TOC

OAUTH	H. Tschofenig
Internet-Draft	Nokia Siemens Networks
Intended status: Informational	B. Cook
Expires: April 21, 2011	BT
	October 18, 2010

Thoughts about Digital Signatures for the Open Web Authentication (OAuth) Protocol
draft-tschofenig-oauth-signature-thoughts-00.txt

Abstract

The initial version of the Open Web Authentication Protocol (OAuth 1.0), often referred to as the community addition, included an mechanism for putting a digital signature (when using asymmetric keys) or a keyed message digest (when using symmetric keys) to a resource request when presenting the OAuth token. This cryptographic mechanism has lead to lots of discussions, particularly about the problems implementers had, the use cases it supports, and the benefit-cost tradeoff.

This document tries to describe the use of the so-called 'OAuth Signature' mechamism in an unbiased and less emotional way with the main purpose to conclude the discussions.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

TOC

The initial version of the Open Web Authentication Protocol (OAuth 1.0) included an mechanism for putting a digital signature (when using asymmetric keys) or a keyed message digest (when using symmetric keys) to a resource request when presenting the OAuth token. OAuth 2.0 generalized the structure a bit and the abstract and simplified description of the protocol has the following structure:

```
                    ,-.
                   (* *)
                    `+'  User
                     '
  :------------   ---|---  ~~~~~~~~~~:
  : Service         / \             : Management
  : Interaction    /   \             : of Resources
  :                  : Consent       :
  :                  :               :
  :                  :               :
  :              +---:-+ Carol       :
  :    1.        |Carol| as          :
  :  Obtain   .'>|     | Asserting   :
  :  Access .'   +-----+ Party       :
  :  Token.'                         :
  :     .'                           :
  :   .'                             :
  : v'                               :
+-:---+                           +--:--+ Bob
|Alice|<------------------------->|Bob  | as
|     |      2. Authenticated     |     | Relying
+-----+         Request +         +-----+ Party
                Access Token
```

Figure 1: OAuth Simplified

We use symbolic names in Figure 1 (OAuth Simplified). A few remarks about the figure. In addition to illustrating the message exchange between Alice, Bob, and Carl it also highlights the importance of the user in the exchange in providing consent, triggering the entire interaction as part of invoking a service, and in managing a resource that is work delegating access to.

From a cryptographic point of view the following aspects of the OAuth 1.0 specification are worth mentioning:

- The format and content of the Access Token is not specified.

- The authenticated request shown in (2) is essentially a basic HTTP authentication mechanism that supports symmetric as well as asymmetric credentials. The purpose is to authenticate Alice to Bob; no mutual authentication. The procedure for obtaining these credentials is outside the scope. To ensure liveness of the authentication a timestamp and a nonce is included in the request (and is included in the digital signature and the keyed message digest).

- The authenticated request signing is optional to implement and optional to use. When the authenticated request signature is omitted (called bearer token) then TLS. Details about what ciphersuite to use with TLS and what required features are needed are not available.

OAuth 2.0 [I-D.ietf-oauth-v2] (Hammer-Lahav, E., Recordon, D., and D. Hardt, “The OAuth 2.0 Protocol,” July 2010.) currently does not provide text for authenticated requests in the specification nor does it mandate the use of TLS.

2. Terminology

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).
This document uses the terminology defined in RFC 4949 [RFC4949] (Shirey, R., “Internet Security Glossary, Version 2,” August 2007.).
The terms 'keyed hash' and 'keyed message digest' are used interchangable.
All discussions in this document refer to the abstract names, namely Alice, Bob, and Carol, shown in Figure 1 (OAuth Simplified).

3. Security Threats

TOC

The following list presents several common threats against protocols utilizing some form of tokens. This list of threats is based on NIST Special Publication 800-63 [NIST800-63] (Burr, W., Dodson, D., Perlner, R., Polk, T., Gupta, S., and E. Nabbus, “NIST Special Publication 800-63-1, INFORMATION SECURITY,” December 2008.). We exclude a discussion of threats related to any form of registration and authentication.

> **Token manufacture/modification:** An attacker may generate a bogus token or modify the token content (such as the authentication or attribute statements) of an existing token, causing Bob to grant inappropriate access to the Alice. For example, an attacker may modify the token to extend the validity period; Alice may modify

> the assertion to have access to information that they should not be able to view.
>
> **Token disclosure:** Tokens may contain authentication and attribute statements that include sensitive information.
>
> **Token redirect:** An attacker uses the token generated for consumption by Bob to obtain access to a second Relying Party.
>
> **Token reuse:** An attacker attempts to use a token that has already been used once with Bob.

A Web authentication protocol must provide a consistent story on how to deal with all these threats. We excluded one threat from the list, namely 'token repudiation'. Token repudiation refers to a property whereby Bob is given an assurance that Carol cannot deny to have created a token for Alice. We believe that such a property is interesting from theoretical point of view but most deployments perfer to deal with the violation of this security property via business actions rather than using cryptography.

---

4. Threat Mitigation

TOC

A large range of threats can be mitigated by protecting the content of the token, using a digitial signature or a keyed message digest. Alternatively, the content of the token could be passed by reference rather than by value (requiring a separate message exchange to resolve the reference to the token content). To simplify the subsequent description we assume that the token itself is digitally signed by Carol and therefore cannot be modified. This also provides the basis for non-repudiation.
To deal with token redirect it is important for Carol to include the identity of the intended recipient, namely Bob. Carol would have to be told that Bob is the intended recipient.
To provide protection against token disclosure two approaches are possible, namely (a) not to include sensitive information inside the token or (b) to ensure confidentiality protection. The latter feature requires at least the communication interaction between the Alice and Carol as well as the interaction between Alice and Bob to experience confidentiality protection. As an example, Transport Layer Security with a ciphersuite that offers confidentiality protection has to be applied. Encrypting the token content is another alternative.
To deal with the last threat, namely token reuse, more choices are available. First, it is highly advisable for Carol to restrict the lifetime of the token by putting a validity time field inside the protected part of the token. This is, however, largely independent of the potential approaches described in the sub-sections below.

4.1. Confidentiality Protection

TOC

In this approach confidentiality protection of the exchange is provided on the communication interfaces between Alice and Bob, and Alice and Carol. No eavesdropper on the wire is able to observe the token exchange. Consequently, a replay is not possible. Carol wants to ensure that it only hands out tokens to entities it has authenticated first and who are authorized. For this purpose, authentication of Alice to Carol will be a requirement. This is, however, true for the description in Section 4.2 (Sender Constraint) and Section 4.3 (Key Confirmation) as well. Furthermore, Alice has to make sure it does not distribute the token to entities other than Bob. For that purpose Alice will have to authenticate Bob before transmitting the token.

4.2. Sender Constraint

TOC

Instead of providing confidentiality protection Carl could also put the identity of Alice into the protected token with the following semantic: 'This token is only valid when presented by Alice.' When the token is then presented to Bob how does he know that it was provided by Alice? He has to authenticate Alice! There are many choices for authenticating Alice to Bob, such as, client-side certificates in TLS [RFC5246] (Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” August 2008.), or pre-shared secrets within TLS [RFC4279] (Eronen, P. and H. Tschofenig, “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS),” December 2005.). The choice of the preferred authentication mechanism and credential type may depend on a number of factors, including

*security properties

*available infrastructure

*library support

*credential cost (financial)

*performance

*integration into the existing IT infrastructure

*operational overhead for configuration and distribution of credentials

This long list hints to the challenge of selecting at least one mandatory-to-implement mechanism.

4.3. Key Confirmation

TOC

A variation of the mechanism of sender authentication described in Section 4.2 (Sender Constraint) is to replace authentication with the proof-of-possession of a specific key, i.e. key confirmation. In this model Bob would not authenticate Alice but would rather verify whether Alice knows a secret. This mechanism corresponds to the message signature approach defined by the OAuth 1.0 signature mechanisms [RFC5849] (Hammer-Lahav, E., “The OAuth 1.0 Protocol,” April 2010.) or by Kerberos [RFC4120] (Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” July 2005.) when utilizing the AP_REQ/AP_REP exchange (see [I-D.hardjono-oauth-kerberos] (Hardjono, T., “OAuth 2.0 support for the Kerberos V5 Authentication Protocol,” June 2010.) for a comparision between Kerberos and OAuth).
To illustrate key confirmation the first examples borrows from Kerberos and symmetric key cryptography. Assume that Carol shares a long-term secret with Bob, called K(Carol-Bob). This secret would be established between them in an initial registration phase outside the scope of the actual protocol run. When Alice requests a token Carol creates a fresh and unique key Ks and places it into the token encrypted with K(Carol-Bob). Additionally, Carol attaches Ks when it sends the token to Alice over a confidentialy protected channel. When Alice sends a request to Bob it has to use Ks to sign the request (in whatever form or whatever layer). Bob, when receiving the message, retrieves the token, verifies it and uses K(Carol-Bob) to decrypt Ks and to verify the signature.
Note that in this example one could imagine that the mechanism to protect the token itself is based on a symmetric key based mechanism to avoid any form of public key infrastructure but this aspect is not further eleborated in the scenario.
A similar mechanism can also be designed using asymmetric cryptography. When Alice requests a token Carol creates an ephemeral public / privacy key pair PK/SK and places the public key PK into the protected token. When Carol returns the token to Alice it also provides the PK/SK key pair over a confidentialy protected channel. When Alice sends a request to Bob it has to use the privacy key SK to sign the request. Again, the details are secondary. Bob, when receiving the message, retrieves the token, verifies it and extracts the public key PK. It uses this ephemeral public key to verify the attached signature.

TOC

5. Summary

In the design of the OAuth security mechanisms the following design decisions have to be made:

> **Threats:** Section 3 (Security Threats) lists a few security threats. Are these the threats you care about? Are threats missing?
>
> **Threat Mitigation:** Section 4 (Threat Mitigation) illustrates how to address the threats listed in Section 3 (Security Threats). Do you agree that these are the approaches to address the threats? Do you agree that the three approaches to address token re-use (see Section 4.1 (Confidentiality Protection), Section 4.2 (Sender Constraint), and Section 4.3 (Key Confirmation)) are roughly equivalent from a security point of view (even though they are not equivalent from an operational perspective)?
>
> **Token Protection and Token Content:** Do you agree that many security properties are dependent on the token content and the token protection?

---

6. Operational Considerations

TOC

It is worth pointing out that the operational aspects in the deployment of OAuth have an impact on the design. While the authors believe that the three approaches to address token re-use are equivalent there are operational challenges with each of them. The purpose of the discussion in this section is therefore to determine, which approach encourages good practices that lead to better security of the overall system based on the most likely behavior of its actors.
The three approaches are:

> **Confidentiality Protection:** The weak point with this approach, see Section 4.1 (Confidentiality Protection), is that Alice has to be careful to whom she discloses the token. Everyone who is in possession of the token is granted access to the data. Furthermore, Alice has to trust Bob to secure the token at rest; unauthorized disclosure could be harmful particularly when the token has a rather long lifetime. Intiuitively, one would argue that tokens are easy to mint and used for a small time duration only.
>
> **Sender Constraint:** The weak point with this approach, see Section 4.2 (Sender Constraint), is to setup the authentication infrastructure such that Alice can be authenticated towards Bob. Additionally, Carol to correctly encode Alice's identity in the

token for verification by Bob. Depending on the chosen layer for providing client-side authentication there may be additional challenges due Web server load balancing, lack of API access to identity information, etc.

Key Confirmation: The weak point with this approach, see Section 4.3 (Key Confirmation), is the increased complexity: a key distribution protocol has to be defined.

7. Security Considerations

TOC

The main focus of this document is on security. Nevertheless, the authors would like to point out that the design of the stoken encoding and token content is currently not standardized. While this motivated by the current OAuth deployment environment and the desire to offer flexibility there are concerns about the ability of those deploying OAuth making reasonable design choices. We recommend to consult 'How to Implement Secure (Mostly) Stateless Tokens' [I-D.rescorla-stateless-tokens] (Rescorla, E., “How to Implement Secure (Mostly) Stateless Tokens,” March 2007.) before designing a custom token format.
RFC 4962 [RFC4962] (Housley, R. and B. Aboba, “Guidance for Authentication, Authorization, and Accounting (AAA) Key Management,” July 2007.) gives useful guidelines for designers of key management protocols. While the document was written with the AAA framework and network access authentication in mind the offered suggestions are useful for the design of OAuth and particularly interesting for solutions belonging to the 'key confirmation' class. These requirements include

1. Cryptographic algorithm independent

2. Strong, fresh session keys

3. Limit key scope

4. Replay detection mechanism

5. Authenticate all parties

6. Keying material confidentiality and integrity

7. Confirm ciphersuite selection

8. Uniquely named keys

9. Prevent the Domino effect

10. Bind key to its context

11. Confidentiality of identity

12. Authorization restriction

The authors believe it is useful to consider these requirements in the protocol design since the 'token' terminology seems to implicitly suggest to hand out cryptographic keying material and meta-data for usage in multiple, potentially unbounded contexts, with a (very) long lifetime.

---

8. Conclusion

TOC

The authors argue that the best approach for providing security for OAuth is to

1. describe security threats similar to the writeup in Section 3 (Security Threats).

2. offer recommendations for the protection and the content of the token. A document with a detailed description of the token encoding and the token content (including security protection) is likely to provide users with help in designing their own custom extensions. This document does not need to be part of the OAuth 2.0 base specification. [I-D.rescorla-stateless-tokens] (Rescorla, E., “How to Implement Secure (Mostly) Stateless Tokens,” March 2007.) and the Kerberos Token format [RFC4120] (Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” July 2005.) will provide valuable source of inspiration.

3. define a mandatory-to-implement solution based on 'key confirmation' (using symmetric keys) since it provides the fewest number of operational drawbacks. A symmetric key based approach was chosen because of performance reasons even though it requires Carol and Bob to share a long-term secret. It is highly recommended to compare the design against the requirements outlined in [RFC4962] (Housley, R. and B. Aboba, “Guidance for Authentication, Authorization, and Accounting (AAA) Key Management,” July 2007.).

4. strongly recommend the usage of TLS between Alice and Bob mainly for the additional security services TLS provides, such

as confidentiality protection. TLS will be useful for access to protected resources for the exchange of sensitive information. In case that server-side authentication is a concern the usage of channel bindings [RFC5056] (Williams, N., “On the Use of Channel Bindings to Secure Channels,” November 2007.) should be investigated since they allow binding an anonymous Diffie-Hellman exchange during the TLS handshake with the high-layer security exchange.

9. IANA Considerations

TOC

This document does not require actions by IANA.

10. Acknowledgments

TOC

The authors would like to thank the OAuth working group for their discussion input.

11. References

TOC

11.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.
[I-D.ietf-oauth-v2]	Hammer-Lahav, E., Recordon, D., and D. Hardt, “The OAuth 2.0 Protocol,” draft-ietf-oauth-v2-10 (work in progress), July 2010 (TXT).
[RFC4949]	Shirey, R., “Internet Security Glossary, Version 2,” RFC 4949, August 2007 (TXT).

11.2. Informative References

TOC

[RFC4279]	Eronen, P. and H. Tschofenig, “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS),” RFC 4279, December 2005 (TXT).
[RFC5246]	

	Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008 (TXT).
[RFC4120]	Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” RFC 4120, July 2005 (TXT).
[I-D.hardjono-oauth-kerberos]	Hardjono, T., “OAuth 2.0 support for the Kerberos V5 Authentication Protocol,” draft-hardjono-oauth-kerberos-00 (work in progress), June 2010 (TXT).
[I-D.rescorla-stateless-tokens]	Rescorla, E., “How to Implement Secure (Mostly) Stateless Tokens,” draft-rescorla-stateless-tokens-01 (work in progress), March 2007 (TXT).
[RFC4962]	Housley, R. and B. Aboba, “Guidance for Authentication, Authorization, and Accounting (AAA) Key Management,” BCP 132, RFC 4962, July 2007 (TXT).
[RFC5849]	Hammer-Lahav, E., “The OAuth 1.0 Protocol,” RFC 5849, April 2010 (TXT).
[RFC5056]	Williams, N., “On the Use of Channel Bindings to Secure Channels,” RFC 5056, November 2007 (TXT).
[NIST800-63]	Burr, W., Dodson, D., Perlner, R., Polk, T., Gupta, S., and E. Nabbus, “NIST Special Publication 800-63-1, INFORMATION SECURITY,” December 2008.

Authors' Addresses

TOC

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland
Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Blaine Cook
BT
Ireland
Email: romeda@gmail.com