending on the browser used.

5.3.1 | Replaying Google SafeFrames

Google SafeFrame uses a random value in the subdomain of the iframe's URL (Figure 3), which prevents replay. Figure 20 shows an example Google ad that failed to load. When the seven replay systems we tested (pywb, OpenWayback, ReplayWeb.page, Conifer, Wayback Machine, and Arquivo.pt) executed JavaScript code that generated a random number, the random value differed from the random value generated during crawl time. These replay

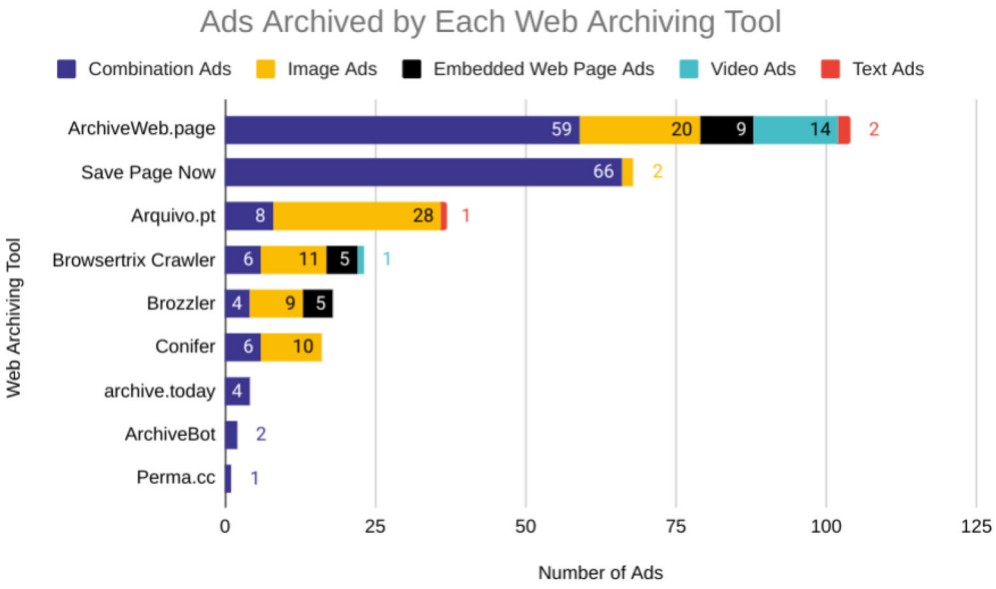


FIGURE 13 The number of ads archived by each web archiving tool.

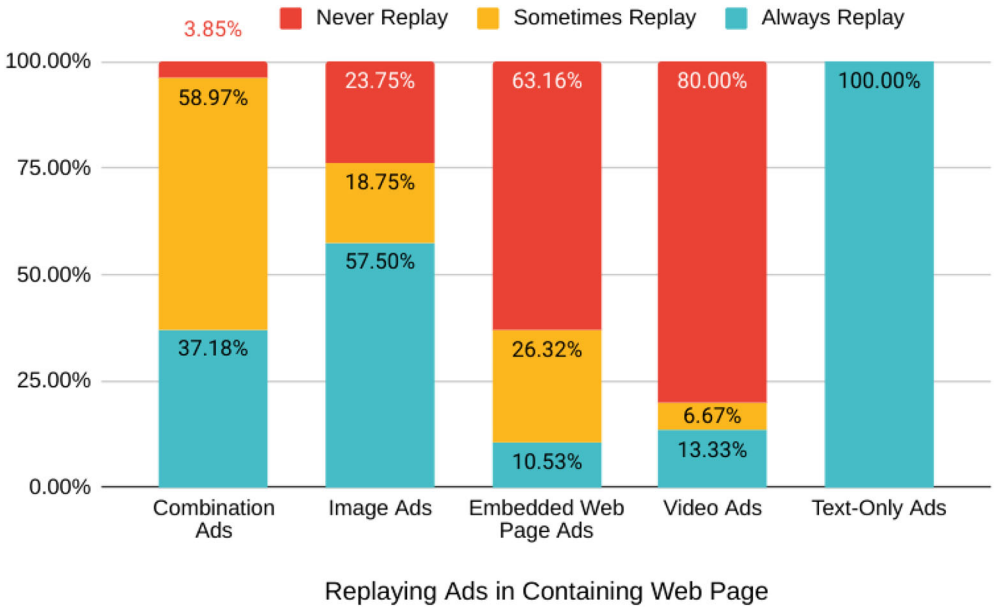


FIGURE 14 Successfully archived ads for each category that replayed in the web page that loaded the ad during the crawling session.

systems also generated different random values on each replay.[20] Since we successfully archived the Google SafeFrame and the ad content (Figure 21), this problem was caused by the replay system, not the crawler. When ReplayWeb.page, pywb, Conifer, or Wayback Machine attempted to load a Google SafeFrame for an advertisement, an HTTP status code of 404 (Not Found) was returned. In contrast, when loading a Google SafeFrame with Arquivo.pt., an HTTP status code of 307 (Temporary Redirect) was returned. Arquivo.pt.'s replay system changed the timestamp for the archived SafeFrame's URI-M, but failed to load the ad content into the successfully archived SafeFrame.

We created a test web page[21] to determine how random values were generated for a Google SafeFrame. Our technical report discusses how Wombat.js initialized the `Math.random()` and `crypto.getRandomValues()` functions and why the random number generators' seeds were different during the crawling and replay sessions (Reid et al., 2025). Replay systems employ rewriting tools like Wombat.js to overwrite the seed for the random number generators. This results in a more consistent replay where the random values generated should be the same upon each replay.

Loading Google ads in a containing web page is an example in which overwriting the seed for the random number generators did not result in the same random numbers being generated during each replay. In this case, the Google SafeFrame URL's random subdomain differed each time the archived web page was

(a) URI-Rs identified during a crawling session. The URI-R for an image ad is highlighted.

(b) The URI-R for the image ad was not archived.

FIGURE 15 Save Page Now identified the URI-R for an image associated with an ad during a crawling session, but did not archive the image. URI-R, URI of an Original Resource.

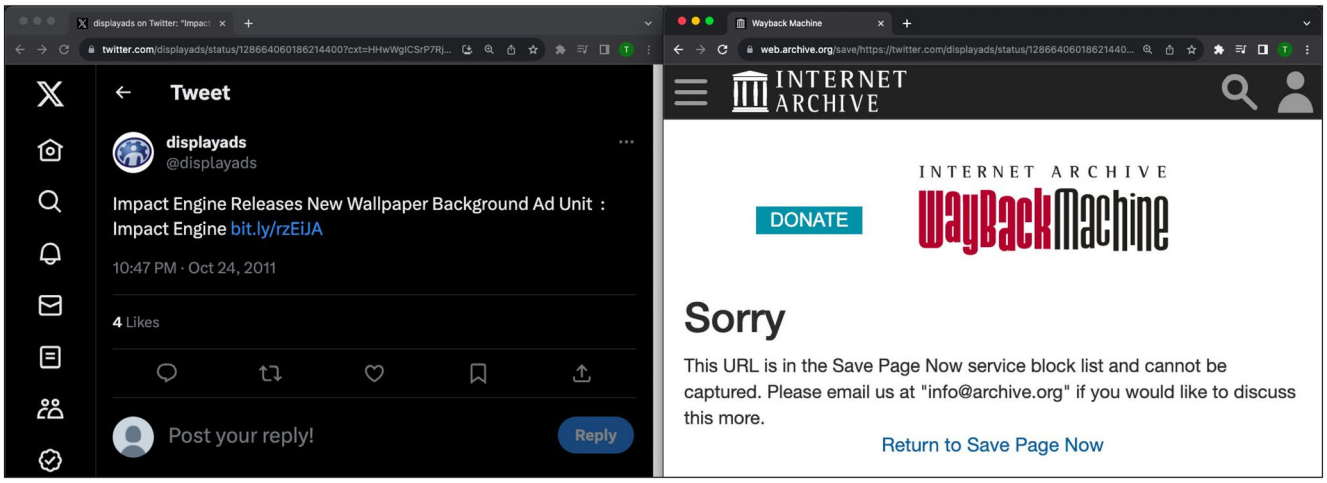
loaded (Table 3). The Google SafeFrame subdomain differed in Table 3 in part because a Google SafeFrame was loaded when the ad slot is close to the viewport, not immediately upon replay. Delaying the ads' loading can produce different timings in network communications, which "lead[s] to a varying execution order and thus a different order of pop-requests from the 'random' number sequence" (Kiesel, de Vries, et al., 2018).

If there are multiple JavaScript files making calls for `Math.random()` and `crypto.getRandomValues()` (e.g., running multiple ad services on a web page) before a Google ad is loaded, then the random number sequence will change, which causes variance with the random value included in the Google SafeFrame URL. In contrast, if the number of function calls to the random number functions is the same, then the random values will be consistent.

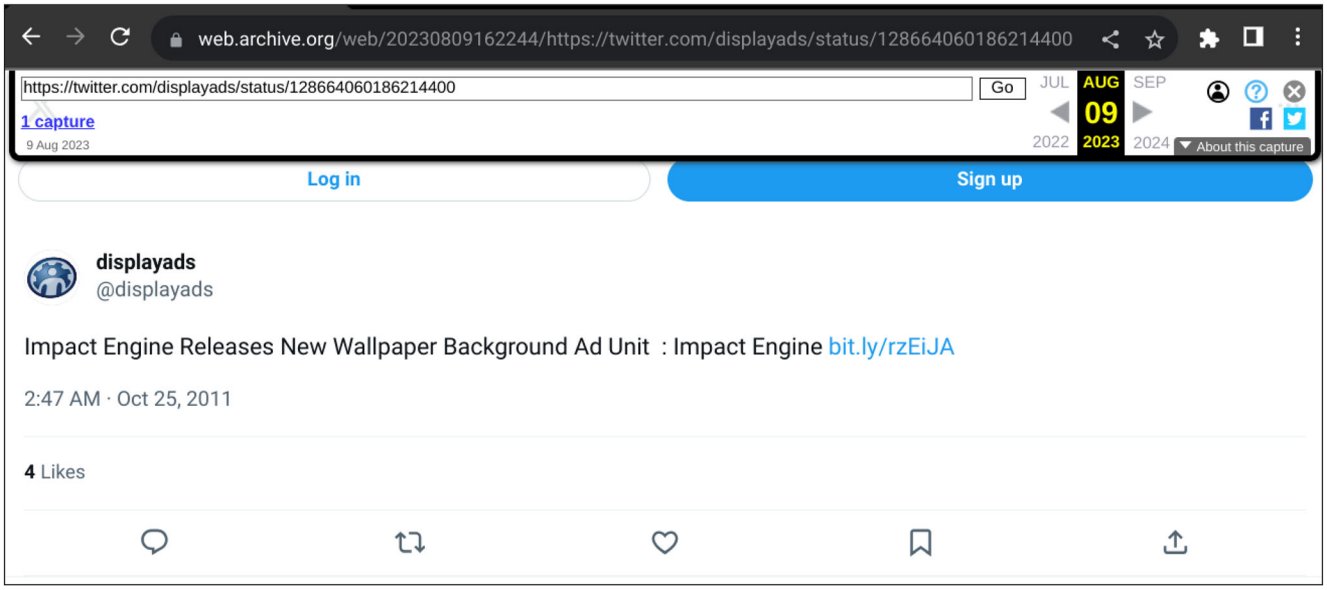
Overall, replay systems cannot generate the same random value that was generated during crawl time. Notably, *any* dynamically loaded resources that use random values will encounter this problem.

5.3.2 | Amazon ad iframe

To replay an Amazon ad, a crawler must archive both the embedded web page for the Amazon ad iframe and the ad loaded inside it. We encountered two challenges with Amazon ads. First, although the ad resources were successfully archived, pywb version 2.7.3 (Kreymer, 2023)



(a) Social media accounts with ad related usernames were blocked by Save Page Now.



(b) It is now possible to archive social media accounts with an ad related username.

FIGURE 16 Before June 2023, some social media accounts with ad related usernames were blocked by Save Page Now.

failed to replay some Amazon ads because a URL for the ad bid contained incorrect `ws` and `pid` query string parameters (Figure 22). The dynamically generated values for these parameters differed during the crawling and replay sessions. Pywb 2.7.3 only replayed Amazon ads initially loaded inside a Google SafeFrame. To enable this replay, however, we not only needed to know the URL for the Amazon ad iframe, but also loaded the ad outside of the containing web page.

Alone among the replay systems we tested, ReplayWeb.page successfully replayed this type of Amazon ad. Amazon ads use a random value in the URL's query string stored in the `rnd` parameter. ReplayWeb.page's approach for fuzzy matching made it possible to replay these Amazon ads even when the `rnd` parameter generated during replay differed from the one generated during crawl time (Figure 23). Requests that have different query string parameters during replay than during crawl time are handled by fuzzy matching to match requests during replay with responses that were captured during crawl time (Kiesel, Kneist, et al., 2018). While the random value generated in the query string of the URL did not prevent ReplayWeb.page from loading an Amazon ad, a random value used in the subdomain of a Google ad's URL did.

Amazon ads' use of random numbers in their iframe URLs caused another problem. Multiple ads may use the same base ad iframe URL, albeit with different query strings. This prevents some of the ads from being shown during replay because of how ReplayWeb.page uses fuzzy matching. If multiple ad iframe URLs only differ by their

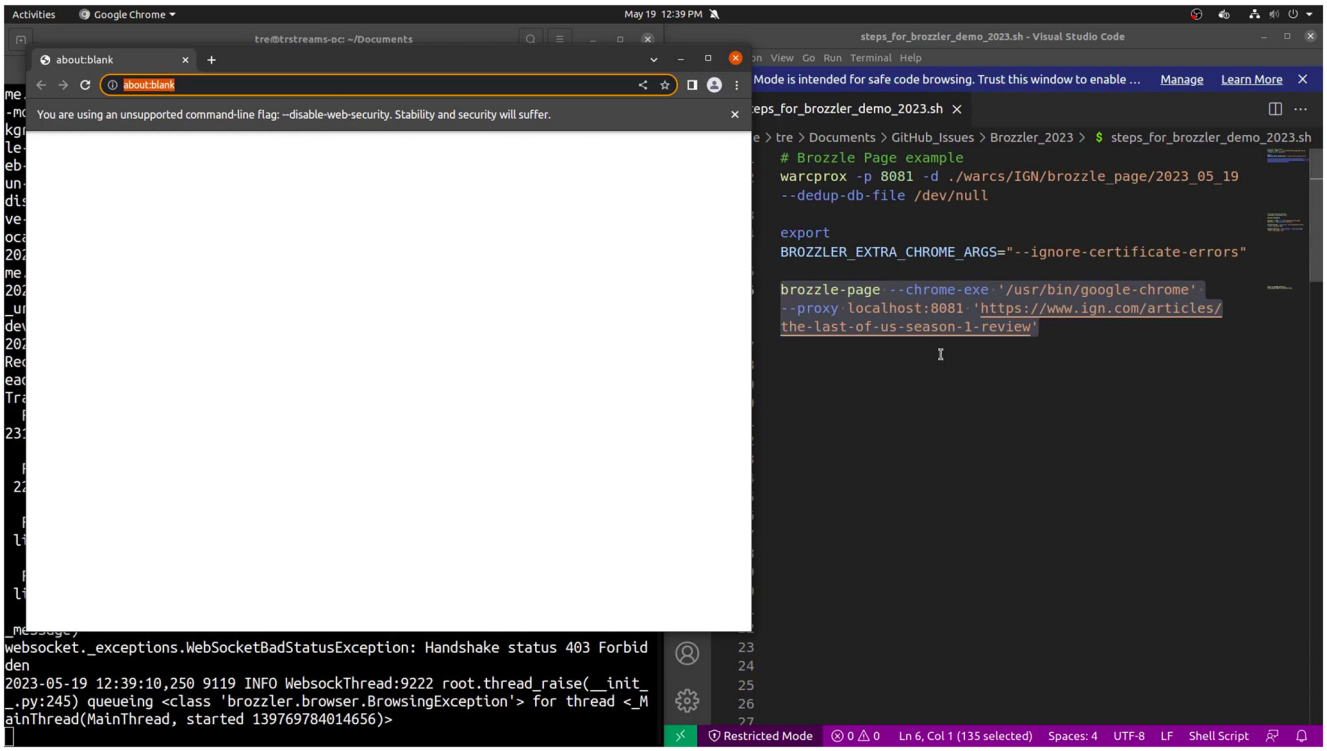


FIGURE 17 The web page does not load when Brozzler is archiving a web page with a Chrome version released after version 110.

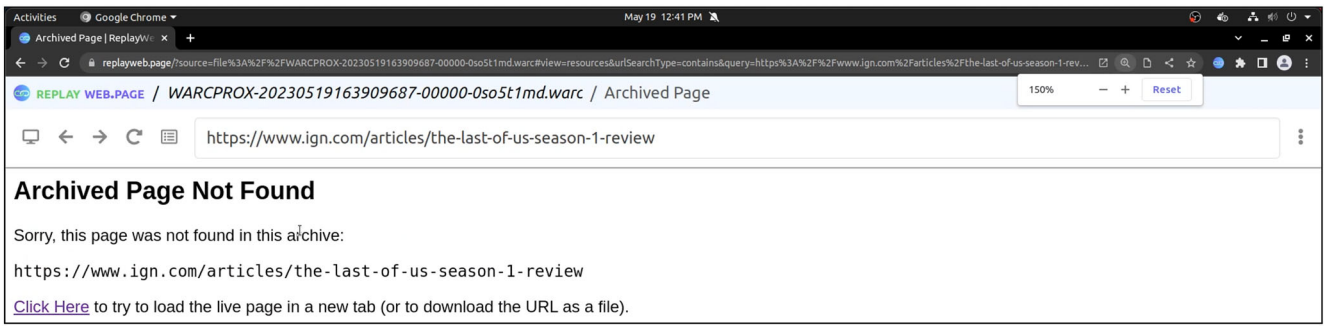


FIGURE 18 Brozzler failed to archive a web page with Chrome version 113.0.5672.126.

query string, then the same ad will be selected and replayed when loading an unarchived ad iframe URL.

5.3.3 | Loading embedded web page ads outside of an ad iframe

The JavaScript for Flashtalking's ad service also dynamically generated a URL that prevented ad replay. This prevented us from replaying the embedded web page ads that we archived during 2023 outside the containing web page because the JavaScript for Flashtalking's ad service loaded an unarchived web resource.

Figure 24 shows an example of an embedded web page ad outside of its ad iframe. The error message shown in Figure 24 is associated with an incorrect resource being loaded that prevents the ad from being replayed. The URI-R https://cdn.flashtalking.com/richLoads/300x600_Master_Richload/index.html is not associated with the current ad. The correct URI-R (includes ad id 173,980) that should have been loaded is https://cdn.flashtalking.com/173980/300x600_Master_Richload_Compressed/index.html. When replaying the embedded web page ad in an Amazon ad iframe, the Richload URI includes the ad id. This enables replay of the other ad resources (Figure 25). However, if we try to access this ad outside of the ad iframe on the live web, the web page will use an incorrect Richload URI and will not load it[22] (Figure 26).

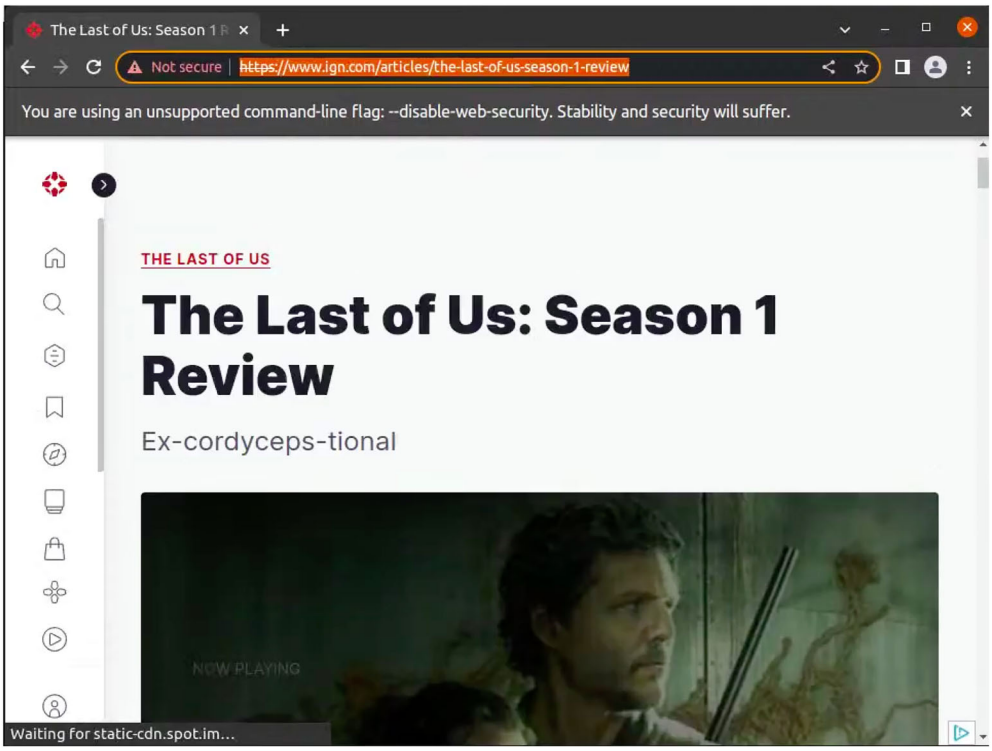


FIGURE 19 Brozzler could load web pages during crawl time with versions of Chrome released at the beginning of 2023.

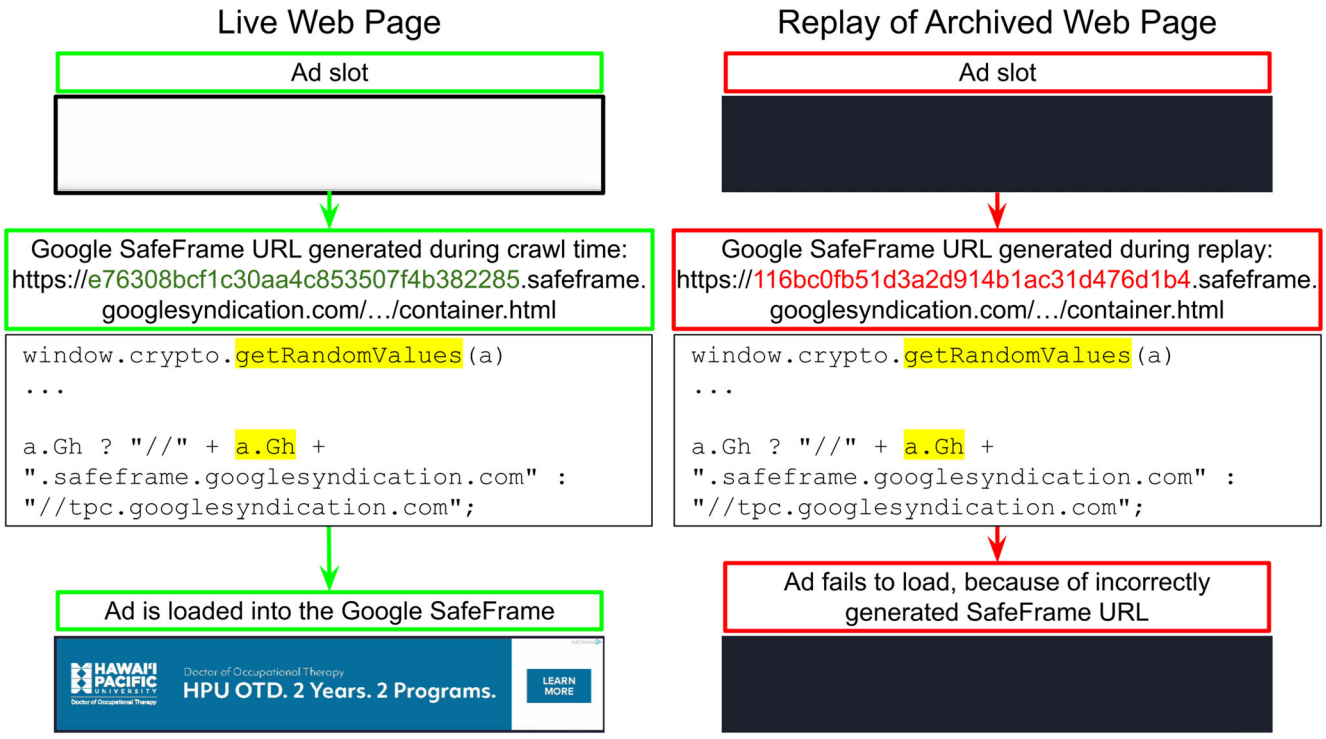


FIGURE 20 Different SafeFrame URLs during crawl and replay sessions.

5.3.4 | Replay of an ad can differ depending on the web browser used

Finally, we identified a replay problem with a replay system (ReplayWeb.page) that used service workers. In January 2023, we archived and replayed a web page[23] that included an ad (Figure 27) whose successful replay depended upon the browser used.[24] When replay systems use service workers, the replay of an archived web page can differ depending upon a browser's implementation of

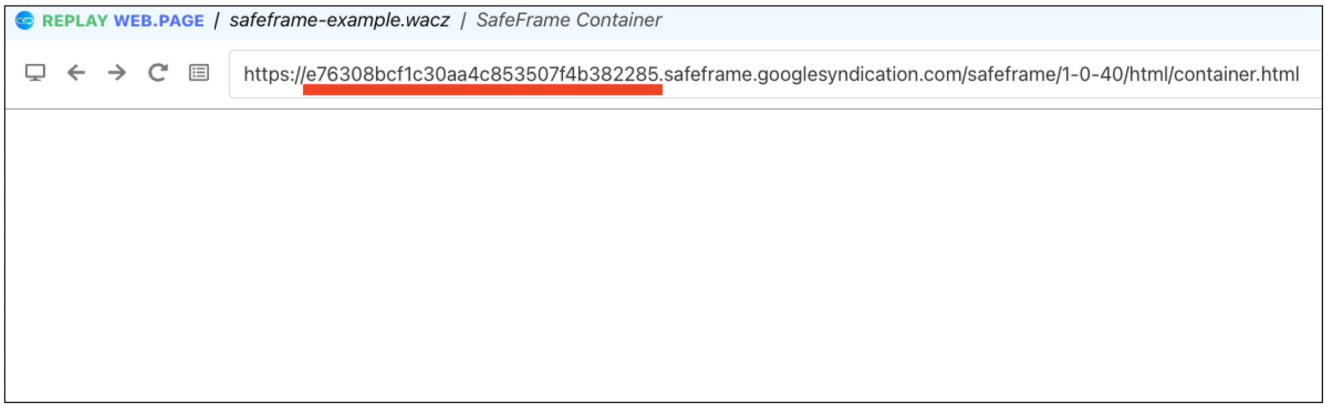


FIGURE 21 Google SafeFrame successfully archived by ArchiveWeb.page.

TABLE 3 When loading the Google SafeFrame for an ad, the random value in the SafeFrame URL differed each time the archived web page was replayed.

Replay session number	Random value in Google SafeFrame URL
1	af393d3d232450caab92d97eaefb484e
2	36dc52191b8e81186b187c938af4b280
3	5f68c90c97e25bf663f52ef786eb49b8
4	f663f52ef786eb49b8d803369fd0abea
5	6d18ef14a03123734d453dee25a8be6e
6	7a9317739c43b98082f4e77ed17fe3fa
7	4f8332dc6d18ef14a03123734d453dee
8	5bf663f52ef786eb49b8d803369fd0ab
9	227cd10c62e4b3ea26c2bd42e587e2c5
10	173e9e9bad424f8332dc6d18ef14a031

service workers. We observed this problem when an ad used an iframe with an `src` attribute value of "`about:blank`."

When we used ReplayWeb.page with Firefox version 109.0 (Mozilla, 2023), the image ad loaded (left side of Figure 27). Conversely, it failed to load when using Chrome version 109.0.5414 (Bommana, 2023) (right side of Figure 27). After identifying this problem, we created a GitHub issue (Reid & Kreymer, 2023) on ReplayWeb.page's GitHub repository. One of the comments[25] mentioned that the service worker had not gained control of the ad iframe, which led to leaked requests. These leaked requests resulted in a 404 status code during replay for a successfully archived resource. There is a Chromium bug[26] related to this issue, where the service worker is unable to access the resources loaded in an "`about:blank`" iframe. However, a ReplayWeb.page update fixed this by overriding the `document.write()` method (Mozilla, 2024) with a blob URL[27] created by the service worker.

6 | DISCUSSION

In this section, we will compare our Display Archived Ads tool, dataset, and key findings with existing works, highlighting similarities and differences. We will also discuss how others can leverage our tool and dataset, and how resolving the problems we identified with existing web archiving tools will impact the user experience. Finally, we will discuss our recommendations for archiving web ads and then outline future research directions.

6.1 | Utilizing display archived ads tool to identify ads

The primary purpose for our Display Archived Ads tool is to help identify archived ads within a WARC file. Since this tool is still a prototype, each use case currently involves a user who is familiar with running a Python script from a command-line interface.

A general use case for this tool is to view HTML, image, video, and audio web resources from a WARC file without the need to manually enter each URL into a replay system. For this use case we can also view a side-by-side comparison of the archived and live versions of a resource, eliminating the need to navigate to the URLs in separate browser windows. This feature is useful for users who want to compare recently archived web resources with their live versions.

Related tools such as ReplayWeb.page's search feature,[28] warcio,[29] warctools,[30] and jwarc[31] can be used to extract ad URLs from a WARC file and, in the case of ReplayWeb.page's search feature and jwarc, to replay these archived resources. Unlike these tools that either only provide a list of ad URLs or require a user to replay each potential ad resource one at a time, our Display Archived Ads tool shows a list of the live and archived versions of web resources side-by-side, which helps with visually determining the ad resources and with expediting the review process.

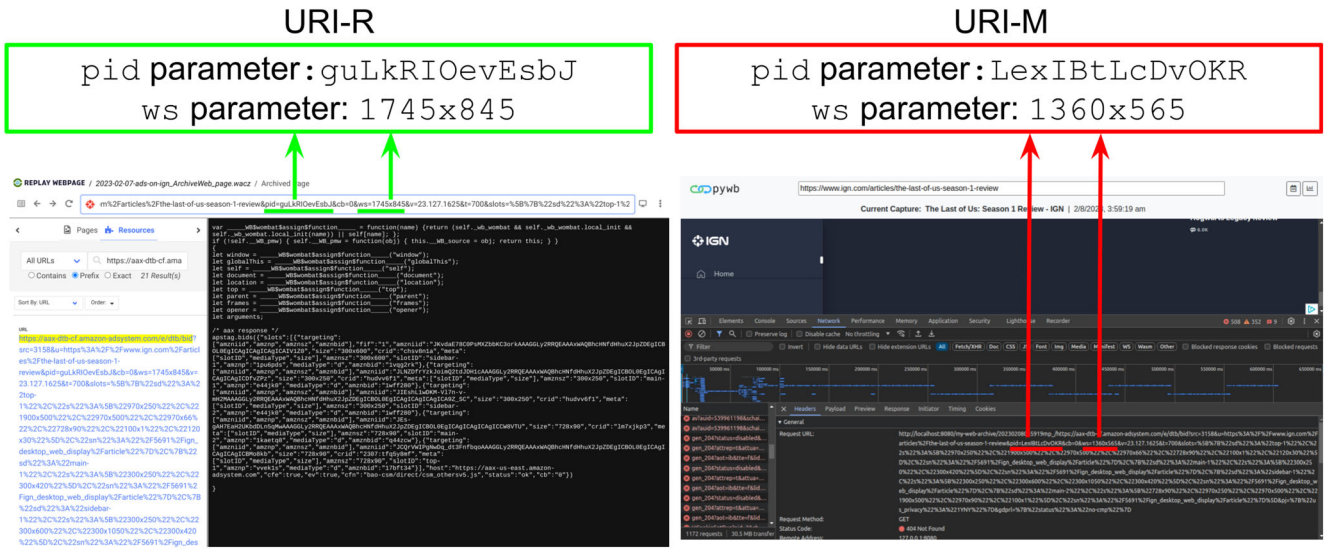


FIGURE 22 Pywb 2.7.3 failed to load Amazon ads because an ad bid URL failed to load.

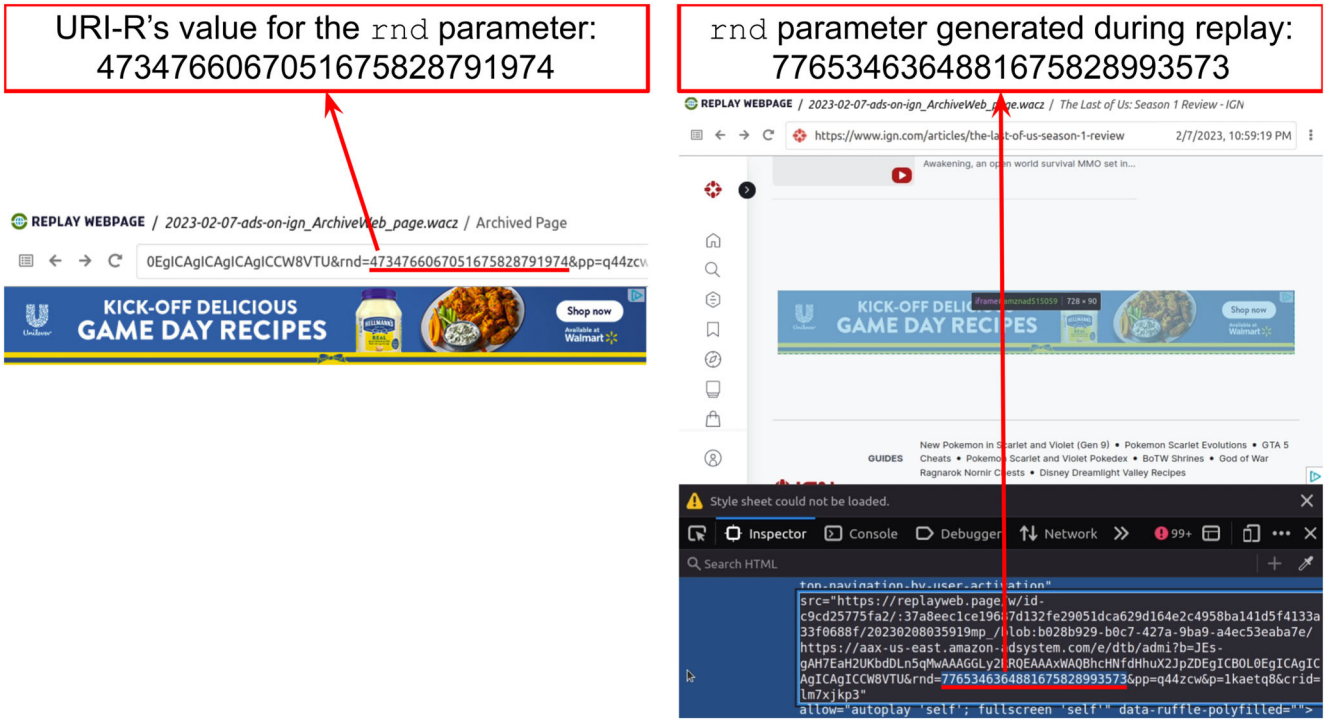


FIGURE 23 When replaying an Amazon ad iframe, the rnd parameter is not the same as the original value that is in the URI-R. Even though an incorrect URI-M is generated, ReplayWeb.page is able to load the ad. URI-M, URI for a memento; URI-R, URI of an Original Resource.

6.2 | Browsing ads from our dataset

Our dataset of 279 unique web ads can be accessed in two ways. First, we can view the CSV file,[32] which contains the details associated with the ads and the tools used to archive and replay the ads. Second, we can explore this dataset through our web page[33] (Figure 28), which allows us to view information from our dataset and replay archived web ads. This web page simplifies the process for viewing ads that we archived with ArchiveWeb.page, Browsertrix Crawler, and Brozzler, because it eliminates the need to download the WARC or WACZ file and then load it with ReplayWeb.page.

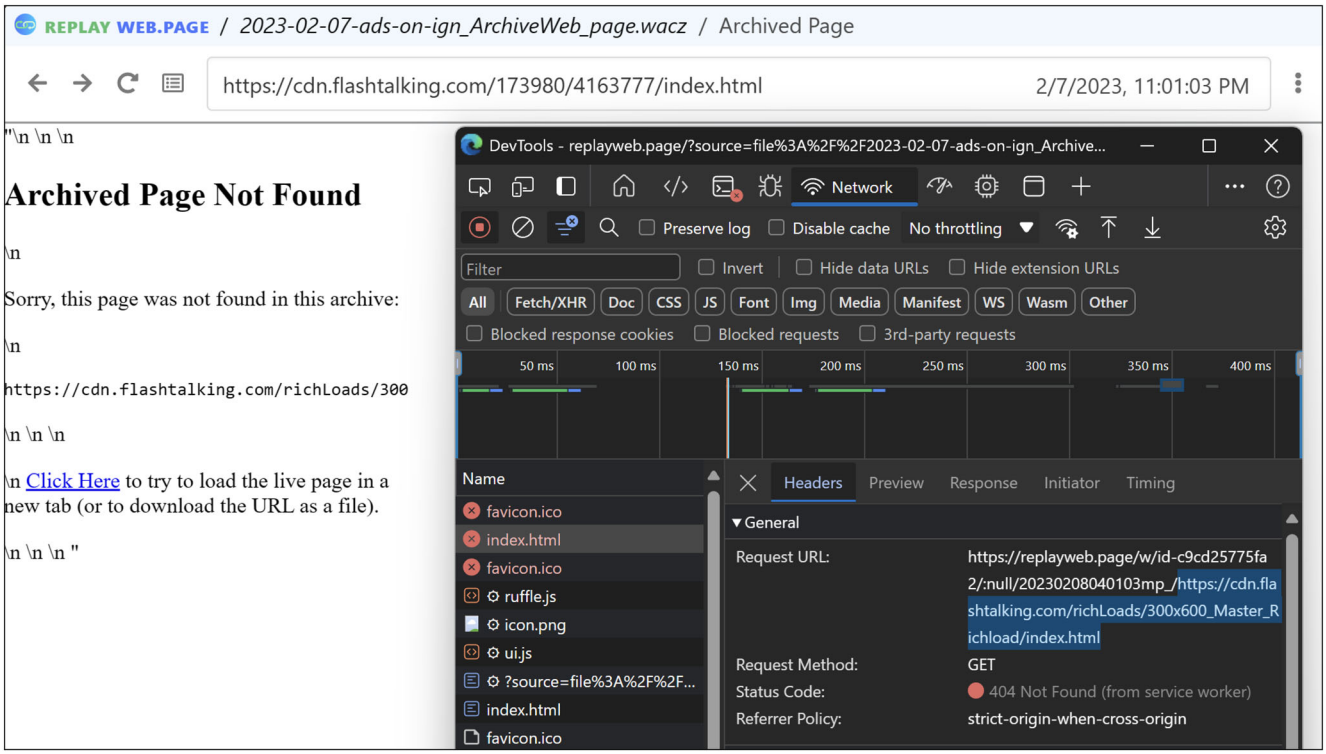


FIGURE 24 Replaying a successfully archived embedded web page ad outside of its ad iframe. This ad uses Flashtalking and Amazon ad services.

There are three main cases where researchers can use ads from our dataset as one of the sources for their research. The first use case is for researchers interested in studying recent web advertising from the 2020s, particularly from 2023. A second more specific use case would be researchers who want to explore how advertisers targeted the average web user. This would be applicable for ads that we archived with ArchiveWeb.page, Browsertrix Crawler, and Brozzler as these were captured locally using our own web browsers. Lastly, researchers who are specifically searching for web ads that were personalized based on location (19 ads) or related to Shopping, Finance, Vehicle and Automotive, and Business Services, since each of these themes has 20 or more ads in our dataset.

Unlike the research projects described in Section 3.4 (Burgess et al., 2022; New York University Cybersecurity for Democracy, 2025; Yoshikawa & Roesner, 2025; Zeng et al., 2021), our dataset includes ads from both social media (23 Facebook ads and 1 from YouTube), and from other web pages. NYU's Ad Observatory project (New York University Cybersecurity for Democracy, 2025) and The Australian Ad Observatory (Angus et al., 2024; Burgess et al., 2022) focused on social media ads. Neither Zeng et al. (2021) nor Yoshikawa and Roesner (2025) crawled social media web pages when creating their ad dataset. Also departing from these projects, our dataset went beyond solely political ads.

Since our ad dataset web page displays information about the ads we archived, it is similar to other digital ad collections. Beard (2018) identified 179 existing ad archives and collections, but only 9 were categorized as digital. Four of these[34] no longer exist. Upon reviewing the five remaining digital ad collections (Ads of the World,[35] AdForum's Creative Library,[36] Advertising Age's [AdAge] Creativity Collection,[37] The Ad Council's Our Campaigns Collection,[38] and Best Ads ontv.com[39]) and two additional collections (History of Advertising Trust [HAT] Library[40] and AdRespect's Ad Library[41]) not initially categorized as digital, we found that all seven contained web advertisements.

Similar to ours, all seven ad collections contained recent ads from the 2020s. For the ad description web page, three of the ad collections (HAT Library, AdForum, and The Ad Council) only showed lists of attributes associated with each ad, similar to our ad collection. The four other ad collections (AdRespect, Ads of the World, AdAge, and Best Ads ontv.com) included an ad story or summary that described what the ad was about and other contextual information associated with the ad campaign. One difference with our ad collection and two of the digital ad collections (AdForum and AdAge) was that it

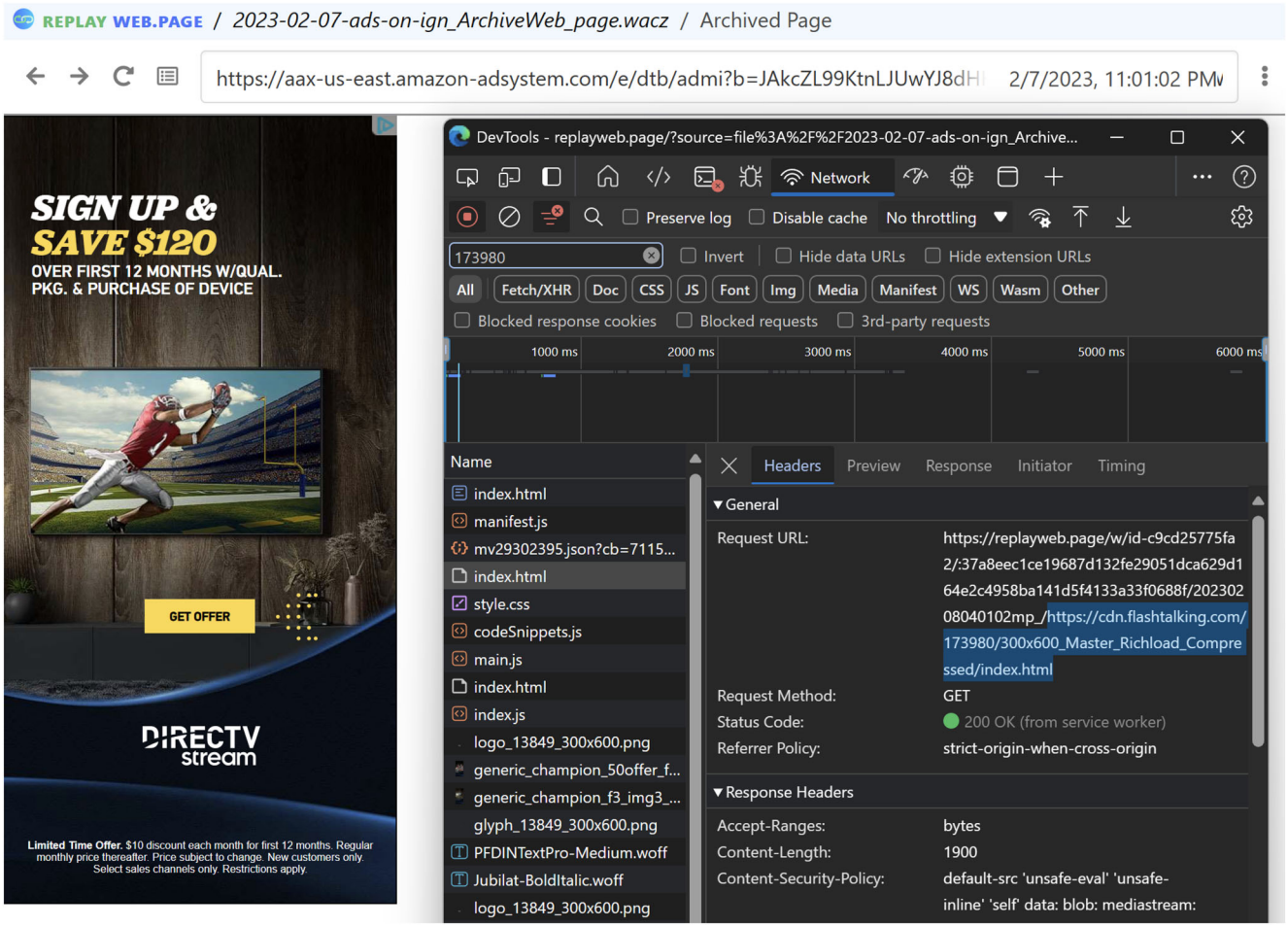


FIGURE 25 This embedded web page ad will successfully replay when it is loaded inside of an Amazon ad iframe. The correct Richload URI will be loaded when the ad is in the iframe. This ad uses Flashtalking and Amazon ad services.

required either a subscription or a user-created account to view the ad description.

Beyond these collections, recent research analyzed web ads in terms of their modality. Hussain et al. (2017) created two large datasets, one comprising more than 64,000 image ads and the other containing 3400 video ads with topic and sentiment annotations. Like these scholars, we also found that image ads outnumbered video ads. Our dataset has 80 image ads and 15 video ads.

6.3 | Problems with archiving and replaying web ads

In creating a dataset of 279 unique archived web ads, we identified 5 key problems with archiving and replaying them. First, Internet Archive's Save Page Now service blocked ads from being archived. Excluding these ads from being archived impacts archivability (based on Kelly et al.'s (2013) metrics) for websites that load ads since some of the embedded resources (ads) on the web page were not archived. According to Reyes Ayala (2022) metrics, the replay quality of these websites will be negatively impacted due to reduced completeness (the archived version will have fewer web resources than the live version), diminished visual correspondence (empty ad slots), and less interactional correspondence (cannot interact with the ads that should have loaded). This finding speaks to Ogden et al. (2024), who described Save Page Now as a "series of black boxes containing their own internal logics, technical affordances and hidden relations that shape the nature of web archive records." The Internet Archive's approach for blocking ad URLs has not been publicly disclosed, and this lack of transparency aligns with concerns raised by Maemura et al. (2018). Not having this documentation made it unclear why most ads were not being archived. After contacting Wayback Machine staff in August 2023, an update was implemented that enabled users to archive ads by directly archiving the ad's URL. However, in April 2025, Save Page Now's default behavior blocked ad URLs, but logged-in users

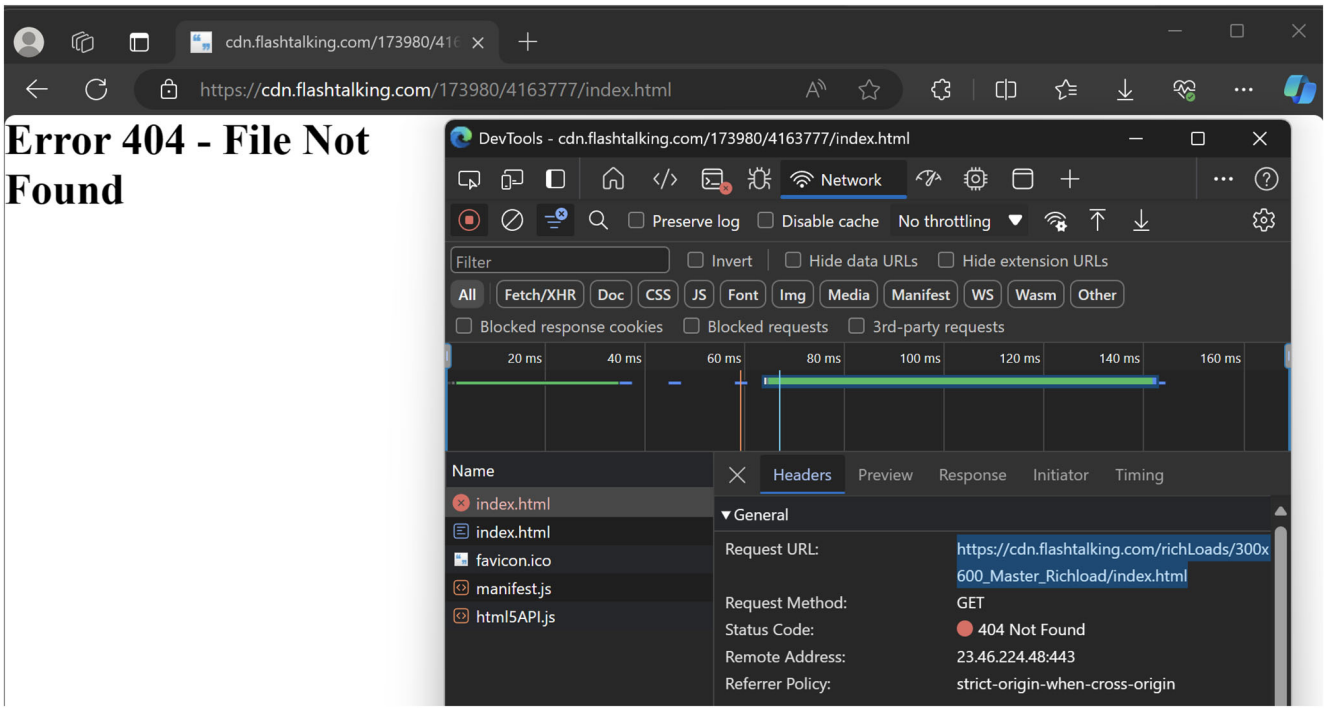


FIGURE 26 The live web version of this embedded web page ad also fails to load outside of the ad iframe.

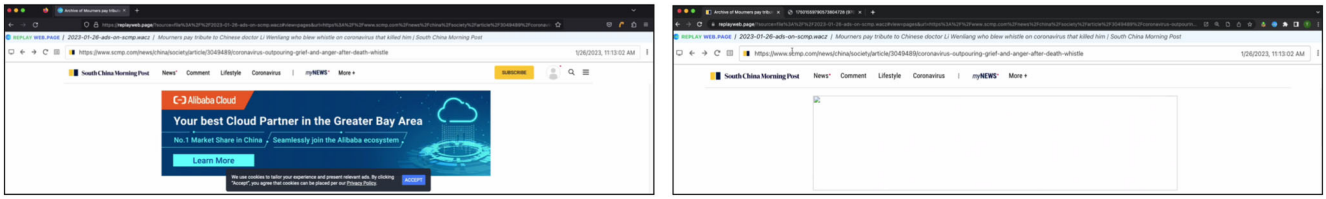


FIGURE 27 The replay of an ad differed depending on the web browser used.

can archive them by selecting the new "Disable ad blocker" option.

Second, Brozzler was incompatible with versions of Chrome released after February 2023. Brozzler's incompatibility problem that we reported[42] was resolved during the preparation of this paper.[43] Brozzler is now compatible with Chrome version 130 and later. This enables users to archive ads and other web resources using the latest browser versions.

Third, the JavaScript from Google and Amazon ad services prevented ads from being replayed successfully since replay systems do not generate the same random value that was generated during the crawling session, which leads to requesting unarchived URLs. Kiesel, de Vries, et al. (2018) also found that unarchived resources are requested when an archived web page executes JavaScript that includes randomly generated values in a URL.

Fourth, Flashtalking ads that were replayed outside of the ad iframe requested a URL that did not exist, thereby preventing the ad's web resources from loading. Updating the fuzzy matching rules used by replay systems might resolve the replay problems with Google, Amazon, and Flashtalking ad services. URL matching algorithms (fuzzy matching and querystrip) could help address the broader problem of dynamically generated, non-deterministic URLs (Goel et al., 2022). When performing a URL match for Google and Amazon ad iframes, the random value should be removed. Google SafeFrame URLs include the random value in the subdomain and Amazon ad iframe URLs include the random value in the query string. For Flashtalking ads that request a non-existent URL that includes "`Richload`" in the path, it needs to be matched with a different URL that includes "`Richload`" and the ad ID in the URL's path. The ad ID can be retrieved from the URL for the web page ad that initiated the request.

Fifth, Chrome's service worker implementation prevented ReplayWeb.page from accessing resources inside of an "`about:blank`" iframe. We reported this problem on ReplayWeb.page's GitHub and a workaround was

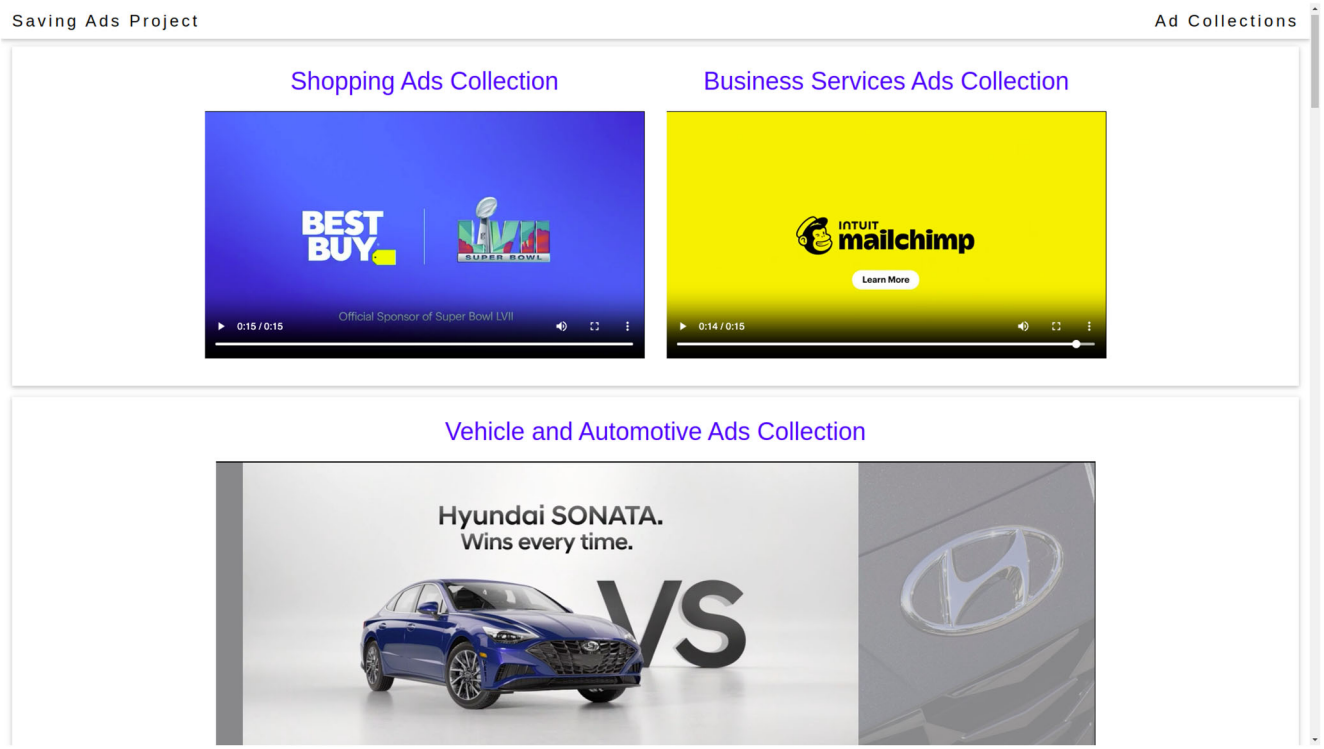


FIGURE 28 Our web page for displaying all the information from the dataset including the replay of the archived ads.

created that used blob URLs for iframes that have "about:blank" as their src attribute. This update[44] ensures that ads and other web resources loaded inside an "about:blank" iframe will be replayed successfully. Alam et al. (2017) used service workers for a replay system, but they did not identify a case where having a different service worker implementation (using a different web browser) would result in problems with replaying a web resource.

6.4 | Recommendations for archiving web ads

When archiving ads, we recommend using ArchiveWeb.page to capture[45] the ad resources and ReplayWeb.page for replay. ArchiveWeb.page is able to archive ads from all of our ad categories (text, image, video, embedded web page, and combination). Additionally, ArchiveWeb.page allows users to interact with the web page during the crawling session, enabling the capture of more user interactions. However, because the browser extension version of ArchiveWeb.page requires the user to manually initiate the crawling session, it is most suitable for archiving a small number of ads. We also recommend ReplayWeb.page because it can replay some complex ads, such as Amazon ads that use an ad iframe, where other replay systems have failed.

For users seeking to employ a web archiving service to capture and replay the ad resources, we recommend using Save Page Now (with "Disable ad blocker" option enabled) and Arquivo.pt. for all ad types, Conifer for most[46] ad types (text, image, video, and embedded web page) and archive.today for text and image ads. When we retested Save Page Now with this new "Disable ad blocker" option while archiving our demo web page,[47] which loads ad resources, we successfully archived and replayed image, video, and embedded web page ads. Arquivo.pt.[48] also archived all ads from our demo web page, but its limitation is that we had to wait a few days for the replay of ad resources to become available. One advantage of Conifer when archiving a web page that loads ads[49] is that it allows users to download the WARC file that can then be used with ReplayWeb.page's search feature or our Display Archived Ads tool to identify the ad resources. One limitation of using Conifer is the free storage limit of 5 GB. Archive.today is only recommended for text and image ads,[50] because it does not preserve videos[51] and the original web page's JavaScript.[52]

Users who want to automate archiving numerous web ads can use either Browsertrix Crawler or Brozzler to capture[53] the ads and then replay with ReplayWeb.page. Both tools require the user to execute from the command line or to use a web archiving service, such as Browsertrix[54] or Archive-It.[55] Browsertrix Crawler and Brozzler are capable of capturing ads from all ad

categories. Although our recommendations should facilitate the process of capturing and replaying web ads, further work is necessary to improve the replay of more complex ads.

Some web archiving services like Save Page Now, Arquivo.pt., and archive.today do not allow users to archive personalized ads. These services either block ads by default or lack the necessary browser feature for using the user's web browser during a crawl. However, Conifer and Browsertrix permit the user either to use their web browser or to create a browsing profile that enables the archiving of personalized ads.

For browser-based crawlers, archiving personalized ads will not always work by default. Browser extensions allow for archiving personalized ads, but if the tool executes from the command line, it will require extra steps[56] which can be another barrier for users. If users capture and archive the personalized web ads, the collections can provide a more in-depth representation of the advertising ecosystem and targeting strategies. Such captures would allow not only researchers but also a broader range of stakeholders to understand how web ads vary across demographics, interests, or browsing histories of individuals in different contexts with different web experiences.

6.5 | Future work

For future work, two replay problems identified in this study need to be fixed. These involve matching the dynamically generated URLs with the URLs that were successfully crawled. One problem involves a random value being used in either the query string or subdomain for a URL. Removing the random value when performing a URL match could resolve this problem, but how would unexpected behaviors be mitigated, such as the case with Amazon ads where multiple ads were using the same base URI for the ad iframe, which prevented the loading of other ads? The other replay problem that should be fixed is when a Flashtalking ad dynamically loads an incorrect URL when it is not loaded in an ad iframe. This problem involves an embedded web page that receives information from the containing web page before it determines which resource to load. If the incorrect URL generated during replay is matched with the crawled URL, then it will resolve this problem, but can this approach be generalized and applied to other similar cases where the embedded web page communicates with the containing web page? In addition to addressing these two problems, we suggest developing web archiving tools to assist with automating the creation of large archived web ad datasets. We also recommend performing a study of older archived web ads. Finally, the archiving of personalized web advertisements presents an important avenue for research.

Our dataset consists of 279 web ads that were manually identified by viewing web resources from a WARC or WACZ file. To create a larger dataset would require more automation. One approach would employ ad filter lists from ad blocking tools to identify the web resources associated with well-known ad services like Google Ads, Amazon Ads, Outbrain, and Flashtalking. The challenge with this approach would be correctly categorizing the ads (image, video, embedded web page, or combination), and replaying ads that are embedded web page or combination ads because they use multiple web resources and in some cases cannot be replayed outside of their containing web page or ad iframe. Even if successfully automated, however, this process would not be comprehensive, since ads *not* loaded through a well-known ad service would be excluded. Another approach involves modifying existing ad scraping tools, such as adscraper,[57] NYU's Ad Observer (New York University Cybersecurity for Democracy, 2025), and the Australian Ad Observatory's tool (Angus et al., 2024; Burgess et al., 2022), for use with browser-based web archive crawlers. This would enable the extraction of ad URIs and metadata, as well as the generation of ad screenshots during the crawling session.

In the future, we hope to draw upon public web archives to create a dataset of archived web ads from web pages spanning from 1996 to the present. This dataset would help assess the relative presence or absence of advertisements, which has yet to be quantified. Moreover, conducting this analysis would reveal the archiving and replay problems that prevent ads from loading during replay on older archived web pages. For example, traditional non-browser-based web archive crawlers like Heritrix will not execute JavaScript during a crawl, so dynamically loaded ads would not be preserved for older archived web pages. Other problems that we have identified, such as random values preventing ads from loading and ads that must be loaded in an iframe, would apply to archived web pages that execute JavaScript. Still another problem with performing such an analysis is that some web archiving services, like Save Page Now, previously blocked ads from being archived, which would result in temporal gaps,[58] with a significant portion of ads missing.

There are also questions specific to archiving personalized ads. To what extent do institutional web archives capture personalized advertisements on the web? Do scholarly or lay web users prefer (re)using archived web pages including personalized ads, a generic comprehensive capture, or a capture with web ads missing? In what ways might the strategic use of diverse personas surface

types of web content that would otherwise go unarchived?

7 | CONCLUSION

Web advertisements represent a significant and rapidly evolving aspect of digital cultural heritage. The need to preserve them is ever-increasing. But serious problems with archiving and replaying current web ads persist. Ads change rapidly and in real time based on each user's unique profile, making them difficult to crawl, much less to preserve, comprehensively.

This paper explored the process of creating a dataset of 279 recent (January to June, 2023) web ads and discussed the problems we encountered while archiving and replaying these ads. This dataset was created by archiving 17 web pages from SimilarWeb's top websites worldwide. When archiving these web pages, we utilized four web archiving services (Internet Archive's Save Page Now, Arquivo.pt., archive.today, and Conifer) and three browser-based tools (ArchiveWeb.page, Browsertrix Crawler, and Brozzler). We replayed these archived web pages with four web archiving services (Internet Archive's Wayback Machine, Arquivo.pt., archive.today, and Conifer) and three other replay systems (ReplayWeb.page, pywb, and OpenWayback).

The process of archiving and replaying these 279 unique web ads yielded 5 key findings. First, Internet Archive's Save Page Now feature excluded web ads from being archived. Second, Brozzler was incompatible with recent versions of Google Chrome released after March 2023, which prevented web pages from loading during the crawl. Third, when executing Google's and Amazon's ad script, the random values generated are not the same during the crawling and replay sessions, because the seed for the random number generator is set by the replay system and the value will not be the same as during crawl time. This resulted in a request for an incorrect URL during replay that was not archived and prevented the ad from loading. Fourth, the JavaScript for Flashtalking's ad service prevented the replay of embedded web page ads outside of an ad iframe, because the ad script dynamically generated an incorrect URL that does not exist on the live web. Fifth, some web ads were not loaded during replay depending on the web browser used, because the service worker implementation can differ between browsers. Chromium had a bug that prevented service workers from accessing resources inside an iframe with the `src` attribute of "`about:blank`." This resulted in leaked requests, which prevented the replay of a successfully archived ad. Complementing these findings, we created the Display Archived Ads tool to help find advertisements that were not able to replay when loading the containing web page.

Once the problems we encountered with Google and Amazon ads are fixed, the replay quality will be improved for ads and other dynamically loaded embedded resources that use random values. By identifying and reporting problems with Brozzler and ReplayWeb.page that are now resolved, our work has helped enhance the experience for users archiving and replaying dynamic web resources, such as web ads.

ACKNOWLEDGMENTS

Thanks to the Internet Archive and Webrecorder staff for resolving the archiving and replay problems that we reported to them. This research was made possible through the support of the Institute of Museum and Library Services (IMLS).

FUNDING INFORMATION

This research was supported by the Institute of Museum and Library Services (IMLS), Grant/Award Numbers: LG-252362-OLS-22 and LG-256695-OLS-24.

DATA AVAILABILITY STATEMENT

The dataset (Reid, 2024b) that support the findings of this study is openly available on GitHub at https://github.com/savingads/Recently_Archived_Ads/blob/main/All_Ads.csv.

ORCID

Travis Reid https://orcid.org/0000-0003-1360-7963
Alex H. Poole https://orcid.org/0000-0001-7738-8929
Hyung Wook Choi https://orcid.org/0000-0002-4075-0768
Mat Kelly https://orcid.org/0000-0002-0236-7389
Michele C. Weigle https://orcid.org/0000-0002-2787-7166

ENDNOTES

[1] Other terms denoting ad space include ad unit (Google, 2024), ad slot (Google, 2020b), and ad inventory (Google, 2019a).

[2] Gpt.js: https://securepubads.g.doubleclick.net/tag/js/gpt.js.

[3] Archived version of pubads_impl.js: https://web.archive.org/web/20230821142108id_/https://securepubads.g.doubleclick.net/pagead/managed/js/gpt/m202308210101/pubads_impl.js?cb=31077272.

[4] Brozzler: https://github.com/internetarchive/brozzler.

[5] The "Adult" category was excluded because we planned on recording and uploading videos of the crawling and replay sessions to YouTube. Websites from this category cannot be included in a video, as it would violate YouTube's Community Guidelines (https://support.google.com/youtube/answer/2802002?hl=en&ref_topic=9282679).

[6] When replaying a web page, https://canalturf.com, archived by Save Page Now, we found that some of the ads that were loaded by Wayback Machine were archived by ArchiveBot and Perma.cc.

[7] This problem with Flashtalking ads is discussed in Section 5.3.3.

[8] Ad URI with e parameter: https://s0.2mdn.net/sadbundle/1407324127475269632 0/970x250-HBO_Max/index.html?e=69&leftOffset=0&topOffset=0&c=qipO0hxSN3&t=1&renderingType=2&ev=01_247.

[9] Ad URI without e parameter: https://s0.2mdn.net/sadbundle/14073241274752696320/970x250-HBO_Max/index.html?leftOffset=0&topOffset=0&c=qipO0hxSN3&t=1&renderingType=2&ev=01_247.

[10] Demo video for our tool: https://youtu.be/Bc2T6ZZd210?t=30.

[11] We replayed the archived resources in an iframe in order to display multiple ads on the same web page.

[12] An example filtered file is pixel.gif, a commonly used image for ad services that only shows a few white pixels.

[13] Brunelle (2012) and Lerner et al. (2017) discuss this problem of live web resources being loaded during replay.

[14] We provide a online table (recently_archived_ads_paper_caption_URLs.md) to include the URLs (including URI-M, WACZ, WARC, and replay video, as applicable) for the web resources shown in our figures.

[15] Prefix search URL for some Google ads: https://web.archive.org/web/*/https://s0.2mdn.net/*.

[16] Prefix search URL for Innovid ads: https://web.archive.org/web/*/https://s-static.innovid.com/*.

[17] Prefix search URL for Flashtalking ads: https://web.archive.org/web/*/https://cdn.flashtalking.com*.

[18] GitHub issue associated with Brozzler's incompatibility: https://github.com/internetarchive/brozzler/issues/256.

[19] Chrome version 130: https://developer.chrome.com/release-notes/130.

[20] Aturban (2023) discussed this problem of inconsistent replay of archived web pages.

[21] Our test web page is described in section 4.2.1 of our technical report (Reid et al., 2025).

[22] Flashtalking might have done this to block ads from loading on a website with which the advertisers do not want to be associated.

[23] Example web page that loaded an ad that would have different replay depending on the browser: https://www.scmp.com/news/china/society/article/3049489/coronavirus-outpouring-grief-and-anger-after-death-whistle/.

[24] Video: https://youtu.be/gCW15i-5teQ?t=40.

[25] GitHub issue: https://github.com/webrecorder/replayweb.page/issues/157.

[26] Chromium bug: https://issues.chromium.org/issues/41411856.

[27] A blob URL is another way to access a File object and it can be used as a src or href attribute (Bidelman, 2011).

[28] ReplayWeb.page search feature: https://replayweb.page/docs/exploring.

[29] Warcio: https://github.com/webrecorder/warcio.

[30] Warctools: https://github.com/internetarchive/warctools.

[31] Jwarc: https://github.com/iipc/jwarc.

[32] Dataset CSV file: https://github.com/savingads/Recently_Archived_Ads/blob/main/All_Ads.csv.

[33] Ad dataset web page: https://savingads.github.io/themed_ad_collections.html.

[34] Ad collections that no longer exist: coloribus.com, www.advertolog.com, CreativeClub.com, and adverlicio.us.

[35] Ads of the World collection: https://www.adsoftheworld.com/collections.

[36] AdForum's Creative Library: https://www.adforum.com/creative-work.

[37] AdAge Creativity Collection: https://adage.com/creativity/search.

[38] The Ad Council's Collection: https://www.adcouncil.org/all-campaigns.

[39] Best Ads ontv.com: https://www.bestadsontv.com/.

[40] HAT Library: https://www.hatads.org.uk/.

[41] AdRespect's Ad Library: https://www.adrespect.org/common/11064/view-page.cfm?clientID=11064.

[42] Brozzler's incompatibility problem that we reported: https://github.com/internetarchive/brozzler/issues/256.

[43] We did not retest the updated version of Brozzler with the web pages we archived in 2023, because the same set of ads would not load during a new crawling session.

[44] The update to ReplayWeb.page was completed before we finalized our dataset and results.

[45] ArchiveWeb.page can be used to create local web archive collections (with the browser extension) or public collections (with the Browsertrix service) that are accessible to everyone.

[46] We do not recommend Conifer for replaying combination ads, because currently Conifer is incapable of replaying the containing web pages from our dataset (diy.com's and sports.yahoo.com's archived web pages) and showed error messages associated with incorrectly generated URLs.

[47] Demo web page archived by Save Page Now: https://web.archive.org/web/20250418062304/https://treid003.github.io/blocked_Ad_URL_example.html.

[48] Demo web page archived by Arquivo.pt.: https://arquivo.pt/wayback/20250418070608/https://treid003.github.io/blocked_Ad_URL_example.html.

[49] Conifer successfully archived and replayed image, video, and embedded web page ads on our demo web page: https://conifer.rhizome.org/treid003/archiving-demo-web-page-with-ads-2025-04/20250418071519/https://treid003.github.io/blocked_Ad_URL_example.html.

[50] Archive.today successfully archived an image ad on our demo web page: https://archive.ph/5hXNZ.

[51] Archive.today excludes videos: https://wiki.archiveteam.org/index.php/Archive.today#Content_type.

[52] Berlin (2018) and Berlin et al. (2023) use archive.today to exemplify Archival Caricaturization replay, discussing how excluding the original JavaScript impacts replay.

[53] Browsertrix Crawler and Brozzler can be used to either create a local web archive collection when executing the tools from the command line or a public collection when using either Browsertrix service or Archive-It.

[54] Browsertrix: https://webrecorder.net/browsertrix/.

[55] Archive-It: https://archive-it.org/.

[56] The extra steps involve creating a new browsing profile that is not at the default user data directory and then adding a command line argument that uses this new user data directory.

[57] https://github.com/UWCSESecurityLab/adscraper.

[58] One example is when Save Page Now blocked ads from being archived around November 2021 through August 2023.

REFERENCES

Alam, S., Kelly, M., Weigle, M. C., & Nelson, M. L. (2017). Client-side reconstruction of composite mementos using serviceworker. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1–4). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/JCDL.2017.7991579

Amazon. (2024). *Bidding*. https://advertising.amazon.com/help/GPDRGYSKVB7H4BUR

Amazon Ads. (2024). *What is CPM?* https://advertising.amazon.com/library/guides/cost-per-mille

Angus, D., Obeid, A., Burgess, J., Parker, C., Andrejevic, M., Bagnara, J., Carah, N., Fordyce, R., Hayden, L., Lewis, K., O'Neill, C., Albarrárran-Torres, C., & Cellard, L. (2024). The Australian ad observatory technical and data report. *ARC Centre of Excellence for Automated Decision-Making and Society*. (pp. 7-43) https://eprints.qut.edu.au/245657/

Archibald, J., Kruisselbrink, M., Russell, A., & Song, J. (2022). Service Workers. *W3C*. https://www.w3.org/TR/2022/CRD-service-workers-20220712/

Aturban, M., Klein, M., Van de Sompel, H., Alam, S., Nelson, M. L., & Weigle, M. C. (2023). Hashes are not suitable to verify fixity of the public archived web. *PLoS One*, *18*, 1–49. https://doi.org/10.1371/journal.pone.0286879

Ball, A. (2010). *Web archiving (version 1.1)*. Digital Curation Centre. https://www.dcc.ac.uk/sites/default/files/documents/reports/sarwa-v1.1.pdf

Beard, F. (2018). Archiving the archives: The world's collections of historical advertisements and marketing ephemera. *Journal of Historical Research in Marketing*, *10*(1), 86–106. https://doi.org/10.1108/JHRM-08-2017-0044

Berlin, J. (2018). *To relive the web: A framework for the transformation and archival replay of web pages*. Master of Science (MS), Old Dominion University. https://doi.org/10.25777/n8mg-da06

Berlin, J., Kelly, M., Nelson, M. L., & Weigle, M. C. (2023). To re-experience the web: A framework for the transformation and replay of archived web pages. *ACM Transactions on the Web*, *17*(4), 1–49. https://doi.org/10.1145/3589206

Besser, H. (2017). Archiving websites containing streaming media. *Archiving Conference*, *14*(1), 11–13. https://doi.org/10.2352/issn.2168-3204.2017.1.0.11

Bhatt, R., Harshitha, P., Jha, K., Bhakthavathsalam, R., Gowranga, K., & Saqquaf, S. (2015). An automated programming tool for archiving and interfacing web contents from static and dynamic sites. *International Journal of Emerging Engineering Research and Technology*, *3*(8), 27–36. https://ijeert.ijrsset.org/v3-i8

Bidelman, E. (2011). *Using the HTML5 filesystem API*. O'Reilly Media, Inc.

Bommana, P. (2023). *Stable channel update for desktop*. https://chromereleases.googleblog.com/2023/01/stable-channel-update-for-desktop_24.html

Brunelle, J. F., Weigle, M. C., & Nelson, M. L. (2015). Archiving deferred representations using a two-tiered crawling approach. In *Proceedings of the 12th International Conference on Digital Preservation (iPRES)* (pp. 44–53). School of Information and Library Science, University of North Carolina at Chapel Hill. https://doi.org/11353/10.429524

Brunelle, J. F., Weigle, M. C., & Nelson, M. L. (2017). Archival crawlers and JavaScript: Discover more stuff but crawl more slowly. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1–10). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/JCDL.2017.7991554

Brunelle, J. F. (2012). Zombies in the archives. https://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html

Burgess, J., Andrejevic, M., Angus, D., & Obeid, A. (2022). *Australian ad observatory: Background paper*. ARC Centre of Excellence for Automated Decision-Making and Society. https://eprints.qut.edu.au/234461/, https://doi.org/10.25916/7bge-bp35

Cohen, D. J. (2010). *Digital ephemera and the calculus of importance*. http://dancohen.org/2010/05/17/digital-ephemera-and-the-calculus-of-importance/

Cohen, L. (2003). *A consumers' republic: The politics of mass consumption in postwar America (1. Vintage Books ed)*. Vintage Books.

Cook, C. (2018). Archiving web content. In B. H. Marshall (Ed.), *The complete guide to personal digital archiving* (pp. 33–55). ALA Editions.

Dulin, K., & Ziegler, A. (2017). Scaling up perma.cc: Ensuring the integrity of the digital scholarly record. *D-Lib Magazine*, *23*(5/6). https://doi.org/10.1045/may2017-dulin

Einstein, M. (2017). *Advertising: What everyone needs to know*. Oxford University Press.

Ewen, S. (2001). *Captains of consciousness: Advertising and the social roots of the consumer culture* (25th Anniversary ed.). Basic Books.

Goel, A., Zhu, J., Netravali, R., & Madhyastha, H. V. (2022). Jawa: Web archival in the era of JavaScript. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)* (pp. 805–820). USENIX Association. https://www.usenix.org/conference/osdi22/presentation/goel

Google. (2015). *How AdSense works*. https://support.google.com/adsense/answer/6242051?hl=en

Google. (2019b). *Cost-per-thousand impressions (CPM): Definition*. https://support.google.com/google-ads/answer/6310

Google. (2020a). *Delivery basics*. https://support.google.com/admanager/answer/9248464

Google. (2024). Add new ad units. https://support.google.com/admanager/topic/10478086

Google. (2019a). Advertising with Google Ad manager. https://support.google.com/admanager/answer/6022000.

Google. (2020b). Glossary. https://support.google.com/admanager/table/7636513

Google. (2020c). Render creatives using SafeFrame. https://support.google.com/admanager/answer/6023110

Graham, M. (2019). *The wayback machine's save page now is new and improved*. https://blog.archive.org/2019/10/23/the-wayback-machines-save-page-now-is-new-and-improved/

Hussain, Z., Zhang, M., Zhang, X., Ye, K., Thomas, C., Agha, Z., Ong, N., & Kovashka, A. (2017). Automatic understanding of image and video advertisements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1705–1715). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/CVPR.2017.123

Hwang, T. (2020). *Subprime attention crisis: Advertising and the time bomb at the heart of the Internet* (1st ed.). FSG Originals × Logic.

Interactive Advertising Bureau. (2016). *SafeFrame implementation guidelines*. https://iab.com/guidelines/safeframe/

International Internet Preservation Consortium. (2012). *OpenWayback*. https://github.com/iipc/openwayback

International Internet Preservation Consortium. (2015). *The WARC Format 1.1*. https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/

Kahle, B. (1997). Preserving the Internet. *Scientific American*, *276*(3), 82–83. https://www.scientificamerican.com/article/preserving-the-internet/, https://doi.org/10.1038/scientificamerican031997-4pGAwcm6xVH2Z3BjThPC6M

Kelly, M., Brunelle, J. F., Weigle, M. C., & Nelson, M. L. (2013). On the change in Archivability of websites over time. In *Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL)* (pp. 35–47). Springer. https://doi.org/10.1007/9783642405013_5

Kiesel, J., de Vries, A., Hagen, M., Stein, B., & Potthast, M. (2018). WASP: Web archiving and search personalized. In *Proceedings of the First Biennial Conference on Design of Experimental Search & Information Retrieval Systems* (pp. 16–21). Sun SITE Central Europe. https://desires.dei.unipd.it/2018/papers/paper4.pdf

Kiesel, J., Kneist, F., Alshomary, M., Stein, B., Hagen, M., & Potthast, M. (2018). Reproducible web corpora: Interactive archiving with automatic quality assessment. *Journal of Data and Information Quality*, *10*(4), 1–25. https://doi.org/10.1145/3239574

Kreymer, I. (2023). *Pywb 2.7.3 release*. https://github.com/webrecorder/pywb/releases/tag/v-2.7.3

Kreymer, I. (2024). *WARCIO: WARC (and ARC) streaming library*. https://github.com/webrecorder/warcio

Kreymer, I., & Summers, E. (2021). *Web archive collection zipped (WACZ)*. Webrecorder. https://specs.webrecorder.net/wacz/1.1.1/

Leach, W. (1994). *Land of desire: Merchants, power, and the rise of a new American culture* (1st ed.). Vintage books.

Lears, J. (1995). *Fables of abundance: A cultural history of advertising in America*. Basic Books.

Lerner, A., Kohno, T., & Roesner, F. (2017). Rewriting history: Changing the archived web from the present. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1741–1755). New York: Association for Computing Machinery. https://doi.org/10.1145/3133956.3134042

Maemura, E., Worby, N., Milligan, I., & Becker, C. (2018). If these crawls could talk: Studying and documenting web archives provenance. *Journal of the Association for Information Science and Technology*, *69*(10), 1223–1233. https://doi.org/10.1002/asi.24048

Marchand, R. (1985). *Advertising the American dream: Making way for modernity, 1920–1940 (1. paperback)*. University of California Press.

Mozilla. (2023). *Firefox release notes*. https://www.mozilla.org/en-US/firefox/109.0/releasenotes/

Mozilla. (2024). *Document: write() method*. https://developer.mozilla.org/en-US/docs/Web/API/Document/write

Nelson, M. L. (2012). *A plan for curating "obsolete data or resources."* arXiv:1209.2664. arXiv. https://doi.org/10.48550/arXiv.1209.2664

Nelson, M. L. (2013). *The conservative party speeches and why we need multiple web archives*. https://ws-dl.blogspot.com/2013/11/2013-11-21-conservative-party-speeches.html

New York University Cybersecurity for Democracy. (2025). *Ad observer*. https://adobserver.org

Niu, J. (2012). Functionalities of web archives. *D-Lib Magazine*, *18*(3–4). https://doi.org/10.1045/march2012-niu2

Ogden, J., Summers, E., & Walker, S. (2024). Know(ing) infrastructure: The wayback machine as object and instrument of digital research. *Convergence*, *30*(1), 167–189. https://doi.org/10.1177/13548565231164759

Packard, V. (2007). *The hidden persuaders* (Reissue ed). Longmans, Green & Co.

Pennock, M. (2013). *Technology report: Web-archiving web-archiving*. Digital Preservation Coalition. https://doi.org/10.7207/twr13-01

Reid, T. (2024b). Recently_Archived_Ads/All_Ads.csv. https://github.com/savingads/Recently_Archived_Ads/blob/main/All_Ads.csv

Reid, T. (2024). *Display archived ads*. https://github.com/savingads/Display-Archived-Ads

Reid, T., & Kreymer, I. (2023). *Unexpected error occurs when replaying an ad in some web browsers*. https://github.com/webrecorder/replayweb.page/issues/157#issuecomment-1444975888

Reid, T., Poole, A. H., Choi, H. W., Rauch, C., Kelly, M., Nelson, M. L., & Weigle, M. C. (2025). *Archiving and replaying current web advertisements: Challenges and opportunities*. arXiv: 2502.01525. arXiv. doi:10.48550/arXiv.2502.01525

Reyes Ayala, B. (2022). Correspondence as the primary measure of information quality for web archives: A human-centered grounded theory study. *International Journal on Digital Libraries*, *23*(1), 19–31. https://doi.org/10.1007/s00799-021-00314-x

Ruest, N. (2019). *openwayback-2.4.0*. https://github.com/iipc/openwayback/releases/tag/openwayback-2.4.0

Selenium. (2024). *Selenium automates browsers. That's it! What you do with that power is entirely up to you*. https://www.selenium.dev/

SimilarWeb. (2023). *Top websites ranking*. https://www.similarweb.com/top-websites/

Van de Sompel, H., Nelson, M. L., & Sanderson, R. (2013). *HTTP framework for time-based access to resource states – Memento (RFC 7089)*. Internet Requests for Comments. https://doi.org/10.17487/RFC7089

Webrecorder. (2013). *Pywb*. https://github.com/webrecorder/pywb

Webrecorder. (2018). *Wombat*. https://github.com/webrecorder/wombat

Webrecorder. (2020). *Archiveweb.page interactive archiving extension and desktop app*. https://github.com/webrecorder/archiveweb.page

Webrecorder. (2024). *Search*. https://replayweb.page/docs/exploring#search

Webster, P. (2020). *How researchers use the archived web*. Digital Preservation Coalition. https://doi.org/10.7207/twgn20-01

Yoshikawa, E., & Roesner, F. (2025). Exploring political ads on news and media websites during the 2024 U.S. elections. arXiv. https://doi.org/10.48550/arXiv.2503.02886

Zeng, E., Kohno, T., & Roesner, F. (2021). What makes a "bad" ad? User perceptions of problematic online advertising. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1–24). Association for Computing Machinery. https://doi.org/10.1145/3411764.3445459

How to cite this article: Reid, T., Poole, A. H., Choi, H. W., Rauch, C., Kelly, M., Nelson, M. L., & Weigle, M. C. (2025). Problems with archiving and replaying current web advertisements. *Journal of the Association for Information Science and Technology*, *76*(13), 1803–1830. https://doi.org/10.1002/asi.70039