JOSE   R. Barnes
Internet-Draft   BBN Technologies
Intended status: Informational   April 6, 2012
Expires: October 8, 2012

Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)

draft-barnes-jose-use-cases-00.txt

Abstract

Many Internet applications have a need for object-based security mechanisms in addition to security mechanisms at the network layer or transport layer. In the past, the Cryptographic Message Syntax has provided a binary secure object format based on ASN.1. Over time, the use of binary object encodings such as ASN.1 has been overtaken by text-based encodings, for example JavaScript Object Notation. This document defines a set of use cases and requirements for a secure object format encoded using JavaScript Object Notation, drawn from a variety of application security mechanisms currently in development.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Internet applications rest on the layered architecture of the Internet, and take advantage of security mechanisms at all layers. Many applications rely primarily on channel-based security technologies, which create a secure channel at the IP layer or transport layer over which application data can flow [RFC4301][RFC5246]. These mechanisms, however, cannot provide end-to-end security in some cases. For example, in protocols with application-layer intermediaries, channel-based security protocols would protect messages from attackers between intermediaries, but not from the intermediaries themselves. These cases require object-based security technologies, which embed application data within a secure object that can be safely handled by untrusted entities.

The most well-known example of such a protocol today is the use of Secure/Multipurpose Internet Mail Extensions (S/MIME) protections within the email system [RFC5751][RFC5322]. An email message typically passes through a series of intermediate Mail Transfer Agents (MTAs) en route to its destination. While these MTAs often apply channel-based security protections to their interactions (e.g., [RFC3207]), these protections do not prevent the MTAs from interfering with the message. In order to provide end-to-end security protections in the presence of untrusted MTAs, mail users can use S/MIME to embed message bodies in a secure object format that can provide confidentiality, integrity, and data origin authentication.

S/MIME is based on the Cryptographic Message Syntax for secure objects (CMS) [RFC5652]. CMS is defined using Abstract Syntax Notation 1 (ASN.1) and traditionally encoded using the ASN.1 Distinguished Encoding Rules (DER), which define a binary encoding of the protected message and associated parameters [X.680][X.690]. In recent years, usage of ASN.1 has decreased (along with other binary encodings for general objects), while more applications have come to rely on text-based formats such as the Extensible Markup Language (XML) or the JavaScript Object Notation (JSON) [XML][RFC4627].

Many current applications thus have much more robust support for processing objects in these text-based formats than ASN.1 objects; indeed, many lack the ability to process ASN.1 objects at all. To simplify the addition of object-based security features to these applications, the IETF JSON Object Signing and Encryption (JOSE) working group has been chartered to develop a secure object format based on JSON. While the basic requirements for this object format are straightforward -- namely, confidentiality and integrity mechanisms, encoded in JSON -- early discussions in the working group indicated that many applications hoping to use the formats define in

JOSE have additional requirements. This document summarizes the use cases for JOSE envisioned by those applications and the resulting requirements for security mechanisms and object encoding.

2. Definitions

This document makes extensive use of standard security terminology [RFC4949]. In addition, because the use cases for JOSE and CMS are similar, we will sometimes make analogies to some CMS concepts [RFC5652].

The JOSE working group charter calls for the group to define three basic JSON object formats:

1. Confidentiality-protected object format

2. Integrity-protected object format

3. A format for expressing public keys

In the below, we will refer to these as the "encrypted object format", the "signed object format", and the "key format", respectively. In general, where there is no need to distinguish between asymmetric and symmetric operations, we will use the terms "signing", "signature", etc. to denote both true digital signatures involving asymmtric cryptography as well as message authentication codes using symmetric keys(MACs).

In the lifespan of a secure object, there are two basic roles, an entity that creates the object (e.g., encrypting or signing a payload), and an entity that uses the object (decrypting, verifying). We will refer to these roles as "sender" and "recipient", respectively. Note that while some requirements and use cases may refer to these as single entities, each object may have multiple entities in each role. For example, a message may be signed by multiple senders, or decrypted by multiple recipients.

3. Basic Requirements

Obviously, for the encrypted and signed object formats, the necessary protections will be created using appropriate cryptographic mechanisms: symmetric or asymmetric encryption for confidentiality and MACs or digital signatures for integrity protection. In both cases, it is necessary for the JOSE format to support both symmetric and asymmetric operations.

- The JOSE encrypted object format MUST support object encryption in the case where the sender and receiver share a symmetric key.

- The JOSE encrypted object format MUST support object encryption in the case where the sender has only a public key for the receiver.

- The JOSE signed object format MUST integrity protection using Message Authentication Codes (MACs), for the case where the sender and receiver share a symmetric key.

- The JOSE signed object format MUST integrity protection using digital signatures, for the case where the receiver has only a public key for the sender.

The purpose of the key format is to provide the recipient with sufficient information to use the encoded key to process cryptographic messages. Thus it is necessary to include additional parameters along with the bare key.

- The JOSE key format MUST include all algorithm parameters necesssary to use the encoded key, including an identifier for the algorithm with which the key is used as well as any additional parameters required by the algorithm (e.g., elliptic curve parameters).

4. Use Cases

Based on early discussions of JOSE, several working groups developing application-layer protocols have expressed a desire to use JOSE in their designs for end-to-end security features. In this section, we summarize the use cases proposed by these groups and discuss the requirements that they imply for the JOSE object formats.

4.1. OAuth

The OAuth protocol defines a mechanism for distributing and using authorization tokens using HTTP [I-D.ietf-oauth-v2]. A Client that wishes to access a protected resource requests authorization from the Resource Owner. If the Resource Owner allows this access, he directs an Authorization Server to issue an access token to the Client. When the Client wishes to access the protected resource, he presents the token to the relevant Resource Server, which verifies the validity of the token before providing access to the protected resource.

```
                 +---------------+          +---------------+
                 |               |          |               |
                 |   Resource    |<........>| Authorization |
                 |    Server     |          |     Server    |
                 |               |          |               |
                 +---------------+          +---------------+
                              ^                |
                              |                |
                              |                |
                              |                |
                              |                |
                 +------------|--+          +--|------------+
                 |            +----------------+            |
                 |               |          |   Resource    |
                 |     Client    |          |     Owner     |
                 |               |          |               |
                 +---------------+          +---------------+
```

Figure 1: The OAuth process

In effect, this process moves the token from the Authorization Server (as a sender of the object) to the Resource Server (recipient), via the Client as well as the Resource Owner (the latter because of the HTTP mechanics underlying the protocol). So again we have a case where an application object is transported via untrusted intermediaries.

This application has two essential security requirements: Integrity and data origin authentication. Integrity protection is required so that the Resource Owner and the Client cannot modify the permission encoded in the token. Data origin authentication is required so that the Resource Server can verify that the token was issued by a trusted Authorization Server. Confidentiality protection may also be needed, if the Authorization Server is concerned about the visibility of permissions information to the Resource Owner or Client. For example, permissions related to social networking might be considered private information. Note, however, that OAuth already requires that the underlying HTTP transactions be protected by TLS, so confidentiality protection is not strictly necessary for this use case.

The confidentiality and integrity needs are met by the basic requirements for signed and encrypted object formats, whether the signing and encryption are provided using asymmetric or symmetric cryptography. The choice of which mechanism is applied will depend on the relationship between the two servers, namely whether they share a symmetric key or only public keys.

Authentication requirements will also depend on deployment characteristics. Where there is a relatively strong binding between the resource server and the authorization server, it may suffice for the Authorization Server issuing a token to be identified by the key used to sign the token. This requires that the token carry either the public key of the Authorization Server or an identifier for the public or symmetric key.

There may also be more advanced cases, where the Authorization Server's key is not known in advance to the Resource Server. This may happen, for instance, if an entity instantiated a collection of Authorization Servers (say for load balancing), each of which has an independent key pair. In these cases, it may be necessary to also include a certificate or certificate chain for the Authorization Server, so that the Resource Server can verify that the Authorization Server is an entity that it trusts.

The HTTP transport for OAuth imposes a particular constraint on the encoding. In the OAuth protocol, tokens frequently need to be passed as query parameters in HTTP URIs [RFC2616], after having been base64url encoded [RFC4648]. While there is no prescribed limit on the length of URIs (and thus of query parameters), in practice URIs of more than around 2,000 characters are rejected by some user agents. So this use case requires that a JOSE object have sufficiently small size even after signing, possibly encrypting, and base64url encoding.

4.2. XMPP

The Extensible Messaging and Presence Protocol (XMPP) routes messages from one end client to another by way of XMPP servers [RFC6120]. There are typically two servers involved in delivering any given message: The first client (Alice) sends a message for another client (B) to her server (A). Server A uses Bob's identity and the DNS to locate the server for Bob's domain (B), then delivers the message to that server. Server B then routes the message to Bob.

```
         +-------+   +----------+   +----------+   +-----+
         | Alice |-->| Server A |-->| Server B |-->| Bob |
         +-------+   +----------+   +----------+   +-----+
```

Figure 2: Delivering an XMPP message

The untrusted-intermediary problems are especially acute for XMPP because in many current deployments, the holder of an XMPP domain outsources the operation of the domain's servers to a different entity. In this environment, there is a clear risk of exposing the domain holder's private information to the domain operator. XMPP

already has a defined mechanism for end-to-end security using S/MIME, but it has failed to gain widespread deployment [RFC3923], in part because of key management challenges and because of the difficulty of processing S/MIME objects.

The XMPP working group is in the process of developing a new end-to-end encryption system with an encoding based on JOSE and a clearer key management system [I-D.miller-xmpp-e2e]. The process of sending an encrypted message in this system involves two steps: First, the sender generates a symmetric Content Encryption Key (CEK), encrypts the message content, and sends the encrypted message to the desired set of recipients. Second, each recipient "dials back" to the sender, providing his public key; the sender then responds with the relevent CEK, wrapped with the recipient's public key.

```
         +-------+   +----------+   +----------+   +-----+
         | Alice |<->| Server A |<->| Server B |<->| Bob |
         +-------+   +----------+   +----------+   +-----+
             |             |              |           |
             |------------Encrypted--message--------->|
             |             |              |           |
             |<---------------Public-key--------------|
             |             |              |           |
             |---------------Wrapped CEK------------->|
             |             |              |           |
```

Figure 3: Delivering a secure XMPP message

The main thing that this system requires from the JOSE formats is confidentiality protection via content encryption, plus an integrity check via a MAC derived from the same symmetric key. The separation of the key exchange from the transmission of the encrypted content, however, requires that the JOSE encrypted object format allow wrapped symmetric keys to be carried separately from the encrypted payload. In addition, the encrypted object will need to have a tag for the key that was used to encrypt the content, so that the recipient (Bob) can present the tag to the sender (Alice) when requesting the wrapped key.

Another important feature of XMPP is that it allows for the simultaneous delivery of a message to multiple recipients. In the diagrams above, Server A could deliver the message not only to Server B (for Bob) but also to Servers C, D, E, etc. for other users. In such cases, to avoid the multiple "dial back" transactions implied by the above mechanism, XMPP systems will likely cache public keys for end recipients, so that wrapped keys can be sent along with content on future messages. This implies that the JOSE encrypted object format must support the provision of multiple versions of the same

wrapped CEK (much as a CMS EnvelopedData structure can include multiple RecipientInfo structures).

In the current draft of the XMPP end-to-end security system, each party is authenticated by virtue of the other party's trust in the XMPP message routing system. The sender is authenticated to the receiver because he can receive messages for the identifier "Alice" (in particular, the request for wrapped keys), and can originate messages for that identifier (the wrapped key). Likewise, the receiver is authenticated to the sender because he received the original encrypted message and originated the request for wrapped key. So the authentication here requires not only that XMPP routing be done properly, but also that TLS be used on every hop. Moreover, it requires that the TLS channels have strong authentication, since a man in the middle on any of the three hops can masquerade as Bob and obtain the key material for an encrypted message.

Because this authentication is quite weak (depending on the use of transport-layer security on three hops) and unverifiable by the endpoints, it is possible that the XMPP working group will integrate some sort of credentials for end recipients, in which case there would need to be a way to associate these credentials with JOSE objects.

Finally, it's worth noting that XMPP is based on XML, not JSON. So by using JOSE, XMPP will be carrying JSON objects within XML. It is thus a desirable property for JOSE objects to be encoded in such a way as to be safe for inclusion in XML. Otherwise, an explicit CDATA indication must be given to the parser to indicate that it is not to be parsed as XML. One way to meet this requirement would be to apply base64url encoding, but for XMPP messages of medium-to-large size, this could impose a fair degree of overhead.

4.3. ALTO

Application-Layer Traffic Optimization (ALTO) is a system for distributing network topology information to end devices, so that those devices can modify their behavior to have a lower impact on the network [I-D.ietf-alto-reqs]. The ALTO protocol distributes topology information in the form of JSON objects carried in HTTP [RFC2616][I-D.ietf-alto-protocol]. The basic version of ALTO is simply a client-server protocol, so simple use of HTTPS suffices for this case [RFC2818]. However, there is beginning to be some discussion of use cases for ALTO in which these JSON objects will be distributed through a collection of intermediate servers before reaching the client, while still preserving the ability of the client to authenticate the original source of the object. Even the base ALTO protocol notes that "ALTO clients obtaining ALTO information

must be able to validate the received ALTO information to ensure that it was generated by an appropriate ALTO server."

In this case, the security requirements are straightforward. JOSE objects carrying ALTO payloads will need to bear digital signatures from the originating servers, which will be bound to certificates attesting to the identities of the servers. There is no requirement for confidentiality in this case, since ALTO information is generally public.

The more interesting questions are encoding questions. ALTO objects are likely to be much larger than payloads in the two cases above, with sizes of up to several megabytes. Processing of such large objects can be done more quickly if it can be done in a single pass, which may be possible if JOSE objects require specific orderings of fields within the JSON structure.

In addition, because ALTO objects are also encoded as JSON, they are already safe for inclusion in a JOSE object. Signed JOSE objects will likely carry the signed data in a string alongside the signature. JSON objects have the property that they can be safely encoded in JSON strings. All they require is that unnecessary white space be removed, a much simpler transformation than, say base64url encoding. This raises the question of whether it might be possible to optimize the JOSE encoding for certain "JSON-safe" cases.

5. Other Requirements

[[For the initial version of this document, this section is a placeholder, to incorporate any further requirements not directly derived from the above use cases.]]

6. Acknowledgements

Thanks to Matt Miller for discussions related to XMPP end-to-end security model.

7. IANA Considerations

This document makes no request of IANA.

8. Security Considerations

The primary focus of this document is the requirements for a JSON-

based secure object format. At the level of general security considerations for object-based security technologies, the security considerations for this format are the same as for CMS [RFC5652]. The primary difference between the JOSE format and CMS is that JOSE is based on JSON, which does not have a canonical representation. The lack of a canonical form means that it is difficult to determine whether two JSON objects represent the same information, which could lead to vulnerabilities in some usages of JOSE.

9. References

9.1. Normative References

[I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-11 (work in progress), March 2012.

[I-D.ietf-alto-reqs]
Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", draft-ietf-alto-reqs-13 (work in progress), January 2012.

[I-D.ietf-oauth-v2]
Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol", draft-ietf-oauth-v2-25 (work in progress), March 2012.

[I-D.miller-xmpp-e2e]
Miller, M., "End-to-End Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)", draft-miller-xmpp-e2e-00 (work in progress), February 2012.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

9.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

[RFC3923] Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)", RFC 3923, October 2004.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[X.680] "Recommendation X.680, "Information Technology -- Abstract Syntax Notation One (ASN.1) -- Specification of Basic Notation", November 1994.

[X.690] "Recommendation X.690, "Information Technology -- ASN.1 Encoding Rules -- Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", November 1994.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium Recommendation REC-xml, October 2000.

Author's Address

Richard Barnes
BBN Technologies
9861 Broken Land Parkway
Columbia, MD 21046
US

Phone: +1 410 290 6169
Email: rbarnes@bbn.com