

# Project Tunestore I

Ethan Kalika

ITIS 4221

February 5 2023

# Table of contents

<u>Section</u>	<u>Page#</u>
<b>1.0 General Information</b>	<b>3</b>
1.1 Purpose	3
<b>2.0 SQL Injection Vulnerability</b>	<b>4</b>
2.1 Login as a random user	4
2.2 Login as a specific user	5
2.3 Register a new user with lots of money	6
<b>3.0 XSS Vulnerability</b>	<b>7</b>
3.1 Stored XSS	7
3.2 Reflected XSS	8

## **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 screenshot shows the Tunestore website interface. At the top, a purple banner reads "the tunestore" and "buy some tunes - give some tunes". Below this, a purple sidebar on the left contains the following text: "Welcome mpurba1@uncc.edu!", "Login Successful", "Your account balance: \$0.00", "Add Balance:", a dropdown menu with "OWASP ZAP" selected and "-- SELECT" as the option, input fields for "Number:" and "Amount:", an "Add" button, and links for "Friends", "Profile", "CD's", and "Log Out". The main content area is titled "Tunestore::List" and displays a grid of music items. The first item is "Classic Songs My Way" by Paul Anka, featuring a photo of Paul Anka. The second item is "The Ultimate Bennett" by Tony Bennett, featuring a photo of Tony Bennett. The third item is "Greatest Hits" by Wayne Newton, featuring a photo of Wayne Newton. Each item has a "Buy/Gift" link and a "Comments" link. A yellow text box is overlaid on the bottom left of the page, containing the text: "login username: billchu" and "Login Password: ' OR'a'='a'".

the tunestore  
buy some tunes - give some tunes

Welcome mpurba1@uncc.edu!  
**Login Successful**  
Your account balance: \$0.00

Add Balance:

OWASP ZAP -- SELECT

Number:

Amount:

Add

[Friends](#)  
[Profile](#)  
[CD's](#)  
[Log Out](#)

**Tunestore::List**

**Classic Songs My Way**  
Paul Anka

**The Ultimate Bennett**  
Tony Bennett

[Buy/Gift](#) [Comments](#)

**Greatest Hits**  
Wayne Newton

[Buy/Gift](#) [Comments](#)

login username: billchu  
Login Password: ' OR'a'='a'

[Comments](#)

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 screenshot shows a web application interface with a purple background. At the top, it says "Welcome chase!" and "Login Successful". Below that, it displays "Your account balance: \$0.00". There is a section for "Add Balance:" with a "Type:" dropdown menu set to "-- SELECT", a "Number:" input field, and an "Amount:" input field. An "Add" button is located below these fields. On the left side, there are links for "Friends", "Profile", "CD's", and "Log Out". A yellow sticky note is overlaid on the bottom right of the interface, containing the text "login username: chase'--" and "Login Password:". The browser's address bar shows "Tu".

Welcome chase!  
**Login Successful**  
Your account balance: \$0.00

Add Balance:

Type: -- SELECT

Number:

Amount:

Add

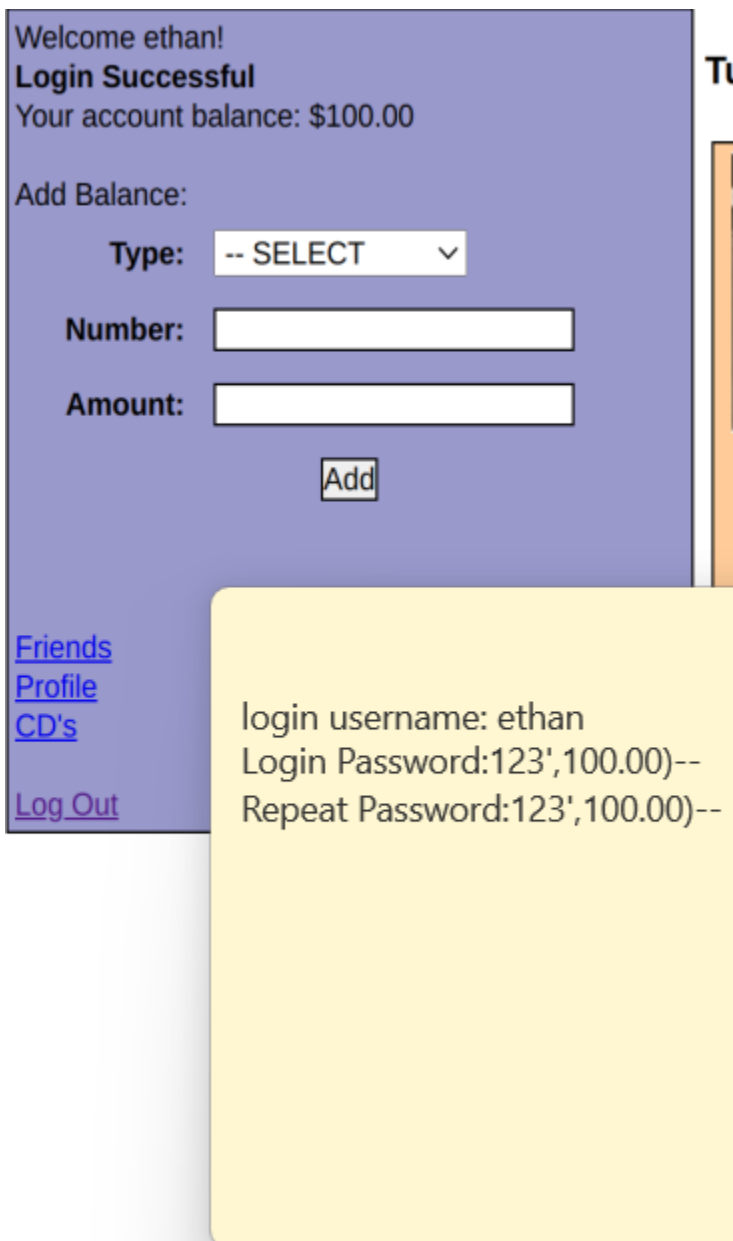
[Friends](#)  
[Profile](#)  
[CD's](#)  
[Log Out](#)

login username: chase'--  
Login Password:

The code to implement this vulnerability is user'-- . This sql injection allows you to input any user that is in the system and enters their account. This code works because the -- means that everything after it 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 add as much money to my account as I please.



Welcome ethan!  
**Login Successful**  
Your account balance: \$100.00

Add Balance:

Type: -- SELECT ▾

Number:

Amount:

[Friends](#)  
[Profile](#)  
[CD's](#)  
[Log Out](#)

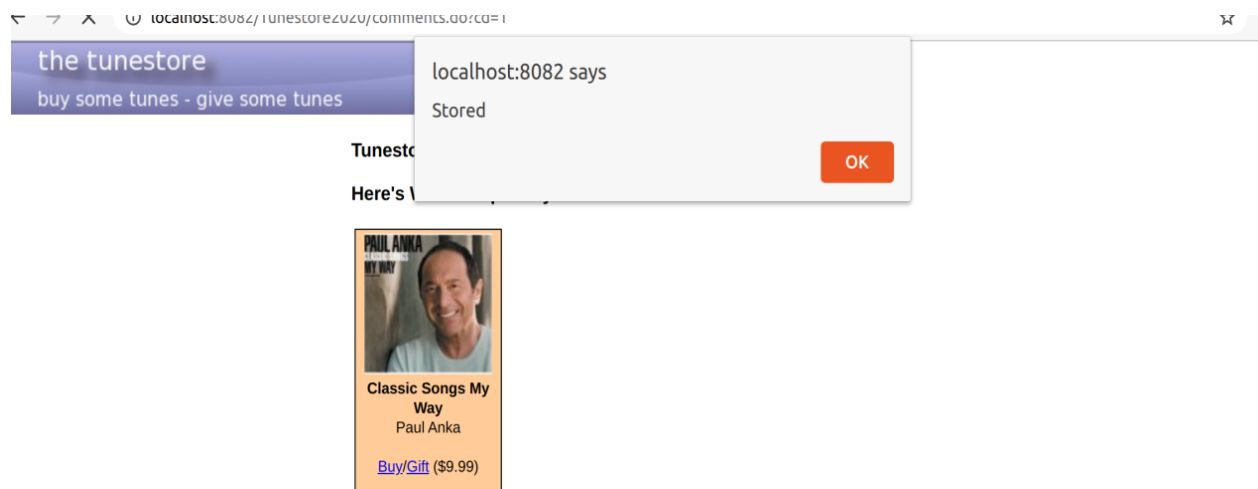
login username: ethan  
Login Password:123',100.00)--  
Repeat Password:123',100.00)--

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

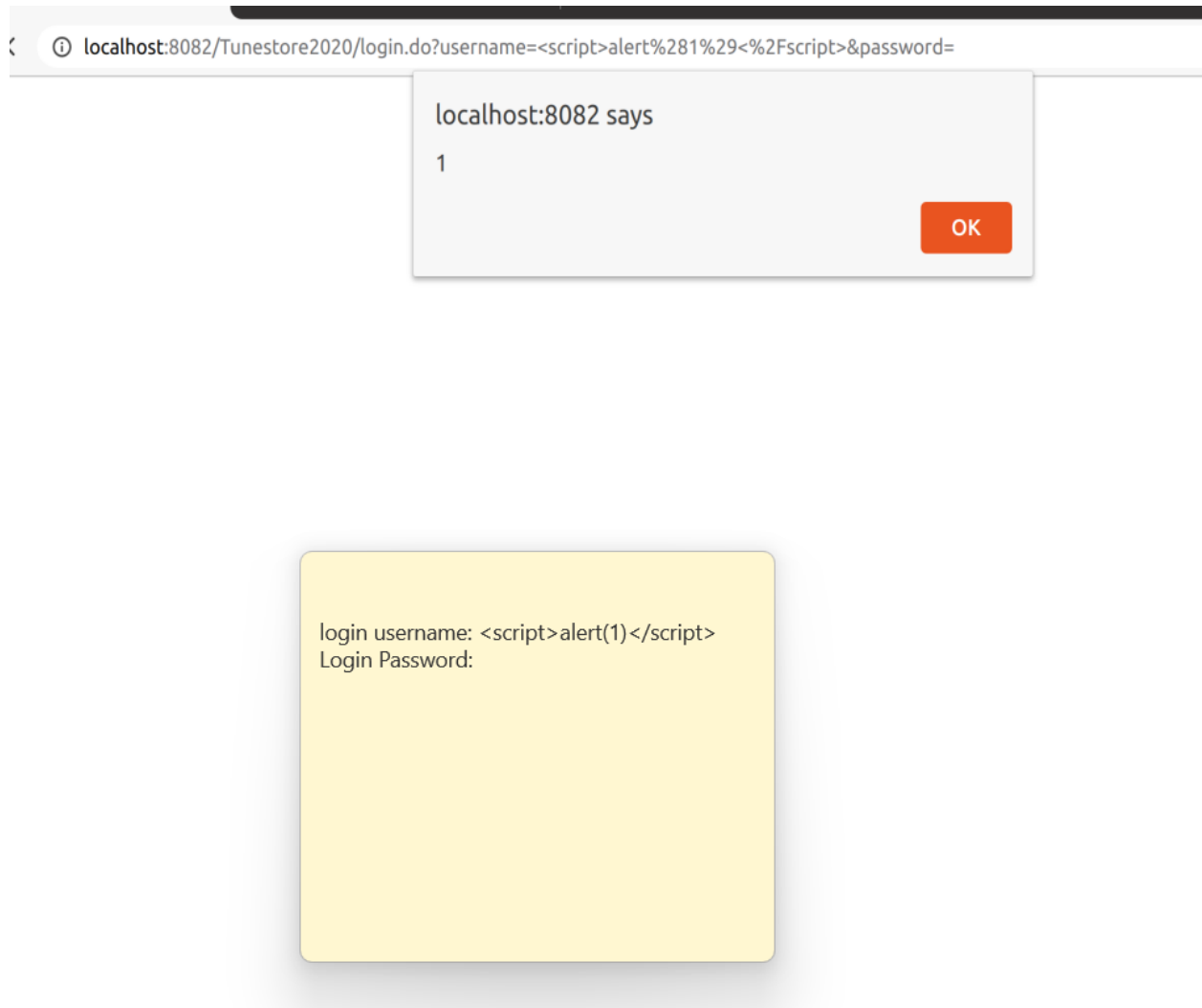
This vulnerability exploits javascript that stores code to steal information/data.



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.