Workgroup: Network Working Group
Internet-Draft:
draft-pauly-privacypass-auth-scheme-00
Published: 31 January 2022
Intended Status: Standards Track
Expires: 4 August 2022
Authors: T. Pauly S. Valdez C.A. Wood
Apple Inc. Google LLC Cloudflare

The Privacy Pass HTTP Authentication Scheme

Abstract

This document defines an HTTP authentication scheme that can be used by clients to redeem Privacy Pass tokens with an origin. It can also be used by origins to challenge clients to present an acceptable Privacy Pass token.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/tfpauly/privacy-proxy.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Privacy Pass tokens are unlinkable authenticators that can be used to anonymously authorize a client (see [I-D.ietf-privacypass-architecture]). A client possessing such a token is able to prove that it was able to get a token issued by a token issuer -- based on some check from a token issuer, such as authentication or solving a CAPTCHA -- without allowing the relying party redeeming the client's token (the origin) to link it with issuance flow.

Different types of authenticators, using different token issuance protocols, can be used as Privacy Pass tokens.

This document defines a common HTTP authentication scheme ([RFC7235]), PrivateToken, that allows clients to redeem various kinds of Privacy Pass tokens.

Clients and relying parties interact using this scheme to perform the token challenge and token redemption flow. Clients use a token issuance protocol to actually fetch tokens to redeem.

```
 Client                              Relying Party (Origin)

    <------------------------------- Challenge \
                                               |
+----------------------------------\           |
|                                  |           |
|  Issuance Protocol               |           |
|                                  |           |
+----------------------------------/           |
                                               |
     Redemption -------------------------- >  /
```

Figure 1: Token Architectural Components

In addition to working with different token issuance protocols, this scheme supports both interactive (online challenges) and non-interactive (pre-fetched) token redemption, as well as the ability to scope a token to a specific resource or origin. Relying parties that request and redeem tokens can choose a specific kind of token, as appropriate for its use case. For example, non-interactive token redemption that is not scoped to a specific origin can be used as a replacement for CAPTCHAs, as exemplified by the original Privacy Pass work [DGSTV18].

1.1. Terminology

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 following terms to refer to various roles and functions in the Privacy Pass architecture:

* Client: A client is a application or device, generally operated by a user, that can be issued tokens via an issuance protocol, and can redeem these tokens with an origin.

* Origin: An HTTP server that acts as the relying party and can redeem tokens presented by a client. When used in a web context, this represents the origin the client is accessing.

* Token: A signed message that can be issued to a client and redeemed without allowing token redemption to be linked to issuance.

* Interactive / non-interactive token: An interactive token signs a nonce generated by an origin as part of a challenge for tokens. This means that the client needs to fetch a new token for this

challenge in order to redeem it and cannot use a pre-fetched token. A non-interactive token is one that can be pre-fetched.

* Issuance protocol: A protocol by which the client fetches tokens. Every issuance protocol includes two functions: validating or authenticating the client, and issuing a token to the client.

* Issuer: An entity that generates tokens for clients using one or more issuance protocols. An Issuer is identified by an Issuer name.

* Issuer key: Keying material that can be used with an issuance protocol to create a signed token.

* Token challenge: A requirement for tokens sent from an origin to a client, using the "WWW-Authenticate" HTTP header. This may be a challenge for an interactive token or a non-interactive token. A challenge defines the issuance protocol and issuer name to use for a token.

* Token redemption: An action by which a client presents a token to an origin, using the "Authorization" HTTP header.

2. HTTP Authentication Scheme

Token redemption is performed using HTTP Authentication ([RFC7235]), with the scheme "PrivateToken". Origins challenge clients to present a token from a specific issuer (Section 2.1). Once a client has received a token from that issuer, or already has a valid token available, it presents the token to the origin (Section 2.2).

2.1. Token Challenge

Origins send a token challenge to Clients in an "WWW-Authenticate" header with the "PrivateToken" scheme. This challenge includes a TokenChallenge message, along with information about what keys to use when requesting a token from the issuer.

The TokenChallenge message has the following structure:

```
struct {
    uint16_t token_type;
    opaque issuer_name<1..2^16-1>;
    opaque redemption_nonce<0..32>;
    opaque origin_name<0..2^16-1>;
} TokenChallenge;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer, in network byte order. This type indicates the issuance protocol used to generate the token. Values are registered in an IANA registry, Section 6.2. Challenges with unsupported token_type values MUST be ignored.

* "issuer_name" is a string containing the name of the issuer. This is a hostname that is used to identify the issuer that is allowed to issue tokens that can be redeemed by this origin.

* "redemption_nonce" is an optional field. If present, it indicates that a client needs to present an interactive token, generated specifically in response to this challenge. If empty, the client can use a non-interactive token. When present, this valid is a fresh 32-byte nonce generated by the origin for each challenge. Valid lengths for this field are either 0 or 32 bytes. Challenges with redemption_nonce values of invalid lengths MUST be ignored.

* "origin_name" is an optional string containing the name of the origin. This allows a token to be scoped to a specific origin. If empty, any non-origin specific token can be redeemed.

When used in an authentication challenge, the "PrivateToken" scheme uses the following attributes:

* "challenge", which contains a base64url-encoded [RFC4648] TokenChallenge value. This MUST be unique for every 401 HTTP response to prevent replay attacks. This attribute is required for all challenges.

* "token-key", which contains a base64url encoding of the public key for use with the issuance protocol indicated by the challenge. This attribute MAY be omitted in deployments where clients are able to retrieve the issuer key using an out-of-band mechanism.

* "max-age", an optional attribute that consists of the number of seconds for which the challenge will be accepted by the Origin.

Clients can ignore the challenge if the token-key is invalid or otherwise untrusted.

Origins MAY also include the standard "realm" attribute, if desired. Issuance protocols MAY require other attributes.

As an example, the WWW-Authenticate header could look like this:

```
WWW-Authenticate: PrivateToken challenge=abc..., token-key=123...
```

Upon receipt of this challenge, a client uses the message and keys in the issuance protocol indicated by the token_type. If the TokenChallenge has a token_type the client does not recognize or support, it MUST NOT parse or respond to the challenge.

Note that it is possible for the WWW-Authenticate header to include multiple challenges, in order to allow the Client to fetch a batch of multiple tokens for future use.

For example, the WWW-Authenticate header could look like this:

```
WWW-Authenticate: PrivateToken challenge=abc..., token-key=123...,
PrivateToken challenge=def..., token-key=234...
```

2.2. Token Redemption

The output of the issuance protocol is a token that corresponds to the origin's challenge (see Section 2.1). A token is a structure that begins with a two-octet field that indicates a token type, which MUST match the token_type in the TokenChallenge structure.

```
struct {
    uint16_t token_type;
    uint8_t nonce[32];
    uint8_t context[32];
    uint8_t token_key_id[Nid];
    uint8_t authenticator[Nk];
} Token;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer, in network byte order. This value must match the value in the challenge (Section 2.1).

* "nonce" is a 32-octet message containing a client-generated random nonce.

* "context" is a 32-octet message containing the hash of the original TokenChallenge, SHA256(TokenChallenge).

* "token_key_id" is an Nid-octet identifier for the the token authentication key. The value of this field is defined by the token_type and corresponding issuance protocol.

* "authenticator" is a Nk-octet authenticator that covers the preceding fields in the token. The value of this field is defined by the token_type and corresponding issuance protocol.

The authenticator value in the Token structure is computed over the token_type, nonce, context, and token_key_id fields.

When used for client authorization, the "PrivateToken" authentication scheme defines one parameter, "token", which contains the base64url-encoded Token struct. All unknown or unsupported parameters to "PrivateToken" authentication credentials MUST be ignored.

Clients present this Token structure to Origins in a new HTTP request using the Authorization header as follows:

```
Authorization: PrivateToken token=abc...
```

For token types that support public verifiability, origins verify the token authenticator using the public key of the issuer, and validate that the signed message matches the concatenation of the client nonce and the hash of a valid TokenChallenge. For interactive tokens, origins store the nonces of previous TokenChallenge structures in order to validate uniqueness. A TokenChallenge MAY be bound to a specific HTTP session with client, but origins can also accept tokens for valid challenges in new sessions. For non-interactive tokens, origins SHOULD implement some form of double spend prevention that prevents a token with the same nonce from being redeemed twice. This prevents clients from "replaying" tokens for previous challenges.

If a client is unable to fetch a token, it MUST react to the challenge as if it could not produce a valid Authorization response.

3. Issuance Protocol Requirements

Clients initiate the issuance protocol using a challenge, a randomly generated nonce, and a public key for the issuer. The issuance protocol itself can be any interactive protocol between client, issuer, or other parties that produces a valid authenticator over the client's input, subject to the following security requirements.

1. Unconditional input secrecy. The issuance protocol MUST NOT reveal anything about the client's private input, including the challenge and nonce. The issuance protocol can reveal the issuer public key for the purposes of determining which private key to use in producing the issuance protocol. A result of this property is that the redemption flow is unlinkable from the issuance flow.

2. One-more forgery security. The issuance protocol MUST NOT allow malicious clients to forge tokens without interacting with the issuer directly.

3. Concurrent security. The issuance protocol MUST be safe to run concurrently with arbitrarily many clients.

4. User Interaction

When used in contexts like websites, origins that challenge clients for tokens need to consider how to optimize their interaction model to ensure a good user experience.

Tokens challenges can be performed without explicit user involvement, depending on the issuance protocol. If tokens are scoped to a specific origin, there is no need for per-challenge user interaction. Note that the issuance protocol may separately involve user interaction if the client needs to be newly validated.

The use of interactive tokens can add user-perceivable latency, since such tokens cannot be pre-fetched. Origins need not block useful work on token authentication. Instead, token authentication can be used in similar ways to CAPTCHA validation today, but without the need for user interaction. If issuance is taking a long time, a website could show an indicator that it is waiting, or fall back to another method of user validation.

An origin MUST NOT issue more than one interactive challenge for a given token type and issuer per client request. If an origin issues a large number of challenges, such as more than once for each request, this can indicate that the origin is either not functioning correctly or is trying to attack or overload the client or issuance server. In such cases, a client MUST ignore redundant token challenges for the same request and SHOULD alert the user if possible.

Origins MAY include multiple challenges, where each challenge refers to a different issuer or a different token type, to allow clients to choose a preferred issuer or type.

5. Security Considerations

The security properties of token challenges vary depending on whether the challenge is interactive or not, as well as whether the challenge is per-origin or not. For example, non-interactive, cross-origin tokens can be replayed from one party by another, as shown below.

```
 Client         Attacker                Origin

                      <----------- Challenge \
                                             |
  <--------- Challenge                       |
                                             |
  Redemption ---->                           |
                                             |
                      Redemption ----------> /
```

Figure 2: Token Architectural Components

Interactive token challenges require clients to obtain matching tokens when challenged, rather than presenting a token that was obtained in the past. This means that issuance and redemption events will occur at approximately the same time. For example, if a client is challenged for an interactive token at time T1 and then subsequently obtains a token at time T2, a colluding issuer and origin can link this to the same client if T2 is unique to the client. This linkability is less feasible as the number of issuance events at time T2 increases. Depending on the "max-age" token challenge attribute, clients MAY try to augment the time between getting challenged then redeeming a token so as to make this sort of linkability more difficult. For more discussion on correlation risks between token issuance and redemption, see [I-D.ietf-privacypass-architecture].

Applications SHOULD constrain tokens to a single origin unless the use case can accommodate such replay attacks.

All random values in the challenge and token MUST be generated using a cryptographically secure source of randomness.

6. IANA Considerations

6.1. Authentication Scheme

This document registers the "PrivateToken" authentication scheme in the "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" established by [RFC7235].

Authentication Scheme Name: PrivateToken

Pointer to specification text: Section 2 of this document

6.2. Token Type Registry

The "Token Type" registry lists identifiers for issuance protocols defined for use with the Privacy Pass token authentication scheme.

These identifiers are two-byte values, so the maximum possible value is 0xFFFF = 65535.

Template:

* Value: The two-byte identifier for the algorithm
* Name: Name of the issuance protocol
* Publicly Verifiable: A Y/N value indicating if the output tokens are publicly verifiable
* Public Metadata: A Y/N value indicating if the output tokens can contain public metadata.
* Private Metadata: A Y/N value indicating if the output tokens can contain private metadata.
* Nk: The length in bytes of an output authenticator
* Nid: The length of the token key identifier
* Reference: Where this algorithm is defined

The initial contents for this registry are defined in the table below.

Value	Name	Publicly Verifiable	Public Metadata	Private Metadata	Nk	Nid	Reference
0x0000	(reserved)	N/A	N/A	N/A	N/A	N/A	N/A

Table 1: Token Types

7. References

7.1. Normative References

[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>.

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

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

[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>.

7.2. Informative References

[DGSTV18] "Privacy Pass, Bypassing Internet Challenges Anonymously", n.d., <https://petsymposium.org/2018/files/papers/issue3/popets-2018-0026.pdf>.

[I-D.ietf-privacypass-architecture] Davidson, A. and C. A. Wood, "Privacy Pass Architectural Framework", Work in Progress, Internet-Draft, draft-ietf-privacypass-architecture-01, 22 February 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-architecture-01>.

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

Steven Valdez
Google LLC

Email: svaldez@chromium.org

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net