Requirements for Web Authentication Resistant to Phishing

draft-hartman-webauth-phishing-09.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on February 19, 2009.
Abstract

This memo proposes requirements for protocols between web browsers and relying parties at websites; these requirements also impact third parties involved in the authentication process. These requirements minimize the likelihood that criminals will be able to gain the credentials necessary to impersonate a user or be able to fraudulently convince users to disclose personal information. To meet these requirements browsers must change. Websites must never receive information such as passwords that can be used to impersonate the user to third parties. Browsers should authenticate the website to the browser as part of authenticating the user to the website. Browsers MUST flag situations when this authentication fails and flag situations when the target website is not authorized to accept the identity being offered as this is a strong indication of fraud. These requirements may serve as a basis for requirements for preventing fraud in environments other than the web.
Table of Contents

1. Introduction .................................................. 4
  1.1. Purpose of this Memo .................................... 5
  1.2. Progress to Date ......................................... 6
2. Terminology .................................................. 7
  2.1. Passwords and Interface ................................ 7
  2.2. Requirements notation ................................... 7
3. Threat Model ................................................ 8
  3.1. Capabilities of Attackers ................................. 9
  3.2. Attacks of Interest ..................................... 10
4. Requirements for Authentication that Protects Credentials ....... 11
  4.1. Support for Passwords and Other Methods ............... 11
  4.2. Trusted UI ............................................... 11
  4.3. No Password Equivalents ................................ 12
  4.4. Mutual Authentication ................................... 13
  4.5. Authentication Tied to Request and Response .......... 14
  4.6. Restricted Identity Providers ........................... 15
  4.7. Protecting Enrollment ................................... 15
5. Is it the right Server? ....................................... 17
6. Iana Considerations ........................................... 19
7. Acknowledgments ............................................... 20
8. Security Considerations ....................................... 21
  9. References .................................................. 23
    9.1. Normative References .................................. 23
    9.2. Informative References ................................. 23
Appendix A. Trusted UI Mechanisms ............................... 25
Appendix B. Change History ....................................... 26
  B.1. Changes Since 08 ........................................ 26
  B.2. Changes since 07 ........................................ 26
  B.3. Changes since 06 ........................................ 27
  B.4. Changes since 05 ........................................ 27
  B.5. Changes since 02 ........................................ 28
  B.6. Changes since 01 ........................................ 28
Author’s Address ................................................ 29
Intellectual Property and Copyright Statements .................. 30
1. Introduction

Typically, web sites ask users to send a user name and password in order to log in and authenticate their identity to the website. The user name and plaintext password are often sent over a TLS [RFC4346] encrypted connection. As a result of this plaintext password protocol, the server learns the password and can pretend to be the user to any other system where the user has used the same user name and password. The security of passwords over TLS depends on making sure that the password is sent to the right, trusted server and on that server not exposing the password to third parties. HTTPs [RFC2818] implementations typically confirm that the name entered by the user in the URL corresponds to the certificate.

One serious security threat on the web today is phishing. Phishing is a form of fraud where an attacker convinces a user to provide confidential information to the attacker believing they are providing the information to a party they trust with that information. For example, an email claiming to be from a user’s bank may direct the user to go to a website and verify account information. The attacker captures the user name and password and potentially other sensitive information. Domain names that look like target websites, links in email, and many other factors contribute to phishers’ ability to convince users to trust them.

Typically the user names and password are not directly valuable to the phisher. However they can be used to access resources of value. For example a bank password may permit money transfer or access to information useful in identity theft.

It is useful to distinguish two targets of phishing. Sometimes phishing is targeting web authentication credentials such as user name and password. Sometimes phishing is targeting other confidential information, such as bank account numbers. This memo presents requirements that can be part of a solution to significantly reduce the effectiveness of the first category of phishing: provided that a user uses an authentication mechanism that meets these requirements, even if the user authenticates to the wrong server, that server cannot impersonate the user to a third party. However, to combat phishing targeted at other confidential information, the best we know how to do is help the user detect fraud before they release confidential information.

The approach taken by this memo is to handle these two types of phishing differently. The user is given new authentication mechanisms. If the user uses these mechanisms, they have strong assurances that their password has not been disclosed and that the ensuing data returned from the server was generated by a party that
either knows their password or who is authenticated by an identity provider (a third party involved in the authentication exchange in order to allow credentials to be used across a wider variety of websites) who knows their password. The server can then use confidential information known to the user and server to enhance the user’s trust in its identity beyond what is available given the social engineering attacks against TLS server authentication. If a user authenticates to the wrong server but discovers this before they give that server any other confidential information, then there exposure is very limited. The success of this solution depends heavily on whether the user uses the new authentication mechanisms; designing ways for users to tell if they are using the authentication mechanisms and encouraging users to use these mechanisms will be critical to achieving any security benefit from these requirements. The success of a solution to preventing the disclosure of other confidential information based on giving users information about whether they are authenticated to the right server depends on the user being able to take advantage of this information and choosing to do so.

The requirements presented in this memo are intended to be useful to browser designers, designers of other HTTP applications and designers of future HTTP authentication mechanisms.

These requirements and mechanisms that meet these requirements are not sufficient to stop phishing; at best, they form part of a solution. The World Wide Web Consortium proposes recommendations on user interface guidelines for web security context [WSCUIG]. These guidelines propose mechanisms that will make it more likely that users will detect fraud before authentication. Efforts to limit the effect of malicious software and to provide trustable software for authentication are also important. Efforts to track known frauds and alert users when they encounter fraudulent sites are also critical. Together, all these efforts may significantly reduce phishing.

1.1. Purpose of this Memo

In publishing this memo, the IETF recommends making available authentication mechanisms that meet the requirements outlined in Section 4 in HTTP user agents including web browsers. It is hoped that these mechanisms will prove a useful step in fighting phishing. However this memo does not restrict work either in the IETF or any other organization. In particular, new authentication efforts are not bound to meet the requirements posed in this memo unless the charter for those efforts chooses to make these binding requirements. Less formally, the IETF presents this memo as an option to pursue while acknowledging that there may be other promising paths both now and in the future.
1.2. Progress to Date

This non-normative section describes the author of this memo’s impressions of the current state of HTTP authentication with regard to these requirements.

In the spring of 2008, Microsoft demonstrated that with no change to the spec, GSS-API and NTLM HTTP authentication could be extended to support channel binding [RFC5056] [RFC4559]. At first glance, the Microsoft extension appears to meet all the requirements outlined in this memo for an authentication mechanism. In addition, Microsoft has outlined extensions to HTTP digest authentication that also appear to meet these requirements [DIGEST-BIND]. The Microsoft extensions do not provide the client with information on whether the server supports the extension; so the client may not know whether it is strongly authenticated or not. Also, the Microsoft extensions are focused for enterprise deployment and so concerns regarding upgrade negotiation and other issues that would be important in a wider deployment are not covered. However Microsoft’s efforts underscore that new security mechanisms are not needed in order to meet these requirements. Originally, I had expected that changes to meet these requirements would be more extensive, but still expected they would be incremental changes to existing mechanisms.

However there is still work that needs to be done in order to make mechanisms meeting these requirements available in a usable manner across the Internet. Most of that work concerns usability and falls outside the IETF; Results of the usability work may fall within the IETF; mechanisms for picking the right credentials to use for a given site may require minor extensions to security mechanisms. Mechanisms to provide smooth upgrades from plaintext password protocols to mechanisms meeting these requirements may require additional HTTP headers, particularly for non-browser agents. In addition, these requirements may be useful to efforts that are designing HTTP authentication mechanisms for unrelated reasons.
2. Terminology

2.1. Passwords and Interface

There are two related concepts: the user interface of passwords and plaintext password protocols. A plaintext password protocol is a protocol where the server receives credentials sufficient to impersonate a user to third parties. A password interface provides a user experience where a user types a password into any computer, including one they have never used before and that is sufficient to authenticate. The requirements in this memo require support for password user interfaces as one option for authentication. The requirements of this memo are incompatible with plaintext password protocols.

2.2. Requirements notation

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. Threat Model

This section describes the assumed capabilities of phishers, describes assumptions about web security and describes what vulnerabilities exist. Human factors issues contribute significantly to these vulnerabilities. For example, security information dialogues in web browsers can provide information on the subject of a certificate. However, users rarely examine this information, so an attacker could be successful even if examining the security dialogue would show an attack. This threat model is intended to include these sorts of attacks and so it is broader than the technical threats against protocols. Efforts are under way to improve these human factors issues [WSCUIG]. However these efforts only reduce the risk that a user will be confused; even given improved user experience for dealing with security context information, users will make mistakes and believe that an attacker’s site is the site they intended to communicate with.

We assume that the implementations of authentication systems can be trusted sufficiently to meet their design objectives. This does not mean that the entire local system and browser need to be trusted. However if there is a component that has access to users’ passwords, that component needs to be secure enough to be trusted not to divulge passwords to attackers. Similarly in a system that used smart cards, the smart cards would need to be trusted not to give attackers access to private keys or other authentication material. Designing implementations to limit the size and complexity of the most trusted components and to layer trust will be important to the security of implementations. Designing protocols to enable good implementation will be critical to the utility of the protocols. As a consequence of this assumption, these requirements are insufficient to provide protection against phishing if malicious browser extensions, Trojan software or other malicious software is installed into a sufficiently trusted part of the local computer or authentication tokens.

We assume that users have limited motivation to combat phishing. Users cannot be expected to read the source of web pages, understand how DNS works well enough to look out for spoofed domains, or understand URI encoding. Users do not typically understand certificates and cannot make informed decisions about whether the subject name in a certificate corresponds to the entity they are attempting to communicate with. As a consequence of this assumption, users will likely be fooled by strings either in website names or certificates that look visually similar but that are composed of different code points.
3.1. Capabilities of Attackers

We assume attackers can convince the user to go to a website of their choosing. Since the attacker controls the web site and since the user chose to go to the website the TLS certificate will verify and the website will appear to be secure. The certificate will typically not be issued to the entity the user thinks they are communicating with, but as discussed above, the user will not notice this. Mechanisms attackers use to accomplish this include links with a misleading name or URI, which they may distribute in emails; attacks against DNS; and man-in-the-middle attacks against a TLS handshake. The former two attacks allow the attacker to pass authentication because the victim user can be tricked into accepting the attacker’s certificate. The latter attack will typically create a warning on the victim user’s side, but many users do not make informed decisions on how to respond to such a warning, making them inclined to accept the bogus certificate.

The attacker can convincingly replicate any part of the UI of the website being spoofed. The attacker can also spoof trust markers such as the security lock, URL bar and other parts of the browser UI sufficiently that a significant class of users will not treat the spoofed security indicators as a problem. There is one limitation to the attacker’s ability to replicate UI. The attacker cannot replicate a UI that depends on information the attacker does not know. For example, an attacker could generally replicate the UI of a banking site’s login page. However the attacker probably could not replicate the account summary page until the attacker learned the user name and password because the attacker would not know what accounts to list or approximate balances that will look convincing to a user. Of course attackers may know some personal information about a user. Websites that want to rely on attackers not knowing certain information need to maintain the privacy of that information.

It’s not clear how valuable this limitation on the attacker’s ability will prove in practice. Research into the effectiveness of security indicators [SECIND] suggests that users do not pay attention to security indicators. One difference between the security indicators tested in today’s research and using private information to detect fraud is that the private information may be directly related to the task the user is trying to perform. However the attacker can attempt to come up with a convincing explanation such as a partial outage or system upgrade for why the private information is not available.

The attacker can convince the user to do anything with the phishing site that they would do with the real target site. As a consequence, when passwords are used, if we want to avoid the user giving the attacker their password, the web site must prove that it has an
established authentic relationship with the user without requiring a plaintext password protocol. One approach could be to transition to a solution where the user could not give the real target site their password if they are using a new mechanism. Instead they will need to cryptographically prove that they know their password without revealing it.

### 3.2. Attacks of Interest

The ultimate goal of these requirements is to provide protection against disclosure of confidential information to unintended parties. These requirements focus on two such disclosures and handle them separately. The first category is disclosure of credentials that could allow an unintended party to impersonate the user, possibly gaining access to additional confidential information. The second attack is disclosure of confidential information not directly related to authentication. The second class of attack cannot be directly defeated, but we can give information to users that they could use to help know when they are communicating with an unintended party.

Note that some authentication systems such as Kerberos [RFC4120] provide a facility to delegate the ability to act as the user to the target of the authentication. Such a facility when used with an inappropriately trusted target would be an instance of the first class of attack. Solutions to these requirements with similar facilities MUST discuss the security considerations surrounding use of these facilities.

Of less serious concerns at the present time are attacks on data integrity where a phisher provides false or misleading information to a user.
4. Requirements for Authentication that Protects Credentials

This section describes requirements for web authentication solutions. These solutions are intended to prevent phishing targeted at obtaining web authentication credentials. These requirements will make it more difficult for phishers to target other confidential information.

4.1. Support for Passwords and Other Methods

The web authentication solution MUST support the password user interface and MUST be secure even when the password interface is commonly used. In many environments, users need the ability to walk up to a computer they have never used before and log in to a website. Carrying a smart card or USB token significantly increases the deployment cost of the website and decreases user convenience. The smart card is costly to deploy because it requires a process for replacing smart cards, requires support staff to be trained in answering questions regarding smart cards and requires a smart card to be issued when an identity is issued. Smart cards are less convenient because users cannot gain access to protected resources without having their card physically with them. Many public access computers do not have smart cards available and do not provide access to USB ports; when they do they tend not to support smart cards. It is important not to underestimate the training costs (either in money or user frustration) of teaching people used to remembering a user name and password about a new security technology. Sites that aggregate identity—for example allowing a user to log into an identity provider and then gain access to other resources may be a significant part of a solution. However we cannot assume that a given user will have only one such website: there are valid and common reasons a user (or the relying party) would not trust all identity information to one such site.

A solution to these requirements MUST also support smart cards and other authentication solutions. Some environments have security requirements that are strong enough that passwords simply are not a viable option. Many efforts are under way to reduce the deployment costs of token-based authentication mechanisms and to address some of the concerns that make passwords a requirement today.

4.2. Trusted UI

Users need the ability to trust components of the UI in order to know that the UI is being presented by a trusted component of the device. The primary concern is to make sure that the user knows any password is being given to trusted software rather than being filled into an HTML form element that will be sent to the server as part of a
plaintext password protocol.

There are many approaches to establishing a trusted UI. One example is to use a dynamic UI based on a secret shared by the user and the local UI; the paper [ANTIPHISHING] recommends this approach. The W3C recommends this approach for security indicators in section 7.1 of its user interface guidelines [WSCUIG]. However, the W3C notes that research suggests users may not pay attention to these trust indicators. A second approach is to provide a UI action that highlights trusted or non-trusted components in some way. This could work similarly to the Expose feature in Apple’s Mac OS X where a keystroke visually distinguishes structural elements of the UI. Of course such a mechanism would only be useful if users actually used it. Finally, another potential approach is to benefit from extensive research in the multi-level security community in designing UIs to display classified, compartmentalized information. It is critical that these UIs be able to label information and that these labels not be spoofable. These approaches are not exhaustive and may not even be good; they are provided to demonstrate that thought into how to design trusted UIs is ongoing. However, designing a user interface that allows users of the web to distinguish trusted components from components potentially controlled by an attacker is an open problem. It is likely that transitioning to many new security protocols will depend on a solution to this problem.

4.3. No Password Equivalents

A critical requirement is that when a user authenticates to a website, the website MUST NOT receive a strong password equivalent [IABAUTH]. A strong password equivalent is anything that would allow a phisher to authenticate as a user with a different website. Consequently, plaintext password protocols are incompatible with these requirements. Weak password equivalents (quantities that act as a password for a given service but cannot be reused with other services) are problematic outside of the context of enrolling a user or changing a password. The requirement for mutual authentication Section 4.4 is incompatible with sending weak password equivalents in every authentication. Even if that requirement is relaxed, the scope of a particular weak password equivalent needs to be carefully considered. Consider for example a protocol that hashes a password and the host name component of a URI together to form a weak password equivalent. The same password equivalent is used regardless of which certificate authority certifies the public key of the website. If an attacker mounted a man-in-the-middle attack, presenting a self-signed certificate, and the user accepted the certificate when asked by the browser, then the attacker would receive the same weak password equivalent needed to access the legitimate website. Such a protocol would not do a good job of addressing the threats outlined in the
threat model. However if mutual authentication were not a requirement, a protocol that hashed a password and the public key from the TLS certificate of the website to form a weak password equivalent might meet the other requirements. In any event, weak password equivalents MUST NOT be sent without confidentiality protection.

There are two implications of this requirement. First, a strong cryptographic authentication protocol needs to be used instead of sending the password encrypted over TLS. The zero-knowledge class of password protocols such as those discussed in section 8 of the IAB authentication mechanisms document [IABAUTH] seem potentially useful in this case at a first glance. However, mechanisms in this space tend to have significant deployment problems because of intellectual property issues.

The second implication of this requirement is that if an authentication token is presented to a website, the website MUST NOT be able to modify the token to authenticate as the user to a third party. The party generating the token must bind it to either the website that will receive the token or to a key known only to the user. Binding could include cryptographic binding or mechanisms such as issuing a one-time password for use with a specific website. If tokens are bound to keys, the user MUST prove knowledge of this key as part of the authentication process. The key MUST NOT be disclosed to the server unless the token is bound to the server and the key is only used with that token or server.

4.4. Mutual Authentication

[ANTIPHISHING] describes a requirement for mutual authentication. A common phishing practice is to accept a user name and password as part of an attempt to make the phishing site authentic. The real target is some other confidential information. The user name and password are captured, but are not verified. After the user name and password are entered, the phishing site collects other confidential information. When mutual authentication fails, there is a strong indication of a problem: either the user supplied the wrong credential or the website is not the one the user intended to communicate with.

Requiring mutual authentication excludes a class of mechanisms where a weak password equivalent is generated for the server and is sent. One prominent member of this class is [PWDHASH]; this mechanism has the desirable property that it requires no change to the server and can be implemented locally on the browser. These mechanisms provide better security than plaintext password protocols. However attacks where the server ignores authentication in order to obtain
confidential information are important enough that it is desirable to
develop mechanisms that provide this assurance. The desire to
develop these new mechanisms is not intended to discourage the
deployment of mechanisms like Pwdhash that improve security today.

Typically one protocol performs authentication of both parties.
There tend to be opportunities for a man-in-the-middle attack when
one protocol authenticates one direction and another protocol
authenticates the opposite direction. Sometimes, as in the case of
TLS and plaintext password protocols, the opportunity for attacks
depends on human factors issues or certificate management. In other
cases, attacks may be more direct. Authentication of the server and
client at the TLS level is sufficient to meet the requirement of
mutual authentication. If authentication is based on a shared secret
such as a password, then the authentication protocol MUST prove that
the secret or a suitable verifier is known by both parties.
Interestingly the existence of a shared secret will provide better
confidence that the right server is being contacted than if public
key credentials are used in their typical mode. By their nature,
public key credentials allow parties to be contacted without a prior
security association. In protecting against phishing targeted at
obtaining other confidential information, this may prove a liability.
However public key credentials provide strong protection against
phishing targeted at obtaining authentication credentials because
they are not vulnerable to dictionary attacks. Such dictionary
attacks are a significant weakness of shared secrets such as
passwords intended to be remembered by humans. For public key
protocols, the mutual authentication requirement would mean that the
server typically needs to sign an assertion of what identity it
authenticated or of the request as a whole.

4.5. Authentication Tied to Request and Response

Users expect that whatever party they authenticate to will be the
party that generates the content they see. One possible phishing
attack is to insert the phisher between the user and the real site as
a man-in-the-middle. On today’s websites, the phisher typically
gains the user’s user name and password. Even if the other
requirements of this specification are met, the phisher could gain
access to the user’s session on the target site. This attack is of
particular concern to the banking industry. A man-in-the-middle may
gain access to the session which may give the phisher confidential
information or the ability to execute transactions on the user’s
behalf.

The authentication system MUST guarantee to the user and the target
server that the request was generated by the authenticated user and
the response is generated by the target server. This can be done in
several ways including:

1. Assuming that only certificates from trusted CAs are accepted and the user has not bypassed server certificate validation, it is sufficient to confirm that the identity of the server at the TLS level is the same at the HTTP authentication level. In the case of TLS client authentication this is trivially true. Note however that [WSCUIG] recommends accepting self-signed certificates in some cases, so relying on this approach for cases other than TLS authentication may be problematic.

2. Another alternative is to bind the authentication exchange to the channel created by the TLS session. The general concept behind channel binding is discussed in [RFC5056]. Channel binding has been added to HTTP authentication mechanisms based on digest authentication and on GSS-API, suggesting that support for channel binding is workable for future HTTP authentication mechanisms.

4.6. Restricted Identity Providers

Some identity providers will allow anyone to accept their identity. However particularly for financial institutions and large service providers it will be common that only authorized business partners will be able to accept the identity. The confirmation that the relying party is such a business partner will often be a significant part of the value provided by the identity provider, so it is important that the protocol enable this. For such identities, the user MUST be assured that the target server is authorized by the identity provider to accept identities from that identity provider. Several mechanisms could be used to accomplish this:

1. The target server can provide a certificate issued by the identity provider as part of the authentication.

2. The identity provider can explicitly approve the target server. For example in a redirect-based scheme the identity provider knows the identity of the relying party before providing claims of identity to that party. A similar situation happens with Kerberos or Digest Authentication in a AAA infrastructure [RFC5090].

4.7. Protecting Enrollment

One area of particular vulnerability to phishing is enrollment of a new identity in an authentication system. Protecting against phishing targeted at obtaining other confidential information as a new service is established is outside the scope of this document. If
confidential information such as credit card numbers are provided as part of account setup, then this may be a target for phishing.

However there is one critical aspect in which enrollment impacts the security of authentication. During enrollment, a password is typically established for an account or other security credentials are associated with an account. The process of establishing a password MUST NOT provide a strong password equivalent (a quantity such as the password itself that could be used to log into another service where the same password is used as the user). That is, parties other than the user and web browser MUST NOT gain enough information to impersonate the user to a third party while establishing a password.
5. Is it the right Server?

In Section 4, requirements were presented for web authentication solutions to minimize the risk of phishing targeted at web access information. This section discusses in a non-normative manner various mechanisms for determining that the right server has been contacted. Authenticating to the right party is an important part of reducing the risk of phishing targeted at other confidential information.

Validation of the certificates used in TLS and verification that the name in the URI maps to these certificates can be useful. As discussed in Section 3, attackers can spoof the name in the URI. However the TLS checks do defeat some attacks. The W3C user interface guidelines may significantly increase the value of these checks [WSCUIG]. As discussed in Section 4.5, TLS validation may be important to higher-level checks.

A variety of initiatives propose to assign trust to servers. This includes proposals to allow users to indicate certain servers are trusted based on information they enter. Also, proposals to allow third parties including parties established for this purpose and existing certificate authorities to indicate trust have been made. These proposals will almost certainly make phishing more difficult.

In the case where there is an existing relationship, these requirements provide a way that information about the relationship can be used to provide assurance that the right party has been contacted.

In Section 4.2, we discuss how a secret between the user and their local computer can be used to let the user know when a password will be handled securely. A similar mechanism can be used to help the user once they are authenticated to the website. The website can present information based on a secret shared between the user and website to convince the user that they have authenticated to the correct site. This depends critically on the requirements of Section 4.5 to guarantee that the phisher cannot obtain the secret.

Various schemes have used a secret shared between the server and the web browser before authentication. Cookies or some other state management mechanism are used to select the right secret to display as the user logs into the site. Unfortunately these schemes have proven ineffective in practice [SECIND]. However, the set of information that can be used as contextual clues to evaluate whether the right server has been reached after authentication is much greater. For example, a bank server knows what accounts a user has and knows their balances. A business partner may have information
about past transactions or the current state of transactions. If this information is related to the task that the user is trying to perform, they may be more likely to evaluate it and notice problems than they are to notice a missing security indicator before login. Strong authentication mechanisms enable this type of evaluation after the user has logged in. However it is not known how effective this will be in practice.
6. Iana Considerations

This document requests no action of IANA.
7. Acknowledgments

I’d like to thank the MIT Kerberos Consortium for its funding of work on this memo prior to April 2008.

I’d like to thank Nicolas Williams, Matt Knopp and David Blumenthal for helping me walk through these requirements and make sure that if a solution met them it would actually protect against the real world attacks consumers of our technology are facing. I was particularly focusing on attacks that financial institutions are seeing and their help with this was greatly appreciated.

I’d like to thank Eric Rescorla and Ben Laurie for their significant comments on this draft.

Eliot Lear provided many last call comments and helped work through several long standing issues with the document.

Christian Vogt provided text and review comments.

The requirements discussed here are similar to the principles outlined in "Limits to Anti-Phishing" [ANTIPHISHING]. Most of this work was discovered independently but work from that paper has been integrated where appropriate. It seems good that these requirements are similar to the principles outlined by someone facing phishing as an operational reality.
8. Security Considerations

This memo discusses the security of web authentication and how to minimize the risk of phishing in web authentication systems. This section discusses the security of the overall system and discusses how components of the system that are not directly within the scope of this document affect the security of web transactions. This section also discusses residual risks that remain even when the requirements proposed here are implemented.

The approach taken here is to separate the problem of phishing into phishing targeted at web authentication credentials and phishing targeted at other information. Users are given some trusted mechanism to determine whether they are typing their password into a secure browser component that will authenticate them to the web server—a component that presents a password interface—or whether they are typing their password into a legacy mechanism that will send their password to the server as part of a plaintext password protocol. If the user types a password into the trusted browser component, they have strong assurances that their password has not been disclosed and that the page returned from the web server was generated by a party that either knows their password or who is authenticated by an identity provider who knows their password. The web server can then use confidential information known to the user and web server to enhance the user’s trust in its identity beyond what is available given the social engineering attacks against TLS server authentication. If a user enters their password into the wrong server but discovers this before they give that server any other confidential information, then there exposure is very limited.

This model assumes that the parts of the browser and operating system with access to passwords or other long-term credentials are trusted software. As discussed in Section 3, there are numerous attacks against host security. Appropriate steps should be taken to minimize these risks. If the security of the trusted software is compromised, the password can be captured as it is typed by the user.

This model assumes that users will only enter their passwords into trusted browser components. There are several potential problems with this assumption. First, users need to understand the UI distinction and know what it looks like when they are typing into a trusted component and what a legacy HTML form looks like. It is not clear that we have yet developed a solution to this user interface problem. Users must care enough about the security of their passwords to only type them into trusted components. The browser must be designed in such a way that the server cannot create a UI component that appears to be a trusted component but is actually a legacy HTML form; the W3C user interface guidelines [WSCUIG] provides
requirements that are designed to prevent security sensitive user interface from being spoofed by attacker-supplied content. The W3C guidelines provide requirements for a more limited context focused around security context but not authentication information. However starting from these requirements may be a successful approach.

In addition, a significant risk that users will type their password into legacy HTML forms comes from the incremental deployment of any web authentication technology. Websites will need a way to work with older web browsers that do not yet support mechanisms that meet these requirements. Not all websites will immediately adopt these mechanisms. Users will sometimes browse from computers that have mechanisms meeting these requirements and sometimes from older browsers. They only gain protection from phishing when they type passwords into trusted components. If the same password is sometimes used with websites that meet these requirements and sometimes with legacy websites, and if the password is captured by a phisher targeting a legacy website, then that captured password can be used even on websites meeting these requirements. Similarly, if a user is tricked into using HTML forms when they should not, passwords can be exposed. Users can significantly reduce this risk by using different passwords for websites that use trusted browser authentication than for those that still use HTML forms.

The risk of dictionary attack is always a significant concern for password systems. Users can (but typically do not) minimize this risk by choosing long, hard to guess phrases for passwords. The risk of offline dictionary attack can be removed once a password is already established by using a zero-knowledge password protocol. The risk of online dictionary attack is always present. The risk of offline dictionary attack is always present when setting up a new password or changing a password. Minimizing the number of services that use the same password can minimize this risk. When zero-knowledge password protocols are used, being extra careful to make sure the right server is used when establishing a password can significantly reduce this risk.
9. References

9.1. Normative References


Publication of this draft needs to block until and unless this references is approved as some form of W3C recommendation.

9.2. Informative References


Appendix A. Trusted UI Mechanisms

There are three basic approaches to establishing a trusted UI. The first is to use a dynamic UI based on a secret known by the user; [ANTIPHISHING] recommends this approach. A second approach is to provide a UI action that highlights trusted or non-trusted components in some way. This could work similarly to the Expose feature in Apple’s OS X where a keystroke visually distinguishes structural elements of the UI. Of course such a mechanism would only be useful if users actually used it. Finally, the multi-level security community has extensive research in designing UIs to display classified, compartmentalized information. It is critical that these UIs be able to label information and that these labels not be spoofable.

See Section 5 for another case where confidential information in a UI can be used to build trust.
Appendix B. Change History

Note to rfc editor: This section should be removed prior to publication.

B.1. Changes Since 08

Propose a new purpose section. Also, add a note describing what has been done to date on these issues.

B.2. Changes since 07

Reword the abstract not to talk about identity providers

Define identity provider. I’m moving away from using it except where necessary, but I think that there a couple of cases where the term is helpful rather than confusing.

Add a paragraph to the introduction helping to define how this work fits in with other work.

Significantly rework the mutual authentication requirement to describe why pwdhash is excluded, to give more motivation and to try and clarify that authentication at different layers is problematic.

Rework the requirement for binding authentication to requests and responses. The discussion of channel binding was obsolete and has been updated based on advances in that area. Drop the comment about redirect based schemes, because that depends on certificate validation and the W3C guidelines recommend accepting self-signed certificates in some cases.

Remove most references to identity providers from restricted identities section and protecting enrollment section. The concepts don’t actually depend on whether an identity provider is used.

Rework the section on finding the right server to provide a more accurate description of image hints prior to login and to discuss the uncertainty surrounding the effectiveness of strategies discussed.

Rephrase terminology in security considerations to be consistent with changes throughout the rest of the document. Refer to the W3C guidelines as appropriate.
B.3. Changes since 06

Much expanded description of concerns about weak password equivalents. They are not excluded except by the mutual authentication requirement. However there are significant scoping issues with them.

Clarify that the effectiveness of confidential information being used to strengthen mutual authentication depends on users taking advantage of that.

Continue to clarify differences between plaintext password protocols and the password user interface.

Reduce the use of the term identity provider; it’s not entirely clear that concept needs to be worked in here and right now identity provider is an undefined term.

The text on how to make trusted UIs sounded very authoritative; that was not the intent, so rework that text.

B.4. Changes since 05

Clarified introduction to distinguish what happens at the TLS layer and what at the HTTP layer. Discuss motivation of phishing more.

In the introduction, restate claims to be more accurate. These requirements are useful if users actually use the authentication mechanisms; convincing them to do so and making it obvious whether they are is a significant risk. Also, we may give them the theoretical information necessary to detect fraud, but whether they act on that is open.

Add a purpose of this memo section. Whatever text ends up there after community discussion needs to be called out in the last call.

Add a section calling out the difference between plaintext password protocols and password interface. This needs to be worked into the rest of the document.

Update the threat model. Significant hopefully clarifying changes.
B.5. Changes since 02

Updated discussion of TLS authentication to point out that it does meet the requirement of mutual authentication.

Added pointer to HTTP TLS channel bindings work

B.6. Changes since 01

Updated threat model to give examples of attacks that are in scope and to more clearly discuss host security based on comments from Chris Drake.

Clarify attacks of interest to be consistent with the introduction.

Fix ups regarding one-time passwords. I’m not sure that OTPs can meet all the requirements but clean things up where they clearly can meet a requirement.

Clarify that in the mutual authentication case I’m concerned about authentication of client to the server.

Clean up bugs in security considerations
Author’s Address

Sam Hartman
Painless Security, LLC

Email: hartmans-ietf@mit.edu
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.