21

Secure World Wide Web access to server groups

A. Hutchison, M. Kaiserswerth, and P. Trommler
IBM Research Division, Zurich Research Laboratory
Säumerstrasse 4, 8803 Rüschlikon, Switzerland
Telephone: +41-1-724 8 {506, 373}; Fax: +41-1-710 36 08

Abstract

Existing World Wide Web (WWW) security is organized around server specific *realms*. When several servers are interacted with in a secure manner, authentication information has to be provided for each server. Where separate servers co-operate to provide a set of distributed information as a server group, it is desirable to make authentication as transparent as possible. By extending the HyperText Transfer Protocol (HTTP) to include server group information it is possible for a user to only provide authentication information once for an entire group of servers. Although we have also implemented these extensions for the Basic and Digest Authentication schemes, we argue that Mediated Digest Authentication is most suitable for secure server group scalability.

Keywords

World Wide Web security, authentication, server group, HTTP extension, mediated digest authentication

1 INTRODUCTION

The World Wide Web (WWW) has become the most popular means of sharing information among individuals and companies connected to the Internet. While most information is shared freely, some information (and service) providers choose to make their data accessible only to registered users, who must authenticate themselves when they access the service. With currently used WWW technology this authentication information travels in the clear over an insecure link and is thus susceptible to observation.

As many institutions and companies have started to exploit the WWW infrastructure in cooperations, they use this same authentication technology to contain information sharing within their consortia. If they operate different WWW servers, they must register each authorized user individually at each server and users must provide this authentication information whenever they access a new server within the consortium. Clearly, this is a cumbersome procedure and hardly transparent for users who will be prompted for authentication information whenever they access a new server in the group.

In this paper we therefore describe a new WWW authentication method that both pro-

P. Horster (ed.), *Communications and Multimedia Security II*

tects the authentication information and allows a user to only provide his authentication information once for an entire group of servers (residing possibly even in different Internet domains, but belonging to a same logical group that has chosen to share its information among its members). Our work is based on the Internet draft proposal *Mediated Digest Authentication* (Raggett, 1995) and extensions we propose to the *HyperText Transfer Protocol* (HTTP) (Berners-Lee et. al., 1996).

After a brief review of the current and newly proposed WWW authentication mechanisms in the next section, we will describe our concept of group authentication and how it ties in with Mediated Digest Authentication (MDA) in section three. Section four describes our implementation and what we chose to change in MDA. Finally, section five is a summary and outlook on further work on WWW authentication.

2 WORLD WIDE WEB AUTHENTICATION

In this section we will briefly present the current and proposed WWW authentication mechanisms as they relate to this paper. The authentication context is HyperText Transfer Protocol accesses from a browser to a WWW server.

2.1 HyperText Transfer Protocol

HTTP is the main protocol used in the WWW to exchange information between a client's browser and a WWW server. We will base our presentation on HTTP V1.0, which is the currently used protocol. HTTP is a stateless protocol built directly on top of TCP/IP. For every information exchange between client and server, a new TCP connection must be established. In the HTTP header, the client specifies, via a so-called *Universal Resource Identifier* (URI), which resource it wants to access and what method should be applied to the resource. The three defined methods are `GET` (retrieving information), `HEAD` (a GET which does not return the actual entity but only an HTTP protocol header), and `POST` (requesting the server to accept the entity enclosed in the request).

Authentication of HTTP V1.0 uses a simple challenge-response mechanism, which works as follows: a server which requires clients to authenticate themselves replies to a client's HTTP method invocation with an `Unauthorized` (401) error code, and an authentication challenge that contains an indication of the required authentication method. The client may then resubmit its request including authentication information, if it wants to access the server's resource. The authentication challenge is always tied to a server-defined *protection space*. The protection space is defined through a combination of the server's root *Uniform Resource Locator* (URL) and a server-specified realm. For all practical purposes, the root URL is the server's name in either DNS or in dotted IP address notation. The realm is a name the server administrator chooses to partition a file system into different protection spaces. In a typical implementation, the client's browser would prompt the user for authentication credentials before resubmitting the request. For future accesses to the same protection space the browser may then cache the user credentials and automatically include them in requests without further prompting its user.

HTTP V1.0 specifies only a Basic Authentication (BA) scheme where the browser includes the user identification and password information as a *base-64* encoded string in an authentication field in the request header. This scheme is described in the draft as "*a*

non-secure method of filtering unauthorized access to resources on an HTTP server", as this authentication information travels in the clear over an essentially insecure network and is thus susceptible to capture by an observer who could then masquerade as the original user.

An example of an authentication challenge in an HTTP response header is:

```
WWW-Authenticate: Basic realm="x-rays"
```

It specifies that the basic HTTP authentication scheme is to be used for authentication and that the information pertains to a protection space defined by the server's root URL combined with the realm *x-rays*. A browser would then request authentication information from the user, re-requesting the information with an HTTP request header that contains, for example, the following credentials:

```
Authorization: Basic YWh1OmFodQ==
```

To address the security concerns of the basic HTTP authentication method, two proposals, *Digest Authentication* and *Mediated Digest Authentication*, have been made over the last year to ensure that the authentication information never travels in the clear and that replay attacks are prevented by inclusion of unique information upon subsequent accesses by the same user to a given server protection space.

A server implementing digest authentication (DA) (Hostetler et. al., 1996) returns a nonce with its authentication failure response as an authentication challenge, with an indication that the authentication method is DA. The browser constructs a digest from the user's password, the accessed realm and the nonce using the MD5 hash. It then transfers this digest, together with the user identification, as authenticator in its request message. The server recalculates this digest, and makes an authentication decision on the basis of this. Note that this protocol may introduce state information in the server as it may have to remember which nonce it used for a given user. At any time the server may re-challenge the client's browser with a new nonce, which will change the password-nonce digest information in the authentication header, to prevent replay of captured authentication information by an intruder.

Mediated digest authentication (MDA) (Raggett, 1995) builds on DA, but introduces an authentication server which shares secret keys with both browsers and servers enabling it to act as a trusted mediator to mutually authenticate client and server to each other. We will describe MDA in more detail in the next section where we will discuss our solution for secure authentication to WWW server groups.

Before finishing this overview though, we point out two more mechanisms that allow for secure authentication, namely *Secure-HTTP* and the *Secure Sockets Layer*.

2.2 Secure HyperText Transport Protocol

Secure HTTP (S-HTTP) (Rescorla and Schiffman, 1996) may guarantee message integrity (and potentially even non-repudiation) through message authentication codes and signatures or use encryption to provide for confidential ity of the communication. Essentially S-HTTP defines a security wrapper which can be used to encapsulate the message content, which could be an HTTP message or simple data. S-HTTP can use multiple key manage-

ment mechanisms such as password style manually shared secrets, public-key exchange, Kerberos ticket distribution, and prearranged symmetric session keys.

A challenge-response mechanism allows both parties to ensure the freshness of transmissions. S-HTTP does not support selective field confidentiality and the granularity of protection is a complete HTTP transaction. S-HTTP even allows for authentication between multiple parties in a transaction path. This means that a proxy (in a proxy/firewall environment) may be authenticated along with the end client or server. S-HTTP currently is an Internet draft and not used widely.

2.3 Secure Sockets Layer Protocol

The Secure Sockets Layer (SSL) (Freier et. al., 1996) was developed by Netscape Communications Corporation*. The protocol is designed to authenticate the server and optionally the client. SSL is independent of the application protocol, and higher level application protocols (such as HTTP) can be layered on top of SSL. The SSL protocol can negotiate an encryption algorithm and session key (and authenticate a server) before the application protocol transmits or receives any data. Data is transmitted in encrypted form, ensuring privacy.

SSL operates on the basis of longer term security associations oriented around the generation of a master key between two parties and short-lived sessions with per-session keys derived from the master key. No selective field confidentiality is provided in SSL, so when confidentiality is provided all data exchanged between parties is encrypted. No non-repudiation services are provided by SSL.

A number of issues are raised with the use of SSL in a proxy/firewall environment. SSL is intended to create a secure channel between transport peers while HTTP proxies expect to operate on application data. This means that either an SSL association can be established through the proxy (in which case the firewall cannot determine what information flows through the SSL pipe), or the proxy can establish SSL associations with the client and/or server.

SSL requires the maintenance of state information (keys, counters, identity and algorithm information) by both the client and the server. SSL does not support authentication by multiple parties along a transaction path.

3 SECURE AUTHENTICATION TO A GROUP OF WWW SERVERS

The goal of our work is to extend single server authentication to allow seamless retrieval of related documents from any other server belonging to a group, without the intrusion of the browser prompting for a user-id and password each time a new server is accessed. As described in the previous section, HTTP authentication information is, however, always relative to a server's protection space which is based on the combination of the canonical root URL of that server and a server-defined realm. As a root URL always gives the

*All versions of the Netscape Navigator browser (beginning with Beta 0.93 for Windows, Mac and Unix) have integrated support for SSL via a new URL access method "https". Due to export restrictions a 40-bit key size is used for the RC4 stream encryption algorithm.

server's Internet address, the protection space is uniquely tied to that server and cannot be extended to other servers that are only known under different Internet addresses.

To extend the HTTP protection space beyond a single WWW server, the definition of the protection space must be made relative to a unique group name rather than a (root) URL and the browser must be notified that the protection space information pertains to a group of servers rather than a single server.

3.1 Group names and HTTP modification

We considered a number of different solutions to define unique group names for the global WWW. X.500-like distinguished group names were one candidate, but we decided that for the time being the overhead associated with maintaining such a naming scheme would be excessive. We rather elected to use the DNS name of a distinguished server in the server group as a unique base for the group name. This server name combined with a freely chosen group name then defines a protection space that can span multiple servers belonging to the same logical group. A world-wide unique group name defined with the above scheme might thus be:

```
hundshorn.zurich.ibm.com:med_image_grp
```

To communicate the fact that the required authentication is no longer for a single server's protection space but rather for an entire group, we extended the HTTP protocol through defining an additional field in the WWW-Authenticate header of an HTTP response message. The header contains, next to the description of the authentication method and the realm, a new token `servgrp`. For the above example, the authentication challenge would thus be:

```
WWW-Authenticate: Basic realm="x-rays",
servgrp="hundshorn.zurich.ibm.com:med_image_grp"
```

In this example a protection space is created that extends across all servers that choose to include the *servgrp* token with the same value (`hundshorn.zurich.ibm.com:med_image_grp`) and also specified the same realm (`x-rays`).

Extending HTTP in this way allowed us to stay upward compatible, as we did not change the authentication scheme. Browsers unaware of groups but able to perform basic authentication, will silently ignore the new token and will simply perform authentication as described in section 2. Note that the new token can now also be used in other authentication schemes, as we did in our implementation of group authentication using DA or MDA in addition to the basic authentication scheme.

A major disadvantage in using BA or DA for group authentication is that each server in a group is responsible for maintaining its own authentication information database and must keep the information in it synchronized with the other servers in the group. Introducing a new user could thus be a lengthy and error prone process. While DA assures confidentiality of the authentication information, BA information travels in the clear and could either be captured on its way to a server in the group or solicited by a server claiming to belong to the group. Clearly a scheme that allows central maintenance of authentication credentials and assures that they cannot be misused by intruders would

be a better way to provide group authentication. We therefore chose to explore MDA as the preferred authentication scheme.

3.2 Mediated Digest Authentication

MDA is based on the concept of using one or more authentication servers that both client and WWW server mutually trust. It ensures mutual authentication, i.e., a client can also be sure that it is talking to the correct server. The MDA protocol works as follows: in its HTTP request a client includes a nonce. The nonce would be ignored by servers employing a different authentication scheme. A server which requires MDA a would respond with an error code 401 (Unauthorized) and a WWW-Authenticate challenge which contains a server nonce and adds one or more *Trusted-Party* headers listing authentication servers. The client now selects an authentication server it shares a secret key with and sends both parties' authentication information to that server. The authentication server authenticates both client and WWW server and generates a secret shared session key for both partners. The client resends its request to the WWW server attaching the protected session key and the authentication information. The server authorizes the request, returns the data and optionally adds a message authentication code for detection of data altered in transit. The client or server may at any time choose to select a new nonce which would require re-authentication via the trusted authentication server. Any encryption of authentication information is based on shared secrets that are XORed with one-way hashes that both parties are able to reconstruct from shared knowledge. The scheme is thus not liable to any export restrictions and can be used world-wide.

The protocol used between client and authentication server is not HTTP but a proposed new protocol Mediated Digest Authentication Protocol (MDAP) which uses UDP as the underlying transport. An overview of the information flow is given in figure 1.

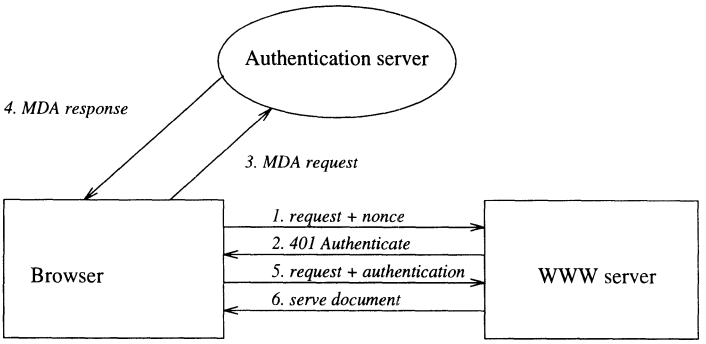

Figure 1 The mediated digest authentication protocol

4 IMPLEMENTATION

In this section modifications required for server and browser implementations of secure WWW access to server groups are described. Our implementation was based on the httpd V1.5a server and Mosaic V2.7b2 browser of NCSA both of which supported BA and DA.

4.1 Server modifications

For the server side, two general modifications were required to support server groups. The first modification was to add to the access control configuration files the information about server group membership. The second modification related to the actual processing of the configuration files. For notifying the browser of membership of a server group, an additional field for the `WWW-Authenticate` header of HTTP must be generated.

To support MDA, the various defined header fields must be evaluated and acted upon. In the HTTP Authenticate response returned by the server to the browser, a list of trusted third parties must be supplied. We decided to put the list of trusted third parties into a separate file referenced from the access control file to facilitate sharing of this information between different access control files.

Since the `WWW-Authenticate` header used in the 'Mediated Digest' scheme is identical to that used in the 'Digest' scheme, this code could be reused. The only difference is that instead of a password, the generated session key is used to construct the *opaque* field. From the mediation response which the client receives, a `Session Key` header is returned to the server. This header contains the encrypted *session key* which the server should cache (Raggett, 1995).

Since HTTP was designed to be stateless, we decided to retain this property by modifying the MDA proposal and requiring the user agent to resend the session key information with each request. Alternatively one could deploy a proposed state mechanism for HTTP as suggested in (Kristol, 1996).

4.2 Browser modifications

As described, the name of a server group is passed as an additional parameter of the `WWW-Authenticate` header. In constructing the HTTP `Authorization` header for return to the server (activated by receipt of a '401', unauthorized, server status), we first check for the presence of a *servgrp* parameter. If there is no such parameter, i.e. the server does not belong to a group or does not even support server groups, the usual processing of single server authentication takes place. Otherwise the group name is looked up in a global (as opposed to a server-specific) list containing the authentication information for each group. In the case of a successful lookup, i.e., a server belonging to a group that has been accessed before, the user name and password are re-used for generating the authentication string. Otherwise the user is prompted for his name and password for the particular group. This information is then added to the global list of groups for later reference. The list of groups contains the group name, the user name, and a password for the group. In other words, a server group is handled like a world wide realm.

The `servgrp` name serves only as a tag to indicate membership of the server to a particular group. To generate the authentication information, the realm name is used (cf. digest authentication). This ensures that the user does not need to remember a second

user name and password, when using a browser that is not group-enabled. Furthermore it simplifies administration of the password files. While the user will not benefit from the single sign on capabilities he will still be able to access his documents.

For MDA, the browser has to interpret the `Trusted-Party` headers. Since a mutually trusted party has to be found, the browser requires a list of parties with which it shares a secret key. The secret keys also have to be stored in some secure manner or generated from a password (e.g., the MD5 hash of the password). If a mutually trusted party can be found, then an authentication request message is constructed. If a mutually trusted party cannot be found, then some out-of-band registration activity is required.

The mediation response received can be either success (status = 100), or failure, indicating that the user-name was unknown (status = 200) or that the WWW server failed the authentication check (status = 201). If a successful response was received, the browser retrieves the session key in a manner similar to that described for the server, and constructs a `Session Key` header for forwarding to the server. The session key is used in constructing the `Authorization` header.

An implementational note (not specified in the draft proposals) is that when existing MD5 digests were used to calculate further digests, they were always included in hexadecimal format. This is important for compatibility, since if one agent calculates a digest containing another digest where the integer representation is used and the other uses hexadecimal, the digests will not match.

4.3 Authentication server

Implementation of the authentication server required a communication function, plus the ability to process authentication requests and generate mediation responses. In order to prevent denial of service attacks to the authentication server the format of packets is checked first and ill-formed packets are dropped. When a request is well-formed, the authentication server uses the browser supplied *user-name* to retrieve the key it shares with the browser. The browser supplied *client-mac* is then recalculated using this key. A match confirms the authenticity of the client. In similar manner, the web server is authenticated by matching the *server-mac* supplied to the browser in the `Trusted Party` header.

If authentication is successful, a session key is generated and returned in the *client-key* and *server-key* fields. These fields are constructed by XORing the session key with a digest including the secret key shared between the browser and the web server, respectively.

5 CONCLUSION

In this paper we have presented a mechanism by which the existing HTTP single-server protection space can be extended to provide protection across multiple servers. The extension enables seamless cooperation amongst servers, and secure authentication if it is used with a method such as DA or MDA.

The nature of our proposal is such that the enhancements can be implemented to coexist with the authentication mechanism of HTTP V1.0, or the various proposed authentication enhancements. The proposal allows upward compatibility, in that non group-enabled browsers simply ignore the additional information.

We also advocate incorporation of secure group enablement as a permanent feature in the revised version of HTTP, V1.1.

An AIX patch for NCSA's httpd version 1.5a and Mosaic for XWindow version 2.7b2 is available via anonymous ftp from `ftp.zurich.ibm.com` in directory `/pub/trp/server-groups`. An implementation of the authentication server can also be found at this location.

6 ACKNOWLEDGEMENT

We are grateful to Günther Karjoth for discussions which initiated this work.

7 REFERENCES

Berners-Lee, T., Fielding, R., and Frystyk Nielsen, H., (1996) "Hypertext Transfer Protocol – HTTP/1.0", Internet Draft, *Work in Progress*, <draft-ietf-http-v10-spec-05.txt>

Freier, A. O., Karlton, P., and Kocher, P. C., "The SSL Protocol Version 3.0", Internet Draft, *Work in Progress*, <draft-freier-ssl-version3-01.txt>

Hostetler, J. L., Franks, J., Hallam-Baker, P., Luotonen, A., Sink, E. W., and Steward, L. C., (1996) "A Proposed Extension to HTTP : Digest Access Authentication", Internet Draft, *Work in Progress*, <draft-ietf-http-digest-aa-03.txt>

Kristol, D. M., (1996) "Proposed HTTP State Management Mechanism", Internet Draft, *Work in Progress*, <draft-kristol-http-state-mgmt-00.txt>

Raggett, D., (1995) "Mediated Digest Authentication", Internet Draft, <draft-ietf-http-mda-00.txt>, *Work in Progress*, (expired)

Rescorla, E., and Schiffman, A., (1996) "The Secure HyperText Transfer Protocol", Internet Draft, *Work in Progress*, <draft-ietf-wts-shttp-01.txt>

8 BIOGRAPHY

Andrew Hutchison obtained his MSc in Computer Science from the University of Cape Town, South Africa in 1991, and his PhD in Computer Science from the University of Zurich, Switzerland in 1996. While studying in Switzerland he worked at the IBM Zurich Research Laboratory†. He is currently a Senior Lecturer in the Department of Computer Science at the University of Cape Town. His e-mail address is `hutch@cs.uct.ac.za`.

Matthias Kaiserswerth received his doctorate in engineering from the University of Erlangen-Nürnberg, Germany. At the IBM Zurich Research Laboratory he manages a group currently working on networked application integration and management. He is also a part-time instructor at the University of Erlangen-Nürnberg. His e-mail address is `kai@zurich.ibm.com`.

---

†This work was done during his time at the IBM Zurich Research Laboratory.

Peter Trommler received his diploma (Dipl. Inf.) in computer science in 1995 from Friedrich Alexander University of Erlangen-Nürnberg, Germany. He is currently working as a Predoctoral Candidate at the IBM Zurich Research Laboratory. His research interests are mobile code and security. His e-mail address is `trp@zurich.ibm.com`.