X509 certificates and Certification authorities

Tufa Alexandru Daniel 1628927

24 November 2017

1 Introduction

X.509 is a standard that defines the format of public key certificates. It has been standardized by the IETF with RFC 5280 [1] and offers two distinct capabilities:

1. An X.509 certificate acts as a container for the public key that may be used to verify or validate an end entity such as a web site.
2. An X.509 certificate is digitally signed by a trusted Certificate Authority, CA, to attest that the public key of the end user truly belongs to him.

A Certification Authority is an entity or organization which signs certificates. We can have different types of authorities such as root authorities that generate root certificates or intermediate authorities that generate intermediate certificates.

2 Certificate types and format

There are many types of X.509 certificates, mainly because of marketing purposes. They are designed to offer differing price/functionality points and we can classify them as:

- **Root certificates**: the *issuer* and *subject* attributes are the same and typically this kind of certificate is signed by the topmost authority in the chain.
- **Intermediate certificates**: any certificate which is not signed by a root CA. These certificates form a chain and there can be many of them between the root certificate and the end user.

- **Qualified certificates**: personal certificates that reference the European Directive on Electronic Signature which is focussed on uniformly defining an individual for the purposes of the digital signature, authorization or authentication.
- **non-Qualified certificates**: a personal certificate which doesn't conform to the standard defined for Qualified certificates
- **End-Entity Certificate**: the private key is used to secure the end-entity and not to sign certificates. It's not an Intermediate certificate nor a root certificate and is considered the last certificate in the chain of certificates.
- **Multi-host certificates**: it's a certificate typically assigned to a server that contains a CN=hostname attribute in the *subject*.
- **Multi-domain certificates**
- **Wildcard certificates**: the *subject* attribute contains CN=*.example.com, where * is the wildcard.
- **Extended Certificates**: include a mixture of improved validations as well as technical processes. They're used to provide increased confidence to end users, though the standard explicitly states that it does not guarantee the business practices of the certificate owner - merely that the owner does exist. They are recognised by the use of a registered OID (unique for each CA) in the CertificatePolicies attributes.
- **Domain Validation certificates**: only the ownership of the domain name by the certificate requestor has been verified by the CA
- **Organizational Validation certificates**: ownership of the domain name of the certificate requestor and its organizational details have been verified by the CA.

We will analyze the format of a X.509 certificate while also using as an example the information shown by www.apple.com when requested the certificate from Safari on MacOS and from Google Chrome on an Android device.
The X.509 certificate definition is written in ASN.1 and looks like this:

```
TBSCertificate  ::=  SEQUENCE  {
      version         [0]  Version DEFAULT v1,
      serialNumber         CertificateSerialNumber,
      signature            AlgorithmIdentifier{SIGNATURE-ALGORITHM,
                                {SignatureAlgorithms}},
      issuer               Name,
      validity             Validity,
      subject              Name,
      subjectPublicKeyInfo SubjectPublicKeyInfo,
      ... ,
      [[2:               -- If present, version MUST be v2
      issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
      subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL
      ]],
      [[3:               -- If present, version MUST be v3 --
      extensions      [3]  Extensions{{CertExtensions}} OPTIONAL
      ]], ... }
```

The X.509 certificate consists of the following attributes:

- **Version**: in general this should indicate version 3. This is the case for the Safari version while absent on the Android one.
- **Serial Number**: positive number up to a maximum of 20 octets. Defines a unique serial number for each certificate issued by a particular Certification Authority. This value is identical in both versions, 18::13::80::9C::1A::F3::B2::AB::A8::9A::DB::37::4B::D4::D1::80.
- **Signature and SignatureAlgorithm**: the algorithm used to sign the certificate identified by its OID and any associated parameters. While absent on the Android certificate on the Safari one we can see that SHA-256 with RSA Encryption (1.2.840.113549.1.1.11) has been used.
- **SignatureValue**: bit string containing the digital signature. Absent on the Android version while present on Safari.
- **Fingerprints**: the unique identifier of the certificate. In this case 2 fingerprints were given using SHA1 and SHA256 equal and present in both versions.
- **Issuer**: the DN (Distinguished Name) of the Authority that signed and therefore issued the certificate. Present in both versions we acknowledge that the organization is Symantec Corporation.

- **Validity**: provides two sub-fields *notBefore* and *notAfter* which define the time period for which the certificate is valid. Present in both versions we see that this certificate is not valid before 2 October 2017 and after 16 October 2019.
- **Subject**: A DN defining the entity associated with this certificate. If this is a root certificate then issuer and subject will be the same DN. In this case the common name is www.apple.com and is present on both versions.
- **Public key**: highlights both the algorithm used and the entity's public key as a bit string. Present on the Safari version while absent on the Android one.
- **Extensions**: additional information that is present only on the Safari version like *Basic Constraints*, a boolean which specifies wheter the certificate is a CA or root certificate, *KeyUsage* that specifies the usage of the key, for example Digital Signature, and *CertificatePolicies* that is used to identity the particular policies of the issuer CA.

Another particular feature that we can observe on both versions is the chain of trusted Certificate Authorities.

3 Certificate Authorities

Usually there isn't only one Certificate Authority that offers certificates that are trusted, but a certificate chain is formed. A certificate chain is a list of certificates that usually starts from an end-entity certificate followed by one or more CA certificates. The *Issuer* of each certificate matches the *Subject* of the next certificate in the list. This is not the case when the CA is the topmost one that signs itself. This can be done only when some standard has been passed and this certificate is called a trust anchor[1]. These root certificates are installed directly in the machines by means of the OS. For example on MacOS we can find them in the *keyChain Access* in the Certificates category. You can view the root certificates of iOS easily on the web [2] while Mozilla offers directly an HTML or CSV representation of the trusted CAs [3]. The intermediate certificates are supposed to be signed by the secret key corresponding to the next certificate in the chain. This is how a chain of trust is formed and we can view it in Safari as follows:

[1]An authoritative entity for which trust is assumed and not derived

- Intermediate certificate authority is represented by Symantec Corporation
- Root certificate authority represented by VeriSign Trust Network. As we mentioned before the *Subject* and *Issuer* attributes are identical.

References

[1] Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. http://www.zytrax.com/tech/survival/ssl.htmlx509-overview.

[2] ios 11 root certificates. https://support.apple.com/en-us/HT208125.

[3] Mozilla ca certificate list. https://wiki.mozilla.org/CA/Included$_{C}ertificates$.