Towards Short-Lived Certificates

Emin Topalovic*, Brennan Saeta*, Lin-Shung Huang†, Collin Jackson† and Dan Boneh*
*Stanford University, {emint, saeta, dabo}@cs.stanford.edu
†Carnegie Mellon University, {linshung.huang, collin.jackson}@sv.cmu.edu

***Abstract*—The Online Certificate Status Protocol (OCSP) is as good as dead. It imposes a massive performance penalty on web traffic and has failed to mitigate the recent high-profile certificate security breaches of certificate authorities. Citing these fundamental protocol flaws, Google Chrome, one of the world's most popular browsers, is permanently disabling OCSP and taking direct ownership over certificate revocation. We will soon be living in a post-OCSP world where Google has become a single point of failure for certificate validation. We argue that certificate authorities should reassert control over the certificate revocation process by issuing certificates with a very short lifetime. These certificates complement browser-based revocation by allowing certificate authorities to revoke certificates without the cooperation of browser vendors, and without imposing a performance penalty on web traffic.**

We have implemented a prototype certificate authority and certificate update plugin for Apache that demonstrates feasibility of short-lived certificates. We have also implemented client-side pinning to short-lived certificates in the Chromium browser. Finally, we show that short-lived certificates complement browser-based revocation and address its major limitations; the two mechanisms can be combined to achieve secure, performant, and backwards-compatible browsing experience on the web.

I. Introduction

X.509 certificates are widely used by web browsers to authenticate destination servers [1]. It is current standard practice for certificate authorities (CAs) to issue certificates with a relatively long validity period such as a year or longer. Ideally, certificates are expected to be used for their entire validity period, but unfortunately, a certificate can go bad prior to its expiration date for many reasons. Private keys corresponding to the certificate can be stolen, the certificate could have been issued fraudulently, or the certificate could have simply been reissued (e.g. due to a change of name or association). For these reasons, X.509 defines mechanisms, including certificate revocation lists (CRLs) [1] and the Online Certificate Status Protocol (OCSP) [2], for revoking a certificate prior to its expiration date.

Notably, recent security breaches of certificate authorities (i.e. Comodo [3] and DigiNotar [4]) have resulted in fraudulently issued certificates exposed in the wild. However, current revocation mechanisms have been ineffective, as discussed in the next section, and browser vendors were forced to issue software updates to block bad certificates instead of relying on revocation checks. In fact, Google has announced plans to disable altogether OCSP in Chrome — one of the world's most popular browsers [5] — and instead reuse its existing software update mechanism to maintain a list of revoked certificates on its clients. For space considerations, their global CRL is not exhaustive, and can exclude the revocations that happen for purely administrative reasons. Network filtering attacks that block updates are still possible, but would require constant blocking from the time of revocation. Google's decision is mainly due to the ineffectiveness of soft-fail revocation checks, and also the obvious costs in performance and user privacy [6].

While revocation by software updates may work well for Chrome, it can be quite difficult on a number of platforms such as smartphones, where issuing a software update involves multiple entities and can take many months until wide deployment. Even now several smartphone vendors have not yet issued software updates to block DigiNotar certificates. Revocation by software update takes control of revocation out of the hands of the CAs where it naturally belongs and puts it in the hands of software and hardware vendors who may have less of an incentive to issue a software update every time a certificate is revoked.

An Old Proposal Revisited

In this paper we propose to abandon the existing revocation mechanisms in favor of an old idea [7], [8] — short-lived certificates — that puts revocation control back in the hands of the CAs. Short-lived certificates are far more efficient than CRLs and OCSP, and require no client-side changes, although a minor client change can strengthen the proposal significantly.

A short-lived certificate is identical to a regular certificate, except that the validity period is a short span of time such as a few days. Such certificates expire shortly, and most importantly, fail closed after expiration on clients without the need for a revocation mechanism. We suggest a certificate lifetime as short as four days, matching the average length of time that an OCSP response is cached [9].

In our proposal, when a web site purchases a year-long certificate, the CA's response is a URL that can be used to download on-demand short-lived certs. The URL remains active for the year, but issues certs that are valid for only a few days. Every day a server-side element fetches a new certificate that is active for the next few days. If this fetch fails, the web site is not harmed since the certificate obtained the previous day is active for a few more days giving the administrator and the CA ample time to fix the problem. In this way the burden of validating certificates is taken off the critical path of HTTPS connection establishment, and instead is handled offline by the web site. We emphasize that short-lived certificates do not change the communication pattern between clients and web servers. Moreover, since clients typically fail closed when faced with an expired certificate, this approach is far more robust that the existing OCSP and CRL based approaches.

Although it is conceptually simple, many issues need to be addressed for this approach to work. First, we hope to use short-lived intermediate certificates, but this requires some additional steps at the CA. Second, we need to ensure that installing a new certificate on a web server does not force a web server restart. Third, for web sites with many servers, we need an architecture to ensure that only one request from the site is issued to the CA per day (as opposed to one request per server). Finally, a small optional change to the client can provide additional defense-in-depth against attacks on the CA. We discuss these issues in the following sections.

We analyze the security of this proposal in Section V, but briefly mention here that when a web site key is compromised by a server-side breach, the certificate can be used for a few days, but becomes worthless once it expires. In real-life this is equivalent or better than the security guarantees offered by OCSP — OCSP responses are usually valid for a few days [9] and therefore revocation via OCSP also need a few days to take affect.

II. Background

We begin by surveying the existing standards for removing trust from a valid certificate before its expiration date, and discuss the deficiencies that have caused Chrome to abandon them.

A. Certificate Revocation Lists

One solution to dealing with certificates that go bad is the certificate revocation list (CRL) [1]. When a certificate goes bad, its identifying serial number is published to a list, signed and timestamped by a certificate authority (CA). In order to trust a certificate, a user must not only verify the signature and expiration date, but also ensure that the certificate is not listed in CRLs.

For CRLs to be effective, one assumes that (1) up-to-date lists are published frequently by the CA, (2) the most recent list is available to the verifier, and (3) verification failures are treated as fatal errors. These constraints on CRLs degrade their effectiveness as a revocation mechanism:

- An earlier study [10] on real-world CRLs indicated that more than 30% of revocations occur within the first two days after certificates are issued. For CAs, there is a tradeoff between their CRL publishing frequency and operational costs. For CAs that update CRL with longer intervals, there is a risk of not blocking recently revoked certificates in time.
- Since CRLs themselves can grow to be megabytes in size, clients often employ caching strategies, otherwise large transfers will be incurred every time a CRL is downloaded. This introduces cache consistency issues where a client uses an out-of-date CRL to determine revocation status.
- Browsers have historically been forgiving to revocation failures (a.k.a "fail open") so as not to prevent access to popular web sites in case their CAs are unreachable [11]. In practice, they either ignore CRL by default, or do not show clear indications when revocation fails [12]. Unfortunately, this lets a network attacker defeat revocation by simply corrupting revocation requests between the user and the CA.
- It should also be noted that the location of the CRL (indicated by the CRL distribution point extension) is a non-critical component of a certificate description, according to RFC5280. This means that for certificates without this extension, it is up to to the verifier to determine the CRL distribution point itself. If it cannot CRLs may be ignored [13].

B. Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP), an alternative to CRLs proposed in RFC 2560 [2], allows client software to obtain current information about a certificate's validity on a certificate-by-certificate basis. When verifying a certificate, a client sends an OCSP request to an OCSP responder, which responds whether the certificate is valid or not. Typically, clients are instructed to cache the OCSP response for a few days [9]. OCSP responders themselves are updated by CAs as to the status of certificates they handle.

In theory, it is possible for CAs to issue OCSP responses with short validity periods (since the response size is smaller than a CRL), however there are many real-world constrains that make this approach infeasible:

- OCSP validation increases client side latency because verifying a certificate is a blocking operation, requiring a round trip to the OCSP responder to retrieve the revocation status (if no valid response found in cache). A previous study [9] indicates that 91.7% of OCSP lookups are costly, taking more than 100ms to complete, thereby delaying HTTPS session setup.
- OCSP may provide real-time responses to revocation queries, however it is unclear whether the responses actually contains updated revocation information. Some OCSP responders may rely on cached CRLs on their backend. It was observed that DigiNotar's OCSP responder was returning good responses well after they were attacked [14].
- Similar to CRLs, there are also multiple ways that an OCSP validation can be defeated, including traffic

filtering or forging a bogus response by a network attacker [15]. Most importantly, revocation checks in browsers fail open. When they cannot verify a certificate through OCSP, most browser do not alert the user or change their UI, some do not even check the revocation status at all [12]. We note that failing open is necessary since there are legitimate situations in which the browser cannot reach the OCSP responder. For example, at an airport, a traveler might be asked to pay for Internet service before connecting to the Internet. In this case, the browser cannot validate the gateway's certificate using OCSP and must implicitly trust the provided certificate so that the user can enter her payment information and connect to the Internet.

- OCSP also introduces a privacy risk: OCSP responders know which certificates are being verified by end users and thereby responders can, in principle, track which sites the user is visiting. OCSP stapling is intended to mitigate this privacy risk, but is not often used. We discuss stapling in Section VI.

III. Design

In what follows we assume a *server* provides a particular service such as a web-based email over HTTPS. A *client* is a user of the service, which will validate whether the server provided certificate is signed by a trusted source to determine the authenticity of the server. Both the client and server trust an intermediate party, the *certificate authority*, typically pre-installed on the client's web browser or operating system. We describe the modifications we make on the three components in this scenario: the certificate authority, the server, and the client.

a) Certificate Authority: The role of a certificate authority is to sign the certificates for subscribing servers. The certificate authority has two modes of operations: *on-demand* and *pre-signed*. What differentiates the two is how the certificates are generated.

- *On-demand mode.* When using the on-demand approach, the CA keeps its private key online, and signs new certificates when requested. In on-demand mode, the online CA keeps a template certificate — a certificate with static information, such as the common name and public-key already populated — which is loaded when the web server requests a new short-lived certificate. The validity period of the certificate is altered such that it begins at the time of the request and ends the configured amount of time in the future, typically within a few days. The certificate is signed using the CA's private key and sent back to the web site.
  In on-demand mode the hardware boxes used at the CA to manage the CA's private key can be configured so that they will never issue a certificate with a validity period of more than a few days beyond the present date. Consequently, a single compromise of the CA will only expose certificates that will expire shortly.
- *Pre-signed mode.* With the pre-signed approach, the CA's private key is kept offline and certificates are signed in batches. When a server requests a certificate, the CA looks through its store of pre-signed certificates to find the appropriate one, in which the validity period begins before the request time and ends at least a day after the request time. The extra overlap allows the requester to not have to worry about automatically re-issuing the request were it to be issued closer to its expiration date. Similar to previous two-level digital signature schemes [16] (using an offline CA to pre-issue certificates), this reduces the computation of the online CAs. It also allows that CA's private key to remain offline, as is often the case for root CAs.

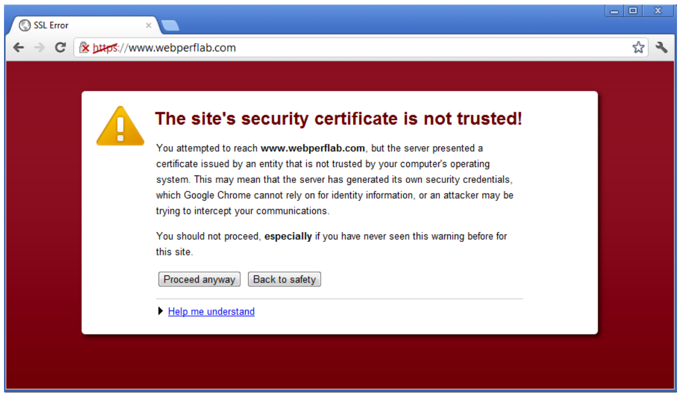
In either case, should the server request its certificate be revoked, the certificates, either the template or pre-signed, are removed from the CA store. Subsequent requests will fail.

b) Server Plug-in: The short-lived certificates themselves are usable by any server which supports standard X.509 certificates. What is required is a side-loaded program (or plug-in) that manages the server's certificates by requesting fresh certificates from the certificate authority as the expiration time of the current certificate approaches. Our server-side program is set-up to automatically to execute after a certain interval of time. It is recommended that the interval is set to at least two days before the expiration date to ensure new certificates are installed in a timely fashion.

When the server certificate-downloading program wakes up, it checks the expiration date of the current installed certificates and if any are approaching expiration, the program issues an HTTP GET request to the CA for a new certificate. The server-side program checks that the only field that changed in the new certificate is the validity period (and in particular, the public-key did not change). If so, it stores the new certificate in the certificate store and alerts the server to load the new certificate. In Section IV we explain how to load a new certificate in popular web server without restarting the server. If the retrieved certificate is corrupt, it is ignored and the site admin is alerted.

c) Client-Side Pinning: In current popular browsers certificate validation fails for expired certificates. Therefore, no client-side modifications are needed to implement our short-lived proposal. Chrome's CRL approach complements this mechanism well in case there is a need to quickly revoke a compromised certificate (including root and intermediate certificates).

In practice, when encountering expired certificates, the browser blocks the page, removes the lock icon from the address bar, and presents a security warning dialog to the user (see Figure 1). We note that previous studies have shown that



The site's security certificate is not trusted!

You attempted to reach **www.webperflab.com**, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Google Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.

You should not proceed, **especially** if you have never seen this warning before for this site.

Fig. 1. **Expired certificate warning in Google Chrome** *When encountering expired certificates, the browser blocks the page, removes the lock icon from the address bar, and presents a security warning dialog to the user.*

users may still click through the warnings [17] and therefore a strict hard-fail approach is suggested for better security, as implemented for HSTS [18] websites.

While no client-side are required, the short-lived proposal can strengthened with a small client-side extension. In particular, we propose to add a new X509 certificate extension to indicate that the certificate is a short-lived certificate. When a client sees such a short-lived certificate, it records this fact and blocks future connections to the server if the server presents a non-short-lived certificate. We call this *client-side pinning* for short-lived certificates, similar in a way to existing certificate authority pinning in browsers [19], [20]. In addition to just pinning the CA used by a site, we are pinning the status of whether a site uses short-lived certificates. Client-side pinning ensures that the number of short-lived enabled certificates (including intermediate certificates) in a certificate chain should never decrease. This optional behavior should still allow short-lived certificates to be incrementally adopted on intermediate CAs.

Client-side pinning prevents two attacks. First, suppose an attacker succeeds in compromising a CA *once* without being detected. Without short-lived pinning on the client, the attacker could simply request long-lived certs from the CA's hardware and these certs let the attacker man-in-the-middle all web sites that use this CA. With short-lived pinning, the attacker must request short-lived certs from the CA's hardware, but by design the hardware will only issue short-lived certs that expire in a few days. Therefore, a one-time compromise of the CA will not help the attacker. The attacker must repeatedly compromise the CA thereby increasing the chance of detection and revocation.

Second, consider a web site that currently uses long-lived certs. If the server's secret key is stolen the site may ask the CA to revoke the long-lived cert and then switch to using short-lived certs. But an attacker can block revocation messages sent to clients and then use the stolen long-lived cert to man-in-the-middle the web site. Clients would have no knowledge that revocation took place and will accept the revoked long-lived cert. With short-lived pinning, if the client connects to the legitimate site after it switched to short-lived certs, the long-lived cert will no longer be accepted by the client.

If a website wishes to stop using short-lived certificates, the X509 extension can provide an option to disable client-side pinning.

IV. Implementation

We developed a prototype to enable and automatically update short-lived certificates for Apache web servers. We also implemented client-side pinning in Chromium web browser.

a) Certificate Authority: Our certificate authority was implemented using Java and is served over Apache Tomcat as a web application. The web server issues an HTTP GET request to the CA server specifying the common name for the certificate it wishes to retrieve, as well as a unique certificate identifier. This identifier allows a web server to have multiple certificates under the same common name stored by its one CA. These identifiers are chosen by the owners of the servers when they register with the CA for short-lived certificates. They allow a server to have multiple certificates under a common name, say if they wish to use a different private/public key pair or want a certificate that is a wild card certificate and one that is domain specific.

In either the pre-signed or on-demand mode, the CA's servlet looks for an appropriate certificate on the filesystem. In on-demand mode, the validity period of the matching template certificate is updated and signed with the CA's private key. The private key is stored encrypted on the CA's server, and is decrypted and brought into memory at start-up. The signing and general certificate handling is done using the IAIK cryptography libraries [21]. The pre-signed certificates are signed offline using a different key and are transferred to the servers manually. The batch can be set by the CA but will present a trade-off between security and ease-of-use. Pre-signing larger batches means less overhead of signing and transferring the certificates, but leaves more signed certificates on an online server and thus at the risk of being stolen. Each pre-signed and and on-demand certificate is made valid for four days to match the length of time for which an OCSP response is cached [9].

b) Server Plug-in: We implemented our server-side program in Java targeting Apache web servers. The program is set as a cron job executing every day. When the program runs, it checks to make sure the certificate is close to expiration. If true, it issues a GET request to our CA for either a pre-signed or on-demand certificate. Once the certificate is obtained it is stored to the filesystem in the standard PEM format.

Our Apache SSL configuration files are set such that the file locations of the certificates are symbolic links. When the new certificate is stored on the filesystem, all our program has to

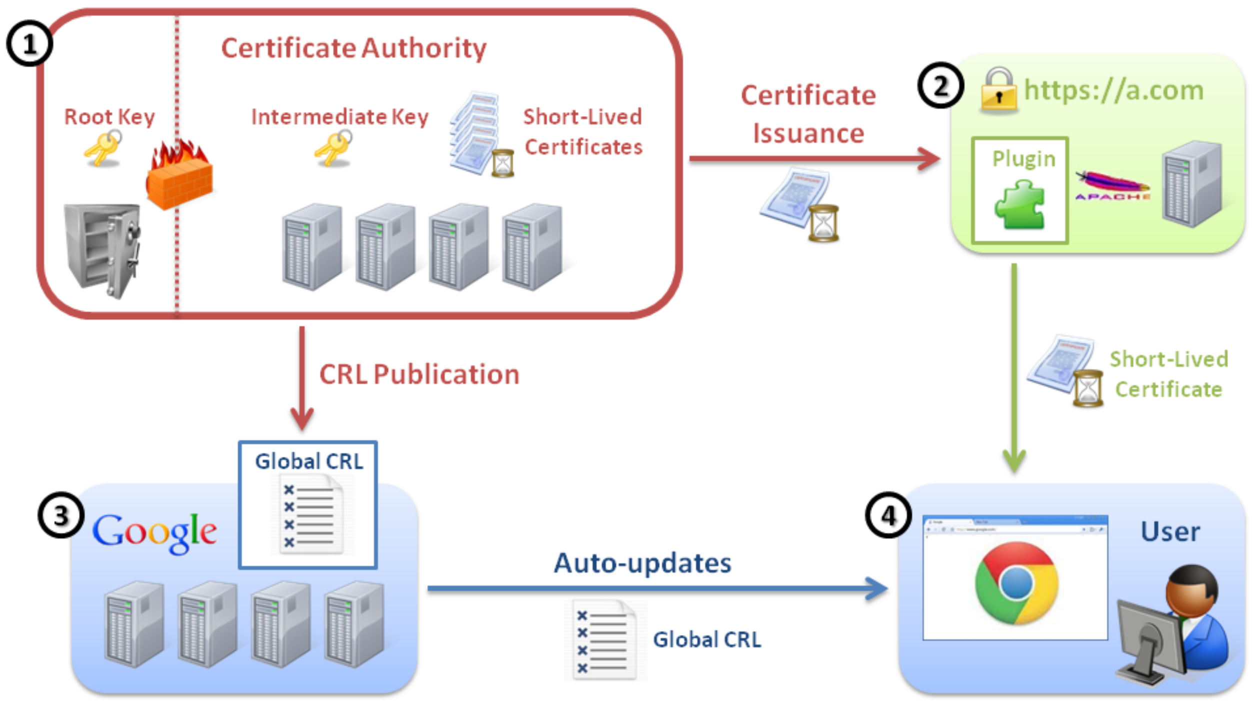


Fig. 2. **Short-Lived Certificates and Chrome's CRL** *1. Certificate authorities pre-sign batches of short-lived certificates. 2. A plugin on the SSL website periodically fetches fresh short-lived certificates from the CA and loads it onto the server. 3. Google generates a list of revoked certificates, and automatically updates the global CRL to browsers. 4. When the user visits the SSL website, the certificate is validated against Chrome's CRL and also fails on expiration.*

do is re-point the symbolic link to the new certificate and optionally clean up old, expired ones. After this, the server certificate-downloading program issues a graceful restart command to Apache. This ensures the web server restarts and loads the new certificates without disrupting any existing connections. Although we prototyped on the Apache web server, our proposal is applicable to other popular web servers such as Nginx and Jetty, without causing server downtime. We confirmed that Nginx [22] supports graceful restart, similar to Apache, and Jetty supports reloading the Java trust store on-the-fly [23]. [1]

c) Client-Side Pinning: We implemented a prototype of client-side pinning for short-lived certificate in Chromium (using revision 61348). Since Chromium utilizes the system cryptography libraries on each platform [24] to handle certificate verification, we implemented our code as a platform-independent function in the browser above the cryptography libraries, instead of modifying a specific library such as NSS. We reused the existing transport security state code in Chromium (for HSTS and public key pinning) to store our short-lived certificate pins persistently. Our patch for Chromium is available online [25].

[1]It is reasonable for a large site using Apache to use short-lived certificates. Apache can restart gracefully with no noticeable impact to end users as our benchmarking has shown.

V. Analysis

In Section II, we described the ineffectiveness of the existing soft-fail OCSP and CRL mechanisms, which have paved the way for recent proposals such as Chrome's browser-based CRLs. In this section, we discuss the benefits and shortcomings of various revocation approaches in a post-OCSP world (summarized in Table I), including (1) Chrome's CRL, (2) hard-fail OCSP, (3) short-lived certificates, and (4) short-lived certificates with Chrome's CRL.

A. *Chrome's CRL*

As mentioned in Section I, Google has announced plans to disable OCSP checks completely. Instead, Chrome will reuse its existing software update mechanism to maintain a list of revoked certificates on its clients.

1) Advantages: In the case of certificate misissuances during CA incidents, Google could push out new CRLs that will block the fraudulently issued certificates on the clients in less than a daily time frame. Due to using software updates, this approach even has the ability to remove the misbehaving root certificates. Browser-based CRLs are updated periodically and not fetched at the time connecting a site, thus there is no additional latency during page load, nor privacy concerns of tracking the user's browsing history. Further, network filtering attacks become more difficult, an attacker would have to consistently block software updates from the time of revocation, instead of only at the time of visit.

TABLE I
COMPARISON OF CERTIFICATE REVOCATION APPROACHES

	Chrome's CRL	Hard-Fail OCSP	Short-Lived Certificates	Chrome's CRL + Short-Lived Certificates
Untrust Rogue Root CAs	✓	✗	✗	✓
Revoke Misissued Certificates	✓	✓	✓	✓
Revoke Benign Certificates	✗	✓	✓	✓
Support Captive Portals	✓	✗	✓	✓
Avoid Page Load Delay	✓	✗	✓	✓
Avoid User Tracking	✓	✗	✓	✓
Avoid Single Point of Failure	✗	✗	✗	✓
Support Legacy Clients	✗	✗	✓	✗

2) Disadvantages: Due to space considerations of maintaining a global CRL, Google will not support a vast amount of revocations that are due to administrative reasons. Should OCSP be disabled, certificate authorities lose control over revocation, therefore unable to revoke certificates for billing, re-issuance, or other benign reasons. Google has become a single point of failure for certificate revocation.

Another major disadvantage of this approach is that legacy clients are currently not supported, such as mobile devices and other browsers. Google may want to provide their global CRLs as a public service for other browsers and applications, in a way similar to their Safe Browsing API [26].

B. Hard-Fail OCSP

In attempt to fix the ineffectiveness of OCSP under existing CA infrastructures, some security researchers as well as CAs have suggested clients enforce hard-fail OCSP. Some browsers do allow users to opt-in to hard-fail for revocation checks, but this must be turned on by default to be effective.

1) Advantages: Unarguably, the security of a hard-fail OCSP is better than existing soft-fail mechanisms. Unlike new browser-based CRL proposals, this approach supports the existing revocation infrastructure managed by CAs.

2) Disadvantages: Unfortunately, there are legitimate reasons why browsers refuse to simply turn on hard-fail OCSP. For example, users may need to log in to captive portals of WiFi hotspots where the login page is protected by SSL. Often, the HTTP traffic is blocked, including OCSP, and will cause certificate validation to timeout. If hard-fail OCSP was implemented, users will not be able to connect to many WiFi hotspots, even though they are probably not under attack.

Also, this approach shares many disadvantages of existing OCSP approach, including the ability for third party to track the user's browsing history. Further, clients may suffer significant connection latency due to OCSP queries, which even worse may discourage sites on adopting SSL. Whenever an OCSP responder goes down, all sites that use their certificates will go down as well.

In the case where a Root CA has been completely compromised, OCSP does not provide any protection since the attacker could have easily generated a valid OCSP response with a far expiration date. Clients would need software updates to untrust the bad root certificate.

C. Short-Lived Certificates

We described the approach of using short-lived certificates in Section III. In short, to revoke a certificate, the CA simply stops reissuing new certificates, as any old certificates either must have expired, or would expire shortly.

1) Advantages: By default, certificate expiration is always strictly validated on clients. All major browsers check for the validity period of certificates and present alerts to users when encountering expired certificates. No modifications on clients are required, thus will work on all platforms and devices without updates.

In the event of a key compromise or fraudulent issuance of site certificates, with short-lived certificates, the CA simply stops renewing the certificate for revocation. The window of vulnerability will be bounded by the short validity period, which should be no more than a few days. We note that short-lived certificates can potentially help if supposedly a large CA (e.g. VeriSign) is hacked — the *too big to fail* problem. In that case, there will be a need to revoke the stolen certificates. Short-lived certificates can make that easier.

Unlike OCSP, user's browsing habits are not reported to any third parties when short-lived certificates are used. Further, this approach does not require additional round-trips to a SSL connection setup, as browsers do not need to verify the certificate status with an OCSP responder.

Since short-lived certificates are regular certificates, they can be chained in exactly the same manner as the existing deployment of X.509 certificates. This allows websites, and even intermediate certificate authorities, to incrementally deploy and take advantage of short-lived certificates, without a large migration, or infrastructure change.

2) Disadvantages: Although short-lived certificates cannot solve the problem of a root CA compromise (neither does hard-fail OCSP), it does improve the security of intermediate certificates that are short-lived. In on-demand mode, the fact that the certificate authorities are online increases the risk of the CA's key being stolen and fraud certificates being generated. This is less of an issue with the pre-signed mode where certificates are signed by an offline CA in batches, allowing the key to be kept safe and isolated from the online servers. However, signing in batches implies that a CA break-in will provide the attacker with a larger pool of pre-signed certificates and thus a longer time during which they can masquerade as the certificate owners. Fortunately, this only hinders security if the attackers have also compromised a client's private key.

One possible issue that could be raised is the fact that if a CA falls under DDoS attacks, servers can not get updated certificates and are thus forced into service outage as soon as their certificates expire. This requires that the attacker is able to take down the CA consistently for at least a few days. Fortunately, there are many mitigations. One approach for CAs is to filter traffic based on IP, only allowing well-known customer IPs through. If using pre-signed mode, the distribution of certificates can further be handled by third-party distributors that specialize in handling DDoS-level traffic.

We note that deploying short-lived certificates might cause errors on clients whose clocks are badly out of sync (e.g. off by a week). It is recommended that clients should sync their clocks periodically, which is critical for preventing replay attacks.

D. Short-Lived Certificates with Chrome's CRL

Lastly, we consider a hybrid approach of using short-lived certificates in cooperation with Chrome's CRL. First of all, by issuing short-lived certificates, CAs immediately regain control of certificate revocation that was disabled by Chrome. Once CAs start to issue short-lived certificates for sites (as well as intermediate CAs), these sites will benefit from the improved security. In the event of keys being stolen or certificates being misissued, short-lived certificates ensures a shorter window of vulnerability by warnings on expiration.

In addition, Chrome's CRL complements nicely with short-lived certificates. With short-lived certificates alone, we mentioned that users may still be vulnerable in the worst case, when a root certificate is deemed untrustworthy. Now that browser-based CRLs provide protection against CA incidents, the combination of these two approaches allows a full spectrum of revocation (on supporting browsers). Browser vendors will be able to revoke fraudulent certificates and rogue CAs, while CAs may control administrative revocations for benign certificates.

This hybrid approach does not have the client side performance or privacy issues caused by OCSP, and does not block users behind captive portals. In contrast, we note that using hard-fail OCSP along with Chrome's CRL would still suffer the compatibility issues with captive portals, as well as page load delay and privacy issues. Given Chrome's CRL in place, we suggest that adopting short-lived certificates gives the maximum security without the obvious shortcomings of OCSP.

VI. Related Work

A. OCSP Stapling

An update to OCSP is OCSP stapling, where the web server itself requests OCSP validation which it passes on the response to inquiring clients. Stapling removes the latency involved with OCSP validation because the client does not need an additional round trip to communicate with the OCSP responder to check the certificate's validity. This also removes the privacy concern of OCSP because the responder does not have access to knowledge about a web site's visitors. Unfortunately this is not widely implemented, only 3% of servers support OCSP stapling [27]. Also, current implementations do not support stapling multiple OCSP responses for the chained intermediate certificates [11]. Further, we note that users are still prone to attacks if clients do not implement hard-fail OCSP stapling (and pin the status of whether a site uses OCSP stapling). In contrast, short-lived certificates essentially fail closed on existing clients without modifications.

B. DANE

A recent proposal called DNS-based Authentication of Named Entities (DANE) [28] allows the domain operator to sign certificates for sites on its domain, without trusting external CAs. By distributing certificates via DNS records [29], certificates can be cached on DNS servers, reducing the connection round trip time while protecting the client's privacy. Most importantly, this approach avoids the danger of an untrustworthy CA being able to issue valid certificates for any domain (given that typically hundreds of CAs are pre-installed on clients). There are other DNS-related proposals that still involve trust in CA, such as publishing OCSP responses over DNS, or providing an exclusion list of certificates in DNS [30], reducing the dependence on the availability of OCSP responders. However, these approaches rely on DNSSEC [31] to prevent forgery and modification, which has not been deployed broadly. Another concern is that the trusted domain operator (like a CA) may be compelled by a state actor to issue false certificates.

C. Public Key Pinning

Google Chrome implements public key pinning [19] for a preloaded list of HTTPS sites. Basically, those HTTPS sites must include a whitelisted, or *pinned*, public key in their certificate chain, otherwise it will be treated as fatal error. Public key pinning address attacks where a trusted CA is compromised (or compelled by a state actor [32]), whenever the client sees a valid certificate that was fraudulently issued by a rogue CA (not within the preloaded whitelist). This approach may be more suitable for a small number of large, high security sites, but less practical for the majority HTTPS sites on the web.

D. Google Certificate Catalog

The Google Certificate Catalog [33] is a database of all the valid SSL certificates collected by Google's web crawlers. By querying Google's DNS, clients may check whether the hash of a certificate matches what is seen by Google, and present a warning to the user upon mismatches. This approach helps clients to detect CA fraud incidents, for a much larger portion of the web (in comparison to the public key pinning approach described in Section VI-C). However, there is possibility that not all legitimate certificates are indexed in Google's database, and may cause false positives. Further, DNSSEC deployment is required to prevent a network attacker from modifying DNS queries. Finally, some valid web pages are crawled less frequently than on a 90 day basis, resulting in a large vulnerability window.

E. Convergence

Convergence [34] is an alternative to the traditional CA system, allowing clients to use a configurable set of external notaries to validate its communications with web sites (based on the Perspectives Project [35]) instead of trusting the pre-installed list of root CAs on their systems. This approach uses multiple trust notaries, avoiding the single point failure problem when one trusted CA is compromised. Convergence requires modifications to existing clients, currently implemented as a Firefox add-on. Unfortunately, convergence as currently implemented has performance costs for requiring multiple network round trips to verify a certificate [36]. Privacy may become an issue if the a notary is compromised, although users may proactively reconfigure their client.

VII. Conclusion

Short-lived certificates complement Chrome's CRL by adding security when software updates are difficult, as in the case of mobile platforms. They also put revocation back in the hands of CAs and remove Google as a single point of failure. In practice, short-lived certificates provide security guarantees that are equivalent to or better than OCSP. They are fully backwards compatible and support increment deployment. Moreover, since OCSP is no longer needed at HTTPS session setup time, short-lived certs reduce the time needed to setup an HTTPS session. Overall, short-lived certs add another layer of protection to web users beyond Chrome's CRL.

Acknowledgments

We are grateful to Rick Andrews, Adam Langley and Charlie Reis for reviewing and providing feedback on this work. This work was supported by NSF and AFOSR.

References

[1] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5280.txt

[2] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," RFC 2560 (Proposed Standard), Internet Engineering Task Force, Jun. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2560.txt

[3] Comodo, "Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011," http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html, 2011.

[4] VASCO, "DigiNotar reports security incident," http://www.vasco.com/company/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx, 2011.

[5] P. Pachal, "Chrome 15 beats Internet Explorer 8 as worlds most popular browser," http://mashable.com/2011/12/15/chrome-leapfrogs-ie8/.

[6] A. Langley, "Revocation checking and Chrome's CRL," http://www.imperialviolet.org/2012/02/05/crlsets.html, 2012.

[7] R. Rivest, "Can we eliminate certificate revocation lists?" in *Financial Cryptography*, 1998.

[8] A. Herzberg and H. Yochai, "Minipay: charging per click on the web," in *Selected papers from the sixth international conference on World Wide Web*, 1997.

[9] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh, "The case for prefetching and prevalidating TLS server certificates," in *Proceedings of the 19th Network and Distributed System Security Symposium*, 2012.

[10] C. Ma, N. Hu, and Y. Li, "On the release of CRLs in public key infrastructure," in *Proceedings of the 15th USENIX Security Symposium*, 2006.

[11] A. Langley, "Revocation doesn't work," http://www.imperialviolet.org/2011/03/18/revocation.html, 2011.

[12] P. Kehrer, "Defective By Design? - Certificate Revocation Behavior In Modern Browsers," http://blog.spiderlabs.com/2011/04/certificate-revocation-behavior-in-modern-browsers.html, 2011.

[13] E. Turkal, "Securing Certificate Revocation List Infrastructures," 2001, http://www.sans.org/reading_room/whitepapers/vpns/securing-certificate-revocation-list-infrastructures_748.

[14] K. McArthur, https://twitter.com/#!/KevinSMcArthur/status/110810801446727681, 2011.

[15] M. Marlinspike, "New Techniques for Defeating SSL/TLS," Black Hat DC 2009.

[16] M. Naor and K. Nissim, "Certificate revocation and certificate update," in *Proceedings of the 7th USENIX Security Symposium*, 1998.

[17] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness," in *Proceedings of the 18th USENIX Security Symposium*, 2009.

[18] J. Hodges, C. Jackson, and A. Barth, "HTTP Strict Transport Security (HSTS)," 2012, http://tools.ietf.org/html/draft-hodges-strict-transport-sec.

[19] A. Langley, "Public key pinning," http://www.imperialviolet.org/2011/05/04/pinning.html, 2011.

[20] M. Marlinspike, "Your App shouldn't suffer SSL's problems," http://blog.thoughtcrime.org/authenticity-is-broken-in-ssl-but-your-app-ha, 2011.

[21] IAIK, "CRYPTO Toolkit," http://jce.iaik.tugraz.at/.

[22] Nginx, "Starting, Stopping, and Restarting Nginx," http://wiki.nginx.org/CommandLine#Stopping_or_Restarting_Nginx.

[23] J. Calcote, “Managing a Dynamic Java Trust Store,” http://jcalcote.wordpress.com/2010/06/22/managing-a-dynamic-java-trust-store/, 2010.

[24] Chromium, “SSL Stack,” http://www.chromium.org/developers/design-documents/network-stack/ssl-stack, 2010.

[25] “Short-Lived Certificate Patch for Chromium r61348,” http://pastebin.com/E6viYsUx, 2012.

[26] Google, “Safe Browsing API,” https://developers.google.com/safe-browsing/.

[27] Y. N. Pettersen, “New CNNIC EV Root, pubsuffix update, and some blacklisted certificates,” http://my.opera.com/rootstore/blog/2011/03/31/new-cnnic-ev-root-pubsuffix-update-and-some-blacklisted-certificates, 2011.

[28] P. Hoffman and J. Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Protocol for Transport Layer Security (TLS),” 2012, http://tools.ietf.org/html/draft-ietf-dane-protocol-17.

[29] S. Josefsson, “Storing Certificates in the Domain Name System (DNS),” RFC 4398 (Proposed Standard), Internet Engineering Task Force, Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4398.txt

[30] A. Langley, “DNSSEC and TLS,” http://www.imperialviolet.org/2010/08/16/dnssectls.html, 2010.

[31] D. Eastlake 3rd, “Domain Name System Security Extensions,” RFC 2535 (Proposed Standard), Internet Engineering Task Force, Mar. 1999, obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845. [Online]. Available: http://www.ietf.org/rfc/rfc2535.txt

[32] C. Soghoian and S. Stamm, “Certified lies: Detecting and defeating government interception attacks against ssl (short paper),” in *Financial Cryptography*, 2011.

[33] B. Laurie, “Improving SSL certificate security,” http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html, 2011.

[34] M. Marlinspike, “Convergence,” http://convergence.io/.

[35] D. Wendlandt, D. Andersen, and A. Perrig, “Perspectives: Improving SSH-style host authentication with multi-path probing,” in *Proceedings of USENIX Annual Technical Conference*, Boston, MA, Jun. 2008.

[36] M. Marlinspike, “SSL And The Future Of Authenticity,” Black Hat USA 2011.