Common Authentication Technology Next Generation
Internet Draft
Intended status: Standard
Expires: March 2018

B. Williams
Independent

September 26, 2017

Open Password Automation Recipe (OPAR) Protocol
draft-bwilliams-kitten-opar-00.txt

Abstract

This document introduces the Open Password Automation Recipe (OPAR) Protocol, a suggested format for providing instructions to password managers programmatically that defines the valid construction of a password. With password managers becoming integrated into browsers, websites should be able to describe the elements of an acceptable password programmatically so password management plugins can read and automatically generate the best option. Without this, manual intervention by the user is required when the password is too long, contains invalid characters, or does not meet complexity requirements.

Status of this Memo

Table of Contents

1. Introduction

Password managers are becoming the norm as both Apple and Google have embedded them into their operating systems and browsers. The goal addressed by password managers is to enable users to use different username/password combinations on every site they visit, while locally storing all of those items encrypted. When a user visits a site, stored passwords are automatically presented for authentication.

Password reuse is a known vulnerability in authentication systems and can lead to identity theft[2]. With several available password management systems such as 1Password, LastPass, and the Apple Keychain, a standardized, programmatic way to read the password policy of a website would further automate these applications. Password managers embed functionality into popular browsers to suggest secure passwords and manage them across devices.

The challenge with password generation is that not all sites use the same requirements to create a password, and individuals will often not maximize the strength of a new password as they take the default suggested value. As an example, a site that suggests a password of 8-20 characters might receive “9ay-mgr-3PO-iaa” from a password manager, five characters short of the maximum. In addition, the site may not permit hyphens in passwords, yet still require a special character. Users would need to randomly create one to comply with the standard.

This document proposes the Open Password Automation Recipe (OPAR) Protocol as a way to define valid password recipes for password managers while improving user experience. OPAR is a simple protocol that can be implemented on any page where password generation is required (such as a sign up or change password page) to inform password managers of the acceptable format for a valid password. Then the password manager can suggest and fill in the strongest possible password without requiring user intervention to tweak the recipe.

Implementing and Executing the OPAR Protocol:

1. Site operator adds markup or JSON in the format below to pages requiring password automation.

2. Password manager interprets format to automatically suggest a password that maximizes password strength according to the recipe.

3. Submit the password while storing credentials inside the password manager.

2. Conventions used in this document

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 RFC 2119[1].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these keywords.

3. Construction of OPAR Policy

This section defines the construction of a JSON object that declares the options in the OPAR protocol.

3.1. OPAR_Policy - OPAR Version 1

The JSON object that stores the OPAR values MUST be named OPAR_Policy. This object SHOULD be included only on the new password and change password pages.

The first element in the JSON object MUST be named “version” with an integer value corresponding to the value of the protocol. The only value that is acceptable at this time is 1, but future versions of the protocol may add additional features.

All of the elements below MUST be included in the OPAR object definition to be considered a valid implementation of version 1 of the protocol. Some elements, however, MAY be listed at 0 (say for example if there is no minimum number of special characters), but the element MUST be present in the object.

The following sub-sections describe the remainder of the OPAR record options available in version 1 of the OPAR protocol.

3.1.1. “min_length” – Integer value

Password recipes require a minimum length to be valid. The integer value in this field represents the absolute minimum number of characters required for a valid password.

3.1.2. “max_length” – Integer value

The integer value here should be the maximum number of characters allowed. Password managers should focus on this value and elect to fill every available position in the password with a character.

3.1.3. “numbers” – Object

The numbers object will define the options for numbers in the password recipe.

3.1.3.1. “allowed” – Boolean value

If numbers are allowed, set the value to true. If not, set the value to false.

3.1.3.2. “minimum” – Integer value

If numbers are required, enter the minimum amount of single digit integers that will constitute a valid password. If numbers are allowed but not required, set this value to 0.

3.1.4. “lowercase” – Object

The lowercase object will define the options for lowercase letters in the password recipe.

3.1.4.1. “allowed” – Boolean value

If lowercase letters are allowed, set the value to true. If not, set the value to false.

3.1.4.2. “minimum” – Integer value

If lowercase letters are required, enter the minimum amount of single digit lowercase characters that will constitute a valid password. If lowercase letters are allowed but not required, set this value to 0.

3.1.5. “uppercase” – Object

The uppercase object will define the options for uppercase letters in the password recipe.

3.1.5.1. “allowed” – Boolean value

If uppercase letters are allowed, set the value to true. If not, set the value to false.

3.1.5.2. “minimum” – Integer value

If uppercase letters are required, enter the minimum amount of single digit uppercase characters that will constitute a valid password. If uppercase letters are allowed but not required, set this value to 0.

3.1.6. “special_characters” – Object

The special_characters object will define the options for special characters in the password recipe.

3.1.6.1. “allowed” – Boolean value

If special characters are allowed, set the value to true. If not, set the value to false.

3.1.6.2. “valid_characters” – String value

A single string value that contains one of every special character. This should be represented as a set of valid special characters that make up a password. Only include special characters that may be used in the password. Remember to escape double quotes if that character is allowed.

3.1.6.3. “minimum” – Integer value

If special characters are required, enter the minimum amount of single digit special characters that will constitute a valid password. If special characters are allowed but not required, set this value to 0.

3.1.7. “wide_characters” - Object

Some locales may require or desire the use of so called wide or multibyte characters. This object will define the options for wide characters in the password recipe.

3.1.7.1. “allowed” – Boolean value

If wide characters are allowed, set the value to true. If not, set the value to false.

3.1.7.2. “minimum” – Integer value

If wide characters are required, enter the minimum amount of single digit wide characters that will constitute a valid password. If wide characters are allowed but not required, set this value to 0.

3.1.8. “include_extended_ascii” – Boolean value

Some locales may leverage characters from the extended ASCII character set, such as Cyrillic, accented Western characters, and Greek characters. If you want to allow these characters, set this value to true. There are no minimums here as these would simply extend the uppercase and lowercase character sets.

3.2. Examples

The following sections provide two examples of JSON objects that define a password recipe.

3.2.1. Password Recipe Example 1

The following OPAR password recipe defines a valid password that must be at least 8 characters, but no more than 20, with numbers, lowercase letters, and uppercase letters all allowed (minimum two of each), and only these special characters allowed (+ - _ ( ) * & ^ % $ # @ ! ?), minimum 2. No wide characters are allowed, but the extended ASCII set is permitted.

```
{
   "version":1,
   "min_length":8,
   "max_length":20,
   "numbers":{
     "allowed": true,
     "minimum": 2
```

```
      },
      "lowercase":{
        "allowed": true,
        "minimum": 2
      },
      "uppercase":{
        "allowed": true,
        "minimum": 2
      },
      "special_characters":{
        "allowed": true,
        "valid_characters": "+-_()*&^%$#@!?",
        "minimum": 2
      },
      "wide_characters":{
        "allowed": false,
        "minimum": 0
      },
      "include_extended_ascii": true
   }
```

3.2.2. Password Recipe Example 1

The following OPAR password recipe defines a valid password that must be at least 6 characters, but no more than 12, with numbers, lowercase letters, and uppercase letters only allowed (minimum one of each). No special characters, extended ASCII, or wide characters are permitted.

```
   {
      "version":1,
      "min_length":6,
      "max_length":12,
      "numbers":{
        "allowed": true,
        "minimum": 1
      },
      "lowercase":{
        "allowed": true,
        "minimum": 1
      },
      "uppercase":{
        "allowed": true,
        "minimum": 1
      },
      "special_characters":{
        "allowed": false,
        "valid_characters": null,
        "minimum": 0
      },
      "wide_characters":{
        "allowed": false,
        "minimum": 0
```

```
    },

    "include_extended_ascii": false

}
```

4. Security Considerations

In order for the recipes to be effective, site managers must ensure that the recipe promotes the maximum password complexity possible. Ideally, two things should happen. Site managers should review existing password storage capabilities to maximize the strength of new passwords, and recipes should mirror the instructions given to users in English on how to construct a secure password.

Password managers should focus on the largest effective strength of password based on the maximum allowed character positions given the defined acceptable keyspace (allowed characters) to provide maximum password effectiveness[3].

5. IANA Considerations

No IANA considerations required.

6. Conclusions

The OPAR Protocol is designed to improve automation and usability of password managers used by individuals at large. Through this additional automation, users will not have to fight password managers and may more readily adopt them as their user experience improves.

7. References

7.1. Normative References

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

7.2. Informative References

[2] Gaw, S., & Felten, E. W. (2006, July). Password management strategies for online accounts. In Proceedings of the second symposium on Usable privacy and security (pp. 44-55). ACM.

[3] O'Gorman, L. (2003). Comparing passwords, tokens, and biometrics for user authentication. Proceedings of the IEEE, 91(12), 2021-2040.

8. Acknowledgments

Special thanks to Matt Springfield for being a sounding board.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses
   Branden Williams
   Independent
   2450 Lakeside Parkway
   Suite 150-1026
   Flower Mound TX 75022

   Email: ietf@brandenwilliams.com