HTTPbis Working Group
Internet-Draft
Obsoletes: 2616 (if approved)
Updates: 2617 (if approved)
Intended status: Standards Track
Expires: January 17, 2013

R. Fielding, Ed.
Adobe
Y. Lafon, Ed.
W3C
J. Reschke, Ed.
greenbytes

July 16, 2012

HTTP/1.1, part 7: Authentication
draft-ietf-httpbis-p7-auth-20

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This document defines the HTTP Authentication framework.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <http://lists.w3.org/Archives/Public/ietf-http-wg/>.

The current issues list is at <http://tools.ietf.org/wg/httpbis/trac/report/3> and related documents (including fancy diffs) can be found at <http://tools.ietf.org/wg/httpbis/>.

The changes in this draft are summarized in Appendix D.1.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines HTTP/1.1 access control and authentication. It includes the relevant parts of RFC 2616 with only minor changes ([RFC2616]), plus the general framework for HTTP authentication, as previously defined in "HTTP Authentication: Basic and Digest Access Authentication" ([RFC2617]).

HTTP provides several OPTIONAL challenge-response authentication mechanisms which can be used by a server to challenge a client request and by a client to provide authentication information. The "basic" and "digest" authentication schemes continue to be specified in RFC 2617.

1.1. Conformance and Error Handling

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This specification targets conformance criteria according to the role of a participant in HTTP communication. Hence, HTTP requirements are placed on senders, recipients, clients, servers, user agents, intermediaries, origin servers, proxies, gateways, or caches, depending on what behavior is being constrained by the requirement. See Section 2 of [Part1] for definitions of these terms.

The verb "generate" is used instead of "send" where a requirement differentiates between creating a protocol element and merely forwarding a received element downstream.

An implementation is considered conformant if it complies with all of the requirements associated with the roles it partakes in HTTP. Note that SHOULD-level requirements are relevant here, unless one of the documented exceptions is applicable.

This document also uses ABNF to define valid protocol elements (Section 1.2). In addition to the prose requirements placed upon them, senders MUST NOT generate protocol elements that do not match the grammar defined by the ABNF rules for those protocol elements that are applicable to the sender's role. If a received protocol element is processed, the recipient MUST be able to parse any value that would match the ABNF rules for that protocol element, excluding only those rules not applicable to the recipient's role.

Unless noted otherwise, a recipient MAY attempt to recover a usable protocol element from an invalid construct. HTTP does not define specific error handling mechanisms except when they have a direct

impact on security, since different applications of the protocol require different error handling strategies. For example, a Web browser might wish to transparently recover from a response where the Location header field doesn't parse according to the ABNF, whereas a systems control client might consider any form of error recovery to be dangerous.

1.2. Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with the list rule extension defined in Section 1.2 of [Part1]. Appendix B describes rules imported from other documents. Appendix C shows the collected ABNF with the list rule expanded.

2. Access Authentication Framework

2.1. Challenge and Response

HTTP provides a simple challenge-response authentication mechanism that can be used by a server to challenge a client request and by a client to provide authentication information. It uses an extensible, case-insensitive token to identify the authentication scheme, followed by additional information necessary for achieving authentication via that scheme. The latter can either be a comma-separated list of parameters or a single sequence of characters capable of holding base64-encoded information.

Parameters are name-value pairs where the name is matched case-insensitively, and each parameter name MUST only occur once per challenge.

```
  auth-scheme    = token

  auth-param     = token BWS "=" BWS ( token / quoted-string )

  b64token       = 1*( ALPHA / DIGIT /
                       "-" / "." / "_" / "~" / "+" / "/" ) *"="
```

The "b64token" syntax allows the 66 unreserved URI characters ([RFC3986]), plus a few others, so that it can hold a base64, base64url (URL and filename safe alphabet), base32, or base16 (hex) encoding, with or without padding, but excluding whitespace ([RFC4648]).

The 401 (Unauthorized) response message is used by an origin server to challenge the authorization of a user agent. This response MUST include a WWW-Authenticate header field containing at least one

challenge applicable to the requested resource.

The 407 (Proxy Authentication Required) response message is used by a proxy to challenge the authorization of a client and MUST include a Proxy-Authenticate header field containing at least one challenge applicable to the proxy for the requested resource.

```
  challenge   = auth-scheme [ 1*SP ( b64token / #auth-param ) ]
```

> Note: User agents will need to take special care in parsing the WWW-Authenticate and Proxy-Authenticate header field values because they can contain more than one challenge, or if more than one of each is provided, since the contents of a challenge can itself contain a comma-separated list of authentication parameters.

> Note: Many clients fail to parse challenges containing unknown schemes. A workaround for this problem is to list well-supported schemes (such as "basic") first.

A user agent that wishes to authenticate itself with an origin server -- usually, but not necessarily, after receiving a 401 (Unauthorized) -- can do so by including an Authorization header field with the request.

A client that wishes to authenticate itself with a proxy -- usually, but not necessarily, after receiving a 407 (Proxy Authentication Required) -- can do so by including a Proxy-Authorization header field with the request.

Both the Authorization field value and the Proxy-Authorization field value contain the client's credentials for the realm of the resource being requested, based upon a challenge received from the server (possibly at some point in the past). When creating their values, the user agent ought to do so by selecting the challenge with what it considers to be the most secure auth-scheme that it understands, obtaining credentials from the user as appropriate.

```
  credentials = auth-scheme [ 1*SP ( b64token / #auth-param ) ]
```

Upon a request for a protected resource that omits credentials, contains invalid credentials (e.g., a bad password) or partial credentials (e.g., when the authentication scheme requires more than one round trip), an origin server SHOULD return a 401 (Unauthorized) response. Such responses MUST include a WWW-Authenticate header field containing at least one (possibly new) challenge applicable to the requested resource.

Likewise, upon a request that requires authentication by proxies that omit credentials or contain invalid or partial credentials, a proxy SHOULD return a 407 (Proxy Authentication Required) response. Such responses MUST include a Proxy-Authenticate header field containing a (possibly new) challenge applicable to the proxy.

A server receiving credentials that are valid, but not adequate to gain access, ought to respond with the 403 (Forbidden) status code (Section 4.6.3 of [Part2]).

The HTTP protocol does not restrict applications to this simple challenge-response mechanism for access authentication. Additional mechanisms MAY be used, such as encryption at the transport level or via message encapsulation, and with additional header fields specifying authentication information. However, such additional mechanisms are not defined by this specification.

Proxies MUST forward the WWW-Authenticate and Authorization header fields unmodified and follow the rules found in Section 4.1.

2.2. Protection Space (Realm)

The authentication parameter realm is reserved for use by authentication schemes that wish to indicate the scope of protection.

A protection space is defined by the canonical root URI (the scheme and authority components of the effective request URI; see Section 5.5 of [Part1]) of the server being accessed, in combination with the realm value if present. These realms allow the protected resources on a server to be partitioned into a set of protection spaces, each with its own authentication scheme and/or authorization database. The realm value is a string, generally assigned by the origin server, which can have additional semantics specific to the authentication scheme. Note that there can be multiple challenges with the same auth-scheme but different realms.

The protection space determines the domain over which credentials can be automatically applied. If a prior request has been authorized, the same credentials MAY be reused for all other requests within that protection space for a period of time determined by the authentication scheme, parameters, and/or user preference. Unless otherwise defined by the authentication scheme, a single protection space cannot extend outside the scope of its server.

For historical reasons, senders MUST only use the quoted-string syntax. Recipients might have to support both token and quoted-string syntax for maximum interoperability with existing clients that have been accepting both notations for a long time.

2.3. Authentication Scheme Registry

The HTTP Authentication Scheme Registry defines the name space for the authentication schemes in challenges and credentials.

Registrations MUST include the following fields:

- Authentication Scheme Name
- Pointer to specification text
- Notes (optional)

Values to be added to this name space require IETF Review (see [RFC5226], Section 4.1).

The registry itself is maintained at <http://www.iana.org/assignments/http-authschemes>.

2.3.1. Considerations for New Authentication Schemes

There are certain aspects of the HTTP Authentication Framework that put constraints on how new authentication schemes can work:

- HTTP authentication is presumed to be stateless: all of the information necessary to authenticate a request MUST be provided in the request, rather than be dependent on the server remembering prior requests. Authentication based on, or bound to, the underlying connection is outside the scope of this specification and inherently flawed unless steps are taken to ensure that the connection cannot be used by any party other than the authenticated user (see Section 2.4 of [Part1]).

- The authentication parameter "realm" is reserved for defining Protection Spaces as defined in Section 2.2. New schemes MUST NOT use it in a way incompatible with that definition.

- The "b64token" notation was introduced for compatibility with existing authentication schemes and can only be used once per challenge/credentials. New schemes thus ought to use the "auth-param" syntax instead, because otherwise future extensions will be impossible.

- The parsing of challenges and credentials is defined by this specification, and cannot be modified by new authentication schemes. When the auth-param syntax is used, all parameters ought to support both token and quoted-string syntax, and syntactical constraints ought to be defined on the field value after parsing

(i.e., quoted-string processing). This is necessary so that recipients can use a generic parser that applies to all authentication schemes.

Note: The fact that the value syntax for the "realm" parameter is restricted to quoted-string was a bad design choice not to be repeated for new parameters.

- Definitions of new schemes ought to define the treatment of unknown extension parameters. In general, a "must-ignore" rule is preferable over "must-understand", because otherwise it will be hard to introduce new parameters in the presence of legacy recipients. Furthermore, it's good to describe the policy for defining new parameters (such as "update the specification", or "use this registry").

- Authentication schemes need to document whether they are usable in origin-server authentication (i.e., using WWW-Authenticate), and/or proxy authentication (i.e., using Proxy-Authenticate).

- The credentials carried in an Authorization header field are specific to the User Agent, and therefore have the same effect on HTTP caches as the "private" Cache-Control response directive, within the scope of the request they appear in.

  Therefore, new authentication schemes which choose not to carry credentials in the Authorization header field (e.g., using a newly defined header field) will need to explicitly disallow caching, by mandating the use of either Cache-Control request directives (e.g., "no-store") or response directives (e.g., "private").

3. Status Code Definitions

3.1. 401 Unauthorized

The request requires user authentication. The response MUST include a WWW-Authenticate header field (Section 4.4) containing a challenge applicable to the target resource. The client MAY repeat the request with a suitable Authorization header field (Section 4.1). If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the representation that was given in the response, since that representation might include relevant diagnostic information.

3.2. 407 Proxy Authentication Required

This code is similar to 401 (Unauthorized), but indicates that the client ought to first authenticate itself with the proxy. The proxy MUST return a Proxy-Authenticate header field (Section 4.2) containing a challenge applicable to the proxy for the target resource. The client MAY repeat the request with a suitable Proxy-Authorization header field (Section 4.3).

4. Header Field Definitions

This section defines the syntax and semantics of HTTP/1.1 header fields related to authentication.

4.1. Authorization

The "Authorization" header field allows a user agent to authenticate itself with a server -- usually, but not necessarily, after receiving a 401 (Unauthorized) response. Its value consists of credentials containing information of the user agent for the realm of the resource being requested.

```
  Authorization = credentials
```

If a request is authenticated and a realm specified, the same credentials SHOULD be valid for all other requests within this realm (assuming that the authentication scheme itself does not require otherwise, such as credentials that vary according to a challenge value or using synchronized clocks).

When a shared cache (see Section 1.2 of [Part6]) receives a request containing an Authorization field, it MUST NOT return the corresponding response as a reply to any other request, unless one of the following specific exceptions holds:

1. If the response includes the "s-maxage" cache-control directive, the cache MAY use that response in replying to a subsequent request. But (if the specified maximum age has passed) a proxy cache MUST first revalidate it with the origin server, using the header fields from the new request to allow the origin server to authenticate the new request. (This is the defined behavior for s-maxage.) If the response includes "s-maxage=0", the proxy MUST always revalidate it before re-using it.

2. If the response includes the "must-revalidate" cache-control directive, the cache MAY use that response in replying to a subsequent request. But if the response is stale, all caches MUST first revalidate it with the origin server, using the header

fields from the new request to allow the origin server to authenticate the new request.

3. If the response includes the "public" cache-control directive, it MAY be returned in reply to any subsequent request.

4.2. Proxy-Authenticate

The "Proxy-Authenticate" header field consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the proxy for this effective request URI (Section 5.5 of [Part1]). It MUST be included as part of a 407 (Proxy Authentication Required) response.

```
  Proxy-Authenticate = 1#challenge
```

Unlike WWW-Authenticate, the Proxy-Authenticate header field applies only to the current connection, and intermediaries SHOULD NOT forward it to downstream clients. However, an intermediate proxy might need to obtain its own credentials by requesting them from the downstream client, which in some circumstances will appear as if the proxy is forwarding the Proxy-Authenticate header field.

Note that the parsing considerations for WWW-Authenticate apply to this header field as well; see Section 4.4 for details.

4.3. Proxy-Authorization

The "Proxy-Authorization" header field allows the client to identify itself (or its user) to a proxy which requires authentication. Its value consists of credentials containing the authentication information of the user agent for the proxy and/or realm of the resource being requested.

```
  Proxy-Authorization = credentials
```

Unlike Authorization, the Proxy-Authorization header field applies only to the next outbound proxy that demanded authentication using the Proxy-Authenticate field. When multiple proxies are used in a chain, the Proxy-Authorization header field is consumed by the first outbound proxy that was expecting to receive credentials. A proxy MAY relay the credentials from the client request to the next proxy if that is the mechanism by which the proxies cooperatively authenticate a given request.

4.4. WWW-Authenticate

The "WWW-Authenticate" header field consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the effective request URI (Section 5.5 of [Part1]).

It MUST be included in 401 (Unauthorized) response messages and MAY be included in other response messages to indicate that supplying credentials (or different credentials) might affect the response.

```
  WWW-Authenticate = 1#challenge
```

User agents are advised to take special care in parsing the WWW-Authenticate field value as it might contain more than one challenge, or if more than one WWW-Authenticate header field is provided, the contents of a challenge itself can contain a comma-separated list of authentication parameters.

For instance:

```
  WWW-Authenticate: Newauth realm="apps", type=1,
                    title="Login to \"apps\"", Basic realm="simple"
```

This header field contains two challenges; one for the "Newauth" scheme with a realm value of "apps", and two additional parameters "type" and "title", and another one for the "Basic" scheme with a realm value of "simple".

> Note: The challenge grammar production uses the list syntax as well. Therefore, a sequence of comma, whitespace, and comma can be considered both as applying to the preceding challenge, or to be an empty entry in the list of challenges. In practice, this ambiguity does not affect the semantics of the header field value and thus is harmless.

5. IANA Considerations

5.1. Authentication Scheme Registry

The registration procedure for HTTP Authentication Schemes is defined by Section 2.3 of this document.

The HTTP Method Authentication Scheme shall be created at <http://www.iana.org/assignments/http-authschemes>.

5.2. Status Code Registration

The HTTP Status Code Registry located at <http://www.iana.org/assignments/http-status-codes> shall be updated with the registrations below:

Value	Description	Reference
401	Unauthorized	Section 3.1
407	Proxy Authentication Required	Section 3.2

5.3. Header Field Registration

The Message Header Field Registry located at <http://www.iana.org/assignments/message-headers/message-header-index.html> shall be updated with the permanent registrations below (see [RFC3864]):

Header Field Name	Protocol	Status	Reference
Authorization	http	standard	Section 4.1
Proxy-Authenticate	http	standard	Section 4.2
Proxy-Authorization	http	standard	Section 4.3
WWW-Authenticate	http	standard	Section 4.4

The change controller is: "IETF (iesg@ietf.org) - Internet Engineering Task Force".

6. Security Considerations

This section is meant to inform application developers, information providers, and users of the security limitations in HTTP/1.1 as described by this document. The discussion does not include definitive solutions to the problems revealed, though it does make some suggestions for reducing security risks.

6.1. Authentication Credentials and Idle Clients

Existing HTTP clients and user agents typically retain authentication information indefinitely. HTTP/1.1 does not provide a method for a server to direct clients to discard these cached credentials. This is a significant defect that requires further extensions to HTTP. Circumstances under which credential caching can interfere with the application's security model include but are not limited to:

- Clients which have been idle for an extended period following which the server might wish to cause the client to reprompt the user for credentials.

- Applications which include a session termination indication (such as a "logout" or "commit" button on a page) after which the server side of the application "knows" that there is no further reason for the client to retain the credentials.

This is currently under separate study. There are a number of work-arounds to parts of this problem, and we encourage the use of password protection in screen savers, idle time-outs, and other methods which mitigate the security problems inherent in this problem. In particular, user agents which cache credentials are encouraged to provide a readily accessible mechanism for discarding cached credentials under user control.

6.2. Protection Spaces

Authentication schemes that solely rely on the "realm" mechanism for establishing a protection space will expose credentials to all resources on a server. Clients that have successfully made authenticated requests with a resource can use the same authentication credentials for other resources on the same server. This makes it possible for a different resource to harvest authentication credentials for other resources.

This is of particular concern when a server hosts resources for multiple parties under the same canonical root URI (Section 2.2). Possible mitigation strategies include restricting direct access to authentication credentials (i.e., not making the content of the Authorization request header field available), and separating protection spaces by using a different host name for each party.

7. Acknowledgments

This specification takes over the definition of the HTTP Authentication Framework, previously defined in RFC 2617. We thank John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart for their work on that specification. See Section 6 of [RFC2617] for further acknowledgements.

See Section 9 of [Part1] for the Acknowledgments related to this document revision.

8. References

8.1. Normative References

[Part1] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "HTTP/1.1, part 1: Message Routing and Syntax"", draft-ietf-httpbis-p1-messaging-20 (work in progress), July 2012.

[Part2] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "HTTP/1.1, part 2: Semantics and Payloads", draft-ietf-httpbis-p2-semantics-20 (work in progress), July 2012.

[Part6] Fielding, R., Ed., Lafon, Y., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1, part 6: Caching", draft-ietf-httpbis-p6-cache-20 (work in progress), July 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

8.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Appendix A. Changes from RFCs 2616 and 2617

The "realm" parameter isn't required anymore in general; consequently, the ABNF allows challenges without any auth parameters. (Section 2)

The "b64token" alternative to auth-param lists has been added for consistency with legacy authentication schemes such as "Basic". (Section 2)

Introduce Authentication Scheme Registry. (Section 2.3)

Change ABNF productions for header fields to only define the field value. (Section 4)

Appendix B. Imported ABNF

The following core rules are included by reference, as defined in Appendix B.1 of [RFC5234]: ALPHA (letters), CR (carriage return), CRLF (CR LF), CTL (controls), DIGIT (decimal 0-9), DQUOTE (double quote), HEXDIG (hexadecimal 0-9/A-F/a-f), LF (line feed), OCTET (any 8-bit sequence of data), SP (space), and VCHAR (any visible US-ASCII character).

The rules below are defined in [Part1]:

```
  BWS           = <BWS, defined in [Part1], Section 3.2.1>
  OWS           = <OWS, defined in [Part1], Section 3.2.1>
  quoted-string = <quoted-string, defined in [Part1], Section 3.2.4>
  token         = <token, defined in [Part1], Section 3.2.4>
```

Appendix C. Collected ABNF

```
   Authorization = credentials

   BWS = <BWS, defined in [Part1], Section 3.2.1>

   OWS = <OWS, defined in [Part1], Section 3.2.1>

   Proxy-Authenticate = *( "," OWS ) challenge *( OWS "," [ OWS
    challenge ] )
   Proxy-Authorization = credentials

   WWW-Authenticate = *( "," OWS ) challenge *( OWS "," [ OWS challenge
    ] )

   auth-param = token BWS "=" BWS ( token / quoted-string )
   auth-scheme = token

   b64token = 1*( ALPHA / DIGIT / "-" / "." / "_" / "~" / "+" / "/" )
    *"="

   challenge = auth-scheme [ 1*SP ( b64token / [ ( "," / auth-param ) *(
    OWS "," [ OWS auth-param ] ) ] ) ]
   credentials = auth-scheme [ 1*SP ( b64token / [ ( "," / auth-param )
    *( OWS "," [ OWS auth-param ] ) ] ) ]

   quoted-string = <quoted-string, defined in [Part1], Section 3.2.4>

   token = <token, defined in [Part1], Section 3.2.4>
```

Appendix D. Change Log (to be removed by RFC Editor before publication)

Changes up to the first Working Group Last Call draft are summarized in <http://trac.tools.ietf.org/html/draft-ietf-httpbis-p7-auth-19#appendix-C>.

D.1. Since draft-ietf-httpbis-p7-auth-19

Closed issues:

- <http://tools.ietf.org/wg/httpbis/trac/ticket/348>: "Realms and scope"

- <http://tools.ietf.org/wg/httpbis/trac/ticket/349>: "Strength"

- <http://tools.ietf.org/wg/httpbis/trac/ticket/357>: "Authentication exchanges"

- <http://tools.ietf.org/wg/httpbis/trac/ticket/361>: "ABNF requirements for recipients"

- <http://tools.ietf.org/wg/httpbis/trac/ticket/368>: "note introduction of new IANA registries as normative changes"

Index

Authors' Addresses

Roy T. Fielding (editor)
Adobe Systems Incorporated
345 Park Ave
San Jose, CA 95110
USA

EMail: fielding@gbiv.com
URI: http://roy.gbiv.com/

Yves Lafon (editor)
World Wide Web Consortium
W3C / ERCIM
2004, rte des Lucioles
Sophia-Antipolis, AM 06902
France

EMail: ylafon@w3.org
URI: http://www.raubacapeu.net/people/yves/

Julian F. Reschke (editor)
greenbytes GmbH
Hafenweg 16
Muenster, NW 48155
Germany

EMail: julian.reschke@greenbytes.de
URI: http://greenbytes.de/tech/webdav/