

HTTPAuth
Internet-Draft
Intended status: Standards Track
Expires: July 21, 2016

R. Shekh-Yusef
Avaya
Y. Sheffer
Intuit
January 18, 2016

HTTP Secure Remote Password (SRP) Authentication Scheme
draft-yusef-httpauth-srp-scheme-02

Abstract

This document defines an HTTP Authentication Scheme that is based on the Secure Remote Password (SRP) protocol. The SRP protocol is an Augmented Password Authenticated Key Exchange (PAKE) protocol suitable for authenticating users and exchanging keys over an untrusted network.

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 July 21, 2016.

Copyright Notice

Copyright (c) 2016 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. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	2
1.1.	Terminology	3
2.	Operations Overview	3
3.	Protection Space	5
4.	Discovery Request	6
5.	Initial Request	6
6.	Challenge	6
7.	Response	8
8.	Confirmation	9
9.	Username Hashing	10
10.	Internationalization Considerations	10
11.	Integration with Other Protocols	10
12.	Security Considerations	10
13.	IANA Considerations	10
14.	Normative References	11
	Authors' Addresses	11

[1.](#) Introduction

Some protocols (e.g. HTTP [HTTP-P7], SIP [[RFC3261](#)], OAUTH 2.0 [[RFC6749](#)], and STUN [[RFC5389](#)]) use a general framework for access control and authentication, via a 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.

Many of these systems that use the challenge-response framework rely

on passwords chosen by users which usually have low entropy and weak randomness, and as a result cannot be used as cryptographic keys. While cannot be used directly as cryptographic keys, the passwords can still be used to derive cryptographic keys.

This document defines an HTTP Authentication Scheme that is based on the Secure Remote Password (SRP) protocol. The SRP protocol is an Augmented Password Authenticated Key Exchange (PAKE) protocol suitable for authenticating users and exchanging keys over an untrusted network, based on a shared password, without requiring a Public Key Infrastructure (PKI) or any trusted third party.

The SRP protocol provides many security benefits: it resists dictionary attacks mounted by either passive or active network intruders. It also offers perfect forward secrecy, which protects past sessions and passwords against future compromises. Finally, user passwords are stored in a form that is not plaintext-equivalent to the password itself, so an attacker who captures the password database cannot use it directly to compromise security and gain immediate access to the host.

1.1. Terminology

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](#)].

2. Operations Overview

The SRP protocol relies on finite field $GF(\text{large-prime})$, which means that during its setup, the server must choose a large prime number (large-prime), for the integers from 1 to (large-prime - 1) to form a group under multiplication modulo large-prime. The server must also choose a primitive root modulo large-prime (generator) for this group.

All values in this document are computed modulo large-prime.

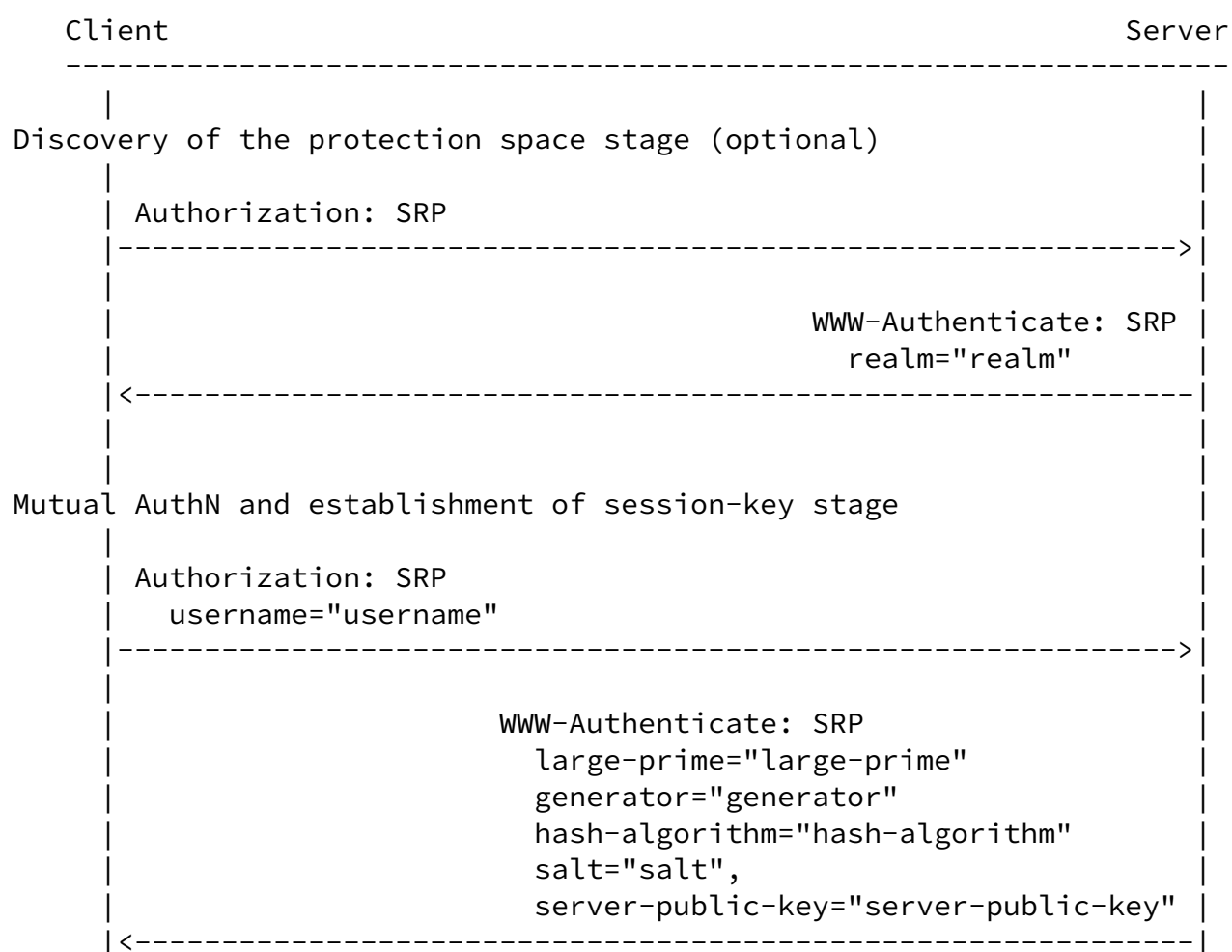
When a user account is created, the server selects a hash function and a user salt, and uses a realm and the user password to create a

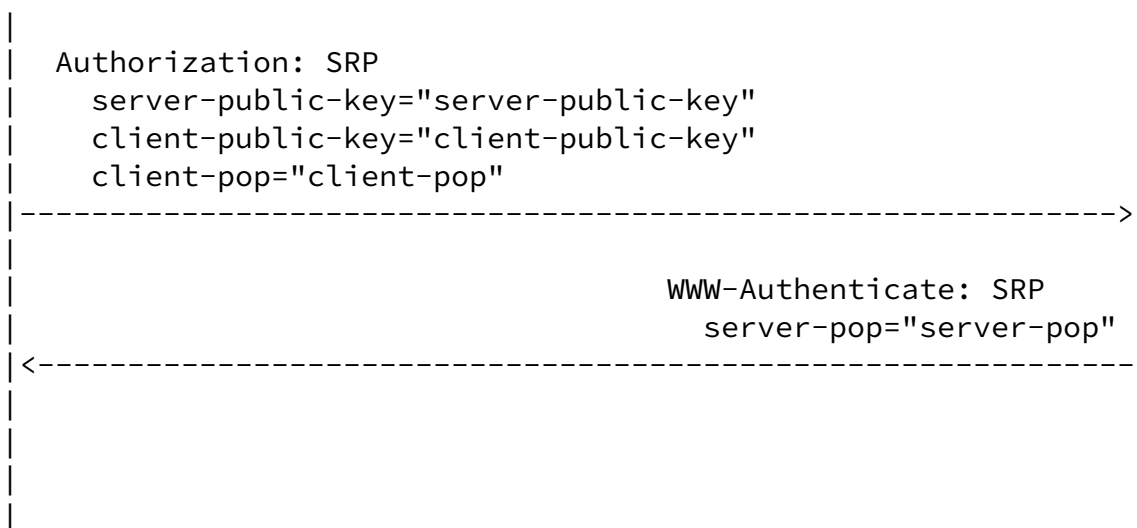
password-verifier as follows:

$$\text{derived-private-key} = H(\text{username}:\text{realm}:\text{password}:\text{salt})$$
$$\text{password-verifier} = \text{generator} ^ \text{derived-private-key}$$

The server then discards the derived-private-key and stores the following information in the database: username, hash-algorithm, salt, and password-verifier.

The following flow describes at a high-level the flow of messages based on the challenge-response framework:





If the client is not aware of the realm associated with the resources it is trying to access, then the initial request will include the SRP scheme with no parameters. This will allow the server to discover the client's capabilities and to challenge the request and provide the client with the realm associated with the resource. The client is then expected to retry the request with the username parameter.

The client then initiates the authentication process with the server by sending an initial request with an Authorization header with SRP scheme which includes the username parameter associated with that protection space.

The server generates its private key and calculates its associated public key, then challenges the request and includes the WWW-Authenticate with the large-prime, generator, hash-algorithm, salt, and server-public-key.

The client calculates the session-key and re-tries the request and includes an Authorization header with client-pop to prove to the server that it is in possession of the session-key.

The server verifies the client-pop and calculates the server-pop to

prove to the client that it is in possession of the same session-key.

At the end of the above process, the client and the server would have established a communication channel after completing a mutual authentication, and each side would be in possession of the same session-key.

[3.](#) Protection Space

The HTTP Authentication Framework [[RFC7235](#)] defines "protection space" as a combination of the canonical root URI of the server being accessed and the "realm" value. The "realm" values allow the partitioning of the server resources into a set of protection spaces, each with its own authentication scheme and/or authorization database.

A protection space determines the scope of protection covered by a set of credentials that can be applied automatically. If a prior request has been authorized, then the client may reuse the same credentials for all other requests within that protection space, for a period of time determined by the server.

[4.](#) Discovery Request

If the client is not aware of the realm associated with the resource it needs to access, then the discovery request **MUST** be sent without any parameters, to allow the server to challenge the request and send the realm to the client. The client is then expected to retry the request with the username associated with the realm of the resource being accessed to initiate the authentication process.

[5.](#) Initial Request

The initial request from the client, that starts the authentication process, **MUST** include an Authorization header field with the SRP scheme.

The request MUST include the following parameter:

username

The user's name in a specific realm.

6. Challenge

If a discovery request is received from the client, then the server MUST challenge the request by responding with 401 and MUST include the realm parameter. The client is expected to retry the request and include the username parameter to start the authentication process.

When the server receives the request with the username parameter, the server looks up the hash-algorithm, salt, and password-verifier associated with the username provided by the client in the initial request.

OPEN ISSUE:

{ To prevent an attacker from identifying the usernames in the DB, if a username does not exist in the DB, the server should still go through the motion of attempting to authenticate the user and fail it only during the last step, to prevent the attacker from recognizing the username existence by analyzing how fast the server responds to the initial request. }

The server generates a random number, $[1, \text{large-prime} - 1]$, as a server ephemeral private key (server-private-key), and computes the associated server ephemeral public key. The server MUST generate a

fresh ephemeral private key for each authentication session, even if the request is coming from the same user.

The server calculates the server-public-key as follows:

$\text{server-public-key} = H(3 * \text{password-verifier} + \text{generator} ^ \text{server-private-key})$

The server then challenges the initial request from the client by responding with a "401 Unauthorized" status code and a WWW-Authenticate header field with an SRP scheme. The header field MUST include the following parameters:

large-prime

The large prime used to form the finite field $GF(n)$ group, selected by the server during setup, formatted as a decimal integer.

generator

A finite field $GF(n)$ group generator selected by the server during setup, formatted as a decimal integer.

hash-algorithm

The hash algorithm used to create the session-key, e.g. SHA256.

salt

A random string used as user's salt.

server-public-key

The server ephemeral public key associated with the server ephemeral private key.

The client generates a random number, $[1, \text{large-prime} - 1]$, as a client ephemeral private key (client-private-key), and computes the associated client ephemeral public key as follows:

$$\text{client-public-key} = \text{generator} \wedge \text{client-private-key}$$

The client calculates the derived-private-key, random nonce, session-tag, session-key, and client-pop as follows:

$$\text{derived-private-key} = H(\text{username:realm:password:salt})$$
$$\text{nonce} = H(\text{client-public-key} : \text{server-public-key})$$
$$\text{session-tag} = (\text{server-public-key} - 3 * \text{generator} \wedge \text{derived-private-key}) \wedge (\text{client-private-key} + \text{nonce} * \text{derived-private-key})$$
$$\text{session-key} = H(\text{session-tag})$$
$$\text{client-pop} = H(\text{client-public-key} : \text{server-public-key} : \text{session-tag})$$

The client is expected to retry the request passing an Authorization header field with SRP scheme. The header field MUST include the following parameters:

server-public-key

The server ephemeral public key that the client received from the server with the challenge request.

client-public-key

The client ephemeral public key associated with the client ephemeral private key.

client-pop

A client proof-of-possession to prove to the server that the client is possession of the session-key.

8. Confirmation

The server MAY use the server-public-key received from the client to correlate this request with the previous response it sent to the client. This is especially important in the case that this request is received on a different connection than the one that delivered the initial request.

The server calculates a random nonce and the session-tag as follows:

$$\text{nonce} = H(\text{client-public-key} : \text{server-public-key})$$
$$\text{session-tag} = (\text{client-public-key} * \text{password-verifier}^{\text{nonce}})^{\text{server-private-key}}$$

At this stage, the server has enough information to verify the client-pop by locally calculating the expected-client-pop and comparing it to the one received from the client. The server calculates the expected-client-pop as follows:

$$\text{expected-client-pop} = H(\text{client-public-key} : \text{server-public-key} : \text{session-tag})$$

The server then compares the expected-client-pop to the client-pop. If they are different, then the server MUST fail the request by responding with a "401 Unauthorized" status code and follow the same procedure used with the initial request.

If the expected-client-pop is the same as the client-pop received from the client, the server continues the process and calculates its server-pop and session-key as follows:

$$\text{server-pop} = H(\text{client-public-key} : \text{client-pop} : \text{session-tag})$$
$$\text{session-key} = H(\text{session-tag})$$

The server then confirms the client's request by responding with 200 OK request. The 200 OK message MUST include an Authentication-Info header field with the SRP scheme. The header field MUST include the following parameter:

server-pop

A server proof-of-possession to prove to the client that the server is possession of the session-key.

When the client receives the confirmation from the server it verifies the server-pop by calculating the expected-server-pop and comparing it to the server-pop. If these values are different, then the client MUST consider the authentication process to have failed; otherwise, the authentication process is complete, and both sides would be in possession of the same session-key.

9. Username Hashing

TBD: <https://tools.ietf.org/html/rfc7616#section-3.4.4>

10. Internationalization Considerations

TBD: <https://tools.ietf.org/html/rfc7616#section-4>

11. Integration with Other Protocols

12. Security Considerations

13. IANA Considerations

14. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", [RFC 2945](#), September 2000.
- [SRP6] Wu, T., "SRP-6: Improvements and Refinements to the Secure Remote Password Protocol", IEEE P1363 Working Group <http://srp.stanford.edu/srp6.ps>, October 2002.

Authors' Addresses

Rifaat Shekh-Yusef
Avaya
250 Sidney Street
Belleville, Ontario
Canada

Phone: +1-613-967-5267
EMail: rifaat.ietf@gmail.com

Yaron Sheffer
Intuit
4 HaHarash St.
Hod HaSharon 4524075
Israel

EMail: yaronf.ietf@gmail.com

