Project Tunestore I

Ethan Kalika

ITIS 4221

February 5 2023

Table of contents

1.0 General Information

1.1 Purpose

In this lab you are to perform a penetration test on an online music store application. This application, named Tunestore, has 14 use cases: Login, Logout, Register user, View profile, Change password, Add balance to account, View friends, Add a friend, View CDs, View CD comments, Buy a CD, Download a CD, Give CD as gift to friends. Tunestore is in the class VM.
Phase I:

You are asked to identify the following SQL vulnerabilities:

- Login in as a random user
- Login as a specific user
- Register a new user with lots money in account without paying for it

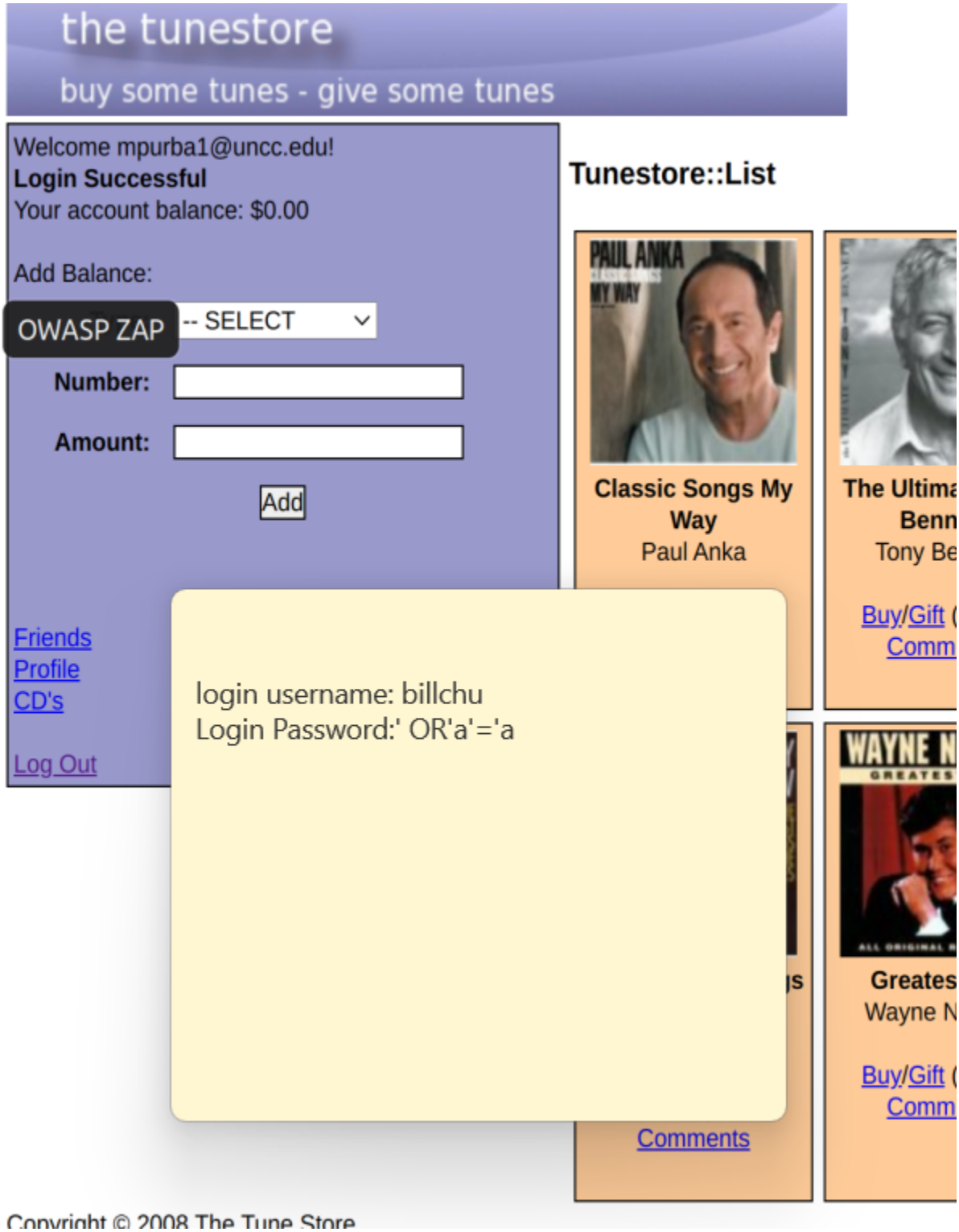
You are also asked for one stored XSS and one reflective XSS vulnerability.

2.0 SQL Injection

2.1 SQL Injection - Logging in as a random user

Tunestore has SQL vulnerability and knowing that it has this kind of vulnerability.

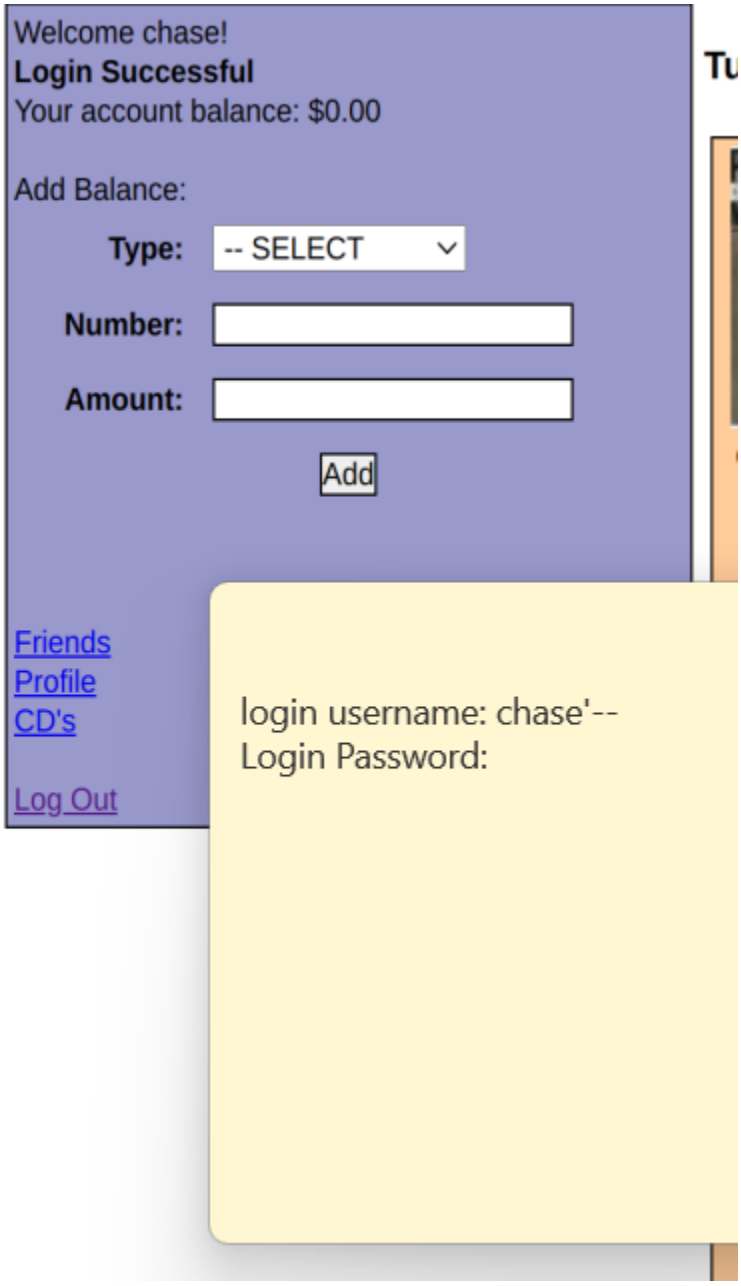
I can login as a random user.



The image above shows how I put billchu into the username and my password was ‘ OR’a’=’a and when I clicked login, it entered into mpurba1@uncc.edu account. This works because the input I put into the password input box is always true, I can login into random accounts, without needing a username.

2.2 Logging in as a specific User

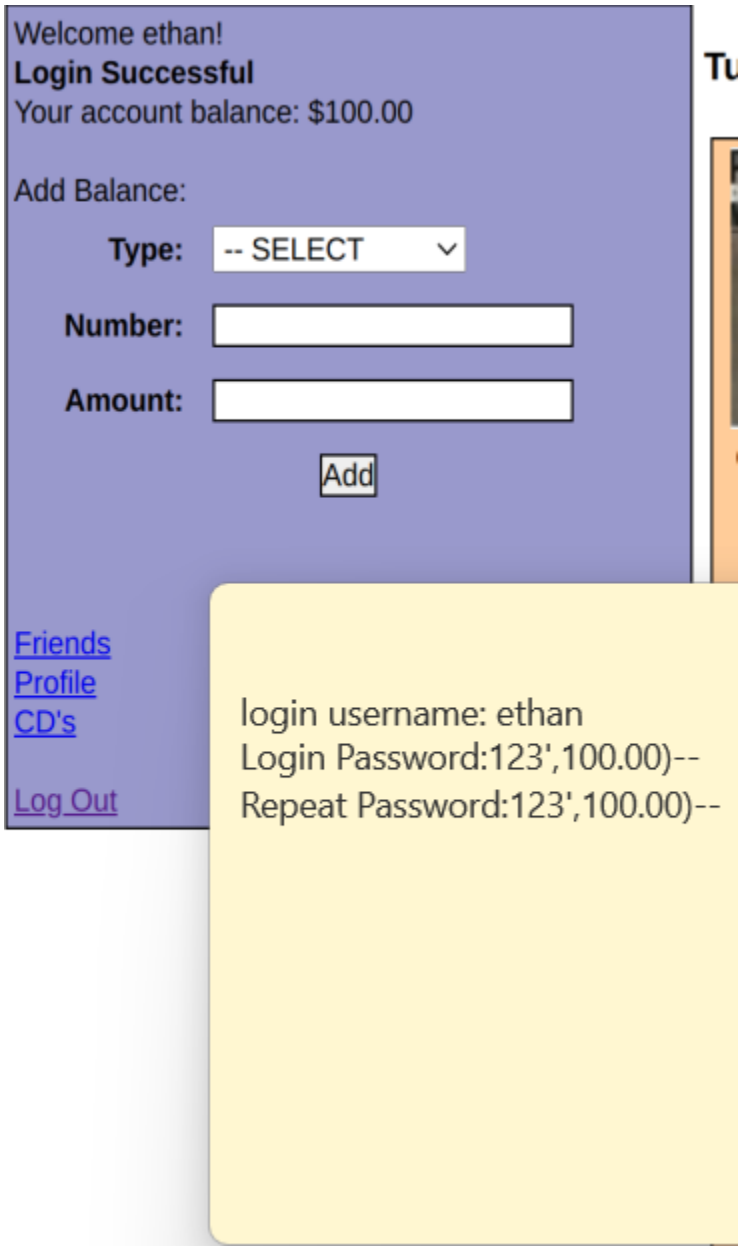
Using the vulnerability that Tunestore has, I can login as a specific user without a password.



The code to implement this vulnerability is user’-- . This sql injection allows you to input any us that is in the system and enters their account. This code works because the -- means that everythi will be removed. So password won't be needed and any error won't occur when logging in.

2.3 Register a new user with lots of money in account without paying for it

Another SQL vulnerability is that I can create a new user and and as much money to my account as I please.

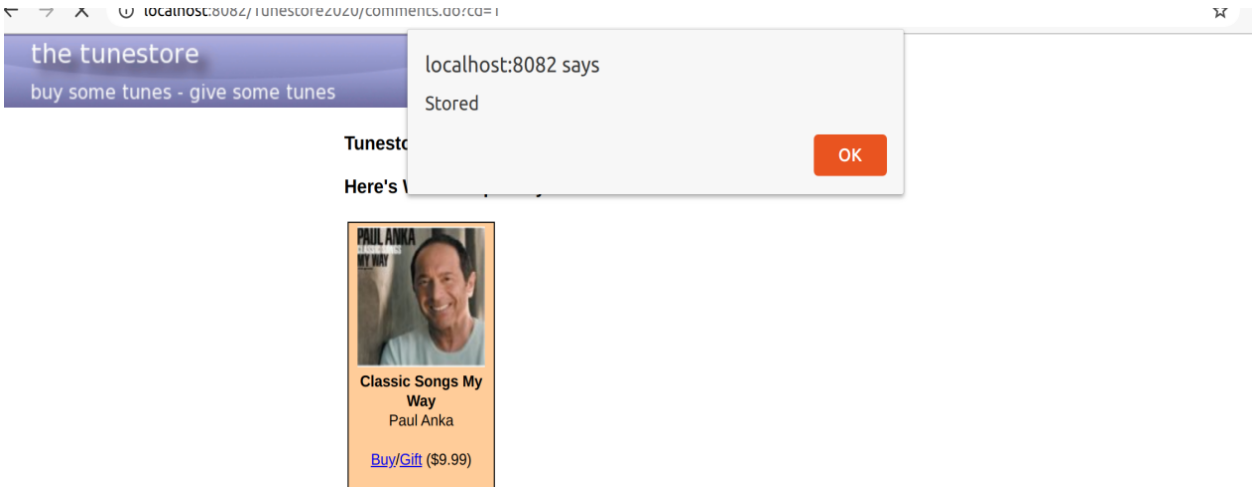


In the image it shows when I went to create an account I imputed ethan as the username and in the login password I imputed 123’,100.00)--. The 123 is the password to login into the account and the 100.00 is the amount inserted into my account without paying for it. The double dashes at the end get rid of the rest of the code when submitted for registration.

3.0 XSS Vulnerability

3.1 Stored XSS

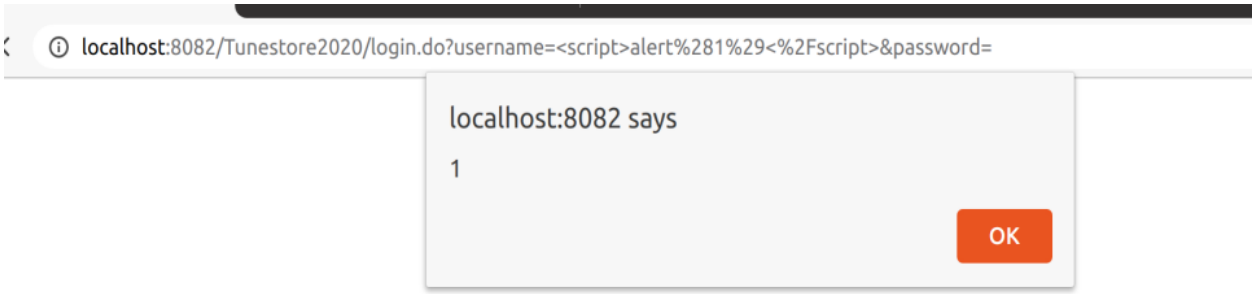
This vulnerability exploits javascript that stores code to steal information/data.

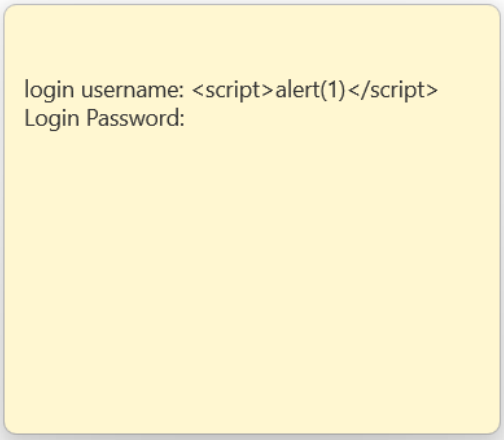


The image shows how I stored an alert script into the comments of the specific user named chase and then when I reload the page it prompts 3 times of alert, showing that there is a stored alert.

3.2 Reflective XSS

Reflective XSS is when you input a script line into an input box and the site just bounces the code back.





The image shows how in the login username I added the line <script>alert(1)</script> and when I clicked submit it bounced back the alert that I put in the box.