Abstract

For a client to access a particular resource on the Web, a server must expend a certain amount of computational effort to respond to the request. In some cases this computational effort is sizeable and the server may want to only respond to certain clients. For example, in a distributed denial-of-service attack, a server may require all clients to expend a certain amount of resources via a client-run proof-of-work algorithm to throttle the number of incoming requests to a more manageable number. This document details a new authentication scheme for HTTP that may be used to request and transmit proofs in HTTP headers.

Feedback

This specification is a thought exercise that may be submitted on a standardization track in the future. Feedback related to this specification should be sent to msporny@digitalbazaar.com [1].

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 http://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 December 30, 2015.
Internet-Draft
Proof-based Authentication for HTTP Messages     June 2015

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors.  All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document.  Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document.

Table of Contents

1.  Introduction .................................................. 2
2.  The Components of a Proof .................................... 3
  2.1.  Proof Parameters .......................................... 3
         2.1.1.  type ................................................. 3
  2.2.  Ambiguous Parameters ...................................... 4
3.  The ’Proof’ HTTP Authentication Scheme ........................ 4
  3.1.  The Authorization Header .................................. 4
         3.1.1.  Initiating Proof Authorization ..................... 4
4.  Proof of Patience .............................................. 5
  4.1.  Proof of Patience Parameters .............................. 5
         4.1.1.  type ................................................. 5
         4.1.2.  token ............................................... 5
  4.2.  Proof of Patience Example ................................. 6
5.  References .................................................... 7
  5.1.  Normative References ..................................... 7
  5.2.  Informative References .................................. 7
  5.3.  URIs ....................................................... 8
Appendix A. Security Considerations .............................. 8
Appendix B. Extensions ............................................ 8
Appendix C. Test Values ........................................... 8
  C.1.  Default Test .............................................. 8
Appendix D. Acknowledgements ..................................... 8
Appendix E. IANA Considerations .................................. 9
  E.1.  Signature Authentication Scheme .......................... 9
  E.2.  Proof Algorithm Registry .................................. 9
Authors’ Addresses ............................................... 9

1.  Introduction

This protocol extension is intended to provide a simple and standard
way for clients and servers to request and transmit proofs to one
another.
For a client to access a particular resource on the Web, a server must expend a certain amount of computational effort to respond to the request. In some cases this computational effort is sizeable and the server may want to only respond to certain clients. For example, in a distributed denial-of-service attack, a server may require all clients to expend a certain amount of resources via a client-run proof-of-work algorithm to throttle the number of incoming requests to a more manageable number.

Fundamentally, proofs need to separate clients that are using the system in good faith versus clients that are attempting to attack the system. In order to do this, a proof must be selected that creates a net economic loss to an attacker of the system while also not making legitimate uses of the system too costly. Proofs must be tuned to the type of protection the server would like to apply to a target resource.

There are many classes of proofs that can be used by a server to identify legitimate clients. Some of these include proof of work, proof of navigation, proof of humanity, and proof of patience. A proof of work typically requires a time consuming mathematical puzzle to be solved, such as prime factorization. A proof of navigation requires tokens to be fetched from a list of servers on a network in a particular order. A proof of humanity requires a problem to be solved, like identifying an emotion in an image, that cannot easily be determined by a computer. A proof of patience requires a client to prove that it has waited for a certain period of time before being allowed to have access to a particular resource on a server.

2. The Components of a Proof

There are a number of components in a proof that are common between the ‘Proof’ HTTP Authentication Scheme. This section details the components of a HTTP Proof.

2.1. Proof Parameters

The following section details the proof parameters.

2.1.1. type

REQUIRED. The ‘type’ field specifies the specific proof algorithm that is employed by the rest of the parameters. The value of this field SHOULD be a registered algorithm as defined in the Proof Algorithm Registry.
2.2. Ambiguous Parameters

If any of the parameters listed above are erroneously duplicated in the associated header field, then the last parameter defined MUST be used. Any parameter that is not recognized as a parameter, or is not well-formed, MUST be ignored.

3. The 'Proof' HTTP Authentication Scheme

The "Proof" authentication scheme is based on the model that the client must establish a solution (aka proof) to a challenge posed by the server. The scheme is parameterized enough such that it is not bound to any particular proofing algorithm.

3.1. The Authorization Header

The client is expected to send an Authorization header (as defined in RFC 7235 [RFC7235], Section 4.1 [2]) where the "auth-scheme" is "Proof" and the "auth-param" parameters meet the requirements listed in Section 2: The Components of a Proof and the specific parameters for the proof algorithm being used.

Authorization: Proof type=example, foo=bar

Note: The example above uses 'example', 'foo', and 'bar' for purely illustrative purposes. A real example is provided in the section below titled 'Proof of Patience'.

3.1.1. Initiating Proof Authorization

A server may notify a client when a protected resource could be accessed by authenticating itself to the server via a proof. To initiate this process, the server will request that the client authenticate itself via a 401 response code. The server should specify which type of proof it would like to have established in the WWW–Authenticate header.

The example below is provided to illustrate a generic proof flow, but the proof parameters are only for illustrative purposes. A more specific example is included in the section titled "Proof of Patience". Given the following request:

GET /examples/protected HTTP/1.1
Host: example.com

The following request for a proof is supplied by the server via the WWW–Authenticate header:
HTTP/1.1 401 Unauthorized
Date: Thu, 05 Jul 2015 21:31:41 GMT
Content-Length: 1234
Content-Type: text/html
WWW-Authenticate: Proof type=example, foo=bar

The client performs what is necessary given the proof and then transmits the established proof back to the server via the 'Authorization' header.

GET /examples/protected HTTP/1.1
Host: example.com
Date: Thu, 05 Jul 2015 21:46:52 GMT
Authorization: Proof type=example, foo=bar

The server would then verify the proof and fulfill the request if the proof is valid.

4. Proof of Patience

The Proof of Patience algorithm is a type of proof that demonstrates that a client waited for a certain period of time before attempting a request.

4.1. Proof of Patience Parameters

4.1.1. type

REQUIRED. The 'type' field must be set to 'patience'.

4.1.2. token

REQUIRED. The 'token' field is typically a server-encrypted secret that, when retransmitted to the server after a defined wait period, proves that the client has waited a certain amount of time that was requested by the server. It is important that the server specify the amount of time to wait via the 'Retry-After' HTTP Header. The client should wait until the number of seconds specified in 'Retry-After' has elapsed. The client should then attempt the initial request again with the contents supplied in the 'WWW-Authenticate' copied into a new 'Authorization' header.

The contents of the token is specific to the server. For example, the token could contain the requesting IP of the client, the specific HTTP resource the token is tied to, and the time period when the token is valid. The server could then encrypt the data, base64...
encode it, and set 'token' to the encrypted, base64-encoded value. This ensures that the client cannot tamper with the data and can simply re-transmit the token back to the server after a certain period of time has elapsed to establish the proof of patience.

4.2. Proof of Patience Example

For example, given the following HTTP request:

```
POST /examples HTTP/1.1
Host: example.com
Date: Thu, 05 Jul 2015 21:31:40 GMT
Content-Type: application/json
Content-Length: 3

{}
```

The following request for a proof of patience is supplied by the server via the WWW-Authenticate header:

```
HTTP/1.1 401 Unauthorized
Date: Thu, 05 Jul 2015 21:31:41 GMT
Content-Length: 1234
Content-Type: text/html
WWW-Authenticate: Proof type=patience, token="jA0EAwMpYIIz/X7aE+u3ADHjb4="
Retry-After: 30
...
```

The client then waits for 30 seconds, as specified in the 'Retry-After' header, and then transmits the proof of patience back to the server via the 'Authorization' header.

```
POST /examples HTTP/1.1
Host: example.com
Date: Thu, 05 Jul 2015 21:32:11 GMT
Content-Type: application/json
Content-Length: 3
Authorization: Proof type=patience, token="jA0EAwMpYIIz/X7aE+u3ADHjb4="
{}
```

The server would then decrypt the token and verify that the contents of the token meet the proof of patience requirement. If the verification is successful, the server would fulfill the request.
5. References

5.1. Normative References

[I-D.ietf-jose-json-web-algorithms]


5.2. Informative References


5.3. URIs


Appendix A. Security Considerations

Editor's note: Clearly there are a number of concerning reverse denial of service attacks that could be executed over non-HTTPS connections. A simple header injection could keep a client spinning for a very long time in the event of a proof of work (e.g. by corrupting the number of iterations to perform). More thinking needs to be put into this section.

Appendix B. Extensions

This specification was designed to be simple, modular, and extensible. Developers that desire more functionality than this specification provides are urged to ensure that an extension specification doesn’t already exist before implementing a proprietary extension.

If extensions to this specification are made by adding new Proof Parameters, those extension parameters MUST be registered in the Proof Parameter Registry. The registry will be created and maintained at (the suggested URI) http://www.iana.org/assignments/http-proof-parameters. An example entry in this registry is included below:

Proof Parameter: iterations
Reference to specification: [SOME_ALGORITHM], Section XYZ.
Notes (optional): The 'iterations' parameter is used to specify the number of iterations to run on the mathematical proof algorithm.

Appendix C. Test Values

Provide a simple test of a Proof of Patience.

C.1. Default Test

Appendix D. Acknowledgements

The editor would like to thank the following individuals for feedback on and implementations of the specification (in alphabetical order): None yet.
Appendix E. IANA Considerations

E.1. Signature Authentication Scheme

The following entry should be added to the Authentication Scheme Registry located at http://www.iana.org/assignments/http-authschemes

Authentication Scheme Name: Proof
Reference: [RFC_THIS_DOCUMENT], Section 2.
Notes (optional): The Proof scheme is designed for clients to authenticate themselves with a server by performing a mathematical proof.

E.2. Proof Algorithm Registry

The following initial entries should be added to the Proof Algorithm Registry to be created and maintained at (the suggested URI)

http://www.iana.org/assignments/proof-algorithms:

Algorithm Name: patience
Reference: [SCRYPT_RFC_WHEN_IT_EXISTS] Status: active

Authors’ Addresses

Manu Sporny
Digital Bazaar
1700 Kraft Drive
Suite 2408
Blacksburg, VA 24060
US

Phone: +1 540 961 4469
Email: msporny@digitalbazaar.com
URI: http://manu.sporny.org/

Dave
Digital Bazaar
1700 Kraft Drive
Suite 2408
Blacksburg, VA 24060
US

Phone: +1 540 961 4469
Email: dlongley@digitalbazaar.com
URI: https://github.com/dlongley