DEPLOYING THE ONLINE CERTIFICATE STATUS PROTOCOL WITHIN HIGH-VOLUME PKI ENVIRONMENTS

Marius Marian

Department of Automation, University of Craiova

Abstract: More and more human activities and businesses are now quite common to take place via Internet. The experience has proved that most network-related applications are affected by security attacks given a relatively moderate amount of time. Public-key cryptography permits to easily obtain integrity and confidentiality for the data exchanged between network applications, and also authentication of the involved parties. The verification of public-key certificates is one essential step when using public-key technology. The Online Certificate Status Protocol (OCSP) is meant to provide a timely and secure solution of getting certificate status information. The paper presents an OCSP implementation respecting most of the standardized lightweight OCSP profile, meant to accomodate the certificate revocation checking service for large scale public-key infrastructures.

Keywords: public-key infrastructures (PKI), online certificate status protocol (OCSP), certificate validation

1. INTRODUCTION

It's been a while since globalization – the name of the latest human cultural phenomenon – has started to affect each and every one of us, although quite often in such a transparent way. One of its characteristics is its capacity of getting more and more human activities on-line. With the arrival of Internet, new opportunities were created for interactive communication between parties who may have no pre-established relationship, and also for the development of new business practices such as electronic commerce (e-commerce), and for the innovation of public services for citizens, private companies, academics, and other public administration services.

But in this networked world, it is evident that communication without security is dangerous as long as undesired third parties can easily read, forge, or block messages. In order to achieve completely these new opportunities of Internet-based businesses, it is essential to guarantee the authenticity, integrity, confidentiality and non-repudiation of the information exchanged [Schneier], [Menezes et al.].

To deploy security within closed user groups, secret-key cryptography could be a solution, but when it comes to large, open groups of users (as it is the case of Internet) *public-key cryptography* represents since the end of the '90s the most suitable technique.

Public-key cryptography, also known as asymmetric cryptography, uses two matched keys in such a way that a message encrypted with one key can only be decrypted with the other. One of these keys will be kept private by the owner of the key pair, while the other will be made publicly-available. With each entity having a private key and a matching public key, the risks and overhead of symmetric cryptography involved in the process of sharing a copy of a secret key between multiple parties are eliminated. It is critical that the user be assured that the public key used is the correct one of its communicating party.

The protections afforded by public-key technology are totally compromised if intruders can substitute non-authentic or use compromised (i.e. non-verified) public keys.

Public-key *certificates* offered a solution to this problem [Kohnfelder]. Also known as digital certificates, they are data structures that securely bind a public-key value to an entity's identity. Currently, the most known and used standard for public-key certificates is X.509 [Housley et al.]. A public-key certificate is digitally signed by an entity, called *certification authority* (CA), that has confirmed the identity and maybe some other attributes of the holder of the corresponding private key. Certificates are a means of achieving scalability, with the final result of multiplying the size of the population using public-key technology.

2. PUBLIC-KEY INFRASTRUCTURES AND CERTIFICATE VALIDATION

The deployment of public-key cryptography in worldwide networks requires the so-called *public-key infrastructures* (PKI) that play the role of *trusted third party* (TTP), enabling thus diffusion of trust and confidence in networked environments. Essentially, a PKI comprises the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. PKI is becoming nowadays the central delivery point of security within companies, academic environments, and public administration.

Certificate validation is a complex process by which a certificate user establishes that the assertions made by a digital certificate can be trusted. During this process, a user will perform several verifications in order to establish whether the certificate is properly formed, signed and currently in force. If any of these verification steps fail the whole certificate validation process will fail.

Revocation checking represents a crucial part of any certificate-based security scheme. In fact, using a digital certificate without verifying its status can lead to serious consequences for both the relying party and the PKI. Consequently, the certificate user must investigate the status of the certificate, that is to verify whether it is still in its operational period or it was revoked, or otherwise announced as invalid.

Traditionally, revoked certificates were published on black lists called CRLs. However, the usage of certificate revocation lists (CRL) proves to have limitations from the scalability point of view and also in what concerns their use for on-line commercial transactions. Without completely replacing the CRL mechanism, the *on-line certificate status protocol* (OCSP) [Myers et al.] tries to provide a flexible scheme for real-time certificate status verification.

In practice, OCSP is suited for those cases where the freshness of revocation information represents a high-priority requirement of the relying party. For example, the need for real-time checking of certificate status is an essential requirement in Internet-based scenarios involving sensitive, high-value transactions. Additionally, even though a CRL is issued every hour, with an application validating a certificate by means of CRLs, it will still be nearly 60 minutes for a revoked certificate to appear as valid. And this is only one of the aspects that OCSP addresses very well.

3. OCSP IMPLEMENTATION

[Marian], [Marian et al.] detail the actual implementation of a real-time revocation-notification service based on OCSP. The performance measurements of this client-server implementation (tested on various platforms including here Win32, Solaris and Linux) and the comparative scrutiny with other revocation schemes confirm the architecture’s capability to fulfill the timeliness requirements needed for high-value Internet transactions.

Additionally, it is described and proposed an OCSP-client API, a very useful tool for PKI-aware applications able to perform certificate status checking in a transparent way on behalf of the user application. Given the fact that revocation checking is poorly implemented in many certificate-using products, we underline the advantages of integrating such an OCSP feature into PKI-based applications, with reduced technological costs.

4. SCALABILITY ISSUES

Different requirements are imposed today on an OCSP implementation [Deacon et al.]:

- to generate a fresh OCSP response for each OCSP request even in environments with bandwidth constraints and with limitations for what concerns the processing resources on both client and server side
- to remain cost-effective even when the environments where the related PKI is deployed, scale up

To exemplify, we will consider the mobile environments. In these environments, the network bandwidth costs and is limited. Mobile devices depend strongly on battery lifetime and are additionally constrained by their computing power. Therefore, the OCSP responders must be carefully implemented such that to reduce at maximum the pressure on the network bandwidth and also on the clients' capability to process OCSP responses.

On the other hand, sensitive transactions (for example, financial applications) have already become available in these mobile, extremely populated environments and PKIs have shortly followed, being deployed to support their security needs. It becomes stringent that OCSP must be used in such a way to minimize the load on the corresponding OCSP responder (since millions of clients may issue requests for certificate status checking) and also on the network infrastructure used.

To address all these scalability issues, RFC 5019 [Deacon et al.] has recommended a message profile and also an optimal behavior for OCSP clients and responders permitting thus:

- OCSP response pre-production and distribution
- Reduced OCSP message size to lower bandwidth usage
- Response message caching both in the network and on the client.

OCSP clients relying on this profile will not be able to differentiate between lightweight-profiled and fully-fledged OCSP responders, unless by means of an out-of-band notification mechanism. Nevertheless, the modifications required by this lightweight profile will not compromise the interoperability between a fully-fledged OCSP client and a lightweight-profiled OCSP responder.

Our client-server OCSP implementation was changed so that to allow configuration and operation of both modes: OCSP fully-compatible and OCSP lightweight-profile.

In brief, the modifications purported by this lightweight profile apply to:

- OCSP message structure (we include here both OCSP request and response)
- OCSP client behavior (how the OCSP responder is discovered and also, the way in which it sends a request)
- the mechanism to ensure fresh OCSP responses
- the transport profile
- the caching of the OCSP responses
- the security of the general OCSP scenario (presenting the measures to mitigate replay, man-in-the-middle and denial-of-service attacks)

5. LIGHTWEIGHT OCSP PROFILE MESSAGE STRUCTURE

For this profile, our OCSP client was modified such that to be able to include only one request in the ***OCSPRequest.RequestList*** structure profile [Myers et al.]. In the same way, the client will use the SHA1 algorithm to compute the digest values corresponding to the ***CertID.issuerNameHash*** and the ***CertID.issuerKeyHash*** values. The client can also be configured not to include the ***singleRequestExtensions*** structure and also to avoid inclusion of the ***requestExtensions*** structure. However, if a ***requestExtensions*** structure is necessary and gets included in the ***OCSPRequest***, it will only contain the nonce extension (*id-pkix-ocsp-nonce*).

When working in this profile the OCSP client will not send signed ***OCSPRequests***. If the ***OCSPRequest*** is signed, the client abiding to this profile specifies its name in the ***OCSPRequest.requestorName*** field. On the server side, the standard states that the lightweight OCSP responder can choose to ignore the signature on ***OCSPRequests***, however in practice our implementation will always check the signature on the ***OCSPRequest***, if present.

Responders generate a ***BasicOCSPResponse*** as identified by the *id-pkix-ocsp-basic* OID (object identifier). Clients must be able to parse and accept a ***BasicOCSPResponse***. ***OCSPResponses*** conformant to this profile include only one ***SingleResponse*** in the ***ResponseData.responses*** structure. Additionally, the responder should avoid including the ***responseExtensions*** in order to minimize the charge on the client. Some of the unrecognized non-critical ***responseExtensions*** in the response will be however ignored by clients.

Clients as usual will always validate the signature on the returned ***OCSPResponse***. When the response is signed by a delegate of the issuing certification authority (CA), a valid responder's certificate is referenced in the ***BasicOCSPResponse.certs*** structure. To avoid further load on the client's computing power the *id-pkix-ocsp-nocheck* extension is always included in the response, to announce clients not to check the responder certificate's status.

One noteworthy feature is that neither an OCSP *authorityInfoAccess* (AIA) extension nor the *cRLDistributionPoints* (CRLDP) extension are included in the OCSP responder's certificate. Consequently, the responder's signing certificate will be relatively short-lived (in order to avoid its compromise) and renewed regularly. To facilitate the identification of the appropriate OCSP responder certificate among all these certificates, client will be using both the *byName* and *byKey* ***ResponseData.ResponderID*** choices. To further reduce the size of the response in scenarios where limitations of the bandwidth are present, the responders should use the *byKey*.

Our implementation assumes that the OCSP database and information sources are authoritative for the PKI's certificates it services therefore the ***OCSPResponseStatus*** will be *successful*. When access to authoritative records for a particular certificate is not available, the responder returns an *unauthorized* ***OCSPResponseStatus***. This is useful when the responder provides pre-produced OCSP responses. One trick to minimize the load on the responder is to remove from the local database of revocation information all records that concern expired certificates. In this way, clients requesting revocation status for expired certificates (and thus, removed from the database) will be returned responses with an *unauthorized* ***OCSPResponseStatus***.

For the purposes of this profile, ASN.1-encoded ***GeneralizedTime*** values such as ***thisUpdate***, ***nextUpdate***, and ***producedAt*** must be expressed in Greenwich Mean Time and must include seconds, even when the number of seconds is zero [Deacon et al.].

Our OCSP client implementation already supports the *authorityInfoAccess* extension as defined in [Housley et al.] and successfully recognizes the *id-ad-ocsp* access method. This extension is used by CAs to inform clients how they can contact the OCSP service. To respect the profile recommendation, in the case where a client is checking the status of a certificate that contains both an *authorityInformationAccess* extension and a *cRLDistributionPoints* extension pointing to a CRL, our client will first attempt to contact the OCSP responder. Clients will try to retrieve the CRL if and only if no OCSP response is received from the responder after a locally configured timeout and number of retries.

The profile [Deacon et al.] states that in order to save resources concerning the network traffic, OCSP applications must first verify the signature of signed data before asking an OCSP client to check the status of certificates used to verify the data. If the signature is invalid or the application is not able to verify it, an OCSP check becomes futile and consequently never requested.

6. TIMELINESS OF OCSP RESPONSES

OCSP clients within this profile must take appropriate measures to ensure that they receive the freshest OCSP response available. At least two mechanisms are available for this.

- either by means of nonces (i.e. no more than once), or
- by checking the time of the OCSP response. For this to happen, both clients and responders must have access to an accurate source of time.

The profile specifies that clients should not include a ***requestExtensions*** structure in OCSP requests, so clients must be able to determine ***OCSPResponse*** freshness based on an accurate source of time. Clients that opt to include a nonce in the request should not reject a corresponding OCSP response solely on the basis of the nonexistent expected nonce, but must fall back to validating the ***OCSPResponse*** based on time. On the other hand, if the client does not include a nonce in the request it must ignore any nonce that may be present in the response.

The verification of freshness imposes that clients check first for the existence of the ***nextUpdate*** field, then obtain an accurate reading of current time, and then compare the current time reading with the values of the ***thisUpdate*** and ***nextUpdate*** fields contained in the response. The absence of the ***nextUpdate*** field is an indication for the client that the response should not be trusted. The relation to check is:

thisUpdate < current time < ***nextUpdate***

Our client implementation does not allow configuration of a small tolerance period for acceptance of responses after ***nextUpdate*** to handle minor clock differences relative to responders and caches.

7. TRANSPORT PROTOCOLS

Both implementations of the OCSP responder and the client support requests and responses over HTTP. When sending requests that are less than or equal to 255 bytes in total (after encoding), clients use the GET method (to enable OCSP response caching). OCSP requests larger than 255 bytes are submitted using the POST method.

7.1 Caching

The ability to cache OCSP responses throughout the network is an important factor in high volume OCSP deployments. Including OCSP responses in protocol exchanges, such as has been defined in TLS [Dierks et al.], is also important for the profile. To minimize bandwidth usage, clients locally cache authoritative OCSP responses (i.e., a response with a signature that has been successfully validated and that indicate an ***OCSPResponseStatus*** of *successful* [Deacon et al.]. Most OCSP clients will send ***OCSPRequests*** only when a cached response expires.

In some scenarios, it is advantageous to include OCSP response information within the protocol being utilized between the client and server. The profile [Deacon et al.] enumerates the interesting effects when such a behavior is adopted:
it allows for the caching of OCSP responses on the server, thus lowering the number of hits to the OCSP responder:

- it enables certificate validation in the event the client is not connected to a network and thus eliminating the need for clients to establish a new HTTP session with the responder.
- it reduces the number of round trips the client needs to make in order to complete a handshake.
- it simplifies the client-side OCSP implementation by enabling a situation where the client need only the ability to parse and recognize OCSP responses.

Currently, this functionality has been specified as an extension to the TLS protocol [Dierks et al.]. The lightweight OCSP profile recommends that both TLS clients and servers implement the certificate status request extension mechanism for TLS.

Our OCSP client implementation is currently missing the possibility of caching and using OCSP responses on-site.

8. PROFILE VULNERABILITIES

The profile is prone to several types of attacks:

- replay attacks.
- man-in-the-middle attacks
- masquerading attacks
- denial-of-service attacks

The replay of OCSP responses can be done when the use of nonces is dismissed. OCSP clients can thus be fed fake *good* responses when the actual status of a certificate is *revoked.* The OCSP clients should strongly rely on accurate sources of time in order to avoid the receipt of an aged response.

A well implemented OCSP client will never fall for a man-in-the-middle attack simply by properly checking the identity of the entity with which they are communicating in order to ensure that it actually is the OCSP responder they suppose it is. The use of signed responses in OCSP serves rightfully to authenticate the identity of the OCSP responder and to verify that it is authorized to sign responses on the CA's behalf.

The above statement also applies for impersonation attacks. The use of signed responses in OCSP serves to authenticate the identity of OCSP responder. OCSP clients must properly validate the signature of the OCSP response and the signature on the OCSP response signer certificate to ensure that an authorized responder created it.

The most difficult attack to combat is denial-of-service. As this profile specifies the use of unsigned OCSP requests, access to the responder may be implicitly given to everyone who can send a request to a responder, and thus the ability to mount a denial-of-service attack via a flood of requests may be greater. To mitigate this other measures can be conceived, the profile suggests that a responder could limit the rate of incoming requests from a particular IP address if questionable behavior is detected. But this assumes that an additional security infrastructure to be built around the PKI services.

9. CONCLUSIONS

The original client-server implementation of the OCSP standard presented in this paper was deployed since 2001 as a service within the EuroPKI [EuroPKI] public-key infrastructure.
The lightweight OCSP profile [Deacon et al.] standard appeared September 2007. Therefore, modifications were brought to the original implementation such that to adopt this standard profile and to allow a more scalable approach for high-volume environments using security based on public-key technology. Further improvements of this implementation will concern the caching mechanisms on the client side, and also a better management of the OCSP pre-produced responses.

REFERENCES

Deacon, A., Hurst, R., (2007) *The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments*, RFC 5019

Dierks, T., and Rescorla, E. (2006) *The Transport Layer Security Protocol Version 1.1*, RFC 4346

EuroPKI public-key infrastructure, available online at http://www.europki.org

Fielding, R., Gettys, J., Mogul, J. Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, (1999) *Hypertext Transfer Protocol -- HTTP/1.1*, RFC 2616

Housley, R., Polk, W., Ford, W., Solo, D. (2002) *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF, RFC 3280

Kohnfelder, L.M., (1978) *Toward a Practical Public-Key Cryptosystem*, B.Sc. thesis, MIT Department of Electrical Engineering

Marian, M., (2001) *An Implementation of the Online Certificate Status Protocol*, Annals of the University of Craiova, pp157-166, ISSN 1223-530x

Marian, M., Berbecaru, D., Lioy, A. (2002) *Security aspects in standard certificate revocation mechanisms: a case study for OCSP*, Proceedings of 7th IEEE Symposium on Computers and Communications, Taormina/Giardini Naxos, Italy, July 1 - 4, 2002, pp: 484 – 489

Menezes, A.J., Oorschot, P.C., Vanstone, S. (1997) *Handbook of Applied Cryptography*, CRC Press, SBN: 0-8493-8523-7

Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C., (1999) *X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol - OCSP*, RFC 2560

Schneier, B., (1996) *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc.