ProxPass

By Brandon Arbuthnot, Joe McCormick, Eric Perez, Michael Hemmelgarn, Hayden Rode

The Problem

- Security from 2FA comes at the cost of convenience
- Authentication = Effort + Time + Money
- Accessibility is paramount

Motivation

- Duo Mobile is a hassle
- The security of 2FA is vital
- Exploits should be far and few
- We want a ‘lazy solution’ that involves minimal effort on the users’ part.

“I will always choose a lazy person to do a difficult job, because he [or she] will find an easy way to do it”

-Bill Gates

The Status Quo

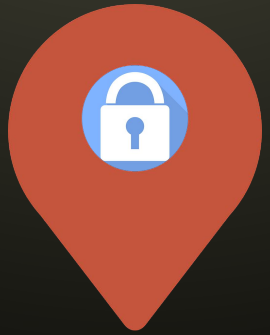
- Find your phone
- Turn on your phone
- Unlock your phone
- Open authentication app
- Record a code
- Enter your code
- Repeat

All Things Considered

- Facial Recognition
- Inaudible Frequencies
- Automatic Refresh
- Physical Hardware Access
- Bypass/Override existing Security Measures

Our InfoSec Solution -

ProxPass



What is ProxPass?

Core Functionality:

A hardware agnostic method of location-based Two-Factor Authentication

Core Requirements:

Wireless Internet access, regular login information and accompanying 2FA device

How it Works

- Authentication is prompted by a login attempt
- **First Factor**: login information is verified against a live server (basic username and password)
- **Second Factor**: Whilst checking those values, the current BSSID wifi identifier is obtained and checked against the whitelisted identifier stored securely
- **Upon Success:** User is logged in
- **Upon Failure**: 2FA authorization code entry is checked

Benefits

Convenience

- Authenticate without the hassle of reaching your device every time
- Transition seamlessly from login to access
- You are likely connected wirelessly regardless
- Login without need for extra auxiliary devices or permissions

Security

- All the benefits of clunkier 2FA methods
- Minimal data is stored and most is replaced as you continue to authenticate
- Failure defaults back to typical methods as a fallback
- Requiring less effort from users limits their exposure to external agents

Anonymity

- Exchange of non-intrusive data
- Universal identifiers obtained from infrastructure
- Permissions are relatively basic
- Stored data very difficult to de-anonymize, has limited value

Obstacles

- Minimizing user effort was a game of seconds
- Cloud messaging service used put a hard limit on priority messaging based on power restrictions
  - Designing for app standby buckets is a nightmare
- Convenience/Security tradeoff was not an enjoyable task
- Despite being hardware agnostic, developing in separate languages made the project relatively disjointed
- No way to achieve our goals without a dedicated app

The Crux - To Whitelist or Not To Whitelist

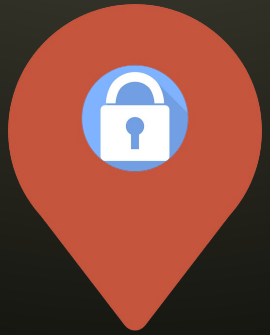
- Prototype flaw - undesired access from previous locations
- Solution - whitelist that would overwrite the last known location

Components

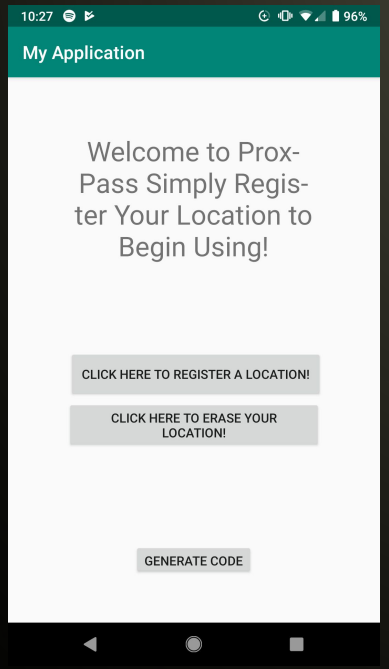
- Languages
  - Java
  - Ruby
  - Python
- Parts
  - Firebase Server
  - Android App
  - Web App

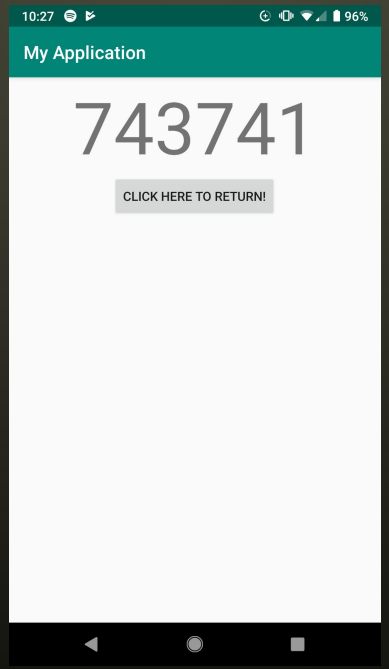
Live Demo

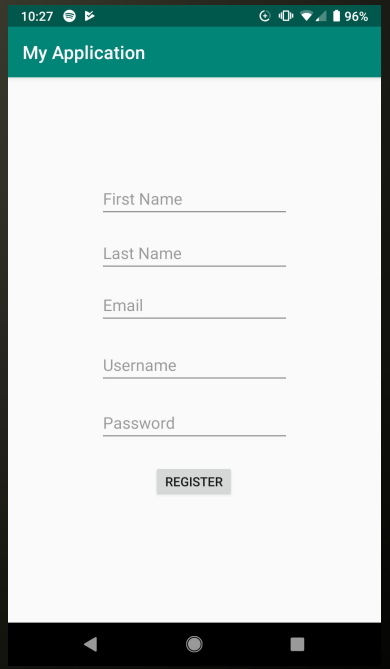
ProxPass



Live Demo - Mobile App Screenshots







Firebase Server

- Codes - used for the web application
  - Username
  - Unique 6-digit code
- Users
  - Pulls from the registration page
  - User info

Android Application

- Main page
  - Code Generate Button
  - Register and Erase location Button
- Registration
  - User Information
- Code Generation
  - 6-digit Unique Code

Web Application

- Proof of concept
  - Demonstration of integration
  - Capable of being adapted to fit specific use (Duo replacement)
- Not a particularly useful implementation
- Compares data stored on server with data obtained by application and provided by user

Web App Implementation

- Used Ruby on Rails
  - Devise Gem for Authentication
  - Google-cloud-firestore gem to connect to Firebase
- First Checks BSSID to authenticate.
- Check OTP second.

Security Assessment

- Remote attackers cannot be at the location, so the system is no different than 2FA
- Stolen devices can only skip authentication if they are at the exact location where the device was located
- Potential issue: user does not proactively erase whitelist
  - Locations are overwritten, but the last location is still whitelisted
  - User should erase whitelist whenever away from their machine in an unsecure location for an extended period of time

Future Work

- Compatibility with other operating systems/mobile devices
- Addition of different location detection methods (bluetooth, ultrasonic audio, etc)
- Revisit multiple whitelisting implementation and consider how to make it more secure (possibly using time to live on each site)

Conclusion

Successes

- Achieved goals
- Base functionality
- Modular design
- Minimum overhead/operational costs

Improvements

- User Interface
- Extra features
- Pattern recognition
- Quicker more acute access