Challenges of Implementing and Managing Public Key Infrastructure

Arun Paul

Abstract—A public key infrastructure (PKI) provides the framework necessary for the creation, storage and revocation of digital certificates. Digital certificates are the crucial constituent of SSL (Secure Socket Layer) and TLS (Transport Layer Security) protocols which are pivotal in securing communication across the Internet. The PKI is a mechanism that binds a Public key with the corresponding identity of the organization or person who owns it. Hence validating the certificate before entrusting the same to secure the communication is an inevitable phase.

When it comes to using development of web-browsers, PKI being a framework does not come with a specified standard of how it should be implemented in products and services. Also the developers may prioritize on performance and other factors of a web-browser over its security best practices. In this study, I have executed a set of experiments with the set of most commonly used web-browser and operating systems and compared their behavior in dealing with invalid and revoked certificates. The various standards and practices that are followed by different browser companies and their effectiveness in real-world is also compared, so as to measure the level of security provided by the browser software in various deployment scenarios. I have also looked into the recommendations towards the best implementations as of today and directions for the future for a practical and standardized certificate validation method for the future.

Index Terms—PKI, SSL, TLS, CA, CRL, OSCP, CRLSet, OneCRL, OpenSSL

I. INTRODUCTION

The internet has evolved and grown exponentially in the last many years and so have the threats and associated risks that affects the trust between two parties who communicate over an untrusted network. The identity and integrity of two communicating parties over the internet boils down to the Digital certificate which is issued by a trusted third party, which is a Certificate Authority. It is essential that both the parties trust the common Certificate Authority who attests the identity of the individual or company on to the Digital certificate associated with it. With increasing attacks on Certificate Authorities or its associated Subordinate Certificate Authorities, the reason to revoke a certificate. Every x.509 certificate comes with its own certificate revocation information that has to be verified before using the certificate to trust and build an encrypted connection. The first and default method of revocation is CRL or Certificate Revocation List. Due to the inherent latency and delay in CRL mechanisms, a more efficient revocation method known as OCSP or Online Certificate Status Protocol was developed. Again the vendors, which in our case are the web browser companies can come up with their own proprietary revocation routines. However the issue is that every vendor is free to deploy the PKI framework based on their priorities and hence there are no standards followed in the implementation. This paper compares the various browsers for their vendor implementations of certificate revocation mechanisms and its effectiveness.

II. THE PKI FRAMEWORK

Certificate Authorities in PKI binds the public key with the corresponding identity of the person or organization owning the same. The client, (which can be a web-browser), can trust certificate of a Server only if the client trust in the CA who has signed the certificate. Hence it is an extension of the trust that the client has on the CA to the owner of the certificate, which was validated and signed by the CA.

![PKI Framework](image)

Fig 1. PKI Framework

A. The functioning PKI

The working of PKI is defined in the RFC5280. It defines the requester as the “End Entity” who falls in the PKI user category. The fundamental purpose of PKI is to make a trusted Certificate Authority sign a Certificate that can bind the identity of the person or company to the certificate vouched by the CA, as listed in Figure 1.

The functioning of a typical PKI framework is explained in the below steps:

i. The Entity who need to be associated with a trusted signed certificate creates his own key pair. Refer Figure 2

ii. The Entity creates a CSR or a Certificate Signing Request that contains all the information about the identity of the Entity. These includes “Common Name”, “Organization”,
“Organization Unit”, “City”, “State”, “Country”, “email”, and
the “Public Key” that will be associated to the certificate.

iii. The CSR is then self-signed by the “Entity” by own keys

iv. The signed CSR is then sent as a Request to the RA
(Registration Authority) which is one of the Subordinate CA in
the PKI chain of trust.

v. The CA offers a service to the “Entity” for verifying its
identity as detailed in the CSR, and charges the “Entity”
accordingly. The CA can also go for EV or Extended
Validation, which is explained in detail in this document. The
validation process consist of verifying the “whois” domain and
other information contained in the CSR. This can be either over
Email or Phone (mostly depreciated method). The validation
process is automated these days.

vi. Once the validation of the identity information is
completed, the CA signs the CSR with its private key and the
output of this process is the signed Certificate, which binds the
Identity of the “Entity” with the contents of the Certificate and
a Public Key, which will be used to secure the communication
across the internet. The signed certificate can be in different
formats. The X.509 digital certificate of base64 ASCII encoding
can be of the following types:
1. *.crt
2. *.cer
3. *.pem
4. *.ca-bundle
5. *.p7s
6. *.p7b
7. *.der

There can also be bundled format which can store multiple
certificates into a single cryptographic file, of *.pfx format.

vii. Once the Certificate is signed, the CA informs the
“Entity” with a Notification for Download for the generated
signed certificate.

viii. The “Entity” shall download its own Certificate and also
the “Chain of Trust” certificates, which includes all the CA
from the VA who signed the certificate, all the way to the root
CA.

ix. The “Entity” shall install the newly signed certificate in the
Server Operating system, or an application that requires the use
of certificate for a secure communication. E.g. Apache2
software.

Once the PKI framework is established, the entity can use the
signed certificate and associate public key to secure the
communication across the internet.

B. Why Subordinate CA to sign the certificate?

The integrity of entire PKI system depends on the integrity of
the Root Certificate Authority, which in turn can be traced back
to the integrity of its Private Key. Root CAs in turn assign
Subordinate CAs who are also known as VA or Verification
Authority, because the VA are the entities who extensively
evaluate the contents of a CSR or Certificate Signing Request,
with the identity of the requester before signing them. In a
scenario where one of the Subordinate CA is compromised, the
Root CA can still be preserved and only the particular branch of
Subordinate CA need to be revoked.

C. SSL/TLS Certificate

The SSL/TLS certificate is what binds the Identity of the
“Entity” with the description, as in the certificate. The validation
of the certificate at the client side is one of the most important
part of the entire PKI process. The validation process consists of
tells for the following –
1. Expiration date
2. The validity of the certificate
3. Match the subject name to hostname as in certificate
4. Check that the certificate is not Self-Signed
5. Check for revocation of the certificate through CRL or
   OCSP
6. Validation of the entire certificate chain from the VA to the
   root CA.

The client can optionally have additional checks for the set
standard for key length and support of legacy or insecure
algorithms.

D. SSL and TLS protocols

SSL or Secure Sockets Layer and TLS or Transport Layer
Security are the two protocols which provides end to end secure
communication using PKI infrastructure. The SSL v2 was the
first public release by Netscape. Later on SSL v3 was released to
fix one of the biggest vulnerability in SSL, which was exposed
by POODLE attack.

The first version of TLS which was the successor SSL was
version 1.0, which was primarily made for compatibility with
SSL versions. The TLS version 1.1 and older SSL versions are
considered vulnerable and are not advised to be used in secure
infrastructure. The current TLS versions are 1.2 and 1.3. [7]

E. EV – Extended Validation

Extended validation of EV is marketed as a symbol of trust on
the internet. This sells for sites which deal directly with the
customers like ecommerce sites. The Certificate Authorities
charge the “Entity” extra for an EV certificate, as in Figure 3.
There are the following additional requirements for an EV
validation compared to a normal certificate validation:
a. Establish the legal identity + operational + physical presence of website owner
b. Establish that the applicant is the domain name owner
c. Confirm the identity and authority of the individuals acting for the website owner

III. THE INTEGRITY OF CERTIFICATE AUTHORITIES

The significance of certificate revocation has become more important because of the recent hacks on Certificate Authorities or Resellers of CA services. One such significant hack of Certificate Authority was the hacking and fraudulent domain creation at one of the Comodo Resellers. Comodo is one of the biggest CA infrastructure in the internet. The hacking of a CA or its subordinate questions the very integrity of the PKI infrastructure, which is based on the trust over the Certifying Authority. Once a CA is hacked, the attackers can create fraudulent domains, which will be used for launching much serious cyber-attacks. Also the attackers can create Wild-card certificates which can be used to hack an entire sub domain. E.g. DigiNotar, a prominent CA was hacked and the hackers were able to successfully generate wild card certificates for *.google.com. The hacking of Comodo or its resellers has happened multiple times. One of the biggest breach was using a script-kiddie tool known as SSLSNIFF, which was based on IE’s (Internet Explorer) vulnerability. The analysis of the attack had shown that the referral links pointed to the site that provided SSLSNIFF tool. However the Comodo Company expected the attack to be an APT (Advanced Persistent Threat), when it was nothing more than a work or a novice script kiddie.

Another reason for certificate revocation is the discovery of vulnerabilities in the existing security protocols. Heartbleed is one such example which exposed the underlying vulnerability of SSL protocol. Such incidents calls for the recall of certificates, before they are expired.

As an end user, one can choose not to trust a CA like Comodo, by removing it from the trusted list of CA, pre-installed in the web browser. However this would lead to a huge blackout of the internet sites which are signed by such CA. The next section details about the current market share of various CA based on the domains certified by them. A W3 Tech survey from the year 2018 show that IdenTrust has overtaken Comodo for the highest Market share as of date.

IV. CA MARKET SHARE

The market share for CA has changed radically in the last many years. This section discusses the updated market share of the top Certificate Authorities in the market as of today. Please refer Figure 4.

Note - I have not considered the CA who hold less than 3% of the current market share and have considered only the top players in this domain.

It is very evident that blacklisting a root CA in a browser will result in a significant impact on the internet accessibility as the top 95% of the domains are held by just five CA or its resellers. This provides more reasons to ensure that certificate revocation mechanism are working well and serves its purpose. https://en.wikipedia.org/wiki/Certificate_authority

V. BROWSER MARKET SHARE

Browser market share

A study into the Web Browsers products for their real world user acceptance numbers were conducted, to find the top web browsers that need to be considered for my tests on certificate revocation mechanism. Please refer Table 1.

<table>
<thead>
<tr>
<th>Browser</th>
<th>StatCounter</th>
<th>NetMarketShare</th>
<th>Wikimedia</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>61.75%</td>
<td>63.50%</td>
<td>44.90%</td>
<td>56.72%</td>
</tr>
<tr>
<td>Safari</td>
<td>15.12%</td>
<td>19.03%</td>
<td>23.00%</td>
<td>19.05%</td>
</tr>
<tr>
<td>Firefox</td>
<td>4.92%</td>
<td>4.56%</td>
<td>5.80%</td>
<td>5.09%</td>
</tr>
<tr>
<td>UC</td>
<td>4.22%</td>
<td>1.01%</td>
<td>0.40%</td>
<td>1.88%</td>
</tr>
<tr>
<td>Opera</td>
<td>3.15%</td>
<td>1.44%</td>
<td>1.20%</td>
<td>1.93%</td>
</tr>
<tr>
<td>IE</td>
<td>2.80%</td>
<td>3.98%</td>
<td>6.60%</td>
<td>4.46%</td>
</tr>
<tr>
<td>Edge</td>
<td>2.15%</td>
<td>1.77%</td>
<td>1.90%</td>
<td>1.94%</td>
</tr>
<tr>
<td>AOSP</td>
<td>1.18%</td>
<td>0.86%</td>
<td>0.70%</td>
<td>0.91%</td>
</tr>
<tr>
<td>Others</td>
<td>1.98%</td>
<td>3.87%</td>
<td>12.30%</td>
<td>4.76%</td>
</tr>
</tbody>
</table>

Table 1. Browser market share

The data from three different sources – StatCounter, NetMarketShare, Wikimedia were combined and averaged to find a more accurate number, which I have plotted in Figure 5.
It is very evident that Chrome, Safari and Firefox together constitutes almost 85% of the entire web-browser market. IE (Internet Explorer) gets only 4% of the entire market, although it comes pre-bundled with every Microsoft products. For my testbed, I have considered the following browsers:

i. Chrome
ii. Firefox
iii. IE
Which were tested across both Windows and Ubuntu platforms.

VI. CERTIFICATE REVOCATION

A. Revocation method – CRL

CRL or Certificate Revocation List is the original certificate revocation method introduced in PKI. The CRLs are usually issued by the issuer of the certificate. This list contains the serial numbers of the certificates which could be revoked due to one of those many reasons discussed earlier in this document. There are other information embedded in the CRL, which includes the revocation date, the reason for revocation and more details about the revoked certificate. The CRL list comes with few global attributes that includes the version, the issuer name, the date of issuing the CRL, the details about signature algorithm used.

The CRL list is published in a publicly accessible site, which need to be downloaded before a particular certificate detail can be compared against all the entries of a CRL. This mechanism was built when the size of the Internet and also the CRL was much smaller than today. Hence scalability is one of the top concerns about the original CRL method of revocation. The CRL again is updated in batches in an interval of few days and hence there a gap between the revocation application and its reflection in the CRL.

I was able to successfully implement the CRL and associate it with the certificates signed, as in Figure 5

1) Concerns with CRL
1. Network Errors – The CRL implementation usually considers a CRL unreachable scenario as a Network error and is not considered as a reason to relocate a certificate. Any DOS attack against the CRL site can help an attacker bypass the CRL check.
2. Overhead – The CRL implementation requires the download of the CRL list every time. A successful verification of certificate revocation with CRL would take close to ~300ms. This is reason enough for the developers to ignore CRL check in the browser and opt for more proprietary methods of revocation check.
3. Software update based revocation – The major revocations due to breaches and significant cyber security changes are released via software updates. Such changes would require the restart of a process or the system, which is usually delayed in production environments. This leads to a gap between detection and implementation of fixes for security vulnerabilities.
4. Captive portal – A captive portal is the front end portal usually seen during login. Such portals may at times block the communication with new CRL sites or OCSP server locations.

B. Revocation method – OCSP

The OCSP or Online Certificate Status Protocol is an alternative to the CRL as a X.509 revocation protocol. The protocol was designed to overcome the inherent weaknesses of the CRL system. The OCSP is much more efficient as it handles only the specific certificate which need to be verified at a time. The transmission is done via HTTP as a “request/response” between an OCSP server and the responder.

The OCSP responder replies back with one of the three status:

i. Good
ii. Revoked
iii. Unknown

Following Figure 6 is a sample response from OCSP server for query on a specific certificate

Figure 6. OCSP response

C. CRL Implementation in Google Chrome

The Google Chromium project has their own proprietary method of certificate revocation. Here a collection of serial numbers of revoked certificates from various CA are collated and published. The size of the CRLSet however is restricted to

X509v3 extensions:
  X509v3 CRL Distribution Points:
    Full Name:
    URI://http://www.mylondonca.com/crl.pem

Figure 5 – X509 extension
less than 300Kb, for faster response time in the browser. However this size limitation hinders the size of the revocation data that can be contained in the CRLSet. Below is the comparison of known CA revocation and what is contained in the CRLSet.

D. CRL Implementation in Firefox

Firefox browser has implemented its own proprietary feature called OneCRL where the list of certificates are pushed into the browser. The web browser will not need to look up for a particular certificate revocation on run time and hence will enhance browser user experience. This design helps in improving the speed and response of the web browser. [10]

![Firefox output in Wireshark](image)

Fig. 7 Firefox output in Wireshark.

There is no reference to the original CRL location in Firefox browser, as in Figure 7.

VII. EXPERIMENTAL TESTBED

A. Operating System

I have used Ubuntu as the base operating system for OpenSSL. However I have tested the web browser in both Ubuntu and Windows10 environments. Please check the Ubuntu version of the testbed in the figure 8.

![Ubuntu version](image)

Fig.8 Ubuntu version

The Ubuntu OS need to be updated with the latest patches so as to eliminate any potential issues with OpenSSL or Apache2 installations, using the following commands

- `apt-get update`
- `apt-get upgrade`

B. OpenSSL for certificate management

Handling SSL certificates and testing its functionality and efficiency being the primary objective of this project, I have installed and used OpenSSL software 1.1.1a. This software was used to simulate a client “Entity”, the CA and the VA or Subordinate CA.

OpenSSL supports entire SSL and TLS stack and has both commercial and non-commercial licensing. The licensing for OpenSSL is similar to that of Apache2 software in this regard.

C. Apache2

Apache project is also known as the Apache HTTP Server project, which is an open source HTTP server. The testbed would require Apache to be installed with SSL option. This is more commonly known as apache with httpd. The documentation for this combination for Ubuntu is hard to find and hence I have consolidated the command sequences that are needed to bring up this testbed in the Ubuntu environment.

1. Install apr-util-1.6.1, using the “--with-apr” option.
   a. `./configure --with-apr=/usr/local/apr/bin/apr-1-config`
   b. `make`
   c. `make install`

2. Install apr-1.6.5, with the libexpat1-dev option, using the following commands.
   a. `./configure`
   b. `aptitude install libexpat1-dev`
   c. `make`
   d. `make install`

3. The “pcre” is the one last component to be installed before httpd installation [9]
   a. First we need to find an appropriate component in Ubuntu, which matches the components needed
   b. for Redhat and other variant using the first command given below.
      i. `sudo apt-cache search pcre | grep pcre | grep dev`
   c. `libpcre3-dev - Perl`

4. Installation of httpd. – Install the latest version of httpd (2.4.37) as on December 2018, using the following commands
   a. `./configure --enable-ssl --enable-so`
   b. `make`
   c. `make install`

5. Installation of Apache. – Once the above dependencies and software are installed without any errors, we can move on to install the Apache2 software, using the following commands [8]
   a. `apt-get install apache2`

D. Two Tier architecture

I have designed a two tier architecture with a Root CA (LONDONCA) who plays the role of the VA as well. The Entity is simulated with a company named [www.breachsafe.com](http://www.breachsafe.com), a domain that I have registered. The Breachsafe.com company would request for a certificate from...
LONDONCA, using a CSR. The details of which are explained in the OpenSSL configuration section below

1) Server Side Initialization

a) Create CA Key Pair

- The first step is to create a key pair for the CA. This is done by the following openssl command

```
openssl genrsa -out LONDONCA.key 4096
```

where LONDONCA.key is the generated private key with a 4096 size for key length.

b) Create a self-signed CA certificate

- A Certificate Authority can issue itself a self-signed certificate. Following is a sample configuration in openssl

```
openssl req -new -x509 -days 1826 -key LONDONCA.key -out LONDONCA.crt
```

where the CA’s own key LONDONCA.key is used to sign its own certificate LONDONCA.crt. The certificate is of x509 format and is valid for 1826 days.

- The self-signed certificate can be verified using the following command

```cfrs@ubuntu:~/Desktop/CERTIFICATE$ openssl x509 -in LONDONCA.crt -text -noout
```

2) Client Side Initialization

The client is simulated by an Apache2 website named www.breachsafe.com. The certificate request will contain the details about this domain and its ownership.

a) Create CA

The first step at the client side is to create a key pair which will be used for the further steps. This is done by the following openssl command

```
openssl genrsa -out MYBREACHSAFE.com.key 2048
```

where, the keysize is 2048

b) Create CSR

Using the 2048 bit RSA key generated in the previous step, we create a CSR or Certificate Signing Request, that will contain all the information that can link the certificate to the identity of the website www.breachsafe.com

```
openssl req -new -sha256 -key MYBREACHSAFE.com.key -out MYBREACHSAFE.com.csr
```

The above command creates a CSR and signs it using the key generated in the earlier step. Once the CSR is generated, it is sent to the CA for validation and signing phases.

3) Server side verification and signing using CRL

a) Creation of CRL

The CA has to publish its CRL location details, which will hence be embedded to all the certificate that the CA will sign. Create crl.cnf that contains publicly available CRL distribution point.

```
crlDistributionPoints=URI:http://www.mylondonca.com/crl.pem
```

b) CSR validation and signing

The server will then validate the CSR and sign it, embedding the CRL in it, in the following steps

```
openssl x509 -req -in MYBREACHSAFE.com.csr -out MYBREACHSAFE.com.crt -days 3650 -Ccreateserial -CA LONDONCA.crt -Ckey LONDONCA.key -Cserial serial -extfile mycrl.cnf
```

Here MYBREACHSAFE.com.csr is the CSR generated at the client, the certificate generated for the client is MYBREACHSAFE.com.crt, which is valid for 3650 days. The Ccreateserial generates a specific serial number for this certificate, which is crucial in the revocation phase, where the certificate that has to be revoked is recalled by the specific serial number.

The serial number for the certificate can be verified in the following file.

```
cfrs@ubuntu:~/Desktop/CERTIFICATES$ ll serial*
```

Signature Algorithm: sha256WithRSAEncryption

<SNIP>
4) Revocation of a certificate
The revocation of a certificate is initiated by the same Entity (client) who generated the CSR or can be issued by the CA during a breach of integrity of one CA or its trusted associates and also in the cases where major security vulnerabilities are discovered or exposed in the existing security algorithms and protocols.

Enter pass phrase for intermediate.key.pem: secretpassword
Revoking Certificate 1223.
Data Base Updated

a) Observation:
1. The certificate index will be prefixed by the character “R” indicating the revocation status for the particular serial number.
2. The server will recreate the CRL to recompile the list of certificates listed in the public CRL listed URL.

5) Reasons for Certificate Revocations in a CRL
The CRL revocation is associated with a specific reason why the certificate was revoked. The below listed are the reasons why a certificate number may be listed in a CRL.
1. KeyCompromise – This category is used when the private key used to sign the certificate has been compromised.
2. CACompromise – This indicates that the CA’s private key has been compromised and all the certificates signed with that key will have to be revoked under this category.
3. AffiliationChanged - The user has concluded its association with the organization listed in the Distinguished Name attribute of the original certificate.
4. Superseded – This happens when a replacement certificate is issued to a user following a planned change of keys.
5. CessationOfOperation – This reason is sited usually when the CA is decommissioned
6. CertificateHold – Is a revocation status which is temporary and indicates that CA will no longer vouch for that certificate.
7. RemoveFromCRL – Once the certificate is removed from the CertificateHold state, and the certificate is un-revoked, the status will be changed to RemoveFromCRL.
8. Unspecified. Any revocation without a specific reason code. This is used for audit trials

6) CRL Fields
The CRL contains information about the revoked certificates in the following format-
- Type - displays the type used. E.g. X.509
- Version - List the CRL version type. E.g. Version 2
- This Update - This field reflects the date of issue of the CRL
- Next Update - This field list the next possible CRL release date. This date give the max duration before which the next CRL has to be issued. More details about this can be find in RFC5280[6].
- Signature Algorithm - This field reflects the Algoritm used to sign the CRL
- Issuer - show the issuer of the CRL
- Extensions - contains one or more pre defined CRL extensions
- Revoked Certificates - gives the list of revoked certificates listed under this particular CRL [4][2][2]

7) Setting up Apache2
Once the CSR generated is signed by the CA into a certificate, the next step is to configure the host operating system and the associated webservers with the location and details of the client certificates and server certificates. Following are the files in Apache2 that would need modification for this test setup
- /etc/apache2/sites-available/default-ssl.conf
  - Configure the following
    - ServerName.
    - ServerAlias.. 
    - SSLCertificateFile SSLCertificateKeyFile
    - SSLCACertificateFile
    - SSLCACertificatePath
- <VirtualHost> configuration in Apache
  - Configure the following
    - DocumentRoot .
    - ServerName
    - SSLEngine on
    - SSLCertificateFile – Location of the SSL certificate file
    - SSLCertificateKeyFile – Location of the SSL certificate folder
- Hosts file in Apache
  - The hosts file contains static ip to domain mapping. Ensure that the configurations are uptodate for the webserver address.

8) Adding CA to the web browsers
The CA certificate and all the associated certificates in the chain of trust need to be downloaded and installed in the web browsers.
In Firefox, open the Tools menu > Options > Advanced > Security >View_Certificates. Import under the “Authorities tab”. You will be add to import own CA keys in this manner, as listed in Figure 9.
9) **Errors handled during testbed buildup**

1. Self-signed certificate

   **MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT**
   Error would mean that the certificate was self-signed. This would result in validation failure for a website. The certificate should be signed by a trusted Certificate Authority.

2. Bad certificate domain

   **SSL_ERROR_BAD_CERT_DOMAIN**
   I had faced this error when there was a mismatch between the real website name and the name given in the CSR. The certificate would reflect the name as given in CSR. Any mismatch in the domain names will result in the above error.

3. Unknown issuer

   **SEC_ERROR_UNKNOWN_ISSUER**
   This would be a common error seen when using a private or experimental CA, which is not listed in the web browser list of known Certificate Authorities. You may need to manually configure the CA details, as explained in the "Adding CA to the web browser section"

10) **Launching the website from Apache2**

   ![Secure website](image)

   **cfrs@ubuntu:~/Desktop/CERTIFICATES$ openssl x509 -in MYBREACHSAFE.com.crt -text -noout**
   **Certificate:**
   **Data:**
   Version: 3 (0x2)
   Serial Number: 17714048861471480003 (0xf5d4f19621eb69e3)
   Signature Algorithm: sha256WithRSAEncryption
   Issuer: C=UK, ST=ENGLAND, L=LONDON, O=LONDON CA PVT LTD, OU=LONDON CERTIFY, CN=www.londonca.com/emailAddress=arun-paul@live.com
   Validity
   Not Before: Dec 6 01:22:34 2018 GMT
   Not After : Dec 3 01:22:34 2028 GMT
   Subject: C=US, ST=VIRGINIA, L=FAIRFAX, O=BREACHSAFE CO, OU=BREACHSAFE SALES, CN=www.breachsafe.com/emailAddress=arun-paul@live.com
   Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   Public-Key: (2048 bit)
   Exponent: 65537 (0x10001)

   **X509v3 extensions:**
   X509v3 CRL Distribution Points:
   Full Name:
   URL: http://www.mylondonca.com/crl.pem
   Signature Algorithm: sha256WithRSAEncryption

   Please check figure 12 for certificate contents.
VIII. BROWSER BEHAVIOR ANALYSIS FOR PKI CERTIFICATE REVOCATION

There were a series of test cases issued on the following browsers in Ubuntu and Windows platforms.
1. Firefox
2. Chrome
3. IE

A. Test cases

1. CRL Unavailable

This scenario is tested by listing a CRL location which is unavailable. The client browsers do not suspect a CRL list unavailability as a threat and ignores the issue as a network reachability category and do not flag any security concerns. This behavior was consistent across all three browsers on both windows and Ubuntu platforms.

2. Invalid Certificate

Invalid certificates are created by intentionally creating a mismatch between the FQDN in the certificate and the actual domain name.

3. Expired Certificate

Expired certificates are made by issuing them with a short expiry date and checking for the browser evaluation and detection of certificate expiry

4. Revoked Certificate (at the Root CA)

The certificate revocation method is explained in detail in the “Revocation of a certificate” section

The findings of this test topology are consolidated below in Table 2.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Expires</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>Invalid</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>Revoked (by Root)</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Fail</td>
</tr>
<tr>
<td>CRL Unavailable</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

Table 2. Test results

B. Out of scope of this project

The following scenarios are out of scope for this project.
1. Have not considered EV certificate – EV certificates are checked in a different methodology in each router and their output would vary with that of a non EV certificate. Testing the EV certificates needed special OID values and I was not able to accommodate them in this test environment.
2. Mobile browsers have found to have the weakest security implementations and they need to be extensively tested and documented. This is not within the scope of this project.
3. Revocation of Subordinate CA is not tested. The real world scenarios will include revocation of an entire Subordinate CA and all its issued certificate if the root CA relocate the Subordinate CA.

C. Conclusion

The PKI framework does not have a standardized implementation method in the web browsers. The vendors prioritize on the response time and usability of the browser over security requirement. In most of the cases it is reflected in the way in which the certificate revocation is handled in the browser software.

CRL and OCSP are the original methods of revocation of certificate in PKI and they are limited by own design. These legacy methods have performance, security and scalability issues as the size of Internet, the number of security events and eventually the sheer number of certificates to be revoked has reached exponential numbers.
The web browser vendors have improvised on the certificate revocation methods to ensure a higher performance, speed and agility. However the efficiency and security of these browsers are not looking good as they cover only a small percentage of the overall CRL certificate numbers considering all the CAs and their revocations.

Two of the promising futures in the certificate revocation technology that I have found are discussed below:

1. **OCSP Stapling**

The original CRL methodology needed download of the entire CRL list from a remote server which added to its latency and delays. OCSP solved this problem by sending a specific http query for the one certificate that need to be queried. However every query mechanism that need to be performed during a web browsing adds to the latency and delays. OCSP Stapling is an innovative methodology where the Certificate Server who owns the certificate performs periodic OCSP requests to the associated CA. The CA will reply back with a time-stamped OCSP response which will be signed by the CA. Such updated certificate validity, vouched by the signing CA will avoid the necessity for a query and reply all together. This methodology improves the security as well as the performance of the web browser.

2. **Certificate Transparency Logs**

Google’s Certificate Transparency project looks like the best promise for a practical, reliable and sustainable certificate validation mechanism. This is a community driven program and is still in the learning and evolution stage although it is put into practice in the google chrome project. This mechanism enables self-audit and revocation based on the certificate validation using certificate hashes directly published by the Certificate Authorities for all the valid certificates in circulation. The validation is done via Merkle Tree which is a hierarchical organization of the hashes certificates and it’s associated CAs. The Certificate Transparency Log is a Publicly Auditable and Append-Only mechanism which is managed by Google. This framework is very beneficial to the parties interested in improving the integrity and security of the internet and it includes domain owners, CAs, web-browser vendors. [6]

REFERENCES