Digital Certificate Revocation

GIAC (GCIH) Gold Certification

Author: Sally Vandeven, sallyvdv@gmail.com
Advisor: Walter Goulet

Accepted:
July 15, 2014

Abstract

Secure communication on the Internet is built around the trust of digital certificates. Web servers present a digital certificate to browsers as authentication much like people present an official picture ID as proof of identity. Digital certificates have an expiration date, however, prior to expiration there are multiple reasons why a certificate may no longer be valid. The Internet's Public Key Infrastructure provides methods for browsers to check the validity of digital certificates but are all browsers configured to perform these checks? The recent Heartbleed vulnerability resulted in thousands of revoked certificates from vulnerable servers that should no longer be trusted. This paper takes a closer look at how digital certificates are revoked and how effectively our browsers use revocation information. It will also examine some of the newer techniques used to detect revocation status like OCSP stapling, OCSP must-staple, OCSP multi-staple and CRLSets.

1. Introduction

Digital certificates are a form of digital identification. Servers may present certificates to client computers. Clients may present certificates to servers. People may present certificates to each other. In all cases, one entity is asserting its digital identity to another. If for some reason an aspect of an identity changes, it should update its digital identification by invalidating or revoking the digital certificate associated with that entity before reissuing it.

This paper will describe, in detail, the certificate revocation process specifically for web servers using the Transport Layer Security (TLS) protocol (Dierks & Rescorla, 2008). TLS is the successor to SSL and is a protocol used to secure sensitive network data such as confidential documents, banking transactions, email and online shopping sites. TLS provides authentication and confidentiality using digital certificates.

Public Key Infrastructure (PKI) refers to the system used by an organization to manage all the tools and components needed to exchange data securely on a public network. A PKI uses public key cryptography, describes a certificate issuing system based on trusted issuing authorities at the root, a certificate validation system and key management system (Vacca, 2004 Ch.1).

When the secrecy of a server's private key has been breached, for example, through a server compromise, the certificate associated with that server could potentially be used by anyone who possesses the private key, to impersonate that server. This would be possible because it is the private key used to sign or "prove" whom the sender of the certificate is. Private key compromise is one of the more important reasons that a revocation mechanism exists.

On April 7, 2014, a vulnerability in the OpenSSL implementation of TLS, called Heartbleed, was made public. The vulnerability had been present for two years and allowed attackers to gather the contents of a server's memory and steal unencrypted sensitive data such as private keys (Codenomicon, 2014). On April 8, 2014, Netcraft, a company that provides Internet performance statistics, reported that approximately

500,000 TLS certificates were vulnerable to the Heartbleed vulnerability and should be revoked. (Mutton, April 2014). With that many potentially compromised servers, the reliability of the revocation process became very important. This paper will detail the revocation mechanism used by browsers and highlight the similarities and differences between their implementations.

2. Digital certificates

A digital certificate is a core element in a public key infrastructure and uses public/private key pairs for encryption and digital signing. Certificates are "…data structures that bind public key values to subjects" as described by the document that defines the current X.509 version 3 standard for certificates, RFC 5280. A certificate authority (CA) "signs" a certificate with its private key and the corresponding public key is embedded in the certificate itself. The only key that will properly verify the signature attached to a certificate is the CA's public key, offering proof that it came from that CA (Vacca, 2004 p.16). It is a model that ultimately relies on trust of the CA. Accepting a certificate as proof of a computer's identity is analogous to accepting a driver's license as proof of a person's identity. We place trust in the processes and procedures of the DMV to issue licenses that contain the picture and information that correspond to the same person being issued the license.

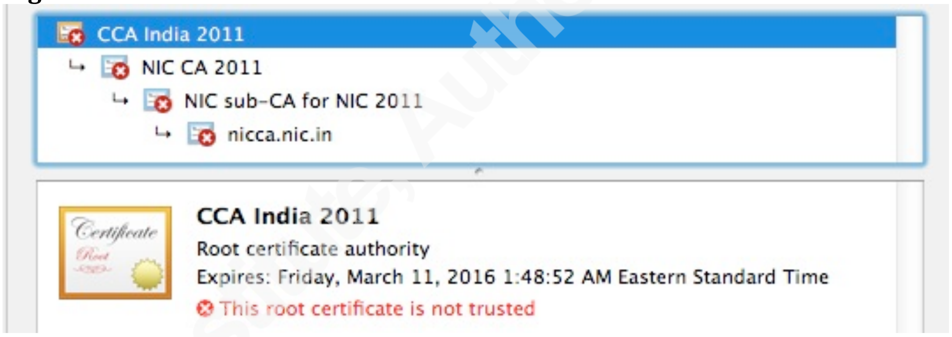
2.1. Certificate authorities

A CA is an entity that issues certificates. There are many well-known commercial CAs such as Verisign, Entrust, Globalsign, Comodo. Any organization can also set up its own local CA to issue certificates for local resources, however, browsers would have to be configured to always trust these certificates in order for them to function without presenting warnings or errors. Commercial CAs typically create a certificate hierarchy. At the top of the hierarchy is the root CA. The root CA usually issues certificates to intermediate CAs. Intermediate CAs can issue certificates for defined uses to any resources including other CAs. All certificates issued by the intermediate CAs will be implicitly trusted according to this chain of trust; the root trusts the intermediate CA, therefore, our browsers trust certificates issued by the intermediate

CA. Each issuing CA (root and intermediate) is responsible for providing revocation information regarding the certificates it issues (Cooper et al., 2008). The various methods for communicating revocation status are discussed in section 3.

On July 2, 2014, Google security engineers discovered fraudulent certificates that had been issued by a CA operated by the Indian government (Langley, 2014, July 8). The CA was a trusted certificate authority for the Windows operating system; therefore any certificates it issued were also trusted. On July 10, 2014, Microsoft issued security advisory 2982792, removing the Indian CA from the trusted root database in Windows (Microsoft, 2014). Figure 1 shows a certificate chain for a certificate issued by the newly untrusted Indian root CA as viewed in Safari[1]. None of the certificates in the chain will be trusted if the root is untrusted.

Figure 1



CCA India 2011
↳ NIC CA 2011
↳ NIC sub-CA for NIC 2011
↳ nicca.nic.in

CCA India 2011
Root certificate authority
Expires: Friday, March 11, 2016 1:48:52 AM Eastern Standard Time
This root certificate is not trusted

2.2. Certificate fields

The content of a certificate varies depending on its prescribed use but most certificates contain X.509 version number, certificate serial number, cryptographic algorithm used for the signature, CA's digital signature, issuing CA, validity dates, server name, server public key, key usage, certificate policies and revocation information.

The certificate serial number is a value assigned by the issuing CA and used to identify the certificate. Every certificate issued by a particular CA will have a unique serial number. This serial number will be used to identify certificates for revocation. A very detailed description of the X.509 version 3 format and standard functionality for digital certificates is documented in RFC 5280 (Cooper et al., 2008) with some recent

[1] This root CA was never in the trusted root store for Safari or Firefox. Only Windows and Chrome running on Windows trusted the CA prior to July 10, 2014.

updates in RFC 6818 (Yee, 2013). For a deeper look at digital certificates including explanations on a certificate's components, uses and certificate chaining see Appendix A.

3. Certificate revocation

Every digital certificate contains an expiration date. The expiration date is often set at one, two or three years from the date of issuance but is determined when the certificate is issued. Because a certificate that has expired is no longer valid, the subject or server using that certificate must acquire a new one. There are reasons, however, that a certificate may need to be invalidated prior to its expiration date. An organization may revoke certificates for its servers at any time and for any reason, however, some of the common reasons include: the private key corresponding to the certificate has been lost or stolen, the domain name of the subject has changed or the subject is no longer in service. When a server administrator discovers, for example, that a server has been attacked and he suspects that the private key may be compromised, he can request that the issuing CA revoke the server's certificate. This would prevent clients from connecting to the potentially compromised server, that is, assuming the clients queried the revocation list. Similarly, client certificates used for individual user authentication/signing may be revoked when an employee leaves an organization or loses control of her private key.

RFC 5280 states that CAs are expected to convey the revocation status for the certificates issued by the CA although the method for doing this is left to the individual CA. The method it recommends is the use of a published certificate revocation list (CRL), however, it suggests that an online lookup method could also be used or any other solution of the CA's choosing. The online method is typically implemented using the online certificate status protocol (OCSP) and is outlined in RFC 6960 (Santesson, et al., 2013). Both the CRL and the OCSP method return a digitally signed response that indicates whether or not a certificate has been revoked. A CRL issuer or OCSP responder is not required to be the same entity that issued the certificate according to the RFC, however, the research done for this paper indicates that revocation information is normally issued by the same entity.

Figure 2

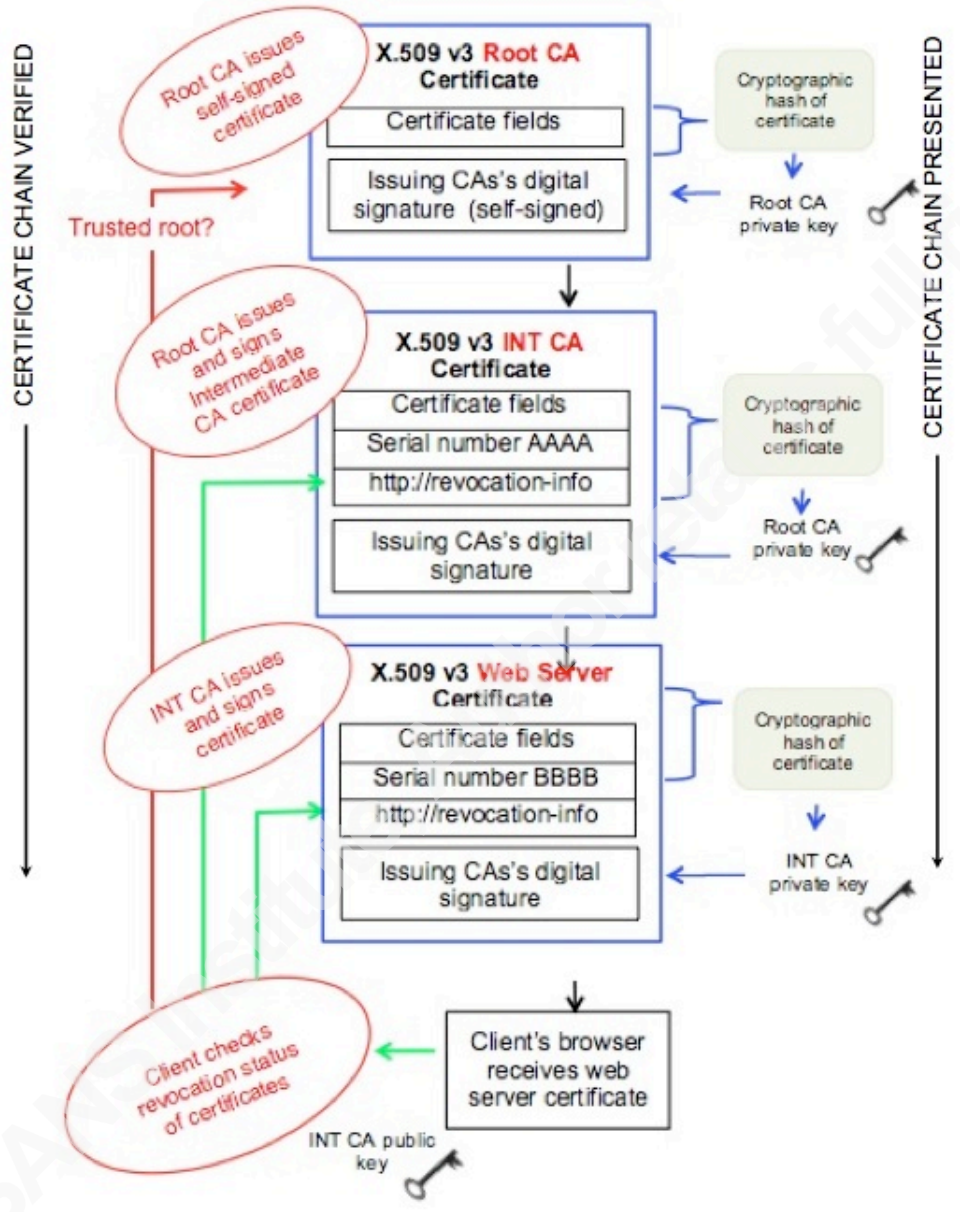

CERTIFICATE CHAIN VERIFIED
Root CA issues self-signed certificate
Trusted root?
X.509 v3 Root CA Certificate
Certificate fields
Issuing CAs's digital signature (self-signed)
Cryptographic hash of certificate
Root CA private key
CERTIFICATE CHAIN PRESENTED
Root CA issues and signs intermediate CA certificate
X.509 v3 INT CA Certificate
Certificate fields
Serial number AAAA
http://revocation-info
Issuing CAs's digital signature
Cryptographic hash of certificate
Root CA private key
INT CA issues and signs certificate
X.509 v3 Web Server Certificate
Certificate fields
Serial number BBBB
http://revocation-info
Issuing CAs's digital signature
Cryptographic hash of certificate
INT CA private key
Client checks revocation status of certificates
Client's browser receives web server certificate
INT CA public key

Figure 2 shows an example chain of certificates that might be presented to a browser establishing a secure connection with a web server. The certificate chain presented consists of a self-signed root certificate, an intermediate CA certificate and finally the web server's certificate, with the non-root CAs providing links for revocation information of their issued certificates with a link to either a CRL or to an OCSP server. When presented with such a chain, a browser will verify the following:

- The web server certificate leads back to a root CA certificate that is trusted and has not expired
- The signature on the intermediate CA certificate is valid and has not expired
- The INT CA certificate (s/n AAAA) has not been revoked
- The signature on the web server certificate is valid and has not expired
- The web server certificate (s/n BBBB) has not been revoked

When the browser is satisfied that all certificates in the chain are valid and have not been revoked, the connection can proceed. A browser's response to a revoked certificate is configurable and can range from "ignore completely and proceed" to "do not proceed with this connection". The default configurations for the major browsers are detailed in section 4.

Every non-root CA is expected to convey revocation information for the certificates it issues, however, there is no requirement that a client use that information. Consider this analogy. When a person exchanges cash for goods at a retail shop, it would be time consuming for the retailer to check the revocation status of the serial number on each paper bill it received and is therefore not common practice. In higher risk environments or with especially large denomination bills the practice of on-the-spot counterfeit detection may be more common. Similarly, some browsers have not been particularly good at thoroughly checking certificates' revocation status by default; however, individual organizations that have a lower risk tolerance are able to enforce revocation checking.

3.1. Certificate Extensions

A CA communicates the location for a certificate's revocation status information by adding a certificate extension to the certificate. The extension would include the link to either a CRL or the URL for the OCSP server. This is an optional extension according to RFC 5280; however, the research done for this paper shows it to be frequently included. Certificate extensions are valid only for the most recent version of X.509, version 3. The extensions used to relate revocation information are *CRL Distribution Points* (for CRLs) and *Authority Information Access* (for OCSP). Additional extensions used for purposes other than revocation include *Key Usage, Alternate Subject Name, Certificate Policies* and are detailed in RFC 5280.

3.2. Certificate revocation request authentication

It is conceivable that an attacker may try to force revocation of a legitimate certificate or otherwise interfere with the normal revocation function. It is therefore a recommended per RFC 5280 that revocation requests be properly authenticated so that only individuals with the proper authority to revoke a certificate are allowed to do so. The methods for authentication vary but most involve creating the revocation request by logging in with a special account or a challenge/response exchange known to the original certificate subscriber only. Every CA publishes a Certification Practice Statement (CPS). The CPS is the CAs rulebook and contains procedures involved with managing certificates, including how revocation requests are authenticated. Links to the CPSs of some of the larger certification authorities can be found in Table 1.

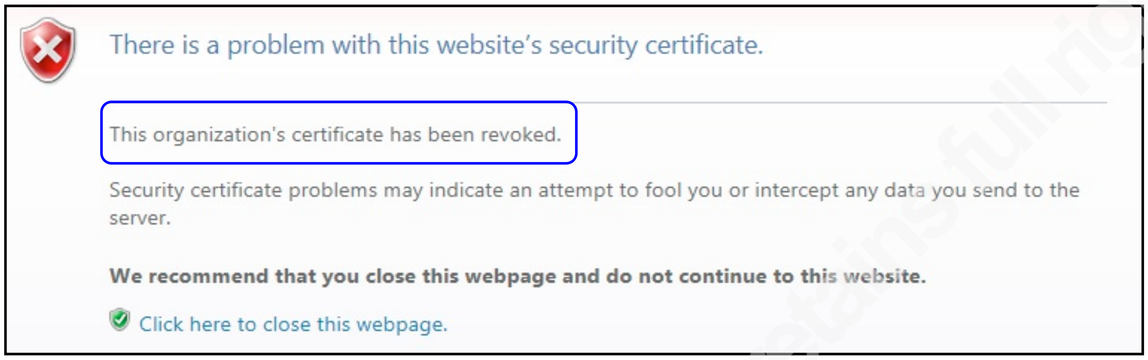
Table 1

Certificate Authority	Certificate Practice Statement URL
CAcert	http://www.cacert.org/policy/CertificationPracticeStatement.php
Digicert	http://www.digicert.com/docs/cps/DigiCert_CPS_v405-May-2-2013.pdf
Entrust	http://www.entrust.net/CPS/pdf/SSL-CPS-English-20140304-Version-2-11.pdf
Geotrust	http://www.geotrust.com/resources/cps/pdfs/GeoTrustCPS-Version1.1.13.pdf
Globalsign	https://www.globalsign.com/repository/GlobalSign_CA_CPS_v7.7.pdf
Godaddy	https://certs.starfieldtech.com/anonymous/repository.pki
Symantec	https://www.symantec.com/content/en/us/about/media/repository/stn-cps.pdf

3.3. How to check revocation status

If a browser is configured to pay attention to revocation status, it will perform a check for each certificate by using a CRL or by processing an OCSP response. If the information received reveals that the certificate has been revoked, the client will either prevent the user from proceeding or warn the user that something might be amiss and allow the user proceed. The response to a revoked certificate varies per browser and is also configurable. Figure 3 shows the response from Internet Explorer when it discovers that it has been presented a revoked certificate.

Figure 3


There is a problem with this website's security certificate.

This organization's certificate has been revoked.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

Click here to close this webpage.

The two methods for checking the revocation status of certificates, CRL and OCSP, will be discussed in separate sections.

3.4. Certificate revocation lists

A CRL is essentially a blacklisting of certificates that can no longer be trusted and is maintained by each CA for its own issued certificates. The CA will update and publish a CRL at regular intervals. A link to the CRL is sent along with each certificate so that a client (browser) can examine the list to determine whether the certificate it is evaluating has not been recalled. Because certificates often are delivered in chains, the client must examine the revocation status for every certificate in the chain up to but not including the trusted root certificate (see section 7.1 for a description of certificate chains).

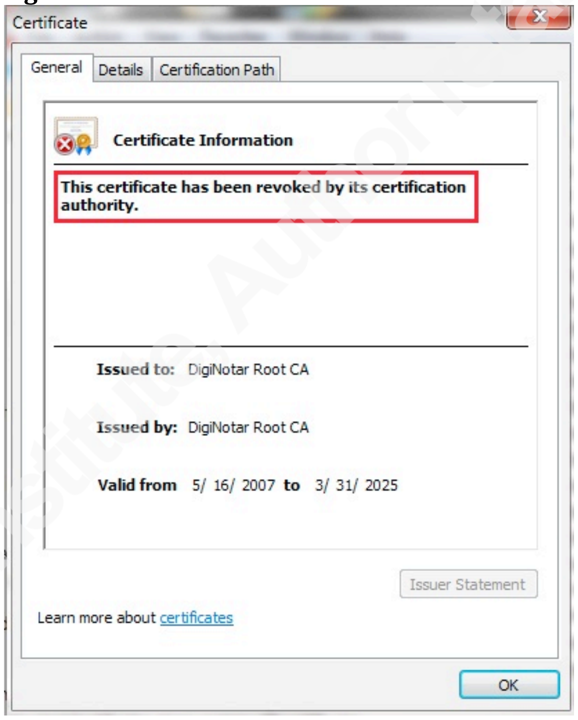
A root CA provides a CRL for the certificates that it issues with one exception; it does not provide revocation information for its own self-signed certificate. Root certificates that need to be revoked should be removed from the *Trusted Root Authority* certificate databases[2]. CRLs are digitally signed by the CA that publishes them in order to verify that the list can be trusted. It is illogical to sign a CRL using the public key

[2] Root certificate revocations are usually delivered in operating system updates, however, Microsoft has an automatic mechanism in place on Windows 8+. Older Windows versions may install the updater from https://support.microsoft.com/kb/2677070

associated with a certificate that is being revoked. Because of this circular trust problem, root CA certificates do not include a link to CRL or OCSP revocation data.

Occasionally, a root certificate does need to be revoked. When this happens, browser manufacturers, as well as operating system vendors like Microsoft and Apple, typically issue updates that will edit the Trusted Root Authority database. An example of an untrusted root certificate authority removed from trusted databases is shown in Figure 4. This was a fraudulently issued certificate that would not expire until 2025 so removing it from the trusted root store was essential.

Figure 4



This certificate has been revoked by its certification authority.

Issued to: DigiNotar Root CA

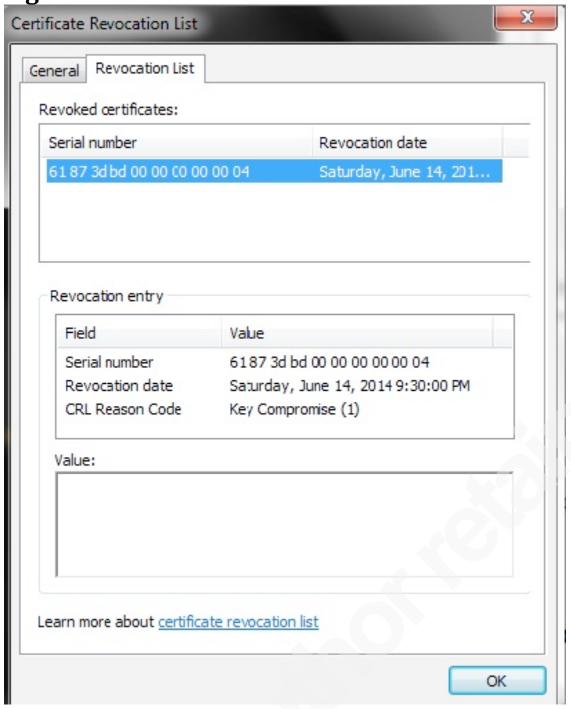
Issued by: DigiNotar Root CA

Valid from 5/ 16/ 2007 to 3/ 31/ 2025

3.4.1. How CRLs work

As noted above, every certificate from a particular CA has a unique serial number. A CRL contains a list of certificate serial numbers that have been revoked according to the CA. Figure 5 shows a CRL containing a single revoked certificate. It is identified by its serial number and shows the date it was revoked along with a reason. If the CA has not revoked any certificates it will publish an empty list.

Figure 5



Serial number	Revocation date
61 87 3d bd 00 00 00 00 00 04	Saturday, June 14, 201...

Field	Value
Serial number	6187 3d bd 00 00 00 00 00 04
Revocation date	Saturday, June 14, 2014 9:30:00 PM
CRL Reason Code	Key Compromise (1)

Learn more about certificate revocation list

CRLs are updated at intervals defined by each issuing CA. Hundreds of CRLs were collected while researching this paper and most common interval used among those CRLs was 7 days. This implies that a certificate that gets revoked the day after a CRL was issued would go unnoticed for 6 days to clients that had a previous cached version of the CRL. This is a window of opportunity that an attacker might be able to take advantage of.

After a TLS handshake, establishing a secure channel with a server, the browser should perform the following steps for each CRL distribution point included in the certificate presented by the server:[3]

1. Client checks CRL cache for the prescribed CRL. If the CRL is not cached locally, a fresh copy will be fetched according to the location provided in the certificate
2. Verify that the issuer of CRL matches CRL issuer field on certificate
3. Validate the digital signature attached to the CRL.

[3] These are the steps that are supposed to be performed if the browser is configured to check CRLs.

4. Search for the certificate's serial number in the CRL.
5. If the serial number is found, the certificate has been revoked. If the serial number is not found, the certificate has *not* been revoked.

CRLs are usually retrieved using unencrypted protocols, that is, the protocol used to retrieve a CRL uses HTTP instead of HTTPS. This is because the security mechanism built-in to protect the integrity of the contents of a CRL is the digital signature of the CRL issuer. There is no need for secrecy of the contents so the overhead of an encrypted connection is not necessary.

3.4.2. What are current sizes of some CRLs?

CRLs may be distributed as a single file or as a base file plus delta CRLs. The base file and the delta CRLs when aggregated will contain the same list as a single master file. The method used is up to the CA. When a browser performs CRL checking, downloading a large CRL file may noticeably slow the initiation of a TLS connection. On June 26, 2014 the current size of Globalsign's CRL was about 1 MB, containing 136,551 revoked certificate serial numbers.

On any platform, a CRL can be downloaded via a browser by entering the URL from the *CRL Distribution Point* extension in the certificate.

Additionally, on Linux or Mac use *wget* or *curl*:

```
$ wget http://crl.globalsign.com/gs/gsorganizationvalg2.crl

$ curl -O http://crl.globalsign.com/gs/gsorganizationvalg2.crl

  %   Total    % Received  % Xferd Average Speed   Time
                                   Dload   Upload  Total

 100 4936k  100 4936k     0     0  1040k      0  0:00:02
```

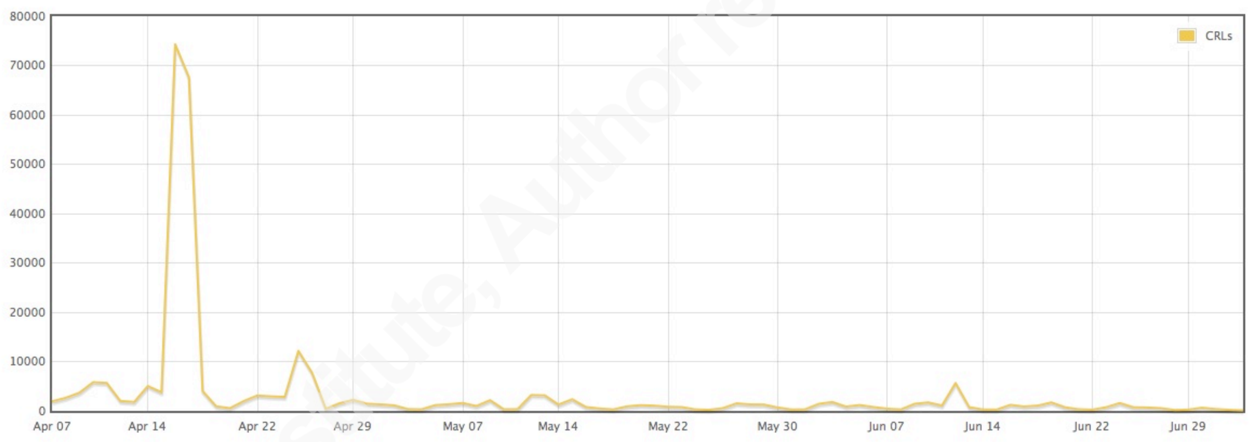
In the example above, a single CRL took about 2 seconds to download from an average home broadband connection. The CRL must subsequently be queried for the certificate's serial number and these operations must be done for all non-root certificates in the chain along with the required signature verification and other steps associated with setting up a TLS session. Because the size of CRLs may present an undue burden on connections, a CA may choose to issue multiple CRLs according to RFC 5280, each with a defined scope to cover a subset of its issued certificates. A browser would need to download and query only the smaller CRL scoped for its certificate.

3.4.3. Revocations over time

Shortly after the Heartbleed vulnerability in OpenSSL was announced on April 7, 2014, the SANS Internet Storm Center began tracking certificate revocations based on well-known CRLs. Figure 6, retrieved from https://isc.sans.edu/crls.html, shows a clear spike in the number of revocations immediately following the announcement.

Figure 6

Certificates Revoked per Day

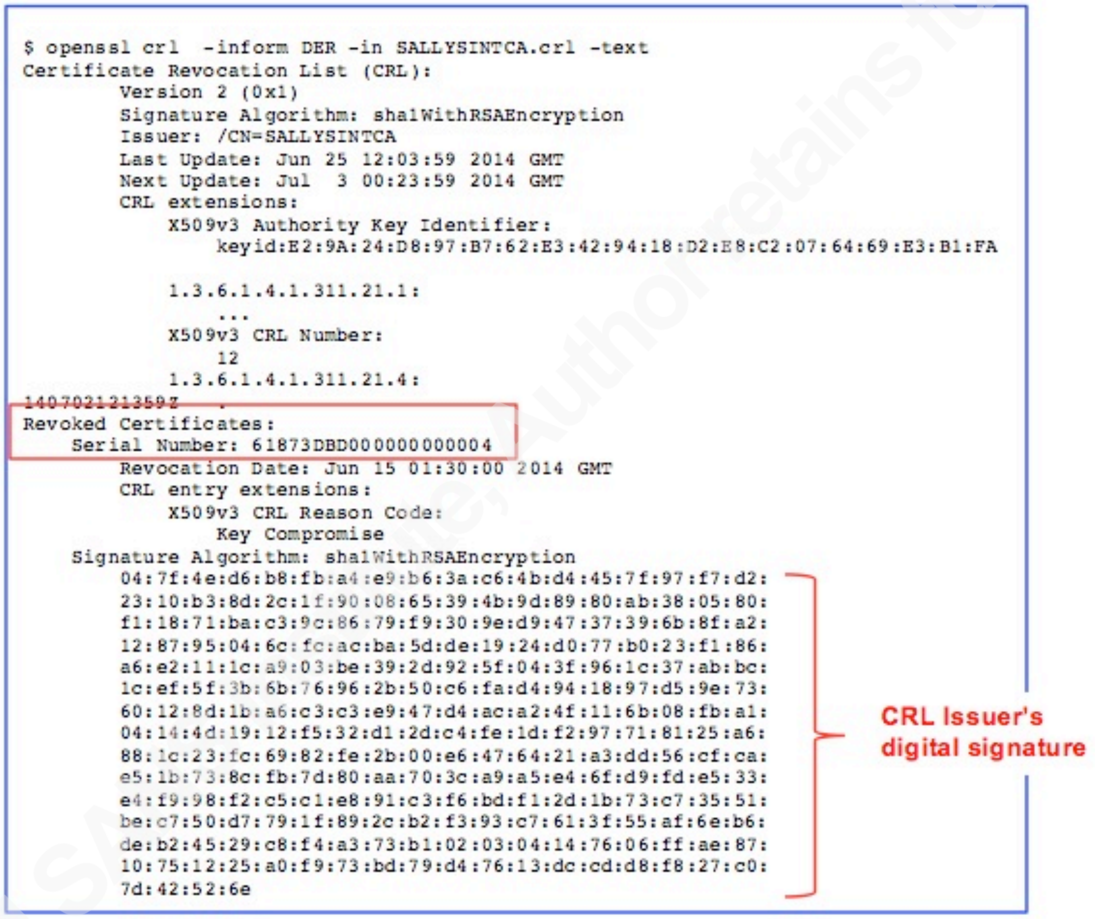


Netcraft, a company that provides Internet performance statistics, reported on April 8, 2014 that approximately 500,000 TLS certificates were vulnerable to the Heartbleed vulnerability and should be revoked. (Mutton, April 2014). Then on May 9th (well after the above revocation spike) Netcraft had calculated that only 14% of the 500,000 certificates had been properly revoked using newly generated private keys. 43% of the certificates were reissued but most either failed to revoke the old certificate or re-used the private key; both are mistakes that leave the server vulnerable (Mutton, May, 2014).

3.4.4. Viewing the contents of a CRL

When a browser makes a TLS connection with a server, the browser will download and examine the CRL without any user intervention. This happens transparently, however, one can manually download a CRL or view the local CRL cache. There are tools for viewing a CRL's contents on both Windows and OSX/Linux machines.

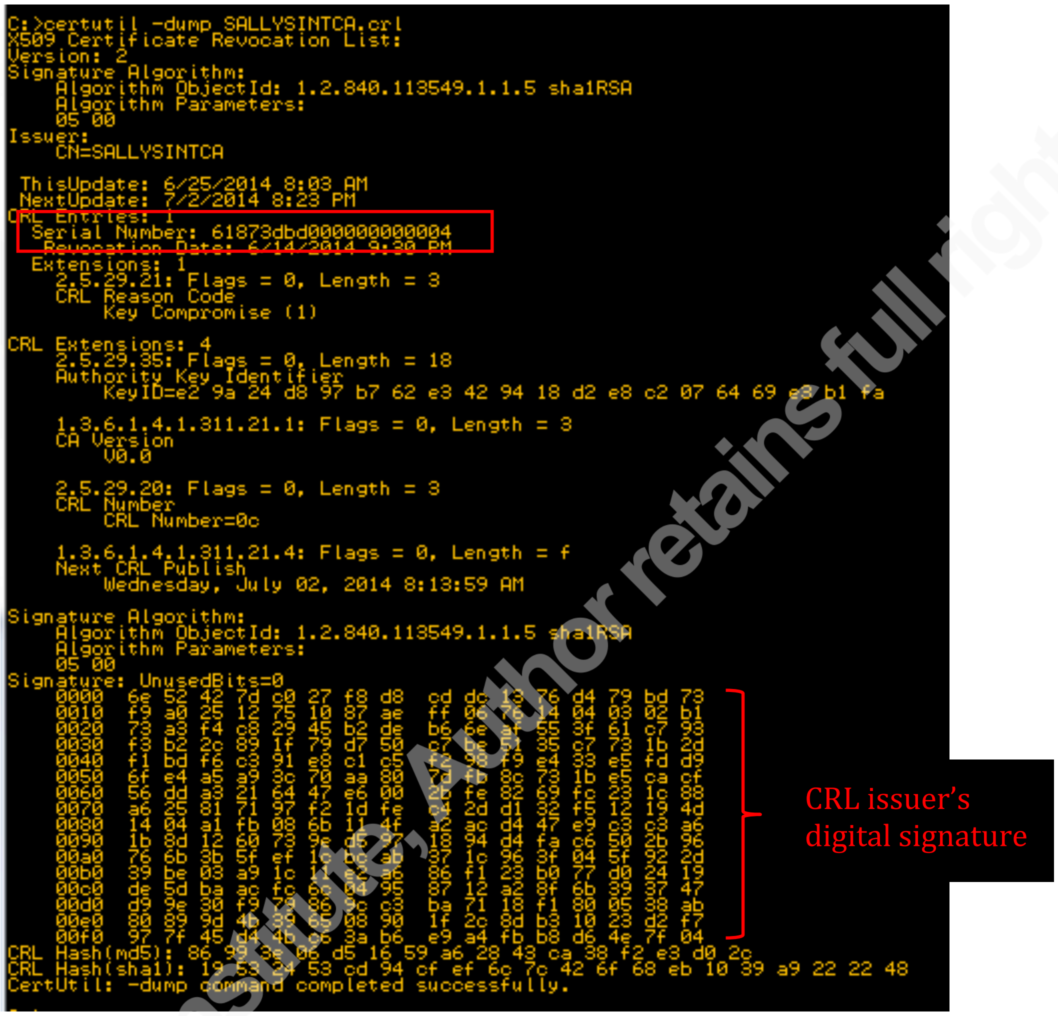
Figure 7



```
$ openssl crl  -inform DER -in SALLYSINTCA.crl -text
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: /CN=SALLYSINTCA
        Last Update: Jun 25 12:03:59 2014 GMT
        Next Update: Jul  3 00:23:59 2014 GMT
        CRL extensions:
            X509v3 Authority Key Identifier:
                keyid:E2:9A:24:D8:97:B7:62:E3:42:94:18:D2:E8:C2:07:64:69:E3:B1:FA

            1.3.6.1.4.1.311.21.1:
                ...
            X509v3 CRL Number:
                12
            1.3.6.1.4.1.311.21.4:
140702121359Z    .
Revoked Certificates:
    Serial Number: 61873DBD000000000004
        Revocation Date: Jun 15 01:30:00 2014 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: sha1WithRSAEncryption
        04:7f:4e:d6:b8:fb:a4:e9:b6:3a:c6:4b:d4:45:7f:97:f7:d2:
        23:10:b3:8d:2c:1f:90:08:65:39:4b:9d:89:80:ab:38:05:80:
        f1:18:71:ba:c3:9c:86:79:f9:30:9e:d9:47:37:39:6b:8f:a2:
        12:87:95:04:6c:fc:ac:ba:5d:de:19:24:d0:77:b0:23:f1:86:
        a6:e2:11:1c:a9:03:be:39:2d:92:5f:04:3f:96:1c:37:ab:bc:
        1c:ef:5f:3b:6b:76:96:2b:50:c6:fa:d4:94:18:97:d5:9e:73:
        60:12:8d:1b:a6:c3:c3:e9:47:d4:ac:a2:4f:11:6b:08:fb:a1:
        04:14:4d:19:12:f5:32:d1:2d:c4:fe:1d:f2:97:71:81:25:a6:
        88:1c:23:fc:69:82:fe:2b:00:e6:47:64:21:a3:dd:56:cf:ca:
        e5:1b:73:8c:fb:7d:80:aa:70:3c:a9:a5:e4:6f:d9:fd:e5:33:
        e4:f9:98:f2:c5:c1:e8:91:c3:f6:bd:f1:2d:1b:73:c7:35:51:
        be:c7:50:d7:79:1f:89:2c:b2:f3:93:c7:61:3f:55:af:6e:b6:
        de:b2:45:29:c8:f4:a3:73:b1:02:03:04:14:76:06:ff:ae:87:
        10:75:12:25:a0:f9:73:bd:79:d4:76:13:dc:cd:d8:f8:27:c0:
        7d:42:52:6e
```

On Windows, OSX and Linux, *OpensSSL* can be used as shown in Figure 7. In addition, Windows has a built-in command line utility called *certutil* that will display the contents of a CRL as shown in Figure 8

Figure 8



```
C:>certutil -dump SALLYSINTCA.crl
X509 Certificate Revocation List:
Version: 2
Signature Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.5 sha1RSA
    Algorithm Parameters:
    05 00
Issuer:
    CN=SALLYSINTCA

 ThisUpdate: 6/25/2014 8:03 AM
 NextUpdate: 7/2/2014 8:23 PM
CRL Entries: 1
  Serial Number: 61873dbd000000000004
   Revocation Date: 6/14/2014 9:30 PM
  Extensions: 1
    2.5.29.21: Flags = 0, Length = 3
    CRL Reason Code
        Key Compromise (1)

CRL Extensions: 4
    2.5.29.35: Flags = 0, Length = 18
    Authority Key Identifier
        KeyID=e2 9a 24 d8 97 b7 62 e3 42 94 18 d2 e8 c2 07 64 69 e8 b1 fa

    1.3.6.1.4.1.311.21.1: Flags = 0, Length = 3
    CA Version
        V0.0

    2.5.29.20: Flags = 0, Length = 3
    CRL Number
        CRL Number=0c

    1.3.6.1.4.1.311.21.4: Flags = 0, Length = f
    Next CRL Publish
        Wednesday, July 02, 2014 8:13:59 AM

Signature Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.5 sha1RSA
    Algorithm Parameters:
    05 00
Signature: UnusedBits=0
    0000  6e 52 42 7d c0 27 f8 d8  cd dc 13 76 d4 79 bd 73
    0010  f9 a0 25 12 75 10 87 ae  ff 06 76 14 04 03 02 b1
    0020  73 a3 f4 c8 29 45 b2 de  b6 6e af 55 3f 61 c7 93
    0030  f3 b2 2c 89 1f 79 d7 50  c7 be 51 35 c7 73 1b 2d
    0040  f1 bd f6 c3 91 e8 c1 c5  f2 98 f9 e4 33 e5 fd d9
    0050  6f e4 a5 a9 3c 70 aa 80  7d fb 8c 73 1b e5 ca cf
    0060  56 dd a3 21 64 47 e6 00  2b fe 82 69 fc 23 1c 88
    0070  a6 25 81 71 97 f2 1d fe  c4 2d d1 32 f5 12 19 4d
    0080  14 04 a1 fb 08 6b 11 4f  a2 ac d4 47 e9 c3 c3 a6
    0090  1b 8d 12 60 73 9e d5 97  18 94 d4 fa c6 50 2b 96
    00a0  76 6b 3b 5f ef 1c bc ab  37 1c 96 3f 04 5f 92 2d
    00b0  39 be 03 a9 1c 11 e2 a6  86 f1 23 b0 77 d0 24 19
    00c0  de 5d ba ac fc 6c 04 95  87 12 a2 8f 6b 39 37 47
    00d0  d9 9e 30 f9 79 86 9c c3  ba 71 18 f1 80 05 38 ab
    00e0  80 89 9d 4b 39 65 08 90  1f 2c 8d b3 10 23 d2 f7
    00f0  97 7f 45 d4 4b c6 3a b6  e9 a4 fb b8 d6 4e 7f 04
CRL Hash(md5): 86 99 3e 06 d5 16 59 a6 28 43 ca 38 f2 e3 d0 2c
CRL Hash(sha1): 19 53 24 53 cd 94 cf ef 6c 7c 42 6f 68 eb 10 39 a9 22 22 48
CertUtil: -dump command completed successfully.
```

3.5. Online Certificate Status Protocol (OCSP)

OCSP is a newer method for checking certificate revocation status and is outlined in RFC 6960 (Santesson, et al., 2013). OCSP was designed to be a more efficient method for certificate status checks than CRLs, allowing the client computer to query a server for information about one particular certificate. The OCSP server, usually called the OCSP responder, does the necessary processing on its end and delivers a status message or "assertion" to the client about a single certificate. With CRLs, the client was required to download a potentially large file and then search through the file looking for an entry corresponding to the certificate in question. In contrast, OCSP sends a small response regarding a specific certificate, reducing both the bandwidth required as well as the amount of processing on the client. Currently, most public OCSP responders are quite reliable as evidenced by the performance statistics gathered by Netcraft. The vast majority of the responders analyzed by Netcraft experience very little down time and fail to provide responses less than 1% of the time (Netcraft, 2014).

3.5.1. How OCSP works

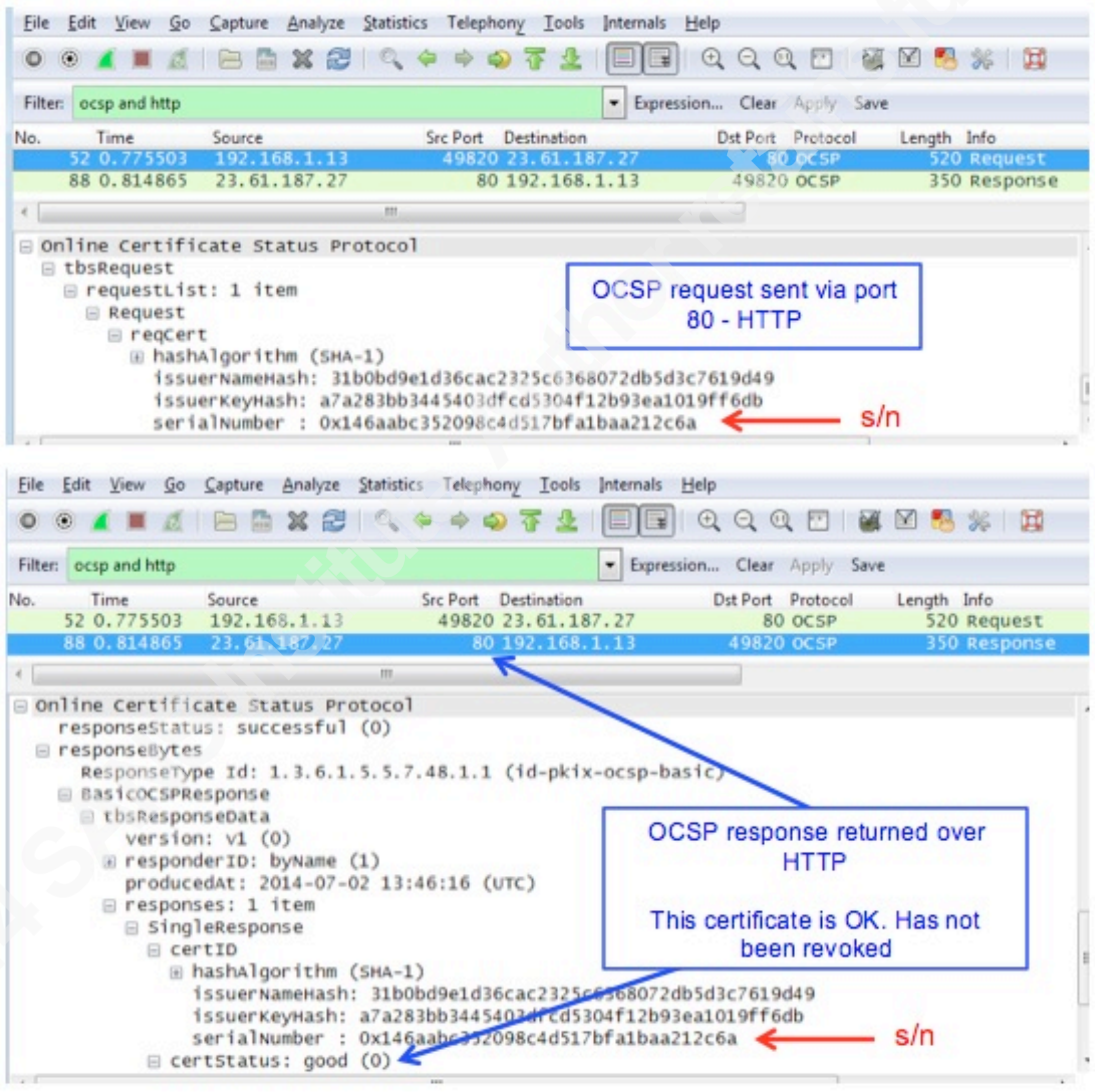
If a CA offers OCSP status checking for its certificate, it will include a link to the appropriate OCSP responder in the certificate as part of the AIA extension (Santesson, et al., 2013, section 3.1). A client sends a request to the OCSP server and receives a response that is digitally signed by the responding server. The responding server may not be the same server that issued the certificate but it must be a server that has been delegated that authority and identified as such in an AIA extension for certificate being checked. A successful response will include a status of *good*, *revoked* or *unknown*. A client that receives a status of *unknown* may make further checks using another server or another method, depending on how revocation checking is configured on the client. Figure 9 shows a packet capture of the OCSP request and response. In this example, OCSP is being sent over HTTP but the protocol being used may vary based on the application requiring the status information. For example, an email application may use SMTP for its OCSP messages. The request is sent to the OCSP responder that was identified in the AIA certificate extension. Figure 9 shows the OCSP response contains the certificate serial number, allowing the browser to match the response to the correct request. A response of good means the certificate has not been revoked. It will be up to the browser to do further evaluation of the validity period of the response, the digital signature, etc. before it will declare the certificate valid.

If the OCSP responder fails to respond then most browsers will default to a "soft-fail"[4] condition. Soft-fail implies that the browser will make a best effort to reach an OCSP server but if it does not succeed it will continue on with connection assuming the certificate has not been revoked. When this happens the client is proceeding to a server whose certificate has not been properly validated. There are many reasons why browsers prefer soft-fail over hard-fail. One of the reasons is that many web applications require a

[4] A "hard-fail" would imply that when no OCSP response is received the browser would refuse to connect to the site.

user to login over a secure connection before proceeding to make any further requests, however, in order to complete the secure connection the client must make an HTTP request to an OCSP server.   A common example of this would be a captive portal or Wi-Fi hotspot. A proposed method called *OCSP must staple* offers a potential solution to these problems and is outlined in section 3.5.4.

Figure 9


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ocsp and http
Expression... Clear Apply Save
OCSP request sent via port 80 - HTTP
s/n
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ocsp and http
Expression... Clear Apply Save
OCSP response returned over HTTP
This certificate is OK. Has not been revoked
s/n

```
No.   Time       Source          Src Port  Destination       Dst Port  Protocol   Length  Info
  52 0.775503   192.168.1.13        49820 23.61.187.27            80 OCSP          520 Request
  88 0.814865   23.61.187.27           80 192.168.1.13         49820 OCSP          350 Response

Online Certificate Status Protocol
  tbsRequest
    requestList: 1 item
      Request
        reqCert
          hashAlgorithm (SHA-1)
          issuerNameHash: 31b0bd9e1d36cac2325c6368072db5d3c7619d49
          issuerKeyHash: a7a283bb344540 3dfcd5304f12b93ea1019ff6db
          serialNumber : 0x146aabc352098c4d517bfa1baa212c6a
```

```
No.   Time       Source          Src Port  Destination       Dst Port  Protocol   Length  Info
  52 0.775503   192.168.1.13        49820 23.61.187.27            80 OCSP          520 Request
  88 0.814865   23.61.187.27           80 192.168.1.13         49820 OCSP          350 Response

Online Certificate Status Protocol
  responseStatus: successful (0)
  responseBytes
    ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
    BasicOCSPResponse
      tbsResponseData
        version: v1 (0)
        responderID: byName (1)
        producedAt: 2014-07-02 13:46:16 (UTC)
        responses: 1 item
          SingleResponse
            certID
              hashAlgorithm (SHA-1)
              issuerNameHash: 31b0bd9e1d36cac2325c6368072db5d3c7619d49
              issuerKeyHash: a7a283bb3445403dfcd5304f12b93ea1019ff6db
              serialNumber : 0x146aabc352098c4d517bfa1baa212c6a
            certStatus: good (0)
```

RFC 6960 does not prescribe a method for OCSP responders to obtain certificate revocation status information. In practice, some CAs have their OCSP responders query their published CRLs while other CAs allow their OCSP responders to access their certificate databases directly (Microsoft, 2013). In the former case, OCSP responders would be able to provide status information only as fresh as the latest issued CRL and in the latter; near real-time revocation status can be provided. Furthermore, OCSP responses may be cached at the client. The research done for this paper shows that OCSP responses are refreshed at a variable rate, but usually between 2 and 7 days.

In general, caching is an effective method to improve response time, however, this efficiency comes with the cost of providing potentially stale information regarding certificate status that could create a window of opportunity for an attacker.

3.5.2. OCSP stapling

OCSP stapling is an extension of the TLS protocol and is the friendly term used for *Certificate Status Request*. A client can request a "stapled" OCSP response, which means the server to which a client is establishing a connection should send a signed OCSP response along with or "stapled to" the certificate during the TLS handshake. The client makes this request during the TLS handshake with the certificate extension *CertificateStatusRequest*. OCSP stapling is defined in RFC 6066 (Eastlake, 2011).

Research done for this paper shows that current versions of Internet Explorer, Firefox and Chrome all support OCSP stapling. Safari, although it does support OCSP, does not support stapling.

Of course, a client can request an OCSP stapled response from a web server but the server must also support OCSP stapling. OCSP stapling is supported by the following web servers according to Globalsign[5]:

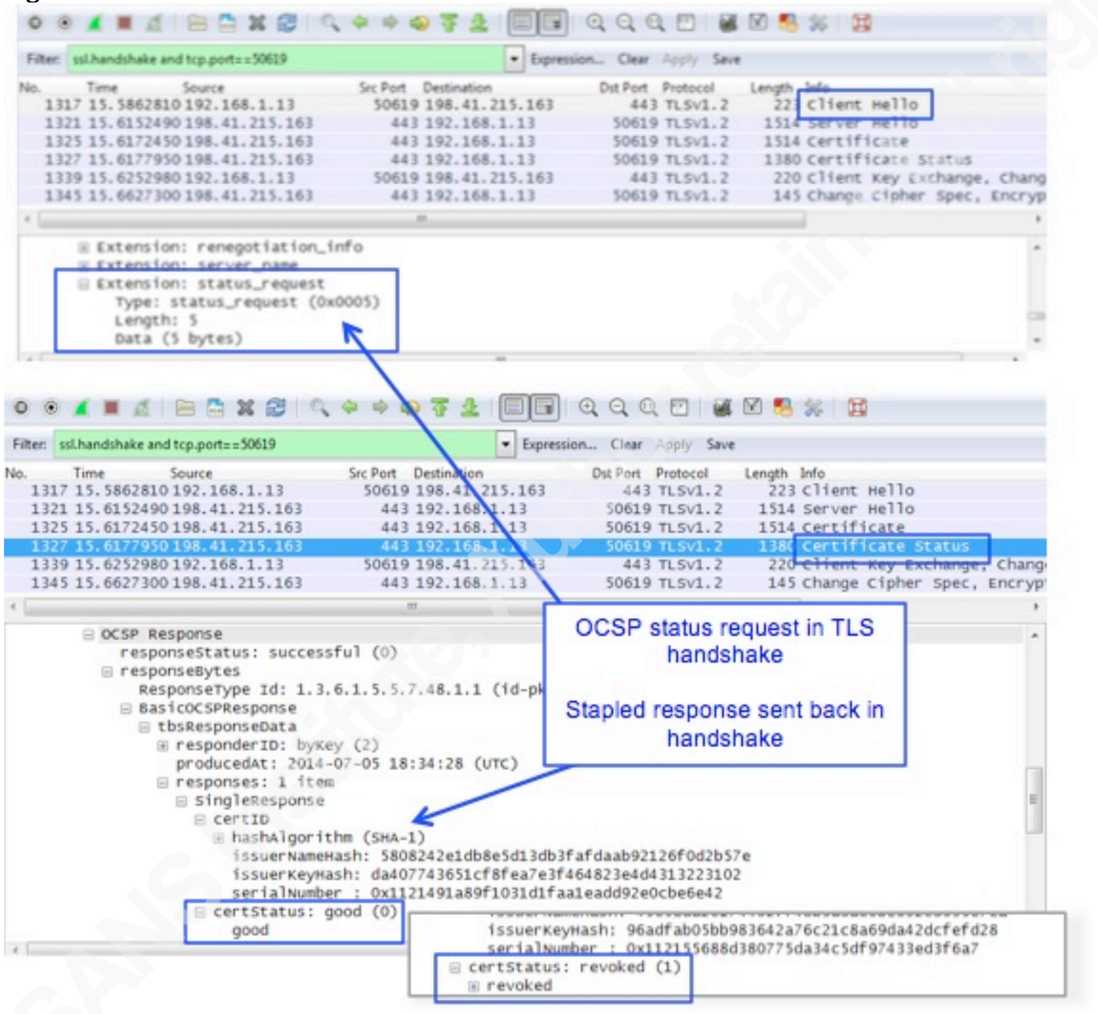
- Apache 2.3 and later
- Nginx 1.3.7 and later
- IIS 7.0 and later

Figure 10 shows the *ClientHello* packet of the TLS handshake in Wireshark. This is the first packet sent to a server when setting up a TLS connection. By including the *CertificateStatusRequest* extension, it is asking the server to send a stapled OCSP response. The response is the *CertificateStatus* message shown in packet 248.

[5] https://sslcheck.globalsign.com/tr/help/26d15ece

As of this writing, OCSP stapling can only be requested for the end server certificate, not other certificates up the chain, although there is currently a draft proposal for *Multiple Certficate Status Request*, also called *OCSP multi-staple* (Pettersen, 2013).

Figure 10



```
Filter: ssl.handshake and tcp.port==50619      Expression...  Clear  Apply  Save
No.   Time       Source          Src Port Destination      Dst Port Protocol  Length Info
 1317 15.5862810 192.168.1.13       50619 198.41.215.163       443 TLSv1.2     223 Client Hello
 1321 15.6152490 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1514 Server Hello
 1325 15.6172450 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1514 Certificate
 1327 15.6177950 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1380 Certificate Status
 1339 15.6252980 192.168.1.13       50619 198.41.215.163       443 TLSv1.2     220 Client Key Exchange, Chang
 1345 15.6627300 198.41.215.163       443 192.168.1.13       50619 TLSv1.2     145 Change Cipher Spec, Encryp

    Extension: renegotiation_info
    Extension: server_name
    Extension: status_request
      Type: status_request (0x0005)
      Length: 5
      Data (5 bytes)
```

```
Filter: ssl.handshake and tcp.port==50619      Expression...  Clear  Apply  Save
No.   Time       Source          Src Port Destination      Dst Port Protocol  Length Info
 1317 15.5862810 192.168.1.13       50619 198.41.215.163       443 TLSv1.2     223 Client Hello
 1321 15.6152490 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1514 Server Hello
 1325 15.6172450 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1514 Certificate
 1327 15.6177950 198.41.215.163       443 192.168.1.13       50619 TLSv1.2    1380 Certificate Status
 1339 15.6252980 192.168.1.13       50619 198.41.215.163       443 TLSv1.2     220 Client Key Exchange, Chang
 1345 15.6627300 198.41.215.163       443 192.168.1.13       50619 TLSv1.2     145 Change Cipher Spec, Encryp

    OCSP Response
      responseStatus: successful (0)
      responseBytes
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pk
        BasicOCSPResponse
          tbsResponseData
            responderID: byKey (2)
            producedAt: 2014-07-05 18:34:28 (UTC)
            responses: 1 item
              SingleResponse
                certID
                  hashAlgorithm (SHA-1)
                  issuerNameHash: 5808242e1db8e5d13db3fafdaab92126f0d2b57e
                  issuerKeyHash: da407743651cf8fea7e3f464823e4d4313223102
                  serialNumber : 0x1121491a89f1031d1faa1eadd92e0cbe6e42
                certStatus: good (0)
                  good

                  issuerKeyHash: 96adfab05bb983642a76c21c8a69da42dcfefd28
                  serialNumber : 0x112155688d380775da34c5df97433ed3f6a7
                certStatus: revoked (1)
                  revoked
```

OCSP status request in TLS handshake
Stapled response sent back in handshake

Web servers can maintain a recent OCSP response in their cache by periodically reissuing the request to the OCSP server. With this method, the server proactively offers the reasonably up-to-date revocation status of its certificate. This saves time for the client because it would no longer need to make an additional connection out to an OCSP server. The RFC states that the server *may* provide a stapled response but it is not required. When a server chooses not to return a stapled response, most clients will proceed with a regular OCSP request over HTTP and if that fails the client will likely proceed to the site anyway; the soft-fail condition.

Some argue that OCSP stapling provides a more private method for clients to check a certificate's status. With regular OCSP queries a client is asking the CA about a certificate for a particular server over an unsecured connection. The CA or someone intercepting the traffic could track that potentially private information. When OCSP stapling is used, however, the client never contacts the CA. Instead the web servers request OCSP responses for their own certificates and staple them to the TLS handshake for the client.

Netcraft reported in May 2013 that approximately 21% of all SSL certificates were presented to clients with an OCSP stapled response (Netcraft, 2013). The SecurityPitfalls website recently performed a scan of the "Alexa top one million"[6] and reports that only approximately %15 of the web servers scanned support OCSP stapling (Kario, 2014). Desktop browser support for OCSP stapling will be detailed in section 4, however, of note here is that all major desktop browsers except Safari will, by default, request OCSP stapling when initiating a TLS connection. This implies that low use of OCSP stapling is due to server configurations. When more servers adopt OCSP stapling, both the performance and privacy problems associated with OCSP can be reduced.

Additional articles regarding OCSP stapling can be found at http://blog.cloudflare.com/ocsp-stapling-how-cloudflare-just-made-ssl-30 and http://nginx.com/news/globalsign-digicert-and-comodo-collaborate-nginx-improve-online-.

3.5.3. OCSP must staple

OCSP must staple means that a signed OCSP response *must* be delivered along with the server's certificate during the TLS handshake. This is a requirement set by the web server. The client could then be configured to accept or refuse a connection based on the response. Currently, a client may request a stapled response but servers are not required to provide one. When that happens the client has a dilemma. It may be that the server does not support stapling but it may also be that an attacker is tampering with the

[6] The Alexa top one million is a list of web sites compiled by Alexa/Amazon and can be downloaded from http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

connection and has removed the stapled response in order to hide the certificate's revocation status.

One of the most important considerations for the implementation of a *must staple* solution is the prevention of just such a downgrade attack or removal of the must staple feature in order to prevent revocation checking. In other words, when a client connects to a web server that communicates the *must staple* requirement, the client knows to look for an OCSP stapled response. When a client connects to a web server that does *not* include the *must staple* requirement, how does it know if the web server does not require it or an attacker has intercepted the connection and removed the requirement? Phil Hallam-Baker of Comodo has proposed a new TLS extension that would prevent this type of attack. The proposed extension would provide a way for the certificate issued to a web server to notify the client of the must staple requirement. Because the certificate is digitally signed, any changes made to the certificate, such as removing the *must staple* requirement, would prevent the certificate from validating properly at the client and would be rejected. The proposal is in a draft stage and does not have an RFC assigned number but it can be found on the IETF website (Hallam-Baker, 2014).

The process of finalizing the RFC for must staple and implementing it is expected to take a long time. It also requires that CAs be prepared to adopt the change and issue new certificates. Mozilla is currently discussing another mechanism to provide must staple in the interim that involves using server HTTP must staple headers. However, this would also have the potential for downgrade attack vulnerability. Discussion and status of this interim solution can be found on the Mozilla wiki (Mozilla, 2014).

There is an active and evolving discussion on the topic of OCSP *must staple* among the IETF-TLS working group members at http://www.ietf.org/mail-archive/web/tls/current/msg12630.html

4. Browser behavior

This section includes results from research done using the 4 major desktop browsers, Internet Explorer, Firefox, Safari and Chrome. The research was done using fresh, default installs of Windows 7 with Internet Explorer 11, Windows 8 with Internet

Explorer 10 and OSX 10.7 with Safari 6.5. Additionally, Firefox 30 and Chrome 35 were installed and used in their default states while examining default behaviors. Settings were then changed where possible to enable and test revocation checks. A test certificate authority infrastructure was configured using Windows 2008 R2 servers and IIS7 for controlled local tests. Also several publicly available web sites whose certificates have been revoked were utilized during the testing procedure.

According to gs.statcounter.com, worldwide desktop browser usage statistics over the past 3 months show that the most common browser used on Desktops is Google's Chrome browser as shown in Figure 11.

Figure 11

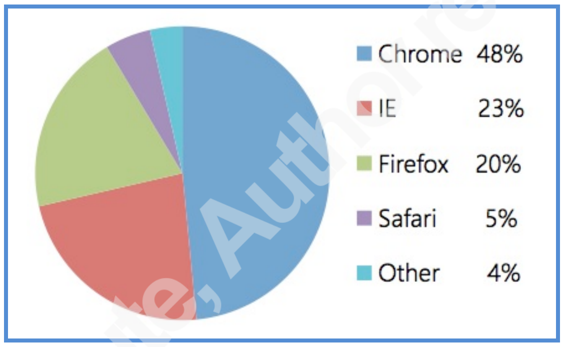
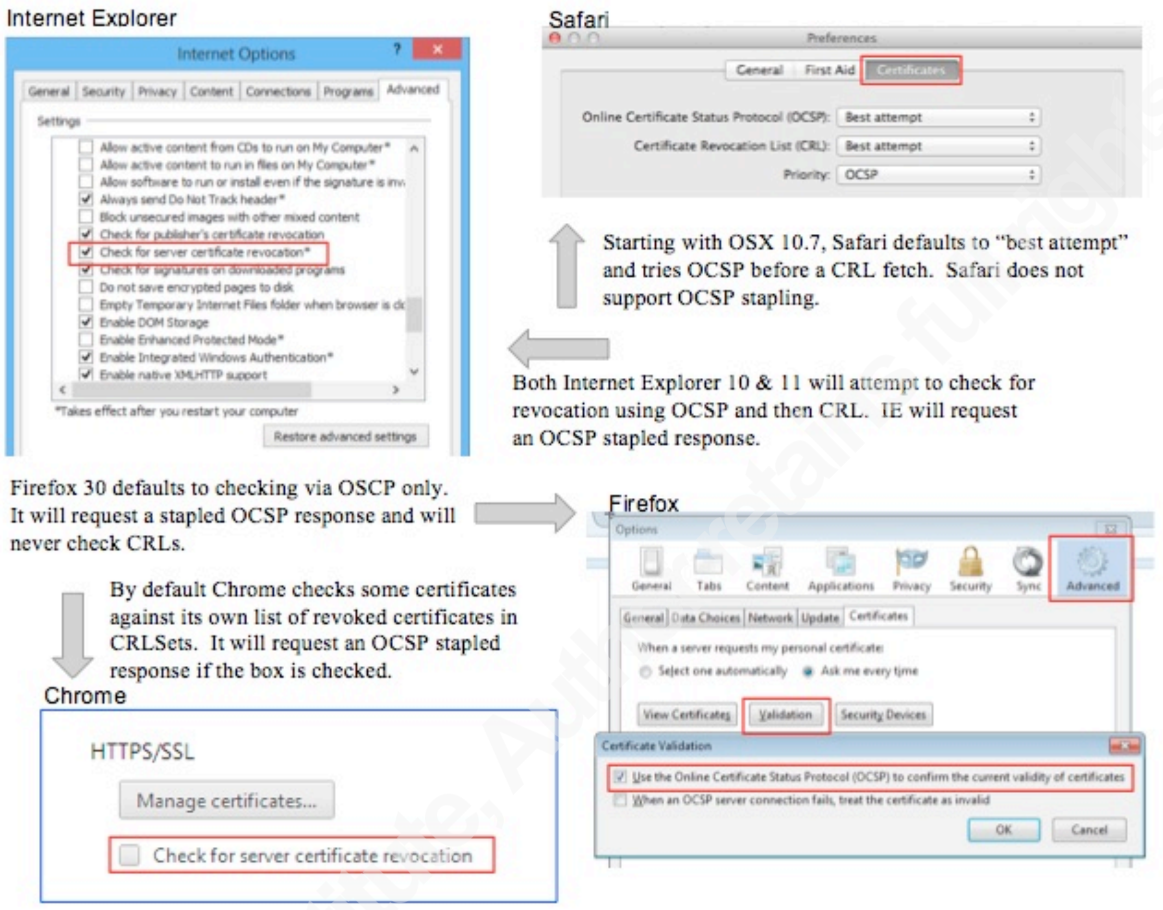


Figure 12


Internet Explorer
Internet Options
General Security Privacy Content Connections Programs Advanced
Settings
Restore advanced settings
Safari
Preferences
General First Aid Certificates
Firefox
Options
General Tabs Content Applications Privacy Security Sync Advanced
General Data Choices Network Update Certificates
View Certificates Validation Security Devices
Certificate Validation
OK Cancel
Chrome
HTTPS/SSL
Manage certificates...

- [ ] Allow active content from CDs to run on My Computer*
- [ ] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [x] Always send Do Not Track header*
- [ ] Block unsecured images with other mixed content
- [x] Check for publisher's certificate revocation
- [x] Check for server certificate revocation*
- [x] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is cl
- [x] Enable DOM Storage
- [ ] Enable Enhanced Protected Mode*
- [x] Enable Integrated Windows Authentication*
- [x] Enable native XMLHTTP support

*Takes effect after you restart your computer

Online Certificate Status Protocol (OCSP): Best attempt
Certificate Revocation List (CRL): Best attempt
Priority: OCSP

Starting with OSX 10.7, Safari defaults to "best attempt" and tries OCSP before a CRL fetch. Safari does not support OCSP stapling.

Both Internet Explorer 10 & 11 will attempt to check for revocation using OCSP and then CRL. IE will request an OCSP stapled response.

Firefox 30 defaults to checking via OSCP only. It will request a stapled OCSP response and will never check CRLs.

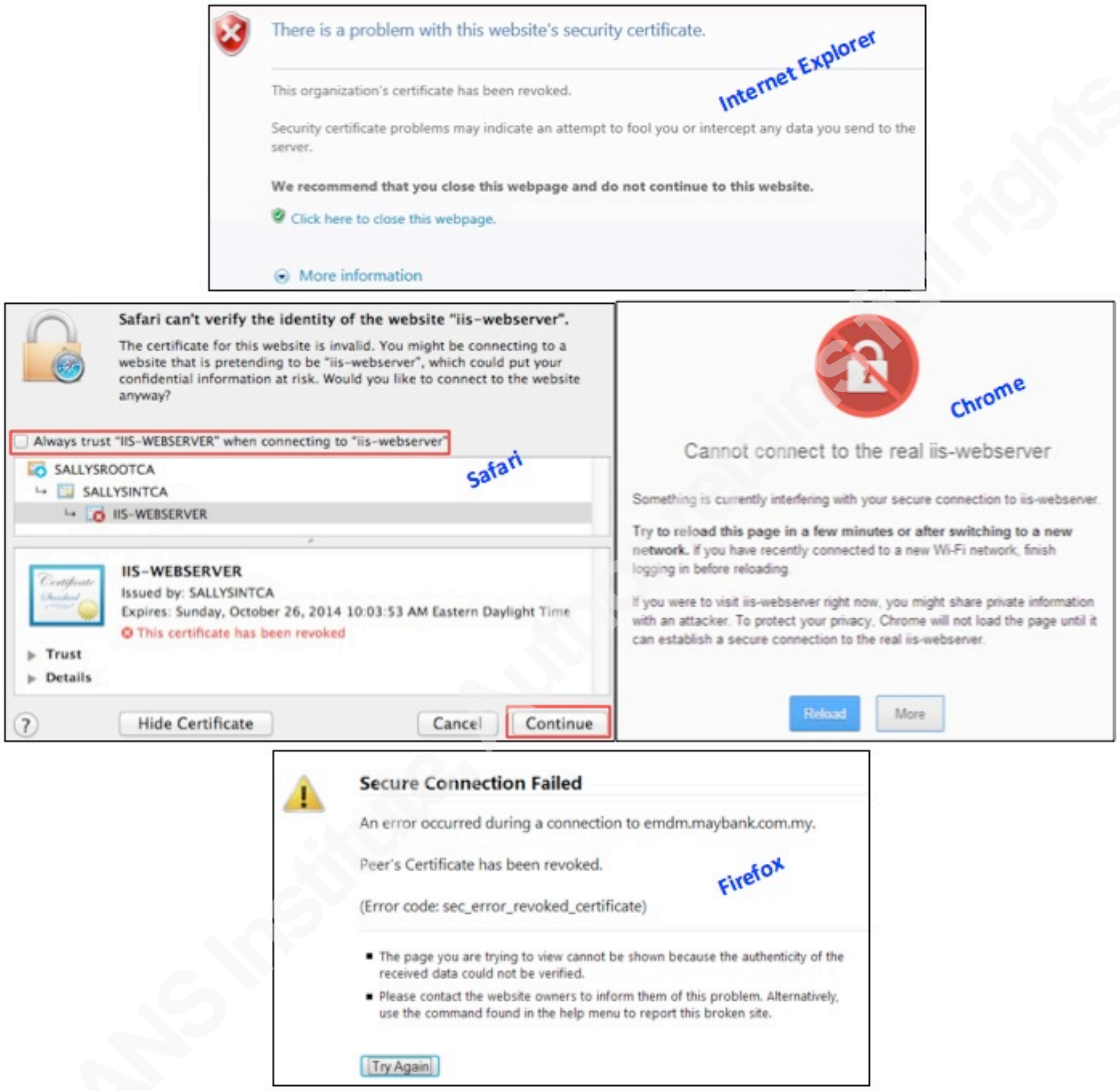
When a server requests my personal certificate:
( ) Select one automatically (●) Ask me every time

- [x] Use the Online Certificate Status Protocol (OCSP) to confirm the current validity of certificates
- [ ] When an OCSP server connection fails, treat the certificate as invalid

By default Chrome checks some certificates against its own list of revoked certificates in CRLSets. It will request an OCSP stapled response if the box is checked.

- [ ] Check for server certificate revocation

The default settings for each of the browsers are shown in Figure 12. All four browsers have the option to check certificate revocation status with only Chrome disabling it by default. Chrome does, however, check certificates against its own list of revocation list, the so-called CRLSets. When configured to perform a check, all browsers except Safari will start with an OCSP staple request when a TLS connection is initiated. When checking is enabled and the browser is notified that a certificate has been revoked all four browsers will notify the user and only Safari will offer the option to continue to the site anyway. Figure 13 shows the warning/error displayed when presented with a revoked certificate.



There is a problem with this website's security certificate.

This organization's certificate has been revoked.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

Click here to close this webpage.

More information

Safari can't verify the identity of the website "iis-webserver".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "iis-webserver", which could put your confidential information at risk. Would you like to connect to the website anyway?

Always trust "IIS-WEBSERVER" when connecting to "iis-webserver"

SALLYSROOTCA
SALLYSINTCA
IIS-WEBSERVER

IIS-WEBSERVER
Issued by: SALLYSINTCA
Expires: Sunday, October 26, 2014 10:03:53 AM Eastern Daylight Time
This certificate has been revoked

Trust
Details

Hide Certificate | Cancel | Continue

Cannot connect to the real iis-webserver

Something is currently interfering with your secure connection to iis-webserver.

Try to reload this page in a few minutes or after switching to a new network. If you have recently connected to a new Wi-Fi network, finish logging in before reloading.

If you were to visit iis-webserver right now, you might share private information with an attacker. To protect your privacy, Chrome will not load the page until it can establish a secure connection to the real iis-webserver.

Reload | More

Secure Connection Failed

An error occurred during a connection to emdm.maybank.com.my.

Peer's Certificate has been revoked.

(Error code: sec_error_revoked_certificate)

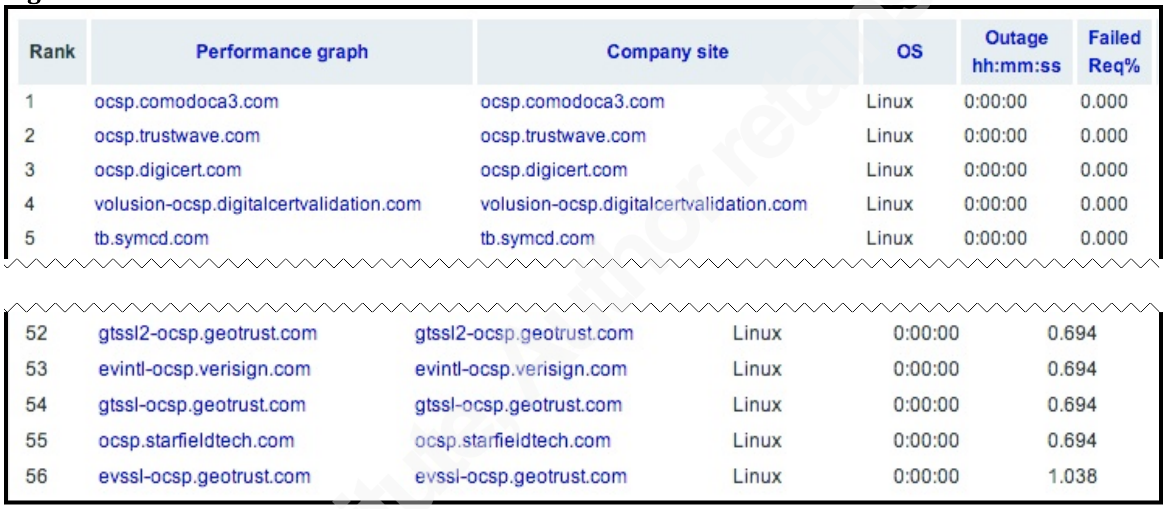
- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem. Alternatively, use the command found in the help menu to report this broken site.

Try Again

Figure 13

The major desktop browsers, Internet Explorer, Firefox, Safari and Chrome are all configurable with respect to how they check for certificate revocation but all have a default configuration that will check OCSP first, CRL second (except Firefox) and will fail-soft in the event that the revocation status of a certificate cannot be determined. That is, if neither an OCSP nor a CRL response is received, the browser will continue on to the site anyway. The reasoning behind this approach seems to be related to the fact that a lack of response from a server is more likely due to a network error or application configuration issue rather than malicious activity [7] [8]. Netcraft maintains a table, currently updated every 15 minutes, of 58 OCSP responders. It collects and aggregates this data from sensors at multiple locations around the world [9]. Figure 14 shows the top 5 performers and the bottom 5 performers from Netcraft's data collection site on July 11, 2014. While the lowest performing OCSP responder had a failed request rate of just over 1%, this means its success rate was almost 99%. Of the 58 responders shown on the site, 29 showed a failed response rate of 0.000%.

Figure 14



Rank	Performance graph	Company site	OS	Outage hh:mm:ss	Failed Req%
1	ocsp.comodoca3.com	ocsp.comodoca3.com	Linux	0:00:00	0.000
2	ocsp.trustwave.com	ocsp.trustwave.com	Linux	0:00:00	0.000
3	ocsp.digicert.com	ocsp.digicert.com	Linux	0:00:00	0.000
4	volusion-ocsp.digitalcertvalidation.com	volusion-ocsp.digitalcertvalidation.com	Linux	0:00:00	0.000
5	tb.symcd.com	tb.symcd.com	Linux	0:00:00	0.000
52	gtssl2-ocsp.geotrust.com	gtssl2-ocsp.geotrust.com	Linux	0:00:00	0.694
53	evintl-ocsp.verisign.com	evintl-ocsp.verisign.com	Linux	0:00:00	0.694
54	gtssl-ocsp.geotrust.com	gtssl-ocsp.geotrust.com	Linux	0:00:00	0.694
55	ocsp.starfieldtech.com	ocsp.starfieldtech.com	Linux	0:00:00	0.694
56	evssl-ocsp.geotrust.com	evssl-ocsp.geotrust.com	Linux	0:00:00	1.038

The data in Figure 14 is representative of server and network related response issues like outages, traffic loads and distances from servers. Response problems due to application specific configuration issues, like captive portals[10], would not be included.

In addition to performance statistics, other factors may affect revocation status responses. If a browser is configured to refuse TLS connections if no revocation status can be determined (hard-fail), then OCSP/CRL servers could become more interesting targets in denial of service attacks. Relying on one server's response for certificate status check represents a single point of failure but certificates can include multiple links for CRLs or OCSP servers in the CDP and AIA extensions, respectively. In addition,

[7] https://www.imperialviolet.org/2014/04/19/revchecking.html
[8] http://news.netcraft.com/archives/2014/04/24/certificate-revocation-why-browsers-remain-affected-by-heartbleed.html
[9] http://uptime.netcraft.com/perf/reports/OCSP
[10] https://www.imperialviolet.org/2014/04/19/revchecking.html

OCSP stapled responses eliminate the need for client queries. The following articles provide additional information regarding soft-fail vs. hard-fail default values and other certificate revocation problems.

- https://wiki.mozilla.org/CA:OCSP-HardFail
- http://news.netcraft.com/archives/2013/04/16/certificate-revocation-and-the-performance-of-ocsp.html
- http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html

4.1. Chrome

Google's Chrome browser handles certificate validation and revocation differently than the other browsers including an added security feature called *public key pinning* and its own revocation mechanism called *CRLSets*.

4.1.1. Public key pinning

Since Chrome 13 was released in May of 2011, it has included a feature called *public key pinning* and sometimes referred to as *SSL pinning* or *certificate pinning*. Chrome stores a whitelist of hashed public keys associated with the certificate authorities that it has authorized to sign Google's certificates. This feature in Chrome allowed Google to detect fraudulent certificates issued for the Google domain on July 2, 2014 (Langley, 2014, July 8) and was the seed for the PKP extension described in section 2.4.1 and was developed to help detect man-in-the-middle attacks of Google's servers. (Langley, 2011).

4.1.2. Chrome CRLSets

Several years ago, Google developed a new method for dealing with revoked certificates called CRLSets (Chromium project, 2011). CRLSets contain a list of revoked certificates, grouped by issuing CA. The CRLSet is pushed to Chrome browsers like an update so the browser need not query a server for revocation status. The maximum size of the file is currently set at 250KB according to the Chromium security team (Chromium project, 2011). Google Chrome will not proceed to a site if it is presented with a certificate included in the CRLSets.

When establishing a TLS connection, Chrome will always send the *status_request* extension, asking for an OCSP stapled response. When presented with a certificate, Chrome will first check the current CRLSet. If the signing CA does not have a set in the current CRLSet, Chrome will process a stapled OCSP response[11] if one was provided. If not, then further revocation checks are only done on EV certificates. The additional check for EV certificates would be to send a query to the OCSP server specified in the AIA extension of that certificate. If there is no response from an OCSP server Chrome will consider the certificate good and proceed to the site.

The use of CRLSets is somewhat controversial because the list of revoked certificates in CRLSets is a small subset of the total revoked certificates according to Adam Langley, one of the engineers on the project (Langley, 2012). Langley argues that there are currently too many problems with regular revocation checking that reduce both browser security and performance (Langley, 2014, April 19). CRLSets addresses some of those problems by eliminating many of the outbound client connections to fetch a CRL or OCSP response because it considers most certificates that have been revoked for administrative reasons as not important to check. See Appendix B for how to view the contents of the current CRLSet.

Research done for this paper shows that reasons are often not given when revoking a certificate and when they are given they may be assigned default values by the web application as it is processing the revocation request. This makes the reason code field an unreliable indicator for the importance of one revocation over another. See Appendix C – Revocation Reasons, for more a more detailed analysis.

By eliminating the majority of the client OCSP queries, Chrome also reduces the privacy concern discussed in section 3.5.2. However, OCSP stapling eliminates the extra client connections as well and similarly improves browser performance and confidentiality.

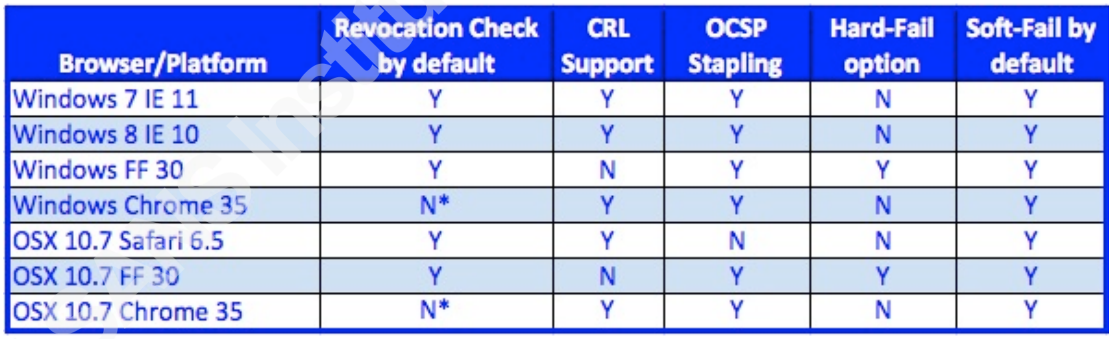
[11] Since Chrome runs on multiple platforms and uses the operating system's cryptographic library, OCSP stapled responses may be processed differently on different platforms http://src.chromium.org/viewvc/chrome/trunk/src/net/cert/.

CRLSets have been successful in providing a fast an efficient method for Chrome to push updates about important revocations. Occasionally fraudulent CA certificates are issued. Any certificates signed by those fraudulent certificates would by definition be trusted; at least until the CA certificate could be removed from browsers' trusted root stores. Chrome CRLSets have been used in a rapid response fashion for such occurrences as recently as July 2, 2014 (Langley, 2014, July 8).

4.2. Browser comparison

Table 2 shows which revocation related features are supported and categorized by browser and platform. According to this table, the Chrome browser does not check revocation by default. As noted in section 4.1, Chrome checks the revocation status of a small subset of all revoked certificates. As of this writing, the only browser that can be configured to refuse a connection if the revocation status for a certificate cannot be determined is Firefox, but this is not a default configuration.

Table 2



Browser/Platform	Revocation Check by default	CRL Support	OCSP Stapling	Hard-Fail option	Soft-Fail by default
Windows 7 IE 11	Y	Y	Y	N	Y
Windows 8 IE 10	Y	Y	Y	N	Y
Windows FF 30	Y	N	Y	Y	Y
Windows Chrome 35	N*	Y	Y	N	Y
OSX 10.7 Safari 6.5	Y	Y	N	N	Y
OSX 10.7 FF 30	Y	N	Y	Y	Y
OSX 10.7 Chrome 35	N*	Y	Y	N	Y

5. Conclusion

The certificate revocation process is adapting in response to researchers' and administrators' discoveries of new attack techniques. Originally, certificates were checked against a CRL or blacklist of known bad certificates. Currently, most browsers use an online check, OCSP, to query for the status of a single certificate before trusting it. One weakness of OCSP is that when the client cannot reach an OCSP server to query the status for a certificate, it defaults to assuming the certificate is OK and continues to the web site. This is referred to as a soft-fail condition. Since the announcement of the Heartbleed vulnerability in April 2014, more attention has been paid to assuring that certificates are properly validated before trusting.

OCSP provides a mechanism for servers to proactively issue a certificate's revocation status during the TLS session setup called OCSP stapling. This mechanism improves browser performance by offloading the revocation status lookup to the webserver that then includes a digitally signed OCSP response along with the presentation of its certificate during the TLS handshake. The problem with OCSP stapling is that the client does not necessarily know if it is supposed to expect a stapled response because servers are not required to provide one. This may lead to the same soft-fail condition; the client continues on to the site assuming the certificate is good. This behavior provides an opportunity to attackers and has lead to the proposal of a *must-staple* option, letting the client know whether or not it should be expecting a stapled OCSP response with the certificate.

The major browsers (Internet Explorer, Chrome, Firefox and Safari) have varying support for the more recent OCSP mechanisms. All 4 browsers support OCSP, however, not all support OCSP stapling. No official standard yet exists for OCSP *must-staple* but it is under development. Furthermore, Chrome has developed its own mechanism for checking revocation status called CRLSets. Chrome uses a combination of CRLSets and OCSP to perform revocation checking.

6. References

Baseline requirements for the issuance and management of publicly-trusted certificates, v.1.1.8 . (2014, June 5). Retrieved July 1, 2014, from https://cabforum.org/wp-content/uploads/Baseline_Requirements_V1_1_8.pdf

CA Browser Forum, (2014, June 5). *Guidelines For The Issuance And Management Of Extended Validation Certificates*. Retrieved June 9, 2014, from https://cabforum.org/wp-content/uploads/EV-SSL-Certificate-Guidelines-Version-1.4.8.pdf

Chromium Project. (2011). CRLSets. Retrieved July 2, 2014, from http://dev.chromium.org/Home/chromium-security/crlsets

Codenomicon. (2014, April 8). The heartbleed bug. Retrieved June 7, 2014, from http://heartbleed.com/

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. Polk, W. (2008) *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* (Request for comments: 5280) Retrieved June 4, 2014 from IETF: http://tools.ietf.org/html/rfc5280

Dierks, T. & Rescorla, E. (2008) *The transport layer security (TLS) protocol version 1.2.* (Request for comments: 5246) Retrieved May 15, 2013 from IETF: http://tools.ietf.org/html/rfc5246

Eastlake, D., (2011) *Transport Layer Security (TLS) Extensions: Extension Definitions* (Request for Comments:6066) Retrieved June 28, 2014 from tools.ietf.org/html/rfc6066

Ellison, C., & Schneier, B. (2000, January 1). Ten risks of PKI: What you're not being told about public key infrastructure. Retrieved June 12, 2014, from https://www.schneier.com/paper-pki.html

Evans, C., Palmer, C., Sleevi, R. (July 4, 2014) *Public key pinning extension for HTTP draft-ietf-websec-key-pinning-19* (Request for Comments) Retrieved July 4, 2014 from http://tools.ietf.org/html/draft-ietf-websec-key-pinning-19

Hallam-Baker, P., (2014) *X.509v3 TLS feature extension draft-hallambaker-tlsfeature-04* (Request for Comments Draft) Retrieved June 28, 2014 from http://datatracker.ietf.org/doc/draft-hallambaker-tlsfeature/

Kario, H. (2014, June 24). RC4 only severs fall below 1% – June 2014 scan results. Retrieved July 3, 2014, from https://securitypitfalls.wordpress.com/2014/06/24/rc4-only-servers-fall-below-1-june-2014-scan-results

Langley, A. (2011, May 4). Public key pinning. Retrieved July 4, 2014, from https://www.imperialviolet.org/2011/05/04/pinning.html

Langley, A. (2012, February 5). Revocation checking and Chrome's CRL. Retrieved July 2, 2014, from https://www.imperialviolet.org/2012/02/05/crlsets.html

Langley, A. (2014, April 19). No, don't enable revocation checking. Retrieved July 2, 2014 from https://www.imperialviolet.org/2014/04/19/revchecking.html

Langley, A. (2014, July 8). Maintaining digital certificate security. Retrieved July 10, 2014, from http://googleonlinesecurity.blogspot.com/2014/07/maintaining-digital-certificate-security.html

Marlinspike, M. (2011, April 11). *SSL and the future of authenticity*. Retrieved from http://www.thoughtcrime.org/blog/ssl-and-the-future-of-authenticity/

Microsoft. (2013, May 3). Online responder installation, configuration, and troubleshooting guide. Retrieved June 19, 2014, from http://technet.microsoft.com/en-us/library/cc770413(v=ws.10).aspx

Microsoft. (2014, July 10). Microsoft Security Advisory 2982792. Retrieved July 13, 2014, from https://technet.microsoft.com/en-us/library/security/2982792

Mozilla. (2014). Plan for improving revocation checking in Firefox. Retrieved June 24, 2014, from https://wiki.mozilla.org/CA:ImprovingRevocation#OCSP_Must-Staple

Netcraft. (2014, July 4). OCSP sites ordered by failures over the last 1 day, updated every 15 minutes. Retrieved July 4, 2014, from http://uptime.netcraft.com/perf/reports/OCSP

Netcraft. (2013, June). SSL survey. Retrieved July 3, 2014, from http://www.netcraft.com/internet-data-mining/ssl-survey/

Pettersen, Y. (2013, June 1). The transport layer security (TLS) multiple certificate status request extension. Retrieved July 13, 2014, from http://www.ietf.org/rfc/rfc6961.txt

Rescorla, E. (2001). *SSL and TLS: Designing and building secure systems*. (1st ed.). Montreal, Canada: Addison-Wesley Professional.

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C., (2013) *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP* (Request for comments: 6960) Retrieved June 8, 2014 from http://tools.ietf.org/html/rfc6960

Symantec Official Blog, (2014 January 23). *Dangers of domain-validated SSL certificates.* Retrieved June 9, 2014, from http://www.symantec.com/connect/blogs/dangers-domain-validated-ssl-certificates

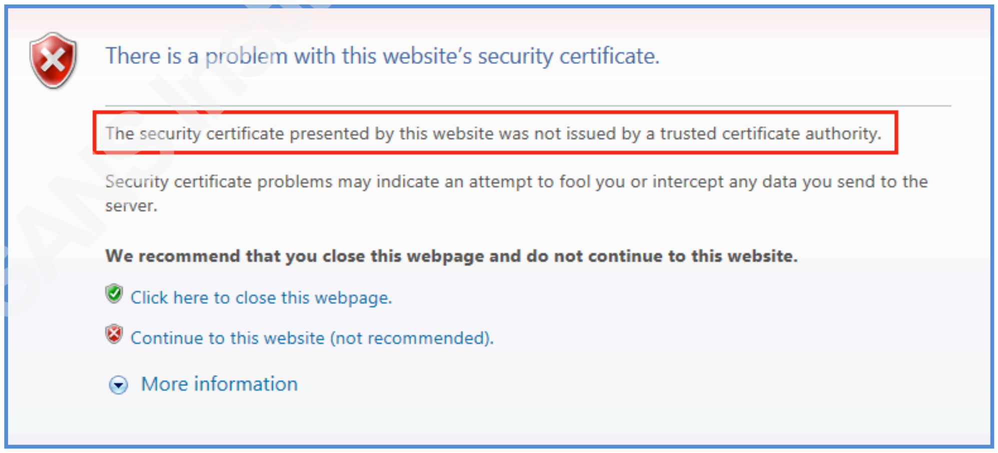
Vacca, J. R. (2004). *Public Key Infrastructure.* Boca Raton: C R C Press LLC.

Yee, P. (2013). *Updates* to the Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. (Request for comments: 6818) Retrieved June 4, 2014 from IETF: http://tools.ietf.org/html/rfc6818

7. Appendix A - Digital Certificates in Depth

7.1. Certificate chains

All certificates must be digitally signed by a certificate issuing authority. Root certificates are self-signed because a root CA is considered the highest-level authority and the authority that we can tell our browsers to "trust". If the issuing CA is not a root CA then one or more certificates must extend back to a root CA, forming a chain. Each certificate in the chain contains a signature from a higher-level authority until the final link in the chain, the self-signed root certificate (Vacca, 2004, Chapter 12). The client receiving a certificate chain will use the information in the entire certificate chain to validate the identity of the server and establish parameters for encryption. If a CA, not known as trusted to the client's browser, issued the root certificate then an error or warning will be issued to the user informing him/her that the certificate cannot be trusted as shown in Figure 15.


There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

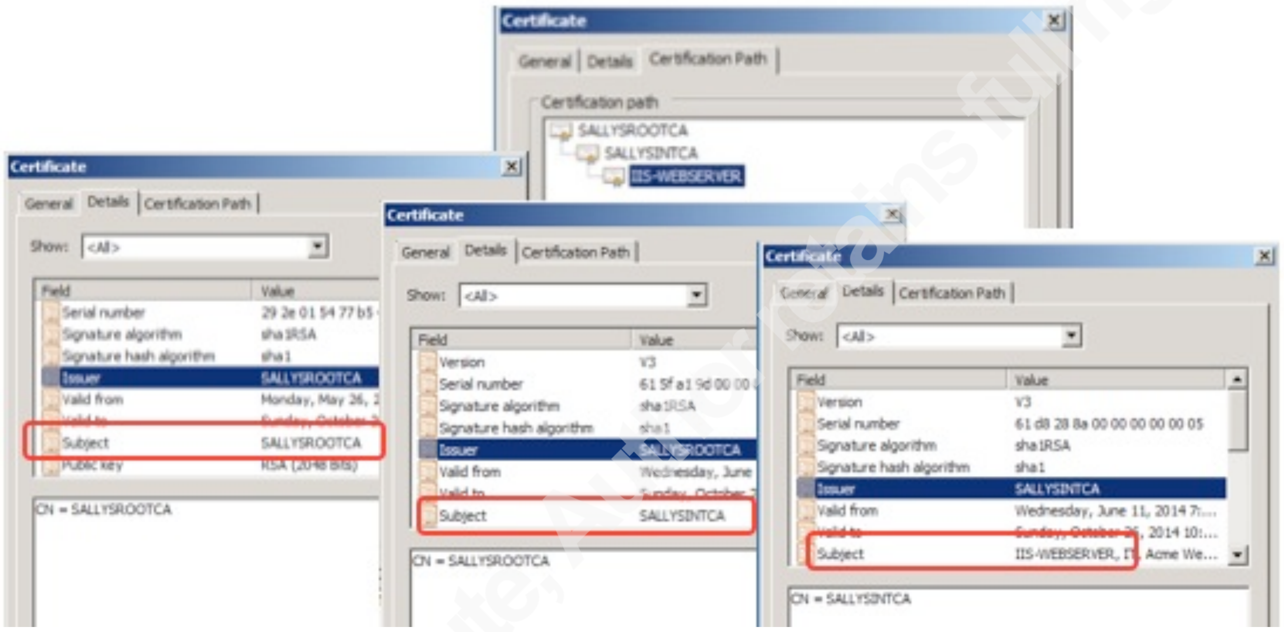
We recommend that you close this webpage and do not continue to this website.

Click here to close this webpage.
Continue to this website (not recommended).
More information

Figure 15

Root CAs are very powerful in the PKI model, therefore, the certificate chaining method is often used because it allows root CAs to perform only the limited duty of signing intermediate CA certificates and can be kept offline and protected most or all of the time. The intermediate CAs also have a more limited duty, issuing certificates that are limited in scope, such as server authentication.

Figure 16



Certificate

General | Details | Certification Path

Certification path

SALLYSROOTCA
SALLYSINTCA
IIS-WEBSERVER

Certificate

General | Details | Certification Path

Show: <All>

Field	Value
Serial number	29 2e 01 54 77 b5
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	SALLYSROOTCA
Valid from	Monday, May 26, 2
Valid to	Sunday, October
Subject	SALLYSROOTCA
Public key	RSA (2048 Bits)

CN = SALLYSROOTCA

Certificate

General | Details | Certification Path

Show: <All>

Field	Value
Version	V3
Serial number	61 5f a1 9d 00 00
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	SALLYSROOTCA
Valid from	Wednesday, June
Valid to	Sunday, October
Subject	SALLYSINTCA

CN = SALLYSROOTCA

Certificate

General | Details | Certification Path

Show: <All>

Field	Value
Version	V3
Serial number	61 d8 28 8a 00 00 00 00 00 05
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	SALLYSINTCA
Valid from	Wednesday, June 11, 2014 7:...
Valid to	Sunday, October 26, 2014 10:...
Subject	IIS-WEBSERVER, IT, Acme We...

CN = SALLYSINTCA

Figure 16 shows a certificate issued to a server named IIS-WEBSERVER. It was issued a certificate from SALLYSINTCA, an intermediate certificate authority, and the three-certificate chain is shown. On the right is the certificate for IIS-WEBSERVER as shown in the *Subject* field. In the middle is the certificate for the issuing CA, SALLYSINTCA, and it was signed by the root CA, SALLYSROOTCA. On the left is the certificate for SALLYSROOTCA and it is self-signed.

7.2. Based on trust

As noted above, the certificate binds a subject to a public key. Since a CA asserts this binding, one relying party accepts the certificate as a method of authenticating a server. Successful authentication of a server is a prerequisite to establishing a secure HTTP (HTTPS) connection between a user's browser and a web server. Using this model, the users must ultimately trust that the CAs have issued legitimate certificates that contain true declarations of identity.

Every browser has a database of trusted root certificates. Each browser manufacturer publishes requirements that a CA must meet in order to be added to its trusted root store. Links to those requirements documents for the four major browsers are:

- Internet Explorer- http://social.technet.microsoft.com/wiki/contents/articles/1760.windows-root-certificate-program-technical-requirements-version-2-0.aspx
- Firefox- https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/inclusion
- Chrome- http://www.chromium.org/Home/chromium-security/root-ca-policy
- Safari- http://www.apple.com/certificateauthority/ca_program.html

Any certificate that meets the following criteria will be considered valid by most major browsers.

- No certificate in chain has expired
- No certificate in chain has been revoked
- Signatures of all certificates in the chain have been validated
- Chain of certificates ends with a trusted root

7.2.1. Public key pinning

To further secure this model, the IETF and Google are developing something called *public key pinning (PKP)*. PKP refers to web servers asking browsers that connect to it to *pin* or remember some of the information in the certificate so that in subsequent visits the client would be better able to detect if an attacker were trying to impersonate the server. Much like browsers contain a store of trusted root authorities, they would also maintain a store of public keys belonging to CAs trusted to sign certificates for a particular site. For example, when mycompany.com sends a certificate chain to the browser in the *ServerHello* message of the TLS handshake, the browser would confirm that at least one of the signing authorities in the chain is considered trusted to issue certificates for mycompany.com. If PKP is widely adopted a browser will no longer trust every certificate signed by a trusted CA. The browser would only trust the certificates signed for hosts that have elected to authorize that CA to sign its certificates (Evans, et. al., 2014).

7.2.2. How a web server administrator obtains a certificate

A user must prove to the CA that s/he has the authority to request a certificate for a particular resource. Procedures for this vary according to CA and certificate type. In general, the user generates a public/private key pair and submits a certificate request containing the public key from the key pair and various other parameters including the desired length of validity and the strength of the encryption used. After the CA has satisfactory authorization of the person requesting the certificate, the CA generates a certificate containing the public key and other fields according to the X.509 specification for digital certificates (Cooper et al., 2008). The CA Browser Forum publishes guidelines for CAs regarding issuing and managing TLS certificates (CA Browser Forum, June 2014) and extended validation certificates (CA Browser Forum, 2014). *Extended validation (EV) certificates* for individual servers that require a more rigorous method of verification of the server owner.

Two important components of the certificate are the public key and the digital signature of the issuing CA. The digital signature is an encrypted hash of the certificate fields. The CA calculates a hash of the certificate fields and then encrypts it with its own private key. Anyone that needs to verify that the CA digitally signed the certificate can use the CA's public key to decrypt the accompanying hash and then compare it to his or her own hash calculation on the certificate fields.

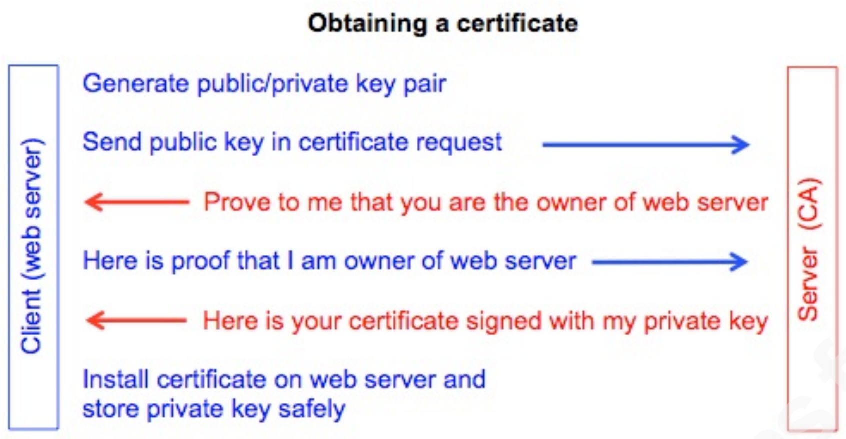

Obtaining a certificate
Client (web server)
Server (CA)
Generate public/private key pair
Send public key in certificate request
Prove to me that you are the owner of web server
Here is proof that I am owner of web server
Here is your certificate signed with my private key
Install certificate on web server and store private key safely

Figure 17

The validity of a digital certificate ultimately relies on our trust of a given CA. We can mathematically prove that a CA signed a certificate by comparing hashes as described above but we cannot mathematically prove that the information contained in the certificate is true. Instead, we must trust that the information in the certificate is true because we trust that the CA has done its job correctly. The CA trust model was questioned by Bruce Schneier many years ago (Ellison & Schneier, 2000) and again more recently by Moxie Marlinspike (Marlinspike, 2011).

7.3. Certificates can have multiple possible uses

When a person requests a certificate from a CA, it will be created after ownership or authority over the domain or resource has been established by the CA as described in section 2.4. The level of identification required varies based on the type of certificate being requested. According to RFC 5280, CA's can issue certificates for the following uses:

- Digital signatures (public key used for signing things other than certificates)
- Non-repudiation (public key used for verifying signatures other than those on its certificates)
- Key encipherment (public key used to encrypt symmetric key)
- Data encipherment (public key used to encrypt data)
- Key agreement (public key used with Diffie-Hellman key agreement)

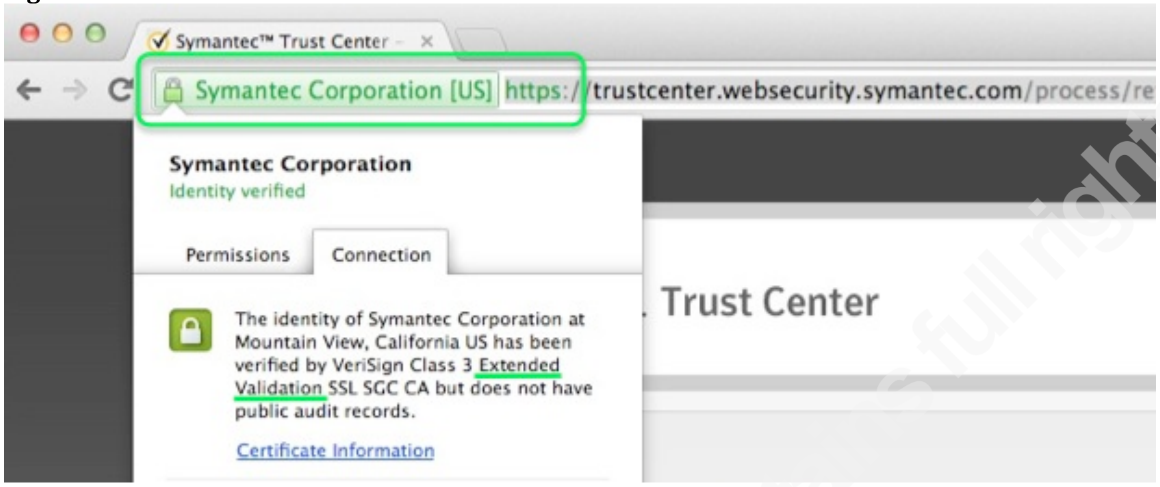
- Key certificate signing (CA's certificate would have this defined)
- Certificate revocation list signing (public key used to verify revocation lists)

Research done for this paper shows that web server certificates are issued with at least the *Digital signature* usage and often for *Key encipherment* as well. X.509 version 3 allows the use of *extensions*. Extensions provide a way to assign additional attributes to a certificate. The extension seen most frequently on web server certificates that are used for TLS sessions is *Server authentication*. According to RFC 5280, including a key usage attribute of *Digital signature* and an extended key usage attribute of *Server authentication* implies that the certificate should only be used to send digital signatures when authenticating itself to a client and any other use of the certificate like data encryption should be rejected. A very detailed description of the format, uses and functionality for digital certificates is documented in RFC 5280 (Cooper et al., 2008) with some recent updates regarding self-signed certificates in RFC 6818 (Yee, 2013).

7.4. Types of server certificates

In addition to multiple uses, there are also multiple types of web server certificates that a CA could issue. Digital certificates can be acquired for a single server or for multiple servers. A certificate that can be used for multiple servers is called a *wild-card certificate* and will be valid for all servers in a particular subdomain, such as *.mycompany.com. There are also *extended validation (EV) certificates* for individual servers that require a more rigorous method of verification of the server owner. The purpose of EV certificates is to raise users' confidence in the validity of the assertion of ownership for a given web site (CA Browser Forum, 2014). More recent versions of the major web browsers are able to recognize an EV certificate and communicate additional information to the user. Figure 18 shows how Chrome color-codes the lock icon as well as the protocol, HTTPS, and includes the *Subject* organization name to indicate a valid EV connected web site. Clicking on the green lock icon produces the drop-down box with further details. Internet Explorer, Firefox and Safari all have similar notification mechanisms.

Figure 18


Symantec™ Trust Center

Symantec Corporation [US] https://trustcenter.websecurity.symantec.com/process/re

Symantec Corporation
Identity verified

Permissions | Connection

The identity of Symantec Corporation at Mountain View, California US has been verified by VeriSign Class 3 Extended Validation SSL SGC CA but does not have public audit records.

Certificate Information

At the other end of the spectrum from EV certificates is the domain validation (DV) certificate. A DV certificate is usually easier to obtain but provides less confidence in the true identity of the person or entity requesting the certificate (Symantec Official Blog, 2014).

7.5. Web server certificates

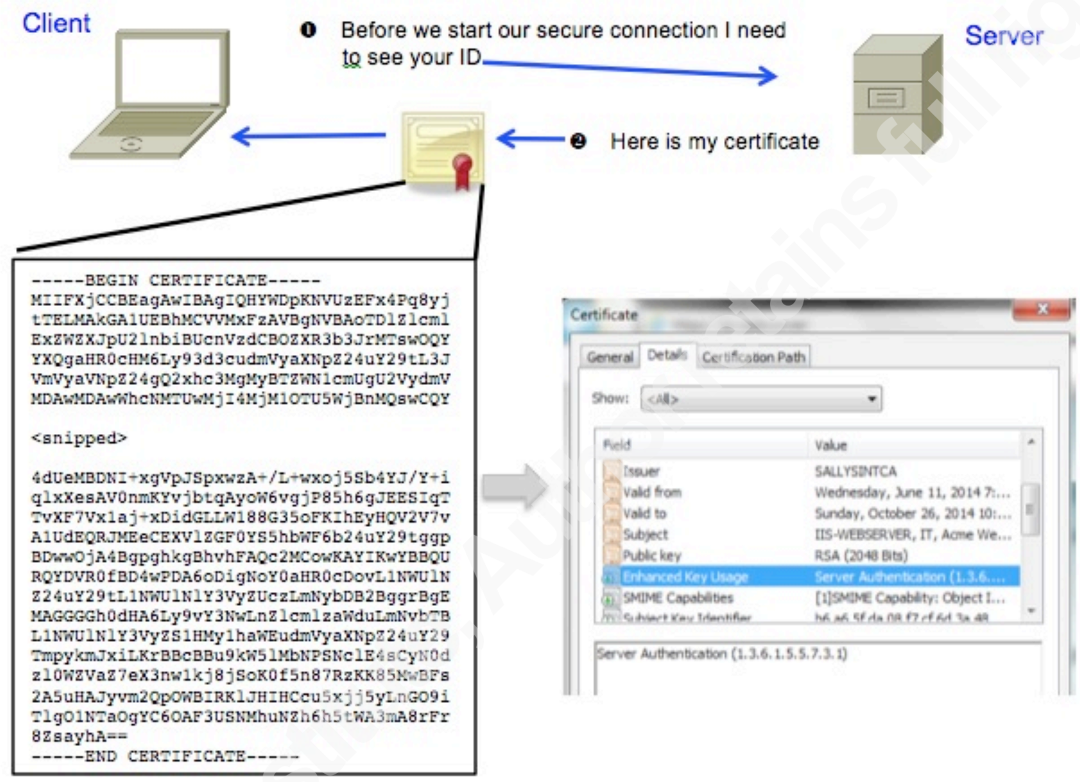
All TLS connections require that the server identify it using a certificate, however, a client certificate is optional and not often used. In other words, the client is usually not required to prove its identity to the server.

When a secure channel is setup between a browser and a web server, the server sends the client its digital certificate. This is how the server identifies itself to the client. The client, in this example a web browser, examines the server's certificate and looks for several things. First, it checks the expiration date of the certificate. If the certificate has not expired it will attempt to determine if the certificate has been revoked. There are various methods to check for revocation depending on the browser configuration but usually involves querying a revocation list or and online certificate revocation database. These methods will be discussed in detail in section 3.

Figure 19 shows a server certificate being presented to a client as part of the TLS handshake. The certificate consists of a string of numbers (shown in Base64) representing the various fields of the certificate according to the X.509 standard.

Browsers and certificate viewing applications will display these fields in a more human readable format as shown on the right in Figure 19.

Figure 19



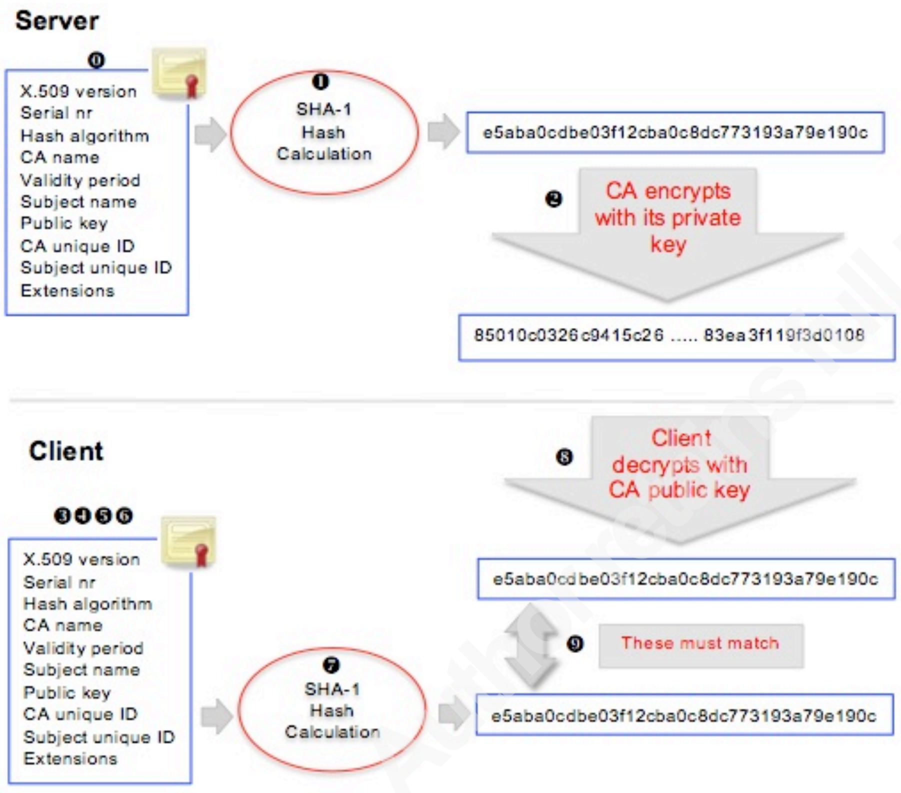
```
-----BEGIN CERTIFICATE-----
MIIFXjCCBEagAwIBAgIQHYWDpKNVUzEFx4Pq8yj
tTELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcml
ExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTswOQY
YXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL3J
VmVyaVNpZ24gQ2xhc3MgMyBTZWN1cmUgU2VydmV
MDAwMDAwWhcNMTUwMjI4MjM1OTU5WjBnMQswCQY

<snipped>

4dUeMBDNI+xgVpJSpxwzA+/L+wxoj5Sb4YJ/Y+i
qlxXesAV0nmKYvjbtqAyoW6vgjP85h6gJEESIqT
TvXF7Vx1aj+xDidGLLW188G35oFKIhEyHQV2V7v
A1UdEQRJMEeCEXVlZGF0YS5hbWF6b24uY29tggp
BDwwOjA4BgpghkgBhvhFAQc2MCowKAYIKwYBBQU
RQYDVR0fBD4wPDA6oDigNoY0aHR0cDovL1NWUlN
Z24uY29tL1NWUlNlY3VyZUczLmNybDB2BggrBgE
MAGGGGh0dHA6Ly9vY3NwLnZlcmlzaWduLmNvbTB
L1NWUlNlY3VyZS1HMy1haWEudmVyaXNpZ24uY29
TmpykmJxiLKrBBcBBu9kW5lMbNPSNclE4sCyN0d
zl0WZVaZ7eX3nw1kj8jSoK0f5n87RzKK85MwBFs
2A5uHAJyvm2QpOWBIRKlJHIHCcu5xjj5yLnGO9i
TlgO1NTaOgYC6OAF3USNMhuNZh6h5tWA3mA8rFr
8ZsayhA==
-----END CERTIFICATE-----
```

Field	Value
Issuer	SALLYSINTCA
Valid from	Wednesday, June 11, 2014 7:...
Valid to	Sunday, October 26, 2014 10:...
Subject	IIS-WEBSERVER, IT, Acme We...
Public key	RSA (2048 Bits)
Enhanced Key Usage	Server Authentication (1.3.6....
SMIME Capabilities	[1]SMIME Capability: Object I...
Subject Key Identifier	b6 a6 5f da 08 f7 cf 6d 3a 48

Server Authentication (1.3.6.1.5.5.7.3.1)

If the certificate has neither expired nor been revoked, the browser will determine if a certificate authority that it trusts has vouched for the server by digitally signing the certificate. The browser does this by checking its *trusted CA* database. If the CA is trusted, the browser proceeds to verify the signature on the certificate. It does this by decrypting the attached hash of the certificate (the digital signature) using the CAs public key and comparing that result to its own calculation of the hash. If the two hashes match then the browser has successfully authenticated the server. If not, an error or warning is returned to the user. Figure 20 shows a step-by-step illustration of a web server authenticating itself to the client's browser during the handshake phase of a TLS connection setup.

Figure 20


Server
⓿
X.509 version
Serial nr
Hash algorithm
CA name
Validity period
Subject name
Public key
CA unique ID
Subject unique ID
Extensions
❶ SHA-1 Hash Calculation
e5aba0cdbe03f12cba0c8dc773193a79e190c
❷ CA encrypts with its private key
85010c0326c9415c26 ..... 83ea3f119f3d0108
Client
❸❹❺❻
❽ Client decrypts with CA public key
e5aba0cdbe03f12cba0c8dc773193a79e190c
❾ These must match
❼ SHA-1 Hash Calculation
e5aba0cdbe03f12cba0c8dc773193a79e190c

⓿ server sends digital certificate to client to identify itself

❶ CA verifies server identity by "signing" the server's certificate. Signature on certificate derived by calculating hash over certificate fields and …

❷ …encrypting the hash with CA's private key. This signature is part of certificate

❸ client receives the digitally signed server certificate

❹ client checks expiration of server certificate

❺ client verifies that certificate's intended use is "server authentication"

❻ client checks revocation status of server certificate using revocation info in certificate

❼ client performs its own hash calculation of the server's certificate

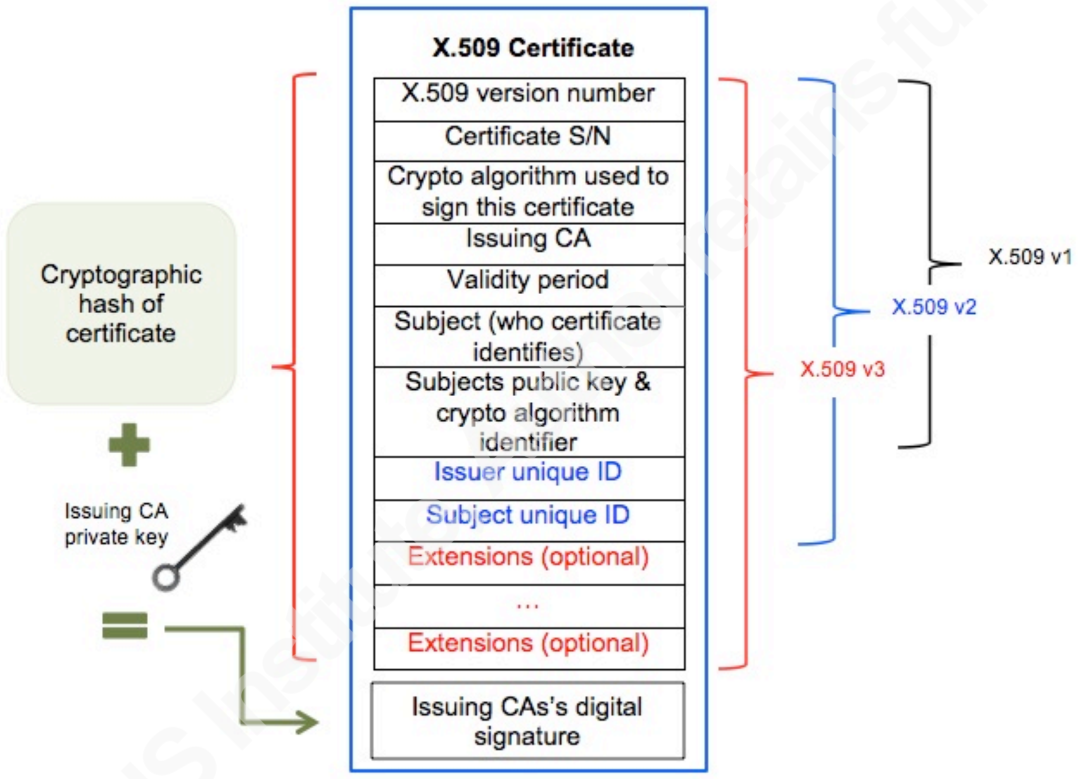
❽ client decrypts the received hash using the CA's public key

❾ Client compares its own hash calculation with decrypted hash from the CA.
If they match and CA is trusted, the server is considered successfully authenticated

7.6. Components of a certificate

The current standard for digital certificates is X.509 version 3 and will be referred to throughout this paper. There are many components of a certificate but the most relevant for this paper are the issuer (the CA that signed the certificate), the *Subject* (the resource to which the certificate applies), the subject's *Public key* and the certificate extensions, *CRL distribution point* (CDP) and *Authority Information Access (AIA)*. Both the CDP and the AIA extensions are used for certificate revocation checking and will be discussed in section 3. Figure 21 shows the fields of an X.509 certificate. Each version added new fields as shown. X.509v3 is the current version as of this writing.

Figure 21



7.7. Client certificates

Servers typically must prove their identity to clients (browsers) but it is also possible for a client to present a digital certificate as proof of identity to a server. This is not commonly implemented, however, and is out of scope for this paper.

8. Appendix B – Viewing CRLSets

The Chromium project has provided open source software to retrieve and examine CRLSets. The software can be downloaded and built according to the instructions at https://github.com/agl/crlset-tools.

Retrieve the current CRLSet as shown in Figure 22.

```
$ ./crlset fetch > crl-set-7-12-14
Downloading CRLSet version 1722
```

Figure 22

Figure 23 shows how to list the contents of the CRLSet. This shows CRLSet number 1722 and contains sets for 57 CAs. The total number of certificates contained in this CRLSet was 18840.

```
$ ./crlset dump crl-set-7-12-14 | head -6
Sequence: 1722
Parents: 57

019406d575cf285a3c2d8bbf8133e0cfae4839c99cc1815bdf244487259e7cd2
  11270b1308d38971db8f728e2956c8d38bc7
  11270b612ddbed52a12dfa3ab5c9317c486a
```

Figure 23

To view the serial numbers for one CA you may either grep the output for the SHA256 hash of the CA's public key or provide the CA's certificate as an argument to the crlset executable as shown in Figure 24.

```
$ ./crlset dump crl-set DigiCertSHA2ExtendedValidationServerCA.pem
013fe134cba9852961c290d8b5c1837f
019c0703ce5fde670d7aa251b7aa1431
01bcb31a59b15c60aea21aa5a22b49b6
       <92 entries snipped>
0fb2f22af5618bf8b4b37627050d8526
0fe6122cd048c6fe90c9a0229a6e97c0
```

Figure 24

9. Appendix C – Revocation Reasons

To determine how many certificates are revoked for administrative reasons, a calculation was done using the CRLS provided at the Internet Storm Center (ISC) (https://isc.sans.edu/crls.html). All the CRLs were downloaded, parsed and sorted by reason given for revocation. RFC 5280 states that when a certificate is revoked, reasons are not required but are "strongly encouraged", nevertheless, an examination of nearly 1.5 million certificate revocation entries since January 1, 2012 shows that nearly one third of those revocations do not include a reason. Figure 25 shows all revocations from the 250+ CRLs listed at the ISC. The total number of certificates that were revoked since January 1, 2012 is close to 1.5 million. The number of certificates that gave no reason for the revocation was about half of one million, or one third of the total. The ratio changes somewhat when only the certificate revocations since the Heartbleed vulnerability was made public on April 7, 2014 are measured in the same way. Of the 450,000 revocations between April 7, 2014 and June 15, 2015, about half of those have given no reason as shown in Figure 26.

Figure 25

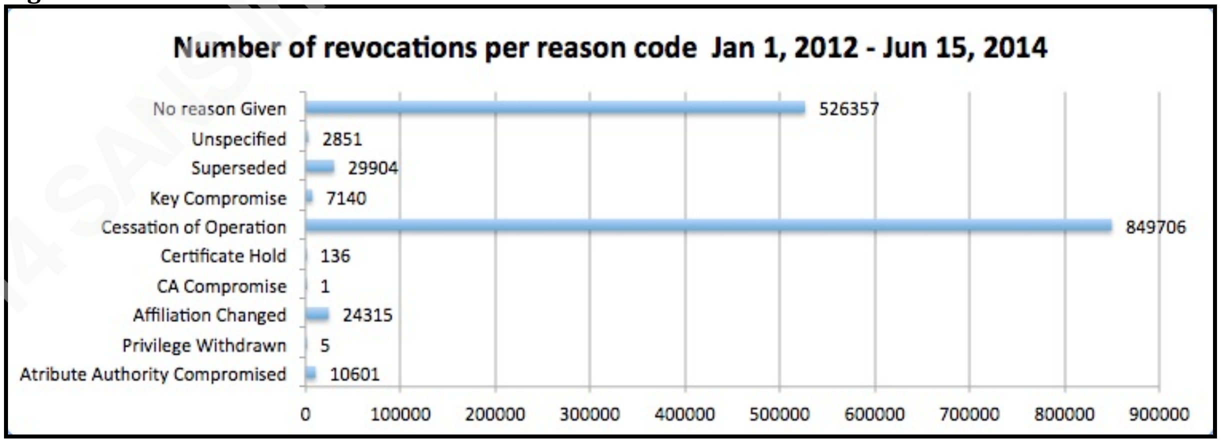
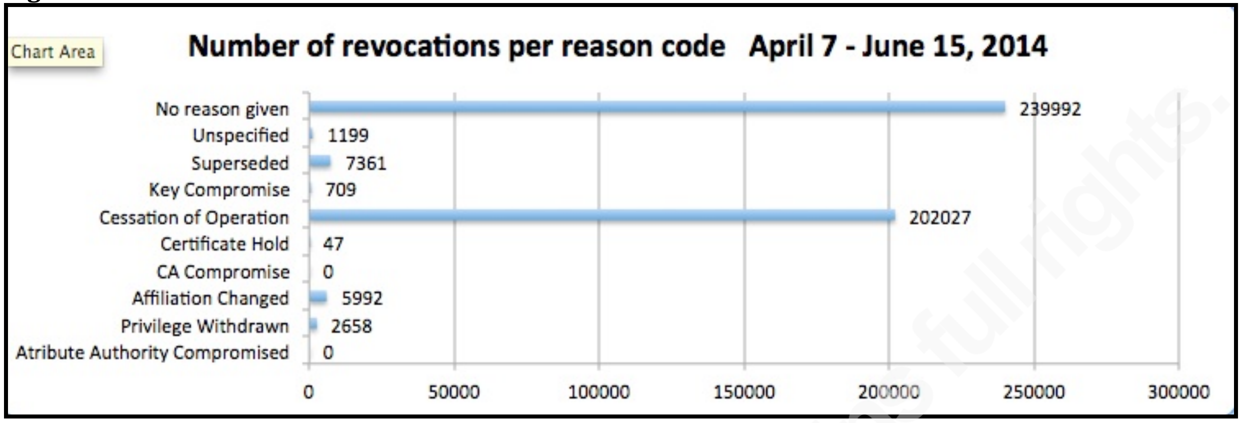


Figure 26



Additionally, three of larger CA's were contacted directly regarding their procedures for certificate revocation.

- **GlobalSign**

  Via an online chat, a Globalsign representative stated that they did not issue CRLs with reason codes. From the CRLs downloaded from ISC, Globalsign had CRLs containing 141,324 revoked certificates and 158 of those contained a revocation reason. Furthermore, none of Globalsign's 179 revoked **EV** certificates contained a reason code.

- **Godaddy**

  Via a telephone call, Godaddy reported that they always use a reason code for revoked certificates. Users access a web page to request revocation for a certificate. The page contains a drop down menu for selecting a revocation reason. The representative did not know the default value.

  The CRLs collected for this research list over 1 million revoked certificates for Godaddy. The reasons given were:

  - 96% *Cessation of Operation*
  - 2% *Affiliation Changed*
  - 1% *Privilege Withdrawn*
  - .3% *Superseded*

- .3% *Key Compromise*

- **Symantec**

  Via an online chat, a Symantec representative reported that revocations are requested via a webpage on which the user must select a reason code from a drop down menu. The representative stated that the default reason on the drop down menu was *New Key Pair*.

Of the 81,322 revoked certificates from the CRLs collected for this research, none of the revoked certificates contained a reason code.