# Kumquat: A Visual Second Factor Alternative in Two-Step Verification

Date: May 26th, 2014

Preksha Kashyap, Jennelle Wong

## Abstract

With password hacking on the rise, two-step verification (2sv) has become an increasingly used measure to protect one's assets and accounts. The use of security questions or a one-time authorization code are common methods to prevent fraudulent behaviour. However, these cumbersome security measures often lead the end user to opt out of these additional features. Kumquat aims to improve the usability of 2sv and in turn, encourage the adoption of 2sv by delivering a more natural second-tier of security through the use of visual stimulus.

## Problem Statement

Design and implement a novel and intuitive second layer of two-step verification that is convenient and does not require the disclosure of personal information.

## Proposal

Web services that deal with sensitive transactions  implement a two factor authentication or two-step verification (2sv), as software giant Google has coined [**1**], as an additional layer of security. However, these features are anything but seamless. Though Google claims it has the largest deployment of 2sv and that it has been adopted by millions of users, it should be noted that even as many as 10 million users only represent a single-digit percentage of their total user base [**2**]. In an effort to mitigate security vulnerabilities, login mechanisms are ostracizing its users through a lack of usability.

### Security

Regardless of the amount of precautionary steps taken to ensure the protection of the user's information, the effectiveness of a security system is only as strong as the weakest link. Often times, that source of weakness stems from the user's inability to thoroughly protect themselves. As a matter of fact, 76% of security breaches stem from the choice of poorly designed passwords [**3**]. An example of weak passwords can be retrieved from the list of most frequently used passwords compiled by SplashData, in which many of the passwords were common words or lacked any real complexity [**4**].

In recent years, measures to counter this have included various implementations of second tier authentication. However, many of these second security measures can be boiled down to receiving a one-time authorization code or to security questions. The former, is highly inconvenient to the user, whereas the latter which are typically based on personal facts and can easily and unintentionally be shared or found out.

Kumquat is designed to be an alternative second factor authentication system that guides the user into choosing a more complicated passphrase without them needing to process any complicated decisions. Specifically, Kumquat uses an image overlaid by a

grid that asks the user to select an area of the image. The coordinates of the encompassed area get stored as the user's password, while the visual stimulus of the picture in the background coupled with the positioning of the grid dots guide the user into remembering what area of the image they selected.

## Usability

The proposed mechanism leverages visual memory to obtain a numeric password that consists of a random sequence of coordinates. While otherwise having to pick and remember the value of the coordinates would be difficult, the user may not have as much difficulty in remembering that they selected a certain object within an image.

The images the user gets to work with will be constructed in such a way that they will be easy to remember. Humans are particularly good at remembering thousands of images, however not just any type of image. Certain images trigger stronger memory associations than other. Specifically, images that contain people, static indoor scenes and human-sized objects were those that were the most easily remembered [**5**].

## Novelty

The system brings a novel spin to the usual two step verification process by obfuscating the actual password that is being saved and enabling a much more interactive and intuitive way for the user to enter in their password. This is all accomplished by using the power of visual stimulus and the predisposition many people have towards remembering images rather than numbers or letters.

## Architecture

The application to which the security measures are being applied to will be a simple Ruby on Rails application, which functions using the model-view-controller architecture, and additionally utilizes JavaScript. By using a web app platform, the application will be able to successfully be accessed from a variety of client-end mediums (laptop, smartphone, desktop). Any sensitive data that is to be stored in the database will be salted and hashed using the Bcrypt ruby gem.

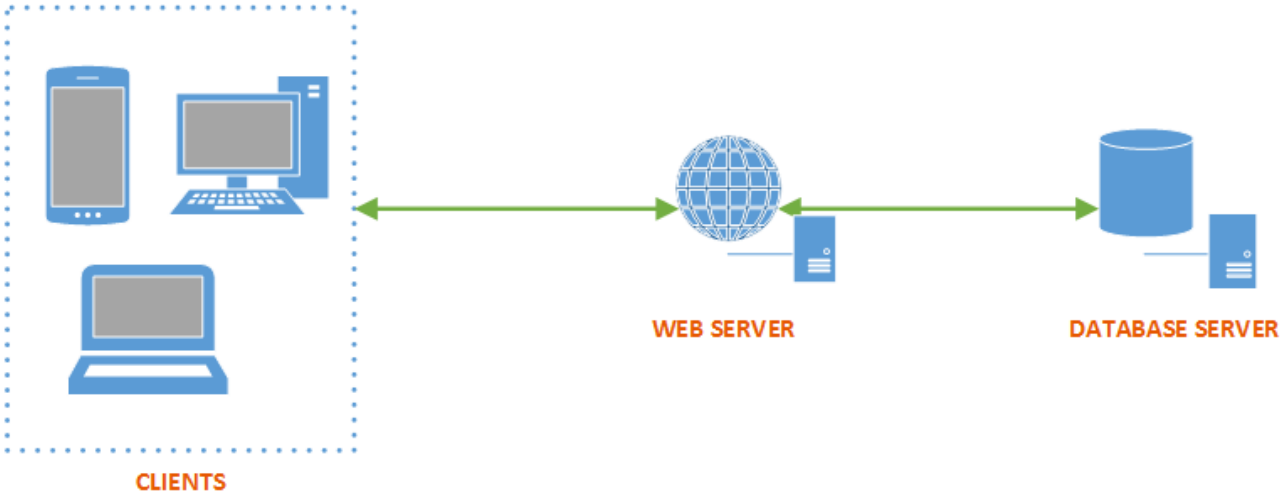


**Figure 1: Application Architecture Diagram**

## System Design

As part of the registration process, after entering in their first level password the portal prompts users to select at least *n+1* images to form their Kumquat password bank, where *n* is the number of images required to actually complete the second factor in a two-step verification process and log in. Next, the user is presented with an image overlaid on a coordinate grid and asked to form two four-sided polygons to highlight a memorable region on the image. Once the user selects and confirms two sets of four coordinates, a password is generated. Note that the user will not be allowed to choose two regions with the exact same coordinates. Each image has its own unique identifier, which is appended to a string containing the selected coordinates. This string is then salted, hashed and stored. The process is repeated until the web service has a bank of *n+1* images.

For every login after completing the registration process, the user will be greeted by a typical username and password form, which is the first layer of authentication. After successfully entering in their login details, the user will then be presented with an image from the bank of images they had previously chosen and set up upon registering.

The user will then select a two groups of four coordinates, effectively highlighting two areas of the image. The coordinates and image identifier are converted to a string and salted to convert the user's input into a password. If validated, the user is prompted to complete a second picture in the sequence.

Kumquat is designed to be a scalable solution. By increasing the number of images, *n*, in a Kumquat sequence, further entropy can be introduced to the login mechanism.

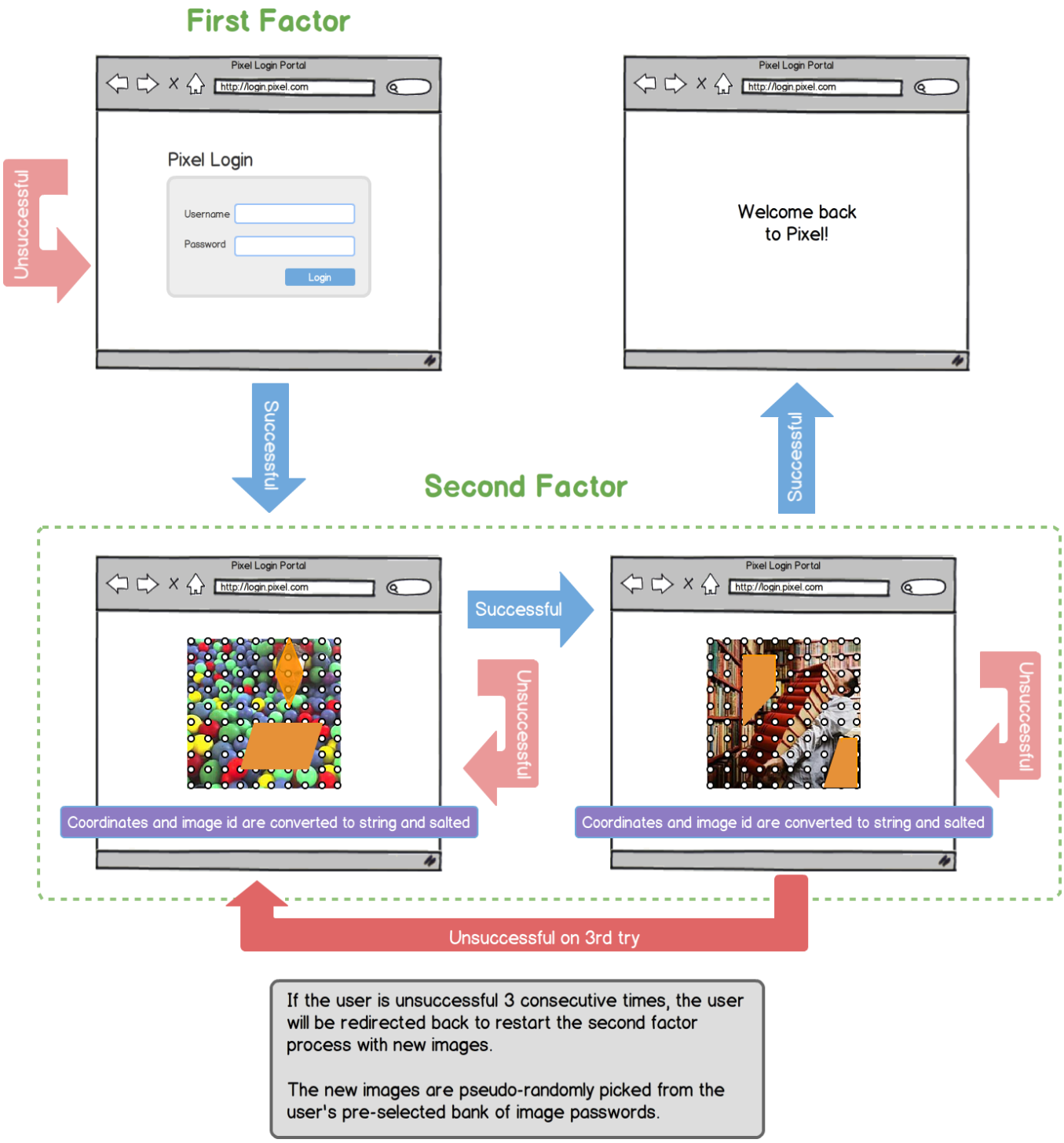


Figure 2: User Flow Diagram of a Two Image Kumquat Process

## Implementation Plan

The proposed two-factor login mechanism will be built into a web application. The group will deliver a basic working login portal with the Kumquat implemented. Ruby on Rails has been chosen to provide a full-stack framework for the project. The advantage of using Ruby on Rails is that it leverages the use of common software engineering patterns and templates to quickly bridge the user facing layer with its database. This will allow the group to divert a greater amount of resources towards building Kumquat since a skeleton portal can be set up with minimal effort.

The first stage of development would be to implement a simple username and password login page. The first line of defense in this web application is to salt and hash the password prior to validating the user.

It is expected that the groundwork for Kumquat will be done in conjunction with the setting up the base web portal. Using HTML5, CSS and JS, a coordinate lattice will be built with an image overlaid. The grid to takes in two sets of four points and converts the eight points into a string. An identifier that corresponds to the current image is appended to said string. The string will then be salted, hashed and validated by the application's backend.

Once the coordinate grid logic is complete, the next stage calls for actually deploying it as the redirected view the user see once the first-tier authentication is successful. The main goal is to allow for a sequence of two Kumquat images as the second factor in a 2sv. At this point, dummy data will have to be used and followed up by extensive testing to ensure that the system works.

If time permits, the group would like to build out the registration process to provide a complete end-to-end proof of concept prototype of Kumquat.

## Test Plan

Testing will occur at two stages of development; when the the coordinate system is initially built and when Kumquat is fully deployed as the second-tier authentication.

At the time of writing the proposal, the group has determined that the project will be enforcing a combination of behaviour-driven development and test-driven development. A table of sample user stories has been compiled for the two mentioned stage of development to outline the expected behaviour of the Kumquat mechanism.

### Kumquat Coordinate Grid

**In order to** generate a password from a coordinate
**As a** password mechanism
**I want to** provide a user interface for users to select their points

**Table 1: Sample BDD Stories**

| Scenario | Narrative |
|---|---|
| **1** | **Given** an empty grid<br>**When** the User selects a number of coordinates<br>**Then** the grid should only accept 8 coordinates |
| **2** | **Given** a grid with 4 or less coordinates selected<br>**When** the user selects additional coordinates other than those selected<br>**And** brings the total selected coordinates to 4<br>**Then** a 4-sided polygon is formed on the grid |
| **3** | **Given** a grid with 4 or less coordinates selected<br>**When** the user selects a coordinates that is already selected<br>**Then** the selection is ignored |
| **4** | **Given** a grid with 4 coordinates selected, completing a polygon<br>**When** the user selects a coordinates that is already selected<br>**Then** the coordinate is valid |
| **5** | **Given** a grid with 4 or less coordinates selected<br>**When** the user selects additional different coordinates other than those selected<br>**And** brings the total selected coordinates to 8<br>**Then** 2 4-sided polygon is formed on the grid |

The group members will also consider employing the use of Ruby gems for additional test coverage. Two gems that are being explored are Brakeman, a static analyser, and Tarantula, a gem that crawls the application and performs data fuzzing.

## Division of Labour

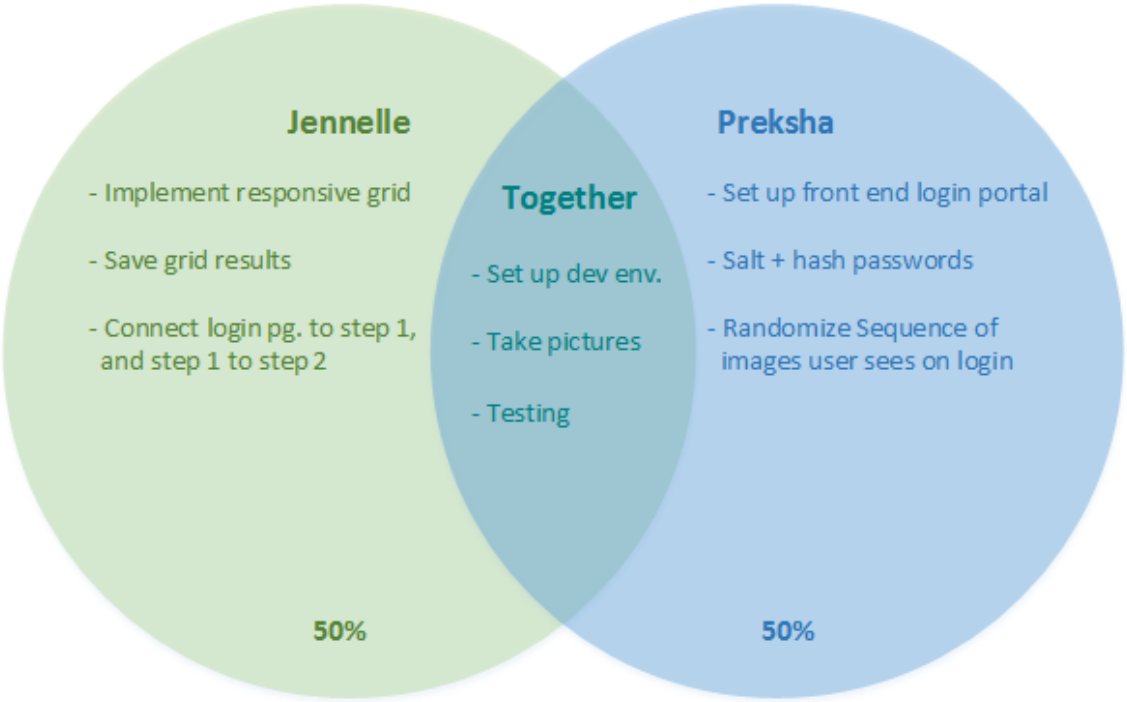


Figure 3: Division of Labour

For this project, the division of labour between both members will be 50%, and the tasks each are given in **Figure 3** above. However, this is simply a tentative designation of the tasks that will be performed and most likely both members will be working collaboratively on every task.

## References


[1] Mayank Upadhyay Eric Grosse. (2013, February) Computer.org. [Online]. http://www.computer.org/cms/Computer.org/ComputingNow/pdfs/AuthenticationAtScale.pdf

[2] John Fontana. (2013, April) ZDNet. [Online]. http://www.zdnet.com/two-factor-authentication-in-two-years-7000013474/

[3] Verizon. (January, 2014) Verizon. [Online]. http://www.verizonenterprise.com/DBIR/

[4] SplashData. SplashData News. [Online]. http://splashdata.com/press/worstpasswords2013.htm

[5] J. Xiao, A. Torralba, A. Oliva P. Isola. (2011, April) What Makes an Image Memorable? [Online]. http://cvcl.mit.edu/papers/IsolaXiaoTorralbaOliva-PredictingImageMemory-CVPR2011.pdf