Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 26, 2013

H. Tschofenig
Nokia Siemens Networks
S. Turner
IECA, Inc.
M. Hanson
Mozilla

October 23, 2012

An Inquiry into the Nature and the Causes of Web Insecurity
draft-tschofenig-secure-the-web-04.txt

Abstract

The year 2011 has been quite exciting from a Web security point of view: a number of high-profile security incidents have gotten a lot of press attention but also new initiatives, such as the National Strategy for Trusted Identities in Cyberspace (NSTIC), had been launched to improve the Web identity eco-system. The NSTIC strategy paper, for example, observes problems with Internet security due to the widespread usage of low-entropy passwords and the lack of widely deployed authentication and attribute assurance services.

With this memorandum we try to develop a shared vision for how to deal with the most pressing Web security problems.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2013.

Table of Contents

1. Introduction

HTTP is an IETF standard and documented in RFC 2616 [RFC2616] and provides the core foundation of the browser-based platform but is also widely used for non-browser-based applications in smart phones and Internet tablets. Like any other specification in the IETF HTTP also comes with various security mechanims. Digest authentication support in HTTP was published in 1997 with RFC 2069 [RFC2069] and later updated in 1999 by RFC 2617 [RFC2617]. The HTTP state management mechanism, namely cookies, was initially published in 1997 with RFC 2109 [RFC2109], revised in 2000 by RFC 2965 [RFC2965], and obsoleted by RFC 6265 [RFC6265].

For client side authentication for HTTP-based protocols two different solution tracks have been offered from the IETF, namely TLS client side authenication and also application level authentication via HTTP basic and digest. TLS-based client authentication using certificates was quite complex for end users to configure (and still is today). HTTP based authentication on the other hand did not found widespread usage either for a number of reasons. First, the user interface was rendered differently than regular Web application forms making it less attractive for Web developers and users. At that time HTTP had a semantic that was closer to file system access control and therefore the decision making process was binary, either the user was granted access to the resource or it wasn't. With the HTTP 401 there was no way for a user to, for example, recover from a lost password or other forms of failure cases. The authentication and authorization process was not seen as continuium but rather as a binary decision. For these reasons form-based authentication mechanisms had found widespread acceptance by the Web application developer community.

Many Web sites decided to deploy their own authentication infrastructure and to store cleartext credentials, since most of them use password-based authentication. As reported in a New York Times article from October 2012 [NYT-2Factor] a recent analysis of a leaked large password database revealed that among 3.4 million passwords (among the 30.3 million passwords) consisting of nothing but four digits. The top 20 passwords account for nearly 27% of the total. This even makes online guessing attacks feasible. Users also share the same password across multiple sites making it easy for an adversary to utilize credentials obtained from one site to also gain access on other Web sites.

Breach notification laws forced companies to inform their customers about incidents. Consequently, the community became aware of the degree of password leakage due to unauthorized access to these credential databases. For example, in April 2011 Sony experienced a

data breach within their PlayStation Network and 100 million users accounts were compromised. This is only one out of thousands of data breaches collected by the Privacy Rights Clearing House [DataBreach]. In most cases these security vulnerabilities are due to misconfiguration, security vulnerabilities in software which can be exploited via buffer overflow attacks, and SQL injection due to insufficient input parameter verification.

In addition, cookies are still the most common mechanism for session management, i.e., a non-cryptographic way to bind the initial, often better protected, authentication procedure to the subsequent protocol exchanges. The non-cryptographic session management gives attackers the ability to perform session hijacking. This is of particular concern when users access Internet services using insecure WLAN hotspots. Firesheep [FireSheep], a Firefox plugin that worked as a packetsniffer demonstrated this vulnerability to the non-expert community and made session hijacking 'friendly to use' for a broader community.

A number of trends had been observed during the last couple of years, as briefly summarized below.

From Documents to Mobile Code: During the last 10 years the Web has changed quite fundamentally with the widespread usage of JavaScript. While Web pages have for a long time been dynamically generated the ever increasing capabilities of JavaScript, with respect to functionality and performance, have changed the security model. A typical Website collects content from multiple other Web sites and delivers it to the user's browser and by delivering code inside HTML new security challenges have emerged. Also the standardization landscape had been challenged by this new development and [I-D.tschofenig-post-standardization] documents architectural implications.

Mashups and Data Sharing: With the increasing specialization of Web sites developers started to outsource functionality to other sites. Partially this is a user-convenience aspect (e.g., users do not want to create a new address book with every site, publish their latest status on each and every site again and again) but often also driven by business interestes. In any case, the need to access resources hosted on other sites emerged and often these resources were not visible to everyone. Sharing long-term passwords is considered a bad habit and consequently the Web Authorization (OAuth) protocol [RFC6749] started to become used widely. OAuth avoids the need to share long-term credentials with random Web sites.

The Real-Time Web: As HTTP became the protocol of choice for many application developers, also because of it's ability to go through firewalls and NATs, requirements for asynchronous protocol communication had to be addressed as well. HTTP, as a request/response protocol, was initially not designed for pushing data from the server-side to the client as soon as it is available. Long polling requests and other tricks had been used to allow bi-directional communication between the HTTP client and the HTTP server. The efforts in the BiDirectional or Server-Initiated HTTP (hybi) working group improves the communication capabilities of HTTP. To allow one Web browser to communicate directly with another Web browser the same-origin security framework utilized by the browser has to be bypassed and the work on Real-Time Communication in WEB-browsers (rtcweb) was created to develop a architecture [I-D.ietf-rtcweb-security] and in [I-D.ietf-rtcweb-overview]. Extending Web clients with real-time communication capabilities opens the doors for a large number of applications that had previously only been available for downloadable applications.

With the increasing number of security challenges and developments in the Web application environment the standards community was challenged to initiate activities. Examples include the development of HTTP Strict Transport Layer Security (HSTS) [I-D.ietf-websec-strict-transport-sec] that allows Web sites to declare themselves accessible only via secure connections. The attempt to clarify the Web Origin Concept [I-D.ietf-websec-origin], which covers the principles that underlies the concept of origin as used to scope of authority or privilege by user agents. The work around Content Security Policy (CSP) [CSP] that allows Web application developers to declare a set of content restrictions for a web resources. The OAuth protocol that allows secure and privacy-friendly sharing of resources. The work on Javascript Object Signing and Encryption to give Web developers better ways to protect the exchange of JSON data structures. The W3C Web Cryptography API that defines JavaScript extensions that enables developers to implement secure application protocols within Web applications.

A lot has changed over the last 10 years in the Web eco-system. While astonishing progress has been made in getting the Web application eco-system on a par with native applications the foundation of the Web platform is still unable to address many of the most common security vulnerabilities; problems the Internet community had been fighting against for over a decade and had solved for other application protocol frameworks and Internet deployment environments.

It is time to tackle this problem and to develop a common understanding of the problem and the desired design goals.

2. Terminology

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 [RFC2119].

3. Passwords

Passwords have a long history in authentication protocols on the Internet. They appear to be convient for users and are easy to provision to users by many Web site. Still, passwords present a number of challenges, including:

- Users re-use the same password at multiple sites. This allows a rouge Website provider to attempt to impersonate users on other sites. It also allows a hacker to use stolen passwords obtained from one site to be used at a non-compromised site.

- Password are stored in cleartext in most cases. In case of a data breach account information, including the password, becomes accessible to an attacker.

- Users are tricked in typing their password into a Website maintained by phishing attempts. Furthermore, some Websites request username and password for access to protected resources maintained by other Websites. While there are technical ways to avoid the need for such long-term password sharing practice using OAuth some Websites still ask users.

- Many password based authenication protocols are not secure against eavesdropping, or allow easy ways for offline dictionary attacks.

- When end systems are compromised as well then a keyboard logger can capture any password sequence a user enters.

So, why do we need passwords at all? It is easy to come up with solutions that use hardware-based mechanisms (e.g., such as OTP tokens), mobile phones, etc. [Quest] lists some of these mechanisms and makes an attempt to classify them. Many of the analysed authentication mechanisms provide additional security but have design limitations regarding usability and incremental deployment. There are, however, reasons why alternatives have not found widespread deployment on the Internet, such as

- Passwords are cheap (at least the primary costs) for user's and service providers. Hardware tokens on the other hand have a certain amount of cost associated with them.

- Provisioning new users with passwords is easy. Tools and processes exist and are widely accepted.

- Service providers have no external dependency when they manage user accounts themselves (unlike with many third party identity management solutions).

- Users are familiar with password-based systems and the acceptance is good.

- Passwords can easily be delegated to others.

- Users typically feel quite secure when they are using shared secrets and it fits into their mental model of self-securing.

- Passwords can easily be transferred to multiple devices used by a single user.

Note that the credential type and the actual form of where these credentials are stored (e.g., software, hardware) is orthogonal to the actual identity proofing process. Stronger forms of identity proofing (e.g., requiring in-person passport verification) can be quite expensive. There are also secondary costs in the form of support calls and education if credential provisioning is more complicated, as it is often the case with client certificates.

Regardless how many disadvantages passwords have they will be with us for a long time. As such, out attempt is therefore to start from the currently deployment and to look towards a future where fewer of them are used, and if they are used then in a more secure fashion.

4. Roadmap

It is our aim to accomplish three types of goals:

1. Reduce the number of passwords used on the Web

2. Increase security of how passwords are used (for example using two-factor authentication). With the RSA patents for one-time password based authentication expiring the usage of the work by the Initiative for Open Authentication (OATH) with their HMAC-Based One-time Password (HOTP) algorithm (RFC 4226 [RFC4226]) and the Time-based One-time Password (TOTP) algorithm (RFC 6238 [RFC6238]) has increased.

3. Broaden the use of other, non-password-based credentials. The weaknesses related to compromised password databases and the unauthorized access to these stored credentials is difficult to avoid entirely without switching to stronger credentials or without outsourcing those functions to specifialized third party identity providers.

A non-goal of this document is to evaluate ways for improving identity proofing, which is a requirement for accomplishing higher levels of assurance.

We do not believe that the technical community should be attempting to come up with the single and best solution to satisfy these three goals. We would like to leave room for innovation and allow many different solutions to co-exist to best suite their deployment.

Subsequently, we try to highlight a few guiding principles in an attempt to come with a way forward.

Move Authentication down into the Platform:

> Exposing authentication protocol functionality to the user and requiring Web application developers to write security related code has proven to lead to problems. Avoid user interaction related to security whenever possible but keep in mind that authorization decisions, particularly with regard to data sharing, require a consent. Ensure that library support is available for Web developers to allow them easy integration of security functionality into their applications. Unfortunately, a protocol design also needs to consider the transition scenario where the Web endpoints are not yet upgraded to support the new functionality and that the authentication functionality is not yet available.

Design for Growth:

No single authentication mechanism nor credential type is able to fulfill all use cases. Design for later extensions and develop the protocol architecture in such a way that components are interchangable. In particular, there are a number of authentication mechanisms already in use in other deployment environments.

Context Matters:

Users require context for all disclosures and the sequence of interactions matters. A monolitic authentication protocol that provides mutual authentication is less likely going to capture the context related disclosures. Server-side authentication is the first interaction that will have to be provided to guarantee genuine content as well as the prerequisity of an early setup for a confidentiality protected channel. Client side authentication may, however, come at a much later stage of the application interaction. It is often bundled with an authorization decision where different application execution paths depend on the level of authorization.

Discussion: Is it indeed given that client authentication will have to happen at a later stage given that platform-level authentication proliferates (particularly or mobile phone applications) and "authenticated by default" becomes the norm? If so, then strong signals in UIs of authenticated status, identity selection, and anonymous/pseudonymous modes become more important. One could compare this to the evolution in the telephony communication where caller ID information was initially not provided but became the norm later and blocking the caller ID instead became the expection.

Transform Long-Term Passwords to Short-Term Credentials:

One of the function of authentication protocols is to transform long-term credentials into short term secrets. Long-term credentials, such as passwords, require substantial protection in a protocol exchange and therefore this interaction often leads to a computationally expensive, multi-roundtrip protocol exchange. We do, however, encourage protocol designers to make heavier use of this transformation step into short term credentials.

Keep the User Experience in Mind:

Design your protocol stack in such a way that developers up the stack can give good advice to users. The use case analysis should include common failure scenarios since error paths need as much expressiveness as success paths, whereby expressiveness refers to the ability to communicate with the user about failure cases.

TLS Always:

The initial step of entity authentication cannot be seen in isolution of the ultimate purpose of securing an entire applicatio interaction that requires session management to take place. While this session management today happens in most cases in a non-cryptographic way (i.e., without data origin authentication) we believe it is time revisit this practice. Designing a new cryptographic session management concept is questionable given the already available tools, such as TLS that provides a secure session management using the TLS Record Layer. The usage of the Record Layer is, compared to the TLS Handshake protocol, fairly computationally less time-consuming

5. From Two-Party to N-Party

It would be short sighted to write about a topic like this without touching a commonly desired way to reduce the number of long term credentials: federated logins

Federated login allows a user to utilize his credential obtained from one organization, acting as the Identity Provider, for accessing a resource at another entity, who acts as a Relying Party. While this approach addresses some of our design goals it causes secondary problems to appear - particularly related to privacy.

The following issues in this transition from a two-party to a three-party model are to observe:

Discovery:

> How do the three parties find each other? In particular, how does the user (via his or her user agent) inform the relying party about the identity provider it wants to use? How does the relying party inform the user agent (and user) about the identity providers it is able and willing to interact with? Without any changes to the end device relying parties often display icons of identity providers: the more identity providers they support the more icons are displayed to the user. This is also known as the NASCAR problem.

Mutual Authentication:

> How do we ensure that each party is authenticated to each other?

Trust and Permissions:

> What information should the user share with the relying party and how can he be reassured that the information is used in the way he permitted? What information is needed by the Relying Party for the application specific functionality? How is the identity provider able to protect its users against misbehaving relying parties?

Collusion:

> There are three related properties systems should be tying to provide:

Relying parties can be prevented from knowing the real or pseudonymous identity of an individual, since the identity provider is the only entity involved in verifying identity.

Relying parties that collude can be prevented from using an individual's credentials to track the individual. That is, two different relying parties can be prevented from determining that the same individual has authenticated to both of them. This requires that each relying party use a different means of identifying individuals.

The identity provider can be prevented from knowing which relying parties an individual interacted with. This requires avoiding direct communication between the identity provider and the relying party at the time when access to a resource by the initiator is made.

Security:

Keeping data secure at rest and in transit is another important component of security and privacy protection. How can it be ensured that the interactions between the three parties are not manipulated? An identity provider providing services to many relying parties is exposed to increased risk of a at breach via an nauthorized access to the credential database. How can this increased security protection be provided?

Note: While this text talks about three parties there may well be more parties involved in the exchange. The role of the identity consists of a credential provider and an attribute provider that may be provided by different parties. Furthermore, attributes associated with personal data may be contributed by multiple attribute providers, not just by a single entity. There may also be additional parties involved in the communication between the identity provider and the relying party the trust path from the identity provider to the relying party.

6. IANA Considerations

This document does not require actions by IANA.

7. Acknowledgments

The content of this document has been created based on discussions with a number of persons, including

- Jeff Hodges
- Michael Garcia
- Adam Barth
- Brad Hill
- Dan Mills
- Ed Felton
- Tara Whalen
- Andy Steingruebl
- Tim Polk
- Dirk Balfanz
- Nico Williams
- Tobias Gondrom
- Julian Reschke

We would like to thank them for their input. We would also like to thank the participants of the May 2011 W3C Identity in the Browser workshop for their discussion feedback.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

[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.

[RFC2109] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2109, February 1997.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

8.2. Informative References

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

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

[I-D.ietf-websec-origin] Barth, A., "The Web Origin Concept", draft-ietf-websec-origin-06 (work in progress), October 2011.

[I-D.ietf-websec-strict-transport-sec] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", draft-ietf-websec-strict-transport-sec-14 (work in progress), September 2012.

[RFC2069] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and L. Stewart, "An Extension to HTTP : Digest Access Authentication", RFC 2069, January 1997.

[I-D.ietf-httpbis-p7-auth]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", draft-ietf-httpbis-p7-auth-21 (work in progress), October 2012.

[I-D.tschofenig-post-standardization]
Tschofenig, H., Aboba, B., Peterson, J., and D. McPherson, "Trends in Web Applications and the Implications on Standardization", draft-tschofenig-post-standardization-02 (work in progress), May 2012.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-04 (work in progress), June 2012.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for RTC-Web", draft-ietf-rtcweb-security-03 (work in progress), June 2012.

[RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, December 2005.

[RFC6238] M'Raihi, D., Machani, S., Pei, M., and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, May 2011.

[CSP] "Content Security Policy 1.0", July 2012.

[FireSheep]
"FireSheep, available at http://en.wikipedia.org/wiki/Firesheep", October 2012.

[NYT-2Factor]
"Doing the Two-Step, Beyond the A.T.M., New York Times, available at http://www.nytimes.com/2012/10/14/technology/two-step-verification-is-inconvenient-but-more-secure.html", October 2012.

[DataBreach]
"Privacy Rights Clearinghouse - Data Breaches, available at https://www.privacyrights.org/data-breach", October 2012.

[Quest] "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes, In

Proc. IEEE Symp. on Security and Privacy 2012 (Oakland 2012)", July 2012.

Authors' Addresses

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

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA

Phone:
Email: turners@ieca.com

Mike Hanson
Mozilla

Phone:
Email: mhanson@mozilla.com