State of Security

## Pigeon

When thinking about security one thing always comes to mind. “Security must be baked in and not bolted on.” Our team has really taken that into consideration each step of the project and thought about security as we built instead of as an afterthought. The next thoughts we have are toward Integrity the ability for others to modify your data. Confidentiality the ability for others to view your data that hasn’t been explicitly shared with them. Finally, to close these to off and the biggest player for us is availability, which is the biggest focus for us as a team. The availability of data or denial of access is really where we cover all these pieces. Since we will be calling functions to retrieve data from a database full of user information there are several steps, we have taken to secure this data, and verify that only the users that should have access to this data have access.

User Information Security

## Identification, Authentication, Authorization

**Database Assertion Functions:**

We have taken user data access very seriously in our database with several checks initiated on a request. To gain access to any user data you must be logged in as a user on the site. This is done through a login portal. The user that you logged in as is what is used to verify any further access to data. We use the user’s ID, which is unique to each user, to check whether they have access to data from group chats of users, messages, and other types of user stored data that should be accessed.

**Messages:**

Messages have several checks that are done using assertions. First, we verify that the user asking for access has permission to view a message, and that they are in the group with the message. Because of this if a user is not in a group, they would access to no messages until they were added to a group. We also assert that a user is the owner of a message, and this assertion will be used with setting message permissions and removing the message. Messages are stored using symmetric AES 256-bit encryption. And only decrypted from a database call using all the previous mentioned verifications to check that the user is in the group and has access within the group.

**Symmetric Encryption:**

We have used 256-bit AES encryption to encrypt all messages. The encryption key is different for every group and is not shared to the web application at any point.

**Passwords:**

User passwords are never stored in plane text. When registering or changing a password the user password is sent to the database and hashed using 128-bit padding, their username and password. This hash, which cannot be reversed, is used to compare every-time the user logs in to the website.

Login Security Expanded

**Password Security:**

Password security is another serious aspect of our development. The user’s password has a required length of 8 characters. No other requirements are currently in place, so users have complete control of their password’s security and convenience. As covered before we also never store passwords in a hash.

**Login Attempts:**

In order to prevent a brute force, attack we have set a maximum login attempt amount. After reaching his amount the user’s login attempts will be temporarily suspended.

**Password Recovery:**

If the user has forgotten their password, we allow a 2-factor password reset using the user’s username, and a code sent to their email. This code is reset after 24-hours or immediately upon their next login.

**Logout:**

A function to log the user out and set whether they are logged in to ‘N’. This also sets a user’s token to NULL. This way if someone uses the same computer that user was logged into, they would not be able to view pages using cookies from the previous user.

Database Security Expanded

**Database Security:**

Database Password is changed every 30 days.

**Assertion Functions:**

We use assertion functions when sending data and receiving data from the database. We have procedures on account creatin to verify that a username and email are unique. Other procedures verify that a user is allowed to access certain data.

## **DBMS\_CRYPTO:**

**Hashing:**

Our hashing function uses DBMS\_CRYPTO. Hash() which takes in the raw data, and the hashing type and returns the hash we compare all user passwords to that is stored in database.

**Encrypt:**

Our encrypt function uses DBMS\_CRYPTO.Encrypt() which takes in the raw data converted from the message (which has a Unicode standard declared with it), the type of encryption we are using AES 256-bit, and the encryption key associated with the group that contains the message.

**Decrypt:**

Our decrypt using DBMS\_CRYPTO.Decrypt() takes in the same parameters minus the Unicode which is applied after the raw data is decrypted, and we would be taking in the raw encrypted text.

C# and ASP.NET Security

**Front End Security:**

Use of Data Bind Variables to Take Only What We are Looking for When Sending Code to the Database.

PL/SQL Anonymous Block

Obfuscated Password Text Box