Applications that you build on top of APEX are not, by default, magically hacker-proof. Even tight platforms such as APEX have several security concerns. In APEX, these concerns usually center around three main areas: authentication, authorization, and vulnerability exploits.

Authentication refers to the process of checking if the user has rights to access (log in to) the application. This is usually done through a username-password challenge. Authorization is the process of specifying access rights for each user to a particular resource in the application. For instance, an authorization scheme may permit a user to view a report but not to delete it. Finally, security vulnerability exploits—events like SQL injection attacks and cross-site scripting attacks—work on the premise of cleverly manipulating input data so that it ends up being executed by your application.

The good news is that APEX provides ample features and allocation to elegantly handle all three concerns. In this chapter, you will learn how to beef up security in your application.

8-1. Creating Your Own Authentication Scheme

Problem

You have an existing database table containing the list of all users in the organization, together with their passwords. This database table is a custom table proprietary to your organization. You try to convince your bosses to migrate the list of user accounts from the custom table into APEX, but they insist that your application authenticate against this table instead in real time.

And so you embark on this task. You want to create a custom authentication scheme to authenticate your APEX application against this external database table.

Solution

Your first task is to create the database objects used in this recipe. To create the CustomLogins table (and sample records), run the following SQL:

```sql
CREATE TABLE "CUSTOMLOGINS"
(   "USERID" VARCHAR2(50),
    "USERNAME" VARCHAR2(255),
    "PASSWORD" VARCHAR2(255),
    CONSTRAINT "CUSTOMLOGINS_PK" PRIMARY KEY ("USERID") ENABLE
)```
Your next task is to define the actual authentication function itself. You will create a very simple authentication function that simply checks if the specified username and password exists in the table. If they do, access is granted. To create this function, open the SQL workshop, and run the PL/SQL shown in Listing 8-1.

**Listing 8-1. Defining the Authentication Function**

```sql
CREATE OR REPLACE FUNCTION MyCustomAuthenticator (
    p_username IN VARCHAR2,
    p_password IN VARCHAR2
) RETURN BOOLEAN IS
    l_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO l_count FROM CUSTOMLOGINS WHERE Username=p_username AND Password=p_password;
    IF l_count > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
```

Your next task will be to define a new authentication scheme. To do so, follow these steps:

1. Open an existing application and click the Shared Components icon.

2. Under the Security section, click on the Authentication Schemes link, as highlighted in Figure 8-1.