A shorter version of this paper appeared in *Proc. 17th Financial Cryptography and Data Security Conference*, April 2013. For updates: jhalderm.com/papers/.

CAge: Taming Certificate Authorities by Inferring Restricted Scopes

James Kasten, Eric Wustrow, and J. Alex Halderman

The University of Michigan
{jdkasten,ewust,jhalderm}@eecs.umich.edu

Abstract. The existing HTTPS public-key infrastructure (PKI) uses a coarse-grained trust model: either a certificate authority (CA) is trusted by browsers to vouch for the identity of *any* domain or it is not trusted at all. More than a thousand root and intermediate CAs can currently sign certificates for any domain and be trusted by popular browsers. This violates the principle of least privilege and creates an excessively large attack surface, as highlighted by recent CA compromises. In this paper, we present CAge, a mechanism that browser makers can apply to drastically reduce the excessive trust placed in CAs without fundamentally altering the CA ecosystem or breaking existing practice. CAge works by imposing restrictions on the set of top-level domains (TLDs) for which each CA is trusted to sign. Our key observation, based on an Internet-wide survey of TLS certificates, is that CAs commonly sign for only a handful of TLDs; in fact, 90% of CAs have signed certificates for domains in fewer than ten TLDs, and only 35% have ever signed a certificate for a domain in `.com`. We show that it is possible to algorithmically *infer* reasonable restrictions on CAs' trusted scopes based on this behavior, and we present evidence that browser-enforced inferred scopes would be a durable and effective way to reduce the attack surface of the HTTPS PKI. We find that simple inference rules can reduce the attack surface by nearly a factor of ten without hindering 99% of CA signing activity over a six-month period.

1 Introduction

Every day, millions of Internet users rely on HTTPS to secure transactions such as online banking, e-mail, and e-commerce against malicious eavesdroppers or tampering through man-in-the-middle attacks. HTTPS combines the Transport Layer Security (TLS) protocol with a public-key infrastructure (PKI) based on certificate authorities (CAs) that are trusted by the browser. Each server presents its public key in the form of an X.509 certificate corresponding to its domain name and digitally signed by a CA, which is responsible for verifying the identity of the website, usually for a small fee. Browsers maintain a set of trusted root CAs and subsequently trust the purported identities of certificates signed by any CA in this trusted set. In addition, these root CAs are typically able to sign certificates for additional CAs, known as intermediate certificate authorities, which are trusted recursively by the browser.

CA-signed certificates help protect users in the presence of man-in-the-middle adversaries, but they cannot protect users from compromise of the CAs themselves. In recent years, there have been several high-profile attacks on CAs that resulted in the signing of fraudulent certificates. For instance, in 2011, an attacker breached the security of a relatively small Dutch CA named DigiNotar and created certificates for dozens of popular sites, including `*.google.com` [4]. An ISP in Iran subsequently abused this latter certificate to conduct man-in-the-middle attacks against Google services. Of course, attackers do not require a malicious or compromised ISP to utilize illegitimate certificates; they can also use DNS cache poisoning attacks [11], intercept and modify traffic on an insecure wireless network, or use ARP spoofing on a local subnet.

Preventing DigiNotar-style attacks is difficult, because there are currently very few technical restrictions on what trusted CAs can sign for—once they convince a browser they are trustworthy, they are given an unrestricted capability to vouch for any domain name they choose. This ability leads to an enormous attack surface: an attacker who compromises any one of over 1200 CAs can then impersonate any website that relies on HTTPS. The status quo violates the principle of least privilege: DigiNotar should not have had the capability to sign certificates for Google, nor should a CA run by a small university in the United States be allowed to sign certificates for another country's intelligence agency, such as a website in the `.gov.ir` domain. In other words, each CA's trust should come with a limited scope.

One way to limit the scope of CA trust is to designate a set of top-level domains (TLDs), such as `.com` or `.uk`, within which each CA may sign. Indeed, we present data that suggests that most CAs currently only sign certificates for sites in a small number of TLDs, and conversely, that sites in most TLDs utilize only a small set of CAs. Many CAs appear to sign exclusively for domains belonging to a single organization, and others appear to operate within a specific country, sector, or both. Although this suggests that TLD-based restrictions could be fruitful, realizing them within the existing PKI is a challenge. The X.509 name constraints extension (see Section 3) introduced the ability to explicitly declare such restrictions in new CA certificates, but it requires participation from each root or intermediate CA, as well as implementation in all client systems, and has seen almost no adoption.

Rather than relying on each CA to explicitly declare a TLD scope, we explore the possibility that browser makers could *infer* such scopes without CA participation. We propose a mechanism called CAge that creates a profile of each CA based on the TLDs of publicly visible certificates it has previously signed. If a browser later encounters a certificate for a site in a TLD a CA has never been observed to sign before, this can be treated as evidence of suspicious behavior. These restrictions can be implemented without cooperation from the certificate authorities, at the risk that CAs will change their behavior over time and begin signing for certificates outside their previous pattern. Empirically, however, we find that this rate of change is quite low, that inferred scopes generated with simple algorithmic rules would result in a low false-positive rate, and that the

CAge approach would allow browser makers to dramatically reduce the attack surface of the HTTPS PKI.

Outline We begin in Section 3 with a discussion of related work. In Section 4, we analyze data from an Internet-wide survey of HTTPS certificates to examine the current practices and distribution of CA's signing. Supported by evidence from this dataset, in Section 5, we give a detailed description of the CAge approach for inferring TLD-based restrictions. In Section 6, we quantitatively evaluate CAge's performance, and in Section 7 we describe a prototype implementation in the form of a browser extension. Finally, we conclude in Section 8.

2 Background

In this section, we present a brief background of X.509 and its extensions that are relevant for this paper. For a more in-depth background on the TLS public key infrastructure, we recommend RFC 5280 [5].

When a browser connects over HTTPS, the server presents an X.509 certificate. The certificate includes an identity (such as the domain name of the server), a validity period, a public key, and a digital signature over the rest of the certificate. The browser checks that the certificate's identity matches the domain the browser is attempting to connect to, that the certificate is still valid, and that the digital signature of the certificate is correct. The public key is then used to authenticate or transmit a shared secret with the server, which is, in turn, used to establish an end-to-end secure channel.

The certificate's digital signature is signed by a trusted Certificate Authority (CA) which is in charge of verifying the identity of the website. When the server presents its certificate, it also includes the certificate of the CA responsible for signing. The signing-CA's certificate may also in turn have been signed by another CA, in which case, this second CA is also included in the bundle of certificates sent to the browser. This bundle of certificates is referred to as a *certificate chain*, and the browser only trusts it if one of the CAs in the chain is in a trusted set of root CAs that are built into the browser.

3 Related Work

Given the documented problems with CAs [4,15,19,20], it is not surprising that providing authentication on the Internet is an active research topic. Many researchers have suggested using new authentication techniques or making changes to the authentication infrastructure. Multi-path probing [3,14,18] has been suggested as a way to move away from certificate authorities. The technique necessitates the availability and access to trusted notaries, which presents problems for captive portals and sites presenting multiple certificates. The suggested changes to the CA infrastructure focus on making certificate signing transparent and publicly verifiable [6,12]. The work needs widespread support with implementation and testing before the effects of these systems can be seen. CAge instead works

with existing authentication mechanisms and does not require collaboration with CAs or additional infrastructure.

Quick local improvements on the security of the current CA system have also been proposed and adopted in limited form. The idea of certificate pinning, where the browser remembers which certificates belong to each domain, has been used to various degrees by existing software. Google Chrome uses certificate pinning for Google's own websites [8], while CertPatrol [13], a Firefox extension, allows the user to pin all certificates that they encounter and warns the user when a certificate changes. However, CertPatrol's fine granularity of control, frequent false positives, and required knowledge deters all but the most dedicated users. Soghoian and Stamm proposed CertLock [16], which pins the country code of the CA to the domain. The design is aimed at preventing compelled certificate creation attacks in which a government forces a CA to issue false certificates. The technique can work fairly well for U.S. domains being attacked by foreign governments; however, due to the dominance of the U.S. in the CA market, the proposal's true effectiveness is uncertain. Our solution, CAge, pins information about the CA rather than the specific domain and does not demand additional knowledge from the user. CAge does not protect against a specific attack, but rather aims to decrease the attack surface of the PKI in general.

Scopes on certificate authorities have also been proposed through X.509 Name Constraints. X.509 Name Constraints [5] is a certificate extension with the ability to restrict CAs to a particular set of domains. All trusted certificates must conform to the name constraint extensions in their certificate chain. Implementation and adoption of the accepted standard has been slow. There is only one trusted CA that is currently constrained with the extension, the "Hellenic Academic and Research Institutions RootCA 2011", which was adopted in Firefox 11.0 [2]. Name constraints' reluctant adoption is likely due to the high cost of enforcement. Since browsers only have direct control over their root CAs, browsers would have to force tight constraints directly on the root CAs. Complicating matters, root CA certificates have an average lifetime of more than 20 years. Long certificate lifetimes require these constraints to be forecast far into the future, which is both difficult and prone to error. Wide-scale adoption of name constraints also requires all existing certificates to be reissued under constrained CAs, requiring a long-term transitional plan. CAge is able to leverage knowledge of the complete CA system and is able to differentiate the constraints between roots and specialty intermediaries. Root CAs are tightly constrained to their intended purpose and intermediaries are known and constrained to their previously demonstrated behavior. CAge can be applied immediately by browsers without CA collaboration, making immediate adoption feasible.

4 Analyzing the CA Infrastructure

Currently, important aspects of browser trust behavior and CA signing practices are surprisingly opaque to outside observers. For example, certificate chaining allows browser-trusted root CAs to delegate their signing authority to third

parties by issuing intermediate CA certificates; this can be done in private and at any time, making it impossible to determine the full trusted set of CAs by analyzing browser software alone. Furthermore, CAs typically do not publish the set of domains for which they have issued certificates, making it difficult to study their patterns of signing behavior in practice.

To understand these aspects of the HTTPS PKI, we analyzed a large corpus of certificates collected with an Internet-wide scan of HTTPS servers recently conducted in a study by Heninger et al. [9] The study exhaustively probed the IPv4 address space on TCP port 443 (the default port for HTTPS) and collected every certificate chain presented by a responding server. The resulting data, from October 2011, provides a recent and comprehensive view of the TLS certificate landscape. It includes responses from 7.7 million hosts, which returned more than 5.8 million distinct certificates.

First, we determined which of the certificates would be trusted by the major web browsers and extracted the set of intermediate CAs that issued them from the provided certificate chains. We started with the 317 root and intermediate CAs[1] that are directly trusted by Mozilla, Apple, Microsoft, and OpenSSL (Google Chrome defaults to the platform's trusted key store). We then used the OpenSSL API to test each of the collected certificate chains for validity. We deemed a certificate valid if it was not expired at the time of the scan and there existed a chain of valid CA certificates rooted by a directly trusted CA certificate. In all, there were 1,956,267 valid certificates (comprising 2,558,492 unique domain names) issued by 1207 CAs.

The number of certificates signed by each CA varied considerably. The top 20 CAs were responsible for more than 80% of valid certificates. Figure 1 shows the cumulative distribution of signed domains within the top five TLDs and the number of CAs responsible for signing each fraction. Over 90% of all signed `.com` domain names used certificates issued by just 25 certificate authorities.

Despite this lop-sided distribution of CA size, each of the 1207 CAs had the ability to issue trusted certificates for any domain name. To examine how much of this authority each CA exercised, we extracted the set of domain names that each CA had directly issued certificates for, and then examined the set of TLDs to which these domains belonged. We found that 89% of CAs had signed for domains in fewer than 10 unique valid TLDs [10], with the majority (65.8%) of CAs signing for domains in either zero or one TLD.[2]

Figure 2 shows the top 25 TLDs by number of signed domains, together with the fraction of total signed domains that are in each TLD and the number of CAs that issued certificates within each TLD. Although `.com` accounts for 51% of signed domains, fewer than 35% of trusted CAs had signed a certificate for even a single `.com` domain, and only 20% had signed for 10 or more such certificates. There were 787 CAs that had *never* signed for a `.com` domain. Similarly, fewer

[1] Unless otherwise noted, we distinguish CAs by subject and subject key identifier.

[2] A CA can sign for zero valid TLDs if it does not sign domain certificates directly but instead signs other intermediate CA certificates, for example.

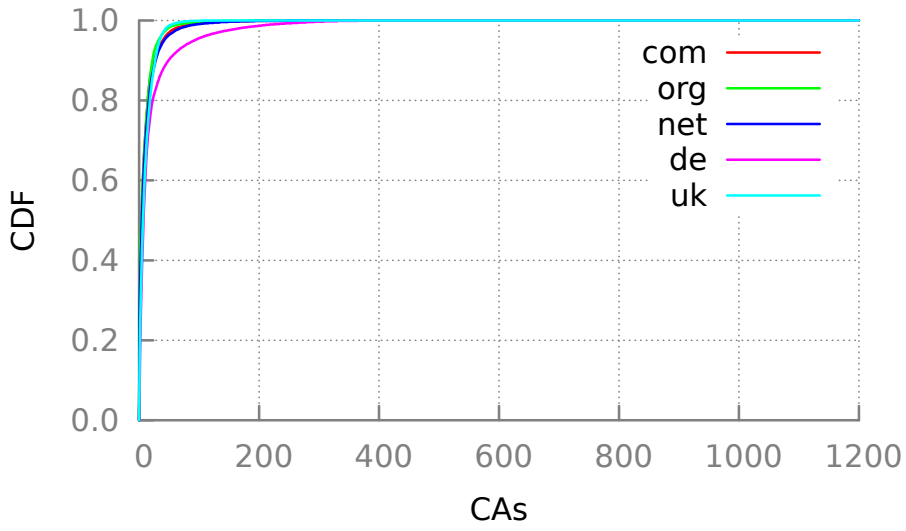

Fig. 1: **CAs Signing for Top 5 TLDs** — This figure shows the cumulative distribution of signed domains within the top five TLDs and how many trusted CAs accounted for each fraction. A small number of large CAs dominate.

than 11% of CAs had signed certificates in the `.uk` TLD, and only 6.6% had signed for 10 or more in the `.uk` domain.

To better understand why most CAs seem to be issuing certificates within so few TLDs, we manually investigated many of the trusted CAs. One reason for these sparse signing practices is that many of the CAs belong to private companies and organizations and are used for domains under their control. For instance, more than 200 German universities and research institutions are browser-trusted CAs; their impact on statistics for the `.de` TLD is clearly visible in Figures 1 and 2. Many corporations including Ford, Disney, Wells Fargo, and Migros also have trusted CA certificates. We observed that they generally limit their signing practices to a few specific second-level domains. For instance, the Walt Disney CA signs almost exclusively for domains under `*.disney.com`. However, we note that some of these CAs may sign for additional private domains that do not show up in our data set of publicly accessible HTTPS servers.

Another reason appears to be that many smaller CAs focus their business within a specific geographic region and tend to sign domains under a country-specific TLD. (We noted that 29% of CAs signed for domains within only a single TLD and not in `.com`.) For example, the public AusCERT CA signs 97% of their certificates under the `.au` and `.nz` TLDs, and the Coop Swiss company signs

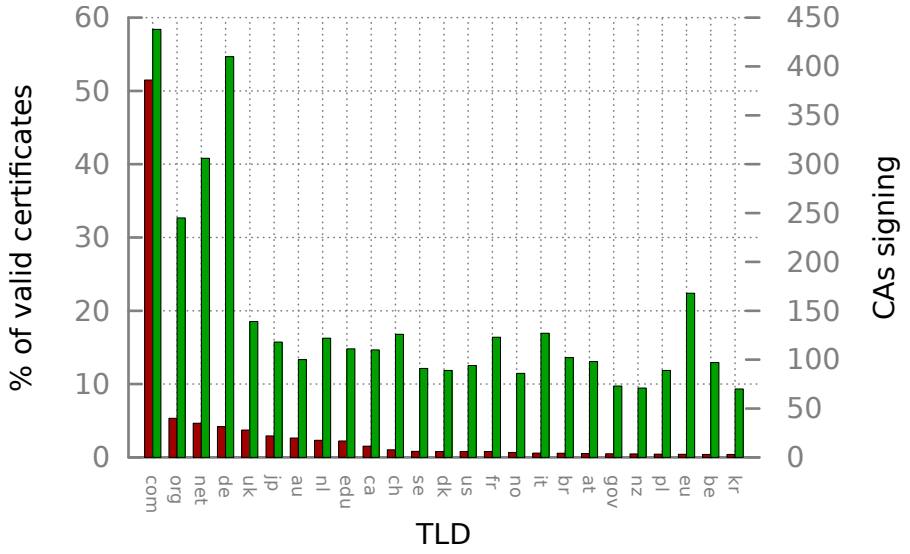

Fig. 2: **Top 25 TLDs for Signed Domains** — This graph shows both the fraction of signed domains falling within each of the top 25 TLDs (*red*) and the number of CAs that sign for at least one of these domains (*green*). Although `.com` is by far the most common TLD, 65% of CAs have never signed a certificate for a `.com` domain. Less common TLDs include signed domains from even smaller fractions of CAs, suggesting that the ability to sign for *all* domains is unused by the vast majority of CAs.

exclusively within `.ch`. The infamously compromised CA, DigiNotar, specialized in the Dutch market and issued 93% of its certificates to `.nl` domains.[3]

5 Our Proposal

Our analysis in the previous section suggests that the vast majority of CAs do not need or use the authority they have—to issue a trusted certificate for any domain in any TLD. Leaving these CAs with such unconstrained signing power leaves Internet security unnecessarily vulnerable: an adversary can choose the weakest of over 1200 CAs to attack in order to gain complete signing authority for any domain. In this section, we propose CAge, a browser-based approach that restricts CA signing to TLDs in which they have *already* signed. We argue that this would improve the ecosystem security of the HTTPS PKI without impairing how it is used today.

[3] DigiNotar was already defunct at the time of our scan, so this figure is based on December 2010 data from the EFF SSL Observatory [7].

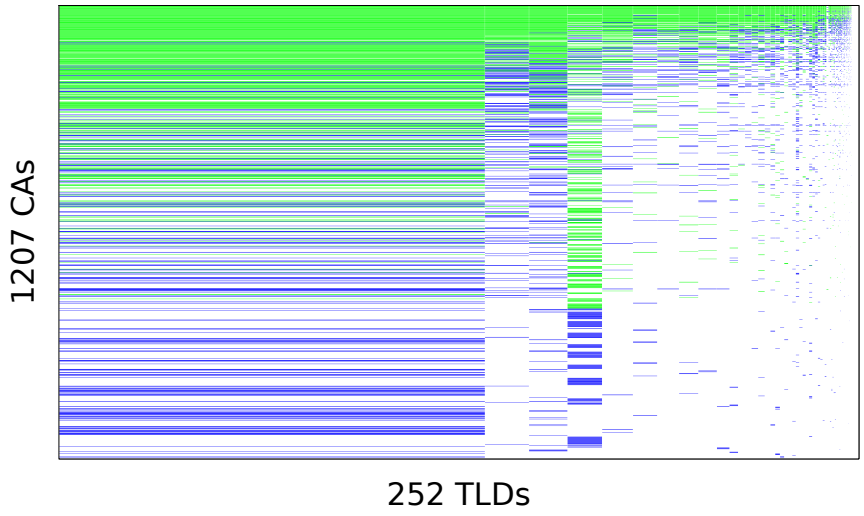

Fig. 3: **CA/TLD Matrix** — This figure shows a matrix of CAs that have signed for certificates in TLDs. Each position is colored if the corresponding CA (row) has signed for at least one (blue) or ten (green) domains in the corresponding TLD (column). Each axis is sorted by total domains signed, putting the most prolific CA at the top and most common TLD at the left. The width of the TLD columns are scaled by the percentage of certificates made up in that column—.com is visible as the left-most column, as it accounts for over 50% of the total domains signed. A significant portion of this matrix is empty, illustrating the sparse nature of CA signing practices.

CAge consists of two phases, an initialization phase and an enforcement phase. In the initialization phase, we collect certificates from an Internet-wide scan and infer rules from the observed current CA signing practices. Browsers then deploy CAge as a browser extension in the enforcement phase to restrict CAs to these inferred scopes and handle exceptions. We describe these phases in more detail below.

5.1 Initialization and Rule Inference

Prior to deploying CAge, the browser maker needs to develop an initial set of restricted scopes to apply to existing CAs. However, creating justifiable rules for existing CAs necessitates knowledge of the current CA market. Given the distributed design of the CA system, a comprehensive scan of HTTPS must be performed like the one completed by Heninger et al. [9]. Such a scan can be used to determine the observable list of intermediate certificate authorities, as well as the domains for which they have directly signed certificates.

After scanning and collecting the raw data, we infer rules and restrictions for the CAs, based on current practices. As stated earlier, there are many CAs that have never signed for particular top-level-domains. If the user was presented such a certificate, there is a high probability that the certificate is fraudulent and the user should be alerted. As a first approach, CAge can generate the inferred scopes by looking at the TLDs for which each CA has previously signed. If a CA has signed for any domains in a particular TLD, the inferred rules will allow the CA to continue to sign for that TLD, which will be referred to as the TLD policy.

Rules are stored for each CA as a set of regular expressions that governs the domains they are allowed to sign. For example, a CA may have the regular expressions `.*\.au` and `.*\.nz`, allowing it to sign for two TLDs. This allows for the rule inference to be extendable to other algorithms in the future. In general, rules should be generated from an algorithm taking the CAs and their signed domains as input and producing the CA restrictions as output. CAs could be constrained to second-level domains or more specific rules could be required for larger TLDs, factoring in the cost of false positives, and both the size and brittleness of the rule set. We explain variations on our generation algorithm in Section 6.1.

5.2 Enforcement and Exception Handling

Once CAge has inferred CA signing rules from the collected scans, CAge relies on browsers to enforce these rules during certificate validation. Browsers have a strong incentive to protect their users from fraudulent certificates, making them a natural place to enforce these restrictions.

Normally, browsers verify that HTTPS certificates have a valid signature chain to a trusted root. With CAge, browsers additionally compare the domain to the set of regular expression rules inferred for that certificate's intermediate (signing) CA. If the domain does not fall within the allowed rules for the given CA, CAge alerts the user with a warning, explaining that the website's origin is certified by an unusual source. CAge also asks the user if they want to send the violation to the browser maker for further inspection. This feedback allows the browser to potentially verify the authenticity of the certificate via other means, while respecting the user's privacy. The browser may use techniques such as multi-path probing [18] to aid the user in authenticating the domain.

5.3 Updating

Keeping the rule set accurate and current is crucial to keeping a low false positive rate and avoiding user habituation to clicking through warning messages. The CAge rule set must be updated as CA policies change and new CAs emerge. Luckily, browser makers are in a good position to provide updates to users, based on newly discovered certificates reported collectively by users. Updates to the CA signing rules can be pushed to users through standard browser update mechanisms.

Any update mechanism based on inferred rules runs the risk of being gamed by attackers; for example, if an attacker compromises a CA that is not currently allowed to sign for a domain in the `.com` TLD, the attacker can request that the CA sign a legitimate `.com` domain, owned by the attacker. Under a naive rule set update approach, inferred TLD rules will soon change to allow this CA to sign for the `.com` TLD, and the attacker can use the previously compromised CA to sign for other `.com` domains fraudulently. This means the browser maker cannot simply look to long-standing valid certificates in order to validate new TLD rules once an attacker knows of the system. Although the browser maker could query certificate authorities about the legitimacy of signings and changes to rules, attackers can still increase the scope of all publicly-signing intermediate certificate authorities. A CA might not turn away business for a domain simply because they have not signed for the top-level domain in the past. While CAge would still protect users from illegitimate certificates signed by certificate authorities that do not sign publicly (including private organizations, root CAs and inactive intermediates), inferred TLD updates would severely increase the attack surface.

For this reason, the CAge rule set is updated on a per domain basis. When a domain exception is reported, the domain is added to a "watchlist" and is manually verified before the specific certificate's domain is whitelisted and pushed as an update. We show in Section 6.2 that these updates are infrequent and thus allow manual inspection and verification.

New CAs, without any recorded behavior, may specify their intended scope to the browser maker in advance to avoid the whitelist update mechanism for their domains. While this might be seen as an additional hurdle for new CAs entering the market, ultimately, the authority to say whether or not a particular CA is trusted lies with the browser.

6 Evaluation

In this section, we quantitatively evaluate CAge's performance in experiments based on real world data.

6.1 Attack Surface Reduction

CAge reduces the overall attack surface by restricting the scope of certificate authorities, but a metric is required to evaluate the effectiveness of CAge quantitatively and to compare various rule inference algorithms. Our goal is to quantify the relative risk of damage that could be caused by an attacker-compromised CA. Treating each signed valid domain as a protected entity, we approximate the attack surface by:

$$\sum_{c \in CAs} domains[c]$$

where $domains[c]$ is the number of domains that a given CA c can sign. Currently, all CAs can sign for all domains; therefore, $domains[c]$ is equal to the number of

signed valid domains for all c.[4] Under the CAge policy that restricts CA scopes by TLD, domains[c] is the total number of valid domains across all the TLDs that c is permitted to sign. For example, if a CA is allowed to sign for only .com because they signed for 100 of the 1.3 million .com domains in the dataset, $domains[c]$ for that CA would be 1.3 million.

While this attack surface metric is by no means complete, it does provide a simple and intuitive first-order approximation that allows us to compare the relative risks of different CA restriction policies quantitatively.

As mentioned in Section 5.1, a very simple approach would be to only allow a CA to sign for top-level domain names the CA had previously signed, the TLD policy. Applied to our data set, the TLD policy results in a 75% decrease in attack surface than current practice.

We can improve this result by modifying the inference procedure to only allow a CA to sign for domains in a TLD if it has previously signed for more than a threshold t of unique domains in that TLD. Domains signed by CAs that do not meet the policy cut-off can either be viewed as suspicious anomalies or manually inspected, labeled as an exception, and whitelisted within the database. Figure 3 provides a visualization of this attack surface for two policies. When requiring the CA to sign for 25 domains before allowing authorization over the complete TLD ($t = 25$), the attack surface is reduced to 11.1% of the original. Figure 4 shows the attack surface of the TLD policy as it becomes more strict.

CAge may also be evaluated based on its ability to stop recent CA compromises. In the Comodo attack that occurred in March 2011 [15], an attacker issued fraudulent certificates for `.com` domains signed by "CN=UTN-USERFirst-Hardware", a relatively large CA which had signed over 25,000 other `.com` certificates previously. Due to these signing practices, CAge would have been unable to protect against the Comodo attack. Similarly, all but two of the top 20 CA certificates have signed domains from over 100 unique TLDs, limiting the usefulness of restricting these large CAs to the TLDs they currently sign.

However, the vast majority of CAs do not sign for such a diverse market allowing CAge to provide protection during a CA compromise. For instance, CAge would have detected the DigiNotar compromise. The EFF's SSL Observatory [7] data, which was collected a year before the attack, shows that the issuer of the fraudulent `*.google.com` certificate, "DigiNotar Public CA 2025" [1], had not signed certificates for any `.com` domains. Had CAge been implemented at the time, it would have prevented the attacker from using the `*.google.com` against Internet users in Iran.

6.2 CAge Durability

Although the attack surface metric provides a quantifiable goal, it should not be the only factor when considering the inferred rule set. The minimum attack surface can be attained by pinning every domain to the CA that signed their

[4] There were not any trusted CAs that contained name constraints in the November 2011 scan.

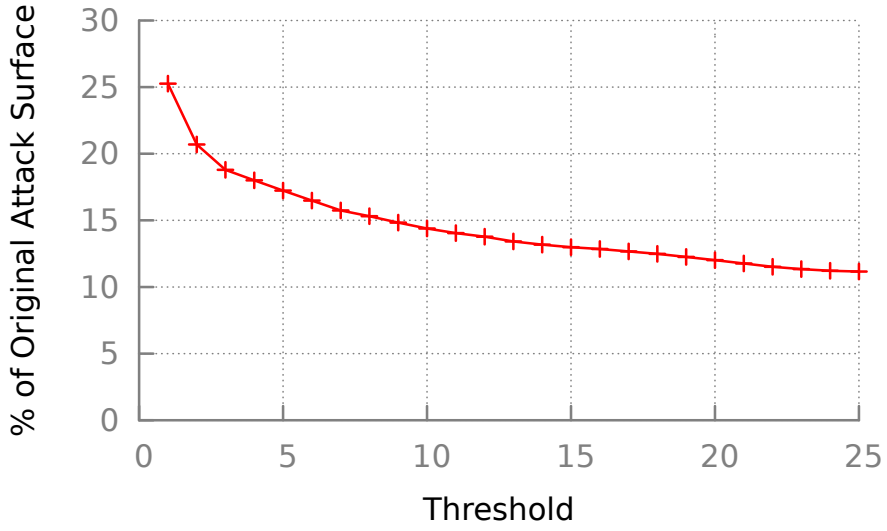

Fig. 4: **Attack Surface Metric** — HTTPS PKI Attack Surface under the basic TLD rule inference policy. Threshold refers to the number of domains signed before the TLD is added to the CA's scope. Even restricting CAs to the TLDs they have previously signed reduces the attack surface to 25% of the status quo.

certificate in the data set. This policy would provide the minimum attack surface, but the constraints would be very brittle and CAge would require a very large rule set that needed to be updated frequently. In addition, excessive false positives are not tolerable in any system, as users learn to ignore warnings [17]. Simply finding the policy with the minimum attack surface is not enough; CAge should attempt to capture the CA's de facto policies to avoid brittleness of the rule set.

In order to test the durability of our inferred rules, we acquired a second data set in April 2012 (6 months after the original scan) to test the viability of our solution. We found that the large majority of domains observed in newly issued certificates conformed to our rules, supporting our hypothesis that the TLDs that CAs sign for are generally stable. The basic TLD policy with $t = 1$ accommodated 99.84% of new certificates. Table 1 summarizes new certificates for domains in the top 10 TLDs. Our results show that only a handful of CAs signed for TLDs they had not previously signed. While several smaller and less stable TLDs had certificates with domains that violated our previous rules, all of the violations could be fixed by simply whitelisting the 1506 domains in the browser's rule set. CAge required an average of fewer than 10 domain updates per day over the 6 month period. Using a more restrictive TLD policy with $t = 25$ results in 10,035 violating domains (98.93% conformed).

TLD	Violating CAs	Violating Domains	Total Issued Domains	% Violated
com	10	27	519174	0.005
org	12	22	50531	0.044
net	14	28	46300	0.060
de	8	30	34160	0.088
uk	5	8	33113	0.024
jp	3	4	30699	0.013
au	3	3	21768	0.014
edu	2	3	20498	0.015
nl	6	20	19076	0.105
ca	5	6	16716	0.036
Total	**152**	**1506**	**937137**	**0.161**

Table 1: **TLD rule violations**— For the top 10 TLDs, we evaluated the certificates seen in April 2012 that contained domains violating inferred rules generated from data collected in October 2011 using a TLD policy with $t = 1$. Violating CAs represents the number of CAs that were not previously seen signing for this TLD in October 2011, but were observed signing for it in April 2012. Violating Domains represents the number of unique domains issued by Violating CAs in that TLD, while total issued domains represents the number of domains observed in new certificates seen in April. As shown by the percentage violation (domains / total issued domains), the vast majority of new certificates conform to the generated rules.

7 Implementation

We have developed a CAge prototype in the form of a Firefox extension. We implemented the simple lenient policy, described in the previous section, in which certificate authorities can sign for any TLD they have previously signed. Better policies could be obtained, but this simple policy is unlikely to produce many false positives, and achieves good theoretical results. The CAge extension downloads the current set of rules and known CAs from our CAge server over HTTPS upon installation. CA certificates are identified by their SHA1 fingerprint and rules are stored as regular expressions within the extension. When the browser is presented a TLS certificate, the browser forwards it to the CAge extension where it is matched against the current rules database. In addition to the standard rules table enabling the issuer to sign certificates, the CAge extension also contains tables for globally blacklisted CAs, locally ignored hosts, local rule sets, a local certificate white list, and a session certificate white list. These tables allow for the customization and protection against known threats to which users have grown accustomed.

If the extension cannot find an appropriate rule for the incoming certificate, it stops the connection and sends a request for updates to the CAge server over

HTTPS[5]. The request contains a time stamp of the last extension update and synchronizes with the server if it is out of date. If the extension is still unable to resolve the certificate, it prompts the user, asking if they would like to query the CAge server with the domain name for more information.

The CAge prototype appears to be non-intrusive in our daily use. It has been used for normal browsing for four months and we have observed zero false positives while using the TLD policy with a threshold of one.

8 Conclusion

In this paper, we presented CAge, a mechanism for inferring TLD-based restricted scopes for HTTPS CAs. Based on the empirical observation that the vast majority of browser-trusted CAs do not utilize their unconstrained signing power, we argue that each CA should be restricted to signing for domains within a limited set of TLDs. We show how such restrictions can be realized in practice by profiling past CA signing behavior, and we find that such an approach would dramatically reduce the attack surface of the HTTPS PKI with a low rate of false positives over time.

While browsers have a positive record of revoking compromised CA certificates once a breach is discovered, we believe much more can be done to proactively mitigate the damage caused by attacks against CAs and to provide defense-in-depth to the HTTPS PKI. Given the relative ease with which CAge could be deployed by browsers, we strongly encourage browser developers to adopt this approach to help combat the growing threats that HTTPS users face.

Acknowledgements

The authors gratefully acknowledge Zakir Durumeric for providing HTTPS certificate data for this study, and we thank the anonymous reviewers for their constructive comments and feedback. This work was supported in part by the National Science Foundation (NSF) under contract numbers CNS 1255153 and DGE 0654014 and an NSF Graduate Research Fellowship.

References

1. Gmail.com SSL MITM Attack by Iranian government, Aug. 2011. http://pastebin.com/ff7Yg663.
2. Bug 581901 – Add HARICA root certificate. Website, Apr. 2012. https://bugzilla.mozilla.org/show_bug.cgi?id=581901.
3. Alicherry, M., and Keromytis, A. D. Doublecheck: Multi-path verification against man-in-the-middle attacks. In *ISCC* (2009), IEEE, pp. 557–563.
4. Bhat, S. Gmail users in Iran hit by MITM Attacks. Website, Aug. 2011. http://techie-buzz.com/tech-news/gmail-iran-hit-mitm.html.

[5] The CAge extension uses a pinned certificate to avoid MITM attacks using fraudulent certificates

5. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.
6. Eckersley, P. Sovereign key cryptography for Internet domains. Website, Nov. 2011. https://www.eff.org/sovereign-keys.
7. EFF. The EFF SSL Observatory. https://www.eff.org/observatory.
8. Evans, C. New Chromium security features, June 2011. Website. http://blog.chromium.org/2011/06/new-chromium-security-features-june.html.
9. Heninger, N., Durumeric, Z., Wustrow, E., and Halderman, J. A. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX conference on Security symposium* (Berkeley, CA, USA, 2012), Security'12, USENIX Association, pp. 35–35.
10. IANA. Top level domains. http://data.iana.org/TLD/tlds-alpha-by-domain.txt.
11. Kaminsky, D. Black Ops 2008: It's the end of the cache as we know it, Aug. 2008. BlackHat USA.
12. Laurie, B., Langley, A., and Kasper, E. Certificate transparency. Website, Sept. 2012. http://tools.ietf.org/html/draft-laurie-pki-sunlight-00.
13. Loesch, C. Certificate patrol. Website. http://patrol.psyced.org/.
14. Marlinspike, M. SSL and the future of authenticity, Aug. 2011. BlackHat USA.
15. Richmond, R. Comodo fraud incident, Mar. 2011. http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.
16. Soghoian, C., and Stamm, S. Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In *Proc. 15th Intl. Conf. on Financial Cryptography and Data Security* (2012), FC'11, pp. 250–259.
17. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and Cranor, L. F. Crying wolf: An empirical study of SSL warning effectiveness. In *Proceedings of the 18th conference on USENIX security symposium* (Berkeley, CA, USA, 2009), SSYM'09, USENIX Association, pp. 399–416.
18. Wendlandt, D., Andersen, D. G., and Perrig, A. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX 2008 Annual Technical Conference* (Berkeley, CA, USA, 2008), USENIX Association, pp. 321–334.
19. Zusman, M. Breaking the security myths of extended validation SSL certificates. In *Black Hat USA 2009* (2009). http://www.blackhat.com/presentations/bh-usa-09/ZUSMAN/BHUSA09-Zusman-AttackExtSSL-SLIDES.pdf.
20. Zusman, M. Criminal charges are not pursued: Hacking PKI, Aug. 2009. DefCon 17.