Workgroup: HTTPBIS
Internet-Draft:
draft-ietf-httpbis-unprompted-auth-06
Published: 24 January 2024
Intended Status: Standards Track
Expires: 27 July 2024
Authors: D. Schinazi
Google LLC
D. Oliver
Guardian Project
J. Hoyland
Cloudflare Inc.

The Signature HTTP Authentication Scheme

Abstract

Existing HTTP authentication schemes are probeable in the sense that it is possible for an unauthenticated client to probe whether an origin serves resources that require authentication. It is possible for an origin to hide the fact that it requires authentication by not generating Unauthorized status codes, however that only works with non-cryptographic authentication schemes: cryptographic signatures require a fresh nonce to be signed, and there is no existing way for the origin to share such a nonce without exposing the fact that it serves resources that require authentication. This document proposes a new non-probeable cryptographic authentication scheme.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://httpwg.org/http-extensions/draft-ietf-httpbis-unprompted-auth.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-httpbis-unprompted-auth/.

Discussion of this document takes place on the HTTP Working Group mailing list (mailto:ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/. Working Group information can be found at https://httpwg.org/.

Source for this draft and an issue tracker can be found at https://github.com/httpwg/http-extensions/labels/unprompted-auth.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 July 2024.

Copyright Notice

Table of Contents

1. Introduction

HTTP authentication schemes (see Section 11 of [HTTP]) allow origins to restrict access for some resources to only authenticated requests. While these schemes commonly involve a challenge where the origin asks the client to provide authentication information, it is possible for clients to send such information unprompted. This is particularly useful in cases where an origin wants to offer a service or capability only to "those who know" while all others are given no indication the service or capability exists. Such designs rely on an externally-defined mechanism by which keys are distributed. For example, a company might offer remote employee access to company services directly via its website using their employee credentials, or offer access to limited special capabilities for specific employees, while making discovering (probing for) such capabilities difficult. Members of less well-defined communities might use more ephemeral keys to acquire access to geography- or capability-specific resources, as issued by an entity whose user base is larger than the available resources can support (by having that entity metering the availability of keys temporally or geographically).

While digital-signature-based HTTP authentication schemes already exist ([HOBA]), they rely on the origin explicitly sending a fresh challenge to the client, to ensure that the signature input is fresh. That makes the origin probeable as it send the challenge to unauthenticated clients. This document defines a new signature-based authentication scheme that is not probeable.

1.1. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the notation from Section 1.3 of [QUIC].

2. The Signature Authentication Scheme

This document defines the "Signature" HTTP authentication scheme. It uses asymmetric cryptography. User agents possess a key ID and a

public/private key pair, and origin servers maintain a mapping of authorized key IDs to their associated public keys.

The client uses a TLS keying material exporter to generate data to be signed (see Section 4) then sends the signature using the Authorization or Proxy-Authorization header field. The signature and additional information are exchanged using authentication parameters (see Section 5).

3. TLS Usage

This authentication scheme is only defined for uses of HTTP with TLS [TLS]. This includes any use of HTTP over TLS as typically used for HTTP/2 [HTTP/2], or HTTP/3 [HTTP/3] where the transport protocol uses TLS as its authentication and key exchange mechanism [QUIC-TLS].

Because the TLS keying material exporter is only secure for authentication when it is uniquely bound to the TLS session [RFC7627], the Signature authentication scheme requires either one of the following properties:

- The TLS version in use is greater or equal to 1.3 [TLS].
- The TLS version in use is 1.2 and the Extended Master Secret extension [RFC7627] has been negotiated.

Clients **MUST NOT** use the Signature authentication scheme on connections that do not meet one of the two properties above. If a server receives a request that uses this authentication scheme on a connection that meets neither of the above properties, the server **MUST** treat the request as malformed.

4. Computing the Authentication Proof

The user agent computes the authentication proof using a TLS keying material exporter [KEY-EXPORT] with the following parameters:

- the label is set to "EXPORTER-HTTP-Signature-Authentication"
- the context is set to the structure described in Section 4.1
- the exporter output length is set to 48 bytes (see Section 4.2)

4.1. Key Exporter Context

The TLS key exporter context is described in Figure 1:

```
Signature Algorithm (16),
Key ID Length (i),
Key ID (..),
Public Key Length (i),
Public Key (..),
Scheme Length (i),
Scheme (..),
Host Length (i),
Host (..),
Port (16),
Realm Length (i),
Realm (..),
```

Figure 1: Key Exporter Context Format

The key exporter context contains the following fields:

Signature Algorithm: The signature scheme sent in the s Parameter (see Section 5.4).

Key ID: The key ID sent in the k Parameter (see Section 5.1).

Public Key: The public key used by the server to validate the signature provided by the client (the encoding is described below).

Scheme: The scheme for this request, encoded using the format of the scheme portion of a URI as defined in Section 3.1 of [URI].

Host: The host for this request, encoded using the format of the host portion of a URI as defined in Section 3.2.2 of [URI].

Port: The port for this request, encoded in network byte order. Note that the port is either included in the URI, or is the default port for the scheme in use; see Section 3.2.3 of [URI].

Realm: The real of authentication that is sent in the realm authentication parameter (Section 11.5 of [HTTP]). If the realm authentication parameter is not present, this **SHALL** be empty. This document does not define a means for the origin to communicate a realm to the client. If a client is not configured to use a specific realm, it **SHALL** use an empty realm and **SHALL NOT** send the realm authentication parameter.

The Signature Algorithm and Port fields are encoded as unsigned 16-bit integers in network byte order. The Key ID, Public Key, Scheme, Host, and Real fields are length prefixed strings; they are preceded by a Length field that represents their length in bytes. These length fields are encoded using the variable-length integer encoding

from Section 16 of [QUIC] and **MUST** be encoded in the minimum number of bytes necessary.

The encoding of the public key is determined by the Signature Algorithm in use as follows:

RSASSA-PSS algorithms: The public key is an RSAPublicKey structure [PKCS1] encoded in DER [X.690]. BER encodings which are not DER **MUST** be rejected.

ECDSA algorithms: The public key is a UncompressedPointRepresentation structure defined in Section 4.2.8.2 of [TLS], using the curve specified by the SignatureScheme.

EdDSA algorithms: The public key is the byte string encoding defined in [EdDSA].

This document does not define the public key encodings for other algorithms. In order for a SignatureScheme to be usable with the Signature HTTP authentication scheme, its public key encoding needs to be defined in a corresponding document.

4.2. Key Exporter Output

The key exporter output is 48 bytes long. Of those, the first 32 bytes are part of the input to the signature and the next 16 bytes are sent alongside the signature. This allows the recipient to confirm that the exporter produces the right values. This is described in Figure 2:

```
Signature Input (256),
Verification (128),
```

Figure 2: Key Exporter Output Format

The key exporter context contains the following fields:

Signature Input: This is part of the data signed using the client's chosen asymmetric private key (see Section 4.3).

Verification: The verification is transmitted to the server using the v Parameter (see Section 5.5).

4.3. Signature Computation

Once the Signature Input has been extracted from the key exporter output (see Section 4.2), it is prefixed with static data before

being signed to mitigate issues caused by key reuse. The signature is computed over the concatenation of:

- *A string that consists of octet 32 (0x20) repeated 64 times
- *The context string "HTTP Signature Authentication"
- *A single 0 byte which serves as a separator
- *The Signature Input extracted from the key exporter output (see Section 4.2)

For example, if the Signature Input has all its 32 bytes set to 01, the content covered by the signature (in hexadecimal format) would be:

```
2020202020202020202020202020202020202020202020202020202020202020
2020202020202020202020202020202020202020202020202020202020202020
48545450205369676E617475726520417574686E7469636174696F6E
00
0101010101010101010101010101010101010101010101010101010101010101
```

Figure 3: Example Content Covered by Signature

This constructions mirrors that of the TLS 1.3 CertificateVerify message defined in Section 4.4.3 of [TLS].

The resulting signature is then transmitted to the server using the p Parameter (see Section 5.3).

5. Authentication Parameters

This specification defines the following authentication parameters.

All of the byte sequences below are encoded using base64url (see Section 5 of [BASE64]) without quotes and without padding. In other words, these byte sequence authentication parameters values **MUST NOT** include any characters other then ASCII letters, digits, dash and underscore.

The integer below is encoded without a minus and without leading zeroes. In other words, the integer authentication parameters value **MUST NOT** include any characters other than digits, and **MUST NOT** start with a zero unless the full value is "0".

Using the syntax from [ABNF]:

```
signature-byte-sequence-param-value = *( ALPHA / DIGIT / "-" / "_" )
signature-integer-param-value =  %x31-39 1*4( DIGIT ) / "0"
```

Figure 4: Authentication Parameter Value ABNF

5.1. The k Parameter

The **REQUIRED** "k" (key ID) parameter is a byte sequence that identifies which key the user agent wishes to use to authenticate. This can for example be used to point to an entry into a server-side database of known keys.

5.2. The a Parameter

The **REQUIRED** "a" (public key) parameter is a byte sequence that contains the public key used by the server to validate the signature provided by the client. This avoids key confusion issues (see [SEEMS-LEGIT]). The encoding of the public key is described in Section 4.1.

5.3. The p Parameter

The **REQUIRED** "p" (proof) parameter is a byte sequence that specifies the proof that the user agent provides to attest to possessing the credential that matches its key ID.

5.4. The s Parameter

The **REQUIRED** "s" (signature) parameter is an integer that specifies the signature scheme used to compute the proof transmitted in the "p" directive. Its value is an integer between 0 and 65535 inclusive from the IANA "TLS SignatureScheme" registry maintained at <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-signaturescheme>.

5.5. The v Parameter

The **REQUIRED** "v" (verification) parameter is a byte sequence that specifies the verification that the user agent provides to attest to possessing the key exporter output (see Section 4.2 for details). This avoids issues with signature schemes where certain keys can generate signatures that are valid for multiple inputs (see [SEEMS-LEGIT]).

6. Example

For example, the key ID "basement" authenticating using Ed25519 [ED25519] could produce the following header field:

```
NOTE: '\' line wrapping per RFC 8792

Authorization: Signature \
  k=YmFzZW1lbnQ, \
  a=VGhpcyBpcyBh-HB1YmxpYyBrZXkgaW4gdXNl_GhlcmU, \
  s=2055, \
  v=dmVyaWZpY2F0aW9u_zE2Qg, \
  p=SW5zZXJ0_HNpZ25hdHVyZSBvZiBub25jZSBoZXJlIHdo\
    aWNoIHRha2VzIDUxMiBiaXRz-GZvciBFZDI1NTE5IQ
```

Figure 5: Example Header Field

7. Non-Probeable Server Handling

Servers that wish to introduce resources whose existence cannot be probed need to ensure that they do not reveal any information about those resources to unauthenticated clients. In particular, such servers **MUST** respond to authentication failures with the exact same response that they would have used for non-existent resources. For example, this can mean using HTTP status code 404 (Not Found) instead of 401 (Unauthorized). Such authentication failures can be caused for example by:

- absence of the Authorization (or Proxy-Authorization) field
- failure to parse that field
- use of the Signature authentication scheme with an unknown key ID
- mismatch between key ID and provided public key
- failure to validate the verification parameter
- failure to validate the signature.

In order to validate the signature, the server needs to first parse the field containing the signature, then look up the key ID in its database of public keys, and finally perform the cryptographic validation. These steps can take time, and an attacker could detect use of this mechanism if that time is observable by comparing the timing of a request for a known non-existent resource to the timing of a request for a potentially authenticated resource. Servers can mitigate this observability by slightly delaying responses to some non-existent resources such that the timing of the authentication verification is not observable. This delay needs to be carefully considered to avoid having the delay itself leak the fact that this origin uses this mechanism at all.

Non-probeable resources also need to be non-discoverable for unauthenticated users. For example, if a server operator wishes to

hide an authenticated resource by pretending it does not exist to unauthenticated users, then the server operator needs to ensure there are no unauthenticated pages with links to that resource, and no other out-of-band ways for unauthenticated users to discover this resource.

8. Intermediary Considerations

Since the Signature HTTP authentication scheme leverages TLS keying material exporters, its output cannot be transparently forwarded by HTTP intermediaries. HTTP intermediaries that support this specification have two options:

* The intermediary can validate the authentication received from the client, then inform the upstream HTTP server of the presence of valid authentication.

* The intermediary can export the Signature Input and Verification (see Section 4.2}), and forward it to the upstream HTTP server, then the upstream server performs the validation.

The mechanism for the intermediary to communicate this information to the upstream HTTP server is out of scope for this document.

Note that both of these mechanisms require the upstream HTTP server to trust the intermediary. This is usually the case because the intermediary already needs access to the TLS certificate private key in order to respond to requests.

9. Security Considerations

The Signature HTTP authentication scheme allows a user agent to authenticate to an origin server while guaranteeing freshness and without the need for the server to transmit a nonce to the user agent. This allows the server to accept authenticated clients without revealing that it supports or expects authentication for some resources. It also allows authentication without the user agent leaking the presence of authentication to observers due to clear-text TLS Client Hello extensions.

The authentication proofs described in this document are not bound to individual HTTP requests; if the key is used for authentication proofs on multiple requests on the same connection, they will all be identical. This allows for better compression when sending over the wire, but implies that client implementations that multiplex different security contexts over a single HTTP connection need to ensure that those contexts cannot read each other's header fields. Otherwise, one context would be able to replay the Authorization header field of another. This constraint is met by modern Web browsers. If an attacker were to compromise the browser such that it

could access another context's memory, the attacker might also be able to access the corresponding key, so binding authentication to requests would not provide much benefit in practice.

Key material used for the Signature HTTP authentication scheme **MUST NOT** be reused in other protocols. Doing so can undermine the security guarantees of the authentication.

Origins offering this scheme can link requests that use the same key. However, requests are not linkable across origins if the keys used are specific to the individual origins using this scheme.

10. IANA Considerations

10.1. HTTP Authentication Schemes Registry

This document, if approved, requests IANA to register the following entry in the "HTTP Authentication Schemes" Registry maintained at <https://www.iana.org/assignments/http-authschemes>:

Authentication Scheme Name: Signature
Reference: This document
Notes: None

10.2. TLS Keying Material Exporter Labels

This document, if approved, requests IANA to register the following entry in the "TLS Exporter Labels" registry maintained at <https://www.iana.org/assignments/tls-parameters#exporter-labels>:

Value: EXPORTER-HTTP-Signature-Authentication
DTLS-OK: N
Recommended: Y
Reference: This document

11. References

11.1. Normative References

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

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[EdDSA] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/

RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[FOLDING] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[KEY-EXPORT] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/rfc/rfc5705>.

[PKCS1] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/rfc/rfc8017>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

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

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <https://www.rfc-editor.org/rfc/rfc7627>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

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

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8824-1:2021 , February 2021.

11.2. Informative References

[ED25519] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/rfc/rfc8410>.

[HOBA] Farrell, S., Hoffman, P., and M. Thomas, "HTTP Origin-Bound Authentication (HOBA)", RFC 7486, DOI 10.17487/RFC7486, March 2015, <https://www.rfc-editor.org/rfc/rfc7486>.

[HTTP/2] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/rfc/rfc9113>.

[HTTP/3] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[MASQUE-ORIGINAL] Schinazi, D., "The MASQUE Protocol", Work in Progress, Internet-Draft, draft-schinazi-masque-00, 28 February 2019, <https://datatracker.ietf.org/doc/html/draft-schinazi-masque-00>.

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[SEEMS-LEGIT] Jackson, D., Cremers, C., Cohn-Gordon, K., and R. Sasse, "Seems Legit: Automated Analysis of Subtle Attacks on Protocols That Use Signatures", CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2165–2180, DOI 10.1145/3319535.3339813, 2019, <https://doi.org/10.1145/3319535.3339813>.

Acknowledgments

The authors would like to thank many members of the IETF community, as this document is the fruit of many hallway conversations. In particular, the authors would like to thank David Benjamin, Nick Harper, Dennis Jackson, Ilari Liusvaara, François Michel, Lucas

Pardue, Justin Richer, Ben Schwartz, Martin Thomson, and Chris A. Wood for their reviews and contributions. The mechanism described in this document was originally part of the first iteration of MASQUE [MASQUE-ORIGINAL].

Authors' Addresses

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America

Email: dschinazi.ietf@gmail.com

David M. Oliver
Guardian Project

Email: david@guardianproject.info
URI: https://guardianproject.info

Jonathan Hoyland
Cloudflare Inc.

Email: jonathan.hoyland@gmail.com