AS-SU/99-004

Electronic Document Handling

World Wide Web Security

Security implications of EDH's migration to the World Wide Web

By Derek Mathieson (AS-SU)

First published: 14th of April 1997

Revised: 19th of May 1999

World Wide Web Security

Security implications of EDH's migration to the World Wide Web

Notes on Revised Version

See the last page of this document for information on how we have actually implemented Web EDH.

Introduction

With the proposed EDH access from the World Wide Web a method of preventing unauthorised access to private CERN information must be found. There are many possible ways of restricting access to information on the World Wide Web; this document outlines some of them and discusses which will most suitable for EDH.

Current Situation

In its current form EDH uses three methods of access control.

- Access to EDH documents is via an EDH login password, which is encrypted using a modified DES (Data Encryption Standard) algorithm with a static 'key'. In reality this method is only slightly more secure than the 'Telnet' program used to connect to a computer on the network.

- Access to individual fields within the documents. This access control is made at the level of the application and is not affected by EDH's migration to the Web.

- Approval of EDH documents is made with a second authorisation password, which is encrypted using the same DES algorithm but this time using a 'key' which changes every time. This encryption, although not completely invulnerable, can be considered sufficiently secure, that is, it would require an unreasonable amount of effort in order to approve a document without the correct password.

The document data itself is not encrypted while travelling over the network between the database and the EDH Client software.

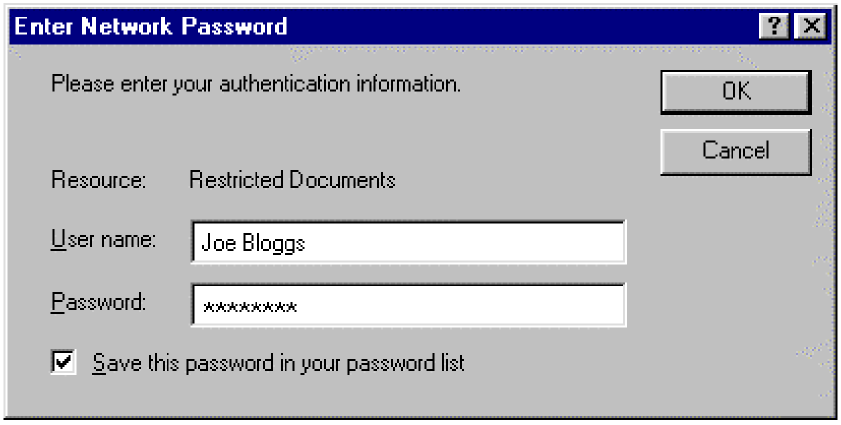
Web Technologies

Several security schemes have been developed for use on the World Wide Web; the following is a description of the most popular:

Basic Client Authentication

This is the most commonly used form of access control, and the least secure. In this scheme the Web server asks the browser to provide authentication information when the user tries to access the secure

Web site. You will be familiar with Netscape or MSIE prompting for a username and password when accessing certain sites.


Enter Network Password

Please enter your authentication information.

Resource: Restricted Documents

User name: Joe Bloggs

Password: *********

☑ Save this password in your password list

Figure 1: Basic Authentication Dialog

After completing the dialog your browser will send the username and password every time you access a Web page from the secure site. The username and password is not encrypted when it is sent over the network. This makes this method even more vulnerable to attack than `telnet` since the password can potentially be sent over the network many times. It would be relatively easy for a network spy (a piece of software which can be obtained free of charge for any PC and is standard on many UNIX platforms) to obtain the username and password information.

This is the form of encryption used by the CERN Materials Catalogue on the Web.

Digest Authentication

This is an improvement to the Basic Client Authentication mechanism that has been proposed by the Internet Engineering Task Force (IETF). The technique uses a secure, one-way hash function called MD5 (Message Digest version 5) to encrypt the username, password and a random number sent each time by the server. The server can authenticate the user by performing the same operation and comparing it with the value sent by the client.

The security of this mechanism is quite high, since the use of the one-way hash function means that a network spy cannot use any information he captures in order to discover the user's password. Unfortunately, we cannot currently use this technique for EDH, as most browsers do not support this authentication mechanism.

Encrypted Cookies

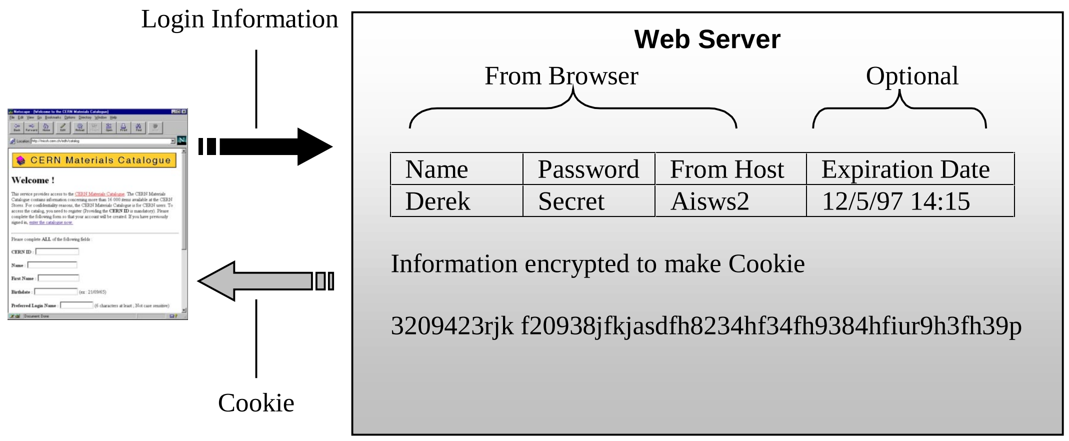
After Netscape introduced the concept of cookies (see box What are Cookies?), some Web sites have begun to use them as a form of access control. In this scheme the 'logs in' to the Web site by completing an HTML form. The username and password information is sent to the server (unencrypted) then the user submits the form, and in reply the Web server installs a cookie in the users browser.

What are Cookies?

'Cookies' is the plural of the American word used for biscuits…

Cookies are a general mechanism which a Web Server can use to both store and retrieve information on the Web Browser. A server, when returning an HTTP object to a client, may also send a piece of state information which the client will store. Included in that state object is a description of the range of URLs for which that state is valid, and an expiration date after which the cookie is no longer stored. Any future requests made by the client which fall in the specified range of URLs will include a transmittal of the current value of the cookie from the client back to the server.

The content of the cookie is encrypted in such a way that when the Web server is given the cookie in future, it can verify who is the person making the request. The encryption of the cookie ensures that an attacker cannot falsify a cookie in order to gain access to another persons account.


Login Information
Cookie
CERN Materials Catalogue
Welcome !
Web Server
From Browser
Optional
Information encrypted to make Cookie
3209423rjk f20938jfkjasdfh8234hf34fh9384hfiur9h3fh39p

Name	Password	From Host	Expiration Date
Derek	Secret	Aisws2	12/5/97 14:15

Figure 2: Encrypted Cookies

If the Browser tries to access a secure area and is unable to supply a valid cookie, they would be immediately redirected to the login screen. The cookie would normally be set to expire when the user quits their browser or after 24hours (whichever happened first).

I have an example implementation of this mechanism on aisws5 that uses the IDEA (see box What Is IDEA? How Secure Is IDEA?) encryption algorithm to encrypt the cookie.

What Is IDEA? How Secure Is IDEA?

IDEA is a high performance symmetric encryption algorithm created by ASCOM Systec AG in Switzerland.

There have been no advances in the cryptanalysis of standard IDEA that are publicly known. The only method of attack, therefore, is brute force. For a brute force attack of IDEA with a 128 bit key, if you had 1,000,000,000 machines that could try 1,000,000,000 keys/sec, it would still take all these machines longer than the universe as we know it has existed, to find the key. IDEA, as far as present technology is concerned, is not vulnerable to brute-force attack.

Encrypted cookies are as secure as telnet since the username and password still travel over the network, but only once, and encryption should prevent the forging of cookies.

An improvement to this mechanism would be the use of Java to encrypt the initial username and password before it travels over the network. Thus making it less susceptible to attacks from network spies.

Secure Authentication Using Java

By making use of Java technology it is possible to implement a more secure means of authentication over the network. The mechanism operates as follows:

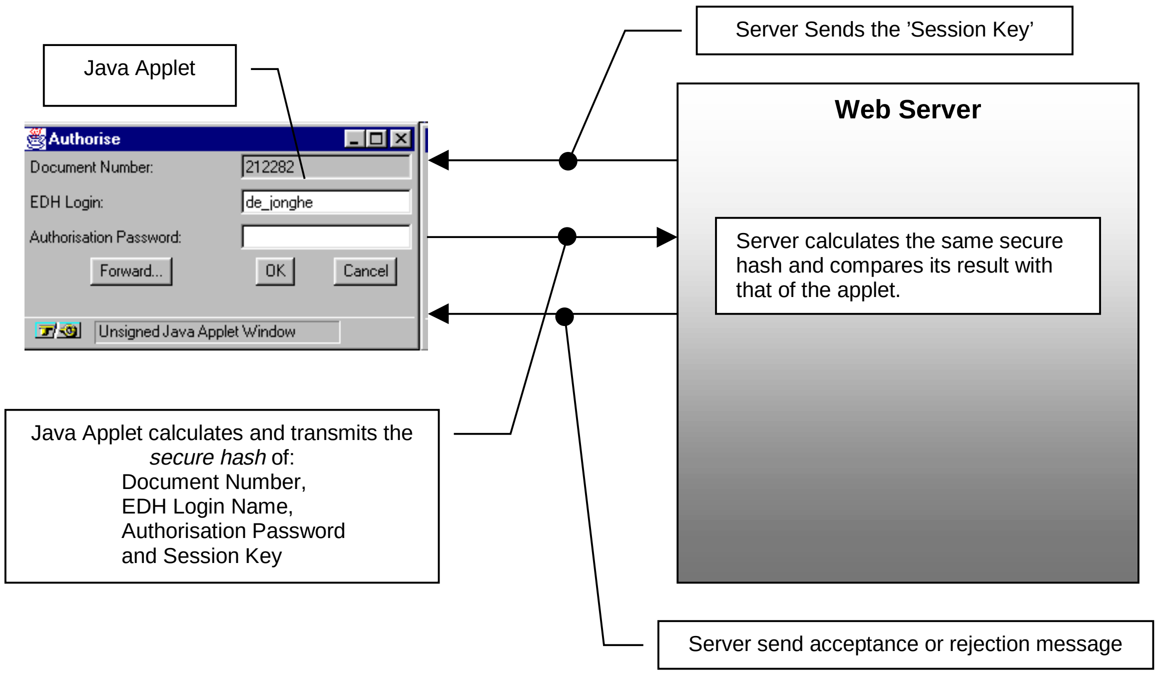

Server Sends the 'Session Key'
Java Applet
Web Server
Authorise
Document Number:
212282
EDH Login:
de_jonghe
Authorisation Password:
Forward...
OK
Cancel
Unsigned Java Applet Window
Server calculates the same secure hash and compares its result with that of the applet.
Java Applet calculates and transmits the *secure hash* of: Document Number, EDH Login Name, Authorisation Password and Session Key
Server send acceptance or rejection message

Figure 3: Secure Authentication Using Java

The secure hash algorithm used is called MD5. The MD5 algorithm produces a 128bit hash from arbitrary length input message. The reason that it is *secure* is that it is mathematically difficult to find another message that will produce the same 128bit hash, this is called a hash collision. A study in 1994 estimate that a collision search machine designed specifically for MD5 (costing $10 million) could find a collision for MD5 in 24 days on average.

The session key is a random number that is generated by the server each time a signature is required. This number guarantees that the secure hash is always different, even if the same person is asked to sign the same document more than once.

Secure HTTP (`https`)

The most secure form of access control is the use of secure HTTP which makes use of public key encryption (see later). This technology was primarily developed for electronic commerce over the Internet where confidential information, such as Credit Card numbers, needs to be protected from access by third parties.

What is Public Key Encryption?

In public key encryption, each person gets a pair of keys, called the public key and the private key. Each person's public key can be published freely, while the private key is kept secret. Using an asymmetric encryption algorithm such as RSA (see What Is RSA? How Secure Is RSA?), the message is encrypted using the receivers public key before transmission, the asymmetric algorithm means that the message can only be decrypted with the receivers private key.

In general the public key (asymmetric) algorithm is complicated (read *slow*) to calculate, therefore in most public key implementations, the public key encryption is only used to encrypt another *secret key* which is then used to decrypt the bulk of the message using a faster (symmetric) algorithm.

International Regulations

The law in most developed countries classifies the use of 'strong encryption' in the same category as munitions. The use of encryption where the secret key is longer than 40-bits is generally prohibited in most countries unless the key is made available to the government. There are specific restrictions in each country, but the widespread use of secure Web Browsers such as Netscape and MSIE indicates a general acceptance of the 40-bit limitation.

What Is RSA? How Secure Is RSA?

RSA, the first full-fledged public key cryptosystem was designed by Rivest, Shamir and Adleman in 1977.

RSA gets its security from the apparent difficulty in factoring very large composites. However, nothing has been proven with RSA. It is not proven that factoring the public modulus is the only (best) way to break RSA. There may be an as yet undiscovered way to break it. It is also not proven that factoring **has** to be as hard as it is. There exists the possibility that an advance in number theory may lead to the discovery of a polynomial time factoring algorithm. But, none of these things has happened, and no current research points in that direction. However, three things that are happening and will continue to happen that take away from the security of RSA are: the advances in factoring techniques, computing power, and the decrease in the cost of computing hardware. These things, especially the first one, work against the security of RSA. However, as computing power increases, so does the ability to generate larger keys. It is much easier to multiply very large primes than it is to factor the resulting composite (given today's understanding of number theory).

Current U.S. Government policy generally limits exportable mass-market software that incorporates encryption for confidentiality to using the RC2 or RC4 symmetric encryption algorithms with 40-bit keys. A 40-bit key length means that there are 2^{40} possible keys. On average, half of these (2^{39}) must be tried to find the correct one.

The recent successful brute-force attack by two French graduate students on Netscapes 40-bit RC4 algorithm demonstrates the dangers of such short keys. These students at the Ecole Polytechnique in Paris used "idle time" on the school's computers, incurring no cost to themselves or their school. Even with these limited resources, they were able to recover the 40-bit key in a few days.

Secure HTTP guarantees that the messages between your browser and the Web Server cannot be read by a third party, it does not, however, guarantee that the person running the Web Server is whom they say they are. It was because of this problem that digital certificates were developed.

Certificates

A digital certificates contains the following information:

- The certificate issuer's name
- Who the certificate is being issued for (a.k.a. the subject)
- The public key of the subject
- Some time stamps

The certificate is *signed* using the certificate issuer's (commonly known as a Certifying Authority or CA) private key. Everybody knows the CA's public key therefore they can verify the authenticity of the Certificate. Certificates are a standard way of binding a public key to a name. If you trust the CA, then you can trust the certificates issued by it. Netscape and MSIE come pre-loaded with a variety of certificates from CAs that are 'trusted'. If you apply to one of these CAs then you can obtain a certificate that will be trusted by all Netscape or MSIE browsers. At the moment only Verisign (http://www.verisign.com) are willing to issue certificates to other organisations, below is there current pricing policy (as of 14th April 97).

Secure web servers require Digital IDs for servers. A one-time setup fee is included in the initial cost of a server Digital ID. The setup fee is reduced for Digital IDs for additional servers within the same organisation.

Secure Web Server Digital ID Pricing	
First Server Digital ID	First year: US$290
Additional Server Digital Ids	First year: US$95 each
Server Digital ID Renewal	Annual fee: US$75 each

Alternatively CERN could become a CA itself, this would give it the ability to produce certificates for any secure application at CERN without the need to apply to Verisign each time. For this to work, the Web Browser needs to be told to trust certificates issued by CERN. Luckily, this is a relatively simple operation that only requires the user to click on a link with a special CA certificate in it. I have prepared an example on a secure Web server that I have set up on aisws5. Point your browser to http://aisws5 and follow the instructions. The installation of the CA certificate needs to be done only once, after which all secure documents that are counter-signed by the CERN certificate will be accepted by your browser.

Options

Basic Client Authentication is too weak for use in EDH. It is even more susceptible to attack than programs such as `telnet`, which are already regarded by most people as relatively insecure.

Digest Authentication is sufficiently secure, but not supported by most browsers, therefore cannot be considered for use by EDH.

Encrypted Cookies provide a good means for access control but do not provide a sufficiently secure way for approving the document, since it is still possible (albeit difficult) to view the password as it travels over the network. As stated earlier, this mechanism can be improved with the use of Java.

Secure Authentication Using Java can provide the most secure authentication mechanism of any available technology.

Secure HTTP is the easiest security mechanism, with the advantage that all information sent over the network is encrypted. Unfortunately, the legal requirement to have a key length less than 40-bits means that the messages could be decrypted in about 5 days using off-the-shelf technology (much less if the attacker was willing to spend money on custom hardware).

Proposal

I would prefer to use secure HTTP, plus Java to secure the authorisation process. We could apply directly to Verisign for a digital Certificate, or CERN could issue its own certificate. In reality, the application to Verisign may be very time consuming since, I suspect, it ought to be made via the CERN Web-Office.

Therefore, I propose that, initially, we make use of Encrypted Cookies (made more secure by using Java) to perform access control, and use Java to secure the authorisation process.

Later, in a second phase, we should define CERN as a CA, thus making the use of secure HTTP possible, although we should continue to use Java to secure the authorisation process.

Update to Original Document

Since this document was originally published, the EDH application has now moved to the World Wide Web.

Implementation

Following the proposal above, we first implemented a login mechanism using Encrypted cookies and a Java applet to encrypt the password information. Later we defined CERN as a CA and changed to using a secure (HTTPS) server.

Currently the EDH web server is a Netscape Enterprise Server operating using the secure HTTPS protocol. Information passed over the network between our server and a user is encrypted with a 128 bit symmetric cipher, although US export regulations restrict the security of this protocol such that only 40-bits are truly secret. This level of encryption is felt to be sufficient for the encryption of the contents of the electronic documents.

When a user connects to EDH for the first time that day, they are requested for their Username and Password. If they provide valid information, an encrypted Cookie is sent back to their browser containing their login details. This cookie is encrypted using the RSA public key algorithm with an 800-bit modulus. Full details of our 'Common Login System' can be found at http://edh.cern.ch/Help/Security.

When a user is required to 'sign' a document we increase the security of the HTTPS connection by masking their authorisation password using a JavaScript implementation of the MD5 algorithm plus a unique 'Authorisation Key' generated by our server.

CERN has been defined as the Certifying Authority (CA) for our secure servers. This means that before an external user can use EDH they must download our Root Certificate from our web site (http://aislogin.cern.ch/cert). The CERN Root Certificate uses a modulus size of 1024 bits.