Vulnerability Assessment and Systems Assurance Report

TuneStore 1

Tristan Allison

ITIS 4221/5221

February, 2022

VULNERABILITY ASSESSMENT AND SYSTEM ASSURANCE

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1.1 Purpose - The purpose of this project is to analyze the vulnerability of the application. This is to determine the security of the app. The application will be tested by using XSS cross-site scripting, and SQL injection.

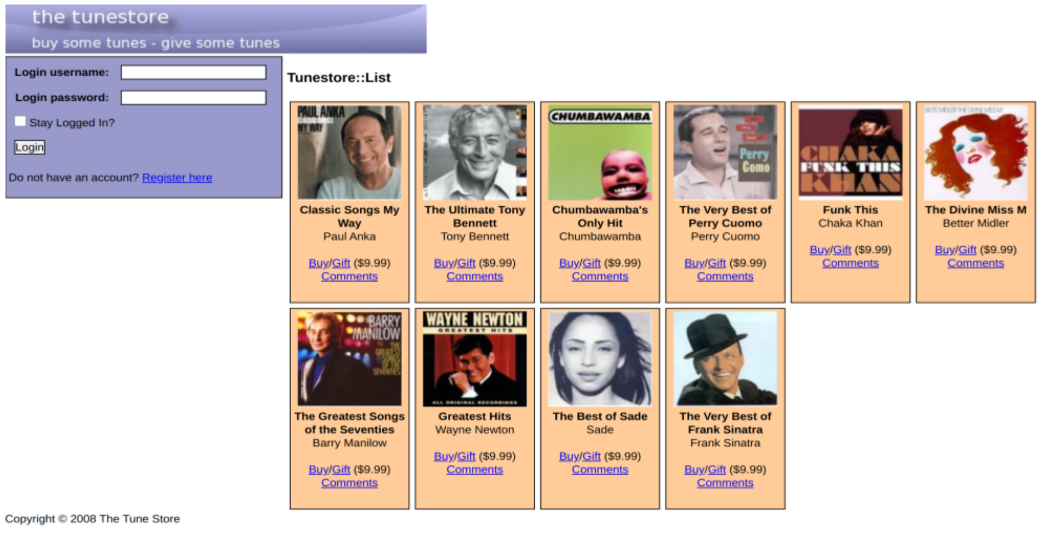
1.2 Points of Contact
Dr.Bill Chu: billchu@uncc.edu

2.0 SQL Injection

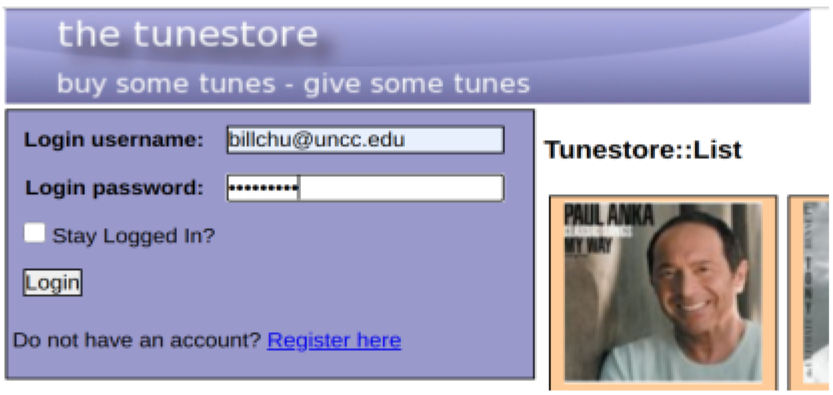
2.0 SQL Injection - A SQL injection attack is where a malicious user changes the hash that they send in order for it to execute SQL instructions to get around a login or change the database.

2.1 Login in as a random user

2.1 Login in as a random user - An example of a SQL injection vulnerability allows the attacker to log in as the first user in the system. The log in function checks if the password is correct so changing the hash to make the password return true logs in the first user. Below is the login page



An attacker can login as a random user by sending the password‘ OR ’1’=’1 to the database because this makes the password true so SQL will automatically log in the first user. The screenshots below will show how to do it.
Use any user and enter ‘ OR ‘1’=’1 into the password.


the tunestore

buy some tunes - give some tunes

Login username: billchu@uncc.edu

Login password: •••••••••

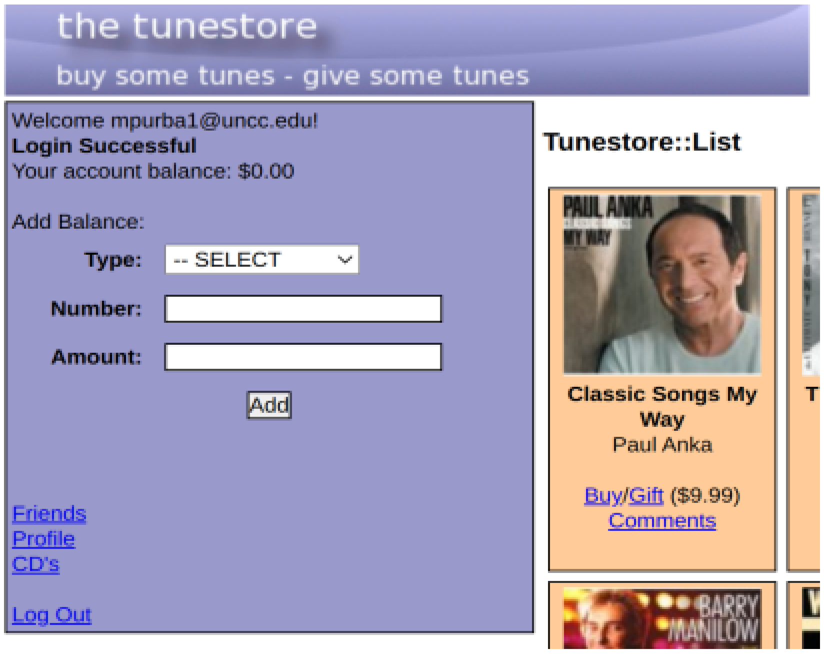
Stay Logged In?

Login

Do not have an account? Register here

Tunestore::List

This logs in the first entry in the database no matter what the user name entered was


the tunestore

buy some tunes - give some tunes

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

Add Balance:

Type: -- SELECT

Number:

Amount:

Add

Friends
Profile
CD's

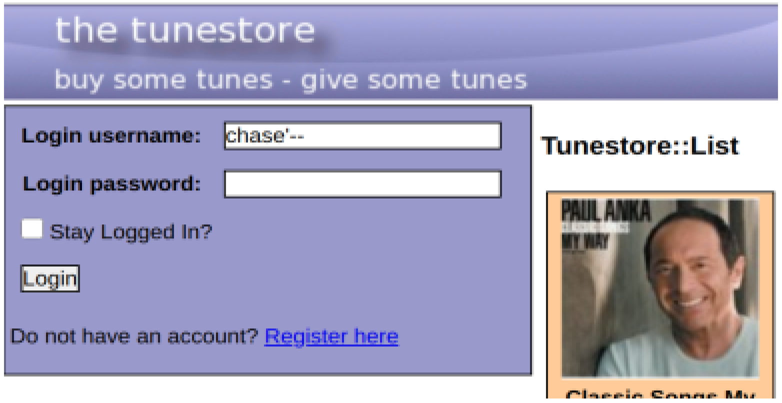
Log Out

Tunestore::List

Classic Songs My Way
Paul Anka

Buy/Gift ($9.99)
Comments

2.2 Login as a specific user- For this I will use Chase as the random user. An attacker can log in by causing the sql database to only check the username and to not check what password is being used. This is done by taking the username and adding '-- after it because that makes the password check get dropped so it only checks to see if the username exists.


the tunestore

buy some tunes - give some tunes

Login username: chase'--

Login password:

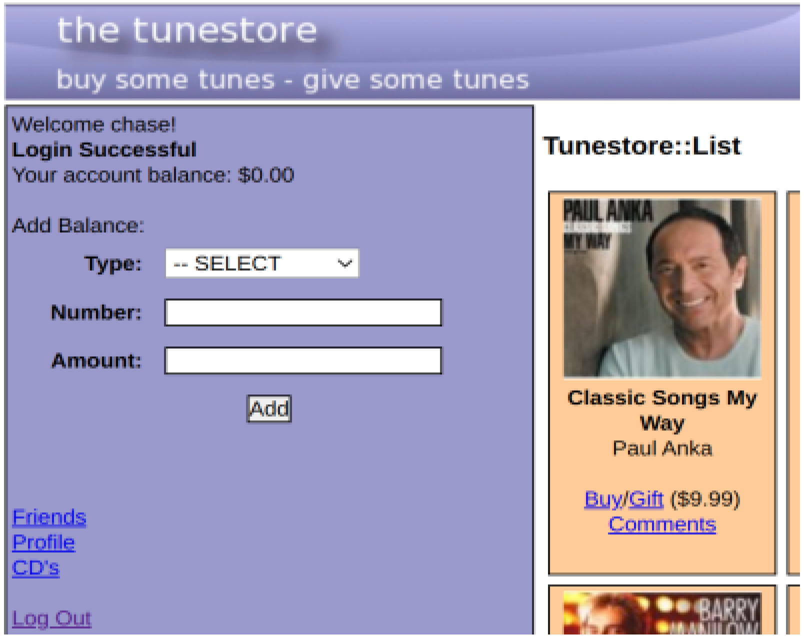
Stay Logged In?

Login

Do not have an account? Register here

Tunestore::List

Classic Songs My


the tunestore

buy some tunes - give some tunes

Welcome chase!
Login Successful
Your account balance: $0.00

Add Balance:

Type: -- SELECT

Number:

Amount:

Add

Friends
Profile
CD's

Log Out

Tunestore::List

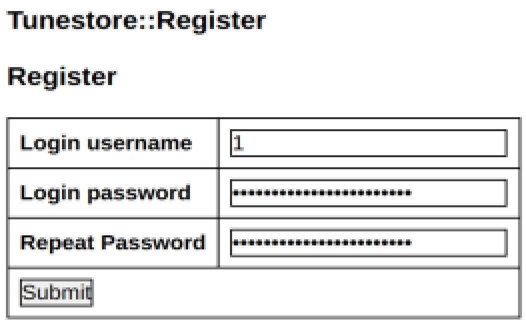
Classic Songs My Way
Paul Anka

Buy/Gift ($9.99)
Comments

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

To do this a SQL injection can be used that updates the user

First start by going to register a new user where the password is put **1", 999999999); --** this will fill out a password and balance and comment the rest out.


Tunestore::Register

Register

Login username	1
Login password	••••••••••••••••••••••••
Repeat Password	••••••••••••••••••••••••
Submit	

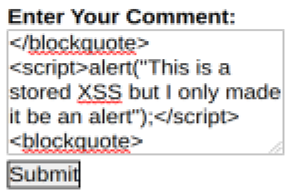
3.0 XSS Vulnerability

An XSS vulnerability is where javascript is inserted into a webpage to make it do something when the user loads the page. Two different types of XSS vulnerabilities are stored and reflected. The difference between them is that stored is stored on the website itself so whenever anyone goes to the website it loads, but Reflected is where the link that the user uses has been modified.

3.1 Stored XSS

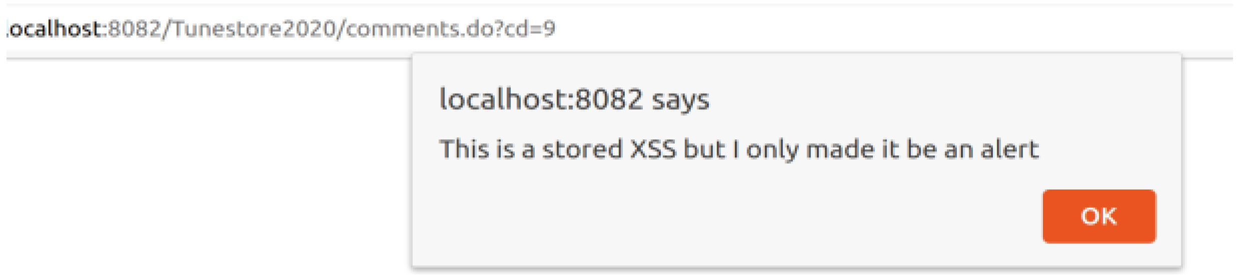
A stored XSS vulnerability that TUNESTORE has is in the comment section: if a person closes the blockquote tag they can insert malicious javascript code to do whatever they want.

This is the comment being made


Enter Your Comment:

```
</blockquote>
<script>alert("This is a
stored XSS but I only made
it be an alert");</script>
<blockquote>
```

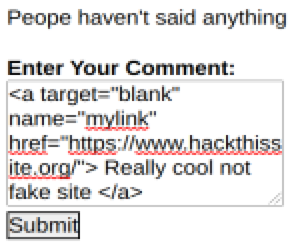
This is what happens when the page is loaded after the comment is made:



This is a stored XSS but I only made it be an alert

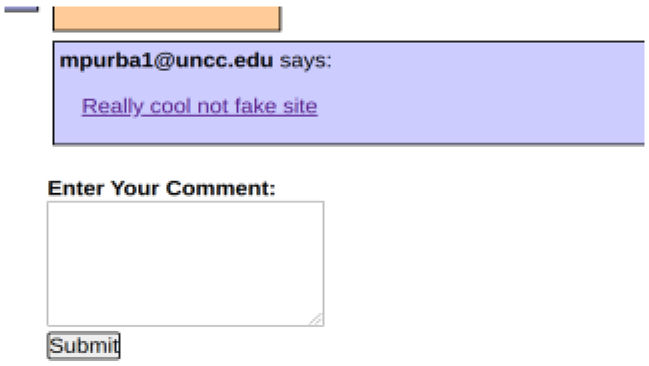
3.2 Reflected XSS

Links can be provided in the comments and the site does nothing to make sure they are not malicious.


Peope haven't said anything

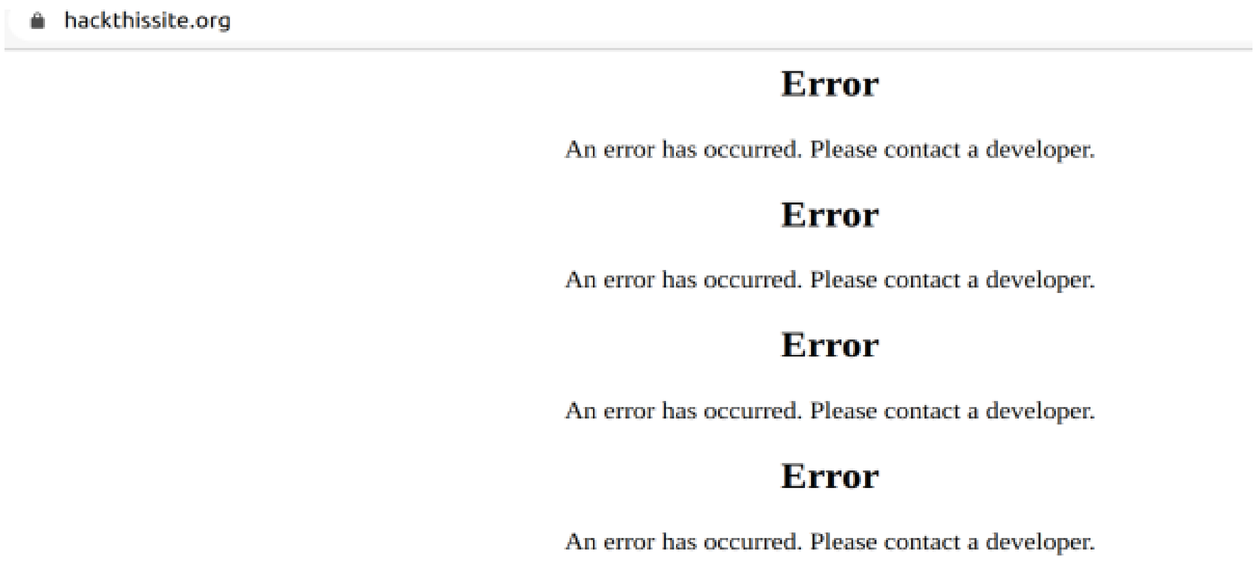
Enter Your Comment:

```
<a target="blank"
name="mylink"
href="https://www.hackthiss
ite.org/"> Really cool not
fake site </a>
```


mpurba1@uncc.edu says:

Really cool not fake site

Enter Your Comment:



Error

An error has occurred. Please contact a developer.

Error

An error has occurred. Please contact a developer.

Error

An error has occurred. Please contact a developer.

Error

An error has occurred. Please contact a developer.