Alternative (ab)uses for HTTP Alternative Services

Trishita Tiwari
trtiwari@bu.edu
Boston University

Ari Trachtenberg
trachten@bu.edu
Boston University

Abstract

The HTTP Alternative Services header (`Alt-Svc`) was introduced in 2013 in a bid to streamline load balancing, protocol optimizations, and client segmentation, and it has since been subsequently implemented in almost all mobile and desktop browsers. We show that the major implementations of the header are independently susceptible to a variety of stealthy abuse. Indeed, we demonstrate how Alternative Services may be leveraged to scan ports blacklisted by browsers, probe firewalled hosts, and mount Distributed Denial of Service attacks. These services may also be misused to bypass popular phishing and malware protection services like Safe Browsing, and also online site checkers like VirusTotal, URLVoid, Sucuri and IPVoid. In the privacy realm, the `Alt-Svc` header may be abused for user tracking: at the network layer by Internet Service Providers (ISPs), and at the application layer by first and third party websites (where we bypass third-party tracking protections on Firefox, Chrome and Brave). In a similar manner, the header may be used by transiently connected ISPs to exfiltrate parts of a victim's browser history. Our attacks work, to varying extents, on Firefox, Tor, Chrome, and Brave browser, and have been disclosed accordingly–so far, one of our vulnerabilities been patched by Mozilla as CVE-2019-11728. We conclude with proposed mitigations for some of these abuses.

1 Introduction

HyperText Transfer Protocol (`HTTP`) headers date back to the very earliest web servers, providing an out-of-band mechanism for clients and servers to exchange metadata. Early headers included basic information about the `HTTP` request context, like the domain name of the server or the name of the client's browser. However, as web content grew in complexity, so did the the the scope of `HTTP` headers, expanding to include expiration dates, user-side cached data (cookies), and control options for connection reuse. Accompanying this growth in header complexity was a proliferation of attacks to the privacy and security of user data, and corresponding mitigations [43].

The web community now has almost thirty years of experience with the `HTTP` protocol, and mature browser development teams span a multitude of companies, some with market capitalizations approaching one billion dollars each [10]. And yet, despite all this knowledge and experience, we aim to show that it is still extremely difficult to introduce even simple improvements to the HTTP protocol without introducing a variety of security vulnerabilities.

Specifically, in this work we evaluate `HTTP` Alternative Services header (`Alt-Svc`), a simple optimization introduced in 2013 with three potential use-cases in mind: streamlining server load-balancing without the need for fine-grained Domain-Name-Service (`DNS`) maintenance, readily making clients aware of endpoints supporting newer protocols (*e.g.*, `HTTP/2` or `QUIC`), when available, and segmenting clients into groups with prescribed capabilities [26]. We show that, though well-intentioned and apparently straightforward, this new header may be leveraged for a variety of less innocuous purposes, including third-party network reconnaissance and service denial, bypassing phishing and malware protections, and user tracking. Indeed, the fundamental reason behind all these attacks is that the `Alt-Svc` header introduces unchecked background connection attempts from the victim's browser to attacker specified endpoints. Worse yet, these endpoints may also be cached to enable automatic future connections.

1.1 Overview of abuse

Network reconnaissance is one of the first steps in many electronic attacks, and it is often initiated with a scan of accessible ports on the target. Defenders typically impede port scans in a number of ways, including (i) using an anomaly detection system to recognize scans coming from a common startpoint, and (ii) placing hosts behind firewalls or Network Address Translators (NATs). Unfortunately, the current implementations of `Alt-Svc` allow attackers to instigate stealthy distributed port scans from a web server under their control. The scans are initiated by victim machines that connect to the controlled server, and thus are both distributed across many startpoints

and able to target any machine visible from the victim, even if the victim is behind a firewall. Unlike known JavaScript versions of this attack, the victim sees no indication of the attack within her browser, and the attack can target ports that are otherwise blocked by the browser due to security concerns.

Indeed, this scanning ability can also be translated into a Distributed Denial of Service (DDoS) attack, wherein an `Alt-Svc` connection from a browser opens long-lasting connections on a target server, or a short client message leads to a much larger server response against an intended target. Unlike existing DDoS approaches [6, 28, 29, 36, 53], our methods allow an attacker to target *any* arbitrary port of the target host by means of client web browsers. The distributed nature of such attacks, coming from a variety of seemingly unrelated and otherwise typical clients, make them very hard to attribute to a root cause or mitigate; indeed, distributed attack mitigation has become the expertise of niche providers such as Cloudflare [1]. The same scanning ability also carries the potential of confusing Intrusion Detection Systems by triggering anomalies that are tied to unusual patterns of IP or port accesses.

Another potential abuse of `Alt-Svc` that we highlight in this work relates to user tracking. Privacy concerns have also lead to a proliferation of client-side blocking tools such as AdBlock, Ghoster, Disconnect, Privacy-Badger and other tracker/advertisement blockers [18]. Indeed, most modern browsers either block third-party trackers by default, or offer the option to do so [37]. Furthermore, a number of security-focused browsers have been introduced from both industry and academia, including Tor, Brave Browser, FuzzyFox [38] and DeterFox [17]. On the network layer, privacy protections include web traffic encryption (`HTTPS`), `MAC` address randomization [3, 12], and elimination of other trackable network identifiers (for instance, `TLS` resumption cache isolation in Tor [4]). Unfortunately, `Alt-Svc` can be abused to circumvent a variety of these protections, allowing, for example, tracking by first- and third-party websites, network-layer tracking, and bypass of some standard third-party blocking mechanisms in common use [37].

Similar Alternative Service attacks can be used to perform network-layer browser history exfiltration. Unlike most prior works based on third-party websites [14, 46, 56], our approach for this attack targets a network-level adversary (e.g., an Internet Service Provider or an owner of a Wifi router) that exfiltrates a user's browser history, including web accesses performed outside the observation window of the attacker.

Finally, Alternative Services also provide a mechanism for bypassing browser defenses against phishing. The prevalence and damage of phishing sites, which impersonate well-known company sites for the purpose of gathering sensitive personal user data [9, 45, 47], has led browsers to implement a variety of blocking filters [15, 41], which, for example, display conspicuous warnings if the user navigates to a suspicious website, or even a clean website that draws some content from a suspicious site. Indeed, various online tools like VirusTotal, URLVoid, Sucuri and IPVoid also help to identify unsafe websites [35, 48, 50, 54]. The general approach of these blocking mechanisms is to check website domains against an independently generated blacklist of suspicious content. Unfortunately, this domain checking can be rather simply evaded with the use of the `Alt-Svc` header.

Main Contributions The main browser-based attacks presented in this work are thus:

1. Distributed `TCP` and `UDP` port scanning;
2. Bypass of malware and phishing protections;
3. Distributed Denial of Service of generic services; and,
4. Tracking and history exfiltration.

We note that different browsers are affected to varying extents by these different attacks. Further, most of these attacks have direct impact to modern browsers, but others serve as cautionary threats toward future adoption of full Alternative Services functionality.

Our attacks have been disclosed to Firefox, Tor, Chrome, and Brave browser development teams.

Roadmap The remainder of this paper is organized as follows: in Section 2 we discuss the background of the `Alt-Svc` header. We then discuss prior works that relate to our contributions in Section 3. Section 4 describes our various `Alt-Svc` header attacks, together with potential mitigations for some of them. Finally we conclude in Section 5.

2 Background

We next provide some historic and current context for the Alternative Services header.

2.1 Why Alt-Svc?

Under the older `HTTP/1.1` regimen, servers would typically load balance their clients by means of short-lifetime `DNS` query responses; as one server would get loaded, the `DNS` responses would shift toward a different, less busy, server. This worked well because `HTTP/1.1` involved many short-lived `TCP` connections, each with individual `HTTP/1.1`-requests. However, `HTTP/2` introduced various optimizations, most notably the use of longer `TCP` connections to multiplex multiple `HTTP/2` requests. In these cases, clients could bind to `DNS` queries for longer periods of time, thereby complicating load-balancing efforts among servers.

The Alternative Services `HTTP`-header (`Alt-Svc`) was designed, in part, to help alleviate this problem by allowing a server to specify alternative endpoints for client loads [26].

For example, a client loading a page from `www.example.com` could receive a server response specifying an `Alt-Svc` header pointing to another endpoint, say `other.example.com`. For subsequent requests, the browser may choose from which domain to load the page (*i.e.*, the primary or one of the alternates). It could also cache the header values for a period of time (specified via the `ma` parameter). By default, these cached values are supposed to be invalidated when the client machine changes network, unless the `persist` parameter is set. A typical header thus looks as follows:

```
Alt-Svc:    http/1.1="other.example.com:443";
            ma=2928333;
            persist=1
```

The header here suggests that an alternate endpoint for the `HTTP/1.1` protocol (specified via the `http/1.1` keyword) is available at `other.example.com`. This endpoint is valid for 2928333 seconds and should be persisted across network changes. Indeed, the `ma` parameter could even be as short as a few seconds. This makes `Alt-Svc` extremely flexible–websites can conveniently advertise new endpoints and remove existing ones very quickly – more so than other load balancing approaches such as `DNS` Round Robin, where such on-the-fly changes may be more limited.

Websites may also choose to advertise endpoints for protocols other than `HTTP/1.1`, such as `HTTP/2` over `TLS` (`h2`), `HTTP/2`-over plain text (`h2c`), `QUIC` (`quic`), `SPDY` (`spdy`), and the like. This enables gradual adoption of new protocols, and is in fact the main reason why browsers today support the `Alt-Svc` header. Note that the `Alt-Svc` header is not used to upgrade a browser's connection to a newer protocol–that happens lower on the network stack, during protocol negotiation. The `Alt-Svc` header is used merely to make the client browser aware of new endpoints that support other protocols.

2.2 Browser implementation

The typical function of browsers that support Alternative Services (such as Firefox, Tor, Brave and Chrome), when encountering a page or an `iframe` that presents an `Alt-Svc` header, follows the scheme in Figure 1: the browser attempts to connect to the endpoint specified in the header, in the background, while the content of the page continues to load in the foreground. When the background connection to the alternate endpoint succeeds, the client browser validates the alternate site's certificate against the *original* domain endpoint. If validation succeeds, the client proceeds to load some of the site content from the alternate endpoint. Note that the reason for this cross-validation of the alternate certificate against the original domain is to establish the original web server's control over the alternate endpoint (i.e. maintaining the "same-origin" policy [26]).

However, the mere fact that the browser initiates a connection to *any* specified endpoint (whether or not the endpoint is eventually deemed as valid) is ripe for exploitation, as we demonstrate in Sections 4.1 and 4.3. When the endpoint specified has been validated, most browsers cache the endpoint for future visits, and this too can be exploited for tracking and history exfiltration (Sections 4.4 and 4.5).

Note that the user has no control over or browser-based visibility into this alternative services behavior. Worse yet, `Alt-Svc` header adherence is enabled by default on all four browsers (Firefox, Chrome, Tor, and Brave), making these attacks both more insidious and stealthy.

2.3 Uses in the wild

Alternative Services have grown in their use-cases over the years. On the server-side, Google services (the search engine, Gmail, etc) advertise an alternate endpoint for serving content over their own `UDP`-based `QUIC` protocol. Other websites, like Facebook, detect Tor client browsers and use the `Alt-Svc` header to advertise `onion` hidden service endpoints. Websites may also use the header to enable opportunistic encryption, "opportunistically" loading some content over an encrypted channel without committing to site-wide `HTTPS` [27].

On the client side, Chrome and Brave Browser have supported `QUIC`-based alternative endpoints since 2015 [7] (where support for the `QUIC` protocol is, for now, denoted as an experimental feature and is gated by a preference flag). Firefox and Tor browsers, in turn, have supported `HTTP/2`-based `Alt-Svc` connections since 2015 [8]. In fact, most of these browsers aggressively adhere to the `Alt-Svc` header for the protocol each supports, loading all content from the alternative endpoint for *all* subsequent requests when the original endpoint did not already serve content over the more advanced `HTTP/2` or `QUIC` protocol. The lone exception is the Tor browser, which tends to be more conservative with `Alt-Svc` endpoints and only uses them some, not all, of the times.

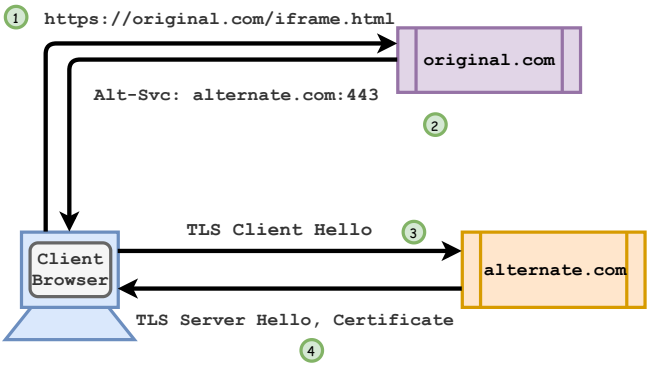

Figure 1: Typical functioning of `Alt-Svc`.

3 Related Work

There have been numerous works that demonstrate how a malicious website can force a browser to scan target TCP ports from the context of users that visit the website [30, 33, 34, 57]. Such third-party scanning enables the attacker to remain hidden from the target, since the scanning is done by the victim's browser. It also allows attackers to scan hosts that are only accessible to the victim (*e.g.*, machines on a private network). However, the potential for cross protocol scripting attacks [49] has led major modern browsers prohibit access to certain ports [22, 44]. This port blacklist includes most interesting or privileged ports, and this limits the effectiveness of existing browser-based port scanning solutions. A different approach by Heralla [33] claims to leverage features such as WebRTC to perform the port scan; however, the author insists that this attack only works against a dedicated server, and not against any arbitrary target of the attacker's choice. Instead, we present a method for scanning *any* TCP port on *any* host visible to the victim (including hosts on a private subnet) by using Firefox and Tor's implementation of the Alt-Svc header.

Our approach also works for UDP-port scanning through the Chrome and Brave browsers, albeit only for ports that are not blacklisted because Chrome and Brave's Alt-Svc implementation strictly adheres to the port blacklist [44]. To the best of our knowledge, this is a new attack capability, since the existing approaches generally leverage TCP-based HTTP-requests for their activities. Herrala [34] does claim a technique for UDP scanning, but, by his own report, it suffers from the same limitations as in the TCP case, namely that it cannot be used against an arbitrary target.

Today, there are many services that protect Internet users from malicious websites. Some of these are browser based, such as built-in features in Chrome, Firefox, and Brave [15, 41]. Others are online tools that evaluate any given website or URL [35, 48, 50, 54]. These scanners operate by blacklisting domain names known to be associated with malicious content. Chrome and Brave will typically display conspicuous warnings if a user directly visits a suspicious website, or even a white-listed website loads some content from a blacklisted domain (see Figure 4). Firefox will also display a similar warning page when a user directly visits a malicious website, and it refuses to load and render third-party content from blacklisted domains, throwing a security error visible in the JavaScript Console. The online scanners like VirusTotal, IPVoid, URLVoid, Sucuri, etc. [35, 48, 50, 54], on the other hand, maintain independent evaluations of most websites, and will appraise user-specified sites against their database for safety. In this work, we show how we can use Alt-Svc to bypass these blacklists.

There has also been prior work on browser based BotNets used for DDoS attacks [29, 36, 53]. However, since attackers are operating typically through HTML and JavaScript, they are limited to attacking services on TCP ports that are not on the browser blacklist, and thus typically only attack other web servers. However, our techniques on Firefox and Tor allow attackers to flood *any* port with packets, thus allowing attacks on other services (*e.g.*, e-mail servers) from within the confines of a browser.

There has also been much work done on browser-based user tracking [11, 16]. Prior literature has shown how various browser caches can be used for tracking users, for example: the HTTP 301 redirect cache [55], the HTTP authentication cache [31], the HSTS cache [32] and DNS caches [24]. There has also been work that demonstrates the ability of several identifiers such as Etag and Last-Modified header values to be used for tracking [5, 13]. More recent independent (and concurrent published) work briefly discussed some of the concerns with tracking using Alt-Svc [51]. Device fingerprinting presents another class of techniques to *statelessly* track a user. Various attributes such as the User-Agent string, HTML5 canvas fingerprinting, screen-size, resolution, list of installed plugins, and the like have been shown to reveal identifying information about users [23]. Other identifying information can be obtained through JavaScript performance and conformance tests [40, 42], mobile sensors [21] and font enumeration [25]. In this work, we present yet another technique for user tracking using the Alt-Svc header. Our techniques bypass third-party tracker blocking options instituted in most browsers today [37].

Further, in addition to browser based tracking, there have also been techniques for network-based tracking and fingerprinting [39, 52, 58]. The technique widely used for this is MAC address tracking, which is why most companies today randomize their MAC addresses [3, 12]. To overcome these challenges in tracking, we present a new technique for network-local adversaries to track a particular user using Alt-Svc. This can be done without the need to install any third-party cookies or known trackers, thus presenting a convenient way to track using network traffic data.

Finally, we also show how the Alt-Svc header can be used by a network level attacker to exfiltrate a user's browser history. Our attack requires a threat model similar to the history exfiltration technique presented by Dabrowski et. al. [20]. Their work demonstrates how a Captive Wi-Fi Portal (*i.e.*, an internet provider that also has a network login webpage) can be used to exfiltrate a victim's browser history by leveraging the HSTS cache and cached HTTP cookies. However, the techniques they present have certain limitations–for instance, their exfiltration techniques do not work against websites that are on the browser's pre-loaded HSTS list, or against websites that use secure cookies. Our technique does not face these limitations, and thus may be used to complement their attacks. Our work also differs from other prior works on history inference [14, 56], which require a browser-based adversary rather than a network-based adversary.

Finally, unlike most JavaScript based website attacks, our techniques operate at a layer below on the browser level,

and thus are not visible to both the victim and any JavaScript-based detection techniques. This, coupled with the fact that all our attacks require no user interaction and could be mounted via third-party web content, makes our techniques especially pernicious. It is thus essential that these vulnerabilities be fixed at their source–in the browser. For this, we are currently working with the affected browser vendors to implement the appropriate patches, where possible.

4 Attacks

We next discuss the different ways in which `Alt-Svc` can be abused. For each attack, we describe, in technical detail, the impact, the browsers affected, and our experimental results. We note that the impact of the attacks in Sections 4.1 through 4.4 depend on *browser* adoption of `Alt-Svc`, which is widespread. The impact of the attacks in Sections 4.5 and 4.6, on the other hand, depend on *server* adoption, which is currently limited to large web companies.

Threat Model Our attacks can be classified into two different classes, based on the threat model involved. The first class involves an adversary that runs one or more websites (i.e., through a single domain, or multiple colluding domains). The second model involves an attacker that runs one website, but can also monitor the victim's network traffic. The second attacker could be the victim's internet provider or a public network provider akin to those found in coffee shops, both of which typically can monitor the victim's (encrypted) traffic and also control certain web pages that the victim would need to use (*e.g.*, network login pages).

In both attacker models, we assume that the attacker has no access to the victim's machine outside of the browser sandbox–*i.e.*, no hardware or operating system access. Furthermore, even within the browser, we assume that the attacker can only control data within her own website origin. Finally, it is important to stress that none of the attacks require any interaction from the user (assuming, for Chrome and Brave users, that they have the `QUIC` protocol enabled, which, for now, is an experimental feature behind a preference flag).

4.1 Distributed port scanning

The `Alt-Svc` header may be leveraged to force an unwitting Firefox or Tor user to scan any `TCP` ports of any host that the victim can access. This vulnerability has been publicly disclosed by Mozilla as CVE-2019-11728 [2]. This includes ports that Firefox and Tor mark as unsafe and are otherwise inaccessible through JavaScript. For Firefox, the attacker can access hosts behind the victim's firewall or even the victim's machine itself (*i.e.*, `localhost`), and for Tor, the attacker can scan public targets that the exit node can access. The attack naturally lends itself to a stealthy distributed port scanning of a target from multiple victim browsers connecting to the attacker's website. The header can also be used to scan UDP ports on Chrome and Brave, although some of these ports *are* blacklisted [44].

The basis of this attack is the observation that if a website specifies an `Alt-Svc` header to a secondary host with an `HTTP/2`/`QUIC` endpoint, then browsers immediately try to initiate a handshake with the secondary host, without performing any checks on the host or port; the secondary host could even be a private `IP` or `localhost`, and the port could be on the browser's `HTTP` port blacklist.

The port-scanning thus proceeds with the attacker creating a webpage (*e.g.*, `https://evil.com/p1`) whose `Alt-Svc` is set to the target host and port (see Figure 2). When the victim then visits this page, the victim's browser parses this alternative endpoint and attempts to initiate a connection to the target host on the specified port:

1. If the target port is in a `closed` state, the target immediately returns a `RST` packet to the browser and the browser registers the alternative service as broken.

2. However, if the target port is in an `open` state, more packets are typically exchanged (*e.g.*, `ACK`s) and the browser thus does not immediately know whether the `Alt-Svc` host is valid.

To discern between these two possible states of the target port, the attacker then automatically redirects the victim to a second webpage on the same attacker host (*e.g.*, `https://evil.com/p2`), but whose `Alt-Svc` host is under the attacker's control (*e.g.*, `moreevil.com`). Two cases emerge:

1. The browser has determined that the original `Alt-Svc` (*i.e.*, the target host and port) is broken, and the redirect will produce an immediate connection to the secondary attacker site (`moreevil.com`) that can be logged by the attacker.

2. The browser has not yet determined that the original `Alt-Svc` is broken, and the redirect will not produce a connection to the secondary attacker site (`moreevil.com`).

In this way the attacker can infer whether the target port was `open` or `closed`.

We have implemented this on Firefox and Tor using an `h2` (`HTTP/2`) `Alt-Svc` endpoint advertisement, sent upon loading a (hidden) `iframe` embedded within a page. We provide a screenshot of our demo website on Firefox as it scans port 25 (a port whose access is otherwise restricted through the browser) on the victim's `localhost` in Figure 3.

Within Firefox, we are able to scan public and even private hosts, including `localhost` and private `IP` addresses. Within Tor, we are not able to scan `localhost` and private `IP` addresses, presumably because the Tor exit nodes cannot access

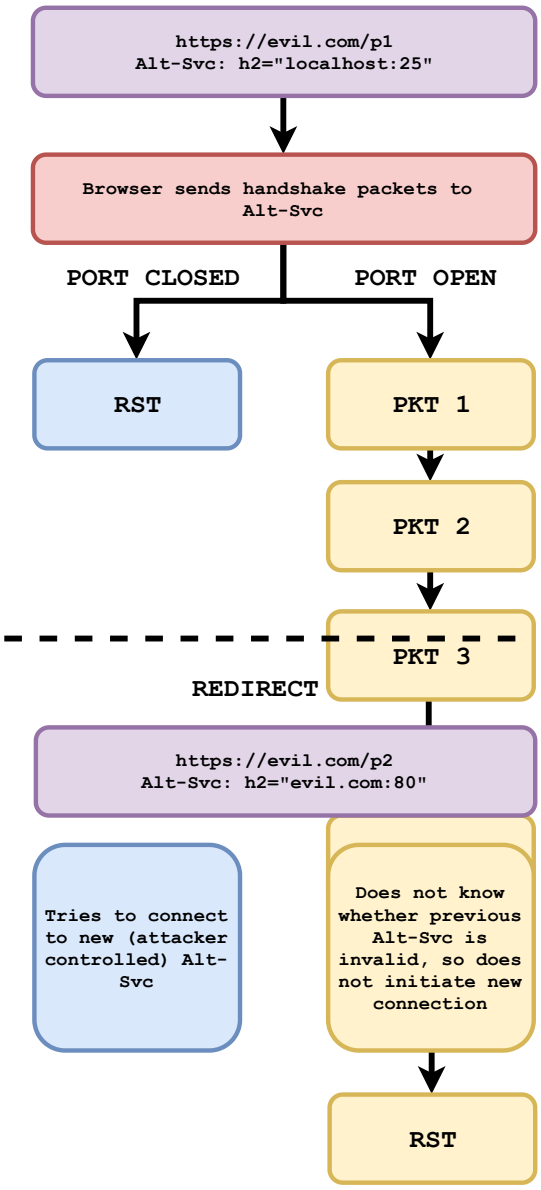

Figure 2: A browser based port scanner.

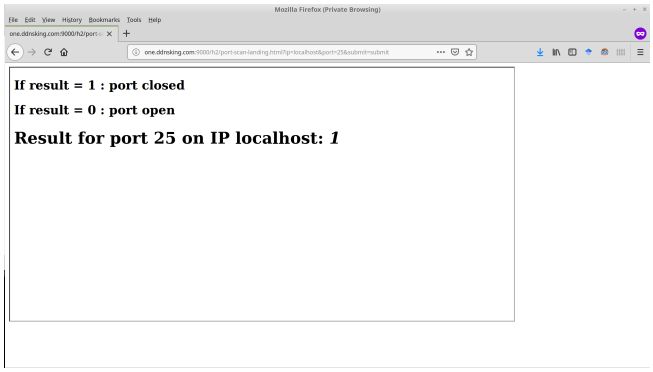

Figure 3: A screenshot of our tool for browser based port-scanning in action, as it scans port 25 on the victim's `localhost` through Firefox

private addresses; however, we were still able to scan public hosts in a distributed fashion. We have disclosed this issue to both Tor and Firefox. Mozilla has acknowledged the issue as CVE-2019-11728 [2] and has instituted the mitigation we suggest in Section 4.7. Tor will also receive Firefox's patch, since the Tor browser is based on Firefox's ESR series.

We have also implemented this attack on the Chrome and (Chromium based) Brave browser using a `quic` (`QUIC`) `Alt-Svc` endpoint advertisement. Here we are able to scan `UDP` ports, although both browsers prevent even alternative connections to certain blocked ports (*e.g.*, `SMTP` port 25) [44]. We are, however, able to scan arbitrary hosts for `UDP` ports through the browsers, which we believe is novel. We have disclosed this vulnerability to Google, and are in discussions about mitigations.

4.2 Bypassing malware/phishing protections

Chrome, Firefox and Brave browsers currently block domains known to host malware and phishing websites by checking for blacklisted domains through Google's Safe Browsing [15,41]. When a user attempts to visit a domain that is known to be associated with deceptive activity, the page is not loaded and a conspicuous warning is displayed (see Figure 4). Safe Browsing is triggered even if the first-party domain is safe but contains *any* third party content from a suspicious website (*e.g.*, an image from a blacklisted domain). In this scenario, Chrome and Brave display the same warning as in Figure 4, blocking all content even from the clean first-party webpage. Firefox, on the other hand, only blocks the the unsafe third-party content, while still displaying all content from the safe first-party and other safe third parties. Under these conditions, the victim is precluded from visiting a site that is known to be malicious, even if a white-listed first-party domain is convinced (or colludes) to load content from the malicious website.

However, if a clean, whitelisted first-party specifies a black-

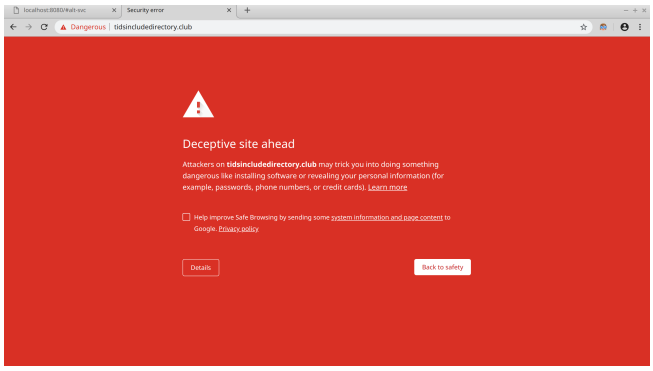

Figure 4: Warning for deceptive website on Chrome

listed domain as its `Alt-Svc`, then the Safe Browsing checks are skipped and all content is loaded from the malicious domain. This happens as long as the blacklisted website presents a proper certificate for the clean site (which may happen if, for example, the attacker controls or colludes with both sites). This works on all browsers that rely on Safe Browsing: Firefox, Brave, and Chrome. The attacker need only specify an `Alt-Svc` for `HTTP/2` with a malicious domain in Firefox, and an `Alt-Svc` for `QUIC` in Chrome and Brave. This loophole in Safe Browsing has been disclosed to Google and Mozilla, as of yet.

Effectively, `Alt-Svc`s allow the attacker to have "two-faced" content–clean content on the original endpoint and malicious content on the alternate endpoint. And so, anyone viewing the website through an `Alt-Svc`-aware browser will see the malicious content, while others will see the clean content. Hence, even if a user tries reporting the "clean" site as malicious to an online clearinghouse (like Safe Browsing's Reporting page), automated tools that are not `Alt-Svc`-aware will not see any of the malicious content.

In fact, a website that is itself whitelisted but has a blacklisted `Alt-Svc` also bypasses many popular online site checking tools like VirusTotal, URLVoid, Sucuri and IPVoid [35, 48, 50, 54]. These websites accept a user-provided `URL` and evaluate the safety of the corresponding website by comparing it against their own database. We have found that if the main first-party domain is not on any blacklist for these scanners, the website is reported as safe even if the `Alt-Svc` points to a blacklisted domain. This should not happen because even though the main domain points to a whitelisted website, all the content is effectively loaded from the blacklisted site. Hence, all protection mechanisms should not only check the first-party domain, but also check the `Alt-Svc` to see if it is in the blacklist before marking a website as safe.

4.3 Distributed Denial of Service

Firefox and Tor browsers do not maintain a memory of broken `Alt-Svc` endpoints (unlike the Chrome and Brave browsers).

Server	Client data (KB)	Server data (KB)	Round trip length (s)
`smtp.bu.edu:465`	0.5	29	0.3
`smtp.gmail.com:465`	0.6	3.0	0.1
`www.google.com:443`	0.5	2.8	0.05
`www.facebook.com:443`	0.5	3.1	0.03
`smtp.zoho.com:465`	0.7	4.5	0.4
`imap.gmail.com:993`	0.6	3.0	0.07

Table 1: Servers that send back a lot of data compared to the client. Hence, this is low effort on the part of the client, but high effort on the server, making it ideal for DDoS.

Hence every time a page is reloaded, a connection to its `Alt-Svc` is attempted (if one is specified). This happens even if the browser had attempted a connection to the same `Alt-Svc` in a previous load and realized that it was broken (*i.e.*, due to certificate mismatch or some other issue). As such, if we produce a reload loop in an `iframe`, we could force the victim's browser to repeatedly initiate `TLS` connections with a target server/port combination (as in the "port scanning" attack in Section 4.1), effectively denying service to it.

This is different from the standard JavaScript DDoS attacks, because Firefox and Tor do not check the port number of the `Alt-Svc` connection. For example, the attacker could force a number of distributed victims to repeatedly initiate `TLS` handshakes to specific non-`HTTP` speaking services (*e.g.*, secure email and `FTP` servers that run on ports whose access is otherwise blocked from JavaScript), and force them to repeatedly present their certificates. This is a classic reflected amplification attack: fairly low effort on the client's part because the client hello packet is very small, but high effort on the server's part as the server hello packet may be quite large due to large server certificates. The ampflication is presented in Table 1 (columns 2 and 3), for some concrete servers; for example, a connection request to `smpt.bu.edu` (port 465) requires the server to respond with roughly 60x more data than the client sends. Furthermore, the short `TLS` handshake round-trips in the final column of the table are reflective of the correspondingly short intervals at which the browser can force the target machine to present its certificate, further amplifying the transmission/reception imbalance over a short period of time. This amplification attack could thus exhaust a target server's resources, either from a single client or as part of a distributed attack where multiple browsers continuously ask these non-`HTTP` services to present their certificates.

Amplification attacks, in general, have been known to take down web servers, as happened for hours in 2014 [19, 53] with Wordpress's `pingback` feature . However, the Wordpress techniques, as well as other prior JavaScript-based DDoS

Vulnerable Servers
smtp.bu.edu:25
ftp.bu.edu:21
smtp.cornell.edu:25
smtp.zoho.com:587
smtp.outlook.com:25

Table 2: Live servers that have long time-outs (at least 30 seconds) for Alt-Svc connections, making them vulnerable to DDoS attacks.

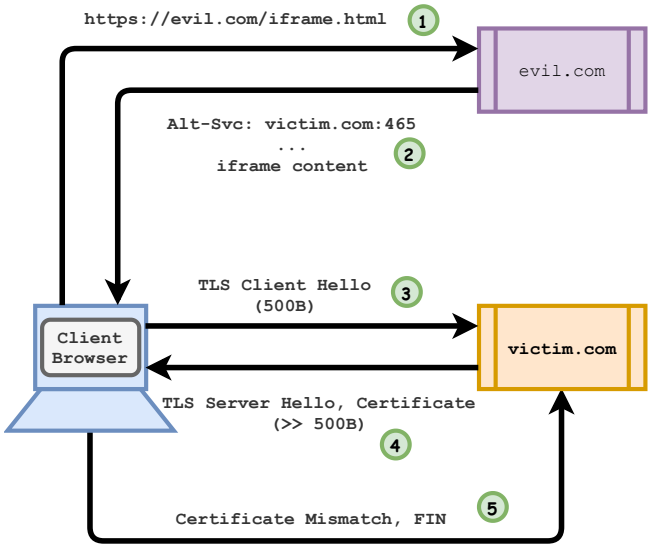

Figure 5: Data amplification DDoS attacks.

techniques, were limited to attacking web servers, whereas our attack can attack a broader range of services (*e.g.*, e-mail severs). Needless to say, our attack should not be possible from a typical browser, which should only be connecting to HTTP speaking endpoints.

More generally, even services that do not speak TLS can be targeted. Many plain-text email and FTP servers have long timeouts for every connection; an example list of such (live) services is provided in Table 2. For instance, consider the situation where the attacker forces a victim browser to send a TLS client hello packet to a plain-text email service (i.e., SMTP port 25) by appropriately setting its Alt-Svc to point to that service. Even though the email server does not understand the hello packet, it still keeps the connection open for as long as the browser requests–30-40 seconds, after which point the browser itself typically terminates the connection. If enough victim browsers are made to target a server, it may be possible to exhaust all available connections on the target server as yet another denial of service.

Both Firefox and Tor have confirmed this issue and have instituted the patch we suggest in Section 4.7.

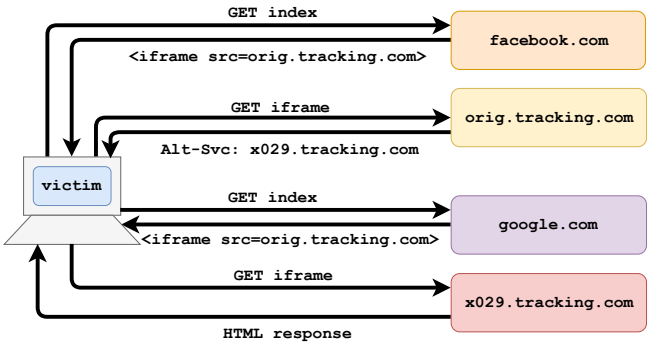

Figure 6: Third party tracking using Alt-Svc.

4.4 Tracking

By specifying a unique Alt-Svc for each user, and observing subsequent user requests, an attacker could track a user both as a first-party website and a third-party iframe or image, as illustrated in Figure 6. On Firefox, if the original endpoint uses HTTP/1.1 and specifies an Alt-Svc endpoint for HTTP/2, the Alt-Svc endpoint is always used over the original, since it presents the newer protocol. Further, the Alt-Svc header is cached for as long as the attacker specifies via the ma parameter, persisting across sessions. Hence, as long as the user does not clear their browsing history, this can be reliably used for persistent tracking. Similar behavior is seen for the QUIC protocol with Chrome, and even Brave, which is supposed to safeguard users from such persistent tracking. In fact, Alt-Svc tracking also bypasses the built-in third-party tracking protection mechanisms on Firefox and Chrome, when such features are enabled. We have notified all affected vendors of this issue and suggested mitigations to prevent abuse of the header. The only exception to this attack is Tor, which disallows disk persistence and supports first-party cache isolation, thus preventing any cross-session and third-party tracking, but still allowing same-session first-party tracking.

Finally, since the Alt-Svc header allows specifying a unique IP address and port combination, an ISP, which cannot decrypt connection data, can track the use of a unique Alt-Svc header. Most hardware vendors today randomize MAC addresses, a common tracking identifier of recent use, making Alt-Svc an attractive alternative [3, 12]. In a sample use-case, an ISP can embed within its network login page a tracking iframe that specifies a unique Alt-Svc as in Figure 7. This makes the victim cache the unique Alt-Svc header for the tracking domain as part of the "planting phase" noted as Step 1 in Figure 7. In the "tracking" phase (Step 2 in the figure), the ISP (and this could be a different ISP than before) inserts the same tracking iframe into their network login page and checks which unique Alt-Svc endpoint the client loads. Unique third-party cookies could perform iframe tracking in a similar manner, but they are currently

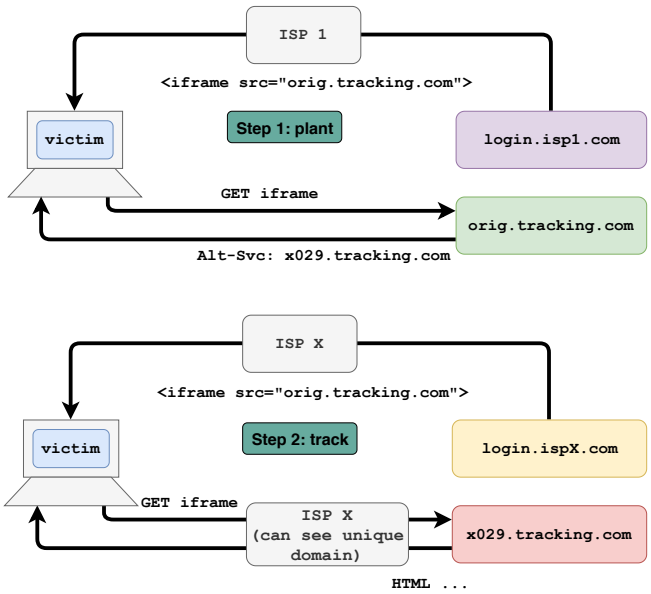

Figure 7: Network level tracking using `Alt-Svc`.

blocked by many services [37]. We tested, and appropriately disclosed, ISP tracking on Chrome, Brave and Firefox and Tor, and it is effective on all browsers except Tor (which practices first-party cache isolation).

4.5 Network level browser history exfiltration

ISPs (or any other active man-in-the-middle attacker) may also abuse `Alt-Svc` to exfiltrate a browser's history, without requiring any user interaction.

This history exfiltration attack involves a technique similar to the network level tracking presented in Section 4.4. Let us assume that the ISP wants to know if a particular user visited `www.illegal.com` sometime in the past (while on a different ISP that was outside the window of observation of the current ISP). For the sake of exposition, assume that `www.illegal.com` also sets an `Alt-Svc` header that points to its load-balancer `alt.illegal.com`. Now all the ISP needs to be able to do is to insert an image or `iframe` from `www.illegal.com` within their login web page (or any other webpage that they control). Then the ISP needs to monitor from where the image/`iframe` is loaded –the original domain, or the `Alt-Svc` domain. If it is loaded from the `Alt-Svc` endpoint, then the user must have visited the website before because there is no other way for the user's browser to know of the `Alt-Svc` endpoint than to have cached it from a prior visit. Otherwise, the user has not visited the website before. The granularity with which the ISP can exfiltrate history depends on the `ma` value set in the website's `Alt-Svc`, which is as short as 4 weeks for websites like the Google search engine.

4.6 Triggering anomaly false positives on Intrusion Detection Systems

Our port access capabilities might be useful in triggering Intrusion Detection Systems (IDSs). An attacker need only specify `Alt-Svc` values for the target service, and let the victims visit the malicious website to initiate connections to the `Alt-Svc` endpoint. The victim could, for example, be forced to connect to unusual ports (like port 0 through Firefox or Tor), access long chains of consecutive ports (mimicking a port scan), or send `UDP`-traffic to services not typically expecting it. All of these unusual behaviors may trigger some IDS systems, adding to their false positive maintenance load. Of course, the effectiveness of this approach depends on the specific configuration of a given IDS, and thus we only present it as a *potential* malicious use-case for the `Alt-Svc` header.

4.7 Mitigations

Many of the issues discussed about `Alt-Svc` are fundamental to its design, and so mitigations are not straightforward. Nevertheless, there are a few fixes that can be employed. First of all, like Chrome and Brave, Firefox and Tor should make `Alt-Svc` headers adhere to the same port blacklist as regular URLs [22]. Indeed, Firefox lets a browser access *any and every* port through an `Alt-Svc` header, whereas it blacklists certain ports otherwise. Blocking popular non-`HTTP` ports will make distributed port scanning of these ports a lot harder. It will also deter DDoS attacks, since ports that are likely to be running interesting services will no longer be accessible through `Alt-Svc`. In response to our disclosure, Mozilla has updated the blacklist to prevent these two attacks, and published this as CVE-2019-11728 [2].

In fact, all browsers should also check both primary domains and alternative service endpoints against their blacklists. Skipping blacklist checks for `Alt-Svc` allows an attacker to load content from malicious blacklisted domains and undermines the phishing and malware protections of Safe Browsing. For the same reasons, website evaluators such as VirusTotal URLVoid, IPVoid, and Sucuri should also check `Alt-Svc` endpoints against blacklists. Even automated scanners that assess the content of websites to generate these domain blacklists should be `Alt-Svc` aware due to the "two-faced" content that `Alt-Svc` headers make possible.

Further, like Tor, all browsers–Firefox, Chrome and Brave, should practice first-party cache isolation for the `Alt-Svc` header, which can prevent the tracking and history exfiltration attacks presented in Section 4.4 and 4.5. If not this, then they should at least prevent websites from setting `Alt-Svcs` through third-party content like `iframes` and images. This would help impede third-party tracking.

Finally, all browser vendors should also present a user option to disable `Alt-Svc` headers–currently this feature is enabled by default and cannot be disabled by users.

5 Conclusion

We have shown how the relatively new but widely adopted `Alt-Svc` header can be abused for various attacks, including distributed port-scanning (of external and internal networks), bypassing domain blacklists, DDoS of non-`HTTP` services, history exfiltration and tracking (on first- and third-party websites, and at the network layer), and possibly triggering Intrusion Detection Systems. This is despite (i) a considerable history of `HTTP` attacks and defenses over almost thirty years, (ii) development by independent, highly competent, browser-developer teams across several companies, and (iii) the simplicity and straightforwardness of the header. In the immediate mode, we hope that our mitigations will be implemented to resolve as many of these attacks as possible. However, as `Alt-Svc` gains support throughout the browser community, we further hope that this work highlights and adds to the types of concerns that should be considered.

References

[1] Advanced ddos attack protection. https://www.cloudflare.com/ddos/.

[2] CVE-2019-11728: Mozilla found. security advisory.

[3] Privacy: Mac randomization. https://source.android.com/devices/tech/connect/wifi-mac-randomization.

[4] Disable tls session resumption and session ids. https://trac.torproject.org/projects/tor/ticket/4099, 2011.

[5] Persistent and unblockable cookies using http headers. https://www.nikcub.com/posts/persistantand-unblockable-cookies-using-http-headers, 2011.

[6] Ddos attack from browser-based botnets that lasted for 150 hours. https://thehackernews.com/2013/11/ddos-attack-from-browser-based-botnets.html, 2013.

[7] Issue 392575: Implement the alt-svc spec. https://bugs.chromium.org/p/chromium/issues/detail?id=392575, 2014.

[8] http/2 alt-svc support. https://bugzilla.mozilla.org/show_bug.cgi?id=1003448, 2015.

[9] Non traditional costs of financial fraud. https://www.saveandinvest.org/sites/default/files/Non-Traditional-Costs-Of-Financial-Fraud-Survey-Findings.pdf, 2015.

[10] List of public corporations by market capitalization. https://en.wikipedia.org/wiki/List_of_public_corporations_by_market_capitalization, 2019.

[11] Furkan Alaca and Paul C Van Oorschot. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 289–301. ACM, 2016.

[12] Nick Arnottt. What's really happening with ios 8 mac address randomization? https://www.imore.com/closer-look-ios-8s-mac-randomization, 2014.

[13] Mika D Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. *Available at SSRN 1898390*, 2011.

[14] David Baron. :visited support allows queries into global history. https://bugzilla.mozilla.org/show_bug.cgi?id=147777, 2012.

[15] Google Safe Browsing. Making the world's information safely accessible. https://safebrowsing.google.com/.

[16] Tomasz Bujlow, Valentín Carela-Español, Josep Sole-Pareta, and Pere Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[17] Yinzhi Cao, Zhanhao Chen, Song Li, and Shujiang Wu. Deterministic browser. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 163–178. ACM, 2017.

[18] Hanqing Chen. Privacy tools: How to block online tracking. https://www.propublica.org/article/privacy-tools-how-to-block-online-tracking, 2014.

[19] Daniel Cid. More than 162,000 wordpress sites used for distributed denial of service attack. https://blog.sucuri.net/2014/03/more-than-162000-wordpress-sites-used-for-distributed-denial-of-service-attack.html.

[20] A. Dabrowski, G. Merzdovnik, N. Kommenda, and E. Weippl. Browser history stealing with captive wi-fi portals. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 234–240, May 2016.

[21] Anupam Das, Nikita Borisov, and Matthew Caesar. Tracking mobile web users through motion sensors: Attacks and defenses. In *NDSS*, 2016.

[22] MDN Web Docs. Mozilla port blocking. https://developer.mozilla.org/en-US/docs/Mozilla/Mozilla_Port_Blocking.

[23] Peter Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[24] Edward W Felten and Michael A Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 25–32. ACM, 2000.

[25] David Fifield and Serge Egelman. Fingerprinting web users through font metrics. In *International Conference on Financial Cryptography and Data Security*, pages 107–124. Springer, 2015.

[26] Internet Engineering Task Force. Http alternative services. https://tools.ietf.org/html/rfc7838.

[27] Internet Engineering Task Force. Opportunistic security for http/2. https://tools.ietf.org/html/rfc8164.

[28] Dan Goodin. How a website flaw turned 22,000 visitors into a botnet of ddos zombies. https://arstechnica.com/information-technology/2014/04/how-a-website-flaw-turned-22000-visitors-into-a-botnet-of-ddos-zombies/, 2014.

[29] J Grossman and M Johansen. Million browser botnet (2013), 2013.

[30] Jeremiah Grossman. Browser port scanning without javascript. https://blog.jeremiahgrossman.com/2006/11/browser-port-scanning-without.html.

[31] Jeremiah Grossman. Tracking users with basic auth. http://jeremiahgrossman.blogspot.com.es/2007/04/trackingusers-without-cookies.htm, 2007.

[32] Leviathan Security Group. The double-edged sword of hsts persistence and privacy. http://www.leviathansecurity.com/blog/the-double-edged-swordof-hsts-persistence-and-privacy, 2012.

[33] Ossi Herrala. How did i turn my browser into a port scanner? tricksy but doable. https://medium.com/hownetworks/how-did-i-turn-my-browser-into-a-port-scanner-tricksy-but-doable-c37db85f9adc.

[34] Gareth Heyes. Exposing intranets with reliable browser-based port scanning. https://portswigger.net/blog/exposing-intranets-with-reliable-browser-based-port-scanning.

[35] IPVoid. Ip blacklist check. https://www.ipvoid.com/ip-blacklist-check/.

[36] Ryo Kamikubo and Taiichi Saito. Browser-based ddos attacks without javascript. *International Journal of Advanced Computer Science and Applications*, 8(12):276–280, 2017.

[37] Matt Klein. How to block third-party cookies in every web browser. https://www.howtogeek.com/241006/how-to-block-third-party-cookies-in-every-web-browser/.

[38] David Kohlbrenner and Hovav Shacham. Trusted browsers for uncertain times. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 463–480, 2016.

[39] Declan Mccullagh. Exclusive: Google's web mapping can track your phone. https://www.cnet.com/news/exclusive-googles-web-mapping-can-track-your-phone/, 2016.

[40] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in javascript implementations. In *Proceedings of W2SP*, volume 2, 2011.

[41] Support Mozilla. How does built-in phishing and malware protection work? https://safebrowsing.google.com/.

[42] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FC Wien. Fast and reliable browser identification with javascript engine fingerprinting. In *Web 2.0 Workshop on Security and Privacy (W2SP)*, volume 5, 2013.

[43] Dror Nahumi. Web isolation – a paradigm change in enterprise cyber attack defense. https://www.nvp.com/blog/web-isolation-paradigm-change-enterprise-cyber-attack-defense/, 2016.

[44] The Chromium Project. Ports blocked on chromium. https://chromium.googlesource.com/chromium/src/+/ba78b86d7b4ef5140ef9838d41ca471b3075ef32/net/base/port_util.cc.

[45] Get Cyber Safe. Phishing: How many take the bait? https://www.getcybersafe.gc.ca/cnt/rsrcs/nfgrphcs/nfgrphcs-2012-10-11-en.aspx.

[46] Michael Smith, Craig Disselkoen, Shravan Narayan, Fraser Brown, and Deian Stefan. Browser history re: visited. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[47] The SSL Store. 1.4 million new phishing websites are created every month. https://www.thesslstore.com/blog/1-4-million-new-phishing-websites-created-every-month/.

[48] Sucuri. Free website malware and security scanner. https://sitecheck.sucuri.net/.

[49] Jochen Topf. The html form protocol attack. https://www.jochentopf.com/hfpa/hfpa.pdf.

[50] Virus Total. Analyze suspicious files and urls to detect types of malware, automatically share them with the security community. https://www.virustotal.com/#/home/url.

[51] Matthew Traudt and Paul Syverson. Does pushing security on clients make them safer? hotpets 2019, 2019.

[52] Morgan True. Church street tracking of visitors via wi-fi raises privacy concerns. https://vtdigger.org/2016/03/16/church-street-tracking-of-visitors-via-wi-fi-raises-privacy-concerns/, 2016.

[53] Krassi Tzvetanov. Wordpress pingback attack. https://www.a10networks.com/resources/articles/wordpress-pingback-attack, 2016.

[54] UrlVoid. Website reputation checker. https://www.urlvoid.com.

[55] Patrick Verleg, MCJD van Eekelen, and HPE Vranken. Cache cookies: searching for hidden browser storage. https://www.cs.ru.nl/bachelors-theses/2014/Patrick_Verleg___3049701___Cache_Cookies_searching_for_hidden_browser_storage.pdf, 2014.

[56] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks. In *2011 IEEE Symposium on Security and Privacy*, pages 147–161, May 2011.

[57] Berend-Jan Wever. Localnetworkscanner. https://github.com/SkyLined/LocalNetworkScanner/.

[58] Michal Zalewski. p0f v3 (version 3.09b). http://lcamtuf.coredump.cx/p0f3/, 2014.