Self-Authenticating Traditional Domain Names

Paul Syverson Matthew Traudt
Center for High Assurance Computer Systems
U.S. Naval Research Laboratory
Washington, DC 20375
Email: firstname.lastname@nrl.navy.mil

***Abstract*—We introduce *Self-Authenticating Traditional (SAT)* domain names. SAT domains are traditional recognizable domains resolvable via the Domain Name System (DNS). They are also self-authenticating—they encode in the name itself a public key for authenticating the SAT domain. We present an implementation of our SAT domains for servers and a corresponding Firefox WebExtension that validates connections to them.**

SAT domains weave security directly into the fabric of the Web by building authentication into URLs themselves. Thus, by simply posting links to other SAT domains, a SAT site that a user trusts assures that user of the ability to make hijack-resistant connections to any of those domains. Because just the address attested in this way is sufficient for users to create a secure connection to a recognizable domain, we call this dirt simple trust. We present implemented examples of this and describe other channels to establish dirt simple trust.

The public keys we embed in SAT domain names are in the format of Tor onion service keys. Specifically, a SAT domain includes the encoding of an onion service public key as a subdomain of a registered domain name. But the client can be ignorant of Tor and need not direct traffic over any onion routing network to obtain our protections. This makes SAT domains compatible with other browsers and standard routing infrastructure. Nonetheless, our extension also works in Tor Browser.

We also explore systems developed and deployed by others that associate a self-authenticating domain with a traditional DNS domain using existing Web authentication mechanisms, but without building security in directly. Recently, major providers have deployed .onion alternative services to support load balancing and improved performance for Tor users. Though superficially similar to SAT domains, our analysis indicates that these alternative services are not actually self-authenticating. They also increase the effectiveness and impact of client tracking attacks acknowledged in the design of alternative services. We describe such attacks and describe another benefit of our WebExtension: it provides an interface allowing users to selectively block or permit alternative services.

***Index Terms*—Self-authentication, TLS Certificates, Tor Onion Services, Website Authentication, Alternative Services**

I. Introduction

Consider Alice, who is aware that her Internet connections to popular news and social media sites (e.g., cnn.com, news.yahoo.com, facebook.com, twitter.com) are sometimes hijacked by authorities. Alternatively, suppose Alice's connections to popular commercial sites (e.g., amazon.com, ebay.com) may be hijacked by criminals. Currently, Alice has limited protections against such concerns. Authentication of intended destination websites via TLS should prevent her from accepting connections attacked by address-lookup or route hijacks. But, as we will discuss, authentication can also be hijacked. She could avail herself of a VPN, if access to VPNs is not also blocked or affected by hijackers. Even if she can connect through a VPN, she must trust the VPN to not hijack her activity or sell information about it, or simply to adequately protect it—trusts that would be misplaced in a nontrivial fraction of commercial VPNs [1]. Furthermore, whatever local protections a VPN would provide, it would not prevent hijacks from occurring between exiting the VPN and her intended destinations.

Suppose, however, that Alice is accessing a self-authenticating domain name, a domain name that incorporates into itself the information needed to authenticate a connection to the domain. Specifically, suppose public keys are appropriately embedded into domain names of such recognizable sites that Alice attempts to visit, and suppose her browser is able to check the proper mutual authentication of such public keys with the TLS key in the certificate received when she visits one of these domains. In that case, she would be immune to today's common connection hijacks derived from DNS (Domain Name System) and TLS-certificate hijacks.

The threat of such hijacks has long been a concern and is well documented. As one recent example, in January 2019 FireEye reported very large scale manipulation of DNS records to facilitate the obtaining of fraudulent TLS certificates [2]. Shortly thereafter, the U.S. Department of Homeland Security's Cybersecurity and Infrastructure Security Agency (CISA) issued an emergency directive outlining such attacks targeting numerous U.S. excecutive branch domains and setting out required actions to mitigate them [3]. DNSSEC and other mitigations to DNS and TLS-certificate hijack and misuse have been in active development and deployment since the 1990s. While these efforts have made important infrastructure security improvements, incidents such as the above indicate that a significant threat remains.

Challenges to providing secure site access stem partly from an inherent tension between desired properties: a user should be able to readily identify or recognize the site, it should be hard to deceive users' perceptions about recognizing the site, and this should work for arbitrary users who may have no prior familiarity with the specific site. This tension is somewhat similar to Zooko's triangle, which conjectures that names can be any two of human-meaningful, secure, and decentralized, but not all three at once [4]. The compromise solution generally accepted by the current infrastructure puts the authentication of connecting to recognizable traditional domains under the control of certificate authorities (CAs) that vouch for the binding of the domain name to the TLS key used to secure a connection. We intend to maintain existing protections but to combine them with decentralized security mechanisms. Amongst other things, this will help prevent the sorts of attacks we described above.

A. Contributions

We introduce *self-authenticating traditional (SAT)* domains. These embed security directly into the URLs that comprise the nodes of the graph that is the World Wide Web, and do so in a way that counters hijack by either Certificate Authorities or someone attacking certificate issuance to perform such hijack. Nonetheless, SAT domains also incorporate meaningful, familiar names. Whether a client has linked to a SAT URL from another site or found it via a search engine, a trusted source can support hijack resistant connections to a site with a SAT address by attesting to just a meaningful displayed URL *by itself* without stating anything more than the URL, e.g., without saying anything about keys or other data. And all of this is backwards compatible with existing browsers and Web security protocols that have no awareness of SAT domain properties or mechanisms. Specific contributions set out in this paper include the following.

- We describe SAT domains and provide a specification of them. The SAT domains we describe are comprised of a traditional domain name base with a self-authenticating subdomain derived from Tor's .onion addresses. A SAT domain for the base domain example.com has the structure [onion-address].example.com; we thus also refer to the self-authenticating subdomain as an *onion subdomain*. SAT domain names are cryptographically bound to the keys that authenticate them, as well as to a DNS name, which may be familiar and recognizable by users. They are backwards compatible in that browsers with no cognizance of the offered protections will operate normally.
- We present examples of TLS Domain Validation (DV) certificates we obtained for SAT domains. We describe our implementation and deployment of corresponding SAT websites, to which we validate connection in a browser with a Firefox WebExtension for this purpose.
- We present our implementation of that Firefox WebExtension, which checks both that the key encoded in the onion subdomain authenticates a presented TLS certificate and that the TLS certificate authenticates the onion subdomain. We describe the various checks it performs using satis.system33.pw as an example traditional domain name, together with an associated SAT domain. We make all client and server software along with supporting files and demonstration videos freely available at https://github.com/pastly/satis-selfauth-domains .
- We describe other features of the WebExtension. These include a means for validation of SAT domains in the form of automatically updating lists. The lists are themselves provided via self-authenticating channels, and we offer a simple interface for users to select whether or not they trust a channel for attesting to SAT domain validity. The WebExtension also supports automatic rewriting of base domain URLs to their corresponding SAT domains, similar to HTTPS Everywhere [5]. In this way, they also support discovery of SAT domains.
- We describe HTTP Alternative Services that use Tor's .onion services as alternatives, for example as recently deployed by Facebook and Cloudflare. Like SAT domains, these also combine traditional domain names with self-authenticating domains. We analyze these and find, however, that they are not actually self-authenticating at all. We also discuss how they facilitate first and third party tracking and censorship. In light of this we add a feature to our WebExtension allowing users to permit or block alternative services before they are used.
- We discuss possible future developments and research directions, in particular those stemming from the novel SAT domain property of weaving security into the fabric of the Web itself.

II. Background and Basic Goals

Our focus in this paper is authentication, assurance that a client is connecting to a destination that is intended or recognizable. Our adversary may be able to direct clients to unintended destinations, but it should not be be able to convince clients that they are connected to the URL displayed in the address bar when they are not.

A. Adversary model

We are concerned with an adversary that can direct connections of clients to incorrect destinations. As in our opening example, this could be an adversary that manipulates DNS lookups for particular domains for a targeted class of users, such as those using ISPs in a particular jurisdiction. It could also be an adversary capable of BGP hijack of traffic to an IP range covering an intended destination.

Our adversary's other primary capability is to obtain fraudulent TLS certificates. In practice, this too can be accomplished in a number of different ways. The adversary might be a misbehaving Certificate Authority, in which case the CA can simply issue the fraudulent certificate itself. Alternatively the adversary might be able to attack the means by which a CA validates control of a domain when issuing a TLS certificate.

By far the most prevalent type of TLS certificate is a Domain Validation (DV) certificate. To validate domain control for issuing a DV certificate, the CA will check if the applicant can respond to a particular email message to that domain, can publish a specified DNS TXT record for that domain, or can publish a challenge nonce at a URL under that domain [6]. An adversary able to subvert any of these checks for the domain in question would be able to obtain a fraudulent TLS certificate for it. As mentioned in Section I, despite many advances such hijack is a significant threat. Note that even if DNS could not be directly hijacked for fraudulent TLS certificate issuance, BGP hijack has been shown to be effective [7].

TLS certificates can also be issued under Extended Validation (EV). This requires more extensive checks of legal ownership and control over a domain. Note, however, that even if a domain has an EV certificate, an adversary capable of obtaining a DV certificate by any of the above means can still hijack TLS connections to that domain for those clients it can direct to the wrong destination. Also, because it currently remains relatively easy for an adversary to manipulate the checks a CA performs for issuance of a DV certificate, a CA that directly issues a fraudulent certificate may have plausible deniability that this was the result of external attack. We will discuss in Section VII ways in which our approach can help counter such deniability.

B. System Properties

Our systems support the four following properties.

Authority-independent Authentication: No CA can usurp from a site's owner control over that site's authentication.

Dirt Simple Trust: Learning a site address (domain name) from a trusted party is itself sufficient to assure a user of her ability to securely connect to a familiar or recognizable destination.

Synergistic Backwards Compatibility: Existing Internet and Web protocols operate normally (absent attacks), and without the need for duplicated versions of content or links on sites. Names are traditional domain names as governed by DNS. Existing web security is supported and reinforced by the new mechanisms and vice versa.

Network Embedded Security: The World Wide Web is a graph comprised of nodes (URLs) and arcs between them (hyperlinks). Most existing security mechanisms are externally tacked-on to the nodes or arcs of the Web. SAT domains are Web nodes themselves, thus weaving security directly into the fabric of the Web.

These are glosses intended to convey the gist of each property. Thus, e.g., a simplification in glossing 'dirt simple trust' was to run together discovery and validation of an address by saying "learning an address" from a trusted party. But addresses are often discovered from a search result, text message, etc. that is "clicked on" rather than learned directly from a trusted party. For site addresses not learned from a relevantly trusted party, the trusted party would only be validating a site address discovered elsewhere. Nonetheless, the SAT address by itself is sufficient for a trusted party to validate it as correct: the meaningful name it contains and the contained information needed to authenticate connection to it are correctly associated with each other.

Also, SAT domains support but do not provide these properties by themselves. They rely on additional assumptions we will discuss below. For example, dirt simple trust relies on client software capable of checking properties of connections to such domain names. And, authority-independent authentication does not imply that authentication is established without any role for CAs. Rather, CAs can support site owner control over authentication, but they are unable to subjugate it.

Self-authenticating domain names obviously have advantages: As Vint Cerf recently observed, "Suppose, in lieu of domain names, one used a public key as an identifier and associated this with an Internet Protocol (IP) address. If one looked up the IP address in a registry of public key identifiers, one could then challenge the device at that IP address to show it still has the associated private key using a challenge/response protocol" [8]. And the advantages of self-authentication have long been recognized as useful more broadly than just for Internet destinations. PolicyMaker, the first trust management system, "binds public keys to predicates that describe the actions that they are trusted to sign for, rather than to the names of keyholders as in current systems" [9]. Though "current systems" was as of 1996, the previous quote from Cerf shows that providing this kind of trust for access to Internet sites remains a hope today.

C. Related Work

Our discussion of existing related work focuses on whether it does or does not provide our four security properties for websites: authority-independent authentication, dirt simple trust, backwards compatibility, and network embedded security. This is summarized in Table I.

1) Traditional domain names and certificates: As a baseline we consider a traditional domain name (e.g., example.com) with a valid TLS certificate from a CA. Obviously the TLS certificate is an example of authentication dependent on an authority. The domain name is merely human meaningful, lacking anything that would enable a user to confirm secure connection to the site, which makes dirt simple trust impossible. As this HTTPS-protected traditional domain is our baseline, it is by definition backwards compatible with itself. Finally, traditional domain names do not embed security into the structure of the Web at either the nodes (URLs) or arcs between them. While HTTPS and TLS are majorly beneficial and important to secure network communication, they remain external to the fabric of the Web itself.

2) Tor Onion Services: We emphasized in Section I the value of bringing self-authentication to traditional domain names. Another important contribution of our approach, however, is that it brings traditional domain names to self-authentication. There is a widely deployed system for self-authenticated site access, namely Tor's .onion addresses [10]. As vetted software used by millions, adopting the format of .onion addresses within our SAT domains and using their server code for corresponding signature generation was more prudent than creating our own.

Onion services have been available via the Tor network since 2004. They have the .onion top level domain, which was reserved in an IETF standard in 2015 [11]. The current version of onion addresses are 56 characters long, comprising of a base-32 encoding of an ed25519 key, a checksum, and a version number followed by ".onion" [12]. As an example, the Qubes secure operating system's homepage is

	Authority Independent	Dirt Simple Trust	Backwards Compatible	Embedded Security
Traditional Domain	N	N	-	N
.onion Address	Y	N	Y*	Y
YURL	Y	N*	N	Y
.onion Alt-Svc	N	N	Y	N
DNSSEC/DANE	N	N	N*	N
HSTS	n/a	n/a	Y	N
HPKP	Y*	N	Y	N
Cert Transparency	N	N	Y	N*
SAT Domain	Y	Y	Y	Y

TABLE I
SYSTEM PROPERTY GOALS PROVIDED SAT DOMAINS AND RELATED WORK. (AN ASTERISK IMPLIES A MORE SUBTLE ANSWER, DISCUSSED IN THE TEXT.)

sik5nlgfc5qylnnsr57qrbm64zbdx6t4lreyhpon3ychmxmie m7tioad.onion. Tor client software automatically verifies whether a connection to this address is signed by someone possessing the private key associated with the public key encoded in it. Onion addresses are thus self-authenticating, which implies authority-independence. But, authority-independent authentication is for the onion address, which is unlikely to be meaningful to users. As such, by themselves they can not provide dirt simple trust. For example, receiving the above onion address from a trusted party via a trusted channel and connecting to it would not authenticate for the user a connection to Qubes unless the user already associated that address with Qubes. How would the user know which meaningful entity she is being given the address for? The channel would have to separately convey that the onion address is bound to Qubes rather than that being implicit in the address.

We have previously described and implemented binding .onion addresses with traditional domain names via GPG signatures [13]. Though this binding could provide authority independence for meaningful, traditional domain names, it could not provide dirt simple trust: in that system, addresses by themselves are either a meaningful name or a name including the means to its own authentication, but not both at once. We later suggested integrating these using the approach described in the present paper [10].

Looking up Tor onion service directory information and communicating with Tor onion services both have some nice security properties, but they are not our focus in this paper, and the reader need not know anything about these topics to understand our work.

A major limitation on the protection onion services provide is that they are only reachable by clients running Tor. (Proxies such as Tor2web [14] can allow access to onionsites via browsers that do not direct connections over Tor. This provides broader access, but at the expense of security. For example, the user must completely trust the proxy for address lookup and address self-authentication checks and thus has virtually no authentication assurance when using them to connect to onionsites. In addition, such proxies are a simple centralized point for tracking users' onionsite activities.)

Besides not providing dirt simple trust or being reachable by non-Tor clients, only EV certificates are currently obtainable from CAs for onion addresses. As already noted, EV certificates require a much more extensive check of association between the party to whom the certificate is issued and the domain name for which it is issued. This is generally expensive and time consuming enough that it is mainly pursued by fairly large or established enterprises, such as corporations or sites with a need to offer content in censored environments. Facebook, Cloudflare, Duckduckgo, Buzzfeed, and ProtonMail are some of the entities with TLS certificates for onion addresses. DV certificates, which are available quickly and without cost from Let's Encrypt [15], cannot be issued for onion addresses.

Because onion addresses appear meaningless to humans, an adversary could set up an onion service and try to trick users into thinking that connections to his website are, for example, to the Qubes OS website. And if the adversary brute forces an address that matches the first and last several characters of a real address, it is all the more likely to escape user notice. We were able to brute force 6 characters in 3 minutes on a single 56 core machine running an early vanity onion service generator [16]. A recent survey of Tor users showed that about half of respondents verify 9 characters or less to check if an onion address is what they expect [17]. An adversary with a better-optimized vanity onion service generator or more computing power can easily deceive these users. Previous versions of Tor onion addresses were sixteen characters long. The same survey found over half of respondents resort to memorizing their favorite 16-character onion addresses, a feat we cannot expect of users with Tor's current 56-character onion addresses.

3) YURLs: While .onion addresses are global and secure, they are not human meaningful. They are also limited in compatibility to those few million users accessing the Web via Tor. In contrast to meaningless .onion addresses, the YURL approach to self-authenticating Internet sites is to provide "trust management for humans" [18]. YURLs provide secure, human meaningful names, but they do this by abandoning globally meaningful names. The name is whatever the user locally chooses to associate with the public key. They are authority independent and are clearly not backwards compatible. (Though never significantly deployed, YURLs were intended to replace rather than synergize with traditional domain names and associated security infrastructure.) Because they only use locally meaningful names, they do not so much fail to provide dirt simple trust as consciously eschew talking about it. Nonetheless, inasmuch as YURLs do embed public keys in pointers to resources, they do embed security into Web structure. Unlike YURLs, SAT addresses support both decentralized and centralized trust establishment for traditional domain names and the ability to rely on either one or on the strength that comes from appropriately combining the two.

4) Onion Services as Alternative Services: The HTTP Alternative Services (alt-svc) IETF standard permits servers to suggest to connecting clients an alternative network location (and possibly protocol) to be used when connecting to the origin server. "Alternative services do not replace or change the origin for any given resource; in general, they are not visible to the software 'above' the access mechanism. The alternative service is essentially alternative routing information that can also be used to reach the origin in the same way that DNS CNAME or SRV records define routing information at the name resolution level." [19]. Unlike HSTS and HPKP, which will be discussed presently, alt-svc settings are only established via headers sent by the origin server: there is no preload list, and clients must thus contact a server initially to be rerouted to the alternative service. Also unlike HSTS and HPKP, alternative services are intended to be optional. Clients generally should follow an alt-svc header if possible, but are not required to do so. As noted above, a user connecting to https://foo.com/ and rerouted to an alternative service will continue to see https://foo.com/ in the URL bar, and the TLS protection is essentially the same. (Thus the alternative service must have access to the same keys needed to authenticate a connection to foo.com as the origin server.)

Recently, some prominent Internet destinations, notably Facebook and Cloudflare, have begun using alt-svc headers to route visitors who are coming over Tor to onion services. A user connecting to, e.g., facebook.com via Tor Browser, will be rerouted for subsequent connections to the .onion address given in the alt-svc header, though this will essentially still appear to be a connection to facebook.com. Subsequent lookup should also be via Tor's onion service directory system. The initial connection, however, will be to Facebook's server as normally resolved by DNS for requests coming from the network location of that initial connection's Tor exit relay.

Alternative onion services have the potential to enhance usability for Tor users, although by their nature there is no simple way for a user to know if she is getting their protections or not. Consequently, when used by themselves they undermine the self-authentication that onion addresses provide. Thus, though they combine onion addresses with traditional domain names, they do so in a way that provides neither authority independence, dirt simple trust, nor Web embedded security. We will discuss other security aspects of .onion alt services in Section VI.

5) DNSSEC and DANE: The IETF's Domain Name System Security Extensions (DNSSEC) is a suite of specifications that add authentication and data integrity to DNS [20]. DNSSEC has achieved somewhat significant adoption: the root zone adopted it in the summer of 2010 [21] and about 8% of clients were using DNSSEC to some extent as of 2013 thanks primarily to Google's public DNS resolver [22]. Because the root zone and many TLDs support DNSSEC, domain owners can create proper chains of trust to the trusted root and disallow an adversary from forging responses from authoritative servers in the chain. DNSSEC protects name resolution no matter what application layer protocol is in use. Users have no obvious way of knowing whether their DNS lookups are protected by DNSSEC or not.

DANE [23] is a protocol that uses DNSSEC to give domain name owners control over which TLS certificates are allowed to be presented to users visiting their domains. They can specify that clients should only trust a specific certificate, certificates issued by a specific CA, or certificates passing validation under a specific trust anchor.

DNSSEC and DANE do not provide authority-independent authentication as they generally depend on TLS keys and CAs for authentication. They potentially provide something in the direction of dirt simple trust for users with a properly working DNSSEC-validating resolver and who know the domain they're resolving should be protected by DNSSEC. But, as noted above, there is currently no way for users to know this. In theory, clients who know nothing about DNSSEC/DANE can simply ignore these special DNS records, so the protocols are potentially backwards compatible. In practice, significant fractions of clients were found unable to resolve addresses at all for domains with DNSSEC deployed in a 2013 study [24]. Incentives and deployment issues continue to plague DNSSEC, with only three percent of domains worldwide adopting [25]. DNSSEC and DANE do not embed security into the Web, but for sites adopting them and clients properly configured to use them they do counter the DNS-based attacks that an adversary might use for carrying out the attacks we address.

6) HSTS and HPKP: HTTP Strict Transport Security (HSTS) is built into all major browsers, is widely adopted by site owners large and small, and its adoption continues to expand. In addition to a preload list shipped with browsers, dynamic HSTS enables web servers to send to clients a special HTTP header indicating that the clients should continue to use HTTPS for all requests to the current domain for a given length of time. If a client is unable to establish a secure HTTPS connection to the server, the browser will fail hard and safe, disallowing the user from visiting the site until the problem is resolved. HSTS provides significant security gains for users; however, it gives third parties the ability to track and censor users via the pushing of header state that is not blockable by any current browsers and is harder to detect or remove than cookies [26].

HTTP Public Key Pinning (HPKP) allows web servers to send a similar HTTP header to clients, instructing them to only trust certain TLS certificates when visiting the server. This gives site owners control over secure site access not subject to CA hijack *once a client has visited a site and pinned the correct key(s) for it*. Dynamic HSTS and HPKP both require trust on first use (TOFU).

Once HPKP pins the correct key, this establishes authority-independent authentication of the website, even though the public key is from a TLS certificate likely signed by a CA. HPKP does not provide dirt simple trust because the user requires additional information (the correct key). It is backwards compatible because client software ignorant of HPKP headers will just ignore them. While HPKP does enable TOFU for TLS certificates, this is merely tacking on security as opposed to embedding it into the Web.

7) Certificate Transparency: Certificate Transparency (CT) is an initiative spearheaded by Google that supports auditing and monitoring the issuance of certificates [27]. CT provides publicly auditable append-only logs (cryptographically verifiable lists of issued certificates) operated by entities such as Google and DigiCert, as well as monitors that check log servers for suspiciously issued certificates, and auditors embeded in client software such as browsers. Major organizations and companies can be expected to set up monitors due to their self-interest in keeping track of what certificates exist for their domains, to detect issuance of malicious certificates, and to report incidents or take appropriate actions.

Though CT does not directly support authority independence, its record of issued certificates potentially provides incentive against CAs intentionally issuing fraudulent certificates to avoid the consequences of such misbehavior. Although, as noted above, for DV certificate issuance the plausibility of hijacked validation of domain control remains real enough that malicious CA behavior is neither necessary nor easily proved in general even if fraudulent certificates are discovered. CT does not support dirt simple trust. It also does not directly provide Web embedded security as we have defined it (security built into URLs or links). But, it does build logging into certificate issuance. In a future where CAs require (automated) proof of ownership of the private key corresponding to the self-authenticating key in a SAT domain, DV cert issuance becomes harder to hijack. In the case of fraudulent certificates for a particular SAT domain, CT logs would then provide evidence of CA misbehavior that is no longer easy to plausibly deny.

CT is backwards compatible with existing infrastructure. As of April 2018, Chrome acts as a simple auditor, requiring that TLS certificates are accompanied by signed commitmentments from CT logs to include the certificate [28]. In October 2018, Safari similarly began requiring for all TLS connections such commitments from CT logs [29]. Mozilla "supports the objective", but has no concrete plans to make Firefox require commitment to certificate presence in CT logs at this time [30].

III. SAT Domain Design

A self-authenticating domain name is one that incorporates into itself the information needed to authenticate a connection to the domain. A self-authenticating traditional (SAT) domain name therefore starts as a traditional domain such as example.com that also somehow binds to itself everything needed for self-authentication.

A. Strawman Design

A SAT domain needs a way to bind a self-authenticating identity (i.e., a key that a client expects from the URL to which it is connecting and that a website can prove it possesses) to its traditional cryptographic identity (created by the issuance of a TLS certificate by some trusted certificate authority). As the website already possesses a TLS key, it might make sense to use it for self-authentication too.

Suppose domain example.com has a TLS certificate with fingerprint $ABCDEF$. When referencing resources on their domain, the site owner could provide hyperlinks to ABCDEF.cert.example.com and instruct clients to verify the certificate they are presented with is $ABCDEF$. Browsers could be taught to recognize domains in this form as special, and they could require communication with these domains to be protected by a TLS connection with a valid TLS certificate with the fingerprint encoded in the name. This scheme would successfully bring self-authentication to traditional domain names and would be a valid SAT domain design.

- Notwithstanding the fact that an authority issued the TLS certificate to the site owner in the first place, no authority can prevent the site owner from using it for self-authentication.
- Names in this form enable dirt simple trust, as learning the name over a secure channel from a trusted party is enough to ensure secure connection to the recognizable destination example.com.
- Backwards compatibility is maintained: client software ignorant of these additional checks it should be performing will simply not perform the checks and be no more or less secure than before.
- The names themselves bootstrap properties for a secure connection, which weaves security directly into the Web.

This design is too inflexible, however. Site owners may want to use a variety of TLS certificates because, for example, they have data centers all over the world for load balancing purposes and want to limit the damage of accidentally leaking any one TLS private key. With this design they would no longer be able to have a single canonical name for their website, consequently making linking to their own site while maintaining load balancing difficult, if not impossible.

Furthermore, site owners would no longer be able to rotate their TLS certificates without invalidating links to their website all over the Web. We expect this to be unacceptable for certificates with lifetimes measured in years, but especially so for the 90-day certificates from the increasingly popular CA Let's Encrypt.

For these reasons, an identity key not directly and irrevocably tied to a specific TLS certificate is desired.

B. Design

Our SAT domain design has site owners generate an ed25519 key, encode its public part, and prepend it to their traditional domain. To support future integration with Tor onion services as well as take advantage of existing cryptographic software, we use Tor's onion addresses, and encode their ed25519 public key in the same way as Tor. As a working example, we have set up a website for the domain satis.system33.pw, and created an onion address for it. Similar to the strawman design, the DNS-resolvable domain name for this site simply prepends the onion address as a subdomain, namely, hllvtjcjomneltczwespyle2ihuaq5hy pqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw. This is the self-authenticating traditional (SAT) domain name associated with the base domain satis.system33.pw.

Since this design involves two different keys, the keys need to both bind themselves with the other. To bind the TLS key towards the ed25519 SAT domain key, we can simply include both the SAT domain and its traditional domain part in the TLS certificate. The standard TLS extension *Subject Alternative Name* (SAN) allows more than one domain name to be listed in a certificate.

To bind the ed25519 SAT domain key towards the TLS key, the site owner periodically signs messages indicating it wants to be bound to a specific TLS certificate and configures their Web server to include this message as a new HTTP header in response to all client requests. This completes the binding of a self-authenticating identity to a traditional identity with fewer downsides than the strawman design:

- No authority is involved in the process of the Web server authenticating itself to clients using the ed25519 encoded in its SAT domain.
- SAT domain names securely learned from a trusted party enable dirt simple trust.
- Browsers unaware of SAT domain protections will connect as normal and simply ignore the additional HTTP header, leaving users to be no more or less secure.
- SAT domain names embed security directly into the Web instead of simply tacking it on.

Site owners are free to use as many different TLS certificates as they wish as long as they also provide a signed message from their ed25519 key indicating the client should expect the given TLS certificate. Likewise they may rotate their TLS certificate as frequently as desired without ever having to change the hyperlinks to their site scattered all over the Web.

Our SAT domain design may seem similar to HTTP Public Key Pinning (HPKP) [31], as both effectively pin a specific public key. However, HPKP requires trust on first use and raises tracking and censorship concerns that SAT domains avoid [32]. Onion service protocols also allow identity private keys to be kept offline and only used to effectively delegate to signing keys, which makes them less vulnerable to theft or disclosure [12], though this is not yet implemented or deployed at time of writing. And, as noted above, HPKP supports neither dirt simple trust nor Web embedded security.

IV. Implementation

To implement our SAT domain design, only simple Web server configuration changes and software to generate ed25519 signatures are required on the server side, and the client side is entirely implemented in a Firefox WebExtension. We start this section with describing the necessary server side changes, proceed to describing the client browser extension, and finish with a security relaxation we optionally allow in our implementation in order to more easily facilitate adoption of SAT domains.

A. Server Implementation

We made simple additions to the Tor daemon software so that it generates the required signatures with the ed25519 key associated with an onion service. Tor makes a signature over the following data.

- A magic string
- A timestamp indicating the middle of a signature validity window
- Width of the signature validity window, in seconds
- A nonce
- The SAT domain name
- The fingerprint of the TLS certificate with which the domain should be associated.

Tor appends its signature to the end of the above data and writes everything to a file. These files are about 250 bytes in size (depending on the length of the SAT domain name). Tor regenerates this file periodically on an interval shorter than the signature validity period. Our implementation defaults to having a validity window of 7 days and to regenerating this file every 3 days, though these times are configurable.

A script runs every few hours that reads the file Tor creates, encodes the contents with base64, and configures the Web server to add the base64-encoded data as an HTTP header in its responses. Browsers ignore the header unless they have our extension installed. Popular Web servers such as Nginx and Apache support the simple configuration option necessary.

Because generating an ed25519 signature is relatively cheap, only needs to be done once every few days, and is performed on less than a kilobyte of data, the additional server-side cryptographic activity is negligible and not worth evaluating. We also expect the overhead from periodic reading of the signature file and updating of the Web server configuration to be negligible and on the order of regular server maintenance tasks, such as log rotation, thus not worth evaluating.

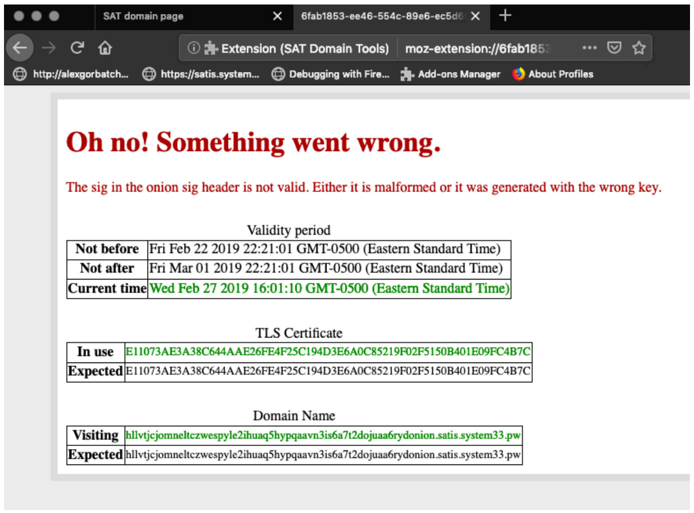
B. Client Implementation

We implemented a WebExtension that runs in Firefox 63 and newer. However, ultimately the authentication it provides is probably best incorporated directly in the browser rather than in an extension.

First of all, the extension does nothing to change normal browser TLS certificate checks. If the certificate is not trusted by the browser for any reason, the browser will halt the connection as usual. Only if the browser trusts the certificate does the connection proceed. At this point our extension performs the following checks.

- Is the user visiting a domain name formatted as a self-authenticating name (does it start with a string formatted as an onion address)? If not, it stops and returns control to the browser.
- Does the TLS certificate indicate an intended connection between the base domain name and the SAT domain name? Using our running example, it checks if both satis.system33.pw and hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw are listed in the Subject or SAN fields of the certificate.
- Is there an HTTP header containing data signed by the ed25519 key encoded in the domain?
- Is the current time within the signature validity window from the signed data, is the domain the client is visiting the same as the one in the signed data, and is the fingerprint of the server's TLS certificate the same as the fingerprint in the signed data?

If all checks pass, the connection completes as usual for a secure TLS connection that is now also self-authenticated. Other than the first one, all checks fail hard and safe, producing an error message such as in Figure 1. These checks are fast enough to be imperceptible.



Oh no! Something went wrong.

The sig in the onion sig header is not valid. Either it is malformed or it was generated with the wrong key.

Validity period

Not before	Fri Feb 22 2019 22:21:01 GMT-0500 (Eastern Standard Time)
Not after	Fri Mar 01 2019 22:21:01 GMT-0500 (Eastern Standard Time)
Current time	Wed Feb 27 2019 16:01:10 GMT-0500 (Eastern Standard Time)

TLS Certificate

In use	E11073AE3A38C644AAE26FE4F25C194D3E6A0C85219F02F5150B401E09FC4B7C
Expected	E11073AE3A38C644AAE26FE4F25C194D3E6A0C85219F02F5150B401E09FC4B7C

Domain Name

Visiting	hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw
Expected	hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw

Fig. 1. SAT header signature failure

C. Deployment Rollout Trade-off

For a domain to offer a SAT version of itself there are three primary requirements: it must configure its Web server to respond to requests to the SAT domain, it must obtain and offer a TLS certificate containing both the SAT domain and its base domain, and it must support the production and offering of onion-cert credentials—signed HTTP headers in which the onion address effectively authenticates the TLS certificate.

An institution may find that obtaining a new TLS certificate is the hardest requirement for setting up a testbed deployment of a SAT domain for their website, but they may already have a certificate containing an appropriate wildcard. We thus relax the requirement for including the SAT domain in the certificate in order to facilitate initial adoption of SAT websites. As normal, if the wildcard *.example.com is present in the TLS certificate, the browser allows a TLS connection to [onion-address].example.com. Our extension will then perform all the same checks as above, except for requiring the inclusion of the SAT domain in the TLS certificate.

The relaxation gives up any public logging or validation of that binding by Certificate Transparency [33] or similar mechanisms. It also means that, under the relaxed checks, anyone able to fraudulently obtain a certificate for *.foo.com can set up and authenticate a SAT domain for the base foo.com using any onion address for which they know the private key. This won't fool anyone who already has the correct SAT domain for foo.com and is checking for it, but it is a valid concern. We will discuss mitigations in the next section. Still, given the advantage of incorporation into Certificate Transparency logs and for greater security in other respects, we intend this relaxation to be a temporary trade-off to be phased out as adoption becomes more widespread and standard.

V. Dirt Simple Trust

Recall the example with which we started: Alice knows she is a likely victim of certificate hijack when connecting to all or some destinations. Nonetheless, if she already knows the SAT domain of her destination website, then she is guaranteed protection from the adversary described in Section II-A. An adversary that performs a DNS or BGP hijack against her and that obtains a TLS certificate that her browser would trust will still lack the ed25519 identity key needed for the SAT domain's self-authentication header, causing Alice's browser to fail safe and refuse the connection. But if she doesn't already know her destination's SAT domain, she needs a secure way to obtain it. Alternatively, if she has found a SAT domain for her intended destination via a search query or visit to an untrusted Web page, she needs a secure way to establish that it is the proper SAT domain for her destination.

If she knows and trusts Tom, who controls foo.com, and she has a way to receive authenticated messages from him, then he can give her the SAT address for foo.com over that channel. If she keeps track of this SAT domain somehow (as a toy example, in a bookmark), then her connections to it cannot be hijacked even if her WebExtension is operating under the rollout relaxation and he only has a wildcard certificate. He might give this SAT address to her on a business card, in a GPG signed message, in a link sent over a secure messaging application like Signal, etc. This is already a usable security win for many use cases. People at significant risk may not readily be in a position to learn about and properly employ the security they need, or they may simply slip up operationally. Requiring only a way to receive an address (and having properly functioning software) is both easier and less prone to error than, e.g., the Qubes .onion address scenario from the Section II-C2.

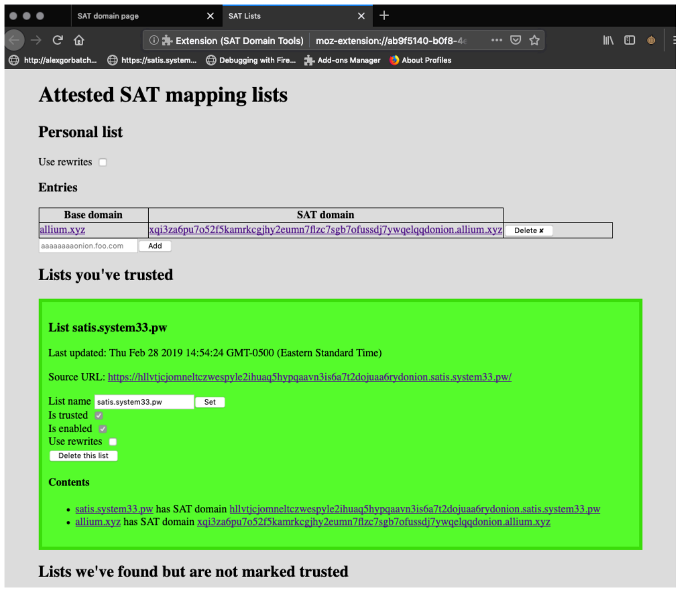
This baseline example of dirt simple trust has obvious scalability and usability concerns. Most notably, it requires Alice to set up secure communication channels with the owner of every website she intends to visit, which seems tautological (why not simply use those secure channels to actually visit the website?). We now describe ways to make dirt simple trust more usable and scalable while maintaining the fundamental element: a trusted party conveying or validating in one address a meaningful name indicating a destination and sufficient information to authenticate connection to it. Journalists and individuals who work with technologically-unsophisticated users significantly at risk from site-authentication hijack have told us this would be a valuable advantage in protecting such users and their communications.

As a first step to addressing these, imagine Alice also trusts Tom to properly validate the binding of other SAT domains to their base domains: Tom communicates over an authenticated channel with people he knows to control those base domains, verifies possession of appropriate keys, etc. He attests to such binding, and does so over a channel Alice trusts to authenticate Tom. These attestations should only be about such binding. If Tom is attesting about, say, cnn.com, he should not be implying anything about the accuracy of their reporting or quality of their editorial policy. To make clear that we are only describing attestation about binding of a SAT domain and nothing else, we will refer to it henceforth as *sattestation*.

For the original example of Alice trying to reach news and social media domains, Tom can provide Alice the SAT domain for some news reliability and safety organization, for example, Freedom of the Press Foundation or the Berkman Klein Center for Internet & Society. If Freedom of the Press Foundation's SAT domain sattests for the SAT domains of various news sites, Alice can use their sattestation to be assured her connections to CNN, Yahoo News, Facebook, Twitter, etc. are free from hijack—provided Alice trusts the sattesting organization. *All that needs to be communicated to Alice to establish this protection is domain names*: first the SAT domain of the sattesting organization, and upon successfully visiting that sattestor's SAT domain, the SAT domains for which it sattests. Note the dirt-simple requirement of just an address makes things simple and less error prone for the sattestor—as well as for the user and client. We have implemented SAT-domain-based sattestation in our WebExtension.

We emphasize that our sattestation lists do not automatically provide transitive trust à la PGP. While a PGP-esque system would instruct Alice to trust a SAT domain for CNN if, e.g., its mean shortest sattestation distance from her SAT domain is low enough, sattestation lists only provide "one layer" of trust. Tom's sattestation of the Freedom of the Press Foundation's SAT address does not by itself give Alice grounds to trust the SAT addresses on that organization's sattestation list. To correctly trust those SAT addresses, Alice would either have to independently trust the Freedom of the Press Foundation's ability and integrity to do sattestations, or she would have to separately trust Tom if he also asserts their trustworthiness in doing sattestations. And if she has established that trust, it does not provide grounds to further trust sattestations by any SAT domain on Freedom of the Press Foundation's sattestation list.

A. *Sattestation Lists Implementation*



Attested SAT mapping lists

Personal list

Use rewrites ☐

Entries

Base domain	SAT domain	
allium.xyz	xqi3za6pu7o52f5kamrkcgjhy2eumn7flzc7sgb7ofussdj7ywqelqqdonion.allium.xyz	Delete ✘

Lists you've trusted

List satis.system33.pw

Last updated: Thu Feb 28 2019 14:54:24 GMT-0500 (Eastern Standard Time)

Source URL: https://hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw/

List name satis.system33.pw
Is trusted ✓
Is enabled ✓
Use rewrites ☐

Contents

- satis.system33.pw has SAT domain hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw
- allium.xyz has SAT domain xqi3za6pu7o52f5kamrkcgjhy2eumn7flzc7sgb7ofussdj7ywqelqqdonion.allium.xyz

Lists we've found but are not marked trusted

Fig. 2. List of trusted SAT domain lists

As shown in Figure 2, our WebExtension records sattestation lists that the user comes across while browsing the Web, and provides a checkbox for her to indicate which sattestation lists she trusts. (They default to untrusted.) Elsewhere there is a settings checkbox that only permits connections to SAT domains if they are sattested by a trusted source. If Alice checks this box and has at least one SAT list marked as trusted, then even in the case that an adversary obtained a fraudulent certificate for a SAT domain of, e.g., cnn.com, her browser would not permit connection to that domain and would warn her accordingly. Note that even without requiring sattestation, this fraudulent certificate would have to be for a different SAT domain than the one CNN themselves set up; otherwise her browser would detect the failure to provide an appropriate signature header.

The sattestation list component of our WebExtension is also automated: it will periodically update any known list. The last simplifying usability element of dirt simple trust and sattestation that we have implemented to date is rewrite rules in the manner of HTTPS Everywhere [5]. A user can check a box stating that if the given sattestation list contains a base domain that she visits, then the WebExtension will rewrite the URL to contain the SAT domain instead of the base domain. A user can also provide her own rewrite rule for a SAT domain, which can be useful if, e.g., she learns of it via a Signal message from an appropriately trusted friend rather than from a sattestation list.

B. Sattestation Flexibility

A sattestation site trusted by Alice need not be a publicly reputed organization; the SAT domain she is told about by a trusted friend might simply be his own sattestation site. At the most extreme, Alice might trust only herself to sattest domains, thus the only SAT domains that she will count as sattested are those for which she personally validated with a party she trusts about control of the base domain. This underscores a valuable design property of sattestation: the same mechanism and user interface works across a range of use cases from accessing a small number of personally familiar domains to a large set of domains all over the Internet discovered by happenstance. Here are some examples.

- Alice could choose to only trust her Linux User Group's sattestation list of members' blogs and code repositories.
- Alice may use many Microsoft products, and subscribe to Microsoft's sattestation list for microsoft.com, live.com, office.com, office.net, microsoftonline.com, etc. Microsoft's software could be written to trust the same list in order to more securely download updates.
- A U.S. Government agency such as the Department of Homeland Security or the General Services Administration could produce a sattestation list for all .gov and .mil domains, which would obviate the real DNS attack mentioned in Section I.
- Companies may create a list for their domains to help protect their employees and contractors against accidentally leaking login credentials or sensitive information.
- A web-based VPN could be hosted at a SAT domain to make it hijack-resistent. Man in the Middle attacks have occurred against domains not intended to be reachable—much less accessed—by the public [34]. Making SAT versions of these domains can provide defense in depth against attacks on these internal namespaces.

Note that Microsoft could instead set up its own CA. While that is possible for entities the size of Microsoft or the U.S. Government, it is not for smaller entities. And running one's own CA by itself does not provide the same usability or protections. Many browsers do not by default trust certificates issued by the U.S. Government for example. And owning a CA will not prevent other CAs from issuing certificates for Microsoft's domains. Microsoft could pin certificates to their own or an existing CA. But, HPKP has already been deprecated in popular browsers, it would not protect first-time visitors, and pinning provides neither dirt simple trust nor Web embedded security.

C. List Performance

As SAT domain adoption grows, people are likely to adopt some sort of dirt simple trust scheme that requires local state, such as the one we implemented and described in Section V-A. As these lists of SAT domains grow in size, their impact on browser performance and page load time stops being negligible.

Table II shows the worst case performance impact a user can expect from our toy proof-of-concept WebExtension with aggregate list lengths up to 100,000 entries, assuming she either requires the SAT domains she visits to be sattested or she enables base-into-SAT domain rewriting (enabling both doubles the worst case). By repeatedly performing a search over all the configured sattestation lists for a domain that none of them contain—thereby forcing the code to check every entry of every list—we simulate the worst case when a user requires a domain to be sattested in order to allow herself to visit it or when she has enabled domain rewriting. These tests were performed on a 2015 MacBook Pro running macOS 10.14.3.

Num. SAT lists	Each list's length				
	1	10	100	1,000	10,000
1	0.00	0.02	0.06	0.48	4.14
10	0.02	0.04	0.32	3.96	41.8
100	0.12	0.42	4.04	41.3	-
1,000	1.96	6.30	50.2	-	-
10,000	24.4	75.1	-	-	-

TABLE II
WORST CASE TIME ADDED TO PAGE LOADING TIME (MS)

Even our naive unoptimized code can check 10,000 aggregate list items in less than 10 milliseconds in most cases, a duration still imperceptible to humans and an aggregate list size large enough to give programmers time to implement optimizations before the performance impact is too great. Straightforward caching optimizations such as memoization would yield significant savings for non-artificial work loads, and moving our WebExtension's functionality into the browser directly would not only allow it to be written in a much faster language such as C, C++, or Rust, but would also remove an artificial memory allocation limit placed on WebExtensions allowing for even larger lists and more memory-hungry storage solutions that enable constant-time lookup (e.g. a pair of hash tables, one for each direction of the SAT domain/traditional domain mapping).

VI. The Onion Alternative

SAT domain names combine the recognizability of registered domain names with the self-authentication of onion addresses, and as we have previously described, the user can choose an option to permit entering a base domain name (e.g., foo.com) in the URL bar of her browser and have this rewritten to a full SAT domain.

There is an at least superficially similar system currently deployed and in use that, given a regular domain name in the URL bar, will reroute a requested connection to an HTTP Alternative Service [19] at an .onion address. Given the apparent similarity, it is worth exploring these onion alternative services. Like SAT domains, they would seem to connect traditional domain names with self-authentication. Unlike SAT domains, they are intended only for use with Tor Browser. We will discuss other important differences below.

Recall from Section II-C4 that the alt-svc standard permits servers to suggest to connecting clients an alternative network location to be used when connecting to the origin server. Clients generally should follow an alt-svc header if possible, but are not required to do so, and a user rerouted to an alternative service will continue to see the original URL in the URL bar. The alternative service uses the same TLS keys needed to authenticate a connection to the origin server.

A. *Onion Or Not, Here I Come*

Starting in September 2018, Tor users visiting Cloudflare-backed websites (hereafter collectively simplified to cloudflare.com) could be rerouted via alt-svc header to one of ten .onion addresses cflare2n[...].onion [35], which would seem to offer the protection of self-authenticating addresses. And since the URL leading to these alt-service connections and displayed in the browser URL bar would simply be cloudflare.com (or the basic domain that cloudflare is backing), the user sees such connections to these self-authenticating domains as to a recognizable, traditional domain name. Further, since the TLS certificate supporting the authentication is for the displayed URL, such alt-service connections can use a DV certificate. Such .onion alternative services thus seem to provide the same properties as SAT domains. There are important differences, however, besides the obvous one of only being reachable by Tor Browser clients. To understand those other differences, it will be necessary to understand a bit more about onion alternative services in particular.

After an initial connection to cloudflare.com, further communication with that server should be re-routed to a .onion domain given in the alt-svc header; however, the RFC allows the client to choose to continue connecting to the origin server [19]. The user will always keep seeing cloudflare.com in the URL bar, and there is no simple way for them to determine if re-routing took place short of interaction with command line tools that indicate the state of Tor connections. An alt-svc header has a maximum age parameter that defaults to 24 hours. So, assuming her browser chooses to use the onion alt service, the user receives for that site and that day the routing and lookup protections Tor provides to onion services; but again, she has no usable way to verify this.

Inability to tell if a particular connection is via onion service protocols and inability to tell *which* onion alternative service is in use undermines onion service self-authentication. Onion addresses may not be human meaningful, but they are at least human verifiable. As long as the user can see the onion address, she can verify it's the one she is connecting to (alternatively, application software can trivially validate the match). And as long as she has successfully loaded content from it, she knows she is communicating with the correct server with no Men in the Middle. As an alternative service, neither the user nor even "the application that is using HTTP" [19] can verify this anymore: the user cannot tell if an adversary has hijacked her connection to cloudflare.com and given her a bogus .onion alt-svc header.

What's worse is that the adversary may give each user he attacks a different bogus .onion alt-svc header, allowing him to recognize, track, and censor individuals. "HSTS Supports Targeted Surveillance" describes both first-party and third-party surveillance and censorship of users via HSTS header state [26]. The same attacks can be conducted using alternative services rather than HSTS state. In fact they are easier to encode since a single .onion alt-svc header can encode the Web behavior history of previous alt services visited, without the need for multiple redirects.

RFC 7838 explicitly acknowledges the tracking potential of alternative services, although not the possibility of censorship. Nor does it discuss the prospect of another means to do cross-site tracking by advertisers, CDNs, etc., or of encoding the history of visit behavior in the offered alt services themselves and possibly using this to selectively offer content. Fortunately, because of Tor Browser's first-party isolation protections, third-party versions of these attacks are not possible via onion alt services. Nonetheless, onion alt services make first-party tracking and censorship attacks harder to detect: for example, after the initial obtaining of a TLS certificate, no DNS or BGP hijack is needed for clients to be routed to the wrong location.

Our purpose in describing onion alternative services was to describe a protocol currently in use that, at first glance, might appear to accomplish the goals of SAT domains. We thus say no more about these attacks. We will, however, end this section with a description of how we have added a feature to our WebExtension that counters the attacks we discovered.

B. *Alty Alty Onion Free*

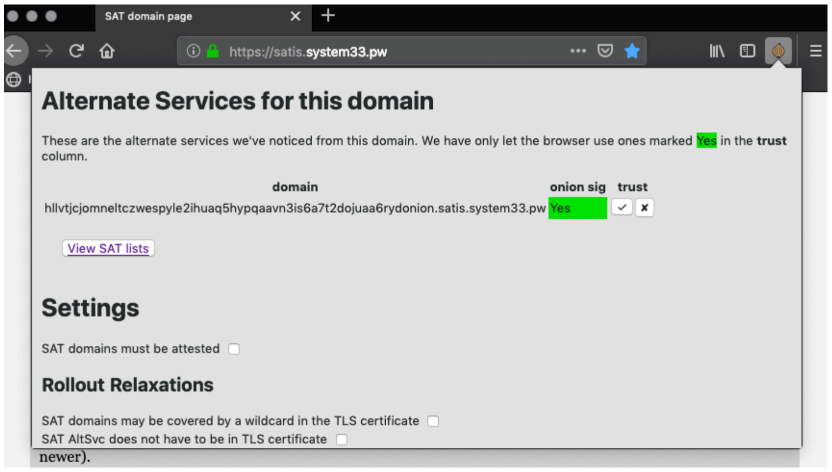
Our WebExtension filters out all alt-svc headers before the browser is allowed to learn of them and use them, taking advantage of RFC 7838's option of ignoring these headers. As shown in Figure 3, our WebExtension remembers alt-svc headers that it has seen, and it does not pass on alt-svc headers unless the user indicates she wants to allow them. For SAT domains and .onion domains listed as alternative services, our extension performs the client checks described in Section IV-B. If they fail, we indicate *No* in the "onion sig" cell seen in Figure 3.

This is simply a proof-of-concept interface and functionality that arose indirectly in our investigation of technologies related to SAT domains. Though it may prove useful for alt services in general, analysis and/or studies would be needed to determine a worthwhile usable security approach.

VII. Weaving Security into the Web's Future

The Web has followed a rich, centuries-old information technology tradition of developing something really cool that becomes massively popular or important, and then—as its general lack of any security becomes painfully clear—bolting some onto the developed technology.

This is not simply laziness or perpetuation of shortsightedness; predicting even the concepts of security applicable to a radically new technology and how it will be used and embedded into the structure of daily life and business is notoriously problematic. And once an indication of these begins to emerge, requiring whole-cloth adoption of "better" alternatives is generally as unrealistic as is preserving the status quo. Still the more we are able to integrate and improve security without



Alternate Services for this domain

These are the alternate services we've noticed from this domain. We have only let the browser use ones marked Yes in the **trust** column.

domain	onion sig	trust
hllvtjcjomneltczwespyle2ihuaq5hypqaavn3is6a7t2dojuaa6rydonion.satis.system33.pw	Yes	✓ ✗

View SAT lists

Settings

SAT domains must be attested ☐

Rollout Relaxations

SAT domains may be covered by a wildcard in the TLS certificate ☐
SAT AltSvc does not have to be in TLS certificate ☐

newer).

Fig. 3. Extension filters out alt-svc headers by default, and empowers user with choice to use them or not

doing too much violence to existing infrastructure, including existing security infrastructure, the better.

Abstractly, the World Wide Web is a network comprised of labeled nodes and directed arcs between them. Node labels are URLs consisting in full or part of domain names, and domains are the labels for which TLS certificates are issued. While TLS certificates add security to the Web, they are attached externally rather than internally as part of its structure. By contrast, SAT domains are themselves unique labels for nodes capable of having arcs directed to or from them. They thus weave security directly into the structure of the Web.

Recall our example of a U.S. Government sattestation domain for .gov and .mil SAT domains. If Alice is in a communications environment where almost all news sources (including Internet, print, broadcast, etc.), are censored, she might need a specific trust root to reliably discover and validate this sattestation domain. If, however, she has means to access Internet sites from multiple diverse network locations (e.g., her local ISP plus a few Tor exits in diverse locations), as well as relatively unrestricted access to popular print and broadcast media, then an announced U.S. Government SAT domain for .gov and .mil sattestation that appears in several of these places and goes unrefuted for a few days can be taken as genuine and correct. And because the SAT domain is a label for a Web node, her media sources are likely to reference it explicitly. Unlike bolted-on security such as TLS, Alice does not have to assume she's receiving the sattestation domain's security benefits without interference: if she connects without errors to its URL, she is receiving its security benefits.

As SAT domains become wider spread and the number of links between them grows, the search engine rankings of SAT domains will increase as well. As legitimate SAT domains bubble to the top of search results, it will be harder to hijack those clients that check SAT authentication even if they have not yet determined the correct SAT domain for their intended destination. This is gameable of course by using Search Engine Optimization (SEO), but not easily without detection by someone who would raise an alarm and report it. The more that security is woven directly into the Web, the higher the bar is for malicious parties to impersonate prominent organizations, companies, etc. in search results.

There is currently no simple way to determine if a fraudulent DV certificate was obtained from a misbehaving CA or if the CA experienced DNS or BGP hijack when validating control of the domain for which it issued the certificate. Honest CAs wishing to enhance their reputation for reliability thus have incentive to raise the bar for deceiving them—particularly if it can be done relatively cheaply, quickly, and easily. If, when issuing a DV cert that includes a SAT domain, the CA checked for possession of the private onion key along with its other usual checks, then it could not be tricked into issuing a cert for any existing SAT domain by anyone not possessing that key. TLS hijacks that are based on DNS or BGP hijack (such as those announced in January 2019 [2]) would not be possible. (It would still be possible to obtain a cert for the same parent domain but with a *different* onion subdomain, which presumably would not be on the same sattestation lists, would not match URLs for existing links from other sites to the original SAT domain, etc.)

The checks that should be performed to place a SAT address on a sattestation list have some similarity to those required to obtain an EV certificate. If possession of a private onion key were incorporated into the checks performed for issuance of an EV certificate, then the CA could also serve as sattestor for any EV certificate it issued. This highjack resistance for any browser trusting the CA for sattestation is a value add that can be advertised, creating an incentive for CAs to incorporate SAT addresses into the issuance of EV certificates, not merely DV certificates. This also shows the flexibility of the contextual trust provided by sattestation: it can scale down to trust in a single person or scale up to purely structural requirements of a CA issuing EV certs.

Not all impacts of SAT domains are simplifying. There are complexities that arise from weaving security into the Web's structure in this way. Revocation or rotation of the keys encoded in SAT domains requires updating links—changing the structure of the Web. As we noted in Section III-B, SAT domain keys could be kept offline exposing them to less risk. Nonetheless, it will sometimes be necessary to change them. The most reasonable way for this to interplay with Web dynamics in the context of SAT domains is an interesting research question that it would be better to more fully understand before significant deployment.

On the other hand, Web dynamics are not only made more complex by SAT domains. For example, abandoned domain names and subdomain names—particularly those for financial institutions—have been the target of adversaries hoping to capitalize on the former domain owner's reputation [36]. If the abandoned domains were SAT domains, however, and if CAs checked for control of the private onion key before issuing a new cert, then it would be impossible to obtain a new cert for any of these domains from an honest CA.

Also, note the mechanisms for authentication of SAT domains contain a built-in means to revoke TLS keys. If the onion-cert credential header does not sign the TLS certificate for the TLS key being used, then the WebExtension will flag this as an authentication failure. Further simplifying, there is no need provide a separate revocation list or OCSP status update for SAT domains—assuming the future validity of the header (default of about three days) is narrow enough or can be tuned appropriately. Besides reducing overhead of communication with a CA (by clients or by servers in the case of OCSP stapling), this means of TLS key revocation is authority-independent.

In a future where SAT domains have become widespread, there is a degree of protection provided even to those clients that do not perform the checks associated with SAT domains. As SAT domains become more commonly linked and sattested, the fraction of users that can be tricked by gaming ranking algorithms to get them to connect to doppelganger SAT domains goes down. While such attacks may remain useful in targeted situations, the incentives to perform them stemming from their general level of success diminish. Similar to the case above of protection for SAT-authentication-checking clients lacking sattestation information, the level and structure of protection from such "immunological" effects given different network and user parameters is another interesting avenue for research that arises from SAT domains.

Our extension does work in Tor Browser for purposes of checking self-authentication and valid binding of SAT domain with base domain. And as noted above, our onion subdomains have the same encoding as Tor's .onion addresses, and we use the same signature generation code as they do. But SAT address lookup in Tor Browser currently uses DNS, and Tor connections to SAT domains use the same routing as for any other destination on the Internet that is not a .onion domain. It would be natural to add for Tor users the superior protections of lookup and routing to .onion services by making the relevant changes needed so that these apply to SAT domains as well.

The interfaces we have designed are merely proof-of-concept. Best usability designs, best choice of parameter and settings, etc. would benefit from usability analysis, as would some of the dynamics issues raised above (though those are not merely usability issues). We have not, for example, provided a general policy framework or mechanism to decide for which SAT domains a particular sattestation site should be trusted to attest, much less how or by whom to enforce this. And, our proof-of-concept sattestation interface is via lists retrieved by clients from trusted entities, but it may be preferable to instead (or in addition) manage this via credentials that a SAT domain provides as a header.

Further, self-authentication of the address need not be in the subdomain; it is possible to put it in the URL path. This opens up many possibilities. For example, if CA/Browser Forum guidelines are ever relaxed to permit .onion addresses as SANs in DV certificates, then the WebExtension check could apply to SAT addresses in URL paths or path queries to make sure the registered domain name and .onion address are both appropriately in the certificate. Subject to the details, this would still support dirt simple trust and web-embedded security without the need to change DNS records. It might also remove a motivation to take advantage of the rollout relaxation. (Note that at the time .onion addresses were restriced to EV certificates, they were based on a much earlier, cryptographically weaker onion address system than the current one on which the SAT addresses and signatures described above are based.)

SAT domains address significant, recently demonstrated threats to Web security. And they do this in a way that is incrementally deployable, available now, and synergizes with existing Web infrastructure. But they also raise many interesting research questions, which we welcome you to join us in exploring.

Acknowledgments

The authors wish to thank Richard Barnes, Roger Dingledine, and Peter Eckersley, for helpful conversations, and thank Bill Budington, Georg Koppen, Ben Laurie, Wouter Lueks, Eric Mill, Seth Schoen, Nick Sullivan, Ryan Wails and our anonymous reviewers for helpful comments on drafts of this paper that greatly improved it.

References

[1] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, "An empirical analysis of the commercial VPN ecosystem," in *Internet Measurement Conference (IMC '18)*. ACM, 2018, pp. 443–456.

[2] M. Hirani, S. Jones, and B. Read, "Global DNS hijacking campaign: DNS record manipulation at scale," https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html, January 9 2019.

[3] C. C. Krebs, "Emergency directive 19-01: Mitigate DNS infrastructure tampering," https://cyber.dhs.gov/assets/report/ed-19-01.pdf, January 22 2019.

[4] B. Z. Wilcox-O'Hearn, "Names: Distributed, secure, human-readable: Choose two," https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html, October 12 2001.

[5] "HTTPS Everywhere," https://www.eff.org/https-everywhere.

[6] "How it works," https://letsencrypt.org/how-it-works/.

[7] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Using BGP to acquire bogus TLS certificates," in *Hot Topics in Privacy Enhancing Technologies (HotPETs)*, 2017.

[8] V. G. Cerf, "Self-authenticating identifiers," *Communications of the ACM*, vol. 61, no. 12, p. 5, December 2018.

[9] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *IEEE Symposium on Security and Privacy (SP), 1996*, May 1996, pp. 164–173.

[10] P. Syverson, "The once and future onion," in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Springer-Verlag, LNCS 10492, 2017, pp. 18–28.

[11] J. Appelbaum and A. Muffett, "The .onion special-use domain name," https://tools.ietf.org/html/rfc7686, 2015.

[12] D. Goulet, G. Kadianakis, and N. Mathewson, "Next-generation hidden services in Tor (Tor proposal 224)," https://gitweb.torproject.org/torspec.git/tree/proposals/224-rend-spec-ng.txt.

[13] P. Syverson and G. Boyce, "Bake in .onion for tear-free and stronger website authentication," *IEEE Security & Privacy*, vol. 14, no. 2, pp. 15–21, 2016.

[14] "Tor2web: browse the anonymous internet," https://www.tor2web.org/.

[15] "Let's Encrypt: Delivering SSL/TLS Everywhere," https://letsencrypt.org/2014/11/18/announcing-lets-encrypt.html, November 2014.

[16] Y. Angel, "horse25519 - an ed25519 vanity public key generator," https://github.com/Yawning/horse25519.

[17] P. Winter, A. Edmundson, L. M. Roberts, A. Dutkowska-Żuk, M. Chetty, and N. Feamster, "How do Tor users interact with onion services?" in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/winter

[18] T. Close, "Waterken™ YURL: Trust management for humans," http://www.waterken.com/dev/YURL/Name/, July 12 2004.

[19] M. Nottingham, P. McManus, and J. Reschke, "HTTP Alternative Services," https://tools.ietf.org/html/rfc7838, April 2016.

[20] D. Eastlake, "Domain Name System security extensions," https://tools.ietf.org/html/rfc2535, March 1999.

[21] "Root DNSSEC: Information about DNSSEC for the root zone," http://www.root-dnssec.org/.

[22] "DNS, DNSSEC and Google's public DNS service," http://www.circleid.com/posts/20130717_dns_dnssec_and_googles_public_dns_service/.

[23] R. Barnes, "DANE: Taking TLS authentication to the next level using DNSSEC," https://www.ietfjournal.org/dane-taking-tls-authentication-to-the-next-level-using-dnssec/.

[24] W. Lian, E. Rescorla, H. Shacham, and S. Savage, "Measuring the practical impact of DNSSEC deployment," in *Proceedings of the 22nd USENIX Security Symposium*. USENIX Association, August 2013, pp. 573–587.

[25] T. Le, R. van Rijswijk-Deij, L. Allodi, and N. Zannone, "Economic incentives on DNSSEC deployment: Time to move from quantity to quality," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018.

[26] P. Syverson and M. Traudt, "HSTS supports targeted surveillance," in *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2018.

[27] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," https://tools.ietf.org/html/rfc6962, July 2013.

[28] "2018 Certificate Transparency requirements by Google Chrome," https://www.ssls.com/blog/2018-certificate-transparency-requirements-google-chrome/.

[29] "Certificate Transparency policy," https://support.apple.com/en-us/HT205280.

[30] "PKI:CT," https://wiki.mozilla.org/PKI:CT.

[31] C. Evans, C. Palmer, and R. Sleevi, "Public Key Pinning Extension for HTTP," https://tools.ietf.org/html/rfc7469, April 2015.

[32] L. Tung, "Google: Chrome is backing away from public key pinning, and here's why," https://www.zdnet.com/article/google-chrome-is-backing-away-from-public-key-pinning-and-heres-why/.

[33] "Certificate Transparency," http://www.certificate-transparency.org/.

[34] Q. A. Chen, E. Osterweil, M. Thomas, and Z. M. Mao, "MitM attack by name collision: Cause analysis and vulnerability assessment in the new gTLD era," in *IEEE Symposium on Security and Privacy (SP), 2016*. IEEE, 2016, pp. 675–690.

[35] M. Sayrafi, "Introducing the Cloudflare onion service," https://blog.cloudflare.com/cloudflare-onion-service/, September 20 2018.

[36] T. Moore and R. Clayton, "The ghosts of banking past: Empirical analysis of closed bank websites," in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*. Springer-Verlag, LNCS 8437, 2014, pp. 33–48. [Online]. Available: http://tylermoore.ens.utulsa.edu/fc14ghosts.pdf