Frank Fiedler, frank.fiedler@student.fh-nuernberg.de, Matrikelnummer: 670530

HTTP Authentication: Basic and Digest Access Authentication
RFC 2617
Obsoletes RFC 2069

Table of Contents

Positioning at the Internet Layer:

HTTP-Authentication is part of the HTTP-Protocol. The HTTP-Protocol is positioned at Layer 5/Layer 6.

Overview / Purpose

HTTP Authentication wants to provide a build in mechanism for requiring a valid username/password to gain access to web recourses.

HTTP Authentication is initiated by the web server or an external cgi-script

There are currently 2 modes of authentication built into HTTP 1.1 protocol, termed “Basic” and “Digest” Access Authentication.

Basic Access Authentication:

Example:
The HTTP-Header of a standard client requests on some Document in a protected Area:

```
GET /download/report.doc HTTP/1.1
Accept: application/msword, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 10.0.0.5:81
Connection: Keep-Alive
```

Server reads configuration files and determines that resource falls within a protected directory.
→ Server can only allow access to known users.

Server Sends HTTP 401 Authorization Required Response Error:

```
HTTP/1.1 401 Authorization Required
Date: Tue, 22 Jun 2004 03:54:06 GMT
Server: Apache/1.3.29 (Unix)
WWW-Authenticate: Basic realm="Protected"
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

Now the Browser displays Username/Password prompt displaying host name and authentication realm.
→ If the User hits the cancel-Button of this Dialog the HTTP 401 Authorization Required Response Error page send together with the HTTP 401 Authorization Required Response will be shown.

→ If the User enters an Username/Password: the Client Resubmits Request with Username/Password:

```
GET /download/report.doc HTTP/1.1
Accept: application/msword, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 10.0.0.5:81
Connection: Keep-Alive
Authorization: Basic ZnJhbms6ZmllZGxlcg==
```

Server compares client information to its user/password list:

- username : password is valid:
  server sends requested content.
- authorization fails:
  server resends 401 Authorization Required header
- Client hits cancel:
  browser shows error message sent along with 401 message.

The problem with Basic Access Authentication is that the Username/Password is sent in clear text:

`Authorization: Basic ZnJhbms6ZmllZGxlcg==` is just the Base64 encoded username/password pair entered by the User in clear text.

`"ZnJhbms6ZmllZGxlcg==" → base64decode() → "frank:fiedler"`

Digest Access Authentication:

A possible solution of this problem is offered by the Digest Access Authentication. Password won't be sent in clear text. The password will be sent encrypted (normally as md5 hash of the password and some other values)

Therefore some additional Headers are required:

Server requests Authorisation:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
  realm="Protected",
  qop="auth,auth-int",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Description of the additional attributes

realm: Displayed to User in Login-Formula (like at Basic Access Authentication)
qop: quality of protection
Is required for backward compatibility with RFC 2069
- The value "auth" indicates authentication;
- The value "auth-int" indicates authentication with integrity protection. Therefore the Hash value is calculated over the whole entity-body.

nonce: server-specified quoted data string uniquely generated each time a 401 response is made. It will be used for the encryption of the username/password pair.
opaque: quoted data string replied unchanged the whole session by the client; it might be used for example for session tracking by the web server. This field is optional.
stale: flag set if the client or the server requests a new nonce value
TRUE: - if the client wants to reauthenticate
- if the server gets an outdated nonce value but correct user/password from the client calculated with this outdated nonce value

algorithm: one or more algorithms used to encrypt user/password. Normally MD5 is used to encrypt the username/password pair within the Digest Access Authentication.

The nonce value and the opaque value are sent in hexadecimal notation.

Client replies Authorisation:

```
Authorization: Digest username="frank",
               realm="Protected",
               nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
               uri="/dir/index.html",
               qop=auth,
               nc=00000001,
               cnonce="0a4f113b",
               response="6629fae49393a05397450978507c4ef1",
               opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Description of the additional attributes

username: the username in clear text
realm: the realm the user wants to authenticate to
qop: the quality of protection selected by the client
- must be present if the server sent a qop directive
cnonce: client generated unique data string
- must be present if qop is present
nc: nonce-count - the count of requests sent by the client
- must be present if qop is present. The nonce-count is used to avoid reply-attacks.
response: the encrypted password

How the response is encrypted

Depending on quality of protection it's a Hash value of various attributes:

If quality of protection is "auth" and the Algorithm is MD5 the response will be calculated this way:

The Response is the MD5 Hash value of various Attribute values are merged to a long String value separated by colons:

```
Response = MD5( username-value:realm-value:password:nonce-
value:nc-value:cnonce-value:qop-value:request-method-
value:digest-uri-value )
```

Proxy-Authentication and Proxy-Authorization

This authentication scheme may also be used for authenticating users to proxies, proxies to proxies, or proxies to origin servers by use of the Proxy-Authenticate and Proxy-Authorization headers.

Just replace the Authentication-Header:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
...
```

would be at the Proxy Authentication:

```
HTTP/1.1 407 Proxy Authentication Required
Proxy-Authenticate: Digest
...
```

Security Considerations

Basic Authentication is very insecure because of clear text transition of username/password (vulnerable to man in the middle attacks or Network sniffering)

Digest Authentication:

Replay attacks:

The vulnerability depends on the way the nonce-value is created. It can be completely avoided (if necessary) if each nonce-value is only used once. But this would mean a higher CPU load for the web server.

Multiple Authentication Schemas:

Some old Browsers may only support Basic Authentication, so if you offer both – basic and digest access authentication – in some cases the insecure basic access authentication would be forced by the client. Some other web browsers may choose the first offered authentication mechanism. If secure Authentication is required by the Application the web server should only offer digest Authentication.

Online dictionary attacks:

To avoid online dictionary attacks, the usage of "strong" passwords, not listed in any dictionary, must be forced.

Man in the Middle:

Remove all offered choices, replacing them with a challenge that requests only Basic authentication (may realized as http-proxy). So the attacker would receive the clear text username/password bundle from the client.

To avoid this, web browsers should display the auth-mechanism. But most users don’t care about these warnings given by their web clients.

Another possible Man in the Middle attack would be:

- eve sends the same nonce to more clients

→ The time to find the first password will be reduced by each client reply.
→ If one password is known all other passwords can be decrypted

This can be avoided using the cnonce-directive by the clients. So the encryption does not only depend on the nonce value created by the Server but also on the cnonce value created by the client.

Another security problem is the Password-File at the Server. It must be stored at a safe location! The passwords should not be stored not as clear text. The Attacker should never be able to read this file in any way.

Conclusion

If you want to provide User Authentication HTTP-Authentication using Digest Authentication is a good Replacement for Basic Authentication, but compared to modern cryptographic standards HTTP-Authentication using Digest Authentication is also a weak mechanism. Better would be any kind of Authentication using a SSL-Tunnel (HTTPS). Another Problem are weak passwords, so it is also important to force the Users to use strong passwords.

Basic Authentication is very insecure, so it should only be used if no security is needed. In all other scenarios it should be replaced by Digest Authentication. The strength of Digest Authentication depends on the implementation of the creation of the nonce value.