Internet Engineering Task Force (IETF)
Request for Comments: 7617
Obsoletes: 2617
Category: Standards Track
ISSN: 2070-1721

J. Reschke
greenbytes

September 2015

The 'Basic' HTTP Authentication Scheme

Abstract

This document defines the "Basic" Hypertext Transfer Protocol (HTTP) authentication scheme, which transmits credentials as user-id/password pairs, encoded using Base64.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7617.

Table of Contents

1. Introduction

This document defines the "Basic" Hypertext Transfer Protocol (HTTP) authentication scheme, which transmits credentials as user-id/password pairs, encoded using Base64 (HTTP authentication schemes are defined in [RFC7235]).

This scheme is not considered to be a secure method of user authentication unless used in conjunction with some external secure system such as TLS (Transport Layer Security, [RFC5246]), as the user-id and password are passed over the network as cleartext.

The "Basic" scheme previously was defined in Section 2 of [RFC2617]. This document updates the definition, and also addresses internationalization issues by introducing the 'charset' authentication parameter (Section 2.1).

Other documents updating RFC 2617 are "Hypertext Transfer Protocol (HTTP/1.1): Authentication" ([RFC7235], defining the authentication framework), "HTTP Digest Access Authentication" ([RFC7616], updating the definition of the "Digest" authentication scheme), and "HTTP Authentication-Info and Proxy-Authentication-Info Response Header Fields" ([RFC7615]). Taken together, these four documents obsolete RFC 2617.

1.1. Terminology and Notation

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].

The terms "protection space" and "realm" are defined in Section 2.2 of [RFC7235].

The terms "(character) repertoire" and "character encoding scheme" are defined in Section 2 of [RFC6365].

2. The 'Basic' Authentication Scheme

The Basic authentication scheme is based on the model that the client needs to authenticate itself with a user-id and a password for each protection space ("realm"). The realm value is a free-form string that can only be compared for equality with other realms on that server. The server will service the request only if it can validate the user-id and password for the protection space applying to the requested resource.

The Basic authentication scheme utilizes the Authentication Framework as follows.

In challenges:

- The scheme name is "Basic".
- The authentication parameter 'realm' is REQUIRED ([RFC7235], Section 2.2).
- The authentication parameter 'charset' is OPTIONAL (see Section 2.1).
- No other authentication parameters are defined -- unknown parameters MUST be ignored by recipients, and new parameters can only be defined by revising this specification.

See also Section 4.1 of [RFC7235], which discusses the complexity of parsing challenges properly.

Note that both scheme and parameter names are matched case-insensitively.

For credentials, the "token68" syntax defined in Section 2.1 of [RFC7235] is used. The value is computed based on user-id and password as defined below.

Upon receipt of a request for a URI within the protection space that lacks credentials, the server can reply with a challenge using the 401 (Unauthorized) status code ([RFC7235], Section 3.1) and the WWW-Authenticate header field ([RFC7235], Section 4.1).

For instance:

```
HTTP/1.1 401 Unauthorized
Date: Mon, 04 Feb 2014 16:50:53 GMT
WWW-Authenticate: Basic realm="WallyWorld"
```

where "WallyWorld" is the string assigned by the server to identify the protection space.

A proxy can respond with a similar challenge using the 407 (Proxy Authentication Required) status code ([RFC7235], Section 3.2) and the Proxy-Authenticate header field ([RFC7235], Section 4.3).

To receive authorization, the client

1. obtains the user-id and password from the user,

2. constructs the user-pass by concatenating the user-id, a single colon (":") character, and the password,

3. encodes the user-pass into an octet sequence (see below for a discussion of character encoding schemes),

4. and obtains the basic-credentials by encoding this octet sequence using Base64 ([RFC4648], Section 4) into a sequence of US-ASCII characters ([RFC0020]).

The original definition of this authentication scheme failed to specify the character encoding scheme used to convert the user-pass into an octet sequence. In practice, most implementations chose either a locale-specific encoding such as ISO-8859-1 ([ISO-8859-1]), or UTF-8 ([RFC3629]). For backwards compatibility reasons, this specification continues to leave the default encoding undefined, as long as it is compatible with US-ASCII (mapping any US-ASCII character to a single octet matching the US-ASCII character code).

The user-id and password MUST NOT contain any control characters (see "CTL" in Appendix B.1 of [RFC5234]).

Furthermore, a user-id containing a colon character is invalid, as the first colon in a user-pass string separates user-id and password from one another; text after the first colon is part of the password. User-ids containing colons cannot be encoded in user-pass strings.

Note that many user agents produce user-pass strings without checking that user-ids supplied by users do not contain colons; recipients will then treat part of the username input as part of the password.

If the user agent wishes to send the user-id "Aladdin" and password "open sesame", it would use the following header field:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

2.1. The 'charset' auth-param

In challenges, servers can use the 'charset' authentication parameter to indicate the character encoding scheme they expect the user agent to use when generating "user-pass" (a sequence of octets). This information is purely advisory.

The only allowed value is "UTF-8"; it is to be matched case-insensitively (see [RFC2978], Section 2.3). It indicates that the server expects character data to be converted to Unicode Normalization Form C ("NFC"; see Section 3 of [RFC5198]) and to be encoded into octets using the UTF-8 character encoding scheme ([RFC3629]).

For the user-id, recipients MUST support all characters defined in the "UsernameCasePreserved" profile defined in Section 3.3 of [RFC7613], with the exception of the colon (":") character.

For the password, recipients MUST support all characters defined in the "OpaqueString" profile defined in Section 4.2 of [RFC7613].

Other values are reserved for future use.

> Note: The 'charset' is only defined on challenges, as Basic authentication uses a single token for credentials ('token68' syntax); thus, the credentials syntax isn't extensible.
>
> Note: The name 'charset' has been chosen for consistency with Section 2.1.1 of [RFC2831]. A better name would have been 'accept-charset', as it is not about the message it appears in, but the server's expectation.

In the example below, the server prompts for authentication in the "foo" realm, using Basic authentication, with a preference for the UTF-8 character encoding scheme:

```
WWW-Authenticate: Basic realm="foo", charset="UTF-8"
```

Note that the parameter value can be either a token or a quoted string; in this case, the server chose to use the quoted-string notation.

The user's name is "test", and the password is the string "123" followed by the Unicode character U+00A3 (POUND SIGN). Using the character encoding scheme UTF-8, the user-pass becomes:

```
't' 'e' 's' 't' ':' '1' '2' '3' pound
74  65  73  74  3A  31  32  33  C2  A3
```

Encoding this octet sequence in Base64 ([RFC4648], Section 4) yields:

```
dGVzdDoxMjPCow==
```

Thus, the Authorization header field would be:

```
Authorization: Basic dGVzdDoxMjPCow==
```

Or, for proxy authentication:

```
Proxy-Authorization: Basic dGVzdDoxMjPCow==
```

2.2. Reusing Credentials

Given the absolute URI ([RFC3986], Section 4.3) of an authenticated request, the authentication scope of that request is obtained by removing all characters after the last slash ("/") character of the path component ("hier_part"; see [RFC3986], Section 3). A client SHOULD assume that resources identified by URIs with a prefix-match of the authentication scope are also within the protection space specified by the realm value of that authenticated request.

A client MAY preemptively send the corresponding Authorization header field with requests for resources in that space without receipt of another challenge from the server. Similarly, when a client sends a request to a proxy, it MAY reuse a user-id and password in the Proxy-Authorization header field without receiving another challenge from the proxy server.

For example, given an authenticated request to:

```
http://example.com/docs/index.html
```

requests to the URIs below could use the known credentials:

```
http://example.com/docs/
http://example.com/docs/test.doc
http://example.com/docs/?page=1
```

while the URIs

```
http://example.com/other/
https://example.com/docs/
```

would be considered to be outside the authentication scope.

Note that a URI can be part of multiple authentication scopes (such as "http://example.com/" and "http://example.com/docs/"). This specification does not define which of these should be treated with higher priority.

3. Internationalization Considerations

User-ids or passwords containing characters outside the US-ASCII character repertoire will cause interoperability issues, unless both communication partners agree on what character encoding scheme is to be used. Servers can use the new 'charset' parameter (Section 2.1) to indicate a preference of "UTF-8", increasing the probability that clients will switch to that encoding.

The 'realm' parameter carries data that can be considered textual; however, [RFC7235] does not define a way to reliably transport non-US-ASCII characters. This is a known issue that would need to be addressed in a revision to that specification.

4. Security Considerations

The Basic authentication scheme is not a secure method of user authentication, nor does it in any way protect the entity, which is transmitted in cleartext across the physical network used as the carrier. HTTP does not prevent the addition of enhancements (such as schemes to use one-time passwords) to Basic authentication.

The most serious flaw of Basic authentication is that it results in the cleartext transmission of the user's password over the physical network. Many other authentication schemes address this problem.

Because Basic authentication involves the cleartext transmission of passwords, it SHOULD NOT be used (without enhancements such as HTTPS [RFC2818]) to protect sensitive or valuable information.

A common use of Basic authentication is for identification purposes -- requiring the user to provide a user-id and password as a means of identification, for example, for purposes of gathering accurate usage statistics on a server. When used in this way it is tempting to think that there is no danger in its use if illicit access to the protected documents is not a major concern. This is only correct if the server issues both user-id and password to the users and, in particular, does not allow the user to choose his or her own password. The danger arises because naive users frequently reuse a single password to avoid the task of maintaining multiple passwords.

If a server permits users to select their own passwords, then the threat is not only unauthorized access to documents on the server but also unauthorized access to any other resources on other systems that the user protects with the same password. Furthermore, in the server's password database, many of the passwords may also be users' passwords for other sites. The owner or administrator of such a system could therefore expose all users of the system to the risk of

unauthorized access to all those other sites if this information is not maintained in a secure fashion. This raises both security and privacy concerns ([RFC6973]). If the same user-id and password combination is in use to access other accounts, such as an email or health portal account, personal information could be exposed.

Basic authentication is also vulnerable to spoofing by counterfeit servers. If a user can be led to believe that she is connecting to a host containing information protected by Basic authentication when, in fact, she is connecting to a hostile server or gateway, then the attacker can request a password, store it for later use, and feign an error. Server implementers ought to guard against this sort of counterfeiting; in particular, software components that can take over control over the message framing on an existing connection need to be used carefully or not at all (for instance: NPH ("Non-Parsed Header") scripts as described in Section 5 of [RFC3875]).

Servers and proxies implementing Basic authentication need to store user passwords in some form in order to authenticate a request. These passwords ought to be stored in such a way that a leak of the password data doesn't make them trivially recoverable. This is especially important when users are allowed to set their own passwords, since users are known to choose weak passwords and to reuse them across authentication realms. While a full discussion of good password hashing techniques is beyond the scope of this document, server operators ought to make an effort to minimize risks to their users in the event of a password data leak. For example, servers ought to avoid storing user passwords in plaintext or as unsalted digests. For more discussion about modern password hashing techniques, see the "Password Hashing Competition" (<https://password-hashing.net>).

The use of the UTF-8 character encoding scheme and of normalization introduces additional security considerations; see Section 10 of [RFC3629] and Section 6 of [RFC5198] for more information.

5. IANA Considerations

IANA maintains the "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" ([RFC7235]) at <http://www.iana.org/assignments/http-authschemes>.

The entry for the "Basic" authentication scheme has been updated to reference this specification.

6. References

6.1. Normative References

[RFC20] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2978] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, DOI 10.17487/RFC2978, October 2000, <http://www.rfc-editor.org/info/rfc2978>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <http://www.rfc-editor.org/info/rfc5198>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <http://www.rfc-editor.org/info/rfc6365>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

[RFC7613] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 7613, DOI 10.17487/RFC7613, August 2015, <http://www.rfc-editor.org/info/rfc7613>.

6.2. Informative References

[ISO-8859-1] International Organization for Standardization, "Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1", ISO/IEC 8859-1:1998, 1998.

[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, DOI 10.17487/RFC2617, June 1999, <http://www.rfc-editor.org/info/rfc2617>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, DOI 10.17487/RFC2831, May 2000, <http://www.rfc-editor.org/info/rfc2831>.

[RFC3875] Robinson, D. and K. Coar, "The Common Gateway Interface (CGI) Version 1.1", RFC 3875, DOI 10.17487/RFC3875, October 2004, <http://www.rfc-editor.org/info/rfc3875>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7615] Reschke, J., "HTTP Authentication-Info and Proxy-Authentication-Info Response Header Fields", RFC 7615, DOI 10.17487/RFC7615, September 2015, <http://www.rfc-editor.org/info/rfc7615>.

[RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <http://www.rfc-editor.org/info/rfc7616>.

Appendix A. Changes from RFC 2617

The scheme definition has been rewritten to be consistent with newer specifications such as [RFC7235].

The new authentication parameter 'charset' has been added. It is purely advisory, so existing implementations do not need to change, unless they want to take advantage of the additional information that previously wasn't available.

Appendix B. Deployment Considerations for the 'charset' Parameter

B.1. User Agents

User agents not implementing 'charset' will continue to work as before, ignoring the new parameter.

User agents that already default to the UTF-8 encoding implement 'charset' by definition.

Other user agents can keep their default behavior and switch to UTF-8 when seeing the new parameter.

B.2. Servers

Servers that do not support non-US-ASCII characters in credentials do not require any changes to support 'charset'.

Servers that need to support non-US-ASCII characters, but cannot use the UTF-8 character encoding scheme will not be affected; they will continue to function as well or as badly as before.

Finally, servers that need to support non-US-ASCII characters and can use the UTF-8 character encoding scheme can opt in by specifying the 'charset' parameter in the authentication challenge. Clients that do understand the 'charset' parameter will then start to use UTF-8, while other clients will continue to send credentials in their default encoding, broken credentials, or no credentials at all. Until all clients are upgraded to support UTF-8, servers are likely to see both UTF-8 and "legacy" encodings in requests. When processing as UTF-8 fails (due to a failure to decode as UTF-8 or a mismatch of user-id/password), a server might try a fallback to the previously supported legacy encoding in order to accommodate these legacy clients. Note that implicit retries need to be done carefully; for instance, some subsystems might detect repeated login failures and treat them as a potential credentials-guessing attack.

B.3. Why not simply switch the default encoding to UTF-8?

There are sites in use today that default to a local character encoding scheme, such as ISO-8859-1 ([ISO-8859-1]), and expect user agents to use that encoding. Authentication on these sites will stop working if the user agent switches to a different encoding, such as UTF-8.

Note that sites might even inspect the User-Agent header field ([RFC7231], Section 5.5.3) to decide which character encoding scheme to expect from the client. Therefore, they might support UTF-8 for some user agents, but default to something else for others. User agents in the latter group will have to continue to do what they do today until the majority of these servers have been upgraded to always use UTF-8.

Acknowledgements

This specification takes over the definition of the "Basic" HTTP Authentication Scheme, 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, from which significant amounts of text were borrowed. See Section 6 of [RFC2617] for further acknowledgements.

The internationalization problem with respect to the character encoding scheme used for user-pass was reported as a Mozilla bug back in the year 2000 (see <https://bugzilla.mozilla.org/show_bug.cgi?id=41489> and also the more recent <https://bugzilla.mozilla.org/show_bug.cgi?id=656213>). It was Andrew Clover's idea to address it using a new auth-param.

We also thank the members of the HTTPAUTH Working Group and other reviewers, namely, Stephen Farrell, Roy Fielding, Daniel Kahn Gillmor, Tony Hansen, Bjoern Hoehrmann, Kari Hurtta, Amos Jeffries, Benjamin Kaduk, Michael Koeller, Eric Lawrence, Barry Leiba, James Manger, Alexey Melnikov, Kathleen Moriarty, Juergen Schoenwaelder, Yaron Sheffer, Meral Shirazipour, Michael Sweet, and Martin Thomson for feedback on this revision.

Author's Address

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

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