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 1200 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) under which each CA is trusted to sign certs. Our key observation, based on an Internet-wide survey of TLS certs, is that CAs commonly sign for sites in only a handful of TLDs. 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 activity over a 6 month period.

Keywords: TLS, HTTPS, Certificate Authorities, Authentication

1 Introduction

Every day, millions of Internet users rely on HTTPS to secure their online transactions against malicious eavesdroppers or tampering through man-in-the-middle attacks. HTTPS relies on a public-key infrastructure (PKI) based on certificate authorities (CAs) that are trusted by the browser. Each server presents an X.509 certificate tying its public key to its domain name. This certificate is digitally signed by a CA, which is responsible for verifying the site's identity.

CA-signed certificates cannot protect users from compromise of the CAs themselves. Several recent high-profile attacks on CAs resulted in the signing of fraudulent certificates. For instance, in 2011, an attacker breached the security of the Dutch CA DigiNotar and created certificates for dozens of popular sites, including `*.google.com` [3]. An ISP in Iran subsequently abused this latter certificate to conduct man-in-the-middle attacks against Google services.

Preventing DigiNotar-style attacks is difficult, because there are currently very few technical restrictions on what domains trusted CAs can sign for. Once the CA

convinces a browser (or another CA) that they are trustworthy, they are given an almost 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. This 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 be allowed to sign certificates for foreign government agencies. 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 2) introduced the ability to explicitly declare such restrictions in new CA certificates, but 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. These restrictions can be implemented without cooperation from the CAs, at the risk that CAs will change their behavior over time and begin signing for certificates outside their previous pattern. Empirically, 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.

For further details, see the full version of this paper, which is available online at https://jhalderm.com/papers/.

2 Related Work

There have been several prior proposals for addressing CA shortcomings [3,11,14]. Multi-path probing [2,10,13] has been suggested as a way to reduce reliance on CAs; however, it necessitates the availability and access to trusted notaries. Browser extensions have also been proposed to pin previously seen certificates or CAs to domains [6,9,12].

Scopes on CA signing have previously been proposed through X.509 Name Constraints [4], a certificate extension with the ability to restrict CAs to a particular set of domains. However, this approach has yet to see significant adoption due to several practical impediments, including lack of direct browser authority over intermediate CAs, the long lifetime of root CA certs, and the need to replace site certs when an issuing CA implements constraints. In contrast, CAge can be applied immediately by browsers without CA collaboration.

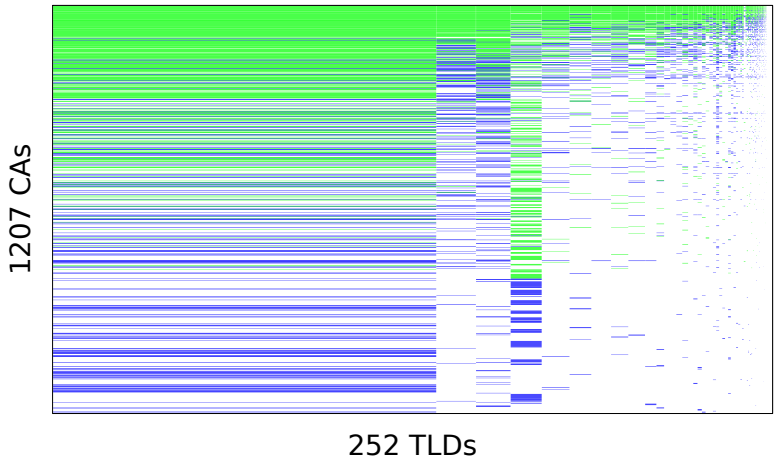

Fig. 1: This matrix shows which trusted CAs have signed certs for at least one domain (blue) or at least ten domains (green) in each TLD. Columns are scaled by fraction of valid certs (left is `.com`). Note sparseness of CA signing practices.

3 Analyzing the CA Infrastructure

Currently, important aspects of browser trust behavior and CA signing practices are surprisingly opaque to outside observers. Certificate chaining conceals the full set of trusted CAs by including an unknown number of intermediate authorities. Furthermore, CAs typically do not publish the domains for which they have issued certificates, obfuscating their signing patterns.

To understand these aspects of the HTTPS PKI, we analyzed a large corpus of certificates collected for another recent study [7] using an Internet-wide scan of HTTPS servers. We determined which certificates would be trusted by major web browsers and extracted the set of trusted CAs. See the full paper for details.

The number of certificates signed by each CA varied considerably; the top 20 CAs were responsible for more than 80% of valid certificates. Over 90% of all signed `.com` domain names used certificates issued by just 25 CAs.

Despite this lopsided distribution of CA size, 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 find that 89% of CAs had signed for domains in fewer than 10 unique valid TLDs [8], with the majority (65.8%) of CAs signing for domains in either zero or one 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 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.

Many CAs belong to private companies and organizations and are used for domains under their control. More than 200 German universities and research institutions control browser-trusted CAs, as do corporations such as Ford, Disney, and Wells Fargo. We observed that such CAs generally limit their public signing practices to a few specific second-level domains. Other smaller CAs appear to focus their business within a specific geographic region and tend to sign domains under a country-specific TLD.

4 Our Proposal

In this section, we propose CAge, a browser-based approach that restricts CA signing to TLDs in which they have *already* signed. CAge consists of two phases: In the initialization phase, we collect certificates from an Internet-wide scan and infer rules from the observed current CA signing practices. Browsers then apply these rules in the enforcement phase to restrict CAs to the inferred scopes and handle exceptions. See the full paper for more details and for a description of our browser extension prototype.

4.1 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 current CA practice. A comprehensive survey of public HTTPS servers (like that completed by Heninger et al. [7]) can be performed to determine the observable list of intermediate CAs and 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 a user is later presented such a certificate, this may indicate 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 that each CA has previously signed for. Under the simplest form of this approach, the inferred rules will allow a CA to sign for domains in a given TLD only if that CA has signed for a domain in that TLD before.

Rules are stored for each CA in the form of a set of regular expressions that governs the domains the CA is trusted to sign. This allows for the rule inference to be improved with more sophisticated algorithms in the future. In general, rule inference 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.

4.2 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 signed 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 developers for further inspection. This feedback allows the browser to potentially verify the authenticity of the certificate via other means, while respecting the privacy of its users.

4.3 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 CAge rules can also be pushed to users through browser update mechanisms.

The update mechanism must be carefully designed to avoid being gamed by attackers. For example, we might be tempted to regenerate the inferred rules based on any newly signed domain. However, in that case, an attacker who compromised a CA that was not allowed to sign for a domain in `.com` could simply purchase a certificate from that CA for a `.com` domain the attacker legitimately controlled. The inferred TLD rules would then update to allow this CA to sign for `.com`, and the attacker could use their compromise to sign for other `.com` domains fraudulently.

While CAge would still protect users from illegitimate certificates signed by CAs that do not sign publicly (including private organizations, root CAs and inactive intermediates), attackers can still try to increase the scope of all publicly signing intermediate CAs. For this reason, we propose that the CAge rule set should be updated on a per-domain basis. When a domain exception is reported to browsers, the domain should be added to a "watchlist" where the domain can be manually vetted before the specific certificate is whitelisted and pushed as an update. We show in Section 5.2 that these updates are infrequent and thus enable manual inspection and verification.

Over the long term, new CAs, without any recorded behavior, can be added by browsers after interrogating the CAs about their intended scope and policies. While this might pose an additional hurdle for new CAs entering the market, ultimately, the authority to say if a particular CA is trusted or not lies with the browser, and users' security interests demand a high level of scrutiny.

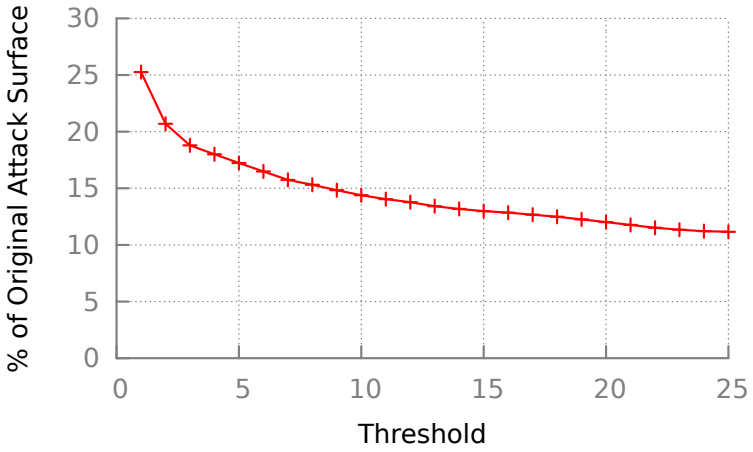

Fig. 2: HTTPS PKI attack surface under the basic CAge inference policy, compared to current practice. Even this simple approach achieves 75–90% reduction.

5 Evaluation

5.1 Attack Surface Reduction

While restricted scopes would reduce the attack surface across a large number of CAs, they are most effective against small and private CAs. In the Comodo attack that occurred in March 2011 [11], 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 [5] 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 attack against Internet users in Iran.

In light of these conflicting case studies, we attempt to quantify what kinds of attacks CAge would or would not protect against by developing an attack surface metric. The goal of this metric is to quantify the relative risk of damage that could be caused by an attacker-compromised CA. We compare this metric under different scenarios; namely, current CA practice (all CAs can sign for all domain names) versus CAge (CAs are restricted to a particular set of domains).

We approximate the attack surface by $\sum_{c \in CAs} domains[c]$. The function $domains[c]$ is the number of existing, validly signed domains for which a certificate signed by a given CA c would be trusted under a given set of policies. (Intuitively, the number of signed valid domains represents the number of protected entities on the Internet.) Under current practice, $domains[c]$ is constant across all CAs and is simply the number of signed valid domains in existence. Under CAge, $domains[c]$ is reduced to domains in TLDs that are allowed under the inferred trust scope for c. For example, if a CA is trusted to sign for only `.com` because it previously signed for 100 of the 1.3 million `.com` domains (and zero domains under other TLDs), then $domains[c]$ for that CA would be 1.3 million.

While this attack surface metric is by no means complete, it provides a first-order approximation that allows us to quantitatively compare the risks of different CA restriction policies. Applied to our data set, the simple CAge rule set inference method described in the previous section yields an attack surface that is 75% smaller 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 a minimum threshold t of unique domains in that TLD. If a CA has signed fewer than t domains in a particular TLD, these domains can either be viewed as suspicious anomalies or whitelisted as individual rules within the rule set. Applied to our scan data with $t = 25$, this policy would reduce the attack surface by 89% compared to current practice.

5.2 Rule Set Durability

Although the attack surface metric provides a quantifiable goal, reducing it is not our only objective. The minimum attack surface would be achieved by pinning every observed domain to the CA that signed its cert, but the result would be an enormous rule set that would require constant updating and lead to an impractical number of false positives. CAge must instead attempt to capture CAs' actual signing policies so as to produce rules that are compact and stable.

In order to test the durability of our inferred rules, we acquired a second scan in April 2012 (6 months after the original scan). Focusing on changes during this interval, 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 static. The basic policy, restricting CAs to TLDs they have signed in the past, accommodated 99.84% of new certificates. Most of the 1506 violations that occurred were in unpopular or small TLDs. See the full version of this paper for additional analysis.

6 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 restrictions can be realized in practice by profiling past CA behavior, and we find that such an approach would dramatically reduce the attack surface of the HTTPS PKI without a high rate of false alarms 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. Alicherry, M., and Keromytis, A. D. Doublecheck: Multi-path verification against man-in-the-middle attacks. In *ISCC* (2009), IEEE, pp. 557–563.
3. Bhat, S. Gmail users in Iran hit by MITM Attacks. Website, Aug. 2011. http://techie-buzz.com/tech-news/gmail-iran-hit-mitm.html.
4. 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.
5. EFF. The EFF SSL Observatory. https://www.eff.org/observatory.
6. Evans, C. New Chromium security features, June 2011. Website. http://blog.chromium.org/2011/06/new-chromium-security-features-june.html.
7. 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.
8. IANA. Top level domains. http://data.iana.org/TLD/tlds-alpha-by-domain.txt.
9. Loesch, C. Certificate patrol. Website. http://patrol.psyced.org/.
10. Marlinspike, M. SSL and the future of authenticity, Aug. 2011. BlackHat USA.
11. Richmond, R. Comodo fraud incident, Mar. 2011. http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.
12. 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.
13. 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.
14. Zusman, M. Criminal charges are not pursued: Hacking PKI, Aug. 2009. DefCon 17.