Online Certificate Status Protocol (OCSP)

RFC 6960

W.J.B. Beukema

June 2015

1 Introduction

The *Online Certificate Status Protocol* (OCSP) is a Proposed Standard that enables applications to determine the revocation state of X.509 certificates [7]. Thus, OCSP is a means to inform clients about updates in the chain of trust in Public Key Infrastructures (PKIs).

1.1 Background

In systems that rely on Public Key Infrastructures, such as TLS/SSL, certificates are issued by Certificate Authorities. Even though certificates contain an expiration date on which the certificate will no longer be considered as valid, it might occur that the certificate's trustworthiness is damaged before that date. For instance, a certificate might be issued to the wrong person or institute, or the private key of the certificate might get compromised, allowing unauthorised people to use the certificate for signing. Such certificates should be no longer trusted by applications that rely on them.

A recent example would be the *Heartbleed* bug, which made it possible to steal private keys from vulnerable servers. It has been estimated that over 500,000 certificates could have been compromised, and should have been revoked as a result [8].

In general, the challenge is to communicate the revocation of certificates as effectively as possible. Ideally, applications should be able to determine as quickly and efficiently as possible whether or not a certificate has been revoked or not.

1.2 History

The predecessor of OCSP as a means to revoke certificates is the *Certificate Revocation List* (CRL) mechanism [2, 4]. These lists (the CRLs) are issued by the Certificate Authority which issues the corresponding certificates. A CRL contains the serial number, revocation date and revocation reason of all certificates of that Certificate Authority that are either temporarily revoked or permanently revoked. Applications that want to use a certificate should check (prior to its use) whether the certificate's details are not on the Certificate Authority's CRL, in order to determine whether or not to rely on it. A CRL is usually generated and published periodically, and have a lifetime during which they are valid (often 24 hours or less).

The CRL mechanism suffers from some serious issues. For large Certificate Authorities, the CRL might grow quite large over time. This results in an unnecessary waste of bandwidth and time as a result. A slight improvement is possible using delta-CRLs (RFC5280). Another disadvantage is the update mechanism, as it is not real-time. If a certificate is revoked by the Certificate Authority, it can take several hours before clients have obtained the updated CRL. In the hours in between, the revoked certificate will be considered to be valid by clients that have not updated the CRL: this might cause serious problems. Finally, according to the specification or CRL, if a CRL could not be obtained the client should accept the certificate. This again is a

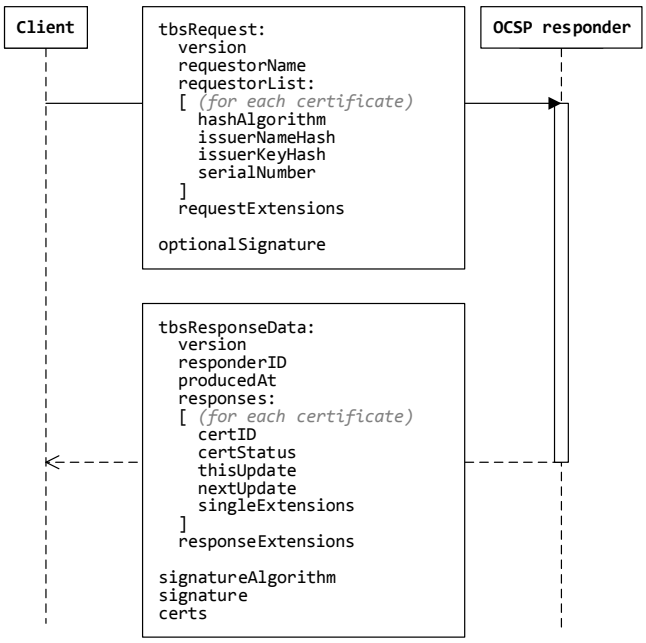

Figure 1: A general overview of the exchange of messages in OCSP as defined in RFC 6960 [7].

vulnerability in the protocol: if the CRL server is down or blocked, a certificate might considered to be valid even though it is revoked.

2 The protocol

Using OCSP, applications can determine the state of identified certificates. Whereas CRL was designed in a time when the Internet was small and didn't take scalability into account, OCSP is designed with the current state of the Internet in mind. As we will see, it scales better to large networks compared to CRL and the determination of the certificate's status happens in real time. Another difference with CRL is that OCSP only accepts a certificate after it received a valid response.

2.1 Overview

The protocol consists of two parties: the *client*, the party that wants to determine the revocation status of a certificate, and the *OCSP responder*, the party that has authority over the revocation status of certificates.

A general description of the protocol can be given as follows:

0. The client receives the public key certificate of a service he wishes to use, for instance a TLS/SSL certificate.
1. The client creates a OCSP request, containing (amongst other details) the serial number of the public key certificate. He sends the request to the responsible OCSP responder, usually the Certificate Authority that issued the certificate.
2. The OCSP responder checks if the request is well-formed and looks up the serial number provided in it. It might decide the certificate is:

good Indicates the certificate is currently not revoked. The user can use the certificate.

revoked Indicates the certificate has been revoked, either temporarily or permanently. This response might also indicate the associated Certificate Authority has no record of ever having issued the certificate. `revoked` certificates should be rejected by the client.

unknown Indicates the OCSP responder cannot determine the status of the certificate. This might occur when the request is sent to the wrong OCSP responder. The client should decide whether or not to try another source of status information.

3. The OCSP responder creates a OCSP response, containing (amongst other details) the serial number of the public key certificate together with its status (`good`, `revoked` or `unknown`). Also, for the status of each certificate it is indicated until when the client may cache the certificate's status. The OCSP responder signs the request and sends it back to the client.

4. The client verifies the signature (it should be signed by an authoritative party and the signature should be valid), makes sure that the response is well-formed and checks that the received certificate's status is still valid (i.e. has not expired). Subsequently, it decides whether or not to rely on the certificate based on the OCSP response it received.

2.2 Security considerations

2.2.1 Cryptographic aspects

The specification of OCSP states that clients should at least support RSA with SHA-256 signatures, as well as RSA with SHA-1 signatures and DSA with SHA-1 in order to verify the signature of an OCSP response [7, p. 18]. The client may also support other algorithms that the OCSP responder can use to sign the OCSP response. The client can indicate which cryptographic algorithms it prefers using the `requestExtension` field [7, p. 21-24].

It is however the responder's responsibility to ensure that the chosen cryptographic algorithm is sufficiently strong, even if the client requests weaker cryptographic algorithms. It is recommended for the signing algorithm to be at least as secure as the signing algorithm that was used to sign the certificate of which the status is queried [7, p. 26].

2.2.2 Signing

The OCSP responder is, as mentioned before, usually the Certificate Authority. The OCSP response is typically signed directly by the Certificate Authority that issued the certificate of which the status is queried. However, it is also possible to let a delegated party (i.e., that did not sign the certificate) sign the OCSP response. In this case, the certificate the OCSP responder uses to sign OCSP responses should be signed directly by the Certificate Authority [7, p. 15] and it must proof it is authorised to sign OCSP responses by having the `id-kp-OCSPSigning` property in its certificate.

Optionally, the client can prove his identity to the OCSP responder by signing the OCSP request, using the `optionalSignature` field [7, p. 11-12]. The OCSP responder can require clients to sign their requests in order to authenticate them by sending back an error message upon receiving an unsigned OCSP request.

2.2.3 Replay attacks

As all required fields of both the request and the response are (to some extent) static, the protocol is vulnerable to a replay attack. For instance, a `good` OCSP response may be replayed to a client after the certificate has been revoked: this works as long as the OCSP response has

not expired. As a response typically are valid several hours, this attack could be used in practice. In order to overcome this, the client may choose to include a nonce in the `requestExtensions` field of the OCSP request to make sure he receives a fresh response. However, including a nonce is not required and may not be supported at the client side and/or responder side.

Making nonces optional is a design choice that has been made in order to improve performances at the side of the OCSP responder. Without nonces, the OCSP responder can pre-generate responses in advance of a request, which lowers the burden on the responder's side. Since an OCSP response does not contain details of the original requester, a single pre-generated response can be used in responses to multiple requesters.

2.3 Usage

An example of the application of OCSP can be seen in the TLS/SSL system. All major Web browsers (including Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari and Opera) support OCSP in their most recent versions. All of them, for the exception of Google Chrome[1], enable OCSP by default for each request.

For OCSP to work, Certificate Authorities should support OCSP as well by having OCSP responders. As of 2015, most Certificate Authorities support OCSP [5].

3 Criticism

Even though the protocol is a major improvement of its predecessor CRL, there are some debatable aspects of OCSP.

3.1 Burden on popular OCSP responders

OCSP might impose a significant burden on Certificate Authorities that issue a large number of certificates or that have issued certificates to popular websites. As every client will individually contact the OCSP responder to obtain the certificate's status, the OCSP responder might get flooded with requests. This might result in the OCSP responder being unreachable at peak hours or will force the OCSP responder to invest in more bandwidth.

However, as mentioned before, OCSP allows a responder to have pre-generated a response for (popular) queries, which lowers the computational burden (e.g. for computing the signature).

3.2 Burden on clients

OCSP might reduce the client's performance of using secure services compared to CRL, as for every secure service it might need to contact the OCSP responder, send the request, get the response and verify the response. Although the overall performance of OCSP at the client side might be better than CRL, it might slow down opening a secure connection for the first time.

3.3 Privacy issues

The OCSP system also raises some privacy concerns. OCSP requires the client to query the status of a certificate at the responsible OCSP responder. As the client usually will send out the request at the point that the user needs the certificate, the request might disclose sensitive information about the client to the OCSP responder: for instance, in the case of TLS/SSL, it might reveal to the OCSP responder which websites the client is accessing at what times.

Because of these privacy issues as well as the latency issues, Google Chrome disables OCSP by default [1].

[1] See section 3.3.

4 Suggested improvements

In order to solve some of the problems as mentioned in the previous section, *OCSP Stapling* has been suggested as an improvement [3, 6]. OCSP Stapling builds upon the OCSP standard, although the protocol itself is changed fundamentally. The most important difference with OCSP is that the party that is providing the public key certificate should also provide a valid OCSP response (signed by an authorised OCSP responder), e.g. as part of the TLS handshake. As a result, the client does not need to contact an OCSP server, but just has to verify the OCSP response it received.

By this means, most privacy-related concerns are solved, as the OCSP responder cannot directly link a request to an individual client. It also lowers the burden on the OCSP responders, as a single OCSP response can be used for multiple clients by the certificate provider. As a result, there will be less traffic to OCSP responders. Finally, it also lowers the burden on clients, as they do not have to contact the OCSP responder themselves, which saves time and bandwidth and hence improves performance and reduces overhead.

All in all, OCSP Stapling is a promising improvement of OCSP that - when it is fully deployed on all levels - can optimise the certificate revocation process substantially.

References

[1] Constantin, L. Google Chrome will no longer check for revoked SSL certificates online. `http://www.computerworld.com/article/2501274/desktop-apps/google-chrome-will-no-longer-check-for-revoked-ssl-certificates-online.html` [Accessed May 27, 2015], feb 2012.

[2] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[3] Eastlake, D. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), Jan. 2011.

[4] Housley, R., Polk, W., Ford, W., and Solo, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), Apr. 2002.

[5] Netcraft Ltd. OCSP sites ordered by failures over the last 1 day. `http://uptime.netcraft.com/perf/reports/performance/OCSP` [accessed June 2, 2015], jun 2015.

[6] Pettersen, Y. The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961 (Proposed Standard), June 2013.

[7] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (Proposed Standard), June 2013.

[8] Zhang, L., Choffnes, D., Levin, D., Dumitraş, T., Mislove, A., Schulman, A., and Wilson, C. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC* (2014), pp. 489–502.