PAKE-based mutual HTTP authentication for preventing phishing attacks (Extended Abstract)

Yutaka Oiwa Hiromitsu Takagi Hajime Watanabe Hideki Imai
Research Center for Information Security (RCIS)
National Institute of Advanced Industrial Science and Technology (AIST), Japan
y.oiwa@aist.go.jp

1. INTRODUCTION

We developed a new password-based mutual authentication protocol for Web systems which prevents various kinds of phishing attacks. This work is a part of the joint research project between "Yahoo! Japan" and RCIS, AIST. We have finished designing the protocol specification, and have implemented extension modules for Apache and Mozilla Firefox for this protocol.

Recently, phishing attacks are getting more and more sophisticated. Phishers not only steal user's password directly, but imitate successful authentication to steal user's sensitive information, check the password validity by forwarding the password to the legitimate server, or employ a man-in-the-middle attack to hijack user's login session. Existing countermeasures such as one-time passwords can not completely solve these problems.

Our protocol prevents such attacks by providing users a way to discriminate between true and fake web servers using their own passwords. Even when a user inputs his/her password to a fake website, using this authentication method, any information about the password does not leak to the phisher, and the user certainly notices that the mutual authentication has failed. Phishers cannot make such authentication attempt succeed, even if they forward received data from a user to the legitimate server or vice versa. Users can safely input sensitive data to the web forms after confirming that the mutual authentication has succeeded.

To achieve this goal, we use a mechanism in ISO/IEC 11770-4, a kind of PAKE (Password-Authenticated Key Exchange) authentication algorithms as a basis. The use of PAKE mechanism allows users to use familiar ID/password based accesses, without fear of leaking any password information to the communication peer. The protocol, as a whole, is designed as a natural extension to the current HTTP authentication schema such as Basic and Digest access authentication (RFC 2617). To use PAKE mechanism for such a purpose, we had to modify it to prevent credential forwarding (man-in-the-middle) attacks.

We also invented new user-interface for this authentication system. To prevent phishing attacks, it is important to make users easily determine whether the server authentication has been succeeded or not. This information must be protected from forgery by phishers, otherwise phishers deliberately convince users that the mutual authentication is established, and let users input sensitive information to the phishing sites.

2. RELATED WORK

There are several existing proposals which can be used for preventing phishing attacks. TLS-SRP extension introduces a kind of PAKE into TLS. Although it can be a good solution for closed applications like VPN or IPP, it is not convenient for general web systems.

Several web toolbar plugins have own login facility to the specific sites, which effectively provides host-authentication. However, it is unacceptable to install a number of plugins for all possible web sites the users access.

TLS server authentication is not powerful enough to prevent phishing, as certificates can be acquired by any party including phishers. Phishing using HTTPS certified by publicly-accepted PKI is already a real issue.

Another solution is to issue a special certificates for each "genuine" site (as accepted by the vendors of anti-phishing products). However, it is completely "closed" solution with unclear criteria for "genuineness".

"pwd_hash" prevents phishing by making the transmitted data different for each sites, even for the same password. However, to prevent phishers from guessing the real password, the original password must be very long (e.g. more than 30 characters) to prevent off-line attacks.

3. DESIGN PRINCIPLES

Our design of the protocol addresses the following criteria for preventing phishers from forging users and stealing private information including passwords and other data.

- The protocol is generic: As the algorithm uses passwords as a basis of authentication, a single implementation of the algorithm can be used for any web sites without specific authorizations for "genuine sites".
- The protocol is a natural extension to existing HTTP authentication algorithms defined in RFC 2617. It can be easily integrated to web servers and clients, and the protocol can pass through existing web proxies, load balancers, or TLS accelerators without modification to such intermediate machines.

  We also rely on existing TLS (HTTPS) mechanisms for confidentiality and transport-layer safety. We assume that the users may reach phisher's site with wrong host-name, but the communication to the genuine server with the correct host-name is safe.
- The authentication is sound: if users have connected to a phishing site with wrong host-name (which do not know the user's password), the authentication will never succeed, and the users can reliably be aware to

the authentication failure. This property must hold even if the genuine site is available as an oracle (i.e. on-line attack must be impossible).

- Further more, in such cases no information about the user's password is leaked to the phishers. By analyzing the communication data, even if they performed exhaustive search (off-line attack), no information about the password can be acquired. This means it is *safe* to input their password to the phisher's site.

4. PROTOCOL OVERVIEW

Brief view of our protocol is described in this section. Our protocol uses the "Key Agreement Mechanism 3" (KAM3, Section 6.3) defined in ISO/IEC 11770-4 as a cryptographic basis. Server-side password database, which contains pairs of a user-name and the password verification element v calculated from user's password ("π" below), is prepared beforehand.

Firstly, as a response to a client's request (without any authentication), the server sends a usual HTTP 401 response to request authentication.

```
GET / HTTP/1.1
Host: www.example.com

HTTP/1.1 401 Authentication required
WWW-Authenticate: Mutual algorithm=iso11770-4-ec-p256,
        validation=host, realm="Protected Contents",
        stale=0
Content-Type: text/html; charset="ISO-8859-1"
Content-Length: xxx
```

After the user has input the user-name and the password, the client calculates the password hash π by combining host-name, realm, user-name and the password. the use of the host-name here prevents phishers from exploit credential forwarding attacks.

The client also constructs a value w_a according to ISO/IEO 11770-4, and sends the second request. Then the server respond with an intermediate 401 response containing the value w_b using the password verification element v in the password database. It also sends the session id "sid" which is used to distinguish concurrent authentication sessions.

```
GET / HTTP/1.1
Host: www.example.com
Authorization: Mutual algorithm=iso11770-4-ec-p256,
      validation=host, user=foobar, wa=xxxx

HTTP/1.1 401 Authentication required
WWW-Authenticate: Mutual sid=yyyy, wb=zzzz,
      nc-max=256, nc-window=64, time=300, path=/
Content-Length: 0
```

After that, the client sends a third request containing value of o_a. The value o_a is a hash value calculated according to ISO/IEO 11770-4 except that it also uses the host-name, the value of nonce counter (nc) to calculate it. The server calculates the same value and verifies whether the client is acceptable. If so, the server sends the final response with the value o_b. The receiving client *must* verify the value o_b to check whether the server is genuine.

```
GET / HTTP/1.1
Host: www.example.com
Authorization: Mutual sid=yyyy, nc=0, oa=wwww

HTTP/1.1 200 OK
Authentication-Info: Mutual sid=yyyy, ob=vvvv
Content-Type: text/html; charset="ISO-8859-1"
Content-Length: xxx
```

If second request to the same host is sent, the client can reuse the session key by directly sending the third message to the server reusing the session ID. In this case, we only need to perform a hash operation but no public-key operations.

5. SECURITY ANALYSIS

The possible attack stories for phishers under assumptions in Section 2 is (1) to retrieve a password from received data w_a and o_a, (2) to imitate that the authentication has been succeeded, or (3) to forward received data to the genuine host to perform authentication. However, by the design of the KAM3, it is information-theoretically impossible to gain any information about the password π from w_a and o_a, thus the attack (1) is impossible even if the dictionary of possible passwords are used for off-line attack.

For attack (2), the client-side checking of o_b in the final message is important. Unless phishers knows the password hash π (or the verifier v), they cannot construct the correct value of o_b.

Finally, for the case of attack (3), The value π used for the the phisher's site is different from the correct value for the genuine site, because it encodes the peer host-name. Thus the phisher cannot use the received w_a and o_b to make mutual authentication succeed with the genuine host.[1]

6. USER INTERFACES CONSIDERATION

Another possibility for an attack is to forge the UI dialogs for asking passwords using this protocol. To prevent such kind of attacks, our extension for Mozilla Firefox uses the address-bar area (where web-pages do not have access to) for password input instead of using dialogs, and introduces a indicator for displaying the authentication status.

7. CONCLUSION AND FUTURE WORK

We have designed a new password-based Web mutual authentication protocol which prevents various kinds of phishing attacks. We are now improving the implementations of the browser extension and the web server module. These implementations will be available as open-source software. we are planning to perform demonstration on a part of "Yahoo! Auction" website in Japan in near future.

[1]In usual setting of PAKE protocol, the client and the server agrees on common secret key during negotiation, and by using this key with encryption, the forwarding (man-in-the-middle) attack is prevented. However, because our protocol only uses PAKE as a authentication layer, we need an alteration here to prevent forwarding attacks in a different way.