DOI: http://dx.doi.org/10.26483/ijarcs.v10i3.6408

Volume 10, No. 3, May-June 2019

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

OVERVIEW OF TLS CERTIFICATE REVOCATION MECHANISMS

Jayanth Rajakumar
Student, Department of Electronics
and Communication Engineering
R.V. College of Engineering
Bangalore, Karnataka, India

Subrahmanya K.N.
Assistant Professor, Department of Electronics
and Communication Engineering
RV College of Engineering
Bangalore, Karnataka, India

Abstract: TLS Certificates are the backbone of the World Wide Web's Public Key Infrastructure. In case of a compromise of private cryptographic keys, it is vital to have the ability to revoke certificates before their validity period expires. This paper describes and contrasts the two major mechanisms for certificate revocation – Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP). It is found that modern web clients and browsers such as Google Chrome do not perform stringent checking of certificate revocation status, leaving users open to attackers who use revoked certificates to spoof web sites and services. A browser extension is proposed and implemented for Google Chrome that checks CRL and OCSP status and notifies the user. It can also automatically navigate away from the page if the certificate is found to be revoked. The extension is created using JavaScript and uses a background process written in Python to handle the revocation checking. It is found to be able to complete CRL and OCSP requests for common websites in under a second, and under 200 milliseconds for locally cached responses.

Keywords: TLS, SSL, HTTPS, X.509, Digital Certificates, Certificate Revocation

I. INTRODUCTION

Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) secure a large proportion of the websites we visit on the Internet. From looking up information to conducting bank transactions, TLS ensures that users' private data is not intercepted by unauthorized parties. TLS relies on the Public Key Infrastructure (PKI) defined by the X.509 standard in order to provide authentication and to facilitate encryption. Web servers use X.509 certificates to prove their identity to clients such as web browsers. Certificates are issued to web domain owners by well-known authorities called Certificate Authorities (CAs) and are typically valid for a period of a few months to years. In order to maintain the security of the PKI, it is necessary for the CAs to be able to revoke the certificates before they expire.

The most common reason for this is when the private key corresponding to the certificate gets stolen or compromised. In such situations, the CA can be notified and requested to revoke the certificate. Several mechanisms exist for the client to be made aware that the server certificate has been revoked, the most common ones being Certificate Revocation Lists (CRL) and Online Certificate Status Protocol (OCSP). However, it is seen that a large number of clients, including web browsers, either do not respect the revocation status or do not check it in the first place. The commonly stated reasons for this are the high cost associated with checking the revocation status and the fact that CAs are not diligent in updating revocations. Unfortunately, this leaves users vulnerable to attackers who use revoked certificates to spoof their identity and intercept users' data.

This paper first discusses the working of digital certificates and how they are used to prove the authenticity of web servers on the internet. Next the two major methods of revocation – OCSP and CRL are described in detail, and their pros and cons are discussed. Finally, the paper proposes a web browser extension which independently performs revocation checking and notifies the user, optionally redirecting away from the offending website if its certificate is found to be revoked.

II. DIGITAL CERTIFICATES

A digital certificate is a document used to prove ownership of a public key. During the handshake phase of the TLS protocol, the server sends its certificate to the client to authenticate itself. The certificate also serves as a means of sharing the public key and the algorithm to be used in order to exchange the symmetric secret key for the application phase of the TLS session.

On the Web, certificates are purchased by web server owners from companies called Certificate Authorities (CAs), which digitally sign the certificates that they issue. CAs possess a self-signed certificate called root certificate that is implicitly trusted by clients such as browsers and operating systems. During a TLS handshake, the client must validate the incoming server certificate against the list of Root CAs by verifying its digital signature [1]. Next it uses the public key present on the certificate to encrypt and send the symmetric key which is used for the actual data transfer.

The de facto standard for digital certificates on the internet is X.509, which is defined in RFC 5280 along with the web's PKI [2]. Some of the fields present in an X.509 certificate include Serial Number, Subject, Issuer, Validity, Public Key and Signature. In addition, there are optional fields called extensions which provide optional information such as the intended usage of the certificate, the domains where the certificate can be used and where to obtain revocation information. The subject includes the details of the domain where the company is used, while the Issuer refers to the details of the CA which issued the certificate. The issuer is also responsible for appending the signature field, which is the hash

of the certificate encrypted with the issuer's private key. When a client has to verify the certificate, it does so by decrypting the signature using the issuer's public key and comparing it with the observed hash. Since only the issuer is assumed to be in possession of the private key, matching hashes indicates that the certificate's contents are exactly as they were when the CA issued them.

The X.509 standard allows for a chain of certificates where the root certificate is signed by an CA intermediate certificate, which in turn is signed by a CA's root certificate. For security reasons, only the intermediate keys are used in the day to day operation of the CA to issue certificates, and the root certificate's private key is kept offline. In the event of a key compromise, only the certificates issued by a particular intermediate certificate will be affected.

When a CA issues a certificate for a website, it ensures that the entity requesting the certificate actually controls the domain, in a process called Domain Validation (DV). This is typically done by having the claimed domain owners publish a random string (nonce) on the domain address or publish a DNS TXT record. For more stringent verification, Extended Validation (EV) certificates are used. To obtain an EV certificate, a domain owner must be able to prove their legal existence as a company. An EV certificate cannot be applied to more than one domain or subdomain. When a website is using an EV certificate, the browser shows the issuer's name prominently next to the address bar, indicating that the website is to be trusted.

III. CERTIFICATE REVOCATION

CAs allow for certificate owners to request the certificates to be revoked before their validity has been expired. Reasons for doing so can include – compromise of the private key, closure of the company operating the website or errors in the issued certificate. The CAs respond to the requests by marking the certificate as revoked. This does not stop web servers from continuing to use the revoked certificate. Y. Liu *et al.* [3] found that 1% of the certificates surveyed were being advertised in spite of being revoked. Clients must be made aware of the revocation when validating the certificate during the TLS handshake, and refuse the connection. There are two major methods of conveying the revocation status from the certificate authority to the clients – Certificate Revocation List (CRL) and Online Certificate Revocation Protocol (OCSP).

A. CRL

A CRL is a file issued by a CA with a list of certificates issued by them that have been revoked [2]. It specifies the serial number of the revoked certificate, the reason for revocation, and the time when the certificate was revoked (such as key compromise, CA compromise or cessation of operation of the subject). The time when the CRL was issued is indicated, as well as the time of the next update. The file is signed using the issuer's private key, similar to the certificate itself.

In order to verify a certificate's revocation status, a client must download the CRL file of the CA from a URL specified as an extension in the certificate. If the server certificate's serial number is present, then it is found to be revoked. Additionally, the CRL's signature may be verified to ensure that the list is authentic.

Since CRLs are large and require the download of information related to all the certificates issued by a CA, they are very inefficient. To reduce the size of the data to be downloaded, delta CRLs can be used [2]. Delta CRLs contain only updates to previously distributed CRL information. This reduces network load and processing times. The location of a delta CRL is specified as an extension to the base CRL, called Freshest CRL. The base CRL can be cached on the client machine, and subsequently only the delta CRLs are downloaded. However, delta CRLs are rarely used in practice.

CRLs introduce a dependency on the CA server and are prone to being affected by its slowdowns and down times. But since CRLs are typically valid for a period of a few weeks, they may be cached by the client and used to verify different certificates issued by the same domain.

B. OCSP

OCSP is a protocol for determining the status of a certificate without requiring the use of CRLs [4]. Figure 1 shows the flow of the protocol. A client sends a HTTP request to a dedicated OCSP responder maintained by the certificate issuer. The domain name of the responder is obtained from an extension in the server certificate. The response to the request indicates if the certificate is good, revoked or has an unknown status, and this can be used to make a decision on whether or not the web domain is to be trusted. Responses are optionally signed using the issuer's private key, similar to certificates and CRLs.

An OCSP request contains the serial number of the certificate, the hash of the issuer's domain name and the hash of the issuer's public key, in the specified ASN.1 format. The encoded requests are converted to base-64 representation and sent to the responder as part of a GET request. The response, when decoded, gives the status of the certificate as good, revoked or unknown, as well as the time when the status will be updated next (NextUpdate).

- The "good" state indicates that no certificate with the requested serial number is currently revoked.
- The "revoked" state indicates that the certificate is no longer to be trusted, and clients typically consider the certificate validation as failed and terminate the handshake.
- The "unknown" state indicates that the revocation status could not be verified, usually because the responder does not serve the issuer of the certificate. In this case, the client can choose to accept the certificate, reject the certificate or try to verify it using CRLs.

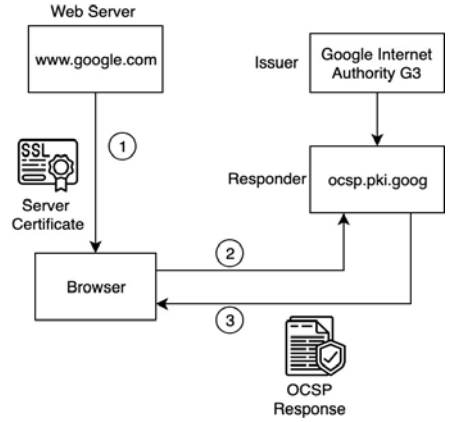

Figure 1. Working of OCSP

Compared to CRLs, OCSP suffers from privacy issues since it involves transmitting the request in plaintext, and

potentially revealing the sites being visited. To solve this issue, the concept of OCSP stapling (depicted in Figure 2) was introduced [5]. An extension called Certificate Status was added to the certificates, which can be used by bandwidth constrained clients to obtain the certificate status without having to contact the responder on their own, saving roundtrips and resources. Web servers periodically poll the CA's OCSP responder to retrieve the response and send it directly to the client during the handshake.

In order to use OCSP stapling, clients must include a Status Request extension in their ClientHello message. This causes the server to return the revocation status immediately after its Certificate message, in the same format as regular OCSP. Since the response has limited validity and is signed by the CA, clients can validate it before trusting it.

However, the issue with stapling is that if an attacker gets hold of the certificate and the private key, they can simply ignore the Status Request and refuse to staple the status response. To ensure that the server always returns a status response, the Must-Staple extension is used [6].

For this system to work, the CAs must include the Must-Staple extension in the server's certificate at the time of issuing it. When clients receive this certificate during the handshake, they must recognize the extension, and terminate the connection unless the revocation status is also transmitted by the server. So, in the case of a website's private key being stolen, the attacker is forced to send the OCSP response obtained from the responders. When the web domain owner detects the attack and asks the CA to revoke the certificate, the attacker will have no option but to send the revoked status to the client, which drops the connection.

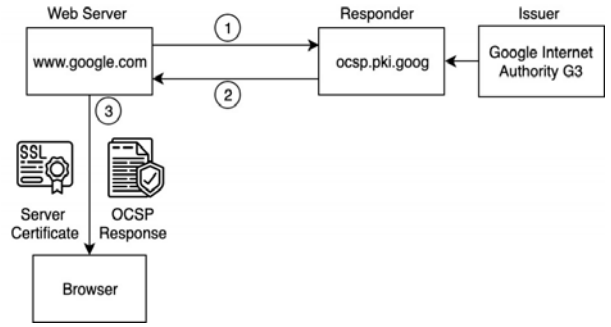

Figure 2. OCSP Stapling

C. OCSP vs CRL

Both OCSP and CRL have their advantages and disadvantages. CRLs are updated only every 7-14 days, hence recently revoked certificates may not be included. But this allows clients to keep a cached copy of CRLs for longer and use them for subsequent connections to the same server, which reduces the latency caused by contacting the CA in the middle of the handshake. If a client has the resources to cache a large number of CRL responses, and the initial performance hit when a new CRL is downloaded is acceptable, then revocation checking can be performed extremely quickly and reliably while also providing privacy.

OCSP is light weight compared to CRL as the size of information retrieved is smaller. It also delivers more up to date information as OCSP responders are updates more often that CRL distribution points. However, it is susceptible to a variety of security flaws, such as replay attacks and man-in-the-middle attacks. The CA's infrastructure responsible for handling the OCSP requests tends to be poor with high latency [7]. The privacy of the user is also compromised, as the browser must, by design, reveal to the CA what domain is being accessed.

Most OCSP client implementations tend to have a soft-fail behaviour – if a response is not received in time, then the connection proceeds instead of being terminated as in hard-fail behaviour. The most common internet browser, Google Chrome, does not support OCSP. Instead, it relies on a limited implementation of CRL which often does not correctly identify revoked certificates.

OCSP with Must-Staple is theoretically the most secure and accurate way for clients to obtain revocation status. However, its adoption remains low. Out of the 1 million top domains on the web, only 0.01% have the Must-Staple extension in their certificates. [8]. Most web browsers also do not check if the response was actually included with the certificate. OCSP stapling is often poorly implemented by server software, which can lead to users getting locked out of websites even when the certificate is valid.

Thus, there are many factors such as privacy, reliability, latency and accuracy which must be considered before picking a method for revocation. If OCSP with Must-Staple sees widespread adoption by both servers and clients, then it would have an edge over the other mechanisms.

D. Certificate Transparency

Both OCSP and CRL do not address the situation where a CA is compromised and taken over by an attacker. If someone possesses the private key of the CA, they can issue certificates in the name of any domain and use them for malicious purposes. There is no mechanism to keep track of the certificates issued for a domain. To address this issue, the Certificate Transparency (CT) protocol was created by Google [9].

It uses Certificate Logs, which are publicly available records of certificates that CAs can write to. Logs can be queried for cryptographic proof that a particular certificate has been logged. There are 'Monitor' servers which periodically communicate with the Log servers to watch for unauthorized certificates. TLS clients have 'Auditor' software which checks the consistency of the logs, as well as verifies if a particular certificate is in the chain. If a browser finds that a CA's certificate has not been logged to the system, then the connection may be dropped.

When a CA submits their certificate to the CT logs, it receives a Signed Certificate Timestamp (SCT) in response, which states that the certificate will be added within a particular period of time. The SCT is sent by the web site to the clients while making a TLS connection. This can be done through a X.509 certificate extension, a TLS message extension, or along with the stapled OCSP response. The client verifies the signature on the SCT to verify that it came from a legitimate log, and whether it was issued for the same certificate that was received from the server.

CT can also be used by website administrators to keep track of all the certificates issued for their domains. If a third party compromises a CA and obtains a certificate for that domain,

then it can be observed through the public logs and the appropriate action can be taken.

E. Revocation checking in modern browsers

Y. Liu *et al.* [3] created a test suite to thoroughly test the revocation checking behavior of various web browsers. It was found that no browser by default checks revocation status for all certificates in the chain. Many browsers do not check the status at all or treat an unknown status as good (soft fail behavior).

Google Chrome on OS X does not check any revocation information for regular certificates. For Extended Validation certificates, it checks OCSP and CRL, but if the OCSP returns a revoked status, then instead of dropping the connection it attempts to check the CRL instead. This means that users are left vulnerable to attackers using revoked certificates, especially since Google Chrome is the most commonly used browser in the world. Other browsers like Firefox and Opera vary slightly in what certificates they check the status for, but none of them have a strict process. Most mobile browsers, which serve a lot of internet traffic today, do not even have the option of revocation checking.

In 2013, Google introduced a proprietary revocation mechanism called CRLSets [10] in Google Chrome. CRLSets are small sets of revoked certificates maintained by Google which are automatically pushed to the browser and used to validate web servers' certificates. By limiting the CRLSet file to 250 kilobytes, the process of validation is quick without consuming a lot of network or processor resources. However, this means that it can only include a subset of all the revoked certificates on the web. Y. Liu *et al.* [3] found that only 0.35% of the revoked certificates in their dataset appeared in Chrome's CRLSet. This means that the system is largely ineffective.

IV. GOOGLE CHROME EXTENSION FOR REVOCATION CHECKING

As discussed in the previous version, Google Chrome is particularly lax when it comes to checking server certificates for revocation. Users have no way of knowing for sure whether the site they are browsing has had its certificate revoked or not. This section proposes a browser extension to independently check the OCSP and CRL revocation status of any HTTPS pages visited and display the result to the user. The user can configure the extension to check CRL, OCSP or both. If the certificate is found to be revoked, then the extension can immediately redirect away from the page.

A. Design

Google Chrome's JavaScript extensions API does not provide access to the TLS certificate of the page being downloaded. It also does not allow the TLS handshake to be paused to perform any custom operations. Hence a Python based approach is proposed, as shown in Figure 3. Python has a wide variety of libraries and can make independent TLS connections to the web server and get its certificate for verification.

When a new page starts loading, the JavaScript browser extension sends a HTTP GET request to a Python server running on localhost. The request contains the domain address of the website visited. The Python program performs its own handshake with the website and retrieves the X.509 certificate. From the extensions, the CRL distribution URL and the OCSP responder can be obtained. The CRL file can be downloaded from the URL, and the serial number of the server certificate verified against it. Also, a HTTP request can be sent to the OCSP responder to get the revocation status. After verifying the status, the response is returned to the JavaScript extension, and action can be taken depending on the user's preference.

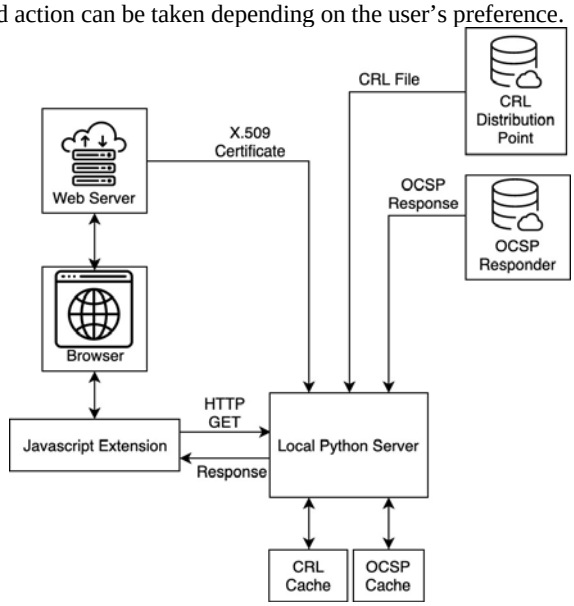

Figure 3. Block diagram of the proposed system

B. Working

- **JavaScript Extension**: This is a lightweight extension that extracts the hostname from HTTPS pages visited. It is based on [11] which uses a third party server to display certificate validation level. Once any webpage starts loading, it creates a HTTP GET request to localhost:8000 with the URL as the hostname of the visited page. Once the response is received, it changes icon of the extension to indicate if the certificate is valid or revoked. A popup window is displayed when the icon is clicked. This icon shows the certificate's organization name, the issuer's organization name as well as the revocation status.

  It provides options for the user to choose between CRL, OCSP and both, as shown in Figure 4. This preference is sent as a header in the GET request. The user can also specify the page should be redirected when a revoked certificate is observed. Hard fail behaviour can be configured, where an unresponsive responder is treated as a revoked certificate.

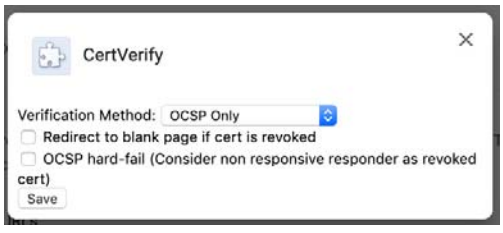

Figure 4. Options window of the extension

- **Python Server:** It is a HTTP server running locally on port 8000 (chosen arbitrarily). The reason for choosing this design is that it the server can be easily moved to

another machine on a local network. This allows the revocation checking software to be shared by all the devices on the network.
Once the GET request is obtained from the extension, the Python program makes a TLS connection to the hostname and retrieves the certificate chain of the web server. By parsing the ASN.1 fields, details such as serial number, subject name and organization can be obtained. The CRL distribution points and the OCSP responder hostname can be retrieved.

If the user has requested CRL verification, the CRL file is downloaded to the local storage and parsed to read the serial numbers. The serial numbers are placed into a Python Set object which provides fast lookup using hash tables. By checking if the serial number is present in the set, it can be ascertained if the certificate is revoked. The HTTP response is created in the form of a Json array consisting of the subject and organization names and the revocation status. After sending the response, the set object is serialized and saved to the disk (CRL Cache) along with the NextUpdate field value. For subsequent requests, the server checks the disk for the file first. If it is found and still valid, then this saves the bandwidth of downloading the file again, as well as the processing time of traversing it.

If OCSP was requested, then an OpenSSL request is created using the certificate chain and send to the responder. The OCSP response directly indicates the revocation status, and this is sent back to the browser extension in the form of a Json string. Since the OCSP response is generally valid for a few days, the response file is saved to disk. On subsequent connections, if the response is still valid, there is no need to send a request to the responder again. This cache system helps to improve the response time of the system.

V. RESULTS AND DISCUSSION

The browser extension was installed on Google Chrome version 74 on macOS and the Python server was started up in the background from the command line. The revocation status was observed from the extension's icon on the address bar.

A. *Browser Screenshots*

Figure 5 shows the browser extension's popup window for google.com when the CRL and OCSP checks are successful.

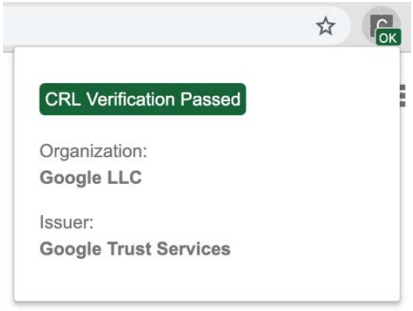

Figure 5. Extension popup window on successful verification

Figure 6 shows the window when visiting the website https://revoked.ecert.gov.hk/, which has a revoked certificate. Here the 'Redirect to blank page' option was turned off, and only OCSP check was performed.

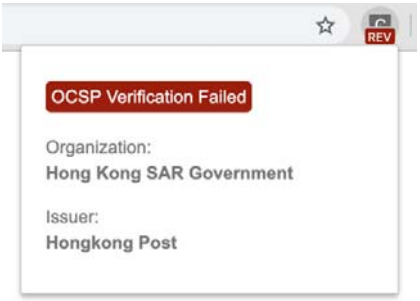

Figure 6. Extension popup window when the certificate is revoked

B. *Extension Response Time*

Quick revocation checking is important for a better user experience. The time taken for the Python server to process the GET request was measured from the JavaScript extension by noting the time stamp before and after the request. Five different websites were chosen from the top ten most visited websites on the Internet [12]. The average response time for checking OCSP, OCSP Cache, CRL and CRL Cache was calculated for these websites over five trials. The graph of the results is shown in Figure 7. It was found that downloading a fresh CRL and processing it takes the most amount of time, but is still within a second. Checking the previously downloaded CRL or OCSP from the local cache (assuming a cache hit) is the fastest, with response times less than 200 milliseconds. OCSP response time is in between the other cases.

This analysis reinforces that fact that caching revocation responses locally is a good way to speed up connections and improve the user experience.

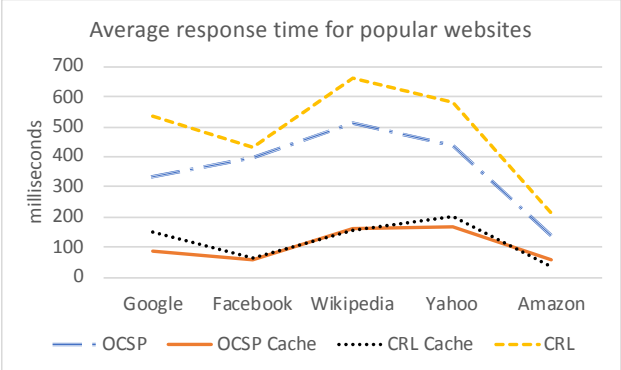

Figure 7. Average response times of the extension measured for five websites

C. *Future Work*

There are many improvements that can be made to the extension to extend its functionality. The current implementation only checks the revocation status of the server (leaf) certificate, and behaves identically for EV and non-EV certificates. Literature survey shows that different browsers perform revocation checks differently depending on the certificate validation type, position in the chain and user preferences [3].

The extension can be modified to support different browsers and offer the checks that are missing. For example, Google Chrome on Windows only checks CRLSet for non-EV leaf certificates. As CRLSet is severely limited in terms of number of revoked certificates covered, the extension can perform OCSP or CRL for both the leaf and the intermediate certificate to protect against compromised CAs. For EV certificates, there is no need for any additional checks as Chrome performs them by default.

To improve cache performance even further, a limited number of most recently accessed revocation check results can be kept in memory instead of on the disk. If it is not found in memory, then the disk can be checked, or else it has to be fetched again.

VI. CONCLUSION

This paper studied how TLS certificates are revoked and the mechanisms that are used to convey the revocation status to web clients – CRL and OCSP. OCSP with Must-Staple extension was found to be the most secure method of checking revocation, but it is not widely adopted. Server software developers, browser developers and website administrators need to upgrade to use the latest developments in revocation protocols.

To improve users' security, a Google Chrome Extension was proposed and implemented to verify the CRL and OCSP status. The response time of the extension for a few select websites was analysed and it was found that caching responses gives the best results.

VII. REFERENCES

[1] M. Nia, A. Sajedi and A. Jamshidpey, "An Introduction to Digital Signature Schemes", In Proceeding of National Conference on Information Retrieval, 2011

[2] D. Cooper et al. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Internet Engineering Task Force RFC 5280, May 2008.

[3] Y. Liu et al., "An End-to-End Measurement of Certificate Revocation in the Web's PKI", Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15, 2015. DOI: 10.1145/2815675.2815685

[4] S. Santesson, A. Malpani and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", Internet Engineering Task Force RFC 6960, June 2013.

[5] D. Eastlake, "Transport Layer Security (TLS) Extensions: Extension Definitions", Internet Engineering Task Force RFC 6066, January 2011.

[6] P. Hallam-Baker, "X.509v3 Transport Layer Security (TLS) Feature Extension", Internet Engineering Task Force RFC 7633, October 2015

[7] L. Zhu, J. Amann and J. Heidemann, "Measuring the Latency and Pervasiveness of TLS Certificate Revocation", Passive and Active Measurement, pp. 16-29, 2016. DOI: 10.1007/978-3-319-30505-9_2

[8] T. Chung et al., "Is the Web Ready for OCSP Must-Staple?", Proceedings of the 2018 ACM Conference on Internet Measurement Conference - IMC '18, 2018. DOI: 10.1145/3278532.3278543

[9] B. Laurie, A. Langley, E. Kasper, "Certificate Transparency", Internet Engineering Task Force RFC 6962, June 2013.

[10] A. Langley, "Revocation checking and Chrome's CRL", 2019 , [Online]. Available at https://www.imperialviolet.org/2012/02/05/crlsets.html.

[11] Y. Li, "certificate-info", GitHub, 2019. [Online]. Available: https://github.com/blupig/certificate-info

[12] "Keyword Research, Competitor Analysis, & Website Ranking", Alexa Internet, 2018. [Online]. Available at https://www.alexa.com