

Position Paper:

Enhancing browsers & servers with Anti-Spoof data elements

Or, thinking outside the box

Background

Users are never going to be safe as long as they are required to manually enter sensitive information into login forms. The current trend of protecting users from attacks is by the use of browser toolbars or enhanced browsers which try to detect a fraudulent website when users navigate to such a site.

While these methods are helpful, they still leave the door open for attacks. Currently, most Phishing sites use 'static cloning' of real sites. However, there is nothing to prevent Phishers from creating dynamically cloned sites. Such sites implement a real-time man-in-the-middle (MITM) attack vector against web surfers and servers.

Even when servers use a secure SSL protocol to communicate with users, they still leave the door open for several MITM modes of attack. That is, unless they enforce a non anonymous mutual authentication using digital certificates.

When a server requires a secure session from a client, a Phisher can start such a secure session with the server on one end and start another session with a user on the other end. All a Phisher has to do is decrypt whatever messages are sent to it and pass them along to the other end. In such a scenario, a user will not even notice that her password has been compromised.

Using One Time Passwords does not help. Intercepting OTP by a Phisher is not as rewarding as when intercepting a regular password but when a real-time MITM is active, there is nothing to prevent an attacker from adding bots which would take a captured OTP and use it to immediately effect a transaction. Even if such attempts are thwarted by a server requiring constant manual interaction with a human being (which is unlikely), the door is still open for targeted Phishing whereby a Phisher manually intervenes once a session is created.

Enhancing Transport Layer Security (SSL/TLS)

A basic solution is to enhance the TLS layer connecting servers and clients to the level possible when employing digital certificates for users, but without requiring such certificates. A possible mechanism is that of "Zero Knowledge Password Proof".

Such mechanisms are based on methods whereby two parties prove to each other that each one is in possession of a shared "weak" secret such as a password are well known and some are proposed as standards. IEEE 1363.2, SRP-6 (RFC 2945), SPEKE etc. Using these techniques as an authentication protocol for TLS is also described by <http://www.ietf.org/internet-drafts/draft-ietf-tls-srp-10.txt> and http://www.semper.org/sirene/publ/SBEW_01EKETLS.pdf.

Implementing a new TLS protocol requires infrastructure upgrades for both client work stations and servers. While this is a worthwhile venue to explore, there is an immediate need for a less demanding solution that can achieve a similar level of security.

Enhancing HTTP Layer Security

An alternative to enhancing the TLS layer is one of enhancing the HTTP layer security. This proposed new method is simple to implement, both on a client side as well as a server side. It

does not require any modifications to existing infrastructure, nor any changes to databases used by servers to store users' credentials.

Actually the name "HTTP layer security", is somewhat misleading. While the proposed method can be implemented as an extension to HTTP, it can also be implemented by existing HTTP where HTML forms are used to communicate security parameters between client and server. This second method is described in this paper.

The proposed method is based on a modified SRP-6 a "Zero Knowledge Password Proof" method (**ZKPP**). In addition, the proposed method relies on a new Anti-Spoof data element exchanged between a client and a server. The Anti-Spoof data element is what protects this method from session hijacking.

The following is one description of a process for authenticating a client to server, but it can be easily extended to mutual authentication.

Although this paper does not attempt to cover all SRP methods, it will be helpful for understanding the proposed method, if the SRP parameters are explained. Please refer to the cited references for proper background.

N and $q = (N-1)/2$ are both prime (N is a safe prime and q is a Sophie Germain prime). All arithmetic is performed modulo N .

G is a generator of the multiplicative group modulo N ,

U is a username,

x is a value derived from user's password and stored at Server's database. It could be the password itself.

$H()$ is a [hash](#) function, e.g. SHA-1

a and b are random.

and $|$ denotes concatenation, $*$ denotes multiplication and $^$ denotes exponentiation.

In the proposed method (Please see Fig 1 below), a server sends a first login form to a user. The login form is an HTML form containing a user name field and other fields that provide parameters to a client software (a browser toolbar or an enhanced browser) for executing ZKPP functions.

The client software detects the special login form. It saves the ZKPP parameters, then it computes $A=G^a$. It enters A into a hidden field provided by the login form and lets a user enter her user name U and submit the form.

When a server receives U and A in the first login form it computes $B=G^b + 3*G^x$. Then the server sends B as a hidden field in a second login form which also prompts for a user's password.

After a user enters her password to the second login form and hits 'submit', the client software intervenes and removes the password from that login form. (Alternatively, a client software could popup a special login dialog and take the password from there).

It then computes x from the captured password. Computing x depends on the method that x was calculated by the server. Please recall that x is the value stored in the server's database. To be able to calculate x , client software needs to learn from a server what function and what parameters were used for such a calculation. This information can be conveyed to the client software in one of the login forms as a set of hidden fields.

After computing x , the client software computes the following:

$u = H(A,B)$

$S = (B-3*G^x)^{(a+u*x)}$.

$C =$ IP address of Server.

$M1=H(A,B,S,C)$

C is the Anti-Spoof element introduced earlier. It holds a string which is the IP address of the server as resolved by the client software.

After computing the values above, the client software enters them into the second login form. If the form had a password field, then M1 replaces that value. Otherwise, it can be entered into a predestinated hidden field. The value C is also entered into a hidden field in that form.

Upon receiving the second login form, the server verifies that C matches at least one IP address used by the server. It then computes the following:

$$u = H(A, B)$$

$$S' = (A * (G^x)^u)^b$$

$$C' = C$$

$$M1' = H(A, B, S', C')$$

Now, if $M1 = M1'$ we have authentication. In addition we also have a proof that the IP address to which the client is connected is that of the server and not a MITM attacker. Thus, server can safely continue the session it has with the client software knowing that there is no MITM involved.

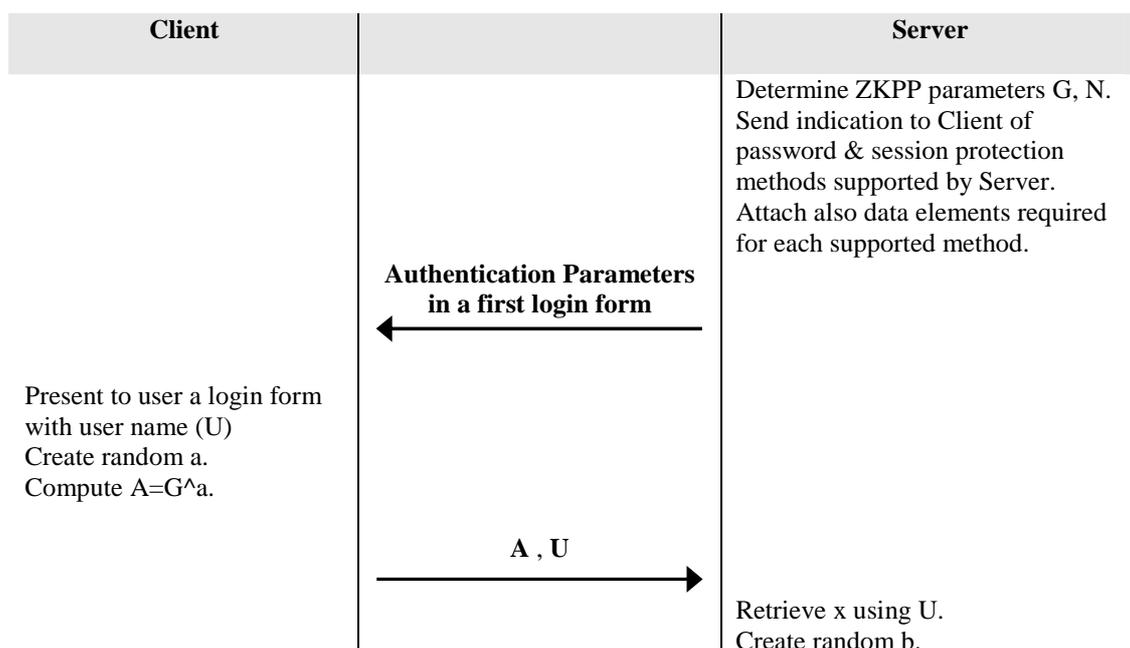
Warning users

All this is nice but not complete as an attacker could simply send a standard login form to a user and prompt her for a password and then use the password to initiate the above protocol with a server.

That brings us back to square one, or, does it?

If users choose to install this enhanced browser or toolbar, they will know that when they enter information to a real site, it will be fully secure. On the other hand, when the client software detects a login form with a password field it will now alert them to that event raising awareness of users to a possible Phishing attack.

When a login page is not compliant with the proposed protocol, client software can invoke legacy protection or, simply alert users that a “non-safe” server asks for authentication. As more and more websites and users migrate to the new method, such incidents will decline in frequency.



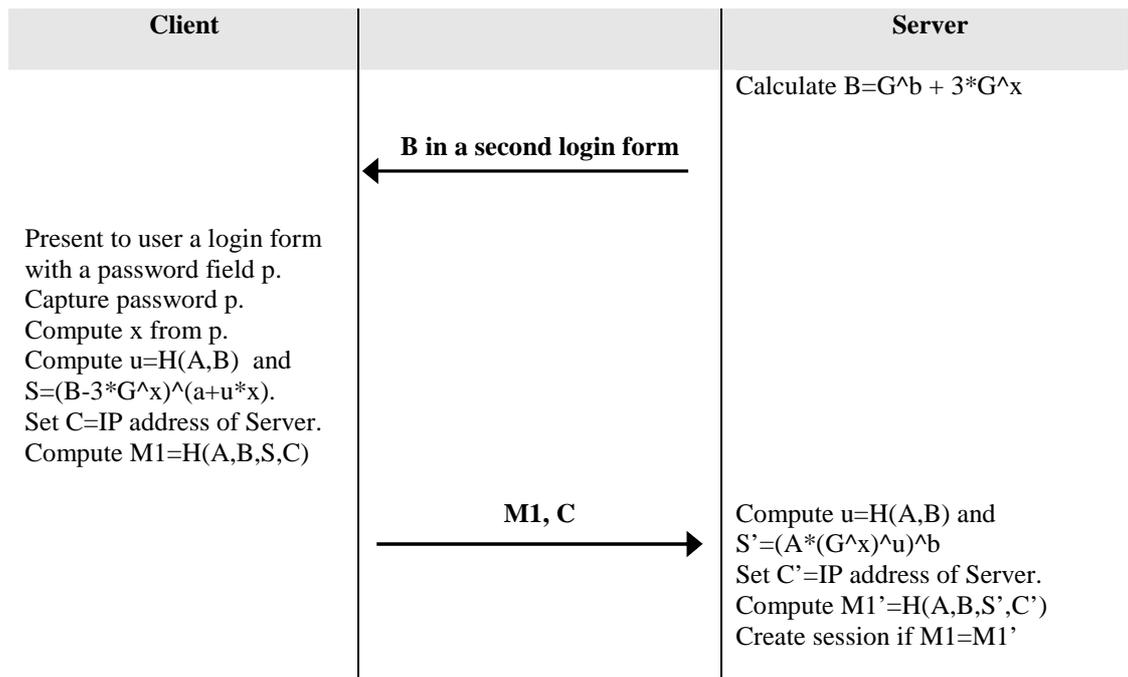


Fig 1.

Summary

The proposed method is incremental in nature. It does not require major modification on either side – client or server. It can leverage the existing database use by servers with no change.

Author: Ami Grynberg
 Protecteer, LLC
 15 Constitution Drive
 Bedford, NH 03110 USA
 Email: www1.7426@ami.susw.com
 Tel: (603) 589-6539