Web authentication systems

Martin Wåger, Josefin Zetterlund, Seung Joo Ha

Table of Contents

Introduction.... 1
Basic HTTP access authentication.... 1
Digest HTTP access authentication.... 2
HTTPS.... 2
The benefits of SSL.... 2
How SSL works.... 2
HTTP over SSL.... 3
HTTP 1.1 Upgrade Header.... 3
E-legitimation or electronic ID.... 4
Where is the e-ID kept?.... 4
Obtaining the certificate.... 4
Cryptographic Algorithm.... 4
Safety.... 5
Summary.... 5
Sources.... 5

Introduction

As more and more people use internet/web, the number of security sensitive data over internet is growing more and more and so does the need of better authentication systems. Unlike the earlier period of web when there were not many nodes connected together and those nodes were trustworthy, now a great deal of systems are connected together and much of the nodes are untrusted. Someone might disguise oneself as a server or client to acquire sensitive data. Therefore both parties participating in the communication (server and client) has to authenticate itself to each other.

There are many different authentication schemes used on the internet. Here we will cover the most common ones. We'll briefly check two HTTP authentication schemes and after that SSL/TLS for secure connection and finally look at how E-IDs can be used to authenticate a user to a system.

Basic HTTP access authentication

The Basic authentication scheme is very simple. It is designed to run in a safe environment and features no encryption. It will, upon receiving a 401 Unauthorized message from the server, send the password base64 encoded back to the server. It is not hard to obtain the password from the encoded string.

Digest HTTP access authentication

Digest authentication scheme scheme adds a level of security by the use of nounces (unique numbers that are sent only once) and salted hashing thus keeping the password safe. However both Basic and Digest schemes suffer from the drawback that the server must know the password in advance. This limits its use in an environment like the internet where you often do not know your users in advance.

HTTPS

HTTPS is currently the most common authentication method on the web. It is basically HTTP over a secure channel. A URL that starts with HTTPS: will result in a HTTP connection being established but unlike normal HTTP the port used is 443 and an additional layer is added between the application and the transportation, the Secure Session Layer (SSL). SSL was designed by Netscape and the 1999 version 3.0 was the basis for its successor the Transport Layer Security (TLS). There are slight differences between SSL and TLS but as they are so similar they are often referred to as SSL/TLS. Here after we will use SSL to mean both SSL and TLS.

The benefits of SSL

SSL gives the users three things

1. Server authentication. A SSL enabled browser has a list of trusted Certification Authorities (CA) and the CA public keys. When a client opens a HTTPS connection the server sends a certificate containing the servers public key and it's name signed by the trusted CA. The client can then verify the servers authenticity.
2. Client authentication. As with the server authentication the client have an option to send it's certificate to the server. This however requires the client to set up and use some public key infrastructure, something that most users do not have. SSL is thus most commonly used with only server side authentication.
3. Encryption. SSL provides the encryption of the information sent between the client and the server.

How SSL works

As mentioned above a HTTPS session starts when the user requests a page where the URL starts with https://. This starts the SSL handshake protocol, this protocol sets up the cryptographic preferences. The handshaking works in the following steps (somewhat simplified):

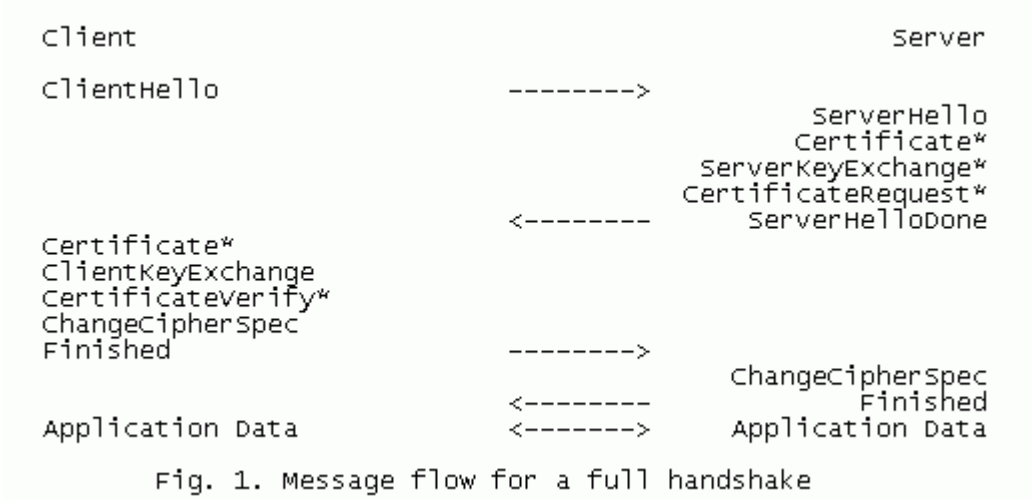

Fig. 1. Message flow for a full handshake

1. The client sends a ”ClientHello” message to the server containing a random key, a ordered list of preferred ciphers and compression methods. It also includes the version number of the SSL protocol.
2. The server then selects a cipher to use from the list and sends back a ”ServerHello” message to the client. Included in this message is the servers certificate (which includes it's public key) and another random key. The server also have the possibility to request a certificate from the client at this stage.
3. The client verifies the server certificate against it's own CA list. If the servers CA is not listed the user is normally notified and given a choice to either disconnect or allow the CA given. After this the client generates a symmetric session key based on the two random keys. The client encrypts the new key with the servers public key and sends it to the server together with a message to switch to encryption based on the new key and a message indicating that the client handshake is done.
4. The server decrypts session key and sends a message saying that the server handshake is finished and that future messages will be encrypted with the new symmetric key.

After this both parties share session keys (in practise the above scheme generates two different session keys one for server to client and one for client to server messages) that can be used to create a safe information exchange.

HTTP over SSL

Now that SSL have created a secure connection with a server that have been authenticated against a known (or accepted) CA the HTTP session can continue. Both sides are now certain that no outsider can read change or fake the communication between them. At this stage the client can use , for instance, the basic HTTP authentication scheme to authenticate the user to the remote system (if the client did not send a certificate of it's own). However both sides should be aware that the safety of SSL is dependant of the negotiated cipher. There are ways for attackers to force the handshaking to use the weakest form of encryption.

HTTP 1.1 Upgrade Header

There is another, newer way of securing a HTTP connection, the HTTP 1.1 Upgrade header. The HTTPS URI scheme has a drawback which is that it needs separate ports for secure and unsecure use of application. This flaw lead to the HTTP 1.1Upgrade header scheme . However browsers that supports the Upgrade header is not widely spread so the https URI scheme is still the dominant method of establishing a secure HTTP connection.

In a upgrade header scheme, the client begins by making a clear-text request, which is later upgraded to TLS. Either the client or the server may request (or demand) that the connection be upgraded. The most common usage is a clear-text request by the client followed by a server demand to upgrade the connection.

When the client is requesting for the upgrade there are optional upgrade and mandatory upgrade. In optional upgrade, the client offers to switch to secured operation during any clear HTTP request when an unsecured response would be acceptable. On the other hand, if an unsecured response would be unacceptable, a client MUST send an OPTIONS request first to complete the switch to TLS/1.0 (if possible). After the request when the server is ready to initial TLS handshake, it will send intermediate "101 Switching Protocol". Once the TLS handshake is completed, the server will continue with the response to original request. The server side view is just about the same.

HTTP 1.1 Upgrade header will indeed reduce the redundancy of port usage. But it does have a security flaw, which is the potential threat for man-in-the-middel attack (by deleting the Upgrade header). It is similar to rewriting web pages to change https:// links to http:// links. This flaw is due to using both unsecure and secure channel for some information. It can be overcome by only sending Upgrade request to the server from the beginning with no-op method such as OPTION. But to do this, client has to know that the server is TLS-compliant.

E-legitimation or electronic ID

There are many schemes for mutual authentication on the internet. One that is growing in Sweden is E-legitimation or electronic ID. Like using your ordinary ID, passport or other kind of identifications, an e-ID works exactly the same but for services available online. Services might be signing your income-tax return form (deklarera) or signing a form for getting the student mortgage (CSN). Here are two commonly used icons, the top one to identify your self and the bottom one to sign a form.

Where is the e-ID kept?

E-ID is based on a public/private key infrastructure. The private key can be saved on a bank card, credit card, smart card (which is e-ID on a card) or in a crypto graphed file on your computer. If you get the e-ID onto a card, you also have to buy a card reader. If saved in a file it should be safe as long as no one enters your computer and get hold of your personal code. You must as well keep the computer up to date to prevent attacks from the internet. It could be safer to save it somewhere external like on a USB-memory… or on a smart card.

Obtaining the certificate

Since there are some different ways of getting the certificate we're not going to go through them all. The main thing is that you must first at some point identify your self in one of the local authorities that offers e-ID. Once home you download a program from the authority that through a password, will generate the private and public keys for your e-ID.

Cryptographic Algorithm

The technology used in E-id is the RSA signature generated from a 8-12 digits password. This is an algorithm suitable for encryption as well as signing. RSA is a deterministic encryption scheme which can leak information through eavesdropping. Therefore when using RSA, it must always be combined with some form of padding scheme (encryption). The RSA keys are normally 1024-2048 bits long but the Swedish e-ID is only a 1024-bit key. There are discussions among experts who some say that 1024-bit keys will with in near future be broken.

In number theory and cryptography, TWIRL (The Weizmann Institute Relation Locator) is a theoretical hardware devise. It's a hardware devise built to speed up the process of the 'general number fields sieve' integer factorization. In mathematics, the general number field sieve is the most effective algorithm known for factoring integers larger than 100 digits. The TWIRL algorithm searches for number with a mathematical relation, also distributed over a large number of

processors. However, TWIRL is yet not built. But the designers think that a 1024-bit key could be broken in a year.

The RSA signature is on a SHA-1 hashed message. This through the SHA-1 (secure hash algorithm) hash which has been proved not to be that safe. Safer is the newer version, SHA-2 as well as a 2048-bit key would be instead of a 1024.

Safety

Why is not the stronger encryption then used? The reasons for this are many but one of the biggest reasons according to many of the suppliers is that an attacker could theoretically mess some ones tax-return form about. But, so far there are no services available where the attacker has the possibility to gain a big amount of money. There of, fewer attacks. A long encryption key also takes longer to encrypt the message, which will maybe lead to unnecessary work for the user.

Some authorities will not allow the e-ID on a file on your computer. If a virus attacks your PC and copy the private key, it will not disappear from your computer and therefore harder for the user to understand that it has been stolen. If the private key is on a card, it will never be under attacks from the internet and therefore a smaller encryption key is Ok. And if a smart card gets lost, it is a physical thing that is lost and the same procedures will take place as if loosing a credit card.

Summary

- HTTP authentication scheme alone is not safe.
- SSL can make http secure but in order to make it mutually authenticated it requires the user to have access to a PKS scheme.
- HTTP 1.1 Upgrade header will reduce the redundancy of port usages but it does have some problem regarding man-in-the-middle attack.
- The e-ID is built on the public key algorithm.
- E-ID has a RSA signature with 1024 bit key which is generated from a 8-12 chars password.
- Signature is on a SHA-1 hashed message

Sources

- Computer Security, D Gollmann, Wiley 2005
- Computer Networking, A top down approach featuring the internet, J F Kurose K W Ross, Addison Wesley 2001
- http://en.wikipedia.org/wiki/Https
- http://en.wikipedia.org/wiki/Secure_Sockets_Layer
- http://tools.ietf.org/html/rfc4346
- http://www.cs.chalmers.se/Cs/Grundutb/Kurser/krypto/lect07_4.pdf
- http://en.wikipedia.org/wiki/SHA-1
- http://en.wikipedia.org/wiki/RSA
- http://www.nordea.se/sitemod/default/index.aspx?pid=207964
- http://www.pts.se/internetsakerhet/Sidor/sida.asp?SectionId=1825
- http://technet2.microsoft.com/WindowsServer/en/library/1b6b0dfa-a7a0-4cc2-adc6-f9dda2bd7e601033.mspx?