# The Power to Never Be Wrong: Evasions and Anachronistic Attacks Against Web Archives

Robin Kirchner
Technische Universität Braunschweig
Braunschweig, Germany
robin.kirchner@tu-braunschweig.de

Chris Tsoukaladelis
Stony Brook University
Stony Brook, USA
ctsoukaladel@cs.stonybrook.edu

Martin Johns
Technische Universität Braunschweig
Braunschweig, Germany
m.johns@tu-braunschweig.de

Nick Nikiforakis
Stony Brook University
Stony Brook, USA
nick@cs.stonybrook.edu

## Abstract

The Web is subject to *link rot*, where links break as webpages are updated or deleted. Web archiving services, such as the Wayback Machine, have emerged as a key solution to address link rot by archiving web content and preserving the look and feel of websites over time. These services offer critical functionality to users, serving as a historical baseline for an ever-changing Web. Implicit in everyone's use of these services is that they are capable of providing an accurate record of the past and can, therefore, provide reliable ground truth for comparing the past to the present.

In this paper, we demonstrate that this implicit assumption does not necessarily hold. To this end, we propose two new threat models against web archiving services in which attackers can exert control over how their websites are archived. *Evasive adversaries* can distinguish crawlers operated by web archiving services from regular users, selectively denying or altering the content delivered to the former. *Anachronistic adversaries* can not only identify archive crawlers but also deliver content that enables them to retain control over archived snapshots. By abusing fundamental access-control mechanisms of the Web, these attackers can effectively alter the past as recorded by web archiving services. We found that *all* web archives we investigated suffer from one or more of these issues, challenging our current reliance on them.

## CCS Concepts

• **Security and privacy → Web application security**.

## Keywords

Web, Archiving, Attacks, Snapshots

**ACM Reference Format:**
Robin Kirchner, Chris Tsoukaladelis, Martin Johns, and Nick Nikiforakis. 2025. The Power to Never Be Wrong: Evasions and Anachronistic Attacks Against Web Archives. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765051


*CCS '25, Taipei, Taiwan*

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765051

## 1 Introduction

In the late 90s, while the Web was still relatively young, people identified the problem of *link rot*. Link rot refers to web links that break over time and no longer direct users to the content they originally pointed to. Whether because a specific webpage was deleted, a website went defunct, or the original content was replaced with unrelated content, link rot affects the usefulness of web links over long periods. For example, as early as 2003, researchers identified that almost 5% of links crawled from the general Web were broken just two weeks later [12]. Later studies discovered that a large percentage of links embedded in academic publications and legal documents suffer from link rot [18, 40, 63, 64], even to the point of affecting documents from the US Supreme Court [37]. More recently, researchers discovered that almost a quarter of the links to COVID-19 dashboards that were released during the COVID-19 pandemic were broken just one or two years later [3].

Web archiving services emerged as one of the key solutions to address link rot, among other digital preservation needs. These services work by visiting links and indexing their contents, making snapshots available to the public as they appeared on the day they were archived, even if the original links break or eventually point to different content. As of January 2025, the Internet Archive—the world's most popular web archiving service—preserves a collection of over 900 billion webpages going as far back as 1996, demonstrating both the scale of digital preservation but also society's reliance on such services [23]. Next to individual users manually requesting the archiving of pages, cornerstone web services, such as Wikipedia, rely on web archiving services to combat link rot [24, 60].

Over time, users discovered more and more use cases for archived web pages. Web security researchers have, for example, relied on web archives to study how third-party JavaScript has evolved over time [43], when popular websites first implemented online tracking [36], and how they adopted web-security mechanisms [21, 38, 47]. Outside of computer science, an increasing number of users rely on archiving services for a multitude of reasons. For instance, patent lawyers use web archiving services to establish the true date that an invention was first described in public [22]. In contrast, some web users identify stealth edits to articles by comparing the live versions of those articles against archived ones [19, 28, 41], a phenomenon which has been studied in prior work [53]. In addition, the large-scale removal of web content by governments [50] demonstrates that both commercial organizations like news outlets

as well as public ones like governments exert control over their web content in ways that are not always transparent.

Given society's increasing reliance on archiving services, it is no surprise that they have recently become targets of attacks. In 2017, Lerner et al. [35] demonstrated isolated cases where accidental remote links and expired domains in archive snapshots could be abused to hijack individual snapshots on the Internet Archive. More recently, in October 2024, the Internet Archive was compromised by attackers who exfiltrated a database of 31 million authentication credentials and defaced its landing page [1, 15].

Our work generalizes control over snapshots to all content, across all major archiving services. In this paper, we shed light on the "invisible" part of web archiving services. Specifically, we study how exactly these services archive content, both in terms of fetching web pages using their own crawling infrastructure and rewriting the fetched content so that all previously external resources can now be served by the archiving service. To this end, we design a web archive observatory and automate the process of requesting the archiving of pages under our control by popular international and regional archives. We show that attackers can exert ongoing control over their own archived past, long after archival, something that was previously **unheard of**.

Given the discovered details of how archiving services employ crawlers and how they rewrite content prior to archiving, we propose two new types of attackers: i) *evasive adversaries* and ii) *anachronistic adversaries*. Evasive adversaries aim to differentiate archive crawlers from regular users so that they can selectively serve different content to the former. We demonstrate how attackers can succeed in this goal by utilizing *server-side evasions* (e.g., identifying crawlers based on their autonomous systems and TLS fingerprints) and *client-side evasions* (e.g., using JavaScript code that will only execute when an archiving service rehosts websites).

Anachronistic adversaries can conduct significantly more powerful attacks where not only can they detect the archiving of their own websites, but they can also affect that archiving to the point where they can modify content *after it was archived*. In this way, anachronistic attackers essentially control the past (as recorded by independent archiving services) and have the ability to "unsay" things they said that ended up being wrong, harmful, or unpopular.

In summary, we make the following contributions:

- We map the server-side infrastructure of eight web archiving services, including their crawlers' IP addresses, ASN, and TLS fingerprints.
- We showcase how an evasive publisher can use this information to conduct targeted server-side cloaking.
- We survey the web archives' security measures and show how attackers can bypass these defenses to carry out evasive and anachronistic attacks.

In addition, we propose two novel classes of evasions against web archives: CSP Stripping and Script Stripping. We show that CSP Stripping is particularly effective and broadly applicable, as demonstrated across all web archiving services (ref. §5.3.2).

Given the highly visual nature of our proposed attacks, we have recorded extensive demos for all presented attacks against all services and made them available in Section 5.1.

**Table 1: Overview of popular web archiving services, ranked in descending order of popularity**

| Service Name | Operating Organization | Tranco Ranking |
|---|---|---|
| Wayback Machine[FP] | Internet Archive | 157 |
| Archive.Today[CO] | Unknown | 5535 |
| Perma.cc[FP], [$] | Harvard Library Innovation Lab | 52 432 |
| Megalodon[CO][1] | Affility Co. Ltd. | 65 845 |
| Ghost Archive[FP] | Unknown | 128 723 |
| ARQUIVO[FP] | Fundação para a Ciência e Tecnologia | 249 057 |
| FreezePage[FP] | USONSE S.R.L. | 818 607 |
| Conifer[FP] | Rhizome | n/a |

[$] Paid service, [CO] content-only archive, [FP] functionality-preserving archive
[1] Megalodon allowed more functionality before its patch in January 2025.

## 2 Background

In this section, we introduce the in-scope web archives, discuss the different attackers' motivations and capabilities, and provide a brief background on specific client-side security guarantees that are at risk when rehosting websites under one archive domain.

### 2.1 Web Archives

The Internet Archive's Wayback Machine is just one of many functional archiving services. Table 1 shows eight services we identified by considering related work and search-engine results. We use the Tranco list[1] [34] for ranking websites. These services vary significantly in ranking, with the Wayback Machine being the 157$^{th}$ most popular website on the Internet and FreezePage being ranked in the 800 thousands. The Tranco list does not differentiate between subdomains, so no Tranco ranking is known for Conifer. However, Conifer's parent organization, Rhizome, is ranked at 21 380. Besides services with a global audience, we also observe regional services, such as Megalodon, which is aimed at Japanese users, and ARQUIVO, which is aimed at Portuguese users. Finally, while the identities of the operators of many of these services are known, the organizations behind two services—Archive.Today and Ghost Archive—are unknown. Notably, Archive.Today is served under multiple top-level domains, including *.today*, *.ph*, *.vn*, and *.is*, of which archive.is the most popular URL with a Tranco ranking of 5535 and the lowest *archive.vn* ranked at 40 503. Preliminary tests revealed no differences in the behavior and infrastructure of these services. Thus, we decided to limit our testing to Archive.Today, which is the title name given on all alternative URLs of the service. Most services are free, donation-based, or offer optional premium features. Conifer, FreezePage, and Perma.cc require an account, while additional URLs have to be bought for Perma.cc after an initial set of 10 free archived pages.

There is an implicit user expectation that web archives provide unchangeable, truthful records of websites. The archives reinforce this expectation through statements such as: "*Websites change. Perma links don't.*"— Perma.cc [45], "*Prove exactly what was at a web address at a specific date and time.*"— FreezePage [16], "*Capture a web page as it appears now for use as a trusted citation in the future.*"— Internet Archive [25], "*Ghostarchive will store a snapshot of the website as it appeared at the time of archival.*"— GhostArchive [17].

[1] Generated on 22 January 2025 and available at https://tranco-list.eu/list/4QJ3X.

However, our analysis reveals that most defenses are insufficient to prevent post-archival modifications to snapshots (ref. §5.3).

*Functionality-preserving web archiving.* Functionality-preserving archives like the Wayback Machine rehost a page with all its subresources. This is invaluable for creating a memory of the Web as it was in the past, including all of its technologies and vulnerabilities. These archives singlehandedly enabled web research over the years, such as the study of historic security vulnerabilities [e.g., 43, 52]. As an example, snapshots of OpenStreetMap in the Internet Archive still allow scrolling and moving on the map[2].

*Content-only web archiving.* Even though the majority of web archives try to preserve a website's core functionality, alternative approaches exist. Archive.Today, as a prominent example, removes all interactive functionality of the page it captures, primarily aiming to retain a website's final rendered appearance. In contrast to the Wayback Machine's snapshot, Archive.Today's snapshot of OpenStreetMap is non-interactive[3].

## 2.2 Reasons for Evading Web Archives

We argue that the threat model of websites altering their own archived past is not only realistic but increasingly relevant. Prior studies have revealed that news websites engage in stealth edits [53], silently altering content after publication. Moreover, governments have been shown to actively remove web content at scale [50]. Despite the trust in archival permanence by legal, journalistic, and scientific communities, recent findings challenge that trust [35]. Undermining that assumption of permanence, especially after the initial archival, has radical and cascading implications.

We differentiate between benign and traditionally malicious websites when systematizing why website operators may have an interest in avoiding their websites being archived or in controlling what content gets archived.

*Benign websites.* Prior research has shown that some users utilize web archiving services in ways that may financially hurt the websites being archived. For example, Zannettou et al. established that some users resort to archiving services to deny ad revenue from websites that they consider objectionable, e.g., due to disagreements between the perceived political leaning of a website vs. the one of these users [61]. By sharing archived links (instead of live links) on social media, news sites would lose page visits and the associated ad revenue from these impressions. Tsoukaladelis et al. discovered that news sites often change their articles post-publication in a way that frequently exceeds the mere fixing of typos and benign rephrasing [53]. These often-silent post-publication changes are an unwanted side effect of online news, prompting users to increasingly rely on web archives as their source of ground truth—that is, to verify what an article stated at the time it was archived. There are multiple reasons why even reputable websites might wish to "unsay" something they previously published by controlling or restraining web archiving. For example, organizations engaging in stealth edits may wish to avoid leaving publicly verifiable evidence of these changes. Likewise, governments may oppose archiving when it exposes inconsistencies or documents that were later removed.

*Malicious websites.* There can be multiple reasons that malicious websites want to detect visits from web archiving services. Malicious sites have long engaged in cloaking [26, 55, 62] to differentiate between visits from security crawlers and those from prospective victims. When a malicious website detects a visit by a security crawler, it may hide its real malicious content by, for example, redirecting users to benign websites. Similarly, malicious sites that attempt to rank highly on search results can show different content to search engine crawlers than to result-clicking users. We anticipate malicious sites may want to evade web archiving services to increase the longevity of their attacks and have later deniability as to what exactly was served by that website.

## 2.3 Attacker Model

We present two primary adversaries: the *evasive adversary*, and the *anachronistic adversary*. Both attackers (ab)use characteristics of the archiving services to exert control over the creation of snapshots of the adversary's website. Thus, they break the assumption that all content shown in a snapshot existed like this at the time of archival. We acknowledge that not all content creators want their pages to be archived, which we believe is a legitimate viewpoint. Malicious evasions, escapes, and anachronistic manipulations, however, go beyond personal preference and subvert the integrity of archives.

*2.3.1 Evasive Adversary.* The evasive attacker aims to prevent their web content from being properly archived by:

- Detecting and blocking archive service crawlers,
- Serving different content to archiving service crawlers, or
- Exploiting technical characteristics and limitations of archiving services to prevent complete content capture.

An evasive adversary can target either all web archives or specific ones. Making snapshots of the adversary website unusable in specific archives can entice web archive visitors to resort to alternative archiving services, which the adversary potentially has more control over.

*2.3.2 Anachronistic Adversary.* The anachronistic attacker attempts to modify archived content *after* it has been captured by:

- Exploiting URL-rewriting mechanisms to inject future references to other archived resources,
- Injecting references to external resources that "escape" the archive,
- Taking advantage of JavaScript execution in archived pages.

## 2.4 Web Security Policies

The Same-Origin Policy [8] (SOP) and the Content Security Policy [59] (CSP) are the Web's most important client-side security policies for our work.

The SOP is a fundamental access control policy of browsers that restricts how web documents and resources from different origins can interact. In the context of web archiving, third-party websites are often rehosted under the archive's own domain. This would break its functionality without adjustments to the pages, as the browser would block all third-party resources when the page is hosted under the new domain. For this reason, web archives often capture subresources, like images, scripts, and documents, and then

[2]Snapshot of OpenStreetMap in Wayback Machine https://web.archive.org/web/20250211015148/https://www.openstreetmap.org/ (2025-02-11).
[3]Snapshot of OpenStreetMap in Archive.Today https://archive.ph/7oVH3 (2025-02-23).

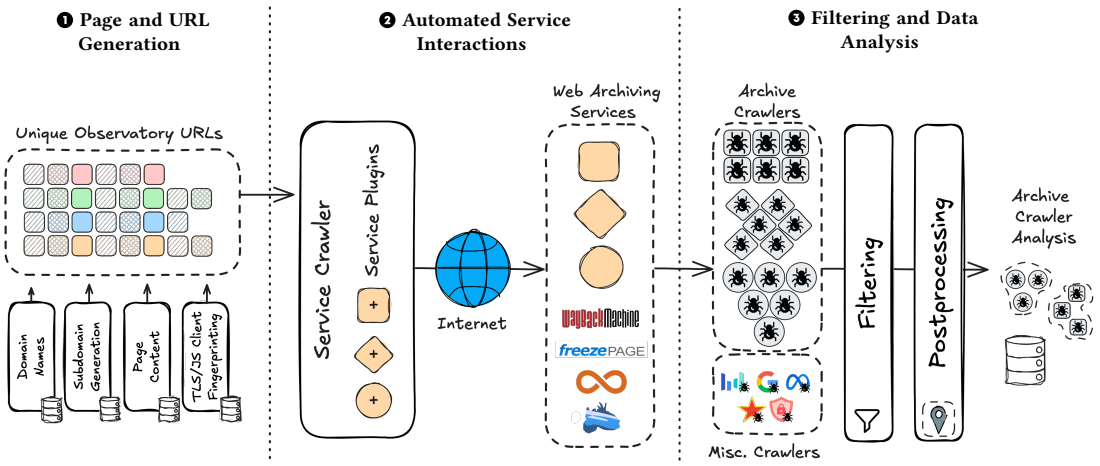


**Figure 1: Overview of our archive observatory, which includes instrumented websites on multiple domains, automated archive interaction, and data analysis.**

rewrite references to these resources on their copies of the captured websites. Oftentimes, scripts have to be rewritten as JavaScript APIs like *document.domain* or *window.location* would return unexpected values under the new domain.

The CSP is a directive-based security policy that controls which resources a page can load. Its purpose is to mitigate security risks like cross-site scripting (XSS) by defining the origins from which scripts are allowed to run. When a web archive service rehosts a website, it is challenging to rewrite that website's original CSP to ensure that the rewritten resources are not blocked. Therefore, one of the discoveries of this paper (Section 4.1.1) is that archives typically strip a website of its CSP when rehosting it, potentially replacing it with the archive's own CSP if it has one.

## 3 Web Archive Observatory

To achieve our goal of mapping and quantifying existing archiving services, we must examine what happens with websites when web archiving services create snapshots. Web archives use crawlers to visit pages they want to create a snapshot of. Studying these crawlers in the wild is best done by triggering archiving requests for controlled websites.

Our web archive observatory platform automates the process of archiving instrumented web pages via different archiving services and helps evaluate collected metadata like network information and client fingerprints. Figure 1 illustrates this platform, which combines both active and passive components and operates as a honeypot accessible under a specific domain. Each deployment of our platform constitutes a separate observatory instance with its own domain. The active part of our platform attracts visits from web archive crawlers by semi-automatically requesting the archiving of generated, unique links with non-predictable subdomains. The passive part of the observatory serves instrumented webpages for the links generated by the active part, similar to a honeypot. When the archiving services' crawlers visit our unique links after we requested their archival, our observatory platform records several pieces of information, such as network information and client fingerprints. The information recorded by our platform can later be used to recognize archiving service crawlers and study the services' vulnerability to evasion and anachronistic attacks.

### 3.1 Website and URL Generation

The passive part of our observatory ❶ involves serving an extensive number of instrumented websites. To allow us to attribute crawler behavior and isolate its characteristics confidently, we distribute our architecture over ten instances, each carrying a separate domain. Each observatory instance acts as a honeypot, serving arbitrary subdomains and URLs of the respective domain while collecting metadata about its visitors.

*3.1.1 Crawler Client Fingerprinting.* The generated observatory web pages include state-of-the-art TLS client fingerprinting techniques to understand and map the capabilities of different web archiving service crawlers.

*TLS Fingerprinting.* We adopt the JA3 [48] and JA4 [14] TLS fingerprints (initially intended for threat detection) to differentiate between archive crawlers based on their TLS stacks. TLS fingerprinting operates on the Client Hello packet that is transferred during the initial TLS handshake to establish an encrypted connection between the client and the server. Salesforce's JA3 [6] method uses information from the Client Hello, including the TLS version, TLS extension lengths, and cipher suites, to create a TLS fingerprint of the TLS client's configuration. The more recent JA4 method reduces the number of fingerprints for modern browsers by accounting for the randomization used by these browsers during the TLS handshake [10]. For a later comparison of archive crawlers with the fingerprints of current browsers, we collected the TLS fingerprints of the Top 20 desktop browser versions by market share from 02/2024 to 02/2025 [51].

*Client-side Browser Fingerprinting.* Browser fingerprinting is a common technique in online tracking that identifies unique devices by leveraging JavaScript APIs exposed by the browser [e.g., 13]. Although we initially considered including browser fingerprinting in

**Table 2: Unique archiving service crawler endpoints and number of requests to our observatory.**

| Archive | Geolocation: US | Canada | Portugal | Japan | Russia | *Other* | Distinct IPs / ASNs | Meta Information: #VPN IPs | #UAs | #JA3 | #JA4 | Number of Requests: Archive Crawler(s) | Others after 1h | 1 day | 7 days |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Wayback | 19 | - | - | - | - | - | **19** / 1 | 0 | 2 | 1061 | 2 | 1894 | +2 | 10 | 95 |
| Archive.Today | 8 | 1 | - | 3 | 10 | 25 | **47** / 27 | 15 | 196 | 3 | 3 | 1752 | +50 | 231 | 262 |
| Perma.cc | 4 | - | - | - | - | - | **4** / 1 | 4 | 42 | 4 | 4 | 2321 | +60 | 551 | 2759 |
| Megalodon | - | - | - | 3 | - | - | **3** / 1 | 3 | 2 | 1 | 1 | 1950 | - | 197 | 246 |
| Ghost Archive | 1 | - | - | - | - | - | **1** / 1 | 1 | 1 | 2 | 2 | 1100 | - | - | - |
| ARQUIVO | - | - | 2 | - | - | - | **2** / 1 | 0 | 4 | 2 | 2 | 1552 | - | - | - |
| FreezePage | 1 | - | - | - | - | - | **1** / 1 | 1 | 3 | 1 | 1 | 978 | - | - | - |
| Conifer | 1 | - | - | - | - | - | **1** / 1 | 1 | 2 | 2 | 2 | 1365 | - | - | - |

our observatory study, we ultimately chose not to do so for multiple reasons. A primary goal of the observatory is to support server-side detection of archive crawlers. This requires that distinguishing information is available in time for the server to tailor its response to a request. However, browser fingerprinting does not meet this criterion, as the relevant results become available only after the client executes JavaScript included in the server's response. By that point, the website's content has already been delivered to the client.

The browser fingerprint could still be used in the client-side code to detect archive crawlers by comparing their fingerprint against a list of known crawler fingerprints and altering the website accordingly when viewed by an archive crawler. However, the archive crawler acts as a client only during the capture. After the archival process, archive visitors are the clients, which can lead to unintended side effects.

Moreover, Gómez-Boix et al. [20] observed a declining number of unique browser fingerprints and found that non-unique fingerprints are brittle. According to their findings, the trend of modern browsers to reduce plugin support substantially lowers fingerprint uniqueness, and even small changes to a single feature can significantly affect the overall fingerprint. For these reasons, we ultimately decided against using browser fingerprinting in our study.

## 3.2 Automated Service Interactions

The active part ❷ of our observatory facilitates the archival of hundreds of instrumented observatory webpages. All archiving services in our study offer a feature that allows users to create live website snapshots on demand. The Wayback Machine, for example, refers to this functionality as "Save Page Now." Following this terminology, we refer to such features as *save-page features* or *save-page requests* throughout the paper. We have developed separate web modules for each archiving service, mimicking how users of that service navigate the website and submit save-page requests. We automate the archiving process as much as possible to facilitate saving a sufficiently large number of websites, allowing us to draw conclusions about the operation of each service. We automated this process for Wayback Machine, Archive.Today, Ghost Archive, ARQUIVO, FreezePage, Conifer, as well as Perma.cc by navigating to the respective save-page feature using a browser instrumented with Selenium [11] and a Chromedriver with basic bot-detection evasion techniques [54]. Megalodon uses extensive anti-bot measures via Cloudflare. Hence, we manually conducted our archiving requests for that service.

A save-page experiment for any of the ten observatories entails assigning a random *unguessable* alphanumeric subdomain for the experiment. The observatory creates a record for that archiving request in its database, associating the newly generated URL with the archiving service. The archival timestamp is stored after the page is saved using the respective web module for the archiving service's save-page process. This data can later be used to associate visits to our observatory addresses and the entailing characteristics, e.g., HTTP headers, TLS fingerprints, with the archiving service crawlers.

## 3.3 Filtering and Data Analysis

In the last part of the observatory ❸, we analyze the collected data regarding each archive's crawlers. In the presence of unrelated web scrapers and scanners, we must ensure that the visitor data we use belongs to the crawlers of the archiving services. Our primary filtering mechanism involves generating observatory pages with unique alphanumeric subdomains to differentiate between archive crawlers and unrelated visitors. For instance, visits to https://observatory.test may come from a wide range of scanning bots, but a visit to a URL like https://2a94b7-97d92f.observatory.test/ that was only revealed to the Wayback Machine at a known time is highly likely, if not guaranteed, to be originating from the archive's crawlers.

Knowing that bots use Certificate Transparency (CT) to identify new targets to crawl [29, 46], we rely on wildcard HTTPS certificates to ensure our subdomains are never revealed in CT logs. As such, initial visits to a specific subdomain can only originate from the archiving service, and potential third parties later learning about it through that service's data.

Regarding that last point, we observed that some archiving services offer searching capabilities or public API endpoints[4] where newly archived pages can be found. We have to expect requests from third-party visitors for these specific archive services. We always attribute the first request to an observatory URL generated explicitly for an archive to belong to one of the archive's crawlers. Modern pooled server-side infrastructure sometimes loads the resources of

[4]Perma.cc has a public API (https://api.perma.cc/v1/public/archives) where newly archived pages can be polled.

a single website using multiple crawlers situated on different IP addresses. Thus, we attribute requests within an empirically-derived time window of 5 minutes after the initial request or sufficiently similar client characteristics (like IP addresses) to the archive's crawlers. We supplement this with additional filters based on the IP ranges of large Internet companies, such as Google or ByteDance, and publicly accessible lists [42] of known web crawlers.

After filtering, we have a database containing metadata about crawlers from different web archives. This data includes all the information our observatory web pages extracted about each archive crawler, i.e., their IP address, declared user agent, and other HTTP headers, as well as TLS fingerprints. This data is then further augmented using open-source intelligence (OSINT) lists [2] to include a host's geographical location, its Autonomous System (AS), and if it is a VPN server or a host belonging to a cloud provider.

## 3.4 Observatory Study

To map the observable infrastructure of the selected targets, we conducted a partially automated study using our ten observatory instances, which perform automated service interactions for archives that can be automated. Each experiment is associated with an archiving service and receives a newly generated URL for a randomly chosen domain of one of our observatory domains.

For our study, each experiment comprises archiving a new URL using the respective archive's save-page feature and observing subsequent HTTP requests targeting the experiment's URL. We limit automated save-page requests to approximately one save per hour to ensure we are not overloading the web archive services. The automated archiving experiments were conducted over the course of 16 days. The manual experiments for Megalodon were done over a period of 37 days. Perma.cc is the only paid web archive service. Thus, it functioned as the floor for our study's number of archiving experiments per archive. We opted to purchase 200 links, of which five were used during testing. In summary, for each web archive, we actively requested the archiving (save-page request) of at least 195 controlled URLs, distributed across the ten observatory instances. Each controlled URL carries a newly generated, random, and unguessable subdomain (see §3.3 for an example). These controlled URLs lead to *observatory pages*, each tied to one specific save-page request. These save-page requests led the archives' crawlers to request and create a snapshot of our observatory websites. Their visits give us information about the infrastructure of each archiving service, as described next.

## 3.5 Archive Crawler Infrastructure

This section presents the dataset resulting from our observatory study with respect to our first research goal of mapping the services' infrastructure. Table 2 (left) displays the number of unique archive crawler endpoints we discovered after 195 saved pages.

*3.5.1 Crawler Endpoints.* Archive.Today was the archive with the most distinct crawlers visiting our observatories, operating 47 distinct IP addresses from 27 different autonomous systems. With 19 different geolocations, it is the only archive service that operates hosts in more than one country. By conducting follow-up experiments that requested from Archive.Today to archive additional URLs, we concluded that we were nowhere near exhausting their pool of crawling hosts. The Wayback Machine—the most popular archive service on the Web—operated 19 crawler endpoints, which were exhausted after the archiving of 21 observatory pages.

As anticipated, the regional services ARQUIVO and Megalodon send requests from their respective country of origin, while most other archives operated crawlers from the US. According to OSINT data [2], Archive.Today about a third of the archive's IP addresses belong to known VPNs. The IPs of Perma.cc, Megalodon, Ghost Archive, FreezePage and Conifer are entirely flagged as VPNs.

The right part of Table 2 shows the total number of requests we received from archive crawlers and other unrelated clients following the archival of our pages on the respective archives. Perma.cc's choice to publish newly archived URLs in a public API (ref. §3.3) attracts one order of magnitude more unrelated visits to our web pages after one week compared to the other archives. Most of Perma.cc's third-party requests come from ByteDance's *ByteSpider* and Majestic's *MJ12bot*, who gather training data for LLM training [9] and search engine improvement [39], respectively. Megalodon, and Archive.Today attract a similar amount of third-party visitors, followed by the Wayback Machine with less than half that amount. Archives without a public search function did not result in visits to the archived websites.

Figure 2 visualizes how the Wayback Machine and Archive.Today differ from the other archives for which we observed precisely one AS with four or fewer unique crawler endpoints. The graph includes only Archive.Today's ASN data, as all others showed just one ASN.

*3.5.2 Fingerprinting Crawlers.* ARQUIVO and the Wayback Machine include the respective archive's name in some User-Agent headers. The remaining user agents are masked as regular browsers. Archive.Today displays a noticeably diverse set of user agents, ranging from different desktop browser versions to mobile browsers. With three-year-old Chrome versions, we received the most outdated user agent strings from the Wayback Machine (Chrome 89–115) and Perma.cc (Chrome 90–131), followed by two-year-old browser versions from Archive.Today (Chrome 100–121) and Ghost Archive (Chrome 106). The remaining archives used only recent Chrome user agents during our study.

All services converge to four or fewer JA4 TLS fingerprints, as displayed in Figure 2's right graph. However, the Wayback Machine stands out. While we only observed 19 IP addresses, their crawling infrastructure shows a surprisingly large variety in TLS client configurations, leading to 1061 JA3 fingerprints after 195 saved pages over ten days. Diving deeper into the Wayback Machine's Client Hello parameters reveals that the TLS client offers three typical cipher suites, followed by a selection of other cipher suites, which explains the various JA3 fingerprints. However, all TLS client configurations negotiated identical final connection parameters, resulting in just one JA4 fingerprint for the Wayback Machine crawler.

The ever-increasing number of crawler endpoints we uncovered for Archive.Today makes network-based detection of the service's crawlers challenging. The limited crawling vantage points of the remaining archives allow attackers to conduct server-side evasion attacks straightforwardly, which we discuss in Section 4.1.3.

*3.5.3 Reflecting the Filtering Method.* To test our assumption that the generated observatory URLs are sufficiently hard to guess, we checked and found no visits to URLs used in experiments before the

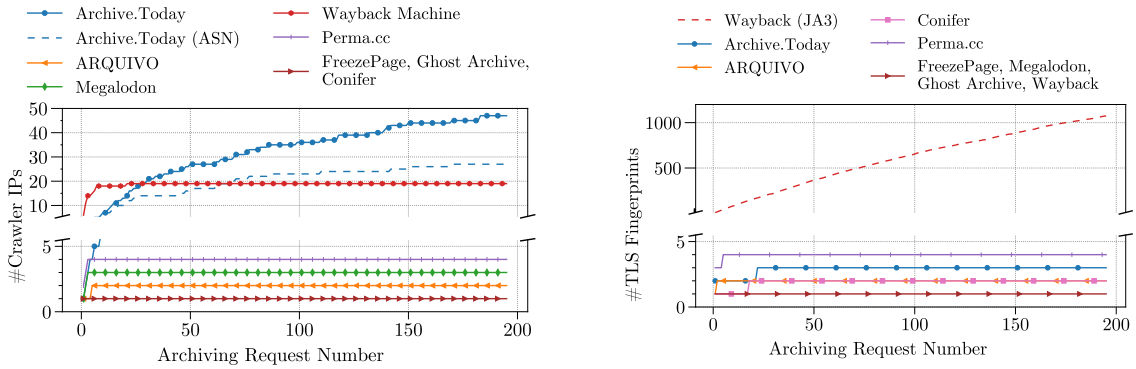


**Figure 2: Unique archiving service crawler endpoints contacting our observatory after our archiving requests.**

experiment started. We conclude that no URL-scraping campaigns were launched against our domains, which would have interfered with our results. ARQUIVO, Ghost Archive, FreezePage, and Conifer are our baseline for the filtering.

Cross-checking the visitor data from our results database with OSINT lists revealed a match with the Yandex bot's user agent, IP addresses, and ASN. Namely, Archive.Today occasionally used this crawler immediately after our save-page request. Thus, we conclude this crawler instance is operating directly for Archive.Today.

False positives in our crawler filtering approach are unlikely due to the short timeframe we allow for crawler visits to occur. Additionally, we observed a wide range of Chrome user agents, with the oldest versions being over three years old. Chrome's automatic updating mechanisms make it highly unlikely that human users operate browsers with such outdated user agent strings.

## 4 Attacks against Web Archiving

While web archives effectively combat link rot, i.e., the disappearance of information previously available at a certain Web destination, not all website owners share the goal of preservation. For some, especially malicious actors, permanence is a threat rather than a benefit. Journalists and researchers, for instance, rely on web archives to reference past content or study how websites were composed at different points in time [e.g. 21, 36, 38, 43, 47, 52]. In contrast, malicious actors may seek to *evade* archival evidence. A website that temporarily hosts illegal or controversial content may benefit from ensuring that no lasting evidence remains in archives. Similarly, political entities or publishers may attempt to control or sanitize the content captured about their web presence. This section proposes five attacks against web archives that aim at one or more of these attacker goals.

Table 3 summarizes these attacks by their intended effect. *CSP Stripping*, *Script Stripping*, and *Server-side Cloaking* aim to prevent the creation of truthful website snapshots. *Archive Anachronisms* attacks and *Live-web Escapes* are even more powerful, as they allow attackers to influence and control a snapshot's appearance even after it has been archived. Figure 3 illustrates a key adversarial goal: a website that appears red on the live web but turns blue when rehosted by a web archiving service. Throughout the paper, we use this color distinction as a running example—red ☐ represents the truthful, original appearance, while blue ☐ denotes the defaced or manipulated version served by the archive.

**Table 3: Formalization of our proposed attacks.**

| Adversary Type | Attack Name | | Scope | Payload Source | Payload Preparation |
|---|---|---|---|---|---|
| Archive-Evasion Attacks | *CSP Stripping* | | client | *n/a* | before |
| | *Script Stripping* | | client | *n/a* | before |
| | *Server-side Cloaking* | IP-based | server | *n/a* | before |
| | | TLS-based | server | *n/a* | before |
| | | UA-based | server | *n/a* | before |
| Anachronistic Attacks | *Archive Anachronism* | | client | archive | anytime |
| | *Live-web Escape* | | client | live-web | anytime |

### 4.1 Archive Evasion Attacks

As discussed in Section 2.2, evasive adversaries may want to prevent archives from creating faithful representations of their websites in the form of a snapshot. To achieve this goal, they can abuse the characteristics of web archives. First, the evasive adversaries can incorporate client-side code in their websites that alters a page's appearance *only* when it is hosted in the archive's context (see §4.1.1 and §4.1.2). Alternatively, these attackers can collect identifying information about the archive's crawlers, such as network specifics, to distinguish between archive visits and regular visits (ref. §4.1.3).

*4.1.1 CSP Stripping.* Copying a live website and automatically placing it in another origin is challenging, as browser security guarantees are tightly bound to a website's origin, i.e., its web address consisting of protocol, domain, and port. One important security mechanism browsers enforce is the Content-Security Policy, which controls what script resources a page can load. While creating a snapshot, keeping the script content of a website is especially dangerous, which is why archives usually have their own defenses or policies in place. Thus, archives may opt to ignore the CSP of the pages they publish as snapshots under their domain. This characteristic effectively creates *archive-only* scripts that conveniently only trigger when a CSP-stripping archive rehosts the website. Specifically, an evasive website could define a CSP that disables execution of one of its own scripts. When this website is rehosted by a web archive that strips its CSP, the initially dormant code will be allowed to run. This effectively creates *archive-only* code which can freely alter a snapshot's appearance on execution.

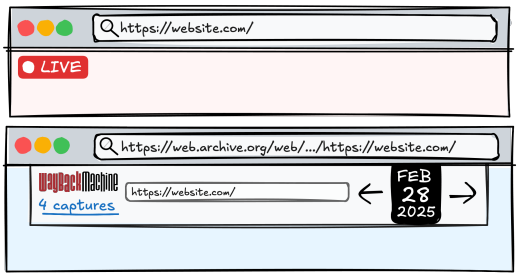


**Figure 3: A website's snapshot appears different (□ blue) from the live version (□ red) due to evasion attacks.**

```
1  Content-Security-Policy: script-src-attr 'none';
```

```
2  <!-- ❶ default state: -->
3  <div id="team-name">□ Red Dragons</div>
4
5  <!-- ❷ archive-only script: -->
6  <img src onerror="document.getElementById('team-name').textContent =
   ↪  '□ Blue Sharks';"></img>
```

In the above example, the evasive website initially displays □ red content, illustrated by the HTML div element *"team-name"* ❶ (Line 3). The onerror handler ❷ (Line 6) is normally blocked by the page's own CSP, which disallows JavaScript inside HTML attributes (Line 1). This block is lifted when the page, excluding its CSP, is re-hosted by a web archive. Therefore, the previously dormant onerror handler runs, modifying the page content that users see (□ blue).

*4.1.2 Script Stripping.* Archives that disable script tags facilitate another attack that relies on the absence of a script. The attack works when an attacker's website defaults to a defaced look (□ blue), and a script is responsible for constructing the page with the correct look (□ red). When an archive removes that constructive script, the page remains defaced, as illustrated below.

```
1  <!-- ❶ default no-script state (defaced) -->
2  <div id="team-name">□ Blue Sharks</div>
3
4  <!-- ❷ constructive script: -->
5  <script>
6      document.getElementById('team-name').textContent = "□ Red Dragons";
7  </script>
```

In the scenario where the constructive script ❷ is never executed (due to rewriting or blocking by the archive), the page remains in the initial (defaced) state ❶, represented by the □ blue color.

*4.1.3 Server-side Cloaking.* Attackers have many "signals" available to distinguish between a regular user visit vs. a web archive crawler. Depending on the targeted web archives, attackers can evade archive crawlers based on their IP address, autonomous system number, and identified browsing software. Using any of these vectors, attackers can serve archive crawlers content that differs from that served to regular web users. Compared to the aforementioned attacks, this evasion is better concealed since there is no evidence of alternative content saved in the page's HTML or JavaScript code for analysts to identify. Moreover, attackers can use server-side cloaking to deny a website's archiving altogether.

The primary criterion for successful server-side evasions is the ability to determine the nature of the current visitor in real-time. Our observations from Section 3 indicate that web archive crawlers reveal sufficient metadata about their identity, allowing attackers to perform server-side evasions for most of the studied services.

> **Key Takeaway 1.** Archive crawlers can be selectively deceived during snapshot creation.

*Robots.txt.* The *robots.txt* [30] file can instruct compliant crawlers on which website resources to access, potentially serving as a mechanism to control what content is visible to different archive user agents. However, an ancillary study revealed that the diversity of user agents is too great, and the in-scope archives either did not request the *robots.txt* file or disregarded its directives.

## 4.2 Anachronistic Attacks

The term *anachronistic attack* refers to content that appears out of its proper time. Here, an archive *(involuntarily)* allows snapshots to use resources that were nonexistent at the time of the archival.

Archive snapshots are typically regarded and advertised as immutable representations of a website's look at the time of archival. However, effective defenses are vital to prevent escapes from the archive's boundaries, especially with functionality-preserving web archiving (ref. §2.1). The anachronistic adversary (ref. §2.3.2) abuses flaws in these defenses to alter the appearance of snapshots *after* their creation, allowing adversaries to exert even more control over snapshots of their page than with evasive attacks. We distinguish between *archive anachronism* and *live-web escape*.

*4.2.1 Archive Anachronism.* Archive anachronism attacks refer to attacks where the resources later used to alter a snapshot come from the archive's origin; see Figure 4. The adversary includes a future reference to a page-altering resource from the archive. Initially, this referenced resource might not exist, making the attack dormant. When the attacker decides to archive the missing resource, an archive-anachronism occurs, in which a new resource is used in an older website snapshot. Using a resource from within the archive's origin means that attackers can bypass even the strictest CSPs that stop all communications of an archived page with third-party websites. An archive anachronism attack has three requirements:

*(1) Script execution and controllable URL rewriting.* The attacker must be able to control the archive's URL rewriting so that a snapshot of the attacker's website can reference an archived resource acting as a payload. A JavaScript file loaded as the source of a script tag in the final snapshot can be such a resource.

*(2) Guessable snapshot URL.* The URL of newly archived resources must be predictable. In Figure 4, the attacker's website refers to an external resource from the year 2050. Wayback Machine is known to utilize nearest-neighbor matching when given the timestamp of a resource. For example, the path /web/2050/<domain>/change.js may redirect to /web/20250325090111/<domain>/change.js if that is the most recent snapshot of the resource.

*(3) Retrievable resource content.* Most web archives conduct some form of script rewriting when archiving a resource. Rewriting subresources would often interfere with correctly displaying the page when replaying a recorded website. Thus, archives have (*often undocumented*) ways to retrieve the "raw" resource without the rewriting. In our example, where a script resource is loaded from a

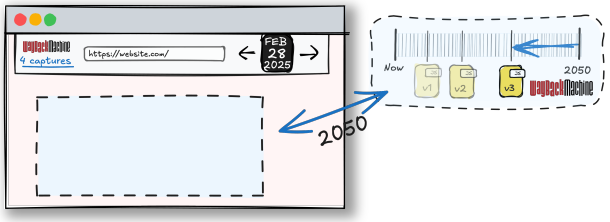


**Figure 4: Archive anachronism attack—future-reference to a resource inside another snapshot.**

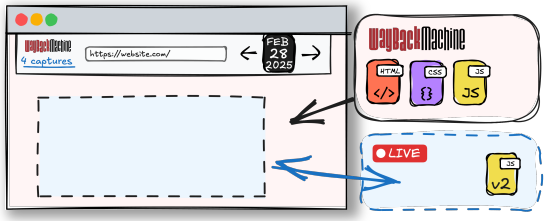


**Figure 5: Live-web escape attack—a snapshot breaks the archive's boundaries to include a live-web resource.**

future snapshot, it is essential that script rewriting does not hinder the script's functionality. To illustrate, the following HTML snippet combines these three techniques.

```
<!-- The new script runs once Conifer rehosts the page -->
<img src onerror="const script = document.createElement('script'); script.src
↪ = 'https://cones.conifer.rhizome.org/<username>/default-collection/
↪ 20500000000000js_/https://<subdomain>.observatory.test/change.js.txt';
↪ document.body.appendChild(script);">
```

At the time of our study, Conifer executed code in onerror handlers, satisfying the first requirement for an archive anachronism attack. The executed code dynamically creates a new script tag from a source within the archive. Here, snapshot URLs are guessable. Namely, the URL includes a future timestamp, which the archive redirects to the nearest existing timestamp of that resource, satisfying the second requirement. Finally, two additional techniques are used to ensure Conifer returns a raw and executable version of the referenced script. Appending the undocumented string *js_* to the timestamp, combined with a *.txt* resource type, prompts the archive to serve the raw script content. Once the page with this code is rehosted under Conifer's domain and the referenced resource becomes available, i.e., after a save-page request for the payload URL, the referenced script can execute and change the snapshot.

*Payload updates.* From an attacker's perspective, a web archive ideally has a timestamp-matching feature that enables them to update the appearance of an older snapshot at their discretion by simply requesting the re-archiving of their payload resource. When the page's snapshot is revisited, the visitor's browser will follow the reference, load the updated resource from the archive's origin, and execute it. Without such a mechanism, the payload itself can alternatively include a forward reference to a following payload, effectively creating a chain of payloads. Either of these two methods would allow attackers to perform anachronistic attacks not just once but an arbitrary number of times in the future.

*Resource deletion attack.* There is a theoretical variation of this attack abusing an archive's delete snapshot functionality. In reverse order to the archive anachronism attack, a resource from the archive is initially used to control the look of a snapshot. The page is built so that if that resource is not present, it defaults to a defaced appearance. The attack involves requesting that the resource be deleted at a later time, e.g., via the archive's delete snapshot functionality. Upon fulfillment of that deletion request, the snapshot's appearance defaults to the defaced look (refer to Appendix A.1).

*4.2.2 Live-web Escape.* In contrast to archive anachronism attacks, live-web escapes completely overcome the archive's boundaries to load resources from another origin, i.e., the live Web. Figure 5 illustrates a live-web escape where a script in a website's snapshot escapes the archive's origin to load another script from the live web that defaces the snapshot.

Like archive anachronism, live-web escape requires script execution. Code included in a website's snapshot will run under the archive's origin after archival. To affect the snapshot's appearance, two cross-site scripting (XSS) capabilities are required: *(1) External requests.* The code in the snapshot must be able to request a resource from an attacker-controlled domain, which typically contains additional attacker code. *(2) DOM manipulation.* Second, to deface the snapshot, the attacker requires DOM manipulation capabilities. Depending on the adversary's goals, they might want to apply a wide range of DOM modifications, ranging from subtle changes to the content of an older snapshot to radically changing the look and feel of the archived website.

As a defense, a sufficiently restrictive Content-Security Policy (CSP) can prevent code execution from third-party sources, assuming attackers cannot somehow bypass the restrictive CSP. In Section 5.2, we discuss the concrete defensive measures—including potential CSPs—of the archiving services.

*Attacks summary.* Archive anachronism attacks execute resources from outside a snapshot's original time period to alter a page's appearance *after* it was captured. Depending on the exact variation of the attack, the attacker-controlled anachronistic resources either come from within the archive's origin or outside of it. In contrast to these attacks, archive-evasion attacks aim to prevent web archives from accurately archiving websites.

**Key Takeaway 2.** Attackers can modify their own archived content long after archival—a novel and previously unheard-of attack vector.

## 5 Results and Evaluation

In addition to the descriptive statistics produced from the metadata collected during our experiment period (ref. §3.5), we analyzed the way each archiving service creates snapshots of pages to understand their vulnerability to evasions and anachronistic attacks (ref. §4). In this section, we describe the different defensive measures that web archives employ and present their vulnerability to our proposed archive attacks, despite these countermeasures.

## 5.1 Demonstration

We have produced detailed recordings of the attacks described in this paper and edited them into a single 36-minute-long video, which we uploaded to Vimeo. We prepared timestamped links for all successful attacks, with four examples in the table below.

| *Timestamp* | *Description* | *Link* |
|---|---|---|
| 26:49 | *Conifer Archive Anachronism* | Vimeo |
| 06:45 | *Wayback Machine Live-web Escape* | Vimeo |
| 07:55 | *Archive.Today CSP Stripping* | Vimeo |
| 20:53 | *Perma.cc Server-side Cloaking* | Vimeo |

The first two videos demonstrate our anachronistic attacks, *Archive Anachronism* and *Live-web Escape* for Conifer and the Wayback Machine, respectively. The remaining two videos show archive evasion attacks on Archive.Today via *CSP stripping* and Perma.cc via *Server-side Cloaking*. All links are available in Table 5.

## 5.2 Web Archive Defenses

Table 4 displays the defensive measures the web archiving services deploy. Megalodon stands out, as it underwent a series of security patches [4] in January and February 2025 during our study. While the patches were effective, their application was incomplete. Specifically, Megalodon offers a "snapshot-only" view[5], which can be accessed both through the menu and by modifying the URL. Placing /ref/ before the timestamp in the URL navigates directly to the snapshot-only mode. Megalodon's website displays a toolbar, which is removed in snapshot-only mode. As both modes work interchangeably, we describe our results for the snapshot-only mode.

A well-defined CSP can prevent the execution of unwanted code (typically as a defense against XSS), such as inline code or code from external origins. While this would be an effective defense against *some* of our anachronistic attacks, only the Wayback Machine and Conifer deploy a CSP on their main website. Perma and Ghost Archive have a subdocument CSP, which only applies to an iframe that embeds the recorded snapshot. HTML5's iframe sandboxes introduced security restrictions and isolation to framed content. Only Megalodon puts the embedded snapshot in a sandbox so that code execution is effectively prevented. All archives conduct some form of static server-side content rewriting when moving a website into the archive's domain. We observed the use of libraries like *Wombat* [58] or *pywb* [57] to rewrite URLs used in the original website and rehost resources inside the archive. Furthermore, five services (Wayback, Perma, Ghost Archive, and Conifer) use mechanisms that dynamically intercept requests. This can be done by either patching JavaScript networking APIs like *fetch* or *XMLHttpRequest* to redirect requests or by registering service workers for the same task. Archive.Today uses the fewest client-side defenses. However, the service strips websites of virtually all dynamic content. It only preserves the page's static content while stripping the page of any executable elements. This approach works well to preserve static web content like articles or blogs, but it is unqualified to capture a website's behavior.

Guessable snapshot URLs are a key requirement for archive anachronism attacks (ref. §4.2.1). As indicated in the second-to-last row, most archives use guessable timestamped URLs for snapshots.

[5] 魚拓のみの表示" translated as "snapshot-only" or "web capture-only" view.

**Table 4: Overview of defensive measures in popular web archiving services.**

| | | *Wayback* | *Archive.Today* | *Perma.cc* | *Megalodon* | *Megalodon "snapshot-only"* | *Ghost Archive* | *ARQUIVO* | *FreezePage* | *Conifer* |
|---|---|---|---|---|---|---|---|---|---|---|
| Archive Measures | **Root CSP** | yes | no | no | no | no | no | no | no | yes |
| | **Subdocument CSP** | yes | no | yes | no | no | yes | no | no | no |
| | **iframe** | no | no | yes | yes | no | yes | yes | yes | yes |
| | ↪ iframe sandbox | | | no | yes | | no | no | no | no |
| | **Source Rewriting** | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| | ↪ Static URL Rewriting | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| | ↪ Dyn. URL Rewriting | yes | no | yes | no | no | yes | yes | no | yes |
| | **Unguessable IDs** | no | no | yes | no | no | no | no | yes | no |
| **Script Execution Possible** | | ● | ○ | ● | ○ | ● | ● | ● | ● | ● |

Script execution is ● directly possible, ● indirectly possible, ○ not possible.

As the last row of Table 4 indicates, script execution is possible for all functionality-preserving archives despite their protective measures. FreezePage is the only service where code execution is directly possible in HTML elements or attributes. Namely, FreezePage snapshots can execute code in onerror, onload, and onclick event handlers. Before Megalodon's patches introduced an iframe sandbox, script tags nested inside SVG tags could also execute JavaScript code. The updates introduced an iframe sandbox around the recorded website. However, that security change was not applied to the snapshot-only mode, which is still prone to XSS. With the exception of Archive.Today, attackers can execute JavaScript code in all other archives by abusing a number of different blind spots and techniques. We discuss these vulnerabilities in the following sections.

## 5.3 Vulnerable Archives

Table 5 shows which web archiving services are vulnerable to which of our proposed attacks.

**Key Takeaway 3.** Despite the archives' claims to provide exact and unchangeable snapshots, *all* archiving services we investigated were vulnerable to at least one group of attacks.

*5.3.1 Evaluation of Anachronistic Attacks.* We found vulnerabilities to anachronistic attacks for seven archiving services. Archive.Today is the only service immune to this attack, as no code execution is possible in snapshots. Note that Megalodon hardened its service against script execution *after* we recorded our proof of concept. In the most recent version of Megalodon, one has to visit evasive snapshots in snapshot-only mode to see the effects of the attack.

The first group of attacks, archive anachronism, requires determining the URL a resource will receive after archiving (ref. §4.2.1). Wayback Machine, ARQUIVO, Archive.Today, and Conifer have nearest-neighbor timestamp matching mechanisms that redirect a snapshot URL with a nonexistent timestamp to the snapshot closest to the indicated time. In contrast, Archive.Today[6] and Ghost Archive[7] appear to use unguessable alphanumeric snapshot IDs. Still, both services also offer alternative URL formats. In the presence of a matching mechanism, guessing the snapshot URL of

[6] Archive.Today has interchangeable timestamped URLs and short URLs.
[7] Ghostarchive has a /longurl/ API endpoint converting short to timestamped URLs.

**Table 5: Overview of the vulnerability of Web archiving services to our attacks, with links to attack demos ▶.**

| Archive Name | Anachronistic Attacks: *Archive Anachronism* | Anachronistic Attacks: *Live-web Escape* | Archive Evasion Attacks: *CSP Stripping* | Archive Evasion Attacks: *Script Stripping* | Archive Evasion Attacks: *Server-side Cloaking*: IP / ASN | *Server-side Cloaking*: JA3 / JA4 | *Server-side Cloaking*: User-Agent |
|---|---|---|---|---|---|---|---|
| Wayback Machine | yes ▶ | yes ▶ | yes ▶ | no | yes | yes[1] | no[3] ▶ |
| Archive.Today | no | no | yes ▶ | no | no[2] | yes | no ▶ |
| Perma.cc | no | yes ▶ | yes ▶ | no | yes | yes | no ▶ |
| Megalodon | yes[4] ▶ | yes[4] ▶ | yes ▶ | no | yes | yes | no ▶ |
| Ghost Archive | yes ▶ | yes ▶ | yes ▶ | no | yes | yes | no ▶ |
| ARQUIVO | yes ▶ | yes ▶ | yes ▶ | no | yes | yes | no[3] ▶ |
| FreezePage | no | yes ▶ | yes ▶ | yes ▶ | yes | yes | no ▶ |
| Conifer | yes ▶ | yes ▶ | yes ▶ | no | yes | yes | no ▶ |

[1] Wayback's JA3's are too numerous for effective detection, but JA4 is suitable.
[2] Detecting Archive.Today's crawlers effectively requires extensive observation to gather sufficient network-level information.
[3] User agents that mention the archive's crawler can be exploited for evasion.
[4] The attack works only in Megalodon's "snapshot-only" mode.

a resource before archiving it is as easy as specifying a future timestamp, e.g., https://arquivo.pt/wayback/20500000000000/<url>, which automatically redirects to the snapshot closest to the year 2050. This makes snapshot URLs guessable and human-readable. We achieve a similar effect for Megalodon by programmatically querying its search API and extracting the newest snapshot of the desired anachronism-resource URL via JavaScript. Extracting the unchanged anachronism resource from a snapshot is the second requirement. Most archiving services have (undocumented) mechanics in their URL, which cause the archive's server to respond with the raw resource. For example, Wayback, ARQUIVO, and Conifer deliver raw resources when "*js_*" is inserted after the timestamp of a snapshot URL. In the case of Ghost Archive, we *pierce* multiple shadow DOMs with chained query selectors to retrieve the archived resource from a snapshot. We found that satisfying the last requirement—controlling URL rewriting—is often straightforward, for example, by dynamically writing URLs through string concatenation or by encoding and decoding them in Base64 (potentially multiple times) as illustrated in lines 1 and 2 below.

```
1 iframe.src = 'https://' + 'archive.org' + '/error-page';
2 fetch(atob(atob(atob('WVVoU01HTklUVFpNZVRsM1dXNU9iR051V21oa1J6bDVaVk0xTUZwWVR
  ↪ qQk1NMVozV2tkR01GcFJQVDA9'))).then(/* use response */)
```

Despite our efforts, we could not identify a method for guessing the URL a resource will receive when archived in FreezePage and Perma.cc, respectively. In other words, the services use a hard-to-guess URL format and do not offer a search function. Thus, forcefully including intra-archive resources in FreezePage and Perma snapshots is more complex than escaping the archives' boundaries and injecting a live-web resource, which we discuss next.

*Leaking into the live-Web.* Contrary to expectations, all functionality-preserving archives are vulnerable to live-web escape, our most potent attack. Only sandboxed Megalodon snapshots and Archive.Today recordings sufficiently prevent code execution that could lead to archive escapes. As previously mentioned, external requests to fetch updated content and the ability to manipulate the DOM to alter the snapshot are prerequisites for a successful live-web escape attack (ref. §4.2.2). We present the specific XSS vectors we used to achieve external requests and DOM manipulation for the archives vulnerable to live-web escapes in Table 6.

**Table 6: Initial XSS vectors to achieve external requests and DOM manipulation in archives vulnerable to live-web escape.**

| Vulnerable **Archives** | Script execution for *external requests & DOM manipulations* via **Script** elements | **Event handlers** | **SVG** with **script** | **iframe** *(same-origin, no CSP)* |
|---|---|---|---|---|
| Wayback Machine | no | no | no | yes |
| Perma.cc | no | no | no | yes |
| Megalodon | no | no | yes[1] | no |
| Ghost Archive | no | no | no | yes |
| ARQUIVO | no | no | no | yes |
| FreezePage | no | yes | no | no |
| Conifer | no | no | no | yes |

[1] Code execution is only possible in Megalodon's "snapshot-only" mode.

Freezepage does not sufficiently sanitize event handlers, making live-web escape attacks easy using one-line exploits in *onload* or *onerror* attributes of images. Megalodon snapshots pre-patch and post-patch in snapshot-only mode can leak into the live web and change themselves through a script nested in an SVG. The remaining vulnerable archives require multi-step exploits. The first step of our client-side attacks is dynamically creating an iframe sourced from a subpage of the archive that does not have a CSP, such as an error page. As a result, this iframe satisfies the SOP because it is from the same origin as the archive and is not subject to a CSP. The second step entails creating a script that fetches a remote resource in that iframe. If needed, the external URL can be Base64 encoded to evade rewriting. The fetch-response is then used to rewrite the iframe's parent document, the snapshot. Depending on the archive, different vectors were used to create a dynamic code-executing iframe. Specifically, scripts in SVG elements (Perma.cc, Ghost Archive, ARQUIVO), onerror handlers (Perma.cc, FreezePage), or simple scripts (Wayback Machine, Conifer) were used.

*5.3.2 Evaluation of Archive Evasion and Snapshot Prevention.* Suppose the most impactful attacks, the anachronistic attacks, are impossible for an archive, which applies to all web archives that are strictly content-only. If a service is immune to anachronistic attacks, adversaries can still attempt to launch archive-evasion attacks. As described in Section 4.1, in these attacks, adversaries can either stop the service from archiving their content or serve the archiving service content that is different than what regular web users would be served. In this section, we evaluate the effectiveness of evasion attacks. Refer to the right side of Table 5.

*CSP Stripping.* As all web archives we tested opt not to adopt the CSP of the pages they snapshot, they are vulnerable to our CSP stripping attack. FreezePage replaces script tags, which makes CSP directives like *script-src* ineffective. Since event handlers run on FreezePage, the same effect can be recreated using event handlers that the attacker's website disabled with the *script-src-attr='none'* CSP directive. The attacker's website now effectively disallows scripts in HTML element attributes, including *onerror* event handlers. When FreezePage—and the other archives—strip the CSP, the event handler—or scripts—run, respectively, and deface the page. Overall, CSP stripping is a convenient attack that allows defining code that is only executed in the context of web archives.

*Script Stripping.* This attack variation is only effective for FreezePage, which makes it less usable as a generic archive-evasion technique. Script stripping works because FreezePage rehosts pages with script tags replaced by custom non-executing *<was_script>* tags. Note that while Megalodon and Archive.Today disable scripts, they do so only after executing them on the server side when creating a snapshot. The archives then *freeze* how the DOM looks after the page has finished loading in the presence of the script.

*Server-side evasion.* While CSP Stripping proved a convenient technique for the evasive adversary to control how a website looks when any service rehosts it, the attack does not differentiate between different archives. The anachronistic adversary wants to allow deceivable archives to record their website while preventing archival in specific content-only archives. Server-side evasion offers individual control at the cost of requiring characteristic information about the different archiving services. Such information has to be acquired through an observatory, like ours (ref. §3).

Table 5 displays which information can be effectively used to detect the archive crawlers of the eight evaluated archiving services. We differentiate between information about IP addresses, TLS fingerprints, and User-Agent information from HTTP headers. The most broadly usable crawler characteristic is its IP address and AS number. While the archive crawlers, except for Archive.Today's, use only one AS (ref. Table 2), most are situated on large networks and cloud providers, making them unusable for targeted IP blocking. Wayback Machine is an exception as its ASN *INTERNET-ARCHIVE (7941)* simplifies detecting the service. Similarly, FreezePage, Ghostarchive, and Conifer use a single IP address for their crawler, making them immediately identifiable. Multiple websites must be archived to detect the remaining IP addresses of the other services. We observed a stable set of three IP addresses for Perma.cc and Meghalodon after three archiving requests, a second IP for ARQUIVO after five requests, and 19 IP addresses for Wayback's crawlers after 21 archived pages (ref. Figure 2). Archive.Today is rotating its crawlers' IPs and ASNs, making the detection of their crawlers prone to errors.

We discovered that TLS fingerprints qualify for archive crawler detection. Figure 2 displays the unique TLS fingerprints we recorded for the archiving crawlers that visited our observatory. Wayback Machine's rotation of cipher suits (ref. §3.5) creates a myriad of JA3 fingerprints, three orders of magnitude higher than the other archives after our 195 archiving requests. JA4 provides a much more manageable amount of archiving crawler TLS fingerprints after one to 25 archiving requests, depending on the archiving service. Overall, the combination of IP address, ASN, and JA4 reliably reveals all archive crawlers and can, therefore, be used by attackers for server-side evasions. Due to its distributed network infrastructure, Archive.Today is more challenging to detect.

## 6 Related Work

Prior work recognizes web archiving services, particularly the Internet Archive's Wayback Machine, as important tools for longitudinal, reproducible, and retrospective web research. Nikiforakis et al. [43] utilized snapshots of websites to research trends in JavaScript inclusion from 2001 to 2010 [43]. Lerner et al. [36] retrospectively analyzed the evolution of third-party web tracking behaviors from 1996. Stock et al. [52] used the Wayback Machine to conclude that websites were vulnerable to a novel class of XSS eight years before the vulnerability was first mentioned. Using web-archive data, Amos et al. [7] compared privacy policies over time, Iqbal et al. [27] studied the evolution of ad-blocker filter lists, and Scheitle et al. [49] investigated the stability of top website lists used in web-related research studies. Recently, Hantke et al. [21] studied security measurements using snapshots as an alternative to conventional live measurements. These works highlight the importance of creating detailed and unchangeable snapshots of websites over time.

Other works show the challenging process of investigating the inner workings of web archives. Ogden et al. [44] conducted a pilot study on how the "Save Page Now" feature of the Internet Archive creates snapshots of websites, finding its inner workings largely opaque. The observatory that we presented in this paper allowed us to shed light on the "invisible" parts (i.e., the crawlers dispatched to archive one's website) of popular web archiving services.

Ainsworth et al. [5] were the first to notice incidental discrepancies between the live Web and the archived Web, showcasing the limitations of current archives. After finding accidental archive escapes into the live web [36], Lerner et al. [35] were the first to write about security issues in the Wayback Machine that can lead to changing past records. The authors found that faulty URL rewriting can lead to accidental leaks of live web content into snapshots. Additionally, subresources that initially fail to archive can later be added by attackers. Both flaws can be abused to alter archived snapshots retrospectively. Follow-up work by Watanabe et al. [56] focuses on the security implications of the SOP when web services rehost websites under one domain. They propose various attacks, including a persistent Man-in-the-middle where an *evil.js* script the SOP would normally block is allowed to execute once it is brought under the archive's domain, as well as privilege abuse attacks aiming at accessing a user's camera or other hardware, and attacks that steal user credentials or browser history.

The study by Lerner et al. motivated our work [35]. The key difference is that we propose entirely new threat models and corresponding attacks based on them. Instead of third-party attackers identifying and abusing live-web leaks in third-party sites, we investigated attackers who want to stop or control the archiving of their websites. These attackers have complete control over their own websites and can, therefore, include code and data that abuses all exploitable corner cases of each archive's logic. The work by Watanabe et al. [56] mainly focuses on web rehosting in general, and the attacks discussed were purely theoretical. Our practical attacks and focus on web archiving services allow us to pinpoint the shortcomings of the web archiving ecosystem more accurately.

## 7 Discussion

All web archives aim to truthfully preserve past content, as stated in their mission statements. Content-only archives take a security-first approach, sacrificing interactivity for simplicity and increased security. Functionality-preserving archives, however, document technical properties of past web applications, which is essential for research on historical web security and code analysis. Achieving these goals securely is challenging, as preserving client-side functionality can introduce vulnerabilities.

Our study identified multiple security measures implemented by archives (ref. §5.2), all of which we were able to circumvent in archives that permitted script execution (ref. §5.3). Some archives attempted to mitigate risks through sandboxing mechanisms, such as iframe restrictions and CSPs, to prevent XSS. However, inconsistent use of CSP on subpages of the archives reduced its effectiveness.

We also observed characteristics that make server-side detection of archive crawlers challenging. Archive.Today stands out with its diverse infrastructure encompassing the largest number of IP addresses and ASNs. The service also uses various user agent strings mimicking real browsers. However, TLS fingerprinting allows detection of their crawlers. The variable TLS client configurations that the Wayback Machine employs were unexpected and resulted in a large number of TLS fingerprints. Nevertheless, the service uses only one specific ASN, making the service immediately recognizable to malicious web servers.

*Implications of our findings.* We have shown that the crawlers of all archives can be effectively detected with a few weeks' worth of data. This capability, combined with the more impactful anachronistic attacks we were able to execute on all script-executing archives, creates the biggest threat to the archives' main goal: faithful representation of past websites. An evasive adversary can focus on individual vulnerable archives and evade archival by all others they cannot exploit. This forces potential readers of the adversary's pages' snapshots to use archives that the adversary can control. The takeaway message is that, in light of these attacks, web archives cannot be fully trusted since adversaries can change the content of their snapshots. Even though some of our attacks have a client-side footprint (e.g., live-web leaks and anachronistic attacks referencing future archived resources), these footprints are invisible to regular users of the archive and would be hard to identify even for experts who are capable of analyzing the DOM of an archived page.

*Limitations and Future Work.* False positives are possible when an evasive adversary attempts to identify web-archive crawlers based on client characteristics. Our observatory does not aim to be a plug-in solution to block archive crawlers, but rather the first attempt to understand the "invisible" server-side infrastructure that these services operate and to what extent attackers can evade or abuse this infrastructure. A concrete future direction is designing a web archive service that is hardened against our identified server-side and client-side attacks without compromising on the functionality of the archived websites. Additionally, access-control mechanisms for web archives are a promising future direction where websites can transparently opt out of being archived (as opposed to trying to detect and evade web archiving services). Our work can hopefully encourage more research in that direction.

## 7.1 Securing Existing Web Archives

Gaps in CSP coverage and insufficient code rewriting were the most common factors enabling our client-side attacks. When rewriting a website's code, future archives must account for obfuscation techniques such as string concatenation and Base64 encoding. Additionally, since the Same-Origin Policy permits interactions between pages with and without CSP, a consistent CSP must be present across all subpages of a web archiving service.

Unguessable snapshot URLs are required to prevent anachronistic attacks. These attacks depend on the ability to predict and prematurely reference future archive URLs. While adopting an unguessable URL scheme is straightforward for future captures, retroactively rewriting existing URLs remains challenging.

To mitigate server-side evasion attacks, employing a diverse set of crawlers, ideally distributed across multiple autonomous systems, can significantly impede detection and blocking by live websites.

## 7.2 Disclosure and Ethical Considerations

We contacted all archiving services via email in early April 2025. Seven of the eight services responded and requested full details. Three services—Perma.cc, the Internet Archive, and Archive.Today—followed up with additional discussions about possible mitigations.

Our findings have already led to a real-world impact. The Internet Archive and Archive.Today acknowledged our findings and are engaging with us directly. Perma.cc has already rolled out a series of targeted patches in response to our CSP escape attacks [e.g. 31–33] and shared their full internal incident report with us. Our discussions with the archives and the evident dedication to urgently addressing the issue we brought to light underscore that the archiving services take our findings seriously. Interested readers can refer to Appendix A.2 for further details on the disclosure process.

*Ethical Considerations.* For the observatory study, we archived only our own websites and maintained detailed records of the archived URLs, in case any archive required deletion. To reduce strain on the archiving services, we stretched out the experiments over 16 days and artificially limited our study to approximately one archival request per hour. This work focuses on attacks enabled by fundamental design choices and technical limitations of current web archiving services. Our goal is to ultimately improve the archives' robustness against evasion and anachronistic manipulation. Due to ethical considerations, attacks against the servers of web archiving services were entirely out of scope.

# 8 Conclusion

In this work, we studied the intrinsic characteristics of eight of the most popular web archiving services and showed that we can detect their archiving crawlers. We have proposed five attacks against web archives that an evasive, anachronistic publisher can leverage to control what archives store about their online presence. Combining powerful anachronistic attacks against specific services with archive evasion attacks against the others ultimately creates *"the power to never be wrong,"* as adversaries can arbitrarily change their past archived content. To our knowledge, we are the first to move from accidental vulnerabilities in snapshots to exploring evasion channels that attackers can intentionally create on their websites. We demonstrated that all in-scope web archives are vulnerable to one or more of our attacks, enabling us to control what content the services can archive. For anachronistic attacks, we retained control over past snapshots for seven of the eight archiving services.

Faithfully archiving yesterday's Web is essential for the preservation of historical information and to enable web security research. Our findings will help harden web archiving services against intentional server-side and client-side attacks that impact their primary goal of truthfully preserving the past.

**Availability.** We have recorded extensive demos for all presented attacks against all services and made them available in Section 5.1. Our archive observatory implementation is publicly available on Zenodo[8] and GitHub[9]. We will only share the collected data on web archive crawlers with bona fide, established researchers to prevent misuse.

## Acknowledgments

We thank our shepherd and the anonymous reviewers for their valuable comments and suggestions. We gratefully acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972. This work was also supported by the Army Research Office (ARO) under grant W911NF-24-1-0051 as well as the National Science Foundation (NSF) under grants CNS-1941617 and CNS-2126654.

## References

[1] Lawrence Abrams. 2024. Internet Archive hacked, data breach impacts 31 million users. https://web.archive.org/web/20250630153136/https://www.bleepingcomputer.com/news/security/internet-archive-hacked-data-breach-impacts-31-million-users/. (2025-06-30).

[2] Abstract. 2025. IP Geolocation API. Online https://www.abstractapi.com/api/ip-geolocation-api. (2025-03-24).

[3] Aaron M Adams, Xiang Chen, Weidong Li, and Chuanrong Zhang. 2023. Normalizing the pandemic: exploring the cartographic issues in state government COVID-19 dashboards. *Journal of Maps* 19, 1 (2023), 1–9.

[4] Affility Co.,Ltd. 2025. 株式会社アフィリティー. Online https://www.affility.co.jp/. (2025-03-26).

[5] Scott G. Ainsworth, Michael L. Nelson, and Herbert Van De Sompel. 2015. Only One Out of Five Archived Web Pages Existed as Presented. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media - HT '15*. ACM Press.

[6] John Althouse. 2025. TLS Fingerprinting with JA3 and JA3S. Online https://web.archive.org/web/20250323120934/https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967/. visited (2025-03-24).

[7] Ryan Amos, Gunes Acar, Eli Lucherini, Mihir Kshirsagar, Arvind Narayanan, and Jonathan Mayer. 2021. Privacy policies over time: Curation and analysis of a million-document dataset. In *Proceedings of the Web Conference 2021*. 2165–2176.

[8] Adam Barth. 2011. The Web Origin Concept. RFC 6454. doi:10.17487/RFC6454

[9] Bit Flip LLC. 2025. Dark Visitors: Bytespider. Online https://darkvisitors.com/agents/bytespider/. (2025-03-28).

[10] Cloudflare. 2025. JA3/JA4 fingerprint. Online https://developers.cloudflare.com/bots/concepts/ja3-ja4-fingerprint/. visited (2025-03-24).

[11] Software Freedom Conservancy. 2025. Selenium. Online https://www.selenium.dev/. (2025-03-19).

[12] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. 2003. A large-scale study of the evolution of web pages. In *Proceedings of the 12th international conference on World Wide Web*. 669–678.

[13] FingerprintJS, Inc. 2025. GitHub: fingerprintjs. Online https://github.com/fingerprintjs/fingerprintjs. (2025-03-19).

[14] FoxIO-LLC. 2025. GitHub: ja4. Online https://github.com/FoxIO-LLC/ja4. visited (2025-03-24).

[15] Chris Freeland. 2024. Internet Archive Services Update: 2024-10-21. https://blog.archive.org/2024/10/21/internet-archive-services-update-2024-10-21/.

[16] FreezePage. 2025. Freeze Any Web Page. Online https://www.freezepage.com/. (2025-03-27).

[17] Ghostarchive. 2025. Ghostarchive, a website archive. Online https://ghostarchive.org/. (2025-03-27).

[18] Dion Hoe-Lian Goh and Peng Kin Ng. 2007. Link decay in leading information science journals. *Journal of the American Society for Information Science and Technology* 58, 1 (2007), 15–24.

[19] C. Douglas Golden. 2022. NYT Makes Stealth Edits to Elon Musk Piece Tainting Him with Apartheid Smears Writer Lashes Out at Critics. https://web.archive.org/web/20250717060102/https://www.westernjournal.com/nyt-makes-stealth-edits-elon-musk-piece-tainting-apartheid-smears-writer-lashes-critics/.

[20] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. 2018. Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Proc. of the World Wide Web Conference*. ACM Press, 309–318.

[21] Florian Hantke, Stefano Calzavara, Moritz Wilhelm, Alvise Rabitti, and Ben Stock. 2023. You Call This Archaeology? Evaluating Web Archives for Reproducible Web Security Measurements. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[22] James L. Quarles III and Richard A. Crudo. 2014. WayBack to the Future: Using the Wayback Machine in Patent Litigation. https://web.archive.org/web/20241114043112/https://www.americanbar.org/groups/intellectual_property_law/publications/landslide/2013-14/january-february/wayback-future/.

[23] Internet Archive. 2025. Internet Archive: Digital Library of Free & Borrowable Texts, Movies, Music & Wayback Machine. https://archive.org/. Accessed: 2025-01-11.

[24] Internet Archive. 2025. InternetArchiveBot: A Wikipedia bot that fights. https://github.com/internetarchive/internetarchivebot. Accessed: 2025-01-11.

[25] Internet Archive. 2025. Wayback Machine. Online https://web.archive.org/. (2025-03-27).

[26] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. 2016. Cloak of visibility: Detecting when machines browse a different web. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 743–758.

[27] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *Proceedings of the 2017 Internet Measurement Conference*. 171–183.

[28] Dana Kennedy. 2022. Washington Post issues two corrections to Taylor Lorenz article that had already been stealth-edited . https://web.archive.org/web/20250331023742/https://nypost.com/2022/06/04/washington-post-adds-two-corrections-to-taylor-lorenz-piece/.

[29] Brian Kondracki, Johnny So, and Nick Nikiforakis. 2022. Uninvited Guests: Analyzing the Identity and Behavior of Certificate Transparency Bots. In *Proceedings of USENIX Security Symposium (USENIX Security)*.

[30] Martijn Koster, Gary Illyes, Henner Zeller, and Lizzi Sassman. 2022. Robots Exclusion Protocol. RFC 9309. https://datatracker.ietf.org/doc/html/rfc9309

[31] Kreymer, Ilya. 2025. GitHub: wabac.js—Pull Request #243. Online https://github.com/webrecorder/wabac.js/pull/243. (2025-07-03).

[32] Kreymer, Ilya. 2025. GitHub: wabac.js—Pull Request #245. Online https://github.com/webrecorder/wabac.js/pull/245. (2025-07-03).

[33] Kreymer, Ilya. 2025. GitHub: wabac.js—Pull Request #246. Online https://github.com/webrecorder/wabac.js/pull/246. (2025-07-03).

[34] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A research-oriented top sites ranking hardened against manipulation. In *26th annual network and distributed system security symposium (NDSS '19)*. doi:10.14722/ndss.2019.23386

[35] Ada Lerner, Tadayoshi Kohno, and Franziska Roesner. 2017. Rewriting History: Changing the Archived Web from the Present. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[36] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016. In *Proceedings of the USENIX Security Symposium*.

[37] Raizel Liebler and June Liebert. 2012. Something Rotten in the State of Legal Citation: The Life Span of a United States Supreme Court Citation Containing an Internet Link (1996-2010). *Yale JL & Tech.* 15 (2012), 273.

[38] Meng Luo, Pierre Laperdrix, Nima Honarmand, and Nick Nikiforakis. 2019. Time Does Not Heal All Wounds: A Longitudinal Analysis of Security-Mechanism Support in Mobile Browsers. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*.

[39] Majestic. 2025. About MJ12Bot. Online https://www.mj12bot.com/. (2025-03-28).

[40] John Markwell and David W Brooks. 2003. "Link rot" limits the usefulness of web-based educational materials in biochemistry and molecular biology. *Biochemistry and Molecular Biology Education* 31, 1 (2003), 69–72.

[41] Ryan Mills. 2021. USA Today Let Stacey Abrams Stealth Edit Op-Ed to Downplay Support for Georgia Boycotts. https://web.archive.org/web/20240603035109/https://www.yahoo.com/news/usa-today-let-stacey-abrams-161435000.html.

[42] monperrus. 2025. GitHub: crawler-user-agents. Online https://github.com/monperrus/crawler-user-agents/tree/283a9df01b. (2025-03-16).

[43] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2012. You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 736–747.

[44] Jessica Ogden, Edward Summers, and Shawn Walker. 2024. Know(Ing) Infrastructure: The Wayback Machine as Object and Instrument of Digital Research. *Convergence: The International Journal of Research into New Media Technologies* 1 (2024).

[45] Perma.cc. 2025. "Websites change. Perma Links don't.". Online https://web.archive.org/web/20250316210209/https://perma.cc/. (2025-03-26).

[8] Zenodo https://zenodo.org/records/17190361

[9] GitHub https://github.com/robinki/archive-observatory

[46] Stijn Pletinckx, Thanh-Dat Nguyen, Tobias Fiebig, Christopher Kruegel, and Giovanni Vigna. 2023. Certifiably vulnerable: Using certificate transparency logs for target reconnaissance. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 817–831.
[47] Sebastian Roth, Timothy Barron, Stefano Calzavara, Nick Nikiforakis, and Ben Stock. 2020. Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies.. In *NDSS 2020*.
[48] salesforce. 2025. GitHub: ja3. Online https://github.com/salesforce/ja3. visited (2025-03-24).
[49] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D Strowes, and Narseo Vallina-Rodriguez. 2018. A long way to the top: Significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference 2018*. 478–493.
[50] Singer, Ethan. 2025. New York Times: Thousands of U.S. Government Web Pages Have Been Taken Down Since Friday. Online https://archive.is/R8iq0. (2025-07-03).
[51] StatCounter. 2019. Browser Market Share Worldwide. Online: http://gs.statcounter.com/browser-market-share.
[52] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. 2017. How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security. In *26th USENIX Security Symposium (Usenix Sec'17)*.
[53] Chris Tsoukaladelis, Brian Kondracki, Niranjan Balasubramanian, and Nick Nikiforakis. 2024. The Times They Are A-Changin': Characterizing Post-Publication Changes to Online News. In *Proceedings of the IEEE Symposium on Security and Privacy (IEEE S&P)*.
[54] ultrafunkamsterdam. 2025. GitHub: undetected-chromedriver. Online https://github.com/ultrafunkamsterdam/undetected-chromedriver. (2025-03-19).
[55] David Y Wang, Stefan Savage, and Geoffrey M Voelker. 2011. Cloak and dagger: dynamics of web search cloaking. In *Proceedings of the 18th ACM conference on Computer and communications security*. 477–490.
[56] Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Tatsuya Mori. 2020. Melting Pot of Origins: Compromising the Intermediary Web Services That Rehost Websites. In *Proceedings 2020 Network and Distributed System Security Symposium*. Internet Society.
[57] webrecorder. 2025. GitHub: pywb 2.8. Online https://github.com/webrecorder/pywb. (2025-03-26).
[58] webrecorder. 2025. GitHub: Wombat. Online https://github.com/webrecorder/wombat. (2025-03-26).
[59] Mike West and Antonio Sartori. 2025. *Content Security Policy Level 3*. W3C Working Draft. W3C. https://www.w3.org/TR/2025/WD-CSP3-20250206/.
[60] Wikipedia. 2025. Wikipedia:Link rot. https://en.wikipedia.org/wiki/Wikipedia:Link_rot. Accessed: 2025-01-11.
[61] Savvas Zannettou, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, and Gianluca Stringhini. 2018. Understanding web archiving services and their (mis) use on social media. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 12.
[62] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. 2021. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1109–1124.
[63] Ke Zhou, Claire Grover, Martin Klein, and Richard Tobin. 2015. No more 404s: predicting referenced link rot in scholarly articles for pro-active archiving. In *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*. 233–236.
[64] Jonathan Zittrain, Kendra Albert, and Lawrence Lessig. 2014. Perma: Scoping and addressing the problem of link and reference rot in legal citations. *Legal Information Management* 14, 2 (2014), 88–99.

# A Appendix

## A.1 Theoretical Attacks

*Delete Snapshot* is a theoretical variation of the archive anachronism attack in reverse order. The attacker built their website to use a resource from the archive to control the look of the website's snapshot, as displayed in Figure 6. The page defaults to a defaced appearance when the resource is not present. They request that the resource be deleted via the archive's *delete-snapshot* functionality to deface the snapshot at a time of the attacker's choice.

## A.2 Disclosure Details

We contacted each concerned archiving service in early April 2025 via email, informing them about our study and offering them the full manuscript of our work. All except one have acknowledged our findings and requested further details. Three services—Perma.cc, the Internet Archive, and Archive.Today—followed up with additional discussions about possible mitigations. Please find a detailed timeline of our disclosure process below.

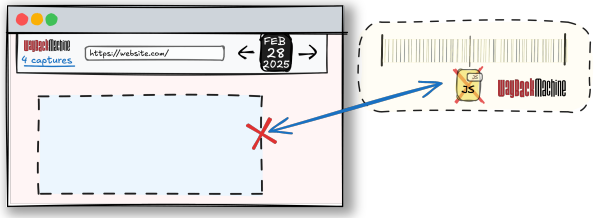


**Figure 6: Client-side cloaking by deleting an already archived subresource.**

- *April 11, 2025* We reached out to all studied web archiving services, describing who we are, giving a high-level overview of our goals and findings, and requesting permission to share the full results with them. We received positive answers, i.e., the wish to receive our findings, from everyone except Megalodon.jp, which never responded to our email.
- *April 15, 2025* We finalized the manuscript and submitted this paper to CCS. To the services that had positively responded to our initial email, we sent full details of the attacks described in this paper, along with an explanation of how each service was susceptible.
- *April 15, 2025* Archive.Today and Perma.cc responded on the same day, thanking us for our report, providing some early feedback (in the case of Archive.Today) on why they do what they do, and promising to follow up with more information.
- *April 24, 2025* We received another answer from Perma.cc. They considered the issue of server-side archive evasions to be "hard to avoid." They requested more information on the live-leak attacks, which they felt should be patched.
- *April 25, 2025* Our team sent Perma.cc additional information, including code snippets and a proof-of-concept that they could use to recreate our live-leak attacks.
- *April 30, 2025* We received an email with an extended response from Perma.cc describing how they addressed our live-leak attacks. Their response included substantial technical details and forensic information, including reasons for why they were vulnerable in the first place. They pointed us to patches on GitHub [31, 32] that were pushed as a result of our reports and asked if we had discovered additional attacks beyond the ones described in our paper (we had not).
- *May 14, 2025* Given the prominence of the Internet Archive, we followed up on our email by sending them our findings to inquire whether they had a chance to review it. On the same day, we received a response from the Internet Archive letting us know that they were still considering out findings and inquiring as to the eventual publication date of this paper.