RFC 7235

Hypertext Transfer Protocol (HTTP/1.1): Authentication

문서 최종 수정일	2020-05-26
원문 복사일	2020-05-11
번역 및 정리	이병록(roka88)
이메일	roka88.dev@gmail.com

PROPOSED STANDARD

Internet Engineering Task Force (IETF)
Request for Comments: 7235
Obsoletes: 2616
Updates: 2617
Category: Standards Track
ISSN: 2070-1721

R. Fielding, Ed.
Adobe
J. Reschke, Ed.
greenbytes
June 2014

Hypertext Transfer Protocol (HTTP/1.1): Authentication

Abstract

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

하이퍼텍스트 전송 프로토콜(HTTP)은 분산, 협업, 하이퍼미디어 정보 시스템을 위한 상태 비저장 애플리케이션-레벨 프로토콜이다. 이 문서는 HTTP 인증 프레임워크를 정의한다.

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.

이 문서는 Internet Engineering Task Force(IETF)의 제품이다. 문서는 IETF 공동체의 합의를 나타낸다. 문서는 공개 검토를 받아왔으며 Internet Engineering Starting Group (IESG)에 의해 발행 승인를 받았다. 인터넷 표준의 추가 정보는 RFC 5741 Section 2에서 확인할 수 있다.

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/rfc7235.

이 문서에 대한 현재 상태 정보는 정오표와 피드백을 어떻게 제공하는 방법은 http://www.rfc-editor.org/info/rfc7235 에서 얻을 수 있다.

Copyright Notice

Table of Contents

1. Introduction

HTTP provides a general framework for access control and authentication, via an extensible set of challenge-response authentication schemes, which can be used by a server to challenge a client request and by a client to provide authentication information. This document defines HTTP/1.1 authentication in terms of the architecture defined in "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing" [RFC7230], including the general framework previously described in "HTTP Authentication: Basic and Digest Access Authentication" [RFC2617] and the related fields and status codes previously defined in "Hypertext Transfer Protocol — HTTP/1.1" [RFC2616].

HTTP는 확장 가능한 일련의 challenge-response 인증 scheme을 통해 접근 제어 및 인증에 대한 일반적인 프레임워크를 제공하며, 이 프레임워크는 클라이언트 요청을 요구하기 위해 서버가 사용할 수 있고 클라이언트가 인증 정보를 제공할 수 있다. 본 문서는 “Hypertext Transfer Protocol (HTTP/1.1) : Message Synthetic and Routing” [RFC7230] 에서 아키텍처 측면에서 HTTP/1.1 인증을 정의하고, 이전에 정의된 "HTTP Authentication: Basic and Digest Access Authentication" [RFC2617]에서 일반적인 프레임워크와, 이전에 정의된 "Hypertext Transfer Protocol — HTTP/1.1" [RFC2616]에서 관련된 필드 및 상태 코드를 포함한다.

The IANA Authentication Scheme Registry (Section 5.1) lists registered authentication schemes and their corresponding specifications, including the "basic" and "digest" authentication schemes previously defined by RFC 2617.

IANA Authentication Scheme Registry(Section 5.1)에는 RFC 2617에서 이전에 정의한 "basic" 및 "digest" 인증 scheme을 포함하여 등록된 인증 scheme과 해당 명세가 열거되어 있다.

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].

Conformance criteria and considerations regarding error handling are defined in Section 2.5 of [RFC7230].

1.2. Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with a list extension, defined in Section 7 of [RFC7230], that allows for compact definition of comma-separated lists using a '#' operator (similar to how the '*' operator indicates repetition). Appendix B describes rules imported from other documents. Appendix C shows the collected grammar with all list operators expanded to standard ABNF notation.

2. Access Authentication Framework

2.1. Challenge and Response

HTTP provides a simple challenge-response authentication framework that can be used by a server to challenge a client request and by a client to provide

authentication information. It uses a case-insensitive token as a means to identify the authentication scheme, followed by additional information necessary for achieving authentication via that scheme. The latter can be either a comma-separated list of parameters or a single sequence of characters capable of holding base64-encoded information.

HTTP는 서버에 의해 클라이언트 요청을 challenge(이하 질의)하고 클라이언트가 인증 정보를 제공하는 데 사용할 수 있는 간단한 challenge-response(질의-응답) 인증 프레임워크를 제공한다. 대소문자를 구분하지 않는 토큰으로 인증 방식을 식별하는 수단으로 사용하며, 그 scheme을 통한 인증 획득에 필요한 추가 정보가 뒤따른다. 후자는 쉼표로 구분된 매개변수 목록 또는 base64로 인코딩된 정보를 저장할 수 있는 단일 문자 시퀀스가 될 수 있다.

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

인증 매개변수는 name=value 쌍이며, 여기서 이름 토큰은 대소문자를 구분하지 않고 일치하며, 각 매개변수 이름은 질의 당 한 번만 발생해야 한다.(MUST)

```
auth-scheme    = token

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

token68        = 1*( ALPHA / DIGIT /
                     "-" / "." / "_" / "~" / "+" / "/" ) *"="
```

The token68 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]).

token68 구문은 66 예약되지 않은 URI 문자([RFC3986])와, 몇 개를 추가하여, 패딩이 있든 없든 base64, base64url(URL 및 파일 이름 안전 알파벳), base32 또는 base16 (hex) 인코딩을 포함할 수 있지만 공백은 제외할 수 있다.([RFC4648])

A 401 (Unauthorized) response message is used by an origin server to challenge the authorization of a user agent, including a WWW-Authenticate header field containing at least one challenge applicable to the requested resource.

401 (Unauthorized) 응답 메시지는 요청된 리소스에 적용할 수 있는 적어도 하나 이상의 질의가 포함된 WWW-Authenticate 헤더 필드를 포함하여 사용자 에이전트의 권한을 질의하기 위해 원서버에서 사용된다.

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

407 (Proxy Authentication Required) 응답 메시지는 요청된 리소스에 대해 프락시에 사용되는 적어도 하나 이상의 질의를 가지는 Proxy-Authenticate 헤더 필드를 포함하여 클라이언트의 인증을 질의하기 위해 프락시가 사용한다.

challenge = auth-scheme [1*SP (token68 / #auth-param)]

Note: Many clients fail to parse a challenge that contains an unknown scheme. A workaround for this problem is to list well-supported schemes (such as "basic") first.

참고: 많은 클라이언트가 알 수 없는 scheme을 포함하는 질의를 구문 분석하지 못한다. 이 문제에 대한 해결책은 잘 지원되는 scheme(예: "basic")을 먼저 나열하는 것이다.

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.

원서버로 인증하려는 사용자 에이전트 -- 일반적으로, 반드시 그런것은 아니지만, 401 (Unauthorized) 을 받은 후 -- 요청에 Authorization 헤더 필드를 포함하면 인증할 수 있다.

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.

프락시로 인증하려는 클라이언트 -- 일반적으로, 반드시 그런것은 아니지만, 407 (Proxy Authentication Required)을 받은 후 -- 요청에 Proxy-Authorization 헤더 필드를 포함하면 인증할 수 있다.

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 in a response (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. Transmission of credentials within header field values implies significant security considerations regarding the confidentiality of the underlying connection, as described in Section 6.1.

Authorization 필드 값과 Proxy-Authorization 필드 값 모두 응답에서 수신된 질의를 기반으로 요청되는 리소스의 영역에 대한 클라이언트의 credentials(이하 자격 증명)을 포함한다(과거 어느 시점일 가능성이 있음). 사용자 에이전트는 자신의 값을 생성할 때, 가장 안전한 auth-scheme로 간주되는 질의를 선택하여 해당 사용자로부터 자격 증명을 얻어야 한다. 헤더 필드 값 내의 자격 증명의 전송은 Section 6.1에 기술된 바와 같이 기본 커넥션의 기밀성과 관련하여 중요한 보안 고려사항을 암시한다.

```
credentials = auth-scheme [ 1*SP ( token68 / #auth-param ) ]
```

Upon receipt of 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 send a 401 (Unauthorized) response that contains a WWW-Authenticate header field with at least one (possibly new) challenge applicable to the requested resource.

유효하지 않은 자격 증명(e.g., 잘못된 암호) 또는 부분 자격 증명(e.g., 인증 구성에서 둘 이상의 라운드 트립이 필요한 경우)을 포함하는, 자격 증명을 생략하는 보호된 리소스에 대한 요청을 수신하면, 원서버는 요청된 리소스에 적용할 수 있는 적어도 하나 이상의(아마 새로운) 질의와 WWW-Authenticate 헤더 필드를 포함하는 401(Unauthorized) 응답을 전송해야 한다.(SHOULD)

Likewise, upon receipt of a request that omits proxy credentials or contains invalid or partial proxy credentials, a proxy that requires authentication SHOULD generate a 407 (Proxy Authentication Required) response that contains a Proxy-Authenticate header field with at least one (possibly new) challenge applicable to the proxy.

마찬가지로, 프락시 자격 증명을 생략하거나 유효하지 않거나 일부 프락시 자격 증명을 포함하는 요청을 수신하면, 인증을 필요로 하는 프락시는 프락시에 적용할 수 있는 적어도 하나 이상의(아마 새로운) 질의와 Proxy-Authenticate 헤더 필드를 포함하는 407(Proxy Authentication Required) 응답을 생성해야 한다.

A server that receives valid credentials that are not adequate to gain access ought to respond with the 403 (Forbidden) status code (Section 6.5.3 of [RFC7231]).

액세스 권한이 충분하지 않은 유효한 자격 증명을 받은 서버는 403(Forbidden) 상태 코드([RFC7231]의 Section 6.5.3)로 응답해야 한다.

HTTP does not restrict applications to this simple challenge-response framework for access authentication. Additional mechanisms can be used, such as authentication 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.

HTTP는 접근 인증을 위해 애플리케이션을 이 간단한 challenge-response 프레임워크로 제한하지 않는다. 전송 레벨 또는 메시지 캡슐화를 통한 인증, 인증 정보를 지정하는 추가 헤더 필드와 같은 추가 메커니즘을 사용할 수 있다. 그러나 이러한 추가 메커니즘은 이 명세에 의해 정의되지 않는다.

2.2. Protection Space (Realm)

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

"realm" 인증 매개변수는 보호 범위를 표시하고자 하는 인증 scheme에서 사용하도록 예약되어 있다.

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 [RFC7230]) 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, that can have

additional semantics specific to the authentication scheme. Note that a response can have multiple challenges with the same auth-scheme but with different realms.

보호 공간은 접속 중인 서버의 표준 루트 URI(유효한 요청 URI의 scheme 및 권한 구성 요소, [RFC7230]의 Section 5.5 참조)에 의해 영역 값과 함께 정의된다. 이러한 영역은 서버의 보호된 리소스를 보호 공간의 집합으로 분할할 수 있도록 허용하며, 각 영역은 자체 인증 scheme 및/또는 인증 데이터베이스를 가지고 있다. 영역 값은 일반적으로 원서버에 의해 할당되는 문자열로, 인증 sheme에 특정한 추가적인 의미론을 가질 수 있다. 응답은 동일한 auth-scheme을 사용하지만, 다른 영역으로 인해 여러 가지 문제를 일으킬 수 있다는 점에 유의한다.

The protection space determines the domain over which credentials can be automatically applied. If a prior request has been authorized, the user agent MAY reuse the same credentials for all other requests within that protection space for a period of time determined by the authentication scheme, parameters, and/or user preferences (such as a configurable inactivity timeout). Unless specifically allowed by the authentication scheme, a single protection space cannot extend outside the scope of its server.

보호 공간은 자격 증명을 자동으로 적용할 수 있는 도메인을 결정한다. 이전 요청이 인증된 경우 사용자 에이전트는 인증 scheme, 매개 변수 및/또는 사용자 기본 설정(구성 가능한 비활성 시간 초과 등)에 의해 결정된 기간 동안 해당 보호 공간 내의 다른 모든 요청에 대해 동일한 자격 증명을 재사용할 수 있다.(MAY) 인증 scheme에 의해 특별히 허용된 경우를 제외하고, 단일 보호 공간은 해당 서버의 범위 밖으로 확장될 수 없다.

For historical reasons, a sender MUST only generate 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.

역사적 이유로, 발신자는 quoted-string 구문만 생성해야 한다. 수신자는 오랫동안 두 가지 표기법을 모두 수용해 온 기존 클라이언트와의 상호 운용성을 극대화하기 위해 토큰과 quoted-string 구문을 모두 지원해야 할 수 있다.

3. Status Code Definitions

3.1. 401 Unauthorized

The 401 (Unauthorized) status code indicates that the request has not been applied because it lacks valid authentication credentials for the target resource. The server generating a 401 response MUST send a WWW-Authenticate header field (Section 4.1) containing at least one challenge applicable to the target resource.

401 (Unauthorized) 상태 코드는 대상 리소스에 대한 유효한 인증 자격 증명이 없기 때문에 요청이 적용되지 않았음을 나타낸다. 401 응답을 생성하는 서버는 대상 리소스에 적용되는 적어도 하나 이상의 challenge를 포함하는 WWW-Authenticate 헤더 필드(Section 4.1)를 전송해야 한다.(MUST)

If the request included authentication credentials, then the 401 response indicates that authorization has been refused for those credentials. The user agent MAY repeat the request with a new or replaced Authorization header field (Section 4.2). 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 agent SHOULD present the enclosed representation to the user, since it usually contains relevant diagnostic information.

요청에 인증 자격 증명이 포함된 경우, 401 응답은 해당 자격 증명에 대한 인증이 거부되었음을 나타낸다. 사용자 에이전트는 새 또는 대체된 Authorization 헤더 필드를 사용하여 요청을 반복할 수 있다.(MAY) (Section 4.2) 401 응답에 이전 응답과 동일한 challenge가 포함되고 사용자 에이전트가 이미 한 번 인증을 시도한 경우, 사용자 에이전트는 일반적으로 관련 진단 정보를 포함하므로 사용자에게 동봉된 표현을 제시해야 한다.(SHOULD)

3.2. 407 Proxy Authentication Required

The 407 (Proxy Authentication Required) status code is similar to 401 (Unauthorized), but it indicates that the client needs to authenticate itself in order to use a proxy. The proxy MUST send a Proxy-Authenticate header field (Section 4.3) containing a challenge applicable to that proxy for the target resource. The client MAY repeat the request with a new or replaced Proxy-Authorization header field (Section 4.4).

407 (Proxy Authentication Required) 상태 코드는 401 (Unauthorized)과 유사하지만, 프락시를 이용하기 위해서는 클라이언트가 스스로 인증해야 한다는 것을 나타낸다. 프락시는 대상 리소스에 대해 해당 프락시에 적용되는 과제가 포함된 Proxy-Authenticate 헤더 필드

(Section 4.3)를 전송해야 한다.(MUST) 클라이언트는 새 또는 대체된 Proxy-Authorization 헤더 필드(Section 4.4)를 사용하여 요청을 반복할 수 있다.(MAY)

4. Header Field Definitions

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

이 절에서는 HTTP 인증 프레임워크와 관련된 헤더 필드의 구문 및 의미론을 정의한다.

4.1. WWW-Authenticate

The "WWW-Authenticate" header field indicates the authentication scheme(s) and parameters applicable to the target resource.

"WWW-Authenticate" 헤더 필드는 대상 리소스에 적용할 수 있는 인증 scheme(s) 및 매개변수를 나타낸다.

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

A server generating a 401 (Unauthorized) response MUST send a WWW-Authenticate header field containing at least one challenge. A server MAY generate a WWW-Authenticate header field in other response messages to indicate that supplying credentials (or different credentials) might affect the response.

401 (Unauthorized) 응답을 생성하는 서버는 적어도 하나의 challenge가 포함된 WWW-Authenticate 헤더 필드를 전송해야 한다.(MUST) 서버는 다른 응답 메시지에서 WWW-Authenticate 헤더 필드를 생성하여 자격 증명(또는 다른 자격 증명)을 제공하는 것이 응답에 영향을 미칠 수 있음을 나타낼 수 있다.(MAY)

A proxy forwarding a response MUST NOT modify any WWW-Authenticate fields in that response.

응답을 전달하는 프락시는 해당 응답의 WWW-Authenticate 필드를 수정해서는 안 된다.(MUST NOT)

User agents are advised to take special care in parsing the field value, as it might contain more than one challenge, and each challenge can contain a comma-separated list of authentication parameters. Furthermore, the header field itself can occur multiple times.

사용자 에이전트는 필드 값을 구문 분석할 때 특히 주의할 것을 권고한다. 필드 값은 둘 이상의 질의를 포함할 수 있으며 각 질의에는 쉼표로 구분된 인증 매개 변수 목록이 포함될 수 있다. 게다가, 헤더 필드 자체는 여러 번 발생할 수 있다.

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".

이 헤더 필드에는 영역 값이 "apps"인 "Newauth" scheme과 영역 값이 "simple"인 추가 매개 변수 "type" 및 "title"인 "Basic" scheme인 다른 두 가지 질의가 포함되어 있다.

Note: The challenge grammar production uses the list syntax as well. Therefore, a sequence of comma, whitespace, and comma can be considered either 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.

참고: 질의 문법 제작은 목록 구문도 사용한다. 따라서 쉼표, 공백, 쉼표의 순서는 앞의 질의에 적용하거나, 질의 목록의 빈 항목으로 간주할 수 있다. 실제로 이러한 모호성은 헤더 필드 값의 의미론에 영향을 미치지 않으므로 무해하다.

4.2. Authorization

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

"Authorization" 헤더 필드는 사용자 에이전트가 401 (Unauthorized) 응답을 받은 후, 원서버로 자신을 인증할 수 있도록 허용하지만 -- 일반적으로, 반드시 그런것은 아니지만, 이 값은 요청된 리소스 영역에 대한 사용자 에이전트의 인증 정보를 포함하는 자격 증명으로 구성된다.

```
Authorization = credentials
```

If a request is authenticated and a realm specified, the same credentials are presumed to 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).

요청이 인증되고 영역이 지정된 경우, 동일한 자격 증명이 이 영역 내의 다른 모든 요청에 대해 유효한 것으로 간주된다.(인증 scheme 자체에 질의 값에 따라 달라지는 자격 증명이나 동기화된 시계 사용과 같이 달리 요구하지 않는다고 가정).

A proxy forwarding a request MUST NOT modify any Authorization fields in that request. See Section 3.2 of [RFC7234] for details of and requirements pertaining to handling of the Authorization field by HTTP caches.

요청을 전달하는 프락시는 해당 요청의 Authorization 필드를 수정해서는 안 된다.(MUST NOT) HTTP 캐시에 의한 Authorization 필드의 취급에 관한 세부사항 및 요건은 [RFC7234]의 Section 3.2를 참조한다.

4.3. 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 [RFC7230]). A proxy MUST send at least one Proxy-Authenticate header field in each 407 (Proxy Authentication Required) response that it generates.

"Proxy-Authenticate" 헤더 필드는 이 유효한 요청 URI에 대해 프락시에 적용되는 인증 scheme(s)와 파라미터를 나타내는 적어도 하나 이상의 challenge로 구성된다([RFC7230]의 Section 5.5). 프락시는 자신이 생성하는 각 407(Proxy Authentication Required) 응답에서 Proxy-Authenticate 헤더 필드를 하나 이상 전송해야 한다.(MUST)

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

Unlike WWW-Authenticate, the Proxy-Authenticate header field applies only to the next outbound client on the response chain. This is because only the client that chose a given proxy is likely to have the credentials necessary for authentication. However, when multiple proxies are used within the same administrative domain, such as office and regional caching proxies within a large corporate network, it is common for credentials to be generated by the user agent and passed through the hierarchy until consumed. Hence, in such a configuration, it will appear as if Proxy-Authenticate is being forwarded because each proxy will send the same challenge set.

WWW-Authenticate와 달리 Proxy-Authenticate 헤더 필드는 응답 체인의 다음 아웃바운드 클라이언트에만 적용된다. 주어진 프락시를 선택한 클라이언트만 인증에 필요한 자격 증명을 가질 가능성이 높기 때문이다. 그러나, 대기업 네트워크 내의 사무실 및 지역 캐싱 프락시와 같은 동일한 관리 도메인 내에서 여러 개의 프락시를 사용하는 경우, 사용자 에이전트에 의해 자격 증명이 생성되어 소비될 때까지 계층을 통과하는 것이 일반적이다. 따라서 이러한 구성에서는 각 프락시가 동일한 질의 집합을 전송하기 때문에 Proxy-Authenticate가 전달되는 것처럼 나타날 것이다.

Note that the parsing considerations for WWW-Authenticate apply to this header field as well; see Section 4.1 for details.

WWW-Authenticate에 대한 구문 분석 고려 사항이 이 헤더 필드에도 적용된다는 점에 유의한다. 자세한 내용은 Section 4.1을 참조한다.

4.4. Proxy-Authorization

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

"Proxy-Authorization" 헤더 필드는 클라이언트가 인증을 필요로 하는 프락시에 대해 자신(또는 그 사용자)을 식별할 수 있도록 한다. 이 값은 요청되는 리소스의 프락시 및/또는 영역에 대한 클라이언트의 인증 정보를 포함하는 자격 증명으로 구성된다.

```
Proxy-Authorization = credentials
```

Unlike Authorization, the Proxy-Authorization header field applies only to the next inbound 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 inbound 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.

Proxy-Authorization 헤더 필드는 Authorization 필드를 사용한 인증을 요구한 다음 인바운드 프락시에만 적용된다. 체인에 여러 프락시가 사용되는 경우, Proxy-Authenticate 헤더 필드는 자격 증명을 받기를 기대했던 첫 번째 인바운드 프락시에 의해 소비된다. 프락시는 프락시들이 주어진 요청을 협력하여 인증하는 메커니즘인 경우 클라이언트 요청의 자격 증명을 다음 프락시에게 전달할 수 있다.

5. IANA Considerations

5.1. Authentication Scheme Registry

The "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" defines the namespace for the authentication schemes in challenges and credentials. It has been created and is now maintained at <http://www.iana.org/assignments/http-authschemes>.

"Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry"는 challenge와 자격 증명의 인증 scheme에 대한 네임스페이스를 정의한다. 그것은 만들어졌고 현재 <http://www.iana.org/assignments/http-authschemes>에서 유지되고 있다.

5.1.1. Procedure

Registrations MUST include the following fields:

등록에는 다음 필드가 포함되어야 한다.

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

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

이 네임스페이스에 추가할 값은 IETF 검토가 필요하다([RFC5226, Section 4.1 참조).

5.1.2. 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 Framework에는 새로운 인증 scheme이 작동하는 방식에 제약을 가하는 측면이 있다.

o 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 2.3 of [RFC7230]).

o HTTP 인증은 상태 저장 없는 것으로 가정된다. 요청 인증에 필요한 모든 정보는 이전 요청을 기억하는 서버에 의존하지 않고 요청에 제공되어야 한다.(MUST) 기본 커넥션에 기반하거나 이에 구속되는 인증은 이 명세의 범위를 벗어나며, 인증된 사용자가 아닌 다른 당사자가 커넥션을 사용할 수 없도록 조치를 취하지 않는 한 본질적으로 결함이 있다([RFC7230] Section 2.3 참조).

o The authentication parameter "realm" is reserved for defining protection spaces as described in Section 2.2. New schemes MUST NOT use it in a way incompatible with that definition.

o 인증 매개변수 "realm"은 Section 2.2에 설명된 보호 공간을 정의하기 위해 예약되어 있다. 새로운 scheme은 그 정의와 호환할 수 없는 방법으로 그것을 사용해서는 안 된다.(MUST NOT)

o The "token68" notation was introduced for compatibility with existing authentication schemes and can only be used once per challenge or credential. Thus, new schemes ought to use the auth-param syntax instead, because otherwise future extensions will be impossible.

o "token68" 표기법은 기존 인증 scheme과의 호환성을 위해 도입되었으며, 질의 또는 자격 증명 당 한 번만 사용할 수 있다. 따라서 새로운 scheme은 대신에 auth-param 구문을 사용해야 한다. 그렇지 않으면 미래의 확장이 불가능하기 때문이다.

o 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.

o 질의와 자격 증명의 구문 분석은 이 명세에 의해 정의되며 새로운 인증 scheme으로 수정할 수 없다. auth-param 구문을 사용할 경우 모든 매개변수는 토큰 구문과 quoted-string 구문

을 모두 지원해야 하며 구문 분석 후 필드 값에 대한 구문적 제약조건(i.e., quoted-string 처리)을 정의해야 한다. 이는 수신자는 모든 인증 scheme에 적용되는 일반 구문 분석기를 사용할 수 있도록 하기 위해 필요하다.

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.

참고: "realm" 매개변수의 값 구문이 quoted-string로 제한된다는 사실은 새로운 매개변수에 대해 반복되지 않는 잘못된 설계 선택이었다.

o Definitions of new schemes ought to define the treatment of unknown extension parameters. In general, a "must-ignore" rule is preferable to a "must-understand" rule, 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").

o 새로운 scheme의 정의는 알려지지 않은 확장 매개변수의 처리를 정의해야 한다. 일반적으로 "must-ignore" 규칙은 "must-understand" 규칙보다 우선된다. 그렇지 않으면 레거시 수신자가 있는 곳에서 새로운 매개변수를 도입하기 어렵기 때문이다. 또한, 새 매개변수(예: "update the specification" 또는 "use this registry")를 정의하기 위한 정책을 설명하는 것이 좋다.

o 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).

o 인증 scheme는 원-서버 인증(즉, WWW-Authenticate 사용) 및/또는 프락시 인증(즉, Proxy-Authenticate 사용)에 사용할 수 있는지 여부를 문서화해야 한다.

o 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 (Section 5.2.2.6 of [RFC7234]), within the scope of the request in which they appear.

o Authorization 헤더 필드에 동봉된 자격 증명은 사용자 에이전트에 한정되어 있으므로 나타나는 요청의 범위 내에서 HTTP 캐시와 관련된 "private" Cache-Control 응답 지시어([RFC7234]의 Section 5.2.2.6)와 동일한 영향을 미친다.

Therefore, new authentication schemes that 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", Section 5.2.1.5 of [RFC7234]) or response directives (e.g., “private").

따라서, Authorization 헤더 필드(e.g., 새로 정의된 헤더 필드 사용)에서 자격 증명을 소지하지 않기로 선택한 새로운 인증 scheme에서는 Cache-Control 요청 지시어(예: "no-store", [RFC7234]의 Section 5.2.1.5) 또는 응답 지시어(예: "private")의 사용을 의무화하여 캐싱을 명시적으로 허용하지 않아야 한다.

5.2. Status Code Registration

The "Hypertext Transfer Protocol (HTTP) Status Code Registry" located at <http://www.iana.org/assignments/http-status-codes> has been updated with the registrations below:

<http://www.iana.org/assignments/http-status-codes>에 위치한 "Hypertext Transfer Protocol (HTTP) Status Code Registry"는 아래의 등록과 함께 업데이트되었다.

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

5.3. Header Field Registration

HTTP header fields are registered within the "Message Headers” registry maintained at <http://www.iana.org/assignments/message-headers/>.

HTTP 헤더 필드는 <http://www.iana.org/assignments/message-headers/>에서 유지되는 "Message Headers" 레지스트리 내에 등록된다.

This document defines the following HTTP header fields, so the "Permanent Message Header Field Names" registry has been updated accordingly (see [BCP90]).

본 문서는 다음과 같은 HTTP 헤더 필드를 정의하므로, 그에 따라 "Permanent Message Header Field Names" 레지스트리가 업데이트되었다([BCP90] 참조).

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

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

6. Security Considerations

This section is meant to inform developers, information providers, and users of known security concerns specific to HTTP authentication. More general security considerations are addressed in HTTP messaging [RFC7230] and semantics [RFC7231].

본 섹션은 HTTP 인증과 관련된 알려진 보안 문제를 개발자, 정보 제공자 및 사용자에게 알리기 위한 것이다. 보다 일반적인 보안 고려사항은 HTTP 메시징 [RFC7230] 및 의미론 [RFC7231]에서 다루어진다.

Everything about the topic of HTTP authentication is a security consideration, so the list of considerations below is not exhaustive. Furthermore, it is limited to security considerations regarding the authentication framework, in general, rather than discussing all of the potential considerations for specific authentication schemes (which ought to be documented in the specifications that define those schemes). Various organizations maintain topical information and links to current research on

Web application security (e.g., [OWASP]), including common pitfalls for implementing and using the authentication schemes found in practice.

HTTP 인증에 관한 주제에 관한 모든 것은 보안 고려사항이므로, 아래의 고려사항 목록은 완전하지 않다. 또한, 특정 인증 scheme(그 scheme을 정의하는 명세에 문서화되어야 한다)에 대한 모든 잠재적 고려사항을 논의하기보다는 일반적으로 인증 scheme과 관련한 보안 고려사항으로 제한된다. 다양한 조직은 실무에서 발견된 인증 scheme을 구현하고 사용하기 위한 공통 문제를 포함하여 웹 애플리케이션 보안에 대한 최신 연구(e.g., [OWASP])에 대한 주제 정보와 링크를 유지한다.

6.1. Confidentiality of Credentials

The HTTP authentication framework does not define a single mechanism for maintaining the confidentiality of credentials; instead, each authentication scheme defines how the credentials are encoded prior to transmission. While this provides flexibility for the development of future authentication schemes, it is inadequate for the protection of existing schemes that provide no confidentiality on their own, or that do not sufficiently protect against replay attacks. Furthermore, if the server expects credentials that are specific to each individual user, the exchange of those credentials will have the effect of identifying that user even if the content within credentials remains confidential.

HTTP 인증 프레임워크는 인증 정보의 기밀성을 유지하기 위한 단일 메커니즘을 정의하지 않는다. 대신, 각 인증 scheme은 전송 전에 인증 정보가 인코딩되는 방법을 정의한다. 이는 향후 인증 scheme 개발에 유연성을 제공하지만, 자체로 기밀성을 제공하지 않거나, 재실행 공격으로부터 충분히 보호하지 못하는 기존 제도의 보호에는 부적당하다. 또한 서버가 각 개별 사용자별로 특정한 자격 증명을 기대하는 경우, 자격 증명 내의 내용이 기밀로 유지되더라도 해당 사용자를 식별하는 효과가 있을 것이다.

HTTP depends on the security properties of the underlying transport- or session-level connection to provide confidential transmission of header fields. In other words, if a server limits access to authenticated users using this framework, the server needs to ensure that the connection is properly secured in accordance with the nature of the authentication scheme used. For example, services that depend on individual user authentication often require a connection to be secured with TLS ("Transport Layer Security", [RFC5246]) prior to exchanging any credentials.

HTTP는 헤더 필드의 기밀 전송을 제공하기 위해 기본 전송 또는 세션 레벨 커넥션의 보안 속성에 의존한다. 즉, 서버가 이 프레임워크를 사용하여 인증된 사용자에 대한 접근을 제한하는 경우, 서버는 사용된 인증 scheme의 특성에 따라 연결이 적절하게 보안되도록 할 필요가 있다. 예를 들어, 개별 사용자 인증에 의존하는 서비스에서는 자격 증명을 교환하기 전에 TLS("Transport Layer Security", [RFC5246])로 보안된 커넥션을 필요 하는 경우가 많다.

6.2. Authentication Credentials and Idle Clients

Existing HTTP clients and user agents typically retain authentication information indefinitely. HTTP does not provide a mechanism for the origin server to direct clients to discard these cached credentials, since the protocol has no awareness of how credentials are obtained or managed by the user agent. The mechanisms for expiring or revoking credentials can be specified as part of an authentication scheme definition.

기존 HTTP 클라이언트와 사용자 에이전트는 일반적으로 인증 정보를 무기한 보존한다. 프로토콜은 사용자 에이전트에 의해 자격 증명을 얻거나 관리하는 방법을 인식하지 못하기 때문에, HTTP는 원서버가 클라이언트가 이러한 캐시된 자격 증명을 삭제하도록 지시하는 메커니즘을 제공하지 않는다. 자격 증명을 만료하거나 취소하는 메커니즘은 인증 scheme 정의의 일부로 지정할 수 있다.

Circumstances under which credential caching can interfere with the application's security model include but are not limited to:

자격 증명 캐싱이 애플리케이션의 보안 모델을 방해할 수 있는 상황에는 다음이 포함되지만 이에 국한되지는 않는다.

o Clients that have been idle for an extended period, following which the server might wish to cause the client to re-prompt the user for credentials.

o 장기간 유휴 상태였던 클라이언트, 그 이후에 서버가 사용자에게 자격 증명을 재계약하도록 할 수 있다.

o Applications that 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.

o 세션 종료 표시(예: 페이지의 "로그아웃" 또는 "커밋" 버튼)가 포함된 애플리케이션은 클라이언트가 자격 증명을 보유할 더 이상의 이유가 없음을 "알고 있다".

User agents that cache credentials are encouraged to provide a readily accessible mechanism for discarding cached credentials under user control.

자격 증명을 캐시하는 사용자 에이전트는 사용자 제어 하에 캐시된 자격 증명을 삭제하기 위한 쉽게 액세스할 수 있는 메커니즘을 제공하도록 권장된다.

6.3. Protection Spaces

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

보호 공간을 설정하는 "realm" 메커니즘에만 의존하는 인증 scheme은 원서버의 모든 리소스에 자격 증명을 노출시킬 것이다. 리소스로 인증된 요청을 성공적으로 수행한 클라이언트는 동일한 원서버의 다른 리소스에 대해 동일한 인증 자격 증명을 사용할 수 있다. 이렇게 하면 다른 리소스가 다른 리소스에 대한 인증 자격 증명을 수집할 수 있다.

This is of particular concern when an origin 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 (or port number) for each party.

이는 원서버가 동일한 표준 루트 URI(Section 2.2)에 따라 여러 당사자를 위한 리소스를 호스트하는 경우에 특히 중요하다. 가능한 완화 전략으로는 인증 자격 증명에 대한 직접 액세스 제한(i.e., Authorization 요청 헤더 필드의 내용을 사용할 수 없도록 함)과 각 당사자에 대해 다른 호스트 이름(또는 포트 번호)을 사용하여 보호 공간을 분리하는 것이 포함된다.

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 10 of [RFC7230] for the Acknowledgments related to this document revision.

8. References

8.1. Normative References

[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.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

8.2. Informative References

[BCP90] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[OWASP] van der Stock, A., Ed., "A Guide to Building Secure Web Applications and Web Services", The Open Web Application Security Project (OWASP) 2.0.1, July 2005, <https://www.owasp.org/>.

[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.

[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.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. Changes from RFCs 2616 and 2617

The framework for HTTP Authentication is now defined by this document, rather than RFC 2617.

HTTP 인증에 대한 프레임워크는 이제 RFC 2617이 아닌 이 문서에 의해 정의된다.

The "realm" parameter is no longer always required on challenges; consequently, the ABNF allows challenges without any auth parameters. (Section 2)

"realm" 매개변수는 더 이상 질의에 항상 필요하지 않다. 따라서 ABNF는 인증 매개변수 없이 질의 허용한다. (Section 2)

The "token68" alternative to auth-param lists has been added for consistency with legacy authentication schemes such as “Basic". (Section 2)

auth-param 리스트에 대한 "token68" 대안이 "Basic"과 같은 기존 인증 scheme과의 일관성을 위해 추가되었다. (Section 2)

This specification introduces the Authentication Scheme Registry, along with considerations for new authentication schemes. (Section 5.1)

이 명세는 새로운 인증 scheme에 대한 고려사항과 함께 Authentication Scheme Registry를 소개한다. (Section 5.1)

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 [RFC7230]:

BWS = <BWS, see [RFC7230], Section 3.2.3>
OWS = <OWS, see [RFC7230], Section 3.2.3>
quoted-string = <quoted-string, see [RFC7230], Section 3.2.6>
token = <token, see [RFC7230], Section 3.2.6>

Appendix C. Collected ABNF

In the collected ABNF below, list rules are expanded as per Section 1.2 of [RFC7230].

```
Authorization = credentials

BWS = <BWS, see [RFC7230], Section 3.2.3>

OWS = <OWS, see [RFC7230], Section 3.2.3>

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

challenge = auth-scheme [ 1*SP ( token68 / [ ( "," / auth-param ) *( OWS "," [ OWS
auth-param ] ) ] ) ]
credentials = auth-scheme [ 1*SP ( token68 / [ ( "," / auth-param ) *( OWS
"," [ OWS auth-param ] ) ] ) ]

quoted-string = <quoted-string, see [RFC7230], Section 3.2.6>

token = <token, see [RFC7230], Section 3.2.6>
token68 = 1*( ALPHA / DIGIT / "-" / "." / "_" / "~" / "+" / "/" ) *"="
```