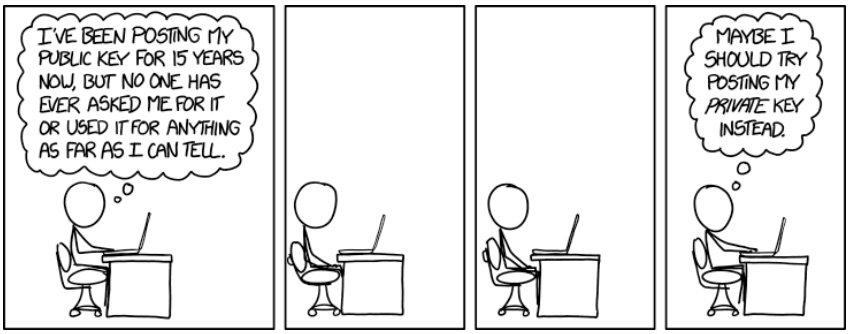
Cryptology (9)

Digital Certificates and Public Key Infrastructure (PKI)

John York, Blue Ridge Community College
Weyers Cave, VA
http://www.brcc.edu

Obligatory XKCD Cartoon



"I guess I should be signing stuff, but I've never been sure what to sign. Maybe if I post my private key, I can crowdsource my decisions about what to sign."
https://imgs.xkcd.com/comics/public_key.png

Digital Certificate—Basics

- A digital certificate distributes a public key
- It contains (among other things)
  - Public key
  - Key valid dates
  - Subject, Common Name (CN) and optional Subject Alternative Name (SAN)
  - Usage: Web server, Signature, Code signing, …
  - Signature of Issuer
  - Algorithms used
- Certificate content is based on X509 standard

The purpose of a digital certificate is to distribute a public key, and bind that key to the user that owns it. In RSA and Diffie-Hellman both, we saw that if a Man in the Middle (MITM) attacker could substitute his public key for the public keys of Alice and Bob, he could break all of their encryption. When Alice encrypts data for Bob using Bob's public key, it is essential that the key she uses is really Bob's and not that of an attacker. Therefore, we need a secure way to distribute public keys.

This is usually done with a trusted party that operates a Certificate Authority (CA). The CA creates a certificate for Alice by merging her public key with other information (her name, validity period, etc.) and then signing the result with the CA's private key. Alice then gives her certificate to Bob. If Bob can verify the certificate with the CA's public key (from the CA's own certificate) then Bob knows he has the correct certificate and public key for Alice. Of course this assumes that the CA has not been compromised, and that Bob received the CA's certificate in a secure manner--there's a chicken and egg problem here.

Key Distribution Problem

- Problem: How do we know that the public key really belongs to our friend, and not to an attacker/Man in the Middle (MITM)?
- If the public key is from a MITM, then all communications are compromised
- Solutions
  - Out of band: get key directly from friend, use other means (flash drive, paper and pencil, etc.)
  - Trusted Third Party
    - Private Certificate Authority (CA) generates certificates
    - Public CA (sort of a solution) generates certificates
  - Self-signed certificate

Digital certificates are an attempt to solve the key distribution problem: How do Alice and Bob get copies of each other's public keys, and not those of an attacker instead?

The simplest method is out-of-band. Alice and Bob meet in person and give each other their private keys on flash drives, on paper, or by whispering the keys to each other. The problem is that this method does not scale well beyond a few people. If thousands or millions of people need to exchange public keys, this will break down.

The most common method is to use a trusted third party (CA). Users give their information and public keys to the CA, the CA verifies their identity and then gives them a certificate. The certificate is signed with the CA's private key, so any other users can verify the certificate with the CA's public key.

Another way is to sign your own certificate. This will allow you do use certificate based protocols like TLS and SSH, but it totally skips the step "verifies their identity." If you are communicating with someone who uses a self signed certificate (say on their HTTPS web site), the burden of verifying their identity falls upon you. Self signed certificates are very common in lab environments where you are testing things like HTTPS servers.

Private CA (Certificate Authority)

- Most large organizations have their own CA
- Organization creates keys for users, computers, and software
- Organization tracks certificates, revokes old certs
- Advantage
  - Control—organization has complete control of process
  - Only people/computers trusted by company have certs
  - Computers in a Windows domain automatically trust domain server cert
- Disadvantage
  - Lots of work (see control, above)
  - User computers must install/trust organization's certificate

A large organization can maintain their own CA, and use it for all communications with and between its members. Virginia Tech does this. Students, faculty, and staff are all issued digital certificates which they use when accessing Va Tech resources, when emailing, etc.

The private CA is probably the most secure option. The organization can control who its members are, and what resources they can access with their certificates.

It takes effort to create and maintain a CA and associated systems. Someone has to enter users in the system, and remove them when they are no longer associated with the organization. You have to run periodic audits to make sure accounts are only for current members. You have to configure the resources (web sites, shared drives, whatever) to accept the certificates.

Public CA

- Public CA issues certificates
- CA is supposed to evaluate applications
  - Only issue certificates to legitimate companies
  - Verifies company owns the resource (web site, etc.)
- Extended Validation (EV) Certificate
  - CA charges more to do what it is supposed to do, above
  - Browsers used to display green icon for EV certs, but no more
- Advantage
  - Browsers already have CA cert installed
- Disadvantage
  - Cost
  - Untrustworthy CAs

The most common method is the Public CA. Examples are Comodo, GoDaddy, … Years ago, a reputable CA would verify that you own the domain you want the cert for, verify that you were an established company (Dunn and Bradstreet rating, etc.) and verify that you are authorized to represent the company with a letter from the CEO/President on company stationary. Now, they may verify that you own the domain.

Extended Validation (EV) certificates were established after many CAs issued certs without bothering to check anything other than domain registration. You pay extra for an EV cert, and then the CA does verify your information. For a while, browers would display a nice green icon for sites that had EV certs, but that stopped in the fall of 2018. https://www.troyhunt.com/extended-validation-certificates-are-dead/

The advantage of using a public CA is that your users' browsers will (most likely) automatically trust your cert if you use a public CA cert on your website. Otherwise you have to find a secure way of distributing your cert to your users. (You do not want your users clicking to accept just any cert they see.)

Usually you have to pay for public certificates, but https://letsencrypt.org/ does have

them available for free.

Public CA Problems

- CA does not evaluate applications for certificates
  - Financial incentive is to sell certs
- CA is compromised
  - Diginotar compromised by Iran in July 2011, Iran attacked Google users in Iran
  - Commodo compromised by Iran March 2011
  - Stuxnet used valid certs from JMicron and RealTek
- Certificate Transparency
  - logs of certificate creation are publicly auditable, detect cert abuse
- Please read:

http://resources.infosecinstitute.com/cybercrime-exploits-digital-certificates/

This site has an excellent overview of problems with public certificates. Read it. https://resources.infosecinstitute.com/cybercrime-exploits-digital-certificates/

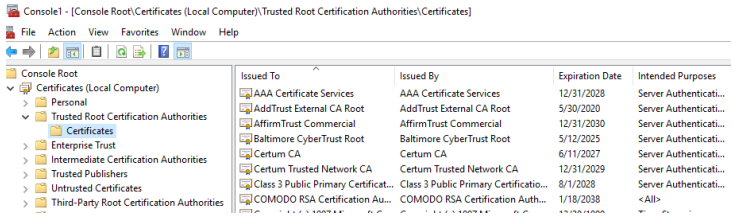
The public CA system is fraught with problems. Attackers can easily register DNS domains and obtain certificates for them, so that victim's browsers will allow the attackers sites to distribute malware over HTTPS. CAs have been compromised, and then used to generate certificates that were used by nation-state attackers (Iran and Diginotar, the US and Stuxnet.)

There aren't many fixes to the public CA system, other than users being careful. The Google Certificate Transparency Project https://www.certificate-transparency.org/ is making auditing logs publicly available so that anyone can determine what certificates are in use on the web and when they were issued. This allows security analysts to track certificates, detect certificates that have been improperly issued for your organization, or block certificates that are involved in malware or that have just been issued. You can search certificate transparency logs at https://transparencyreport.google.com/https/certificates?hl=en

Most improvements in SSL tracking are only useable by security analysts at this time.

Windows Certificate Store

- By default, browser and OS vendors decide which CA's your computer trusts
- Windows has a store of trusted CA's
  - Chrome uses Windows' certificate store
  - Firefox maintains its own store
- Removing compromised CA from Windows certificate store
  - Manually using MMC, or PowerShell
  - Automatically in Windows Updates



You may ask, "How do I decide which CAs to trust and which ones not to trust?" By default, your OS and browser vendors do that for you. Microsoft keeps a list of approved CAs in its certificate store. You can access the certificate store using the Microsoft Management Console (MMC) mmc.exe. You can use File > Add/Remove Snap-in… to load the Certificate store. The CAs you trust are in the Local Computer section, under Trusted Root Certification Authorities > Certificates. There are a lot of them.

In PowerShell there is a certificate provider, which means you can access the certificate store in the same way you would access a hard drive.

```
cd cert:          #change to the certificate provider, cd is alias for Set-Location
dir               #shows top level, CurrentUser (certs for the user) and LocalMachine
cd localmachine   #change to the computer's cert store
dir               #shows headings, the one we want is root
Cd root           #this is where the Trusted Root Certification Authorities  are
dir               #show the certs we trust
dir Cert:\LocalMachine\Root\  #do all of the above in one line
```

Note: Firefox keeps its own store separate from Windows. Chrome uses the windows store.

Self-signed Certificate

- Sometimes called “Snake oil certificates”
- Easily created using openssl https://www.openssl.org/
- Useful for testing and lab use
- Bad for production use
  - Teaches users to trust invalid certificates
  - Easily forged by attackers

Self-signed certificates should never be used in production, as they teach users to click through certificate warnings. Additionally, as browsers work harder to ensure that insecure certificates are blocked, it will become harder and harder for your users to get past the warnings.

It is entirely reasonable to use self-signed certificates in test and lab environments, however. You can easily create certificates with openssl, and then practice with HTTPS or SSL certificates in your test lab.

If you use openssl to generate a certificate, and only use the defaults, the Organization Name will be Internet Widgets Pty Ltd. It has been used so often by attackers that don’t bother to change it that Snort has a rule that detects certificates for Internet Widgets Pty Ltd.

Public Key Infrastructure (PKI)

- PKI includes CA plus functions needed to administer certs
- Registration Authority determines if request for a certificate is valid
- Root CA
  - Original CA, holds cert for entire organization (trust anchor)
  - Often powered down, locked in a safe except when needed to sign intermediate CA cert
- Intermediate CA
  - Does work of issuing and validating certs. If compromised, its cert is revoked by Root CA and reissued for new Intermediate CA

The core of a Public Key Infrastructure (PKI) is the CA. When you add the supporting applications to accept applications for certificates, revoke certificates, and audit accounts associated with certificates, you have a PKI.

The Root CA is the top of your tree if you have several CAs. It has to have a self-signed certificate, since it is the root. All CAs under it will have certificates signed by the root or other subordinate CAs. If the Root CA is lost or compromised, the entire certificate structure will have to be rebuilt. Therefore it is critical that the Root CA be protected. Often the Root CA is offline, or unconnected to the network so it cannot be attacked. If an Intermediate CA is compromised, the Root CA can be used (via USB flash drive, if necessary) to revoke the cert for the compromised CA and issue a cert for a new Intermediate CA. That way you don't have to rebuild everything from scratch.
https://security.stackexchange.com/questions/128779/why-is-it-more-secure-to-use-intermediate-ca-certificates

Revoke Certificates

- CA must be able to revoke certificates before they expire
- Examples
  - User leaves organization
  - Server name changes, needs new cert
  - Discover cert was improperly issued
- Certificate Revocation List (CRL)
  - Server maintains list of revoked certs
  - Browsers/users check CRL before accepting cert
- Online Certificate Status Protocol (OCSP)
  - Newer method to check cert validity (not revoked, etc.)
  - Less bandwidth than CRL

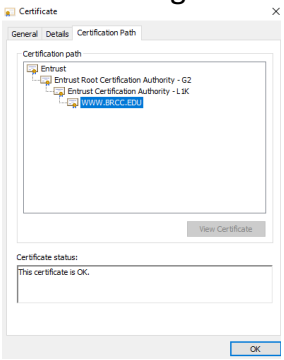
At most organizations people and computers come and go. When a person or computer leaves, their certificates must be revoked just as their accounts must be disabled. The same is true for web servers. The site name may change, or the content may move to a different site, which requires a new certificate. The old certificate needs to be revoked so that attackers cannot attempt to use it improperly.

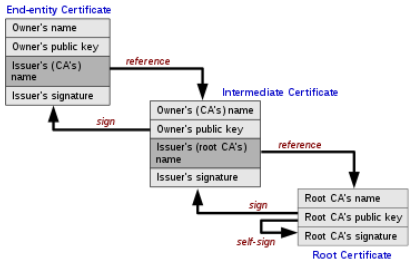
The oldest means of checking for revoked certificates is the Certificate Revocation List (CRL). Most certificates include a field called CRL Distribution Points, which gives web site URLs where browsers can download the CRL for the CA that issued the certificate. They compare the certificate to the CRL and reject the certificate if it has been revoked. A large CA may have thousands of certs that have been revoked but haven't expired yet, so the CRL can get quite large.

The Online Certificate Status Protocol (OSCP) allows the CA to put up a site that supports OSCP. The Authority Information Access field in the certificate lists the OCSP servers for the CA that issued the certificate. The browser then contacts the OCSP server and asks if that certificate is still valid.

Certificate Path

- Path from Root CA to certificate, including all intermediates
- Root CA signs first intermediate, …, last intermediate signs cert





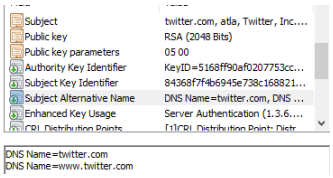
https://en.wikipedia.org/w/index.php?title=File:Chain_of_trust.svg&lang=en

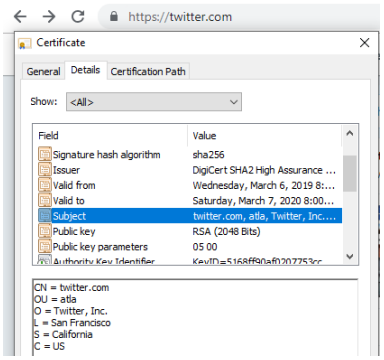
The diagram on the left shows the Certificate path tab in the Microsoft certificate window. The certificate is at the bottom, and is signed by the certificate just above it. It is common for commercial CAs to have several intermediate CAs, so they form a chain. Each certificate is signed by the CA above it, until you final reach the root CA. The root CA should be in your OS or browser Trusted Root CA list if the cert is to be considered valid.

The diagram on the right shows the same information, but with more detail. Unfortunately, the root is at the bottom right and the original cert is at the top left, so it is different from the left diagram. It shows that each cert has an entry with the name of the CA that signed the cert. The browser goes to that CA, grabs its public key, and then uses the public key to check that the cert is valid. The browser continues to work its way up the chain until it reaches the root. All the intermediate CAs and the cert itself should be properly signed by the CA above it.

Browser certificate checks--Name

- Name in browser navigation bar
  - Ex: browsing to www.twitter.com
  - Either the Subject Common Name (CN) or one of the Subject Alternative Names
  - (SAN) must match





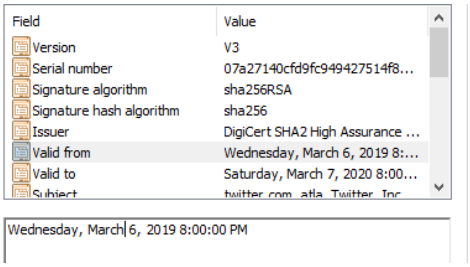
When you browse to an HTTPS site, the browser makes several checks before it accepts the site's certificate. This covers some, but not all, of the checks.

The biggest check is that the name on certificate must match the DNS name in the browser bar. If the site is https://twitter.com, either the Common Name (CN) in the Subject or one of the Subject Alternative Names (SAN) must be twitter.com. Note that www.twitter.com, twitter.com, and mail.twitter.com (if there is one) are all different, and must appear exactly that way in the certificate.

The exception is the wild card certificate. If a SAN is *.twitter.com, any xxx.twitter.com will be accepted.

Browser checks--Valid Dates and Revocation

- The current date must match the validity dates

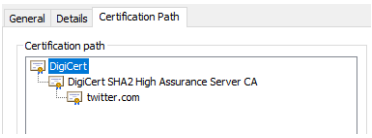


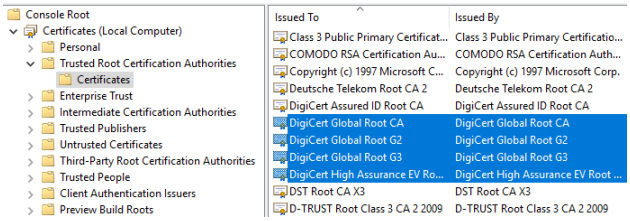
- The browser "should" check either the CA's OCSP or CRL to be sure the certificate has not been revoked.

One of the most common certificate failures happens when the site manager does not renew the certificate on time. This is bad, because it encourages people that use the site to click through the certificate warnings.

Browser Checks--Trusted Root CA

- The certificate for the Root CA at the end of the chain of CAs that issued the cert must be trusted by the browser or OS



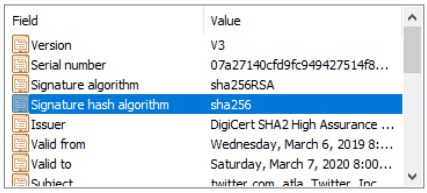


This is the check that self-signed certificates fail. The Root CA of the chain of CAs that issued the cert must be trusted by the browser or operating system.

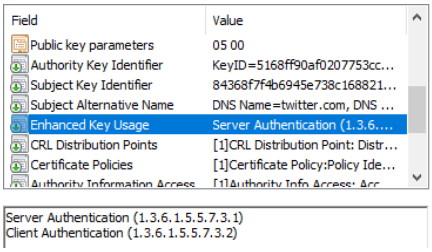
If a person wants to use a site that has a self-signed certificate, they should verify that they are indeed at the correct site (not an attacker's) and then trust the certificate. A more secure way is to obtain a copy of the self-signed certificate in a secure fashion and install the certificate into their store. That way if an attacker creates a certificate with the same name, the browser will reject it because the public key, thumbprint, and other fields will be different from the certificate in its store. The web site signs content with its private key and the browser decrypts the content with the public key. A fake will not have the correct private key and will not work.

Other Browser Checks

- Hash algorithm SHA-1 no longer acceptable



- Key Usage
  - This one can be used to identify a web site but not to sign code, for example



When Google broke SHA-1, most of the certificates in force at the time used the SHA-1 hash. Web sites had to obtain new certificates very quickly, as most common browsers began to refuse certificates based on SHA-1 hashes.

Certificates also specify what they may be used for. The most common certificate is used to authenticate clients. Another common certificate use is code signing, which attempts to prove that the signed application is legitimate and not malware.

Windows PKI

- PKI can be tightly integrated into an Active Directory Domain
- Windows server can be configured as CA
  - Best if no other functions installed on CA for security and ease of upgrade
- If Windows Domain Controller trusts CA, so do computers in domain
- Group Policy can be configured to automatically create certs
  - New computers
  - New users
- Certs can be tailored by need/function
  - For example, computer RDP certs

Windows comes with a built-in PKI, if the administrators choose to implement it. They need to install a server with the Active Directory Certificate Services role. Normally this server is used for no other purpose. The ADCS, or Windows CA server is trusted by all computers in the Windows domain, assuming it is properly installed. By default, all domain computers receive certificates that are used behind the scenes as they are managed by Active Directory (AD). Administrators can use Group Policy to configure AD to automatically issue certs to user accounts, and control the purposes the certs can be used for.

Certificates can also be manually created on the Windows CA. For example, if an administrator creates a web server and configures it with a cert issued by the Windows CA, all computers in the domain will automatically trust the server. However, if users or computers outside the domain browse to the server, they will have to manually trust the Windows CA in order to trust the site. Windows CA is a private CA, so computers outside the domain won't trust its certificates.

A Windows web server can certainly have a public certificate purchased and installed, so it is trusted by non-domain computers.