You Won't Be Needing These Any More: On Removing Unused Certificates From Trust Stores

Henning Perl[1], Sascha Fahl[1], and Matthew Smith[2]

[1] Leibniz University Hannover, Germany, {perl, fahl}@dcsec.uni-hannover.de
[2] University of Bonn, Germany, smith@l3s.de

Abstract. SSL and HTTPS is currently a hotly debated topic – particularly the weakest link property of the CA based system has been heavily criticized. This has become even more relevant in the light of recent spying revelations. While there are several proposals how the CA system could be improved or replaced, none of these solutions is receiving widespread adoption, and even in a best case scenario it would take years to replace the current system. In this paper we examine a root problem of the weakest-link property and propose a simple stop-gap measure which can improve the security of HTTPS immediately. Currently, over 400 trusted entities are contained in each of the common trust stores of various platforms and operating systems. To find out which of these trusted root certificates are actually needed for the HTTPS ecosystem, we analyzed the trust stores of Windows, Linux, MacOS, Firefox, iOS and Android, discuss the interesting differences and conduct an extensive analysis against a database of roughly 47 million certificates collected from HTTPS servers. We found that of the 426 trusted root certificates, only 66 % were used to sign HTTPS certificates. We discuss the benefits and risks involved in removing the other 34 % of trusted roots. On the whole, we argue that this removal is an important first step to improve HTTPS security.

1 Introduction

The TLS/SSL protocol is one of the mainstays of Internet security. However, unrest is growing as more large-scale compromises and real-world MITM attacks are discovered. This reflects the fact that the current certificate authority based public key infrastructure (CA-PKI) is a prominent example of a weakest-link security system: Since all trusted root CAs can issue certificates for any domain, an attacker can pick the weakest or most coercible CA to target for an attack – and a single vulnerable, malicious or coercible CA undermines the security of the entire system. To make matters worse, these attacks can go unnoticed quite easily. According to the EFF's SSL Observatory [1], current browsers trust roughly 1500 different CAs from roughly 650 different organizations.

Although the collection of trusted CA certificates, called *trust store*, can in theory be configured by the user, it is de facto the operating system and browser

vendors that issue the trust in the CAs. And while there is a broad consensus for a set of *common* CAs that are trusted by all common vendors, all vendors trust additional *uncommon* CAs that are not trusted by other vendors. Particularly in light of recent spying revelations, the inclusion of these uncommon CAs should be analyzed and if possible unneeded CAs should be removed.

This is a common-sense step which, surprisingly, is not actively being pursued by any of the companies responsible for the decisions on who we trust. There is a very small community of power-users who manually remove CAs they think they do not need and some tutorials on how this can be done, however, the decision on which CAs should be removed is based on anecdotal evidence and gut instinct.

As we will show in the course of this paper, a broad majority of HTTPS servers use only CA certificates which are in all major trust stores to sign their server certificate. This makes perfect sense: Only by using a CA trusted by all platforms can a server administrator be sure that no user receives warning messages. In contrast, an adversary may be fine with an attack working only under, e.g. Windows. Therefore, those uncommon CA certificates are still a security threat.

This is especially true since an attacker could identify the client's platform by analyzing the choice and order of supported cipher suites in the TLS handshake. If those match a vulnerable platform, a MITM attack is launched; otherwise the connection would be forwarded to the legitimate server. Such an attack could go undetected for a very long time. Additionally, a CA that is present only in a few trust stores may not be subject to as much rigorous auditing as a common CA.

In this paper we conduct a scientific analysis of which CAs are trusted on which platforms and correlate this data with 48 million certificates from Durumeric et al. that were collected by periodically scanning port 443 using ZMAP [2]. Based on this analysis, we identify 148 CA candidates that are never used to sign HTTPS server certificates. Following an in-depth analysis of these certificates, we create a list of CAs that can be removed from users' trust stores without hampering their everyday Internet activities while significantly reducing the attack surface against them. While this reduction of attack surface does not replace the need to find an improved certificate validation strategy, it is a very simple and extremely low cost measure which can be applied with minimal effort and should thus be considered as a first step to improve the security of SSL. We evaluate our reduced set of trust against two months' worth of traffic analysis in our university's network and show that there were no cases in which our proposed improvements would have caused any problems to ours users.

1.1 Outline

In Section 2 we highlight previous and parallel efforts to making SSL and the CA-PKI more secure. Section 3 describes our technical setup. In Section 4 we show which trust stores include which and how many certificates as well as how many certificates are present in every major trust store. Based on those findings,

we propose a set of 140 CA certificates that can be removed from trust stores in Section 5. Section 6 concludes the paper and outlines future work.

2 Related Work

There have been various approaches and attempts to improve the CA-PKI system. Perspectives [3] and Convergence [4] use network perspectives and multi-path probing to validate certificates and were suggested as a way to replace CAs completely. Both approaches need an additional network connection, which significantly impacts performance during connection establishment. Other approaches like Certificate Transparency [5], Sovereign Keys [6], or AKI [7] aim to control the PKI by keeping track of which CA issued which certificate. TACK [8] combines pinning with elegant key rollover. Finally, DNS DANE [9] focuses on putting certificates directly in the DNS record. While elegant, this requires the roll-out of DNSSEC, which also suffers from adoption problems [10]. All of these approaches fundamentally change the way validation is done in TLS. However, the deployment of such a new system is a huge effort. In this paper, we focus on improving the security of the CA-PKI on the short term, offering solutions that can be deployed today to provide additional security benefits to individual users immediately.

Akhawe et al. [11] looked at click-through rates for SSL warning messages in browsers and found that users ignore one quarter to one third of all validation errors. Based on a large dataset of TLS handshakes, Akhawe et al. [12] aimed to reduce the number of warning messages that are due to configuration or administration errors. By relaxing the validation algorithm, i.e. allowing a certificate that was issued for a certain domain to also be used for the *www* sub domain they were able to reduce the number of warnings the end user has to deal with.

In a related effort to reduce the trust put into CAs, Kasten et al. [13] analyzed which CAs usually sign for which TLDs and suggest restricting CA signing capabilities based on their signing history. They show that this can be effective, however, their system also requires some fundamental changes to the CA system.

3 Technical Setup

In order to evaluate which CAs could potentially be removed, we ran extensive analyses and simulations to assert that our recommendations would not lead to false positive SSL warnings. We used two different data sets for the analysis: a collection of certificates from Internet-wide ZMAP scans (the *ZMAP database*) [2], as well as all CA certificates found in trust stores (the *trust store database*). Additionally, we used a collection of two months' worth of TLS handshakes collected in our university's network in order to assert that the reduced set of CAs is still capable of validating all certificates our users encounter.

The ZMAP database consists of approximately 48 million certificates collected in periodical scans of port 443 in 2012 and 2013. For each certificate in the ZMAP database, the chain from the leaf certificate to a self-signed root

was rebuilt by validating the signature of the child certificate with the parent's public key. This step was important as, according to RFC 5280 [14], HTTPS servers only need to supply intermediate CAs, not the trusted root CA. With the reconstructed chain, our dataset is independent of the server administrators' configurations.

For the trust store database, we scraped certificates from twelve trust stores used in smart phone operating systems (Android, BlackBerry, iOS), Linux distributions (CentOS, Debian, Gentoo, openSUSE, Ubuntu), as well as Mozilla Firefox, OpenBSD, OS X and Windows 8. Google Chrome does not have a trust store of its own but rather uses the trust store of the underlying operating system. Since Apple has the same policies for iOS as for OS X, both of those trust stores contain the same CA certificates. Table 1 shows the size of the trust stores we analyzed. Our further analysis is based on these datasets.

4 Trusted Root CA Certificates

The set of CA certificates included in different trust stores varies significantly. While there is a core set of 114 certificates that are included in all major trust stores (Windows, OS X, iOS, Android, Mozilla), only 28 CA certificates are present in all eleven trust stores (counting iOS and OS X as one), c.f. Figure 1.

Fig. 1. How many certificates are included in 11 (all), 10, ..., trust stores?

4.1 Windows Trust Store

With 377 certificates, the Windows trust store is the largest by far. Moreover, of the 202 CA certificates included in only one trust store, 168 are included only in

the Windows trust store. This is partially due to the fact that the Windows trust store also contains a large number of CA certificates used for other purposes like email encryption (S/MIME) or code signing. It is possible for the Windows trust store to restrict the purpose a CA certificate can be used for, however, this is hardly done in practice. This unfortunately means that all these CAs are also trusted for HTTPS connections.

However, users may not notice how many CAs they trust, as additional CA certificates may be downloaded from the Microsoft servers as needed. Certificates can be inspected and manipulated using either the Microsoft Management Console or through the `certmgr.exe` command line tool.

4.2 OS X and iOS Trust Store

In OS X, administration of CA certificates is done through either the Keychain app or the `security` command line tool. Although the trust for a CA certificate can be customized to e.g. never trust the certificate for SSL, no certificate has those restrictions enabled by default. Furthermore, Apple includes their Apple Root Certificate Authority certificate in the iOS and OS X trust stores, which has never been used to sign a certificate used for HTTPS.

4.3 Linux/OpenBSD Trust Stores

On Linux and OpenBSD, the certificates are usually stored in a directory. By default this is `/etc/ssl/certs/`. While this makes adding and deleting certificates trivial, it is not possible to restrict the purpose of the CA certificate, for instance, to only use it for code signing.

However, the trust stores of Linux distributions are more consensus-driven: No CA certificates appear in only one trust store on these platforms. On the other side, OpenBSD is the only trust store that still includes an old CAcert Class 3 Root, while all other trust stores (that trust CAcert) include a newer CA certificate.

4.4 Mobile Trust Stores (Android, BlackBerry)

According to our measurements, trust stores on mobile devices tend to be both smaller in size (146 CA certificates for Android, 90 for BlackBerry), and have less unused CA certificates. Further, none of these trust stores have CA certificates that no one else trusts. This shows that it is possible to build a trust store focusing on small size and consensus while supporting all CAs needed for HTTPS.

4.5 Restricting the Purpose of CA Certificates

The Windows and OS X trust stores theoretically allow restricting CA certificates so that they can only be used for specific purposes like code signing, SSL,

S/MIME, etc. However, we did not find any purpose-restricted CA certificates. While Windows and OS X do not use this sensible option, Linux does not offer it at all.

Table 1. Used and unused CA certificates in trust stores.

Platform	Total certs	Unused certs	To be removed	Unknown purpose	Purpose restrictable?	Restrictions used?
Windows	377	122	114	8	✓	—
Mozilla	172	23	15	8	✓	—
OS X/iOS	207	46	38	8	✓	—
Ubuntu	159	23	15	8	—	—
Debian	159	23	15	8	—	—
Gentoo	159	23	15	8	—	—
Android	146	15	7	8	—	—
openSUSE	144	14	6	8	—	—
CentOS	120	16	10	6	—	—
BlackBerry	90	14	7	7	—	—
OpenBSD	60	17	14	3	—	—
total	431	148	140	8		

5 Removing Unneeded CAs

Roughly 34 % of all CA certificates are never used for signing HTTPS certificates. Obviously certificates could be used for other purposes and HTTPS is not the only (although most prominent) use of TLS. However: these 148 certificates can be used for signing certificates and thus for launching a MITM attack. By distrusting these CAs for SSL connections, the number of potential weakest links is reduced in a simple and straightforward manner.

Instead of removing only non-signing CAs, we further checked in how many trust stores the CAs are included. However, this only makes a difference for very few CA certificates: Of the 148 unused certificates, 140 are not included in all twelve trust stores, and 140 are not included in all major trust stores (Windows, OS X, iOS, Android, Mozilla).

Based on these results, we make two recommendations: conservative and very conservative. In the conservative recommendation, we propose that users distrust (remove/restrict) all CAs that have never signed an HTTPS certificate. This would lead to the removal of 148 CAs over all trust stores. We consider this a safe choice, since it is based on the ZMAP datasets and thus no known HTTPS certificate would create a false positive warning. Our very conservative recommendation only removes those 140 CAs which are not contained in the trust stores of Microsoft, Apple, Google, and Mozilla. Table 1 shows how many certificates could be removed from which trust store. While both recommendations are safe in relation to the ZMAP dataset, the very conservative recommendation is safer with respect to the possible use of a previously unused CA for signing a HTTPS certificate. However, it should be noted that especially the CAs included

in all the major trust stores but have never been seen to sign an HTTPS certificate could be considered a risk factor for government coercion. The number of these CAs per trust store is listed in the *Unknown Purpose* column of Table 1.

5.1 Potential Problems and Current Solutions

Problem: False positive warnings. Removing CA certificates from the trust store could have annoying and – in the long term – potentially dangerous consequences. If users encounter a certificate that was ultimately signed by a removed CA, they will see a warning. No matter whether the users click through the warning message or stop using the site, this would encourage habituation of warning messages and further weaken the effectiveness of SSL warnings – at least for that site (c.f. [15, 16]). Therefore, when removing CA certificates, care must be taken that no legitimate certificates become invalid.

Solution. We ensured this by using a current, extensive database of HTTPS certificates that represents the current SSL landscape. Additionally, we evaluated our solution on a database of 130 million SSL handshakes and found that the proposal would not invalidate any previously valid certificates.

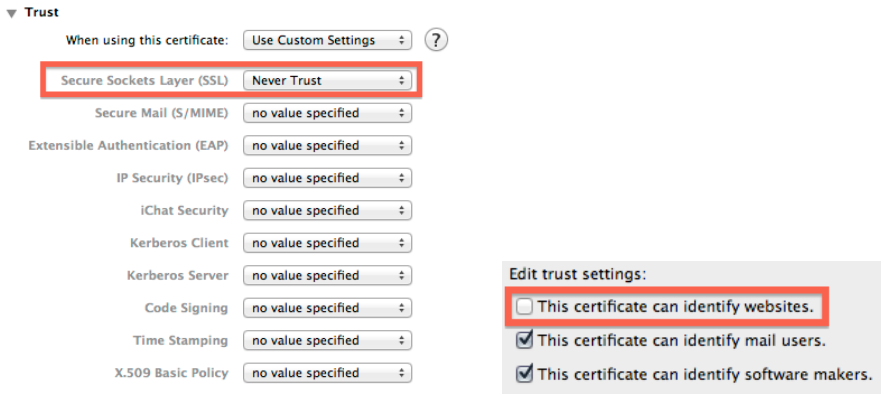
Problem: CA certificates are used for other purposes than SSL. As described above, our database only includes certificates for HTTPS servers. Thus CA certificates that are only used for code signing, IPSec gateways, or S/MIME would go unnoticed, be removed and could break functionality.

Solution. We counter this problem in two different ways. For the browser-based trust stores, there does not seem to be a reason to include CAs that do not sign HTTPS certificates, so they can simply be removed. For the Windows and OS X trust stores we recommend removing the HTTPS capabilities of those certificates (c.f. Figure 2). This is a conservative approach which still leaves the user open to MITM attacks for protocols such as S/MIME, however, further research is needed to determine the relevance of CAs for other protocols. Until then, breaking (non-HTTPS) SSL functionality by removing CAs too aggressively does not seem like a good idea. One caveat lurks on the Windows platform starting with Windows 7. From 7 on, Microsoft only ships a small set of CAs during the installation, but may load additional CAs on demand. This presents a unnecessary danger for the user, since it is not possible to restrict the capabilities of CAs which have not been downloaded yet. To counter this, we trigger the download of all CAs trusted by Windows and then edit the trust settings. This prevents them from being downloaded on demand with more capabilities than they need.

A critical exception to our approach is Linux which is not capable of restricting what a trusted CA can do: It is only possible to remove the CA entirely, which endangers any browser relying on the OS trust store. Interestingly, while Google's Chrome browser relies on the OS trust store on Windows and OS X,

they use their own approach on Linux. The trust settings for Chrome on Linux can be configured using `certutil`, which is part of the NSS command line tools.

The potential problems for mobile devices are still work in progress. Both iOS and Android also use CA certificates for other protocols, such as RADIUS. Thus there could potentially be problems if CAs are removed solely because they have never signed a HTTPS certificate.



(a) Disabling a certificate in OS X

(b) Disabling a certificate in Mozilla Firefox

Fig. 2. Disabling certificates for the purpose of SSL/HTTPS

6 Conclusion

In this paper we argued for the removal of CA certificates that do not sign any certificates used in HTTPS connections from desktop and browser trust stores. We based our analysis on an Internet-wide dataset of 48 million HTTPS certificates and compared them to trust stores from all major browser and OS vendors. We were able to identify 140 CA certificates included in twelve trust stores from all major platforms that are never used for signing certificates used in HTTPS. Based on these findings, we suggest to remove or restrict these CA certificates. Using two months' worth of TLS handshake data from our university network, we confirmed that removing these certificates from users' trust stores would not result in a single HTTPS warning message. Thus, this action provides a simple and low-cost real-world improvement that users can implement right now to make their HTTPS connections more secure. We are working on creating tools and scripts to automate this process for different browsers and operating systems.

Our current list of CAs we recommend for removal is a conservative one. It includes all CAs that have never signed a HTTPS certificate. In future work,

we would like to analyze the trade-off between false positives and the size of the trust store, as well as look into mechanisms to restrict the capabilities of certificates on the Android platform.

References

1. EFF: SSL Observatory
2. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: Fast Internet-wide scanning and its security applications. In: Proceedings of the 22nd USENIX Security Symposium. (2013)
3. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In: USENIX 2008 Annual Technical Conference on Annual Technical Conference, Boston, Massachusetts (2008) 321–334
4. Marlinspike, M.: SSL And The Future Of Authenticity. In: BlackHat USA 2011
5. Laurie, B., Langley, A., Kasper, E.: Certificate Transparency. RFC 6962 (Experimental) (June 2013)
6. Eckersley, P.: Sovereign Key Cryptography for Internet Domains
7. Hyun-Jin Kim, T., Huang, L.S., Perrig, A., Jackson, C., Gligor, V.: Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure. In: Proceedings of the 2013 Conference on World Wide Web. (2013)
8. Marlinspike, M.: TACK: Trust Assertions for Certificate Keys
9. Hoffman, P., Schlyter, J.: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard) (August 2012)
10. Lian, W., Rescorla, E., Shacham, H., Savage, S.: Measuring the practical impact of DNSSEC deployment. In: Proceedings of the 22nd USENIX conference on Security, USENIX Association (2013) 573–588
11. Akhawe, D., Felt, A.P.: Alice in warningland: A large-scale field study of browser security warning effectiveness. In: Proceedings of the 22th USENIX Security Symposium. (2013)
12. Akhawe, D., Amann, B., Vallentin, M., Sommer, R.: Here's my cert, so trust me, maybe?: understanding TLS errors on the web. In: Proceedings of the 22nd international conference on World Wide Web, International World Wide Web Conferences Steering Committee (2013) 59–70
13. Karsten, J., Wustrow, E., Halderman, J.A.: CAge: Taming Certificate Authorities by Inferring Restricted Scopes. In: FC'13: Proceedings of the 17th international conference on Financial Cryptography and Data Security. (2013)
14. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (May 2008) Updated by RFC 6818.
15. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying wolf: An empirical study of SSL warning effectiveness. In: Proceedings of the 18th Usenix Security Symposium. (2009)
16. Egelman, S., Cranor, L.F., Hong, J.: You've been warned. In: Proceeding of the twenty-sixth annual CHI conference, New York, New York, USA, ACM Press (2008) 1065–1074