



An Understand of WebPKI

Or What Does That Green Lock Actually Mean

What is WebPKI?

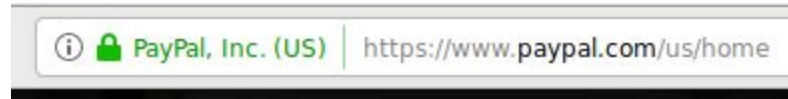
WebPKI is the infrastructure that provides encryption and authentication across the Internet.

Short for Web Public Key Infrastructure, it is a system of interconnected components that provide the backbone of encrypted communications across the Internet. This is built out of two different standards, the Transport Layer Security standard which provides encryption, and the x509 certificate standard which provides authentication.

In short it is this:



The Green Lock



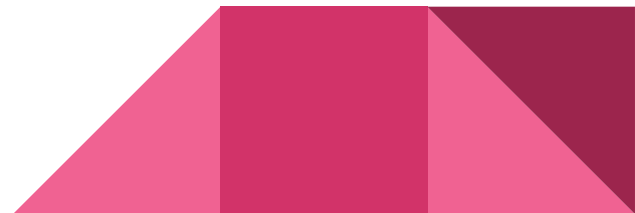
Authentication and Encryption

To provide a secure system, you need two elements.

Authentication: to prove who you are communicating with

Encryption: to prevent people from seeing what you're transmitting

WebPKI is the mechanism that provides both these things through two separate but interconnected services. The same PKI infrastructure is also used for secure email (S/MIME) and code signing (such as Microsoft Authenticode and Secure Boot).



Transport Layer Security

TLS is the encryption part of WebPKI, and provides the S in HTTPS.

Directly descended from Netscape's Secure Socket Layer, Transport Layer Security (TLS) uses secure handshaking and a set of known secure algorithms to provide end-to-end encryption from point to point.

TLS is used in web browsers, VPNs, email transmission and many more products to list here.

TLS was recently updated to version 1.3



Example Session

OpenSSL provides a simple client to connect to TLS-enabled sessions, and provide details.

For example, we'll use `openssl s_client` to connect to a secure webserver on port 443 (the secure HTTP port), and request a webpage.



openssl s_client

```
$ openssl s_client -connect soylentnews.org -port 443

CONNECTED(00000003)

...

SSL handshake has read 3771 bytes and written 302 bytes

Verification: OK

...

GET /

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

    "http://www.w3.org/TR/html4/strict.dtd">

<html>
```



Who Do We Prove Who We're Talking To

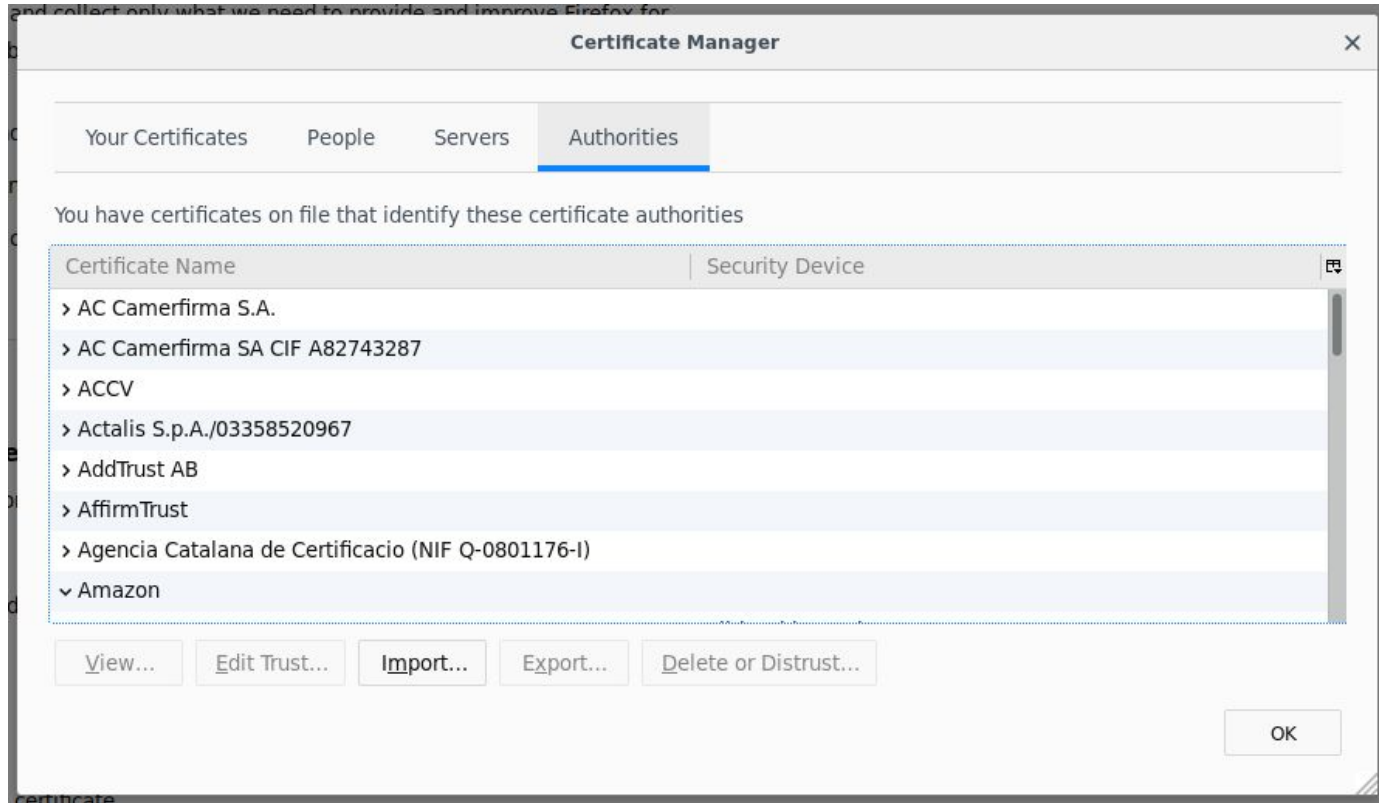
TLS provides encryption, making sure a third party can not intercept communications, but how do we know who we're actually talking to?

That is provided by a chain of trust of certificates issued from a set of root certificates. These root certificates provide the trust anchors in the chain of trust.

Certificate authorities are organizations who are roles who verify a host and issued a signature saying “this person is who they say they are”.



Firefox Certificate Root's



Root Certificates

There are hundreds of root certificates. These are collected in five relatively well known certificate stores.

- Microsoft's Trusted Root Certificate Store
 - Used by Internet Explorer, and any application using Windows SChannel API
 - Also used by Firefox on Windows in certain cases.
- Google's Certificate Storage (used by Chrome and Android)
- Apple's Root Store
- Oracle Java's Root Certificate Store
- Mozilla's NSS Certificate Store



Root Certificates (Continued)

Most open source projects base their root certificate store on Mozilla's roots.

Root certificates are typically hardcoded in applications and libraries and difficult to change. As they are trust anchors, they also can't be directly revoked or disabled short of removing them from the list of "accepted roots".

Furthermore, a root certificate can sign a certificate for any domain or purpose. A malicious actor who could gain access to a root certificate can effectively impersonate any website on the internet, undermine VPN security, or fraudulently create S/MIME signed emails.

In short, compromise of any root certificate is very bad.



Intermediate Certificates

Because of this risk, WebPKI root certificates are stored in hardware security managers (HSM), and not directly connected to the Internet. Furthermore, certain roots are only trusted for certain roles (known as keyUsages); for example, Let's Encrypt's root certificate is only trusted to issue certificates for use by web servers (known as serverAuth certificates).

To partially counteract this risk, intermediate certificates are created and used to delegate the functionality of a root certificate. Signing directly from the root is prohibited by the CA/B Forum Policy guidelines.



Intermediate Certificate Usage

Intermediate Certificates are standard x509 certificates with the CA:TRUE bit set; meaning they can in turn be used to sign for other certificates. In a typical scenario, you may have the following certificates directly signed from the root to provide services.

- Issuance certificates for a given role (such as serverAuth)
 - These are often pathLen restricted, meaning they can't create a third level of CA:TRUE certificates
- Revocation control certificates used for OCSP
- Name restricted certificates (more on this later)



The Chain of Trust of defcon201.org

To help digest this, let's look at the chain of trust for defcon201.org. At the time of writing, defcon201.org is signed by a Let's Encrypt certificate, and has a path length of two, meaning that there is a root certificate, one intermediate certificate, and the leaf certificate that verifies defcon201.org

This is the output of the Qualy's SSLabs testing tool which clearly shows the chain of trust from the root certificate.



Defcon201.org Certificate Chain

Path #1: Trusted

1	Sent by server	defcon201.org Fingerprint SHA256: f68c07de1b51719fb2b1ce18efe90dfee13621925b5318854a1e81f73b7228a5 Pin SHA256: 7w6ePIEOBva9ATlqlGedd47a8qslD90j7AnqtF/jIM= RSA 2048 bits (e 65537) / SHA256withRSA
2	Sent by server	Let's Encrypt Authority X3 Fingerprint SHA256: 25847d668eb4f04fdd40b12b6b0740c567da7d024308eb6c2c96fe41d9de218d Pin SHA256: YLh1dUR9y6Kja30RrAn7JKnbQQG/uEtLMkBgFF2Fuihg= RSA 2048 bits (e 65537) / SHA256withRSA
3	In trust store	DST Root CA X3 Self-signed Fingerprint SHA256: 0687260331a72403d909f105e69bcf0d32e1bd2493ffc6d9206d11bcd6770739 Pin SHA256: Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys= RSA 2048 bits (e 65537) / SHA1withRSA Weak or insecure signature, but no impact on root certificate

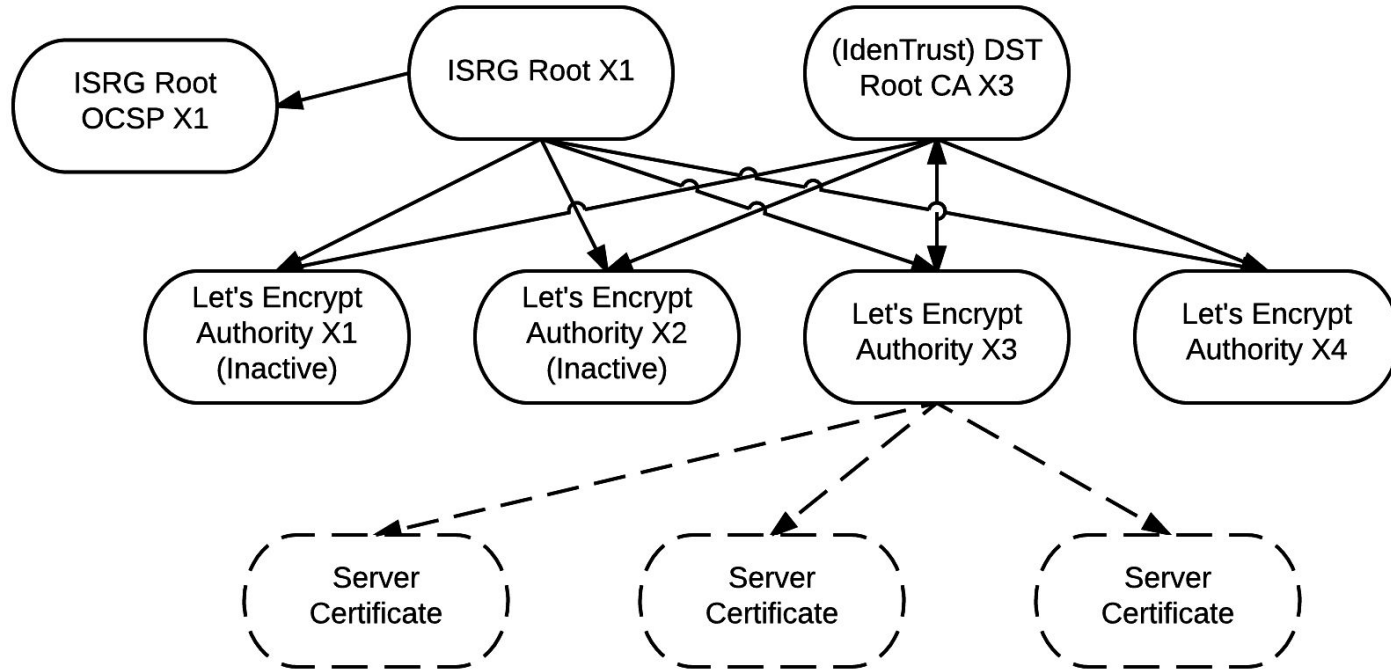
Chain of Trust Breakdown

Although Let's Encrypt is a root certificate authority (and not a subCA), their root certificate is relatively new and not included with all browsers yet. Instead, their intermediate certificates are cross-signed by IdenTrust, allowing legacy browsers to successfully validate them.

Let's Encrypt provides a helpful diagram breaking this down.



Chain of Trust Breakdown



Understand x509 Paths

If your brain has melted, that's not a problem. This is not the easiest concept to grasp. Let's start at the leaf certificate for defcon201.org and look at end user certificate first.

Defcon201.org has what's known as a domain validated (DV) certificate, so only the domain name itself is verified with no other details. The certificate information itself can easily be viewed in Chrome and Firefox.



Defcon201.org's leaf certificate

This certificate has been verified for the following usages:

SSL Server Certificate

Issued To

Common Name (CN)	defcon201.org
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	Let's Encrypt Authority X3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Sunday, November 18, 2018 at 11:35:19 PM
Expires On	Saturday, February 16, 2019 at 11:35:19 PM

Fingerprints


SHA-256 Fingerprint	F6 8C 07 DE 1B 51 71 9F B2 B1 CE 18 EF E9 0D FE E1 36 21 92 5B 53 18 85 4A 1E 81 F7 3B 72 28 A5
SHA-1 Fingerprint	8C 0D 1E F0 00 F6 AB 0A 97 BC 76 6E 03 DE 6D 8F 9D 2F 9C 16

Breakdown of a leaf certificate

Common to all x509 certificates are the following components:

- Allowed key usages, including if it's a CA certificate (CA:TRUE)
- Whom a certificate was issued to
- Whom a certificate was issued by
- A validity period of when the certificate was issued and when it expires.
- The entity the certificate covers (known as the commonName).

There is also a set of extended attributes that are important, but for the moment, we're only concerned with one, the subjectAlternateName.



subjectAlternateName and extendedKeyUsage

Leaf certificates can cover multiple names in a single certificate beyond what's specified in the commonName field. How can can be used is defined by the key usage field. These are known as subject alternative names and are encoded as an extension to the x509 certificate. Let's look at defcon201.org's SAN names and extendedKeyUsage.



DC201's subjectAlternateNames



The image shows a screenshot of a certificate's properties window. The 'Certificate Subject Alternative Name' field is selected and highlighted in blue. Below it, other certificate fields are listed: 'Certificate Policies', 'OID.1.3.6.1.4.1.11129.2.4.2', 'Certificate Signature Algorithm', 'Certificate Signature Value', and 'Fingerprints'. The 'Field Value' section below shows the specific values for the selected field.

Certificate Subject Alternative Name

Certificate Policies

OID.1.3.6.1.4.1.11129.2.4.2

Certificate Signature Algorithm

Certificate Signature Value

▶ Fingerprints

Field Value

Not Critical
DNS Name: dc201.org
DNS Name: defcon201.org
DNS Name: www.dc201.org
DNS Name: www.defcon201.org

DC201 Extended Key Usage

▼ Extensions

Certificate Key Usage

Extended Key Usage

Certificate Basic Constraints

Field Value

Not Critical

TLS WWW Server Authentication (OID.1.3.6.1.5.5.7.3.1)

TLS WWW Client Authentication (OID.1.3.6.1.5.5.7.3.2)

Leaf Certificate Summary

This certificate contains the following information

- It was signed by Let's Encrypt's X3 Intermediate Certificate
- It covers the following domains: dc201.org, defcon201.org, www.dc201.org, and www.defcon201.org
- It is valid for authenticating itself as as a webserver and as a webclient
 - The later is used for certain types of HTTP authentication, but is rare on the public internet
- It is valid until February 16th, 2019
- It does NOT contain authentication information about the organization

In short, it's everything important about the ID of defcon201.org




Sidenote: Types of Leaf Certificate

There are three types of leaf certificate used on the Internet today:

Domain Validated (DV): The certificate assert that the domain belongs to whomever owns it. These are the most common certificates in use on the Internet; and are the type issued by Let's Encrypt

Organization Validated (OV): These certificates also assert the organization that own a domain. They're extremely rare, and can be identified by the organization/state/country fields on the certification being not null. There is normally no user visable difference between a DV/OV certificate.



Sidenote: Types of Leaf Certificate

Extended Validation (EV): EV certifications are OV certificates that have done additional vetting and have additional rules such as no wildcards. EV certificates show up with a special green bar that show up with the organizations name at the top of the page.

Beside cosmetic differences and display of the organization, there is no difference in the type of security or level of encryption offered by these types of certificates.



What About The Intermediate Certificate

Since the leaf certificate contains everything we need to know about defcon201.org, let's go up the chain and take a look at the intermediate certificate, and see what information it contains.

Unfortunately, Chrome doesn't provide a nice interface to look at intermediate certificates, so we'll use the command line openssl utility to examine this certificate a little more in-depth. Some output has been excluded for brevity sake. Important information has been bolded



X3 Intermediate Certificate

```
$ openssl x509 -in /tmp/x3.pem -noout -text
```

```
Certificate:
```

```
Data:
```

```
Signature Algorithm: sha256WithRSAEncryption
```

```
Issuer: O = Digital Signature Trust Co., CN = DST Root CA X3
```

```
Validity
```

```
Not Before: Mar 17 16:40:46 2016 GMT
```

```
Not After : Mar 17 16:40:46 2021 GMT
```

```
Subject: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
```

```
X509v3 extensions:
```

```
X509v3 Basic Constraints: critical
```

```
CA:TRUE, pathlen:0
```

```
X509v3 Key Usage: critical
```

```
Digital Signature, Certificate Sign, CRL Sign
```

```
Authority Information Access:
```

```
OCSF - URI:http://isrg.trustid.ocsp.identrust.com
```

```
CA Issuers - URI:http://apps.identrust.com/roots/dstrootcax3.p7c
```

```
X509v3 CRL Distribution Points:
```

```
Full Name:
```

```
URI:http://crl.identrust.com/DSTROOTCAX3CRL.crl
```




Breakdown of the Intermediate Certificate

Compared to the leaf certificate, the intermediate certificate has relatively little information in it. The most relevant information is the subject, issuer, and the key authentication bits.

Let's start with the subject:

```
Subject: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
```

This identifies the certificate as as the certificate Let's Encrypt's intermediate certificate. The organization and country fields are filled out and set in this certificate.



Breakdown of the Intermediate Certificate

x.509 does not directly support multiple issuers (also known as cross-signing). Instead multiple versions of the certificate are created signed from different roots and a certificate chain is formed through that. In this version of the intermediate certificate, it was signed by the IdenTrust DST root certificate as seen by the issuer

```
Issuer: O = Digital Signature Trust Co., CN = DST Root CA X3
```

The most important part however comes from the usage bits which we'll cover next.




X3 Allowed Key Usage

From the dump

```
X509v3 extensions:  
  X509v3 Basic Constraints: critical  
    CA:TRUE, pathlen:0  
  X509v3 Key Usage: critical  
    Digital Signature, Certificate Sign, CRL Sign
```

This certificate has the CA:TRUE bit set, meaning it can sign for other certificates, but it's restricted to a pathlen of 0.

That means that it can't sign other certificates that are CA:TRUE as there can be no certificate chaining off this one except for leaf certificates.



X3 Allowed Key Usage

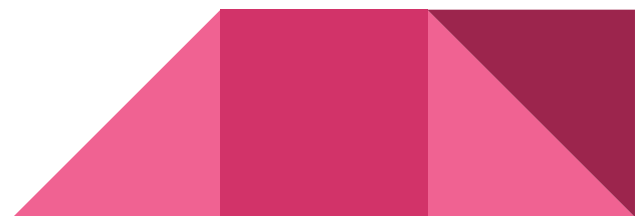
The key usage fields also limit what this certificate can be directly used for. Specifically, this intermediate certificate can sign other certificates (Certificate Sign), provide digital signatures (Digital Signature bit), and sign certificate revocation lists (one of two methods of revoking an otherwise valid certificate)

It also contains AIA information for OCSP, the other revocation method:

```
Authority Information Access:  
  OCSP - URI:http://isrg.trustid.ocsp.identrust.com  
  CA Issuers - URI:http://apps.identrust.com/roots/dstrootcax3.p7c
```

We'll talk about OCSP later in this presentation.

Before moving on, let's take a quick look at the DST root.



DST Root CA 3

Compared to the other certificates, the DST root certificate is relatively barren. It is a self-signed certificate as we'll see in the dump in the next slide), and merely has the CA:TRUE flag set with no other restrictions set on it.

Root certificate capabilities are limited by the certificate storage and the bits assigned to it (for example, is this a EV root, can it sign mail certificates, etc.).



Dump of the DST Root Certificate

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

44:af:b0:80:d6:a3:27:ba:89:30:39:86:2e:f8:40:6b

Signature Algorithm: sha1WithRSAEncryption

Issuer: O = Digital Signature Trust Co., CN = DST Root CA X3

Validity

Not Before: Sep 30 21:12:19 2000 GMT

Not After : Sep 30 14:01:15 2021 GMT

Subject: O = Digital Signature Trust Co., CN = DST Root CA X3

X509v3 extensions:

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Key Usage: critical

Certificate Sign, CRL Sign

X509v3 Subject Key Identifier:

C4:A7:B1:A4:7B:2C:71:FA:DB:E1:4B:90:75:FF:C4:15:60:85:89:10

Summarization of DST X3 Root Certificate

Unlike the intermediate certificate, the root certificate does not provide information relating to revocation; this specific root certificate pre-dates the AIA certificate standard where that information was encoded in the PEM certificate.

Furthermore, the root certificate is unrestrained; it can generate any type of certificate with any path-length and can be used to create sub-CAs; intermediate certificates with no restriction on path length.

In practice, intermediate certificates can't be revoked through traditional means.



Intermediate vs. Root Certificates

In practice, there's relatively little difference between an unrestrained intermediate certificate, and a root certificate.

If an unrestricted intermediate certificate (or even one with limited restrictions such as pathLen 0) were to leak, it would allow attackers to sign valid certificates for any website with any validity bits.


As intermediate certificates do not need to be directly included in a root store to create valid certificate, an undisclosed intermediate is an active threat to the security of the internet and can be very difficult to detect.



It gets worse ...

Throughout this talk, I have alluded to talking about certificate revocation. There are two mechanisms in x.509 certificates to revoke a misissued or lost certificate, known as certification revocation lists (CRLs) and the online certificate status protocol (OCSP).

While these mechanisms can theoretically handle the revocation of any type of certificate beyond a root certificate, in practice, policy and technical constraints prevents certificate revocation from being an effective tool at combating misissued or lost certificates. Let's quickly run through these.



Certificate Revocation Lists

CRLs, as the name suggests, are static files that simply contain a list of certificates that are revoked. CRL lists contain periods they are valid for notifying clients when they have to be updated. CRL download locations can be encoded with a certificate via AIA, or hardcoded within a root store.

They're automatically generated and signed by intermediate and root certificates whenever a certificate is revoked; it is the responsibility of the client to download and check CRL. This is problematic as CRLs can be very large, and for the most part are not used by modern web browsers in favor of OCSP.



Online Certificate Status Protocol

OCSP provides a leaner alternative to CRL lists; instead of downloading massive lists for each and every root certificate, a TLS client can simply query the OCSP server and ask if a certificate is valid or not. As such, OCSP is considerably more scalable than CRLs, and openssl contains a OCSP client that can be used to check certificate status.

Here's an example session



OCSP Session

```
$ openssl ocsp -issuer x3.pem -cert dc201.pem -text -url  
http://ocsp.int-x3.letsencrypt.org
```

OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3

Produced At: Jan 18 05:35:00 2019 GMT

Cert Status: good

This Update: Jan 18 05:00:00 2019 GMT

Next Update: Jan 25 05:00:00 2019 GMT

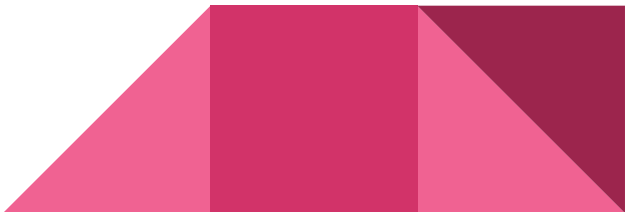
WARNING: no nonce in response

Response verify OK

dc201.pem: good

This Update: Jan 18 05:00:00 2019 GMT

Next Update: Jan 25 05:00:00 2019 GMT



The problems with OCSP

OCSP has two fundamental problems

- First, everytime the web browser (or any TLS client) makes a OCSP request, that information is sent to the certificate authority. **That means the CA can see the client IP address of anyone who's accessing a given website.**
- Second, the OCSP server can be down, firewalled, or filtered
 - This means a TLS client may not be able to query OCSP information for a given certificate.

Because of OCSP responders can be DDoSed or blocked by a firewall, **no web browser hard-fails if a OCSP response can not be obtained.**




It's worse than that.

Given an attacker could theoretically DDoS or otherwise manipulate traffic to prevent OCSP responses, browsers typically only soft-fail if they check OCSP status at all

Firefox, Edge, Internet Explorer, and Safari check OCSP, but do not hard fail.

Chrome does not make OCSP responses at all, nor does it check revocation status through any meaningful system; Google uses a CRLSet system which provides limited revocation for some percentage of the internet, but more likely than not, Chrome will accept a revoked certificate.



And for the cherry on top

Intermediate certificates are not checked for revocation status directly. In theory this is because the OCSP server should not generate good responses for unknown or revoked intermediate certificates.

That means a rogue intermediate certificate can be used with abandon as long as an attacker can successfully block OCSP responses from being received from the browser.

In practice, **WebPKI does not have a functional revocation system due to the insistence on soft-fail.**




The Stopgap Solutions

Undisclosed intermediate certificates are a well-known problem, and one that often comes up in policy discussion.

To deal with revocation, Firefox has implemented their own special system of killing intermediate certificates known as OneCRL. Chrome has similar functionality in CRLSets.

Often-times though, the core TLS libraries used by Firefox and other browsers are hacked to allow revocation in situations such as the recent Symantec certificate authority failure. This only works for the browser however.



These Are Not Theoretical Concerns

There have been multiple occasions where intermediate certificates have been misissued or misused.

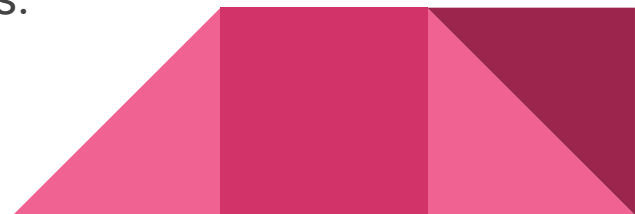
Rather famously, unrestrained intermediate certificates have been issued for TLS man-in-the-middle devices to allow deep packet inspection and allowing interception of encrypted session. Although disallowed by policy now, there is nothing preventing this type of attack from being done by a nation state power with access to a certificate authority.



Commercial Software Products Also Do This

Some commercial AV scanners, such as Anti-Virus Guard, install root certificates into all the certificate stores on a given system and perform interception of TLS traffic. This is done to allow to allow deep packet inspection to detect certain types of malware or data exfiltration.

It is also possible for Windows system administrators to generate and install root certificates via Active Directory which in turn can be used to intercept and man-in-the-middle otherwise secure session. An attacker can easily use this functionality to decrypt traffic from compromised devices.



To Summarize Revocation In a Nutshell

A misissued certificate is a disaster for the ecosystem as there is no meaningful way to revoke a certificate once issued that ensures that it can't be used.

While browsers can hack around the problem, users of the certificate store not using Mozilla's NSS will not receive protection against certificates that has special constraints such as the StartSSL, Symantec, or CNNIC root certificates authority which are still in the root store, but disallowed to issue new certificates due to receiving the certificate authority equivalent of a death sentence.

There is however hope.




Certificate Transparency

The first step in solving the problem is understanding the scope.

Certificate transparency is a mechanism built to allow all known public x.509 certificates to be uploaded to log servers and queried through various services such as crt.sh providing a comprehensive database of known leaf, intermediate, and root certificates served by web servers around the world.

To further aid study, Google's web crawlers automatically upload all certificates to the Google CT log servers. This means it is generally possible to see if any certificate has been issued for a given domain.



crt.sh's Undisclosed Intermediately Page

crt.sh Mozilla CA Certificate Disclosures

Generated at 2019-01-18 23:42:04 UTC

Category	Disclosure Required?	# of CA certs
Disclosure Incomplete	Yes!	0 + 15 Summary
Unconstrained Trust	Yes!	0 + 0 Summary
Unconstrained, but all unexpired observed paths Revoked	Unknown	45
Unconstrained, but zero unexpired observed paths	Unknown	1653
Expired	No	4398
Technically Constrained (Trusted)	Maybe soon?	56
Technically Constrained (Other)	No	39
Disclosed as Revoked, but Expired	Already disclosed	143
Disclosed as Revoked and in OneCRL	Already disclosed	560
Disclosed as Revoked (but not in OneCRL)	Already disclosed	20
Disclosed as Parent Revoked (so not in OneCRL)	Already disclosed	127
Disclosed, but Expired	Already disclosed	397
Disclosed, but zero unexpired observed paths	Already disclosed	500
Disclosed (as Not Revoked), but in OneCRL	Already disclosed	30
Disclosed, but Technically Constrained	Already disclosed	218
Disclosed, but with Errors	Already disclosed	0
Disclosed (as Not Revoked), but Revoked via CRL	Already disclosed	0
Disclosed (as Not Revoked) and "Unrevoked" from CRL	Already disclosed	1
Disclosed	Already disclosed	3203
Unknown to crt.sh or Incorrectly Encoded	Already disclosed	25

OCSP Stapling

In an attempt to fix OCSP, an extension to TLS was added known as OCSP stapling was implemented. OCSP stapling means that a web server queries an OCSP responder for its own certificate status and sends it to the client when they connect to a given secure service.

This prevents a client's IP from leaking to the certificate authority. As OCSP responses are issued from a special intermediate certificate, they also can not be trivially forged. All major web servers today support OCSP stapling.

However, only IIS enables it by default as of writing.



OCSP-Must-Staple

As an additional security measure, it is possible for a leaf certificate to specify Must-Staple. This, unfortunately, is not a default setting of any CA I'm aware of, and requires special effort on the creation of a Certificate Signing Request.

OCSP-Must-Staple is a restriction on a leaf certificate; it specifies that a certificate is only valid if it is used in a TLS stapled response; or in other words; the TLS handshake must have OCSP information in it, and it must be a valid OCSP response that certifies a certificate is valid.



Expect-CT

In addition, an extension to HTTP allows a web server to tell the browser that a certificate must contain certificate transparency log information.

Expect-CT can, for a browser, allow a leaf certificate to only be accepted if that certificate is publicly logged and auditable, which at a minimum makes discovering MITN attacks from rogue intermediates possible.


Expect-CT headers are cached the browser similar to HSTS, and thus operate on a Trust On First Use policy. While less than ideal, it can prevent a man-in-the-middle attack from succeeding.



Key Pinning

In some scenarios, it is also possible to use a technique known as key pinning to prevent unknown certificates from being used in place of proper ones. For example, Google Chrome hardcodes in the public key identifiers of google.com domains.

DANE, and HPKP that can do this for normal websites. However, DANE itself has technical issues as it's dependent on DNSSEC and is not implemented in any major browser. HPKP on the other hand, has been removed from Chrome, and is dangerous to deploy for browsers that do support it.



Conclusions

WebPKI is an extremely complicated beast, and this slide deck is only the tip of the iceberg.

Currently, WebPKI's greatest weakness is the fact that any unrestrained certificate can sign for any domains. While there are some technical constraints possible, they're not wildly deployed on the internet.

However, with the advent of certificate transparency, greater insight is possible into the world of WebPKI and the health of the ecosystem.



Conclusions

Although revocation as it exists today for WebPKI is essentially broken, as OCSP stapling and OCSP-Must-Staple become more commonplace, we will hopefully soon reach a world where a certificate, once revoked, becomes properly useless.

Furthermore, ongoing development on DANE (involving DNSSEC stapling) will hopefully bring a secure and working key pinning system available to all, and not just a small majority of high profile websites.



Conclusions

Furthermore, certificate authorities are (unfortunately) the only method of realistic trust that can be used. With the additional insight and community auditing brought on by certificate transparency, there is hope that the CA system can be brought to a point that it is able to provide truly robust and secure authentication.



Questions/Contact?

Twitter: @FOSSFirefighter

Github: <https://github.com/ncommander>

EMail: michael@casadevall.pro

