Security and Data Analysis
CookChain

Ashlee Boyer
Emma Rogge
Lujia Zhang

01/19/2016

Table of Contents

1 Privacy Analysis

The CookChain database will contain some information that requires sensitive treatment due to its personal nature. Other data within the database is not personal and is of a public nature.

Information considered private, and therefore not accessible to other users, includes the user's name, the user's password, and the user's credit card information. Additionally, the user may choose whether their queue of recipes and ingredients is private (viewing is restricted only to them), or public by invitation (a special link can be emailed by the user allowing others to view either their recipes or their ingredient list). Allergy information is also private, but can be shared in the same manner by the user.

Information that is not private includes recipe information, which is publicly available to all users of CookChain. Since each recipe is available to any users, the recipes are not protected by encryption or other restriction, except that they cannot be accessed without a user account.

2 Security Analysis

The main security concern is any potential leaking of personal information because each CookChain user account contains primarily personal data, such as the user's daily grocery list, ingredients list, eating preferences and food allergies. All this information is highly sensitive since a CookChain account is so closely associated with the user's personal life. The most important security feature aims to prevent anyone other than the user from accessing that particular user's account. A second type of security breach would be one that, like the Ashley-Madison scandal of recent, revealed the identities of CookChain users to the public and the media. While we at CookChain are not ashamed to admit that we need help in the kitchen sometimes, we imagine such a revelation could bring down the empires of chefs like Bobby Flay or Emeril Lagassi. The third type of security breach, and most concerning, would be the phishing of credit card data based upon leaked password and username information. To prevent all three cases of security compromise, CookChain will implement a multiple-step security plan.

Each user's account should be highly secured by two levels of security to prevent hacking and account leaking. First, both the user's username and password will be encrypted and linked to each other with a complicated hashing relation. Additionally, each account has an account ID. When users add or change data related to their account, the CookChain database will first confirm the account ID then add or change data corresponding to the related account. For privacy reasons, no individual will have access to any specific account's data or accountID. Even CookChain's developer team and system administrator can only look at the overall database table with encrypted usernames and other sensitive personal information.

3 Entity Integrity Analysis

a. User Table: The UserID is the primary key for this table and an nvarchar with the maximum length of 15. Name is a 30 character nvarchar. The password is a 10 character nvarchar. No attributes of this table can be null.

b. Credit Card Table: The Number attribute is the unique primary key of this table which is an integer value. The UserID is a foreign key referenced from the User table. The Code attribute is an integer value for the security code located on the credit card. The Type attribute is an nvarchar which holds values such as “MasterCard”, “Visa”, or “American Express.” ExpDate is an nvarchar type which represents the month and year of when the credit card will expire. No attributes in this table can be null.

c. Recipe Table: RecipeID is a unique integer value used to identify every Recipe. The Author attribute is nvarchar with a maximum length of 30 characters. The Time attribute is of type time which stores when the recipe was created by the author. The FilePath is an nvarchar used to store the location of the recipe. Only the Author and Time attributes may be null in this table.

d. Ingredient Table: IngredientID is a unique integer value used to identify different Ingredients. The Name is an nvarchar of maximum length 20 characters. Provenance is also a 20 character nvarchar to store the purchase location and date of the ingredient (for instance, "Kroger, 1/21"). Quantity is an integer value which keeps track of how many of the ingredient a user has in their storage. Only the Provenance attribute may be null in this table.

e. IsAllergicTo, Stores, and Uses Tables: UserID and IngredientID are both foreign keys referenced from the User and Ingredient tables, respectively. IsAllergicTo describes the relationship where a User is allergic to an Ingredient. Stores describes when a User has add an Ingredient to their storage. Uses describes when a User has used or removed an Ingredient from their storage. No attributes of these tables may be null in any instance.

f. Saves Table: UserID and RecipeID are foreign keys referenced from the User and Recipe tables, respectively. The Saves relationship describes when a User saves a Recipe to their personal collection. No attributes of these tables may be null in any instance.

g. Contains Table: RecipeID and IngredientID are foreign keys referenced from the Recipe and Ingredient tables, respectively. The Contains relationship describes when an Ingredient is contained by some Recipe. No attributes of these tables may be null in any instance.

4 Referential Integrity Analysis

Upon delete, all operations will cascade. For example, if a User is removed from the database, all relationships having to do with a UserID must also be deleted. These relationships include the UserIngredients, UserRecipe and UserCreditCard Tables. If the relationships are not deleted, there will be null values for UserID which is not allowed.

Updates to tables will cascade and also reject, but only if the desired operation creates null values where they are not allowed. For example, if a UserID was updated in the User table, then all the instanaces of UserID in other tables would be null, which is never allowed in those tables.

5 Business Rule Integrity Analysis

When a User consumes or disposes of any Ingredients, they will need to specify so using their profile. This will provide the most accurate results when searching for Recipes in the database. When a User searches for recipes, the recipes returned will specify exactly that ingredient, so adding "pasta" to one's ingredient list may not reveal recipes that call for "spaghetti" or "rigatone", even if substitutions could be made. We leave, therefore, artistic license to the user in making such decisions and encourage adding an ingredient in a few different ways if you are not finding suggested recipes satisfactory. Because quantity is not taken into account on the recipe side, as the ingredient quantity is only listed in the UserIngredients table, there is no need to reconcile quantity across tables. Additionally, as no other values need be reconciled across tables, this is the extend of our business logic until future versions of the application are rolled out.