Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 6, 2012

Y. Oiwa
RCIS, AIST
T. Hayashi
B. Kihara
Lepidum

July 5, 2011

HTTP authentication: problem statement draft-oiwa-http-auth-problem-statement-00

Abstract

This document discusses about existing problems in the current authentication technologies around HTTP and some analysis of the requirements for future authentication technologies in HTTP area.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

User authentication is, needless to say, one of the most important building block for the Web applications and other Internet-based systems. As social activities and commerce systems are more and more widely spreading, the importance for the security of the authentication also increase. Furthermore, the recent movement of providing government services on the Web requires user authentication as a key feature. Impersonation of client users in such systems may cause unrecoverable damages such as loss of credits, trusts, or social statuses.

At the same time, the authentication is currently one of the weakest blocks in terms of security. Intrinsically, the Web system as a whole is a multi-party system where the malicious servers cannot be rejected from the world. Unlike other systems such as email (POP [RFC1939] or IMAP [RFC3501]) or VPN (IPsec [RFC4301], L2TP [RFC2661] etc.) where the communication peer is typically pre-configured in the client software, Web clients (Web browsers) communicate with any party which the user insists to. This property leaves malicious servers to forge users to communicate with himself and performs a fiddle with the victim. Once such an attack succeeds, its result is severe: not only user's passwords to be stolen, users are often fooled to provide more critical information such as credit card numbers to the attackers.

On contrary to the current design, the authentication on the Web systems should be bidirectional and mutual: not only the authenticity of the users, but the authenticity and integrity of the servers is really important for protecting user resource stored on the server side. Most users assume that the successful user authentication also implies that they are talking with the genuine server entity: unfortunately, for almost all currently-deployed technologies on Web authentication this is not true, even for the TLS [RFC5246] client certificate authentication.

The motivation of this document is to promote ideas of replacing current systems and mechanisms for authentications on the HTTP/Web systems by more secure building blocks which are carefully designed for both security and usability/deployability. In the following sections, currently available methods of authentication on the Web systems are reviewed, and existing problems are discussed. At last, we conclude with possible action plan proposals for the community.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Existing authentication mechanisms

(To be described)

3. Background: existing threats and contributing factors

3.1. Impersonation of server identity (Phishing)

The term "Phishing" here is used as a generic term of attacks involving some kind of spoofing or impersonation used to socially/technically fool the victim users. From the beginning of the Web system, such "false web sites" are considered as a problem. TLS and its predecessor, SSL, have introduced a PKI-based server identity checking [RFC3647] using trusted certificate authorities (CAs) to address this issue. However, more and more sensitive and valuable information Web systems become to handle, more and more the Phishing attacks become "useful" attack vector. Such Phishing sites typically steal user identity and plain-text passwords from the victim user, and use it to either access sensitive data (such as Web mail services for critical officers), to fool users to provide more sensitive informations to the attackers (such as credit card numbers), or to impersonate victims with a malicious social activities (such as selling stolen items in a net auction).

3.2. Impacts of server-side password database leakage

Technically speaking, security of the server-side data is an out-of-scope issue for the HTTP authentication. However, we should be aware that many real-world security bleaches actually caused (partially) by the leakage of server-side stored information. Impact of such server-side leakage depends on how the authentication mechanism are designed. For Basic authentication [RFC2617], server-side credential used for password verification is usually one-way hashed with a random salt, which mitigates risk of server-side leakage a bit. Public key cryptography-based authentication, if correctly managed and deployed, can also avoid storing of sensitive client-credentials to the server-side. On the contrary, Digest authentication and other hash-based authentication schemes (e.g. CRAM-MD5 in SASL [RFC4422], APOP, etc.) requires raw client-side credentials to be stored in the server side by its nature. Of course, if all other properties are similar, the algorithms which do not require raw client credentials in server side is preferable as far as possible.

3.3. Impacts of complex authentication/authorization technologies

Recently, many complex framework for authentication and authorizations are deployed to realize multi-party authentication. For example, OpenID and SAML [OASIS.saml-core-2.0-os] gives servers an opportunity to authenticate users using identities provided by third parties. OAuth [RFC5849] allows a user to delegate some access rights to servers or other systems without giving client authentication credential itself.

Introduction of such services can have both positive and negative impacts on the Web security. For example, federated multi-party authentication can reduce number of client credentials which are required to access many services, which can reduce risk of server-side information leakage. At the same time, it requires user to authenticate himself in dynamically-generated web pages in the middle of complex page redirection flows, which makes users difficult to discriminate whether the page is a correct one to input her user name and password, increasing risk of Phishing attacks. Detailed analysis of its security, including user experiences in its consideration, might be needed.

Also, in such systems single security bleach among multiple parties involved to federated authentication may or may not impact security of other systems and their users. There are many pitfalls in implementation of such multi-party protocols, such as session fixation, session hijacking and cross-site request forgeries. Especially because most of those technologies are implemented often in application layer, careful observation and analysis of such bleaches caused by mis-implementation of a single party should be performed.

4. Applicable fields for HTTP authentication

Nowadays HTTP is used as a common foundation for various systems including Web systems and other non-Web applications. Depending on the nature of each systems using HTTP, the required properties for the underlying authentication/authorization mechanism may vary. Although the comprehensive analysis of all existing applications are impossible, this document hereby proposes categorizing use cases to three typical groups as a starting point, for further analysis in the sections below.

4.1. Web user authentication

The first group is to authenticate users of common Web applications, typically using Web browsers as clients. This group is very common use cases which exists from very early stage of HTTP, and the ones for which currently existing HTTP authentication mechanisms are designed.

One of the most important security consideration in these scenarios is complex interactions between human users, browser clients and Websites including those of uncontrollable third-party entities. Phishing is a very common attack vector for this scenario, and without having some weapons against protecting authentication credentials and integrity, it is impossible to stop malicious phishers from deploying such attacks.

Authentication credentials used for these scenarios varies for required authentication strength and several social factors, for example passwords, cryptographic secret keys, smart cards, one-time passwords, etc., but in most cases such credentials are belonging to human entities. There is several proposals for unified and federated authentications, but this principle does not change.

4.2. Web client application data accesses

Recently, capability of client-side data processing in Web browser clients are greatly improved, and it introduced a new pattern of client-server relationship in Web applications: Web-application initiated data accesses.

In ancient Web systems, clients are only communicating with the corresponding server providing the current Web page, and if some external data accesses are needed, the server will perform it, process the received data and serve the result to client as a static data embedded in Web contents. In this scheme, the user authentication mentioned above is only necessary, and all other authorization managements are performed in the server-side.

However, evolution of client-side data processing changed the whole story. Now the client-side application can request authentication of itself as an agent of the human user, obtain authorization rights and access several data resources in various servers. Such authorization rights are not needed to be directly corresponding to the active human user's rights: it can be of another user's rights delegated to a user (like what the OAuth protocol [I-D.ietf-oauth-v2] provides), or it can be subset of what the user has access to. In this story, the authentication/authorization of client application are related but not directly connected to the human user's authentication.

Some important points in this group might be flexibility: application-level authentication can be either related or unrelated to human entity, and same application may need to provide more than one methods of authentications in the same framework, possibly with different levels of authorizations.

In this use cases, phishing is not always a key factor for threat analysis. If Web applications are itself faked and thus provided from phishing sites, the user cannot trust provided data regardless of whether the data resource servers accessed are trustful or not. On the contrary, provided Web applications are trustful, these programs typically (but not always) "know" what server is the correct server to interact. Some mutual and eavesdropping-safe authentication technologies are still useful, as many applications nowadays still need some communication in an unencrypted channels because of overhead of secure-channel provisions.

4.3. Non-Web user authentication

The final group is use-cases for non-browser client applications. Nowadays HTTP is becoming a common vehicle for various applications including non-browser clients. Because of its simplicity, many existing services are providing both Web-client and non-Web API accesses using the same HTTP platform. We should not ignore such use cases when considering solutions for the above two groups.

In some aspects, required features for this group of applications are subsets of the above two use cases. Simple user authentication may be mapped to an HTTP authentication scheme provided for the "Web user authentication", and some detailed authorization cases may be mapped to an access grant management used for "Web application authentication" stories. However, because we cannot rely on any aid from Web pages and scripts, some technologies for these groups may be not useful for non-Web applications, or careful design consideration may be needed for applying those to this group of use cases. For an instance, OAuth usually relies on Web-based authentication and page redirections, but to support non-Web application use cases it required some additional features as well. Also, integration with existing authentication framework such as SASL [RFC4422] or GSSAPI [RFC2743] might be important especially in this use cases.

5. Problem statements

5.1. Lack of mutual authentication

Most authentication technologies which are currently used on Web systems are essentially one-directional. A server always checks authenticity of users using client authentication credential, but a user has little control of the process and a user can not know whether the talking peer is the intended entity, or they can not know whether the server is actually performing an authentication and has knowledge of the user. This has been a cause of many Phishing attack instances. All of HTTP Basic authentication, Digest authentication, HTML Form authentication and even TLS client certificate authentication fall into this category of technologies. TLS server authentication are thought to mitigate this factor, but it was too weak to prevent many Phishing attacks. The TLS server authentication only certifies that the server has (in some sense) a right to legitimately use the domain name that the client accessed, and optionally binds the server with some real-world entity. In fact, some phishing sites use their own domains with a valid server certificates, or others use a cracked servers with legitimate certificates to perform an attack. In this situation, the server authentication does not prevent Phishing technically, instead it relies on the careful manual investigation of domain names by an end user.

For secure use of Web systems mutual authentication between users and servers has critical needs. By performing mutual authentication, a user can assure that the peer server has certainly performed an authentication, and that the peer has a prior knowledge of the user, eliminating possibility of man-in-the-middle attacks or false authentications. We should investigate possibilities of performing mutual authentication using various kinds of authentication credentials: passwords (weak secret), strong shared key, multi-factor credentials and even cryptographic public/private key pairs if possible.

This requirement is mainly applicable for both Web and non-Web user authentication. For simple access patterns (where user authentication coincide with data access authorization), it may be also applicable for application data accesses (both Web and non-Web).

5.2. Avoiding use of plain-text passwords on authentication

Two most widely used authentication technologies, Basic and Form-based authentication, uses plain-text passwords as credentials and send these directly on wire. Obviously, using those technologies without encryption will reveal any secret credentials to all eavesdroppers. Even if TLS encryption is used, on-wire plaintext passwords are vulnerable for Phishing and (Web application layer) man-in-the-middle attacks. This weakness is amplified when users are using a single password for several independent systems. Especially, using plaintext passwords in Form-based authentication required handling of passwords in a Web application layer, which has caused many password leakage accidents in many commercial websites. To avoid these problems, we need technologies which prevents leakage of reusable weak secret.

This item applies to all of the previously-mentioned use cases. Especially when applications needs use of unencrypted channels for performance reasons, it is crucial to protect authentication credentials and tokens.

5.3. Functional weakness of HTTP authentication framework

Current basic design of HTTP authentication framework [RFC2617] does not sufficiently provide the features which are required by current Web application logics. This is currently one of the biggest reason why many Web developers prefer HTML Form-based authentication more than HTTP Basic authentication. For example, current HTTP authentication framework lacks support for non-mandatory authentication (aka guest user support), and enforcement of login session termination (log-out control). It also removes application developers detailed control of user experiences, because most of interactive HTTP clients (Web browsers) uses a modal, interruptive dialog user interface for authentication. Some authentication schemes, notably TLS client certificate authentication are further worse, as a single server must serve a single set of authentication, and users can not use several identities simultaneously within one server.

However, due to the nature of HTML form and UI designs, it is almost impossible to fundamentally improve security of Form authentication, mainly due to the fact UI of such authentication can be always faked and imitated. For example, if we had a secure input field for passwords in some HTML extensions, Phishers would simply forge it with usual password field instead to steal any passwords inputs. To avoid this problem, the user agent (browsers) and the HTTP protocol must serve a role of securely handling authentication and user credentials. To make use of such agent-driven authentication in real applications, the authentication framework should be flexible enough so that the application developers can realize any application logics they require for the user and session managements.

This requirement is mainly applicable for browser-based Web user authentication. The provision of such features should be compatible with other use cases, however.

5.4. Lack of bindings between multi-layer authentications/authorizations

Many recent Web applications are implemented in multi-layer technologies and each layer often has own control of authentication and authorization. For example, OAuth [I-D.ietf-oauth-v2] enables application-level delegated access authorization using credentials issued on another authentication/authorization framework. W3C proposed WebID uses result of TLS client authentication for control of upper-layer identification and authorization. The channel binding is a key technology to implement such multi-layer applications. Simply speaking, the channel binding is a technique which relates an upper-layer authentication with a result of lower-layer authentications/key-exchanges. By doing that, any result of upper-layer authentication can not be separately used on any other lower-layer channels which are not authenticated by the same way. For example, by using TLS channel binding [RFC5929] with a signed OAuth request, such request tokens can not be used by any other people to access the protected resources, even if the token has been leaked in some way. Unfortunately, current HTTP-layer authentication schemes does not provide functionality for such channel bindings. Future schemes should consider providing such binding functionality as its building blocks.

(to be mentioned: OAuth HTTP MAC authentication [I-D.ietf-oauth-v2-http-mac])

6. More topics

(TBD)

7. IANA Considerations

None.

8. Security Considerations

This document obviously deals with security technologies. However, the purpose of this document is not to provide specific protocols or technologies to be directly implemented, but to discuss about current status of existing technologies and requirements for future technologies. Therefore, there is no specific security precautions to be mentioned here. When designing some specific technologies mentioned in this document, we MUST have careful consideration of security properties, because the technology area handled in this document has very complex and legacy characteristics and limitations.

9. References

9.1. Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

9.2. Informative References

[I-D.ietf-oauth-v2] Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol," draft-ietf-oauth-v2-16 (work in progress), May 2011 (TXT, PDF).

[I-D.ietf-oauth-v2-http-mac] Hammer-Lahav, E., Barth, A., and B. Adida, "HTTP Authentication: MAC Access Authentication," draft-ietf-oauth-v2-http-mac-00 (work in progress), May 2011 (TXT, PDF).

[OASIS.saml-core-2.0-os] Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," OASIS Standard saml-core-2.0-os, March 2005.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3," STD 53, RFC 1939, May 1996 (TXT).

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, June 1999 (TXT, HTML, XML).

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"," RFC 2661, August 1999 (TXT).

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1," RFC 2743, January 2000 (TXT).

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1," RFC 3501, March 2003 (TXT).

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework," RFC 3647, November 2003 (TXT).

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

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)," RFC 4422, June 2006 (TXT).

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

[RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol," RFC 5849, April 2010 (TXT).

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS," RFC 5929, July 2010 (TXT).

Authors' Addresses

Yutaka Oiwa
National Institute of Advanced Industrial Science and Technology
Research Center for Information Security
AIST Tsukuba Headquarters' building
Tsukuba Central 2
1-1-1 Umezono
Tsukuba-shi, Ibaraki
JP
Phone: +81 29-861-5284
Email: mutual-auth-contact@m.aist.go.jp

Tatsuya Hayashi
Lepidum Co. Ltd.
#602, Village Sasazuka 3
1-30-3 Sasazuka
Shibuya-ku, Tokyo
JP

Boku Kihara
Lepidum Co. Ltd.