X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP

Inspired by RFC 2560

Maikel Zweerink

OCSP and its PKI aspects

Public Key Infrastructure is crucial in today's use of the internet. PKI is a infrastructure with the means to manage (create, validate, revoke) digital certificates within that system. Within a Public Key Infrastructure certificates are used to ensure confidentiality and integrity between entities. The biggest example of an PKI is websites secured via SSL over HTTP (HTTPS).

Digital certificates can be created by anyone, but this does not mean they will be valid within an PKI. To make a certificate become valid within an PKI, a certificate authority (CA) needs to verify that you are allowed to own this certificate. Once this validation succeeds the CA signs your certificate with their root certificate. A list of all these root certificates is present on every system that will use the PKI. This way any system can verify a certificate is trusted by a CA, and therefore the client will be able to trust it.

Once a certificate is signed by a CA, this certificate will always be valid (for the duration) if the client only checks the signature. This is not a good property, and we want to be able to revoke certificates within a PKI. That is why Control Revocation Lists were designed (CRLs) (R. Housley, 2002). A CRL is simply a list of all certificate serials and a reason why they were revoked. As you can guess, a CRL can create the following problems:

- CRLs will grow in size as time goes on, requiring a larger infrastructure over time.
- Clients will only periodically sync CRLs which may cause a client to allow a certificate which is already revoked.
- CRLs are prone to the availability problem, if an entity cannot request the CRL it cannot determine the status of a certificate.

Online Certificate Status Protocol (OCSP) is a protocol designed be a more efficient and accurate alternative to Control Revocation Lists (CRLs). The comparison of the OSCP protocol and the CRL protocol will be described further in the next chapter.

OCSP versus CRLs

OSCP tries to fulfill the same duty as CRLs but in a more direct manner. This means that the client can request on demand if a certificate is revoked. Figure 1 tries to illustrate this difference between OSCP and CRLs.

CRLs are downloaded with a (fixed) interval, once downloaded, the certificate in question will be locally evaluated if it's revoked. OSCP on the other hand works with a request-response mechanism requesting the most accurate information the CA can provide at that moment.

Obvious advantages are that the Entity is no longer required to download the whole list of revoked certificates and the information is more accurate at the time of evaluation compared to CRLs. This comes at the cost of being more prone to availability attacks (such as DDoS) because the service is completely used on-demand.

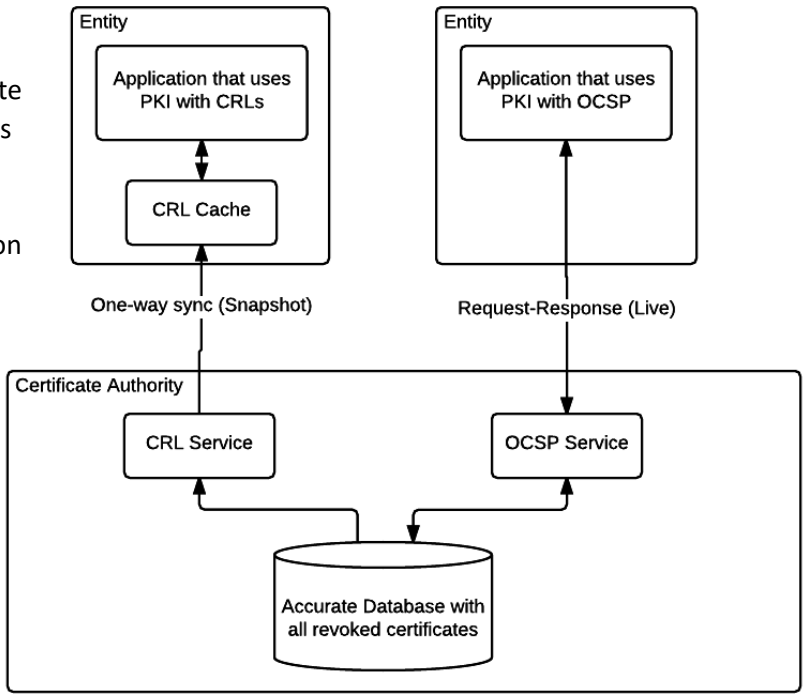

Figure 1- Basic difference between OSCP and CRLs

OCSP and its inner workings

RFC 2560 completely describes the complete implementation for the OCSP protocol. The basic idea is a request-response system in which certificate serial numbers can be queried. A request consists of a protocol version, service request, certificate serial number and optional extension information. A basic response message contains the protocol version, name of the responder, information for each of the sent certificate serial numbers (which consists of status code, validity interval and optional extension data), optional extension information, the signature algorithm (DSA or RSA) used and the signature of a hash of the whole response message (HMAC, SHA1). It is **required** to digitally sign the response message with one of the following certificates: The CA who issued the certificate, a (by the client) trusted party or a CA authorized responder. The status message of an certificate can be either "good" (accepted), "revoked" (not accepted) or "unknown" (this particular service does not know the status).

There is also the possibility that the OSCP service cannot respond to a request because of an error. Possible states are: malformedRequest, internalError, tryLater, signRequired (require the requester to sign the request, this requires a name to be sent with the request and optionally the certificate) and unauthorized. As these error states are self-explanatory, we will not discuss them here. A very important security issue arises from this error handling, because the RFC explicitly states that the error messages should not be signed. In this case a MitM attack (who has no control over availability of the network) can return any error message on a request done by his victim. This way the application that uses OCSP

cannot determine if the certificate is revoked. The only thing the application who gets an error on every OCSP request can do, is to decide to deny access (and interfere with availability) or allow (and interfere with confidentiality/integrity).

Another important part of the protocol is the "cache control". The validity interval within the response contains the following properties: thisUpdate (the time indicating that the CA knows this status is correct), nextUpdate (the time that newer information will be available, can be empty implying that this information is the most accurate to date) and producedAt (time of signing the response). In case of a CA private key compromise the OSCP may return "revoked" for all certificates falling under that CA.

By default the request-response mechanism do not include a nonce to prevent replay attacks. This is part of the extension mechanism. Another extension is the reference to a CRL if a certificate is revoked (to use as audit mechanism). Another example is a response which indicates with which Archive cutoff they work. The CA might decide to not keep all records from the past in their service (because of growing databases), they can indicate how accurate the information is. These extensions described here are just an example of what these extensions could be used for.

The protocol can be deployed over various transportation/application layers (such as HTTP, SMB, LDAP etc.). On the WWW, HTTP is seen as the standard because of the nature of the WWW.

Privacy implications of OCSP

Compared to CRLs, OCSP has some serious privacy implications. With CRLs the CA hosting the CRL service can identify that use the PKI infrastructure, but not at which time you use which certificate. On the other hand, using OCSP reveals information about which certificate you use within the PKI, but also the time of use. What makes it even worse is that once the client is required to sign its request the client cannot be only be identified by IP address, but also by the certificate itself. Beside the CA learning who you are, a possible MitM can also learn behavior and identity based on these requests. The signed requests are also not protected from replay attacks, causing possible evil actors to misuse these requests to mimic unwanted behavior acting like it's you.

Usage in current infrastructure

The biggest PKI used as of today is in the WWW (HTTPS). HTTPS is deployed on global scale, with increasing numbers every year. One of the most used applications to access the WWW is the web browser. A few of the most popular web browsers out there are Chrome, Firefox and Internet Explorer. Chrome by default relies only on CRLs, with OSCP built in but not enabled. Chrome updates their CRLs via their updater which means these lists lag behind even more than natural CRLs (in general) but they improved usage speed (ImperialViolet, 2012). Firefox decided to implement OCSP in Firefox 28 beside CRLs. The original idea was to deprecate the usage of CRLs in Firefox, but after the failing infrastructure (Goodwin, 2015) of OCSP they decided to use it as a soft-fail. This means that Firefox will still accept a certificate if it's not in the CRL list and doesn't get a (correct) response from the OCSP request. Internet Explorer and Opera seem to operate under the same CRLs and OCSP implementations with a soft-fail on OCSP (allowing an attacker to use a revoked certificate) (Mutton, 2014).

Another worrying development is the separation of usage between CRLs and OCSP. Most certificates that support OCSP are not present in the CRLs (Mutton, 2014), creating an even more vulnerable

situation for revoked certificates, since an attacker can easily intercept/manipulate communication between the client and OCSP service, allowing a certificate to be used meanwhile it could be revoked.

The hybrid solution: OCSP-Stapling

Addressing a lot of issues within OCSP, is OCSP Stapling (S. Blake-Wilson, 2006). This TLS-extension allows web services to send a signed response from the OCSP service with the information about the current certificate used. The idea is to let the web service request the status of the current certificate to an OCSP service, and "staple" this signed result to the handshake with the client. The client will be able to verify this OCSP response (because this will be signed by the CA), and use this response to verify that the certificate is not revoked. In this scenario the MitM scenario would not hold up, because the web service is now responsible for serving the correct response. Another important property is the re-use of the OCSP response by the server, possibly cutting down on infrastructure costs (also considering the time interval information prevents a replay attack). By default requiring these OCSP-staples does create availability problems once the OCSP service goes down and the response cached is no longer valid.

OCSP-Stapling also resolves a lot of the privacy implications originally created by OCSP. However, if the amount of OCSP requests based of the server is based on the amount of requests to the server, a CA could still determine if a website is being visited (although visitor numbers should be very low).

OCSP-stapling is implemented on all the popular browsers and web services (Apache, Nginx). Despite the solution that OCSP-Stapling provides there is still a possible attack surface. With a MitM attack, the attacker could still serve a TLS connection, with no OCSP-Stapling extension in the handshake. At this moment all browsers will just verify via CRLs or OCSP, and if the CRLs doesn't include the certificate and OCSP returns an error (due to infrastructure or the MitM attacker itself) the browser can still consider the certificate valid.

Firefox is trying to (partly) solve this problem by including a header which tells the browser to always requires a OCSP response in the handshake (Keeler, 2013). This is similar to the HTTP Strict Transport Protocol which tells the browser to always enforce HTTPS to a certain domain. Of course these mechanisms requires the browser to have at least visited the website once.

Conclusion and Opinion

Scalability is still a huge problem for managing revocations within an PKI. OCSP tries to improve the efficiency and time inaccuracy of RCLs, but also creates a lot of security issues (confidentiality, integrity and availability). When I was reading the OCSP RFC I kept thinking about these security issues, because these issues have serious implications on the validity of the PKI. I Also was genuinely surprised by the fact that the RFC stated that error messages should not be signed (since this allowed easy denial of service attacks by sending fake responses). Current implementations in web browsers could be considered as messy, since they implement three different kind of revocation checking which are not sound in security as a whole. OCSP-Stapling could be the silver bullet for this problem, but it needs to become required to become security sound. Requiring OCSP-Stapling can however impose serious availability problems once such OCSP systems goes down. I cannot determine if OCSP-Stapling can solve the scalability problem, but seems like a huge step forward compared to the implementation state its currently at.

Sources

Goodwin, M. (2015, Nov 23). *Improving Revocation: OCSP Must-Staple and Short-lived Certificates.* Retrieved from Mozilla Security Blog: https://blog.mozilla.org/security/2015/11/23/improving-revocation-ocsp-must-staple-and-short-lived-certificates/

ImperialViolet. (2012, Feb 05). *Revocation checking and Chrome's CRL.* Retrieved from ImperialViolet: https://www.imperialviolet.org/2012/02/05/crlsets.html

Keeler, D. (2013, July 29). *OCSP Stapling in Firefox.* Retrieved from Mozilla Security Blog: https://blog.mozilla.org/security/2013/07/29/ocsp-stapling-in-firefox/

M. Myers, R. A. (1999). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP.*

Mozilla. (2015, Nov 20). *CA:ImprovingRevocation.* Retrieved from Mozilla Wiki: https://wiki.mozilla.org/CA:ImprovingRevocation

Mutton, P. (2014, April 24). *Certificate revocation: Why browsers remain affected by Heartbleed.* Retrieved from Netcraft: http://news.netcraft.com/archives/2014/04/24/certificate-revocation-why-browsers-remain-affected-by-heartbleed.html

R. Housley, W. P. (2002). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.*

S. Blake-Wilson, M. N. (2006). *Transport Layer Security (TLS) Extensions.*