HTTPAuth
Internet-Draft
Intended status: Standards Track
Expires: May 28, 2016

R. Shekh-Yusef
Avaya
Y. Sheffer
Intuit

November 25, 2015

HTTP Secure Remote Password (SRP) Authentication Scheme
draft-yusef-httpauth-srp-scheme-01

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 May 28, 2016.

Table of Contents

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, allowing even weak passphrases to be used safely. 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

During its setup, the server must choose a large-prime, and as a result the integers from 1 to (large-prime - 1) form a group under multiplication mod large-prime, and a generator for this group is also chosen.

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:

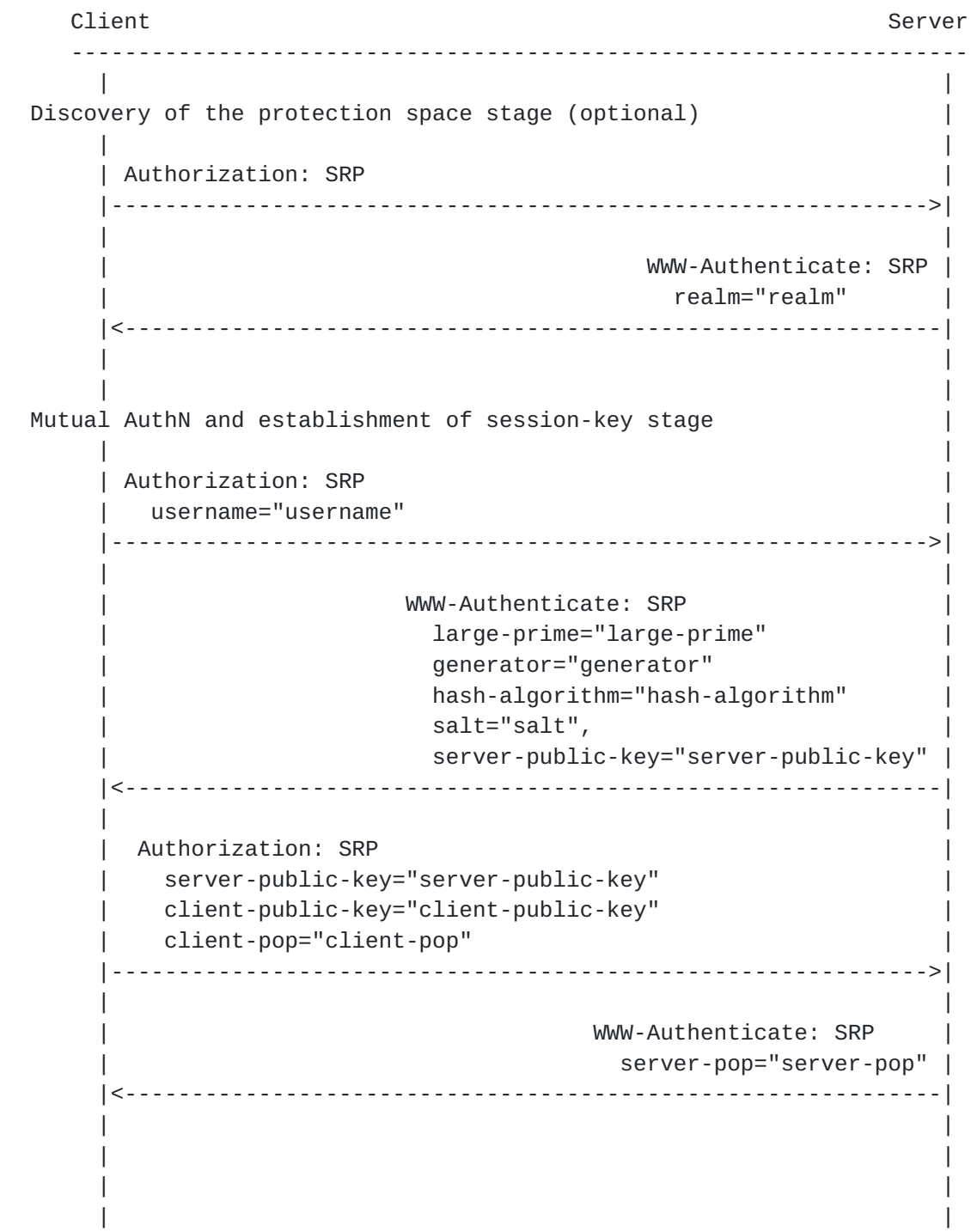
```
derived-private-key = H(username:realm:password:salt)

password-verifier = generator ^ derived-private-key
```

Note that all values in this document are computed modulo large-prime.

The server then stores the following information in the database:

- username
- hash-algorithm
- salt
- password-verifier

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



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 authentication scheme.

If the client is aware of the realm associated with the protection space it is trying to access, then the client initiates the communication to the server by sending an initial request with an Authorization header with SRP scheme which includes the username parameter associated with that protection space.

Otherwise, 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 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 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. Initial Request

The initial request from the client MUST include an Authorization header field with the SRP scheme.

If the client is aware of the realm associated with the resource it is trying to access, then the client MUST include the following parameter:

username

The user's name in a specific realm.

Otherwise, the initial 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.

4. Challenge

If the initial request received from the client does not have a username parameter, 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.

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, 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:

server-public-key = H(3 * password-verifier + generator ^ 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 and 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.

5. Response

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

client-public-key = generator ^ client-private-key

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

derived-private-key = H(username:realm:password:salt)

nonce = H(client-public-key : server-public-key)

session-tag = (server-public-key - 3 * generator ^ derived-private-key) ^ (client-private-key + nonce * derived-private-key)

session-key = H (session-tag)

client-pop = H(client-public-key : server-public-key : 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.

6. 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 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:

nonce = H(client-public-key : server-public-key)

session-tag = (client-public-key * password-verifier ^ nonce) ^ 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:

expected-client-pop = H(client-public-key : server-public-key : session)

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:

server-pop = H(client-public-key : client-pop : session-tag)

session-key = H (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.

7. Username Hashing

TBD: https://tools.ietf.org/html/draft-ietf-httpauth-digest-19#section-3.4.4

8. Internationalization Considerations

TBD: https://tools.ietf.org/html/draft-ietf-httpauth-digest-19#section-4

9. Integration with Other Protocols

10. Security Considerations

11. IANA Considerations

12. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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