Software Requirements

Specification

for

Aerospace Corp (Mustafa)

Prepared by

Chris Orcutt
Jonathan Woong
Rossen Chemelekov
Mengfei Wi
Vu Le
Anthony Nguyen (703877763)

University of California, Los Angeles

April 22, 2016

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
SRS Creation	4/15		V1
SRS Revision 1	4/22	Jae said so	V2

1 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the **2FA** software product, which will be version 1 and the first release. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

The software system will be a 2-factor authentication system for websites with log-in workflow. This system will provide a system which enables 2-factor authentication (2FA) on web sites using primarily by using touch ID on iOS devices, with email based 2-factor codes as a backup method. The application is intended to integrate into Client's websites to protect their user accounts from being hacked.

1.2 Document Conventions

This document follows the IEEE SRS template. Please see the glossary at the end before reading further.

1.3 Intended Audience and Reading Suggestions

This document is intended for Aerospace Corp.'s Mustafa Al-Ammar, as well as Paul Eggert, Jae Lee, other stakeholders, and developers in the open source community who might implement 2FA for their websites..

The document contains 6 sections: Introduction, Overall Description, External Interface Requirements, System Features, Other Nonfunctional Requirements, and Other Requirements. Each section is divided into subsections that talk about different aspects. This document will

provide readers with an overview of how the application will bring benefits to users, how they can use it, and how it is implemented.

Developers should read the Overall Description, System Features, and all the Requirements sections to know the functionality of the application, as well as how to design and implement it.

Aerospace Corp.'s Mustafa Al-Ammar, Professor Paul Eggert, and Jae Lee should read the Introduction, Overall Description, and System Features to know what the application is doing.

Stakeholders should read the Introduction, Overall Description, System Features to understand the overall view and features of the application. It would be useful for them to skim through the Requirements sections. The more information they know, the easier for them to make informed decisions on their investments.

INCLUDE: How different readers should approach this document. Should they read the whole document or should they only read parts of it?

1.4 Product Scope

Any website developers and web application companies can use this application to improve the security for their websites. Individuals and businesses can secure their data because people need 2 authentication factors to log in their accounts. If everything works out, the application can be used nationwide as well as worldwide. Out of scope items include support for Android mobile devices, application-level 2FA, and OS-level 2FA.

INCLUDE: More description about the application. Are there any out-of-scope items?

2 Overall Description

2.1 Product Perspective

This product will be a standalone system that Clients with websites can integrate into their website's pre-existing log-in system. After a Client integrates the system into their website, the website's log-in process for the User will require two steps: the first step being the standard username-and-password authentication and the second step being authentication via the 2FA Server (TouchID). This is accomplished by creating a link between the website and our server, which identifies the User by username and sends a request-TouchID notification to the User's phone.

INCLUDE: Enough details for a reader to get a sense of how the system works. Add high-level diagram of the system (not too much detail… save for product functions)

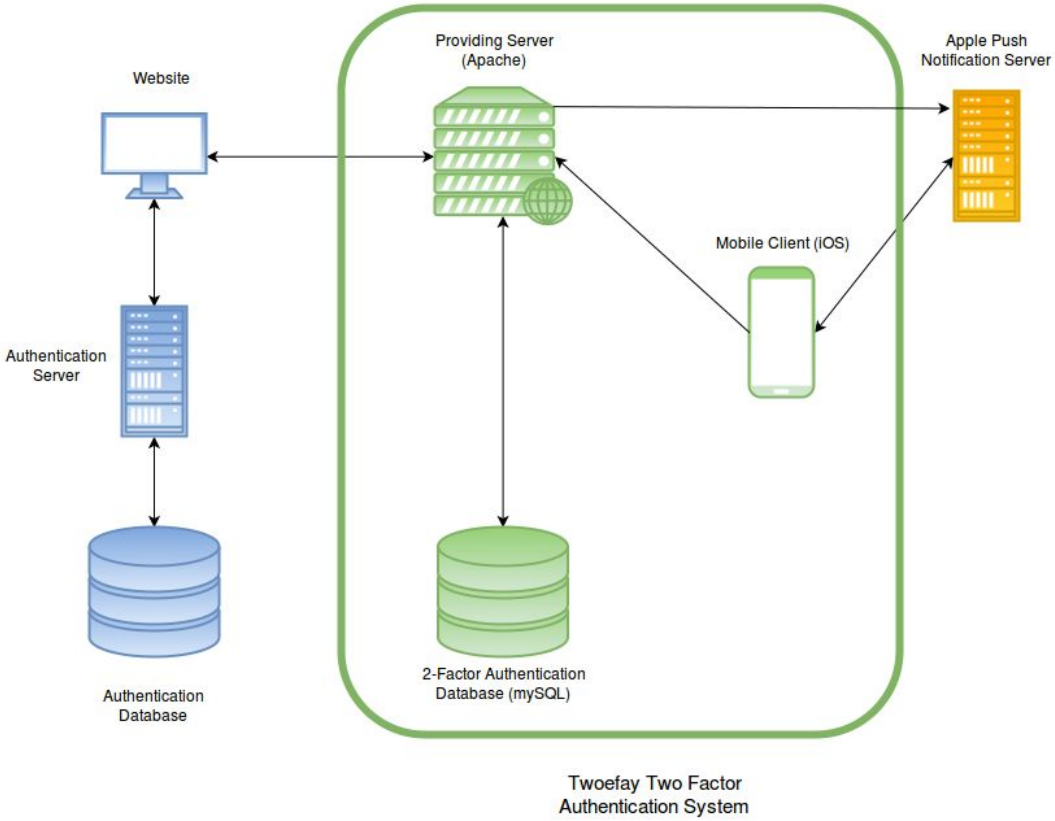

Fig. 1: Architectural representation of the 2FA system and its interaction with other components. In blue is the pre-existing website/web-application service, green is our 2FA system, and yellow is Apple's APN.

2.2 Product Functions

2FA will utilize an Apache server to perform a mySQL database lookup for a token using the User's username as a key. The server will send a request to an Apple Push notification server consisting of the User's token. Apple's Push notification server sends a TouchID notification to the User's phone, requesting fingerprint authorization.

INCLUDE: Enough details for the reader to get a sense of how the system works

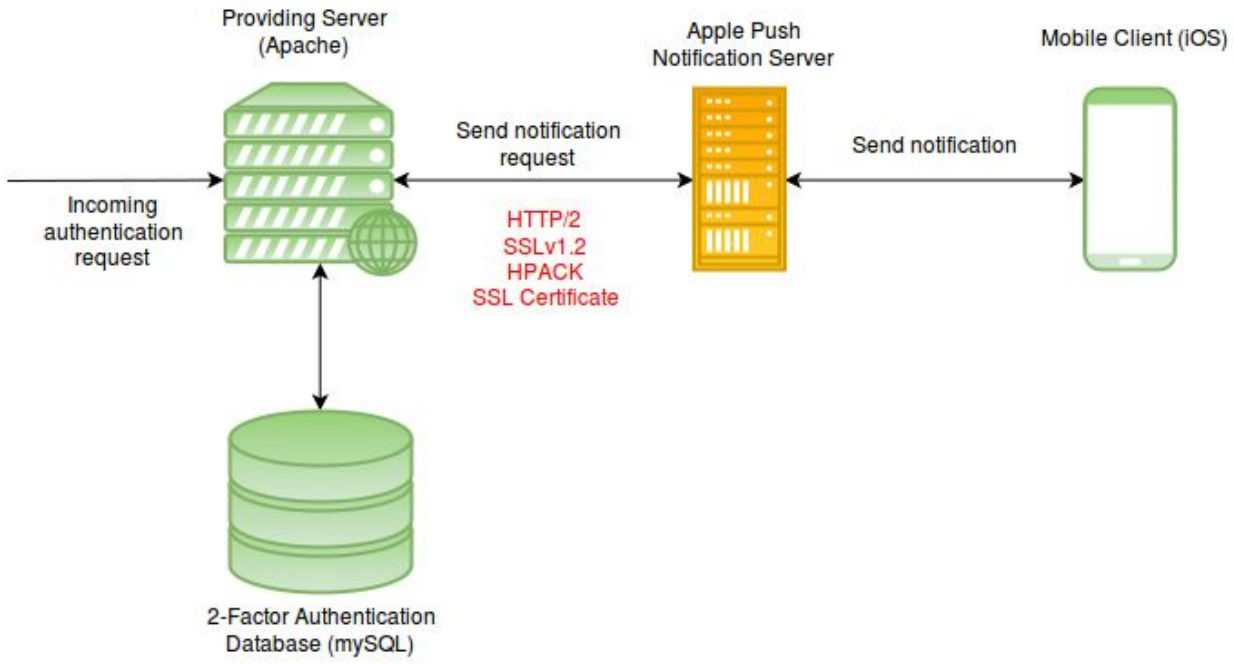

Fig. 2: A more detailed look at the communication between the 2FA system, Apple's APN and The User's iPhone.

When the User's fingerprint is authorized, a message is sent to the Apache server that allows the User to access the Client's website.

If the device is not available, an emailed token can be used as a fallback.

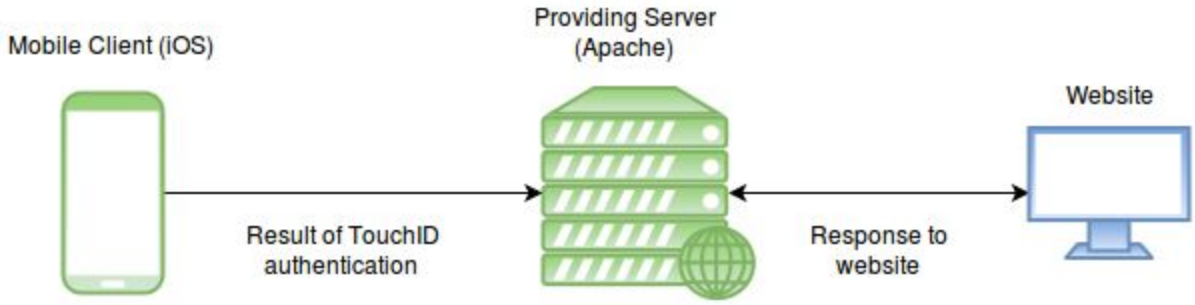

Fig. 3: A high-level demonstration of the login process.

2.3 User Classes and Characteristics

User Class 1: **The User** will be an individual who has a log-in account on a particular website integrated with 2FA.

User Class 2: **The Client** will be a web developer who owns a website with a log-in system.

User Class 1 interacts with the system by providing a means for our server to identify a corresponding token. User Class 2 interacts with the system by utilizing the token to activate fingerprint authentication, which acts as a second form of authentication.

INCLUDE: Brief description on how User Class 1 and User Class 2 interact with the system differently

2.4 Operating Environment

One of the system components, the phone application, will run on Apple's iOS 9.

Apache and MySQL will be used on the server side. A REST style API will be used to communicate with the websites that will implement 2FA.

2.5 Design and Implementation Constraints

The 2FA system must be very easy to deploy and use with minimal setup or tinkering, as per client requirements. The phone application for V1 is for the iPhone only, as Aerospace Corp. only issues iPhones to their employees.

2.6 User Documentation

There should be a help-desk website that The User and The Client can consult for common Q&A about using our services. Suggested https://freshdesk.com/.

2.7 Assumptions and Dependencies

We will assume that the iPhone fingerprint scanning is reliable and secure. We will also assume that the Client's log-in system and database is reliable and secure. A consequence of the Client's log-in system not being secure is the potential for malicious observers to obtain the log-in information of Users, which may allow them to bypass our 2FA system.

Proper communication between the Apache server and Apple Push Server assumes adherence to specific formatting and protocols: HTTP/2, SSLv1.2, HPACK, and SSL Certification. As a consequence, these requirements place a restriction on the communication capabilities of the Apache server.

The functionality of the entire system depends on the availability of our server. This means that there is an assumption that our server will be reliable and secure. A consequence of the server not being reliable is non-functionality of the entire system, and a consequence of the server not being secure is the potential for attackers to obtain usernames that may match the username of the Client's log-in system. In the event that the 2FA server is offline, the Client website can chose between two different fallback modes. In the first fallback mode, they can chose to allow login without the 2nd factor, and in the second fallback mode, they can completely disallow new

logins. If users are already logged in, or if their account is setup to not require the 2nd factor on the device they are accessing from, then they will not notice the second fallback mode.

INCLUDE: Explain why the assumptions are made and if there are any consequences. (Add more assumptions)

3 External Interface Requirements

3.1 User Interfaces

The 2FA App user interface will need primarily three screens to provide an optimal user experience and functionality.

The first should be a login screen for The User to login to the app which will be the method used in our implementation to provide the 2-factor authentication.

The second will be the screen that the The User sees after they login to the app, which displays the application specific authentication tokens generated every 30 seconds by the app, which can be used if the mobile device is offline.

The third will be the confirm/deny access screen that The User will use to verify logins to The Client's website/services, thus serving the dual purpose of providing authentication and an early alert system for unauthorized attempt to access The User's website account.

The user interface will be simple and intuitive. Upon attempting to log into the protected website, the user will be receive a notification prompting her to provide authentication via TouchID, whereupon, having provided a valid fingerprint, the user will be admitted into the website.

A user interface may also be created for the 2FA Server for our internal purposes, but neither The User nor The Client will need to see them.

Lastly, a user interface will be needed for the transition page after The User inputs their username and password.

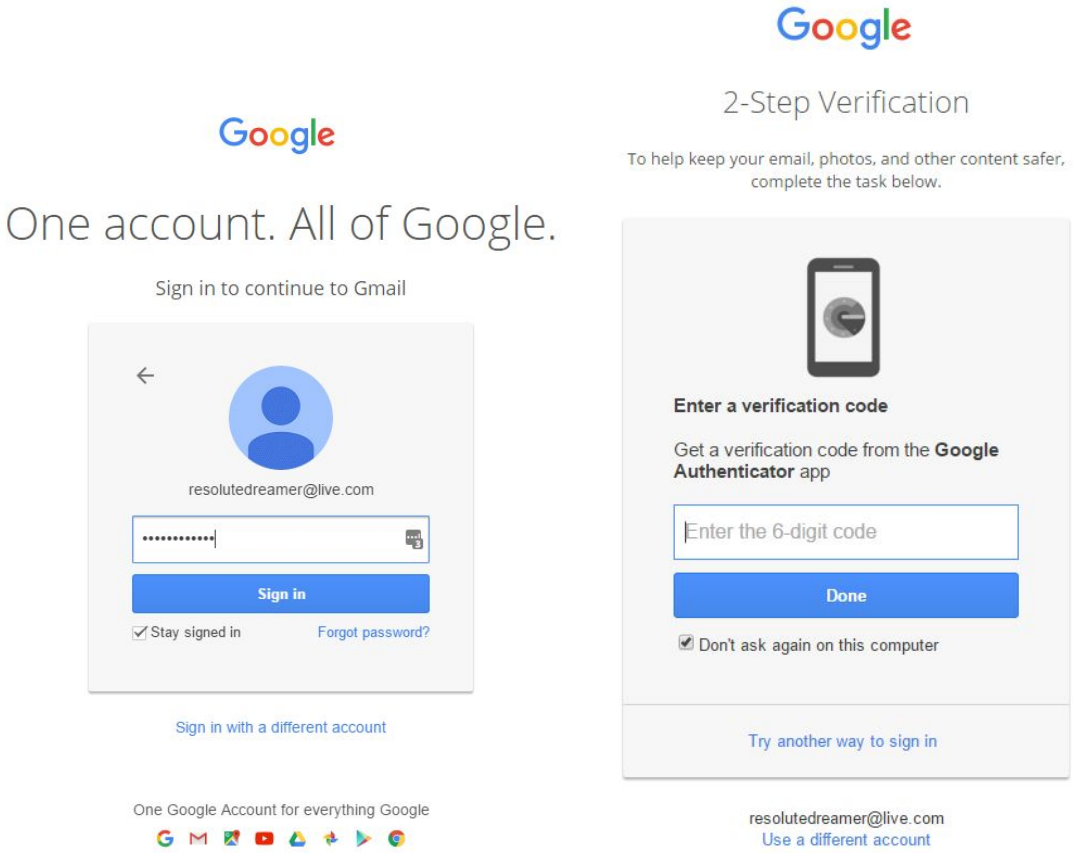

Fig. 4: Example of Google's 2-factor authentication service.

.

If their 2FA App is connected properly, the website will automatically navigate away from this page after the user accepts the login on their 2FA App. This page is an analogous page to the image displayed on the right hand side above. Like in this provided image, this screen is also the screen that allows for The User to put in their verification code which can be used as a fallback

means of logging in if they don't have their phone, or if they have their phone but their phone does not have network access. The "Don't ask again for this computer" button can also be included.

3.2 Hardware Interfaces

The component of our software system that involves hardware devices is the 2FA App. Specifically, the iOS version of the app is intended to work with the fingerprint scanner available on newer iOS devices. However, we do not need to directly interface between our software and the Apple hardware. The Apple Touch ID API handles the actual details of the fingerprint scanner hardware; no actual fingerprint information is given over to the 2FA App.

3.3 Software Interfaces

The 2FA Server will be hosted on a cloud service and need to operate within that environment.

We might consider using the OAuth API to handle the tokens we are passing from our 2FA Server to our 2FA App.

http://oauth.net/

The 2FA App will be distributed through the platform specific App Store for the operating systems we choose to support.

3.4 Communications Interfaces

The 2FA Server will use a RESTful API for communication over HTTPS.

This logon flow is subject to change based on the intended structure of the 2FA Server. This logon flow assumes that the 2FA server is maintained separately from The Client's server.

The User will initiate a login at The Client's website/service. After the login is processed by The Client's server and is authenticated as a valid login, The Client's server will then send a message/token/thingy to the 2FA Server that identifies The User authenticated with them successfully. Once we receive this, the 2FA Server will then send a Push Notification to the 2FA App that The User has logged into. At this point, The User will need to Accept the login on their 2FA App after which the 2FA App will then send back to the 2FA server a confirmation message. Alternatively, if the 2FA App is not able to receive the Push Notification (Battery Saver Mode, Airplane Mode, no internet access, etc.) The User can enter the token generated by the 2FA app and enter that into the logon instead. Lastly, the 2FA App will send a confirmation back to The Client's server that The User authenticated with the second factor successfully.

4 System Features / Functional Requirements

4.1 Use Case 1: The Client integrates Twoefay

4.1.1 Description and Priority

High priority: Communication between The Client and the iOS App must be secured via encryption to prevent attackers from eavesdropping and/or modifying the tokens.

High priority: The Client should be able to set up the website existing log-in system such that before logging the user in, the log-in system would make an API call to the Twoefay server, giving it the username.

High priority: There should be a Twoefay server hosted separately from any Clients for demo purposes.

Medium priority: The Client should be able to view a list of possible API calls in well-documented interfaces.

4.1.2 Functional Requirements

REQ-1: The Twoefay server should accept API calls and perform relevant functions.

REQ-2: The Twoefay server implementation should be able to be copied onto a server at a location of The Client's choice.

4.2 Use Case 2: The User creates account on website

4.2.1 Description and Priority

High priority: The User, upon successfully creating an account on the websites, should receive an email with a link to download the Twoefay mobile application.

High priority: The User, upon downloading the application, should be able to sign up by indicating the website for which he/she has created an account and go through the service initialisation process.

High priority: The service initialisation process should consist of getting an application token for notification services from the Apple notification service APN.

High priority: The token should be stored with the username in a database that is part of the Twoefay system.

4.2.2 Functional Requirements

REQ-1: The email sent to The User should have a link to download the Twoefay mobile application.

REQ-2: The mobile application should have a sign-up form which takes as input which website log-in system The User intends to use and the username.

REQ-3: The mobile application should request a token from Apple APN.

REQ-4: The mobile application should make a request to the Twoefay server to send the username and token.

4.3 Use Case 3: The User logs into account

4.3.1 Description and Priority

High priority: The User, upon visiting the website and signing into their account through the website's original log-in system, should have their cell phone receive a notification.

High priority: The User, upon opening the mobile application, should be requested to authenticate using the fingerprint scanner on the phone.

High priority: The website should log-in The User upon successfully authenticating on The User on his/her mobile application.

Medium priority: The mobile application, upon successfully authenticating The User, should display a "success" message.

Medium priority: The mobile application, upon unsuccessful authentication of The User, should display a helpful error screen.

4.3.2 Functional Requirements

REQ-1: The website should send an authentication request to the Twoefay server upon The User signing into their website account.

REQ-2: The Twoefay server should request a notification to be sent to the mobile application of a specific user upon receiving a request from a website.

REQ-3: The mobile application should request The User to authenticate using Touch ID.

REQ-4: The mobile application should send an authentication accepted message to the Twoefay server upon successful authentication.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The time from The User successfully authenticating with The Client's server to a Push Notification appearing on their 2FA app should be no more than 5 seconds.

The time from The User (responding to the Push Notification) OR (entering their token generated by the 2FA app) to The Client's server receiving the 2FA Server's validation should be no more than 5 second.

Performance is extremely important for our real time system because The User will not be willing to wait over 45 seconds for a webpage to load the desired content.

5.2 Safety Requirements

Apple's iPhone safety regulations with the fingerprint sensor will cover our product's safety requirements.

5.3 Security Requirements

Security is the highest priority for 2FA. The 2FA Server stores information that uniquely identifies The Users and The Clients, and thus it becomes a new vector for potential Attackers who wish to cause harm to either Us, The Clients, or The Users. The 2FA Server must be protected against man-in-the-middle (MITM) attacks, SQL Injections, and other exploitation techniques.

5.4 Software Quality Attributes

Ease of use of 2FA will be the most important software quality aspect for both website developers and Clients who have user accounts. Web developers should be able to quickly add 2FA to their existing infrastructure without the hassle of changing their source code significantly. 2FA should be adaptable and portable across different web technology stacks and reliable in performing the same level of security and functionality for each website and Client.

The product should also be available to anyone who has wishes to add an additional layer of security to their website or web application, since it will be free and open source.

The availability of the service should be close to 100% uptime, otherwise The Users will be locked out of their websites/services.

The product should also be a highly-scalable solution and provide consistently reliable performance even in large companies' environments, consisting of upwards of thousands of users, should management choose to implement the 2FA system.

6 Other Requirements

Appendix A: Glossary

When we refer to "**Us,**" we mean our development team of 6 students working on this project

The "Software Product" that we are developing will be referred as **2FA** (to represent 2-factor authentication) for brevity [Name Subject to Change]. The Software Product has a backend component, which we will refer to as the **2FA Server**, as well as a front end component, which we will refer to as the **2FA App**. This app will most likely be an iOS app.

When we refer to "**The Client**," we mean the website or service owner that will use our product to provide the second authentication factor to their customers/users.

When we refer to "**The User,**" we are referring to the individuals who are trying to log into a website or service secured by "**The Client.**"

When we refer to "**The Attacker,**" we mean any individual or group that wishes to gain unauthorized access to The User's account to obtain private information from any of the other parties.

Appendix C: To Be Determined List

- Web server (Apache vs. Redis/nginx/etc.)
- Language preferences (Java, Python vs. JavaScript, etc).
- Database (MySQL)
- Other

CS 130 Aerospace Security Team Project Timeline

- Key Milestones
    - Built Website with login capability for testing
    - Spun up MySQL Database & Web Service
    - Developed iOS App
        - Push Notifications between the server and the iOS app
    - Integrated Web Service/Client and iPhone apps
    - Security testing
- Plan to Accomplish
    - Midterm (5/5):
        - Research, Design, Learn/familiarize
        - Have login-capable website
        - Start on database + web app + iPhone app
    - Final (6/3):
        - Have basic 2-factor authentication functionality
            - Website login requires user to authenticate via TouchID
        - Tested security + reliability
        - Document functionality and requirements for integration
        - Prepared demonstration of 2-factor authentication
- Reach Goals (i.e. Profit!)