Published at the 22nd ACM Conference on Computer and Communications Security (CCS 2017).
More information at https://rewritinghistory.cs.washington.edu/.

Rewriting History: Changing the Archived Web from the Present

Ada Lerner[*]
Wellesley College
alerner@wellesley.edu

Tadayoshi Kohno
Paul G. Allen School
of Computer Science & Engineering
University of Washington
yoshi@cs.washington.edu

Franziska Roesner
Paul G. Allen School
of Computer Science & Engineering
University of Washington
franzi@cs.washington.edu

ABSTRACT

The Internet Archive's Wayback Machine is the largest modern web archive, preserving web content since 1996. We discover and analyze several vulnerabilities in how the Wayback Machine archives data, and then leverage these vulnerabilities to create what are to our knowledge the first attacks against a user's view of the archived web. Our vulnerabilities are enabled by the unique interaction between the Wayback Machine's archives, other websites, and a user's browser, and attackers do *not* need to compromise the archives in order to compromise users' views of a stored page. We demonstrate the effectiveness of our attacks through proof-of-concept implementations. Then, we conduct a measurement study to quantify the prevalence of vulnerabilities in the archive. Finally, we explore defenses which might be deployed by archives, website publishers, and the users of archives, and present the prototype of a defense for clients of the Wayback Machine, ArchiveWatcher.

CCS CONCEPTS

• **Information systems → Digital libraries and archives**; • **Security and privacy → Web application security**;

KEYWORDS

web archives; web security

1 INTRODUCTION

The Wayback Machine is a publicly browsable web archive which has cataloged and preserved a collection of over 286 billion web pages over the period from 1996 to 2017 [26]. Like other web archives, which use similar techniques and technologies, the Wayback Machine allows clients using ordinary web browsers to access snapshots of past websites through a web interface[1], enabling ordinary citizens as well as technical experts to see how the web has changed and what it once contained. These archival snapshots of websites are rendered in HTML, Javascript, and CSS just like the modern web, preserving not only their content but also their client-side dynamic behaviors, making them a rich cultural and technical preserve.

The Wayback Machine is frequently used in a variety of contexts critical to our free society, including scholarly articles, journalism, and legal proceedings. Scientists may cite archived snapshots in their scientific papers to increase the durability of their references [19, 41], while journalists have used archives to understand how websites such as official government pages have changed [38], and lawyers often use archival snapshots as evidence in legal cases, including civil and criminal cases, administrative proceedings, and patent litigation (e.g., [1, 2, 4, 40]). While other researchers have studied inaccuracies in the Wayback Machine which arise accidentally, we observe that these socially and financially important uses suggest incentives to *intentionally* manipulate archives after the fact. For example, governments might want to suppress or change historical information, companies might want to manipulate evidence of prior art in a patent case, organizations might want to hide evidence of past wrongdoing, and news sources might want to manipulate source material for their reporting.

To our knowledge, this paper is the first to investigate the technical vulnerabilities and attacks that might be used to perform such intentional manipulation. That is: how might attackers attempt to rewrite history? How might they intentionally cause clients who view the archive to see archived websites with content, appearance, and behavior that are different from the actual website at archival time? We analyze the way that the Wayback Machine functions, finding that in fact, there are several types of vulnerabilities which would allow an attacker today to take full control of clients' views of snapshots. For example, snapshots sometimes cause clients to accidentally mix content from the live web into an archived page, allowing servers on the live web to inject content or code into clients' views of the archive. Our attacks are global — they affect the appearance and behavior of snapshots for all visitors, and they do not involve the direct compromise of archival or publisher servers or databases.

We demonstrate the viability of our attacks with proofs-of-concept. For example, we demonstrate the ability to inject arbitrary Javascript code into client views of archival snapshots, allowing us to modify text, images, styling, and behavior, subtly or completely rewriting the web of the past. Figure 1 shows such an attack, in which we took complete control of a snapshot of reuters.com from 2011.[2]

We then quantify the prevalence of the types of vulnerabilities we discovered, seeking them in the wild through a measurement study of archived websites. We find that vulnerabilities to our attacks are very common: over snapshots of the Top 500 most popular websites

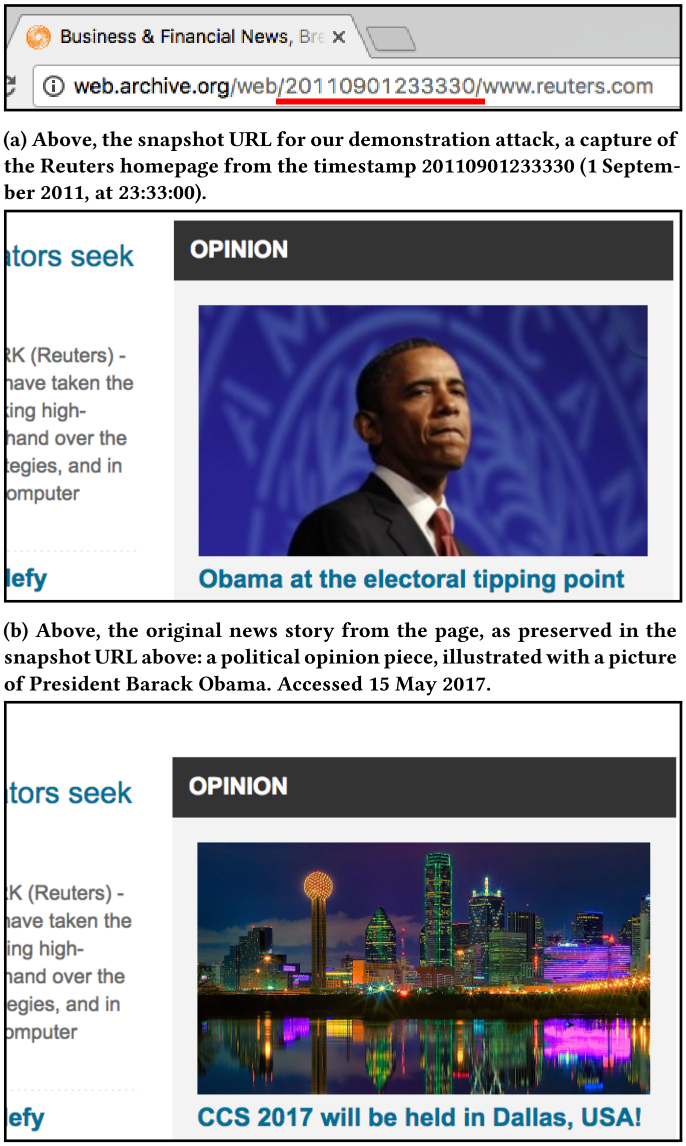
[*] This work was performed while Dr. Lerner was a PhD Candidate at the Paul G. Allen School of Computer Science at the University of Washington.

[1] https://web.archive.org/

[2] For ethical reasons, we disabled our attacks after showing that they worked.

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3134042



(a) Above, the snapshot URL for our demonstration attack, a capture of the Reuters homepage from the timestamp 20110901233330 (1 September 2011, at 23:33:00).

(b) Above, the original news story from the page, as preserved in the snapshot URL above: a political opinion piece, illustrated with a picture of President Barack Obama. Accessed 15 May 2017.

(c) Above, we used an Archive-Escape Abuse attack (Section 5.1) to replace the above article with incorrect content, so that clients would see CCS 2017's cover image and a 6-year-early prediction of CCS 2017's host city rather than the correct election opinion piece.

Figure 1: We enabled this attack only or the purposes of obtaining this demonstration screenshot, and disabled the attack after determining that it worked.

of the past 20 years, 74% contain some vulnerability which exposes the snapshot to complete control by an attacker (65% for URLs sampled from the Top Million). Additionally, we perform these same measurements over a set of website snapshots which have been cited in legal contexts such as court decisions, administrative decisions, and documents filed as trial court and appellate briefs, finding that 37 domains referenced in the 991 legal documents we examined are vulnerable to an attack which would provide complete control to some attacker over the way clients view the snapshot. We note that we are unaware of any attackers who have used these vulnerabilities for malicious purposes in practice — rather, our measurements show that a large fraction of sites are or were vulnerable to such attacks, suggesting that the consumer of web archives should exercise caution.

While an instance of our attacks may be evident upon detailed technical inspection of the way a client renders a snapshot, they are likely to be completely invisible to less technical users of the archive. Even when investigated by technical experts, attackers may have plausible deniability, since modern content can and does become intermingled with archival content in many benign cases [16, 30]. We explore a variety of defenses that could help clients see correct views of snapshots, and we design and build ArchiveWatcher, an end-user defense which demonstrates a subset of our defensive techniques. Our defense focuses on highly motivated users of the archive, aiming to demonstrate techniques which may aid individuals, such as expert witnesses and fact checkers in legal and journalism contexts, in determining when an archived view of a website can be reliably cited.

This paper makes the following contributions:

- We analyze the Wayback Machine in order to identify vulnerabilities which enable adversaries to manipulate clients' view of archival snapshots (Section 4).
- We develop attacks which exploit these vulnerabilities, exploring how an adversary can change the appearance and behavior of snapshots seen by all visitors to the archive, even years after the snapshot was captured. We execute proofs-of-concept of our attacks against real snapshots in the Wayback Machine (Section 5).
- We measure the prevalence in the wild of vulnerabilities which enable our attacks, finding that they are quite common, including a number of vulnerabilities which affect snapshots cited in legal cases and decisions (Section 6).
- We explore the space of possible defenses which might be deployed by archives, website publishers, and end-users, and we build an end-user defense, ArchiveWatcher, that detects and blocks vulnerabilities to our attacks (Section 7).

Before the publication of this paper, we have disclosed these vulnerabilities to the Wayback Machine, and made our defense, ArchiveWatcher, publicly available. Links to the code for ArchiveWatcher, along with links to other artifacts from the paper, including the TrackingExcavator tool used to make our measurements, can be found at https://rewritinghistory.cs.washington.edu.

2 BACKGROUND AND RELATED WORK

2.1 How Web Archives Work

Overview: Archival Protocol and Systems. We focus our analysis of web archives on the Internet Archive's Wayback Machine, since it is the largest publicly available web archive, with a goal of archiving as much of the public available web as possible. We note that while we developed our attacks against the Wayback Machine and did not test them against other archives, our techniques form an intellectual basis for understanding how other archives and systems which similarly rehost content could be manipulated. For example, other archives follow the same pattern as the Wayback Machine of hosting mutually distrustful content from the same domain, and this pattern results directly in vulnerabilities that our attacks exploit. We discuss the generality of our results in more detail in Section 4.4. In this section, we explain the design of the Wayback Machine in order to form a background for how the design of web archives has led to the vulnerabilities we describe later in the paper.

The Wayback Machine consists of two major components relevant to this paper. The first is the **archive crawler**, which visits, retrieves, loads, and archives pages on the web into the archive's database. The second is the **archive front-end**, which is the system of web servers, accessible via https://web.archive.org, which allow anyone to use their browser to view the web of the past.

In this paper, we refer to the archival preservation of a top level page as an **archival snapshot**, or simply **snapshot**, and the archival copies of a page's subresources (e.g., images, scripts, CSS, etc.) as **archival captures**. Each snapshot or capture was saved at a moment in time, called its **timestamp**, which appears in its URL. For example, https://web.archive.org/web/20001110101700/http://www.ccs2000.org:80/ refers to a capture of the homepage page for the 7th CCS which was saved by the archival crawler at 10:17:00 UTC on 11 November 2000. When a web browser visits this snapshot, it does the same thing as when it accesses a normal site on the live web: it recursively downloads, parses, executes, and renders the HTML, Javascript, and CSS of the page. The only difference is that the archive plays the role of the first- and third-party web servers which originally published the the site, serving the resources that make up the snapshot.

The archive crawler performs regular crawls of a large set of pages, providing significant coverage of the web. Internet Archive's Frequently Asked Questions page does not offer details about how they find sites to crawl, but states that "crawls tend to find sites that are well linked from other sites", and that they collect pages that are "publicly available" [27]. Additionally, any person can use a form on the Wayback Machine's website **"Save Page Now"**, which "Capture[s] a web page as it appears now for use as a trusted citation in the future." This feature causes the archival crawler to immediately capture the given page or resource, including its subresources [28]. We discuss additional technical details about the Wayback Machine inline as appropriate.

2.2 How are Web Archives Used?

Web archives are used in variety of important social contexts, including legal proceedings, news articles, academic publications. We take particular interest in their use in legal proceedings for two reasons: because the integrity of the legal process is important to our free society, and because legal proceedings may motivate involved parties to launch attacks that modify evidence in their favor, such as by using the attacks described in this paper. Lawyers use web archives in a wide variety of legal contexts, such as civil lawsuits (e.g., [4]), criminal cases (e.g., [2]), administrative proceedings (e.g., [3]), federal claims court (e.g., [1]), and patent litigation (e.g., [40]), anc they may use archival evidence for various purposes, such as to demonstrate "prior art" in patent litigation [3] or to recover evidence of wrongdoing that has since been deleted from the live web.

Because of these socially important uses, users of archives should take appropriate steps to ensure that archival data they use is trustworthy and not manipulated. We emphasize that we are unaware of any attacks like the ones in this paper being used in practice. However, this work demonstrates that not only are attacks are possible (Sections 4 & 5), but also that the vulnerabilities which enable them are very common in the wild (Section 6).

2.3 Legal Guidance on Web Archives

Legal scholars have written on the evidence standards that do and should govern the admissibility of archival material. Eltgroth encouraged the use of existing evidence standards to allow "reliable evidence from the Wayback Machine [to be] admitted as any other Internet-derived proof" [21], while Gazaryan argued in 2013 argued for the need to lower the difficulty of using archival material as evidence [24]. Others have advised lawyers on best practices such as employing experts to evaluate the technical limitations of the archive [40]. These articles discuss only non-adversarial factors, while we focus on the technical aspects of adversarial manipulation rather than the legal aspects of incidental inaccuracies.

In 2007, Fagan raised the possibility of "E-Evidence Tampering", noting that archival infrastructure may be compromised, or that an archived website might be cached or archived in a compromised state [22]. Our work is different in that we consider less privileged attackers, who do not compromise the archive.

2.4 Technical Work on or with Web Archives

Computer scientists have used the Wayback Machine in research: Nikiforakis et al. measured longitudinal trends in Javascript inclusion from 2001 to 2010 [35]; Soska and Christin used archival data to develop and evaluate a method for determining which websites would become malicious over time [45]; Lerner et al. studied third-party web tracking using archival data [10]; and Hackett et al. studied the evolution of website accessibility from 1997 to 2002 [25].

Others have studied the (non-malicious) incompleteness or inconsistency of web archives (e.g., [13, 17, 31, 34]). We find in our work that the technical limitations of archives that lead to accidental incompleteness can be leveraged intentionally by adversaries.

3 THREAT MODEL

In our threat model, we consider attacks in which clients (both people and automated systems) browsing archival material are maliciously caused to see content that does not accurately reflect the the web of the past. Critically, we show that this is possible *without* requiring attacks to be launched by the archive itself, and *without* compromising website publisher or archival servers. Instead, the vulnerabilities which enable our attacks involve entirely ordinary interaction with archives, such as hosting content on domains and servers the attacker rightfully owns and requesting that the archive capture specific URLs.

[3] Patents must be original to be valid, and prior art is information published prior to a patent which might be relevant to the patent's claims of originality[5].

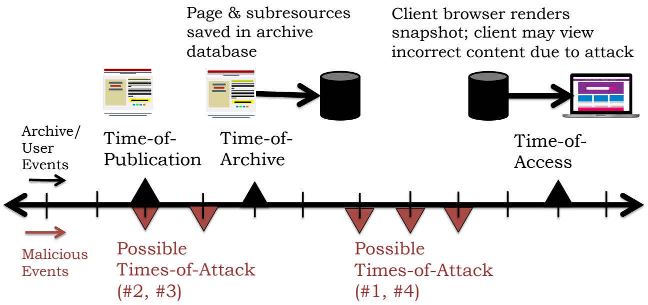

Page & subresources saved in archive database
Client browser renders snapshot; client may view incorrect content due to attack
Archive/ User Events
Time-of-Publication
Time-of-Archive
Time-of-Access
Malicious Events
Possible Times-of-Attack (#2, #3)
Possible Times-of-Attack (#1, #4)

Figure 2: A timeline depicting the (1) lifecycle of archive snapshots (top of figure) and (2) events that make up attacks against the integrity of those snapshots (bottom). The left-hand possible Times-of-Attack, before Time-of-Archive, correspond to Attacks #2 and #3, which require attacker foresight. The right-hand possible Time-of-Attack is after Time-of-Archive (but still before Time-of-Access), for Attacks #1 and #4, which do not require attacker foresight. Attacks are described in detail in Section 5.

We note that the vulnerabilities we consider can also cause non-malicious inaccuracies in the archive. These non-malicious inaccuracies have been discussed in other work (e.g., [11, 12, 44]), and our defenses (Section 7) might incidentally mitigate them. However, we focus on the ways in which our vulnerabilities can be used intentionally by malicious actors.

3.1 Definitions

We refer to a single capture of a web page as a **snapshot** or **archival snapshot**. For example, http://web.archive.org/web/20000101000000/http://example.com is a snapshot of http://example.com which aims to represent its appearance as of 1 January, 2000. We will use the terms **time-of-archive**, **timestamp**, or **archival timestamp** to refer to the time at which a particular snapshot was taken. Prior to time-of-archive, we may refer to **time-of-publication**, when the first-party website chose what content to include in its website and published it on the web. We may use these terms to refer to the domains involved in an attack and their owners at different times. For example, we may refer to the time-of-archive first-party, by which we mean "the entity which owned example.com at the time that the snapshot in question was archived," noting that ownership may change over time. Figure 2 depicts the relationship of different times in the lifecycle of a snapshot.

We will refer to as **clients** the end-users and devices that use the archival front-end to view snapshots, and who may rely upon those snapshots for information about the past. For example, a client may wish to refer to the content of http://example.com in 2000 in the course of a legal argument. To do so, they would use an ordinary browser (the **client browser**) to access the snapshot "http://web.archive.org/web/20000101000000/http://example.com". We will refer to the time at which a client accesses a snapshot as the **time-of-access**. For example, if a client examines the past contents of example.com on 19 May 2017, then 19 May is the time-of-access in this scenario. If an attack has been made against that snapshot, then the client may see a modified version of the snapshot at the time-of-access, rather than something which accurately reflects the site's appearance at time-of-archive.

We refer to the time at which an attacker takes an action to deploy an attack as the **time-of-attack**. Since our attacks sometimes require multiple actions by the attacker at different times, there may be multiple times-of-attack for a scenario. The time-of-attack may be either before or after time-of-archive, depending on the attack, and time-of-attack may precede or coincide with payload delivery to the client at time-of-access.

3.2 Attacker's Goals

Our attacks aim to change what clients see when they view archived snapshots — that is, to cause the client browser to display snapshots incorrectly, rendering content and exhibiting behavior (i.e., running code) which do not reflect the original website nor (in the case of benign archival errors) the website as it had originally been preserved in the archive.

We observe that attackers may have incentives to modify both their own and others' content in the archive. For example, if Alice accuses Bob of publishing slander on his website, then Bob may wish to retroactively remove the slander from the archive of his website. Alternatively, Alice (or an uninvolved party, such as Mallory) may frame Bob by retroactively adding slander to snapshots of his site. Attackers may be motivated by a wide variety of personal, political, legal, and financial motivations.

We emphasize that although our threat model encompasses attacks that add material to the archive's databases, the adversary must only do so legitimately, *not* by compromising those databases. That is, some attacks involve archiving new websites that we create as part of an attack.

By default, successful attacks are visible to *any* client who views that archived resource. However, attackers could also customise their attacks for different clients. For example, attackers might identify clients via techniques like browser fingerprinting [20, 23, 33, 36], or by using tracking cookies [42]. Though we note such customization is possible, we do not explore it further in this paper.

3.3 Possible Attackers

Under our threat model, the attacker owns — at time-of-attack — the domain from which the attack is mounted. For a given victim snapshot, the attacker may either be the owner of the **first-party** domain (e.g., example.com) or the owner of a **third-party** domain on that page (e.g., ads.com, serving an ad embedded inside example.com).

In a third-party attack, an attacker who controls ads.com (either at time-of-archive or in the future) may wish to modify the snapshot of example.com. To motivate a first-party attack — example.com modifying itself — we note that the ownership of domains may change over time. Thus, for example, a different entity may own example.com now than in the past, and that new owner may now wish to modify past archives of example.com. The present first-party owner might also be the same as the past owner, but seeking to alter its own past archives.

Thus, depending on the attack, an attacker must be able to serve content from one of the first- or third-party domains that make up the target snapshot, either at time-of-archive and/or at time-of-access. To meet this criterion, the attacker may either already own relevant domains, or they might purchase domains specifically to perform these attacks. They might also be able to hijack domains illicitly, e.g., through DNS poisoning. The means by which the attacker gains the ability to publish content from the domain of the vulnerable resource is orthogonal to the discussions of this paper.

4 ANALYZING THE WAYBACK MACHINE FOR VULNERABILITIES

We analyzed the Wayback Machine, surfacing three types of vulnerabilities which emerge from its design. Those types of vulnerabilities are **Archive-Escapes**, **Same-Origin Escapes**, and **Never-Archived Resources**, detailed below.

4.1 Archive-Escapes

To deliver snapshot content, the Wayback Machine plays the role of all web servers which were originally involved in serving the archived site. That is, it serves archived versions of all first- and third-party content the client requests while rendering its view of the snapshot. To cause the client to correctly request all these resources from the archive, rather than the live web, the archive performs **URL rewriting**, modifying URLs in archived HTML, Javascript, and CSS to make them refer to archived versions of the same URL. For example, the archive may find the URL http://example.com/script.js in some HTML at time-of-archive, and rewrite the HTML so that the URL instead reads http://web.archive.org/web/<timestamp>/example.com/script.js, where the timestamp of the archived script matches the timestamp of the archived HTML.

URL rewriting is not perfect, primarily because it does not account for client-side dynamically generated URLs. We find that when Javascript computes subresource URLs using computation as simple as string concatenation, then URL rewriting fails and *clients end up making requests to the live web* to load those subresources. For example, if URL rewriting fails, the client might accidentally load a live copy of example.com/script.js instead of its archived version. These live web subresources are incorporated into the client's rendered view of the snapshot, mixing live and archived content and behavior.

We refer to the request and use of live-web resources as part of a snapshot view as an **Archive-Escape**, the first of our classes of vulnerabilities. We refer to the domain contacted for live resources as the **archive-escape destination**, such that in the example above, example.com is an archive-escape destination. Whenever there is an archive-escape, the destination of that escape becomes a potential attacker, since that domain can now serve a malicious payload on the live web at the escaping URL. For example, the live copy of example.com/script.js can be replaced with a malicious payload. Note that the archive-escape destination may be the same domain as that of the victim snapshot.

4.2 Same-Origin Escapes

We discovered a second class of vulnerability, related to the fact that archives take on the role of serving both content from *all* of the domains which were involved in a snapshot at time-of-publication.

As background, browsers prevent third-parties inside `<iframe>`s from accessing or modifying data from the main page. This policy of preventing cross-origin access is called the Same-Origin Policy. So, for example, if http://example.com embeds http://ads.com in a frame, code from ads.com (running inside the frame) will be blocked by the browser from reading or influencing any parts of the page outside of its frame. This allow sites to safely embed content from third-parties within the context of their own pages. The http://ads.com attacker might embed malicious code which attempts to modify the page, but it will be blocked from doing so by the Same-Origin Policy.

The Same-Origin Policy, however, is ineffective in the archival context. Since all archived resources are loaded from the archive, this means that *all* resources making up a snapshot, including both first- and third-party resources, are loaded by the client from *a single domain*, archive.org. When this occurs, a vulnerability arises: code from the embedded frame now executes without the isolation provided on the live web by the Same-Origin Policy, allowing it to reach outside of its frame to modify any aspect of the main page. This allows an attacker to embed an attack payload inside of an `<iframe>`, where it will become active when preserved by the archive and served to clients, modifying the client's view of the containing snapshot.

4.3 Never-Archived Resources and Nearest-Neighbor Timestamp Matching

Our third class of vulnerability arises from the interaction of two properties of the Wayback Machine: its incompleteness, and its nearest-neighbor timestamp matching.

First, we discuss incompleteness. Many pages in the Wayback Machine include resources which the archive has never successfully captured. There are a variety of reasons why this might occur, including archival crawler errors or a partial unavailability of the publisher's web server at time-of-archive. For example, a snapshot's HTML might include an image, but that image has never been saved in the archive's database. When the client asks for a never-archived resource, the archive front-end responds with an HTTP `X-Archive-Wayback-Runtime-Error` header with value `ResourceNotInArchiveException`, and error code 404. Our measurements (Section 6) show that never-archived resources arise quite commonly.

Second, we discuss the archive front-end's **nearest-neighbor timestamp matching** policy. Imagine that a client requests an archived resource R at a timestamp T, and that the archive's database contains captures of R, but only with timestamps $\neq T$. When this happens, the archive will find the capture of R with timestamp as close as possible to T, and redirect the client to that version. For example, imagine a client that requests to visit a March 2005 snapshot of example.com. If example.com was never captured in March of 2005, but was captured in April, then the archive would redirect the browser (302 FOUND) to the April timestamp.

In non-malicious situations, this "nearest-neighbor" behavior allows clients to view a more complete picture of the past in the case that a snapshot's subresources were not captured at the exact moment the snapshot was. However, there is no apparent limit to the time delta permitted by nearest-neighbor timestamp matching.

Thus it is possible, for example, to request a resource from 1996 and be redirected to a capture of that resource from 2016, if no other closer timestamp exists. We refer to instances where client browsers are redirected to timestamps very far in time from the original page as **anachronisms**.

An attacker who owns the domain of a never-archive resource can abuse these observations by inserting a malicious payload as the anachronistic capture of that missing resource, which will be served to clients due to nearest-neighbor matching.

4.4 Generality

We emphasize that while we analyzed these vulnerabilities in the specific context of the Wayback Machine, our insights could form the intellectual basis for developing similar attacks to manipulate other web archive systems. These attacks, and the ideas behind them, are general due to the sharing of both (a) software and (b) design principles across web archives.

Shared Software. Other web archives frequently use the Wayback Machine's software, which is open source. For example, the Wayback Machine's web crawler (Heritrix [14]) and archive hosting/playback software (Wayback/OpenWayback[4] [15, 18]) are used by archives such as the Internet Memory Foundation ([29]), Stanford University Libraries ([46]), OpenGovData's Russia Archives ([39]), and the US Library of Congress ([32]), among at least 22 national web archives [47]. While each deployment may modify the software or deploy it differently, the intellectual basis for the attacks described in this section should apply to these other web archives. For example, as an anecdote, we spot-checked five Library of Congress archived pages, finding archive-escapes to scripts and missing script resources [37, 43]). We also found that the Library of Congress archive (a) performs nearest-neighbor timestamp matching on resource timestamps (enabling Attack #4) and (b) serves all content from the same domain, regardless of its original domain (enabling Attack #2).

Shared Design Principles. Even when code is not shared, a general lesson we take from the vulnerabilities we identified is that by rehosting and remixing web content, web archives can create unexpected situations which violate the threat model underlying web security assumptions and primitives. For example, hosting mutually distrustful content from the same domain violates a key assumption of the Same-Origin Policy. Additionally, the nature of web archives in attempting to reproduce a particular moment in time creates new assumptions that may be violated: e.g., that all resources seen by the user came from the same time, and not from the present. Our work surfaces how these assumptions – common across web archives systems – may be violated.

5 REWRITING HISTORY: OUR ATTACKS

Having discussed our vulnerabilities, we delve into the design of attacks which exploit these vulnerabilities to rewrite history. For reference in discussing these attacks, recall that Figure 2 depicts the lifecycle of a snapshot and possible attacks against it.

5.1 Attack #1: Archive-Escape Abuse

Preliminaries and Attacker. The precondition for Archive-Escape Abuse is the presence of an archive-escape vulnerability in the victim snapshot. The potential attacker is the owner of the destination of the archive-escape, to whom the client makes a request for the vulnerable resource. Because the attacker delivers the payload from their own servers (rather than via the archive) at time-of-access, we refer to this as an **active** attack.

Attack Concept. To mount this attack, the attacker (the destination of an archive-escape), publishes malicious content at the escaping URL. If the archive-escape is to a static resource like an image, then the attacker will only be able to affect that resource; if the archive-escape is a request for a script or stylesheet, then the attacker can choose arbitrary malicious code to execute.

Sequence of Events for Attack #1.

(1) The victim page is published. (Optional: If the attacker is the first-party domain wishing to enable future modifications of itself, the attacker can *intentionally* include requests which will result in archive-escapes.)

(2) The page is archived as the victim snapshot.

(3) The victim snapshot, when loaded, causes the client browser to make an archive-escape request.

(4) The attacker (who owns the domain on which the escaping script is hosted) serves malicious code in response to the archive-escape request. The malicious code runs in the client browser and modifies the appearance of the snapshot so that the client sees an inaccurate view of the page.

Proof of Concept Attack Implementation. We developed a proof-of-concept implementation of Attack #1, demonstrating the ability to attack snapshots of websites over which we have no control and which were archived years ago. We used our measurements (Section 6) to locate archive-escape vulnerabilities where the attacker domain was unowned, using `whois`. Finding that http://web.archive.org/web/20110901233330/reuters.com generates an archive-escape to http://cdn.projecthaile.com/js/trb-1.js, and that as of 19 March 2017, projecthaile.com had no owner. We purchased projecthaile.com and hosted our own version of /js/trb-1.js which modifies specific elements of the reuters.com snapshot. This attack resulted in the screenshot shown in Figure 1, in which we replaced a news article image and headline with our own.

As with all of our attacks against snapshots we do not own, we disabled the attack after confirming that it worked, so as not to disrupt the public's view of the snapshot. Additionally, we have purchased the remaining unowned domains (without hosting anything from them) for this attack to prevent any other attackers from buying and using them.

Advantages and Disadvantages of Attack #1. Attack #1 is an **active** attack, where the attacker's server delivers payloads directly to clients, allowing an attacker to modify their attack over time, customize it per client, or disable the attack entirely. However, it also means the attack is not permanent. Additionally, defenses which block archive-escapes are among the easiest for clients to deploy.

[4] OpenWayback is the community version of Wayback — Internet Archive's Wayback repository is forked from OpenWayback

5.2 Attack #2: Same-Origin Escape Abuse

Preliminaries and Attacker. Potential Same-Origin Escape attackers include all third-parties embedded in `<iframes>` at time-of-archive. However, this attack requires foresight — the attacker needs to have included their payload inside their `<iframe>` at time-of-archive, so that it can be preserved and served from the archive's database. Note that this makes Attack #2 a passive attack, since the payload is stored and delivered to the client by the archive, rather than directly from the attacker's server at time-of-access.

Attack Concept. As described above, this attack abuses the lower level of isolation which the client browser applies to frames when they are delivered from a single origin (the archive's origin) rather than multiple origins, as they are served on the live web. The first-party publisher includes the attacker in their page under the assumption that malicious code the attacker writes to deface the first-party's page will be unable to do so because of the Same-Origin Policy, and this assumption is violated in the archival context.

Sequence of Events for Attack #2.

(1) A victim site includes a third-party in an `<iframe>`, where they are now a potential Same-Origin Escape attacker.
(2) The third-party attacker publishes malicious code in its `<iframe>`.
(3) In the live web, the malicious code executes, but its effects are blocked by the browser, according to the Same-Origin Policy.
(4) The first-party page is archived as a snapshot, including the attacker's `<iframe>`.
(5) When the snapshot is loaded, both the page and the `<iframe>` are served from web.archive.org. Since they are now served from the same domain, the Same-Origin Policy no longer applies, and the malicious code in the `<iframe>` can make arbitrary modifications to client's view of the page.

Proof of Concept Attack Implementation. For Attack #2, we developed a prototype demonstration against a toy website which we created and archived for demonstration purposes. The reason is that this attack requires the attacker to be a third-party with foresight, and we do not have a third-party position on any websites we do not control which we could use to demonstrate the attack.

Thus, to demonstrate this attack, we published, on the live web, the victim page of our first-party domain, including an `<iframe>` of our third-party domain. Inside the `<iframe>`, we then deployed attack code which attempts to modify elements of the first-party page. On the live web, this attack code fails, due to the Same-Origin Policy. We then requested that the Wayback Machine "Save Page Now" for our first-party victim page, causing it to archive that page and, as part of archiving that page, also archive the attacker's `<iframe>` with its attack code. When viewing the snapshot of the victim page in the archive, both first- and third-party content are served from the same domain, causing the Same-Origin Policy to no longer apply, and allowing the third-party code to modify clients' views of the victim snapshot.

Advantages and Disadvantages of Attack #2. This attack has several strengths. First, the prerequisites for performing the attack are minimal, since all that is required is to be a third-party who can execute Javascript. Third-party frames are commonly embedded and trusted by websites, and it may even be possible to purchase advertising space in order to gain the position needed to execute the attack. Additionally, there are some third-parties who are present on a large fraction of websites (see Section 6), meaning that for certain attackers, this attack represents a huge capability to modify snapshots of a large number of websites.

However, this attack is significantly limited because the attacker must have **foresight**: Their attack code, and thus the changes they wish to cause in the client's view, must be chosen before time-of-archive, since the attack code must itself be stored in the archive.

5.3 Attack #3: "Same-Origin Escape" + "Archive-Escape"

Preliminaries and Attacker. Noting the limitation of Attack #2 requiring foresight, we consider a stronger way to use Same-Origin Escapes: Attack #3. This attack uses a Same-Origin Escape to create an intentional archive-escape, allowing the attacker to launch a later attack without foresight. Attack #3 is applicable any time Attack #2 is applicable, since it begins with a third-party in an `<iframe>` executing Attack #2 in order to create a later opportunity for Attack #1.

Attack Concept. This attack combines Attacks #1 and #2. Here, the attacker uses a Same-Origin Escape (malicious code in an `<iframe>`) to intentionally cause archive-escapes, with a destination the attacker controls, in the snapshot of the victim page. Once this has been done, the attacker is now capable of performing archive-escape abuse, immediately or at a later time.

Sequence of Events for Attack #3.

(1) The attacker must be a third-party who is embedded as an `<iframe>` on the target page as of time-of-publication.
(2) The attacker chooses a destination payload URL which they control, and embeds an archive-escape to that URL as the `src` attribute of a `<script>` tag in their `<iframe>`.
(3) The page, along with the `<iframe>`, is archived.
(4) Some time in the future, the attacker chooses and publishes a payload at the archive-escape URL.
(5) When a client browser loads the snapshot, the archived `<iframe>` is retrieved from the archive, including the script which causes an archive-escape. The browser retrieves the payload and executes it in the context of the `<iframe>`. Since the `<iframe>` is archived, it is not isolated by the Same-Origin Policy (see Section 5.2) allowing the modern attack script to cause arbitrary modifications to the client's view of the snapshot.

Proof of Concept Attack Implementation. Since Attack #3 leverage Attack #2 (Same-Origin Escape), we created a similar victim/attacker pair of testbed websites to demonstrate this attack. We again deployed attack code inside a third-party `<iframe>`, but in this case our attack code used string concatenation to create an archive-escape to the third-party domain rather than directly modifying the snapshot content directly. We then hosted the snapshot-modifying code on the live web at the third-party domain.

Advantages and Disadvantages of Attack #3. This attack allows archive-escape attacks against a page which does not naturally generate any archive-escapes to the attacker's domain, making it subject to the disadvantages of archive-escape attacks discussed above.

Since the archive-escape payload can be chosen after time-of-archive, this attack reduces a Same-Origin Escape attacker's need for foresight: they must only choose to enable a future attack by embedding a small amount of archive-escape generating code in the `<iframe>`, without the need to know how exactly they will change the snapshot in the future. An attacker such as a content delivery network or advertiser which appears on many pages could even choose to seed many pages with archive-escapes in order to preserve their ability to attack snapshots of many pages later on.

5.4 Attack #4: Anachronism-Injection

Preliminaries and Attacker. The precondition for Anachronism-Injection is a page which contains at least one resource which has *never* been captured by the archive. The potential attacker is the owner of the domain of that never-archived resource, who is in a position to publish a malicious version of that resource and cause that payload to be preserved in the archive as the resource's first (and at that point only) capture.

Attack Concept. The attacker publishes payload code to the missing-resource's URL on the live web, then uses the archive's "Save Page Now" feature to archive the payload. For example, a snapshot from 2000 might include a script capture, also from 2000. If that script has never been archived, then today, in 2017, the owner of the script's domain can publish a malicious payload at the script's URL and use the archive's "Save Page Now" feature to create a capture of the script with a 2017 timestamp. Once the missing resource has been archived, it will be the only capture of that resource in the archive (since a precondition of the attack was that the resource had *never* before been archived). As the only capture of the resource, its timestamp necessarily is (and always will be) the nearest neighbor to the timestamp requested in the victim snapshot, despite being 17 years distant. Thus the payload will be loaded in the context of the victim snapshot, as client requests are nearest-neighbor redirected to the malicious payload's timestamp. Even if more captures of the malicious resource are made afterwards, the payload will always have a timestamp that is strictly earlier, and thus which is closer to the victim snapshot's timestamp, than those subsequent captures, making the attack permanently effective.

Sequence of Events for Attack #4.

(1) A victim snapshot refers to a vulnerable resource which has never been archived.

(2) The attacker, who owns the vulnerable resource's domain, publishes an attack payload on the live web.

(3) The attacker uses the archive's "Save Page Now" feature to cause the payload to be preserved as the first and only extant capture of the vulnerable resource.

(4) When a client browses the victim snapshot, their browser makes a request for the vulnerable resource at the timestamp of the snapshot. In response, the archival front-end redirects the client browser to the malicious, anachronistic capture of the resource, since it has the timestamp closest to the requested version.

Proof-of-Concept Attack Implementation. As with Attack #1, we *could* demonstrate the Anachronism-Injection attack on snapshots of previously-archived websites over which we have no control. However, because this attack permanently alters the victim snapshot (even if our injected anachronism is not expressly malicious), we chose not to implement this attack on real victim snapshots. Instead, we test it on our own testbed websites, similarly to Attacks #2 and #3.

We note that executing this attack took careful planning, since on several occasions we deployed attack code that was slightly incorrect, forcing us to start over with entirely new victim and attacker pages, since once the attack code is archived, the attacker is unable to replace it with different attack code, since all subsequently archived code will have a timestamp farther from the victim snapshot's timestamp. However, using this attack we were able to take control of our testbed victim snapshot.

Advantages and Disadvantages of Attack #4. This attack is a passive attack, with the advantage that once the attack is in place, it becomes permanent. However, the flip side to this advantage is that the attacker cannot easily disable the attack, since the content which enables the attack has been permanently preserved in the archive's database.

Indeed, this attack's main weakness is that it is a one-time opportunity. Once the attacker has created a payload and caused it to be archived, they no longer have any way to change the behavior of that attack, since it is permanently the closest neighbor to the vulnerable resource. However, an attacker could choose to make two distinct modifications to the attack to gain the ability to continue to modify the payload over time:

(1) **Archive-escape extension**. In this version of the attack, the malicious code creates an intentional archive-escape, allowing persistent control from the present by the attacker. This version fails against archive-escape-blocking defenses.

(2) **Anachronism chaining**. In this version, in addition to performing malicious modifications of the snapshot, the payload also causes the client to make a request for the archival version of another, different URL which has never been archived. In other words, while deploying the payload, the attacker intentionally creates the preconditions for another Anachronism Injection attack, which they can exploit in the future. For example, the archived payload script attack0.js might make a request for the never-archived script attack1.js. This request will fail until the attacker changes the content of the snapshot again, at which point they host and archive `attack1.js`. This chaining can continue indefinitely (`attack2.js`, `attack3.js`, etc.).

5.5 Reflecting on Attacks

We now step back and reflect on our attacks, which are summarized in Table 1. We highlight several axes along which we can distinguish our attacks:

Passive vs. Active Attacks. Attacks differ by whether the payload is loaded from the archive itself — a **passive attack** — or from an attacker's live web server — an **active attack**. In a passive attack, the attacker is not actively involved at time-of-access. Specifically, Attacks #1 and #3, which both use archive-escapes, are active attacks,

#	Name	Requires Foresight?	Passive or Active?
1	Archive-Escape Abuse	No	Active
2	Same-Origin Escape	Yes	Passive
3	Same-Origin Escape -> Archive Escape	Yes	Active
4	Anachronism Injection	No	Passive

Table 1: A summary of the attacks we develop. Attacks requiring foresight necessitate the attacker to plant a payload (e.g., Javascript code) *before* the time-of-archive of the victim page. At the time-of-access, attacks served from an archived version of an attacker's page are passive, whereas attacks served from the attacker's server in the live web are active.

since the attacker's server is the destination of the archive-escape. By contrast, Attacks #2 and #4 deliver payloads the attacker has placed in the archive, and which are delivered to the client by the archive front-end.

Some Attacks Require Foresight. Some attacks require *foresight* on the part of the attacker. By foresight, we mean that the attacker must define the attack payload (e.g., the Javascript code to run on the snapshot when viewed by a client) at the time-of-publication of the victim page. Specifically, attacks based on origin-escapes (Attacks #2 and #3) require the attacker to plant malicious code inside an `<iframe>` on the victim page. Attacks which do not require foresight (Attacks #1 and #4) allowing the attacker to choose a payload at any time, including after time-of-archive. For example, in Attack #1, the attacker can even change this payload over time (whereas once an anachronism has been injected in Attack #4, that payload is fixed).

Partial vs. Full Control. For all attacks, vulnerabilities may permit either *partial-control* or *complete-control* attacks, depending on the type of resource the attacker controls in the specific instance of the attack. If an attacker controls static resources like text or images, the attacker can only changes those particular elements (partial-control). If an attacker controls client-side code, such as Javascript or a CSS stylesheet, the attacker can leverage that code for complete-control, gaining the ability to modify *any* part of the client's view of the snapshot, such as its text, styling, images, layout, client-side dynamic behavior, and so on. We explore the prevalence of partial-control and complete-control attacks in the Section 6.

6 MEASURING PREVALENCE OF ARCHIVE VULNERABILITIES

6.1 Measurement Methods

Measurement Tool. We used TrackingExcavator, the archival measurement tool we developed for a previous project, for our measurements [10]. TrackingExcavator is a Chrome extension which automatically visits an "Input Set" of URLs, locates them in the Wayback Machine at a requested timestamp, and collects event traces as it loads and renders those URLs. These event traces include events for all HTTP requests the browser makes, which we use to locate vulnerabilities to our attacks.

With the publication of this paper, we are releasing TrackingExcavator publicly at http://trackingexcavator.cs.washington.edu/.

Our Datasets. Our measurements include measurement traces from three sets of URLs:

For the **Top 500**, we downloaded the publicly available traces from [10]. [5] For the **Top Million**, we used historical versions of the Alexa Top Million CSV file for the years from 2010-2017, which we located in the Wayback Machine [7]. We sampled every thousandth site from those Top Million lists, such that we visited sites with popularity rank 1, 1001, 2001, ..., etc., similar to other papers that have sampled from the Top Million [42]. These traces cover a different (but sometimes overlapping) set of URLs in each timestamp year, with a trace for each site's snapshot once for each year in which it appeared in the Top 500 or our Top Million sample.

For each of our Top 500 and Top Million datasets, we report on data collected only from the archived *homepages* of each domain examined, e.g., from a snapshot of the url http://example.com. However, in Section 6.2, we report on additional measurements we performed examining other pages from the same domains, finding that sites are often vulnerable not only on their archived homepages, but also on subpages linked from the homepage.

For our **Legal URL** dataset, we searched Westlaw and LexisNexis for court decisions, court filings, and federal agency administrative decisions which contained the phrase "web.archive.org" [8, 9]. We found that both legal databases contained substantially similar results, and so used only the results from Westlaw. We then located Wayback Machine URLs cited in these materials, collecting separate lists of URLs for each category of legal proceeding (court decisions, court filings, administrative decisions). These include 119 URLs cited in 101 court decisions, 255 URLs cited in 302 appellate briefs, 159 URLs cited in 217 expert material documents, and 307 URLs cited in 371 administrative decisions. [6] We collected traces of the snapshots at the exact URLs cited in the legal materials.

Measurement Parameters. We crawled the archive from Amazon EC2 t2.large instances, rendering Chrome (running TrackingExcavator) headlessly inside a virtual frame buffer. We opened 3 tabs at once, one tab per snapshot, and remained on each snapshot for 40 seconds, which [10] found is a sufficient for snapshots to complete loading in the browser. We set TrackingExcavator to block (but still record) archive-escape requests, in order to prevent contaminating our view of the archive with live data. This means we undercount overall archive-escapes that would be seen by an ordinary browser (since we miss archive-escapes caused by other archive-escapes), making our numbers a conservative lower-bound on the archive-escapes a client will encounter in the wild.

6.2 How Often Are Archived Sites Vulnerable?

Figure 3 depicts the prevalence of all types of vulnerabilities to our attacks in the top panel, and the prevalence of vulnerabilities which allow the most powerful attacks (complete-control without foresight) in the bottom panel. This figure depicts only data from the Top 500 — the trends we found in the Top Million were similar.

Three-Fourths of Archived Sites Are Vulnerable. Considering the union of the top sites across all years, we studied 2692 distinct sites from the Top 500 and 7000 distinct sites in the Top Million.

[5] Available at https://trackingexcavator.cs.washington.edu/, Accessed 2017-03-30.

[6] In an administrative decision, a U.S. federal agency resolves lawsuit-like cases related to the agency's jurisdiction. They may replace or precede normal lawsuits.

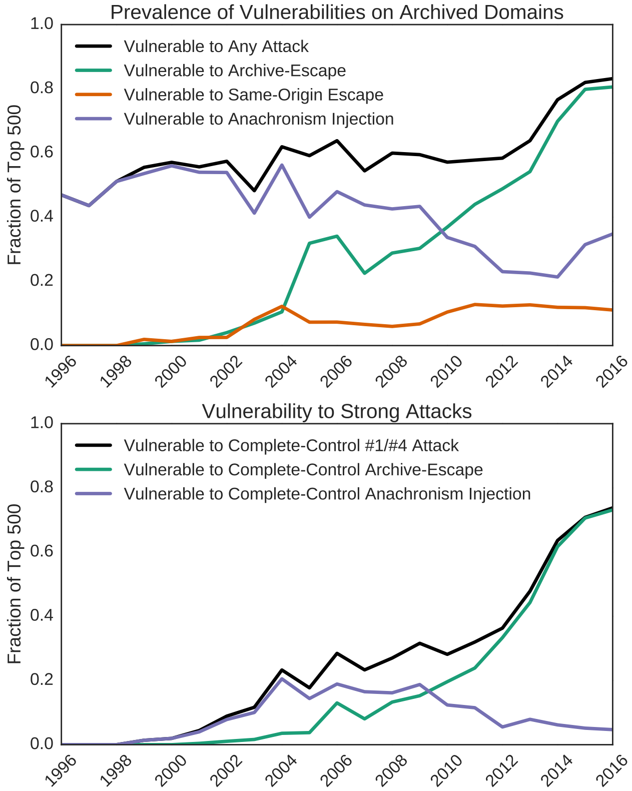


Figure 3: Top: The prevalence of vulnerabilities to our attacks across the Top 500 sites. Bottom: The prevalence of vulnerabilities to the particularly strong class of attacks which provide complete-control without foresight (Attacks #1 and #4 with script/stylesheet as vulnerable resource). Not shown: Our Top Million dataset shows very similar trends to the Top 500.

We found found that 73% of those Top 500 sites and 80% of those Top Million domains were vulnerable to one of our attacks, either now (for Archive-Escape or Anachronism-Injection vulnerabilities, which do not require foresight) or at time-of-archive (for Same-Origin Escape vulnerable snapshots, which do require foresight).

Recall that for each vulnerable snapshot, there is a limited set of domains which are capable of exploiting that vulnerability (e.g., the destination domain of an archive-escape vulnerability, or the owner of the domain of a missing resource). That is, not *anyone* can mount these attacks — only attackers who own or are able to acquire these domains. We consider the number of unowned domains (accessible to anyone) later in this section.

As shown in Figure 3, these vulnerabilities are widespread and varied in type, endangering client views of a large fraction of archived sites. Archives and their users should take care to ensure they put appropriate levels of trust in archival data, given the frequency with which they are vulnerable to manipulation.

Sites Are Vulnerable To Strong Attacks. While the top of Figure 3 considers all of our attacks, the bottom panel considers a particularly strong, category of attacks: Archive-Escape (#1) and Anachronism Injection (#4) vulnerabilities which enable complete-control. Even vulnerabilities to this strong class of attacks are quite common in the archive: 38% of Top 500 domains and 65% of Top Million domains are vulnerable.

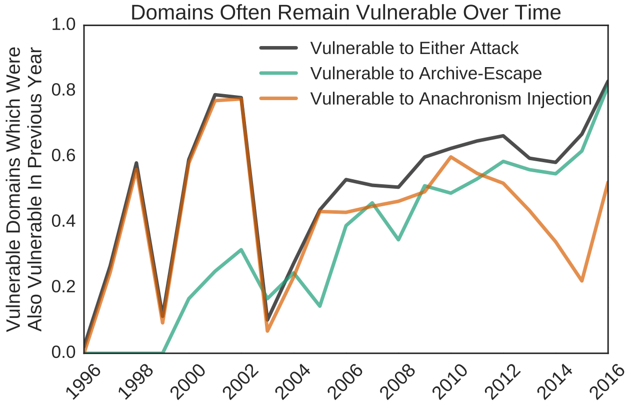


Figure 4: The increasing tendency of snapshots to remain vulnerable to our attacks across subsequent years. This figure represents the number of snapshot domains in each year whose snapshot from the previous year was also vulnerable to the given attack(s).

Prevalence of Some Vulnerabilities Has Changed Over Time. The prevalence of our vulnerabilities varies over the age of snapshots in the archive. For example, more recently captured snapshots are dramatically more likely to be vulnerable to archive-escape abuse. For example, in both the Top 500 and Top Million, the fraction of snapshot domains vulnerable to archive-escape abuse increased from 22% to nearly 80% over the period from 2007 to the present day. We believe that this trend is due to the increasing complexity of sites over the history of the web, since URL rewriting failures, which cause archive-escapes, often result in client-side dynamic behaviors in sites. As sites have grown more complex with more client-side dynamic behaviors, so have the prevalence of archive-escapes and the vulnerabilities that they cause.

Snapshot Domains Remain Vulnerable Over Archival Time. The series of snapshots of a site in the archive may span years or decades, as a site ages. We find that not only are individual snapshots often vulnerable (Figure 3), but also that many of the websites we studied remained vulnerable over long periods of time. Figure 4 shows the number of vulnerable domains in each year which were also vulnerable in the previous year. For example, of the snapshot domains which were vulnerable to Archive-Escape Abuse in 2016, about 80% of them were also vulnerable in 2015.

This type of continuous vulnerability suggests that the appearance of vulnerabilities in these sites may be due to structural elements of the way the sites are designed and published, such as publishers' choices to embed third-parties, to use client-side dynamic behavior, and to include third-party Javascript libraries. This implies both that changes in the architecture of these sites might alleviate these vulnerabilities, but also that they are unlikely to go

Potential Attacker	Number of Possible Victims
google-analytics.com	108
googletagservices.com	78
facebook.net	67
googletagmanager.com	66
doubleclick.net	59
gstatic.com	56
criteo.com	27
amazon-adsystem.com	22
newrelic.com	22
cloudfront.net	21

Table 2: The third-party domains capable of attacking the most snapshot domains we studied. Do we not suggest that any of these domains *have* or *would* deploy any such attacks.

away on their own, especially as many of the more complex aspects of the modern web may lead directly to some of our attacks.

We note that continuous vulnerability of a website may be valuable to attackers who need to modify the appearance of a particular snapshot of a website for their goals. If a large fraction of the snapshots of a website are vulnerable over time, the chances are much greater that an attacker will be able to exploit the particular snapshots needed for their goals.

Both Homepages and Subpages are Vulnerable. In addition to the other measurements described in this section, which examined only homepages, we also performed a smaller measurement of pages linked to from those homepages ("subpages"), to determine whether vulnerabilities also occur off the front page of websites.

For this measurement, we configured TrackingExcavator to visit up to 5 links on each homepage it visited in the archived 2016 top 500. It selected only links which led to snapshots of the same domain. Following this criteria, if a homepage had no within-domain links, or we were unable to follow those links for some reason, we excluded it from this analysis.

We found 236/500 domains on which we were able to follow at least one link which remained within the domain but led to a different page on that domain. Of these 236 domains, we found that 192 (81%) of them contained an archive-escape vulnerability on either a homepage or a subpage, which is roughly consistent with our larger results across the entire top 500%. For 124/236 (52%), had vulnerabilities on both the homepage and at least one subpage, 15/236 (6%) had vulnerabilities only on the homepage, and 53/236 (22%) had vulnerabilities only on subpages.

These results suggest several things. First, the fact that vulnerabilities frequently appeared in this analysis on subpages but not on homepages suggests that our main numbers may undercount the total vulnerability of the archived web, as the rest of the numbers reported in this paper are derived from measurements only of archived homepages. Second, the frequency with which vulnerabilities appear on both homepages and subpages of the same domain suggests support for our hypothesis that these vulnerabilities are often created by structural elements of websites which are used across multiple different pages and remain over time.

6.3 How Many Potential Attackers Are There?

Some Potential Attackers Have the Ability to Compromise Many Domains' Snapshots. Recall that potential attackers are those who own, or can obtain, the domains associated with vulnerabilities. There are a total of 2077 Attack #1/#4 attackers over the 2692 sites in our Top 500 dataset (3298 attackers over 7000 sites in the Top Million). Many of these attackers are quite limited in the targets they can attack, with just over half of attackers in the Top 500 only able to attack a single, particular snapshot domain (40% in the Top Million). However, attackers with more widespread opportunities exist. Table 2 shows the individual third-party domains which could launch Attacks #1 or #4 against the *most* snapshot domains. Many of these domains are third-party domains which appear as across a large number of sites, such as advertising and analytics networks, social network widgets, and content distribution services. We do not expect any of these companies to maliciously modify the archive; rather, we list them to characterize the types of modern web practices which so frequently lead to our vulnerabilities.

First vs. Third Party Attackers. While Same-Origin Escape based attacks (#2 and #3) can only be executed by a third-party domain, both Archive-Escape Abuse and Anachronism Injection attacks (#1 and #4) can be performed by both first- and third-parties. Both of these types of attackers are interesting, although they represent significantly motivated attackers. The first-party is usually the original publisher of the information in the snapshot, and so a first-party attacker is changing content they published, while a third-party attacker is generally changing content which was originally created and published by the first-party. While both first- and third-parties are potentially interesting attackers, we note that individual site owners may be more alarmed by the potential for third-parties to modify their snapshots.

Over the existence of the archive, third-party attackers have become much more common for archive-escape vulnerabilities, to the point that nearly every (97%) recent snapshot with an archive-escape vulnerability includes at least one to with a third-party destination, up from 60% since 2007-timestamp snapshots. We hypothesize that this trend is caused by the combined trends in the modern web of increasing complexity and increasing inclusion of third-parties. By contrast, third-party missing resources have become less common over time. They made up nearly all missing resource vulnerabilities in 1996 (98%), and only about 40% in 2016.

Unowned Attack Domains. Our vulnerabilities enable attacks by particular domains on the Internet, but the ownership of that domain may shift over time. Indeed, attacker domains are sometimes completely unowned. Aggregating across our datasets, we found 23 archive-escape destination domains and 60 never-archived resource domains which were unowned as of Spring 2017. These domains can be purchased by anyone to launch an attack on their vulnerable sites. This is how we performed our proof-of-concept attack (Figure 1). We found no unowned attack domains in our legal dataset.

6.4 Measurements of URLs Used in Court Proceedings

We now analyze our dataset of the archive.org URLs used in court proceedings. Recall from Section 6.1 that this dataset consists of 840 URLs from 991 legal documents. Because they have been cited in court proceedings, the accuracy of these archived pages is critical — or, conversely, the motivation clearly exists for a potential attacker to manipulate one of these snapshots to influence legal proceedings.

In this section, we thus investigate the prevalence of vulnerabilities in these snapshots. We stress that the presence of a vulnerability does *not* imply that an attack actually occurred. Indeed, evaluating the question of whether an attack occurred is challenging, since, for most attacks, they can be temporarily enabled and then disabled. Instead, our goal is to survey the prevalence of these vulnerabilities in specific archives that *have* been used in legal proceedings in the past, to serve as a note of caution for the use of archived URLs in future proceedings.

For these legally referenced snapshots, we considered only Attacks #1 and #4, which do not require foresight, and thus could be mounted after the fact, at the time of legal proceedings. 57 were vulnerable to Attack #1, and 37 of those were complete-control vulnerabilities. However, *none* contained never-archive resources, which is quite unlike the archive at large, which commonly contains never-archived resource vulnerabilities (Figure 3). We hypothesize that URLs cited in legal proceedings may be of higher quality since they were curated by experts deciding which URLs to cite.

If these vulnerabilities had been exploited at the time of these legal cases, they could have given an attacker the ability to hide or plant evidence. Again, we stress that we have no reason to believe that any of these vulnerabilities were exploited at the time of the relevant court proceedings, but emphasize that future use of archived URLs in legal or other similar matters must be treated with caution.

7 DEFENSES

In this section, we explore the space of possible defenses against our attacks, including defenses which *detect* or *block* our attacks. As an overall defensive goal, we aim to allow users of archives to have more confidence in their understanding of the web of the past.

We organize our defenses first by **who** deploys them: website publishers, archives, or clients, and categorize them additionally by **when** they can be deployed (i.e., whether they work retrospectively, after time-of-attack). This breakdown is important, since while end-user defenses are the easiest to deploy for high-value expert users, but we recognize that most ordinary users will *not* install defenses, suggesting that exploring centrally deployed defenses is also important. Table 3 summarizes these defenses, and we discuss them in detail below. We also we present the implementation of ArchiveWatcher, a browser extension which detects and blocks archive-escapes and anachronisms.

7.1 Defenses Deployed by Website Publishers

We begin with defenses website publishers can deploy to protect snapshots of their won websites. These defenses work for all clients, but must be separately deployed by each website, and some are not retroactive, since publishers cannot modify previously archived data. First-party attackers, may avoid deploying these defenses to retain editorial power over their site's past.

7.1.1 Opt-Out of Archives. Websites can opt-out of being preserved in the Wayback Machine, sidestepping the possibility of archival vulnerabilities. The Wayback Machine has long respected website publishers' opt-out preferences in two ways: manual requests, and the use of `robots.txt` policy files. By opting out of preservation entirely, a site would avoid having snapshots which could be manipulated, preventing all attacks in this paper.

The downside to this defense is that the relevant sites are not archived or available for the public to browse in the archive, eliminating all the social and cultural benefits the archive brings. This defense throws the baby out with the bathwater. Some sites may also not be legally permitted opt-out, such as government sites with archival requirements. Additionally, this defense may soon become much less viable: Wayback Machine has expressed, in a recent blog post, an intent to give less weight to `robots.txt` files, saying that as of April 2017 it now ignores `robots.txt` on U.S. government and military websites and is "looking to do this more broadly." [6]

7.1.2 Avoid Dynamically Generated URLs to Avoid Archive-Escapes. Website publishers can reduce the incidence of archive-escapes by designing their websites to use fewer dynamically generated URLs, since these are a common cause of archive-escapes.

This approach has two major weaknesses. The first is that dynamic behavior and URLs are a common, valuable feature of the modern web, and asking engineers to do without them could be inconvenient and expensive. Second, this defense cannot protect against archived-escapes caused by third-party content, such as third-party Javascript libraries, which are commonly used and whose behavior is not fully under the control of the publisher.

7.1.3 Actively Archive Subresources. In Anachronism Injection, the attacker wants to replace a subresource which has never been archived with a malicious payload. One way to defend against this attack is to preemptively replace missing subresources with benign resources, plugging the vulnerability. Though anyone can use the "Save Page Now" feature to plug vulnerabilities — the same feature attackers use to archive their payloads — website publishers wishing to defend their pages in the archive likely have the greatest incentive to do so. However, if no benign resource is published at the URL, the defense will not work. The non-malicious content could be the correct content which was originally present at the URL, an empty response, or even a 404 Not Found response. In all these cases the archive will record the given response as the only capture of the resource and serve it, causing no harm, as the nearest-neighbor to the vulnerable reference.

The most significant limitation of this defense is that only the potential attacker can publish a benign resource to be archived — the permission to enact this defense lies with the potential attacker. While anyone can ask to "Save Page Now" for any URL, this process only works for resources where the server responds to the crawler's response with *some* response, even if it is simply a 404 error. Thus attackers who wish to ensure against malice by themselves in the future, or by later owners of their domain, can use this defense, but it will be ineffective when the first-party wants to launch an attack.

Defense	Goals Prevent	 Detect	Who Deploys?	When?
Opt-Out of Archives	✓		Website Owner	Any Time
Avoid Dynamically Generated URLs	✓		Website Owner	Time-of-Publication
Actively Archive Subresources	✓		Website Owner	Time-of-Archive
Modify Archived Javascript to Avoid Escapes	✓	✓	Archive	Any time
Serve Distinct Archived Domains from Distinct Subdomains	✓		Archive	Any time
Escape-/Anachronism- Blocking Browser Extension	✓		End-user	Time-of-Access
Escape-/Anachronism- Highlighting Browser Extension		✓	End-user	Time-of-Access

Table 3: A summary of the defenses we explore.

7.2 Defenses Deployed by Web Archives

Defenses deployed by archives have the potential to be quite powerful, since archives can change the data they store in their database (as they do with URL rewriting) and the data they collect in the future, to provide both forward-looking and retroactive defense which protect the views of all clients.

7.2.1 Use Content Security Policy Headers to Block Escapes. After we disclosed the results of this paper to Internet Archive, they modified the Wayback Machine to include Content Security Policy (CSP) headers in their responses when serving archived content. These headers can be used to instruct client browsers to block the use of third-party resources in the context of a snapshot, thus preventing both archive-escape requests and their abuse. We confirmed that the Wayback Machine's changes blocked archive-escape requests such as the one which allowed our attack in Figure 1.

7.2.2 Modify/Analyze Javascript to Prevent Escapes. In this defense, the archive would statically and dynamically analyze Javascript code it captures in order to identify scripts might cause archive-escapes. The archive would then rewrite or wrap these scripts, replacing the original script with a version that performs the same operations but avoids generating archive-escapes. For example, such a defense might hook calls to browser APIs which generate HTTP requests, interposing on them to rewrite URL arguments to ensure they do not point outside the archive.

This solution is complex, and its implementation might involve many engineering hours. Additionally, executing the defense on each archived resource at time-of-archive might be computationally expensive. However, if successful, this defense might permit the Wayback Machine's URL rewriting to be much more pervasive, applying even to client-side dynamically generated URLs, the main source of vulnerabilities that we identify in the archive today.

7.2.3 Serve Distinct Archived Domains from Distinct Subdomains. Archives could defend against Same-Origin Escapes by serving content from distinct subdomains, each of which corresponds to the live domain from which that content was originally published. For example, an archive might choose to serve captures of example.com/script.js from the subdomain http://example.com.web.archive.org/ instead of from http://web.archive.org. Since the Same-Origin Policy considers subdomains as distinct domains, this would cause client browsers to provide the same isolation in the archival context as they do in the live context, preserving the same trust model across both live and archival executions of the page. We recommend that archives consider implementing this defense.

7.3 Defenses Deployed by Clients

Finally, we discuss defenses deployed inside the client's browser. Individual clients can unilaterally deploy these defenses, giving them high value today. For example, experts in legal cases might use these defenses to provide more trustworthy testimony. These defenses are limited by the fact that each client must separately install the defense, but they do apply to all snapshots in the archive.

7.3.1 Browser Extensions to Block/Highlight Escapes and Anachronisms. This defense interposes on and blocks Archive-Escape and Anachronistic requests made for subresources while browsing the archive. It prevents Archive-Escape Abuse by blocking all HTTP requests from a snapshot which leave the archive. Since the distinction between archive-escapes and archival requests is cut and dry (distinguishable by the destination domain of the request), such a defense should be highly effective against Archive-Escape Abuse.

This defense protects against Anachronism Injection not by preventing the payload from being stored in the archive (as does the *Actively Archive Subresources* defense, above), but by blocking the anachronistic request which delivers that payload to the client. It does so by blocking anachronistic requests — those requests for archival resources which have timestamps far from the timestamp of the enclosing page. This involves an inherent tradeoff, in which the defense or its user must define *how* anachronistic a resource must be to be blocked. In the most extreme case, only resources with timestamp exactly equal to the snapshot's timestamp can be loaded, leading to complete blocking of the vulnerability, but also preventing many legitimate resources from being loaded, leading to a less complete picture of the past web.

This defense can also (or instead) visibly highlight, log, or summarize archive-escapes and anachronistic requests and the visible page elements which correspond to them. Such a feature can help a human expert to better judge the accuracy of a snapshot. ArchiveWatcher, described in more detail below (Section 7.4), is an example of this type of defense.

7.4 ArchiveWatcher: An End-User Defense

We prototyped ArchiveWatcher, a client-deployed defense consisting of a browser extension which detects and blocks archive-escape request vulnerabilities. ArchiveWatcher is implemented as a lightweight Chrome Extension which interposes on requests made for resources while browsing snapshots https://web.archive.org/web. It is written in 6000 lines of Javascript, CSS, and HTML, and its souce code can be found on Github by following the links at https://rewritinghistory.cs.washington.edu.

As described above in Section 7.3.1, ArchiveWatcher blocks requests for archive-escapes. It can display to the user a summary of the requests it has detected and blocked on the current snapshot and across the current browsing session. ArchiveWatcher suggests directions for defenses which could aid technical experts assessing the veracity of archival snapshots.

8 CONCLUSION

In this paper, we have explored the space of attacks which can rewrite history — i.e., attacks that can manipulate how clients see archived websites, focusing on the Wayback Machine. Though it is known that the archive contains accidental inaccuracies, to our knowledge, we are the first to explore how an attacker might introduce *intentional* errors. We identified and explored several vulnerabilities in how the Wayback Machine archives and serves snapshots of websites, and we developed four attacks that leverage these vulnerabilities. We demonstrated proof-of-concept attacks on the Wayback Machine, showing that we were able to manipulate client views of snapshots without compromising the archive's or any other servers. We then quantified the prevalence of these types of vulnerabilities, finding that over 70% of the sites we investigated are vulnerable to this type of manipulation by *some* attacker.

The web is important to our modern society, making web archives a critical source of socially important information, from journalism to legal proceedings. This work suggests the importance for website publishers, archive designers, and end users to take steps to prevent or detect intentional manipulation.

ACKNOWLEDGEMENTS

We thank Lucy Simko, Anna Kornfeld Simpson, and Eric Zeng for their insightful comments and feedback on the paper; Emily McReynolds for feedback, advice, and consultation on legal concepts referenced in the paper; and Gaites Swanson for his help discovering, parsing, and interpreting the legal URLs we studied.

We thank Mark Graham and his colleagues at Internet Archive for their thoughtful and rapid response to our disclosure of this work.

This work was supported in part by NSF Grant IIS-1302709, the Short-Dooley Professorship, and the UW Tech Policy Lab.

REFERENCES

[1] 2012. Laboratory Corp. of America v. United States, 108 Fed.Cl. 549 (2012). (2012).

[2] 2012. People v. Franzen, 210 Cal.App.4th 1193 (2012). (2012).

[3] 2013. Ex Parte Serguei N. Mamedrzaev. 2013 WL 1558372. (2013).

[4] 2014. Tharpe v. Lawidjaja, 8 F.Supp.3d 743 (2014). (2014).

[5] 2016. The Euroeapn Patent Convention, Article 54: Novelty. https://www.epo.org/law-practice/legal-texts/html/epc/2016/e/ar54.html. (2016). Accessed: 2017-05-17.

[6] 2017. Robots.txt meant for search engines don't work well for web archives. https://blog.archive.org/2017/04/17/robots-txt-meant-for-search-engines-dont-work-well-for-web-archives/. (4 2017). Accessed: 2017-05-19.

[7] 2017. Summary of s3.amazonaws.com. https://web.archive.org/web/*/http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. (2017). Accessed: 2017-05-05.

[8] 2017. Welcome to LexisNexis - Choose Your Path. https://www.lexisnexis.com/en-us/gateway.page. (2017). Accessed: 2017-05-19.

[9] 2017. WestLaw.com. westlaw.com. (2017). Accessed: 2017-05-19.

[10] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Arcahaeological Study of Web Tracking from 1996 to 2016. *25th USENIX Security Symposium* (August 2016).

[11] Scott G. Ainsworth, Ahmed AlSum, Hany SalahEldeen, Michele C. Weigle, and Michael L. Nelson. 2012. How Much of the Web Is Archived? *arxiv.org* (2012), 1–10. arXiv:1212.6177 http://arxiv.org/abs/1212.6177

[12] Scott G Ainsworth and Michael L Nelson. 2004. Only One Out of Five Archived Web Pages Existed as Presented. *ACM HT'15* (2004). http://public.lanl.gov/herbertv/papers/Papers/2015/ht15-ainsworth-submission.pdf

[13] 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*. ACM, 257–266.

[14] Internet Archive. 2017. Heritrix is the Internet Archive's open-source, extensible, web-scale, archival-quality web crawler project. https://github.com/internetarchive/heritrix3. (2017). Accessed: 2017-08-16.

[15] Internet Archive. 2017. IA's public Wayback Machine (moved from SourceForge). https://github.com/internetarchive/wayback. (2017). Accessed: 2017-08-16.

[16] Justin F. Brunelle. 2012. 2012-10-10: Zombies in the Archives. http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html. (2012). Accessed: 2017-05-13.

[17] Justin F Brunelle, Mat Kelly, Hany Salaheldeen, Michele C Weigle, and Michael L Nelson. 2015. Not All Mementos Are Created Equal : Measuring The Impact Of Missing Resources Categories and Subject Descriptors. *International Journal on Digital Libraries* (2015).

[18] International Internet Preservation Consortium. 2017. The OpenWayback Development http://www.netpreserve.org/openwayback. https://github.com/iipc/openwayback. (2017). Accessed: 2017-08-16.

[19] Shawn E. Douglas. [n. d.]. Citing from a Digital Archive like the Internet Archive: A Cheat Sheet. http://www.writediteach.com/images/Citing%20from%20a%20Digital%20Archive%20like%20the%20Internet%20Archive.pdf. ([n. d.]). Accessed: 2017-05-08.

[20] Peter Eckersley. 2010. How unique is your web browser? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6205 LNCS (2010), 1–18. https://doi.org/10.1007/978-3-642-14527-8_1

[21] Deborah R Eltgrowth. 2009. Best evidence and the Wayback Machine: toward a workable authentication standard for archived Internet evidence. *Fordham L. Rev.* 78 (2009), 181.

[22] Matthew Fagan. 2007. Can You Do a Wayback on That-The Legal Community's Use of Cached Web Pages in and out of Trial. *BUJ Sci. & Tech. L.* 13 (2007), 46.

[23] David Fifield and Serge Egelman. 2015. Fingerprinting web users through font metrics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8975 (2015), 107–124. https://doi.org/10.1007/978-3-662-47854-7_7

[24] Karén Gazaryan. 2013. Authenticity of Archived Websites: The Need to Lower the Evidentiary Hurdle Is Imminent. *Rutgers Computer & Tech. LJ* 39 (2013), 216.

[25] Stephanie Hackett, Bambang Parmanto, and Xiaoming Zeng. 2003. Accessibility of Internet websites through time. *ACM SIGACCESS Accessibility and Computing* (2003), 32. https://doi.org/10.1145/1029014.1028638

[26] Internet Archive. 2017. Internet Archive: Digital Library of Free Books, Movies, Music & Wayback Machine. https://archive.org/. (2017). Accessed: 2017-05-12.

[27] Internet Archive. 2017. Internet Archive Frequently Asked Questions. https://archive.org/about/faqs.php#23. (2017). Accessed: 2017-05-04.

[28] Internet Archive. 2017. Wayback Machine. https://web.archive.org. (2017). Accessed: 2017-05-11.

[29] Internet Memory Foundation. 2017. Internet Memory Foundation. http://internetmemory.org/en/. (2017). Accessed: 2017-08-16.

[30] Mat Kelly, Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. 2013. On the change in archivability of websites over time. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8092 LNCS (2013), 35–47. https://doi.org/10.1007/978-3-642-40501-3_5 arXiv:1307.8067

[31] Mat Kelly, Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. 2013. On the Change in Archivability of Websites Over Time. *CoRR* abs/1307.8067 (2013). http://arxiv.org/abs/1307.8067

[32] Library of Congress. 2017. Archived Web Site | Library of Congress. https://www.loc.gov/websites/. (2017). Accessed: 2017-05-12.

[33] Keaton Mowery and Hovav Shacham. 2012. Pixel Perfect : Fingerprinting Canvas in HTML5. *Web 2.0 Security & Privacy 20 (W2SP)* (2012), 1–12. https://cseweb.ucsd.edu/

[34] Jamie Murphy, Noor Hazarina Hashim, and Peter O'Connor. 2007. Take Me Back: Validating the Wayback Machine. *Journal of Computer-Mediated Communication* 13, 1 (2007), 60–75. https://doi.org/10.1111/j.1083-6101.2007.00386.x

[35] 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 2012 ACM conference on Computer and communications security*. ACM, 736–747.

[36] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. *Proceedings - IEEE Symposium on Security and Privacy* (2013), 541–555. https://doi.org/10.1109/SP.2013.43

[37] US Department of Homeland Security. 2016. Homeland Security. http://webarchive.loc.gov/all/20160205185026/https://www.dhs.gov/. (2016). Accessed: 2017-08-16.

[38] Mary Emily Ohara. 2017. Trump Administration Removes LGBTQ Content From Federal Websites. https://web.archive.org/web/20170324052626/http://www.nbcnews.com/feature/nbc-out/trump-administration-removes-lgbtq-content-federal-websites-n711416. (2017). Accessed: 2017-03-27.

[39] OpenGovData Russia Archive. 2017. Arhivacija gosudarstva (konservirovannoe gosudarstvo) | Otkrytye dannye v Rossii. http://opengovdata.ru/projects/govarchive/. (2017). Accessed: 2017-08-16.

[40] James L Quarles III and Richard A Crudo. 2014. Using the Wayback Machine in Patent Litigation. *Landslide Magazine* 6, 3 (Jan/Feb 2014).

[41] Achintya Rao. 2017. Using the Internet Archive to cite websites. https://medium.com/@RaoOfPhysics/using-the-internet-archive-to-cite-websites-89bd3f2ce0fd. (2017). Accessed: 2017-05-08.

[42] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and defending against third-party tracking on the web. *Proc. of the USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2012), 12. http://www.franziroesner.com/pdf/webtracking-NSDI2012.pdf

[43] Ryan North. 2016. Dinosaur Comics - February 3rd, 2016 - awesome fun times! http://webarchive.loc.gov/all/20160203203159/http://www.qwantz.com/index.php. (2016). Accessed: 2017-08-16.

[44] Myriam Ben Saad and Stéphane Gançarski. 2011. Improving the quality of web archives through the importance of changes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6860 LNCS, PART 1 (2011), 394–409. https://doi.org/10.1007/978-3-642-23088-2_29

[45] Kyle Soska and Nicolas Christin. 2014. Automatically Detecting Vulnerable Websites Before They Turn Malicious. *23rd USENIX Security Symposium (USENIX Security 14)* (2014), 625–640. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/soska

[46] Stanford Libraries. 2017. Web Archiving | Stanford Libraries. http://library.stanford.edu/projects/web-archiving. (2017). Accessed: 2017-08-16.

[47] Wikipedia. 2017. List of Web archiving initiatives. https://en.wikipedia.org/wiki/List_of_Web_archiving_initiatives. (2017). Accessed: 2017-08-16.