**Software Requirements Specification**

**2 – Factor Authentication for Firefox Accounts**

* Introduction
  + Problem Statement
  + Purpose
  + Scope
  + Goals
  + Learning Goals
  + Metrics
  + Definitions
  + System overview
  + References
* Overall description
  + Product perspective
    - User Interfaces
    - Operations
  + Site Adaptation Requirements
  + Constraints and assumptions
* Specific requirements
  + User requirements
  + Content-Server requirements
  + Auth-Server requirements

**1. INTRODUCTION**

1. **Problem statement:**

Reduce the ability of attackers to get into another users account. To reduce the chances of accounts getting compromised due to reuse of passwords across multiple sites.

1. **Purpose:**

To use as a learning experiment that determines the viability of 2-factor authentication for Firefox Accounts, and to determine the desirability of the different methods of 2- factor authentication from the user POV.

1. **Scope:**

The 2-Factor Authentication will be set up only for the Desktop version of Firefox and does not include Firefox OS or Android version. This will require us to make modifications to the content-server, the auth-server and set up endpoints for the transmission and collection of data related to the One-Time-Password (OTP) generation.

1. **Goals:**
2. Lay out the infrastructure for implementing and extending 2FA. (Requirement - Must have)
3. Implement Google Authenticator based 2FA. (Requirement - Must have)
4. Implement SMS as a backup option. (Nice to have)
5. Implement E-Mail as a backup option. (Nice to have)
6. Implement WhatsAuth as a backup option. (Nice to have)
7. **Learning Goals**

* Learn whether the 2-FA can be used across all Firefox platforms.
* Learn about older device and Fennec compatibility.
* Learn about the TOTP Algorithm.
* Learn what is needed to implement the TOTP Algorithm and GoogleAuth.
* Learn the methods of 2-FA used in other applications like gmail, facebook etc.
* Learn about the endpoints required for the integration of 2-FA.

1. **Metrics**

* Product Usage and success/failure rates
  + Counts of users who turn on 2FA.
  + Counts of users who turn on 2FA as part of First run flow.
  + Frequency of entering OTP.
  + Desirability of GoogleAuth, SMS, Email from User POV(survey?)
  + Entry points
* Counts of encountering error/success states: 4xx, 5xx
* Counts of slow queries/slow performance

1. **Definitions**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| OTP | One-Time-Password; a throwaway key generated on the fly. |
| 2FA, 2-FA | Two Factor Authentication; a method for implementing an additional layer of authentication for a user. |
| GoogleAuth | Google Authenticator; an app that allows us to store secret keys to be used for OTP generation. |
| TOTP | Time-based OTP Algorithm; uses the current system time to generate a OTP. |
| HMAC | Hash message authentication code; a component used to generate a OTP. |
| QR Code | Quick Response Code; A type of matrix barcode (or two-dimensional barcode) which is a machine-readable optical label that contains information about the item to which it is attached. |

1. **System Overview**

* The 2-FA can be turned on by the user during the ‘first run’ flow, or from the settings/preferences page of the Sync tab.
* When the 2-FA is turned on for the first time, a secret key is generated for the user, and the user is greeted with a QR Code generated using the secret key and the username of the user (in the format username@domain.com).
* The user then scans the generated QR Code with a mobile phone running the Google Authenticator app.
* If the GoogleAuth app recognizes a valid TOTP QR, it sets itself up to generate the OTP required for the 2-FA. This is confirmed once by our 2-FA flow, to ensure that setup was proper.
* When the user then signs in to Sync on a new device or turns on Sync for an existing device (maybe after turning it off previously), the user is presented with his regular username/password fields. If the user is able to authenticate himself with these credentials, he/she is then presented with a field to enter the OTP.
* If the user is able to enter the OTP generated by the GoogleAuth app into this field and it validates, we authenticate him/her and set up Sync on the device.

1. **References:**
2. <http://garbagecollected.org/2014/09/14/how-google-authenticator-works/> - Very good introductory article on Google Authenticator.
3. <https://www.twilio.com/blog/2013/04/add-two-factor-authentication-to-your-website-with-google-authenticator-and-twilio-sms.html> - Detailed article about implementing the GA and sms based 2FA.
4. <http://twilio-tfa.herokuapp.com/> - Demo link from the above article.
5. <https://www.npmjs.com/package/otp-key> - Node based otp-key generator. This is a pseudo random key generator, and works pretty well.
6. <https://www.npmjs.com/package/otp> - Node based OTP generator library, and implements RFC 6238 pretty accurately.
7. <http://tools.ietf.org/html/rfc6238> - The RFC for TOTP - very straightforward and well documented.

**2. OVERALL DESCRIPTION**

* 1. **Product perspective**
* User Interfaces
  + Option to turn on 2-FA in settings page.
  + Option to turn on 2-FA in first run flow.
  + QR code generation page.
  + QR code verification page.
* Operations
  + Turn on/off 2-FA
  1. **Site Adaptation Requirements**
* Modify the user signup flow for Sync to display option for 2-FA.
* Modify the user login flow for Sync to accommodate 2-FA.
* Add verification logic to Sync flow on auth server.
* Make changes to the auth-server and database to include secret key for the user.
* Add logic to display 2-FA option only on supported devices.
  1. **Constraints and assumptions**
* We assume that we are developing only for newer Firefox versions.
* We will not be supporting FxAndroid or older Fennecs.
* We will limit the scope of this to the learning goals and use it as a test for finding the viability of integrating this into other platforms.

**3. SPECIFIC REQUIREMENTS**

**i. User Requirements**

* Installation of Google Authenticator
  + Install includes a QR code reader.
  + Authenticator generates a one time password (OTP) which is entered into web form.

**ii. Content Server requirements:**

* Form: sign in form to include google authenticator 6 char OTP field
* Settings: ability to toggle 2fa on/off
* Requirements for implementing Google Auth:
  + The secret key is a 32 byte sequence that we need to generate for each user who signs up for 2FA.
  + We need to generate a QR containing the "secret-key" along with the [username@domain.com](mailto:username@domain.com).
  + Generation and verification of TOTP codes in a given 30 second interval.

**iii. Auth-Server requirements:**

* Need a new column in db to store 2fa secret key.
* Auth server needs API to accept 2fa OTP value, check and verify it.