Trust and the WebPKI

CS249i
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    03:00:dd:56:14:c0:63:05:ef:75:00:f4:08:da:9f:e5:a5:60
Signature Algorithm: sha256WithRSAEncryption
  Issuer:
    commonName             = PRINTABLESTRING:R3
    organizationName       = PRINTABLESTRING:Let's Encrypt
    countryName             = PRINTABLESTRING:US
Validity
  Not Before: Jan 9  22:06:26 2023 GMT
  Not After : Apr 9  22:06:25 2023 GMT
Subject:
  commonName              = PRINTABLESTRING:zakird.com
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public-Key: (2048 bit)
Symantec / DigiCert operated root c38dcb389593...

<table>
<thead>
<tr>
<th>commonName</th>
<th>UTN-USERFirst-NetworkApplications</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgUnitName</td>
<td><a href="http://www.usertrust.com">http://www.usertrust.com</a></td>
</tr>
<tr>
<td>orgName</td>
<td>The USERTRUST Network</td>
</tr>
<tr>
<td>localityName</td>
<td>Salt Lake City</td>
</tr>
<tr>
<td>stateOrProvinceName</td>
<td>UT</td>
</tr>
<tr>
<td>countryName</td>
<td>US</td>
</tr>
</tbody>
</table>

Comodo / Sectigo operated root 43f257412d44...

<table>
<thead>
<tr>
<th>commonName</th>
<th>UTN-USERFirst-Client Authentication and Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgUnitName</td>
<td><a href="http://www.usertrust.com">http://www.usertrust.com</a></td>
</tr>
<tr>
<td>orgName</td>
<td>The USERTRUST Network</td>
</tr>
<tr>
<td>localityName</td>
<td>Salt Lake City</td>
</tr>
<tr>
<td>stateOrProvinceName</td>
<td>UT</td>
</tr>
<tr>
<td>countryName</td>
<td>US</td>
</tr>
</tbody>
</table>

Figure 2: Misleading Names—The Subject fields of two roots previously operated by Symantec/DigiCert and Comodo/Sectigo illustrate that 1) the names in CA certificates do not reflect their operators, and 2) similar certificate names have no bearing on shared control.

![Table 1: Symantec Distrust—Blacklisting of Symantec-controlled roots involved 58 root certificates [4] with 8 separate orgs. in their X.509 Subject field. These orgs. are linked through a scattered history of corporate spin-offs and acquisitions.](image)

<table>
<thead>
<tr>
<th>Organization</th>
<th>#</th>
<th>Symantec Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symantec</td>
<td>10</td>
<td>–</td>
</tr>
<tr>
<td>TC TrustCenter</td>
<td>10</td>
<td>Acquired by Symantec (2010) [75]</td>
</tr>
<tr>
<td>GeoTrust</td>
<td>8</td>
<td>Acquired by VeriSign (2006) [53]</td>
</tr>
<tr>
<td>Equifax</td>
<td>4</td>
<td>Acquired by GeoTrust (2001) [2]</td>
</tr>
<tr>
<td>UserTrust</td>
<td>1</td>
<td>GeoTrust partnership (2001) [76]</td>
</tr>
<tr>
<td>Thawte</td>
<td>10</td>
<td>Acquired by VeriSign (1999) [34]</td>
</tr>
<tr>
<td>RSA Data Sec.</td>
<td>1</td>
<td>Spun out VeriSign (1995) [33]</td>
</tr>
</tbody>
</table>

What’s in a Name? Exploring CA Certificate Control
• Zane Ma, Joshua Mason, Manos Antonakakis, Zakir Durumeric, and Michael Bailey
•USENIX Security Symposium, August 2021
The Common CA Database (CCADB) is a repository of information about externally operated Certificate Authorities (CAs) whose root and intermediate certificates are included within the products and services of CCADB root store members. Root store operators participate in the CCADB to improve security, transparency, and interoperability. The CCADB is used by a number of different root store operators to manage their root stores, but it is run by Mozilla.
Root Stores
Let's Start at the Top — Root Stores

Figure 2: Root Store Ecosystem—The TLS root store ecosystem is an inverted pyramid, with a majority of clients trusting one of four root families.
Change Over Time

Figure 1: Root Store Similarity—Performing MDS on the Jaccard distance between root store providers from 2011–2021 illustrates four distinct clusters of roots. From left to right: Microsoft, NSS-like, Apple, Java.
Certificate Issuance and Some History
Historical Validation and Issuance

Goals:
1) verify that a network identifier (i.e., IP address or DNS Name) controls some cryptographic public key
2) generate a certificate that attests to this linkage.

How to verify? What does “control” mean?
Historically... (around 2012...)

Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar; Communicating directly with Registrant via address, email, or telephone number provided by the Registrar;

Communicating directly with the Registrant using the contact information listed in the WHOIS record’s “registrant”, “technical”, or “administrative” field;

Communicating with the Domain’s administrator using an email address created by pre-pending ‘admin’, ‘administrator’, ‘webmaster’, ‘hostmaster’, or ‘postmaster’ followed by the Domain Name;

Relying upon a Domain Authorization Document;

Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described.
Since then...

(If CA == DNS Registrar) Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar;

Communicating directly with Registrant via address, email, or telephone number provided by the Registrar; Communicating directly with the Registrant using the contact information listed in the WHOIS record’s “registrant”, “technical”, or “administrative” field; (based on WHOIS info)

Communicating with the Domain's administrator using an email address created by pre-pending ‘admin’, ‘administrator’, ‘webmaster’, ‘hostmaster’, or ‘postmaster’ followed by the Domain Name;

Relying upon a Domain Authorization Document; Removed

Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described.
Certificates were not free!

<table>
<thead>
<tr>
<th>Certificate Authority</th>
<th>Mid-2015 Prices</th>
<th>Mid-2019 Prices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
<td>Wildcard</td>
</tr>
<tr>
<td>GoDaddy [45, 46]</td>
<td>$69</td>
<td>$332</td>
</tr>
<tr>
<td>Comodo/Sectigo [31, 85]</td>
<td>$76</td>
<td>$404</td>
</tr>
<tr>
<td>GeoTrust [43, 44]</td>
<td>$149</td>
<td>$499</td>
</tr>
<tr>
<td>DigiCert [32, 33]</td>
<td>$195</td>
<td>$595</td>
</tr>
<tr>
<td>Symantec [92, 93]</td>
<td>$399</td>
<td>$1999</td>
</tr>
</tbody>
</table>

**Table 1: Prices for a one-year certificate** for non-free CAs with the largest market shares. Single domain offerings are domain-validated; wildcard offerings sometimes require organization validation. The 2015 prices are from shortly before Let’s Encrypt began offering service to the public.
2011: DigiNotar Certificate Authority

Dutch Certificate Authority

Compromised in September 2011 — issued fraudulent certificates

Dutch Government took over operational control. Declared bankruptcy within three weeks — after distrusted by major browsers
Turkish CA issued fraudulent certificate for google.com

“Microsoft is aware of active attacks using one fraudulent digital certificate issued by TURKTRUST Inc., which is a CA present in the Trusted Root Certification Authorities Store,” an advisory from Microsoft noted.

Mozilla distrusted CA.
2013: CAs Discovered through Internet Scanning

Identified 1,800 CA certificates belonging to 683 organizations

- Including religious institutions, libraries, non-profits, financial institutions, governments, and hospitals

- More than 80% of organizations controlling a CA certificate aren't commercial certificate authorities

More than half of the certificates were provided by the German National Research and Education Network (DFN)

All major browser roots are selling intermediates to third-party organizations without any constraints
2013: CAs Discovered through Internet Scanning

The largest commercial provider of intermediate certificates is GTE CyberTrust Solutions, Inc., a subsidiary of Verizon Business, which has provided intermediate signing certificates to 49 third-party organizations ranging from Dell Inc. to Louisiana State University. Comodo (under the name The USERTRUST Network) provided intermediates to 42 organizations and GlobalSign to 20. We also saw a number of commercial authorities that provided a smaller number of certificates to seemingly unrelated entities. For example, VeriSign, Inc. provided intermediates for Oracle, Symantec, and the U.S. Government; SwissSign AG provided certificates for Nestle, Trend Micro, and other Swiss companies; StartCom Ltd.
Let's Encrypt
A nonprofit Certificate Authority providing TLS certificates to **300 million** websites.

Read all about our nonprofit work this year in our [2022 Annual Report](#).

**Get Started**  **Sponsor**
ACME Protocol

1. Client registers, provides public key

2. Client sends certificate order, provides list of identifiers (e.g. domains) to include in certificate

3. CA generates random token

4. CA sends token and list of challenges client can complete to prove ownership of identifier

5. Client selects the HTTP challenge, creates file containing token and hosts it at ./well-known/acme-challenge/<token>

6. Client informs CA that challenge complete

7. CA verifies that file is reachable

8. Client sends PKCS 10 Certificate Signing Request

9. CA issues certificate

For each identifier in the order:

When all identifiers verified:
Validation Methods

There are currently three specified challenge types, all of which are supported by Let’s Encrypt:

1. The **HTTP** challenge requires the applicant to serve an object containing a CA-provided random value at a specific HTTP URL at the domain. The CA makes GET requests for the URL and verifies that the correct object is returned.

2. The **DNS** challenge requires the applicant to provision a DNS record at `_acme-challenge.<domain>` containing a CA-provided random value. The CA fetches this record and verifies that its content is correct.

3. The **TLS-ALPN** challenge requires the applicant to configure a TLS server to respond to a TLS ClientHello message containing a specific ALPN value and an ACME-specific TLS extension [42, 87]. The TLS server must then present a self-signed certificate containing a CA-provided random value and correctly complete the TLS handshake.
Since then... automated issuance

**Automatable**

A. Manual Contact
- Phone
- Fax
- Postal Mail
- SMS
- Email

B. Random Token
- Step 1. CA sends token to client
- Step 2. CA retrieves token from:
  - DNS server
  - HTTP / TLS server

Contact Info Source
- WHOIS
- DNS (TXT, CAA)
- ‘admin’-like email
Figure 8: Certificate authority flow among stable, popular sites. We track CA choice for 141K domains over five snapshots, from 7/2015 to 1/2019. The included sites are those that were ranked in the Alexa Top Million at every snapshot, and so are likely more popular and long-lived than the top million overall.
Figure 5: Firefox HTTPS connections by trust anchor. We show the trust anchors responsible for authenticating full TLS handshakes by Firefox Beta users. Let’s Encrypt has become the fourth largest known CA.
<table>
<thead>
<tr>
<th>parsed.issuer_dn.raw</th>
<th>Certificates</th>
</tr>
</thead>
<tbody>
<tr>
<td>C=US, O=Let’s Encrypt, CN=R3</td>
<td>386,737,770 51.18%</td>
</tr>
<tr>
<td>C=US, O=Cloudflare, Inc., CN=Cloudflare Inc ECC CA-3</td>
<td>63,758,897 8.44%</td>
</tr>
<tr>
<td>C=US, O=Google Trust Services LLC, CN=GTS CA 1P5</td>
<td>39,029,449 5.16%</td>
</tr>
<tr>
<td>C=US, O=Amazon, OU=Server CA 1B, CN=Amazon</td>
<td>30,646,721 4.06%</td>
</tr>
<tr>
<td>C=US, O=Let’s Encrypt, CN=E1</td>
<td>29,673,776 3.93%</td>
</tr>
<tr>
<td>C=US, ST=TX, L=Houston, O=cPanel\ Inc., CN=cPanel\ Inc. Certification Authority</td>
<td>28,955,770 3.83%</td>
</tr>
<tr>
<td>C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo RSA Domain Validation Secure Server CA</td>
<td>27,686,231 3.66%</td>
</tr>
<tr>
<td>C=US, O=DigiCert Inc, OU=www.digicert.com, CN=Encryption Everywhere DV TLS CA - G1</td>
<td>20,653,171 2.73%</td>
</tr>
<tr>
<td>C=US, O=Cloudflare, Inc., CN=Cloudflare Inc RSA CA-2</td>
<td>16,094,882 2.13%</td>
</tr>
<tr>
<td>C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo ECC Domain Validation Secure Server CA</td>
<td>11,421,489 1.51%</td>
</tr>
<tr>
<td>C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA</td>
<td>11,120,657 1.47%</td>
</tr>
<tr>
<td>C=US, ST=Arizona, L=Scottsdale, O=GoDaddy.com\ Inc., OU=<a href="http://certs.godaddy.com/repository/">http://certs.godaddy.com/repository/</a>, CN=Go Daddy Secure Certificate Authority - G2</td>
<td>9,358,404 1.24%</td>
</tr>
</tbody>
</table>
Let’s Encrypt: An Automated Certificate Authority to Encrypt the Entire Web

Josh Aas
Let’s Encrypt

Richard Barnes
Cisco

Benton Case
Stanford University

Zakir Durumeric
Stanford University

Peter Eckersley
Electronic Frontier Foundation

Alan Flores-López
Stanford University

J. Alex Halderman
University of Michigan

Jacob Hoffman-Andrews
Electronic Frontier Foundation

James Kasten
University of Michigan

Eric Rescorla
Mozilla

Seth Schoen
Electronic Frontier Foundation

Brad Warren
Electronic Frontier Foundation

1 INTRODUCTION

HTTPS [78] is the cryptographic foundation of the Web, providing an encrypted and authenticated form of HTTP over the TLS transport [79]. When HTTPS was introduced by Netscape twenty-five years ago [5], the primary use cases were protecting financial transactions and login credentials, but users today face a growing range of threats from hostile networks—including mass surveillance and censorship by governments [99, 106], consumer profiling and ad injection by ISPs [99, 96], and insertion of malicious code by network devices [88]—which make HTTPS important for practically every Web request. Many cryptographic flaws in TLS have been discovered and mitigated (e.g., [11, 13, 17, 23, 37, 69]), but low adoption of HTTPS poses an even starker risk as recently as 2015, 55–70% of browser page loads used plaintext HTTP [47].

A major barrier to wider HTTPS adoption was that deploying it was complicated, expensive, and error-prone for server operators [22, 57]. Most of the difficulty involved interactions with Certificate Authorities (CAs), entities trusted by Web browsers to validate a server’s identity and issue a digitally signed certificate binding the identity to the server’s public key. (Modern TLS implementations have negligible performance overhead in typical applications [48,

ABSTRACT

Let’s Encrypt is a free, open, and automated HTTPS certificate authority (CA) created to advance HTTPS adoption to the entire Web. Since its launch in late 2015, Let’s Encrypt has grown to become the world’s largest HTTPS CA, accounting for more currently valid certificates than all other browser-trusted CAs combined. By January 2019, it had issued over 538 million certificates for 223 million domain names. We describe how we built Let’s Encrypt, including the architecture of the CA software system (Boulder) and the structure of the organization that operates it (ISRG), and we discuss lessons learned from the experience. We also describe the design of ACME, the IETF-standard protocol we created to automate CA–server interactions and certificate issuance, and survey the diverse ecosystem of ACME clients, including Certbot, a software agent we created to automate HTTPS deployment. Finally, we measure Let’s Encrypt’s impact on the Web and the CA ecosystem. We hope that the success of Let’s Encrypt can provide a model for further enhancements to the Web PKI and for future Internet security infrastructure.

CCS CONCEPTS

- Networks → Web enabled security; - Security and privacy →
Certificate Transparency and Ecosystem Health
Certificate Transparency...
What if we put every certificate in a merkle tree?

Abstract

This document describes an experimental protocol for publicly logging the existence of Transport Layer Security (TLS) certificates as they are issued or observed, in a manner that allows anyone to audit certificate authority (CA) activity and notice the issuance of suspect certificates as well as to audit the certificate logs themselves. The intent is that eventually clients would refuse to honor certificates that do not appear in a log, effectively forcing CAs to add all issued certificates to the logs.

Logs are network services that implement the protocol operations for submissions and queries that are defined in this document.
1. Domain owner requests certificate from Certificate Authority (CA).
2a. CA sends pre-certificate.
2b. CA adds certificate to logs, powered by Merkle trees.
3. Domain owner receives certificate and sends certificate and SCTs.
4. Domain owner serves website and certificate.
5. User agents access website via browser through HTTPS.
7. Monitors notify user agents of newly issued certificates.
+ Signed Certificate Timestamp
func (l *InvalidCertificateVersion) Execute(c cert *X509.Certificate) *LintResult {
    if cert.Version != 3 {
        return &LintResult{Status: Error}
    }
    return &LintResult{Status: Pass}
}

func init() {
    RegisterLint(&Lint{
        Name: "e_invalid_certificate_version",
        Description: "Certificates MUST be of type X.509 v3",
        Source: CABFBaselineRequirements,
        Citation: "CABF BR 7.1.1",
        EffectiveDate: util.CABV130Date,
        Lint: &InvalidCertificateVersion{},
    })
}

Code Block 1: Example Lint—Lints are self-contained Go functions that check for adherence with technical standards. This lint checks that a certificate uses the correct X.509 version.

---

Tracking Certificate Misissuance in the Wild

Deepak Kumar*, Zhengqing Wang*, Matthew Hyder*, Joseph Dickinson*, Gabrielle Beck†, David Adrian†, Joshua Mason†, Zakir Durumeric‡, J. Alex Halderman†, Michael Bailey*

*University of Illinois Urbana-Champaign  †University of Michigan  ‡Stanford University

Abstract—Certificate Authorities (CAs) regularly make mechanical errors when issuing certificates. To quantify these errors, we introduce ZLint, a certificate linting tool that codifies the policies set forth by the CA/Browser Forum Baseline Requirements and RFC 5280 that can be tested in isolation. We ran ZLint on browser-trusted certificates in Convy and systematically analyze how well CAs construct certificates. We find that the number of errors has dramatically reduced since 2012. In 2017, only 0.02% of certificates have errors. However, this is largely due to a handful of large authorities that consistently issue correct certificates. There remains a long tail of small authorities that regularly issue non-conformant certificates. We further find that issuing certificates with errors is correlated with other types of mismanagement and for larger authorities, browser action. Drawing on our analysis, we conclude with a discussion on how the community can best use lint data to identify authorities with worrisome organizational practices and errors long-term health of the Web PKI.

I. INTRODUCTION

HTTP(S) depends on a supporting public key infrastructure (PKI) composed of hundreds of certificate authorities (CAs) that verify the identities of websites and issue digital certificates. To ensure compatibility between browsers and HTTPS-enabled websites, standard bodies like the IETF and CA/Browser Forum have developed policies that govern the digital certificates that CAs provide. Unfortunately, there is a long history of certificate authorities failing to adhere to accepted standards, due to both implementation errors and indifference. In this paper, we systematically analyze the errors that authorities make when constructing certificates and consider whether these errors can be used to predict more serious problems.

We begin by describing the policies set forth by RFC...
WELCOME TO THE CA/BROWSER FORUM

Information for the Public

Organized in 2005, we are a voluntary group of certification authorities (CAs), vendors of Internet browser software, and suppliers of other applications that use X.509 v.3 digital certificates for SSL/TLS, code signing, and S/MIME.

Information for Site Owners and Administrators

The CA/Browser Forum began in 2005 as part of an effort among certification authorities and browser software vendors to provide greater assurance to Internet users about the websites they visit by leveraging the capabilities of SSL/TLS certificates. In June 2007, the CA/Browser Forum adopted version 1.0 of the Extended Validation (EV) Guidelines. EV certificates are issued after extended steps to verify the identity of the entity behind the domain receiving the certificate. Internet browser software displays enhanced indication of that identity by changing the appearance of its display (i.e. colors, icons, animation, and/or additional website information).

Ballot SC60: Membership of ZT Browser
2023-01-19 Minutes of the Server Certificate Working Group
2023-01-19 Minutes of the CA/Browser Forum Teleconference
2023-01-18 Minutes of the S/MIME Certificate Working Group
2023-01-12 Minutes of the Code Signing Certificate Working Group

Code Signing Forum
Network Security
S/MIME
Server Certificates
Fig. 5: Percent ZLint Errors by Total Certificates Issued—Large certificate authorities generally issue certificates with fewer ZLint errors than smaller authorities.
Fig. 6: Percent ZLint Warnings by Total Certificates Issued—ZLint warnings are more dispersed throughout the ecosystem, affecting both large and small players. A handful of large players, Symantec, GeoTrust, thawte, and TrustAsia, all issue more than 95% of their certificates with ZLint warnings.
Symantec Distrust

"Over a period of several years, Symantec willfully issues over 100 test certificates for 76 different domains without the authorization of the domain owners. This is discovered when Google's Certificate Transparency log monitor detects an unauthorized certificate for google.com in Certificate Transparency logs."

<INSERT TREMENDOUS DRAMA>

Symantec is distrusted by all major platforms due to general malfeasance.
2016 - StartCom

Thijs Alkemade discovers that StartCom's brand new automated issuance API suffers from numerous flaws, including flaws that had previously been discovered and fixed by other CAs, that would allow attackers to obtain certificates for domains they don't control.

Cause: StartCom ignored developments in the standards community and instead chose to design their own, insecure automated issuance API.

During the ensuing investigation, it is revealed that StartCom had concealed their purchase by WoSign, another incompetent certificate authority.

Initially, StartCom announces that all certificates they issue will be logged to Certificate Transparency logs, but they are ultimately distrusted by all major platforms due to their malfeasance.
Root Store Lag

Figure 3: NSS derivative staleness—No derivative root stores match NSS’s update regularity. Alpine Linux maintains closest parity to NSS, while AmazonLinux, on average, lags more than four substantial versions behind.
## Root Store Lag

<table>
<thead>
<tr>
<th>Root store</th>
<th># Certs</th>
<th>Trusted until</th>
<th>Lag (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DigiNotar [101]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft</td>
<td>1</td>
<td>2011-08-30</td>
<td>-37</td>
</tr>
<tr>
<td>Apple</td>
<td>1</td>
<td>2011-10-12</td>
<td>6</td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>1</td>
<td>2011-10-22</td>
<td>16</td>
</tr>
<tr>
<td>CNNIC [78]</td>
<td></td>
<td>2017-07-27</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>2</td>
<td>2015-06-30</td>
<td>-758</td>
</tr>
<tr>
<td>Android</td>
<td>1</td>
<td>2017-12-05</td>
<td>131</td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>2</td>
<td>2018-04-09</td>
<td>256</td>
</tr>
<tr>
<td>NodeJS</td>
<td>2</td>
<td>2018-04-24</td>
<td>271</td>
</tr>
<tr>
<td>AmazonLinux</td>
<td>2</td>
<td>2019-02-18</td>
<td>571</td>
</tr>
<tr>
<td>Microsoft</td>
<td>2</td>
<td>2020-02-26</td>
<td>944</td>
</tr>
<tr>
<td>StartCom [113]</td>
<td></td>
<td>2017-11-14</td>
<td></td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>3</td>
<td>2017-07-17</td>
<td>-120</td>
</tr>
<tr>
<td>Microsoft</td>
<td>2</td>
<td>2017-09-22</td>
<td>-53</td>
</tr>
<tr>
<td>Android</td>
<td>3</td>
<td>2017-12-05</td>
<td>21</td>
</tr>
<tr>
<td>NodeJS</td>
<td>3</td>
<td>2018-04-24</td>
<td>161</td>
</tr>
<tr>
<td>AmazonLinux</td>
<td>3</td>
<td>2019-02-18</td>
<td>461</td>
</tr>
<tr>
<td>Apple</td>
<td>3</td>
<td>1 root still trusted</td>
<td>1,175+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Root store</th>
<th># Certs</th>
<th>Trusted until</th>
<th>Lag (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WoSign [113]</td>
<td></td>
<td>2017-11-14</td>
<td></td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>4</td>
<td>2017-07-17</td>
<td>-120</td>
</tr>
<tr>
<td>Microsoft</td>
<td>4</td>
<td>2017-09-22</td>
<td>-53</td>
</tr>
<tr>
<td>Android</td>
<td>4</td>
<td>2017-12-05</td>
<td>21</td>
</tr>
<tr>
<td>NodeJS</td>
<td>4</td>
<td>2018-04-24</td>
<td>161</td>
</tr>
<tr>
<td>AmazonLinux</td>
<td>4</td>
<td>2019-02-18</td>
<td>461</td>
</tr>
<tr>
<td>PSPProcert [38]</td>
<td></td>
<td>2017-11-14</td>
<td></td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>1</td>
<td>2018-04-09</td>
<td>146</td>
</tr>
<tr>
<td>NodeJS</td>
<td>1</td>
<td>2018-04-24</td>
<td>161</td>
</tr>
<tr>
<td>AmazonLinux</td>
<td>1</td>
<td>2019-02-18</td>
<td>461</td>
</tr>
<tr>
<td>Certinomis [37]</td>
<td></td>
<td>2019-07-05</td>
<td></td>
</tr>
<tr>
<td>NodeJS</td>
<td>1</td>
<td>2019-10-22</td>
<td>109</td>
</tr>
<tr>
<td>Alpine</td>
<td>1</td>
<td>2020-03-23</td>
<td>262</td>
</tr>
<tr>
<td>Debian/Ubuntu</td>
<td>1</td>
<td>2020-06-01</td>
<td>332</td>
</tr>
<tr>
<td>Android</td>
<td>1</td>
<td>2020-09-07</td>
<td>430</td>
</tr>
<tr>
<td>AmazonLinux</td>
<td>1</td>
<td>2021-03-26</td>
<td>630</td>
</tr>
<tr>
<td>Apple</td>
<td>1</td>
<td>2021-01-01*</td>
<td>577</td>
</tr>
<tr>
<td>Microsoft</td>
<td>1</td>
<td>Still trusted</td>
<td>607+</td>
</tr>
</tbody>
</table>

*Revoked via valid.apple.com at unknown date.

Table 4: High severity removals—Comparison of root store responses to high severity NSS removals.
Announcing the Launch of the Chrome Root Program

Monday, September 19, 2022

In 2020, we announced we were in the early phases of establishing the Chrome Root Program and launching the Chrome Root Store.

The Chrome Root Program ultimately determines which website certificates are trusted by default in Chrome, and enables more consistent and reliable website certificate validation across platforms.

This post shares an update on our progress and how these changes help us better protect Chrome’s users.

What’s a root store or root program, anyway?

Chrome uses digital certificates (often referred to as “certificates,” “HTTPS certificates,” or “server authentication certificates”) to ensure the connections it makes on behalf of its users are secure and private. Certificates are responsible for binding a domain name to a public key, which Chrome uses to encrypt data sent to and from the corresponding website.

As part of establishing a secure connection to a website, Chrome verifies that a recognized entity known as a “Certification Authority” (CA) issued its certificate. Certificates issued by a CA not recognized by Chrome or a user’s local settings can cause users to see warnings and error pages.