Jacob Kilver (jkilver3)

CS 6035: Introduction to Information Security

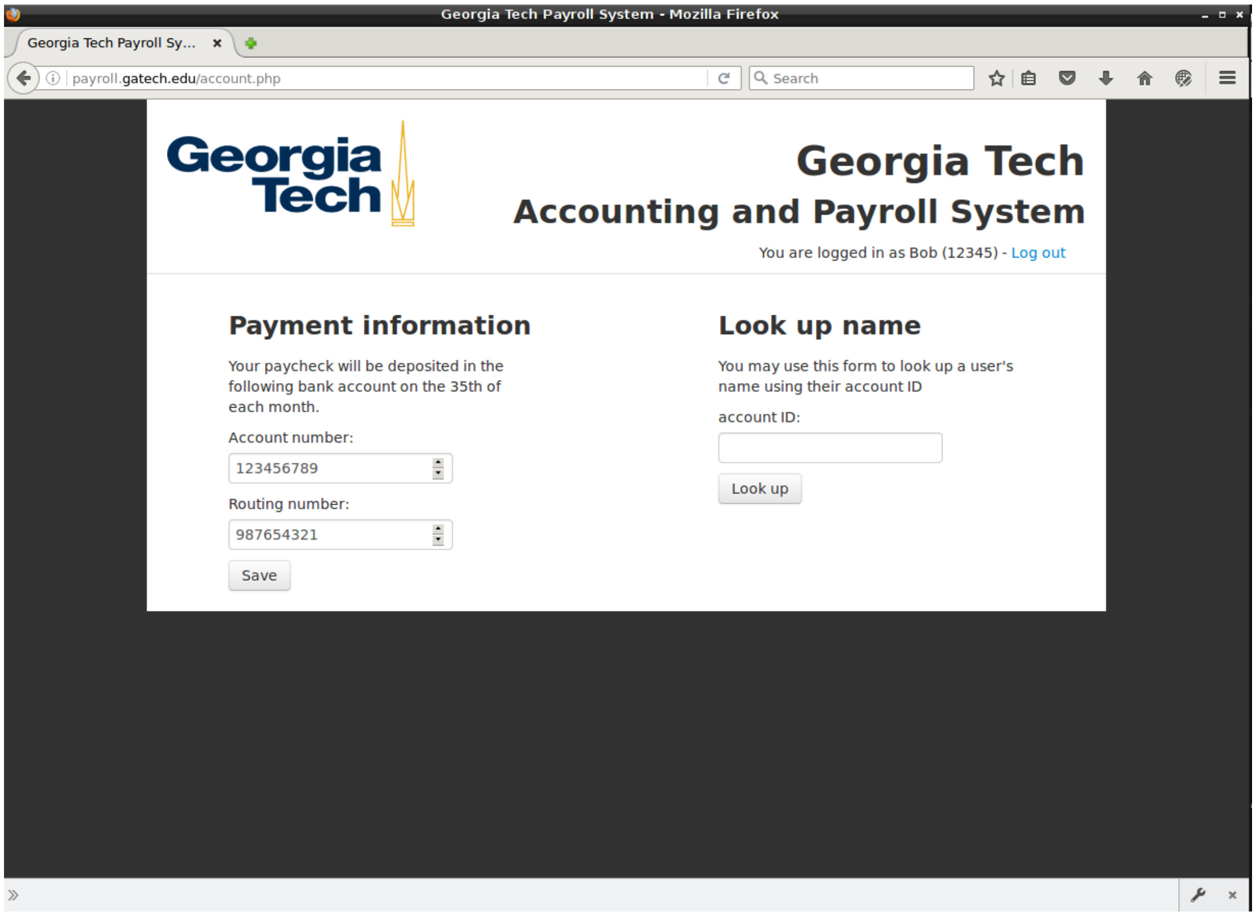
Project 4: Web Security

7 Dec 2016

Target 1: XSRF

The main vulnerability here is the cross-site request forgery prevention mechanism is weak. The first vulnerability is that the test string used to generate the expected response value is based off the account, routing, and challenge fields. However, the challenge field (which is randomly generated) is not required and can simply be left out, meaning that the expected response can be deterministically deduced. Second, when attempting to create this attack and guess the response, the expected value is actually printed out in the message saying the attack was prevented. This is a very poor security choice (though I realize it might just be to make it easier for us).

To prevent this attack, the form must require the challenge (CSRF token) field in order to allow edits to be saved. Alternatively, the standard headers could be checked to ensure that both requests come from the same origin.


Georgia Tech Payroll System - Mozilla Firefox
Georgia Tech Payroll Sy...
payroll.gatech.edu/account.php
Search
Georgia Tech

Georgia Tech
Accounting and Payroll System

You are logged in as Bob (12345) - Log out

Payment information

Your paycheck will be deposited in the following bank account on the 35th of each month.

Account number: 123456789

Routing number: 987654321

Save

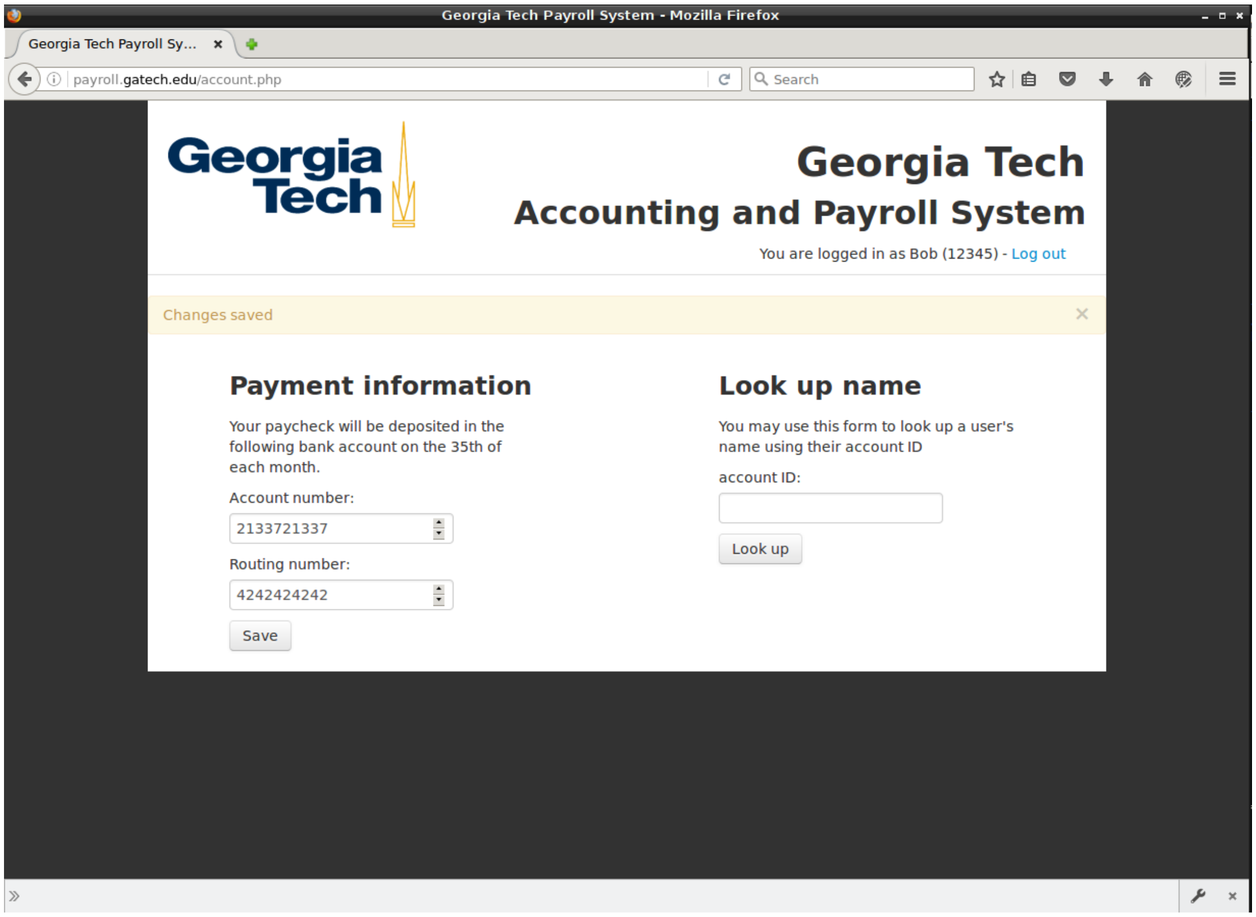
Look up name

You may use this form to look up a user's name using their account ID

account ID:

Look up

Figure 1: Screenshot of account page before attack is executed


Georgia Tech Payroll System - Mozilla Firefox

payroll.gatech.edu/account.php

Georgia Tech
Accounting and Payroll System

You are logged in as Bob (12345) - Log out

Changes saved

Payment information

Your paycheck will be deposited in the following bank account on the 35th of each month.

Account number: 2133721337

Routing number: 4242424242

Look up name

You may use this form to look up a user's name using their account ID

account ID:

Figure 2: Screenshot of account page after attack is executed. This was accomplished by simply navigating to the attack page's HTML on the VM

Target 2: XSS

In this case the weakness is that user input is not scrubbed before it is inserted into the webpage. All user input needs to be considered as being malicious and must be validated (https://www.sitepoint.com/php-security-cross-site-scripting-attacks-xss/). Proper validation of user input could prevent this attack, or simply not echoing back the user input into the webpage. The input should be validated, sanitized, and any escape characters should be treated as literal characters, not their escaped equivalents.

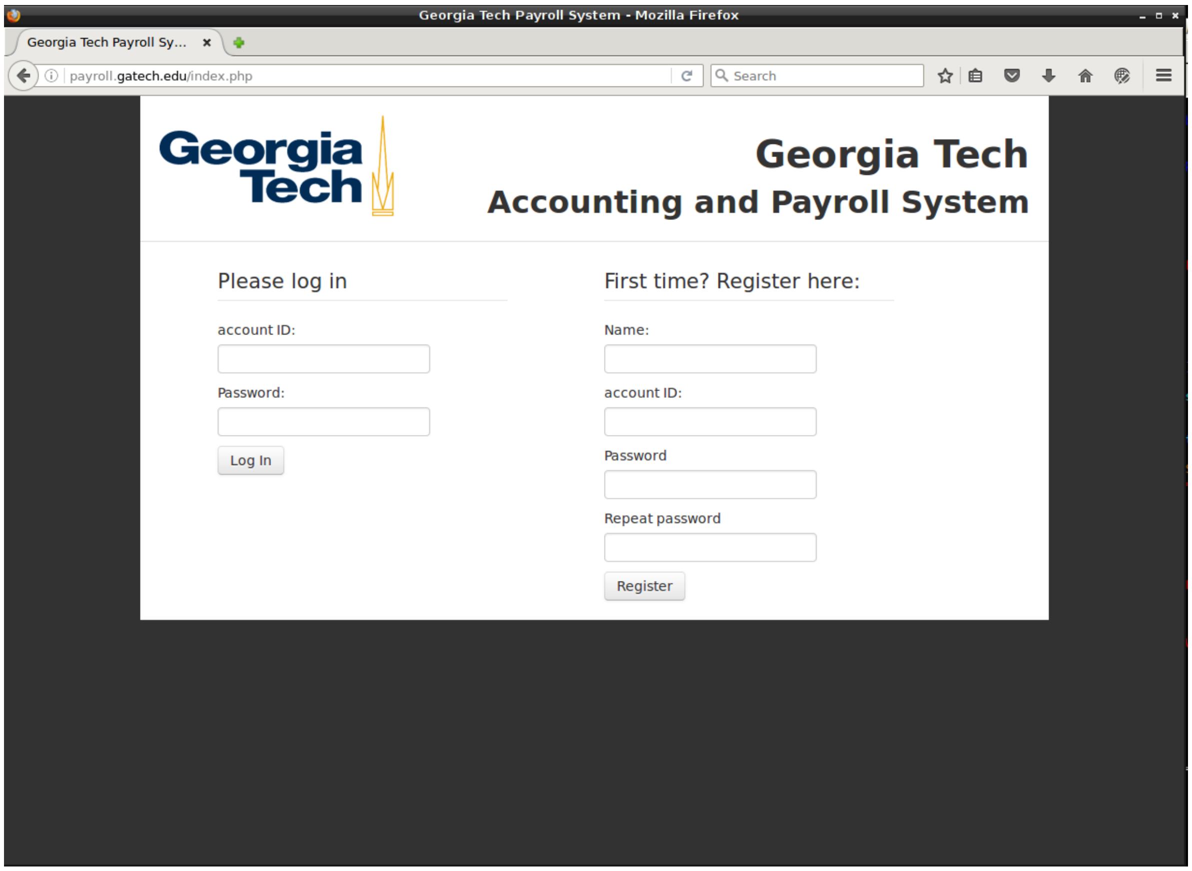
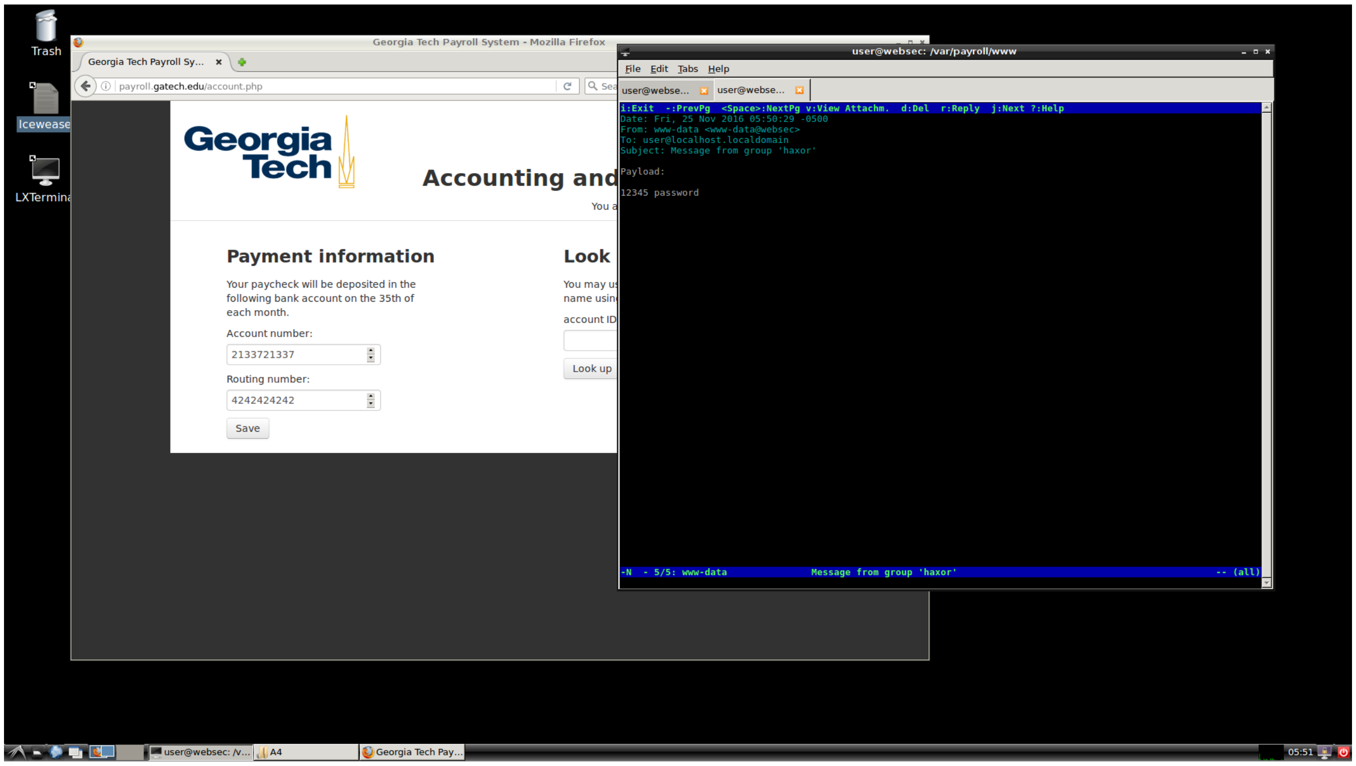

Georgia Tech Payroll System - Mozilla Firefox
Georgia Tech Payroll Sy...
payroll.gatech.edu/index.php
Search
Georgia Tech
Georgia Tech
Accounting and Payroll System
Please log in
account ID:
Password:
Log In
First time? Register here:
Name:
account ID:
Password
Repeat password
Register

Figure 3: Malicious login page. Note that the look is identical and the address is the same as the normal login page



Georgia Tech Payroll System - Mozilla Firefox

payroll.gatech.edu/account.php

Accounting and

You a

Payment information

Your paycheck will be deposited in the following bank account on the 35th of each month.

Account number: 2133721337

Routing number: 4242424242

Look

You may us name usin

account ID

user@websec: /var/payroll/www

```
i:Exit  -:PrevPg  <Space>:NextPg v:View Attachm.  d:Del  r:Reply  j:Next ?:Help
Date: Fri, 25 Nov 2016 05:50:29 -0500
From: www-data <www-data@websec>
To: user@localhost.localdomain
Subject: Message from group 'haxor'

Payload:

12345 password

-N  - 5/5: www-data              Message from group 'haxor'                     -- (all)
```

Figure 4: After successful login, email is sent with username and password information

Target 3: SQL Injection

The vulnerability here is that the SQL query in index.php (or more specifically auth.php) is not properly validated before it is used. Single quotes are not removed from the input string. In fact, it seems that this website does not employ whitelisting as recommended. As such, it is possible to inject SQL commands into the query, allowing the user to login even without a password. Use whitelisting to prevent this attack or better input validation, though whitelisting is preferred. Alternatively, a parameterized query can be used in order to distinguish between code and data.

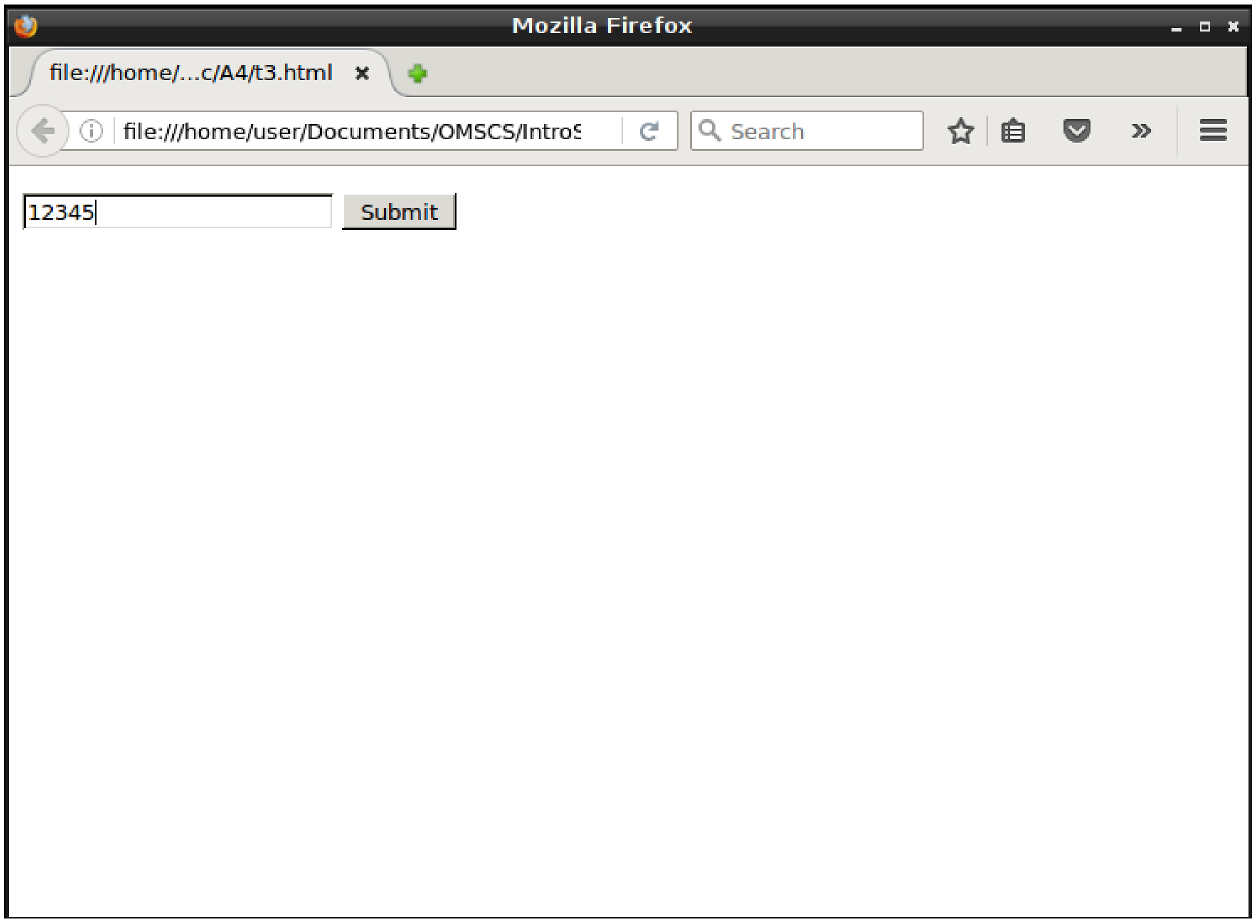
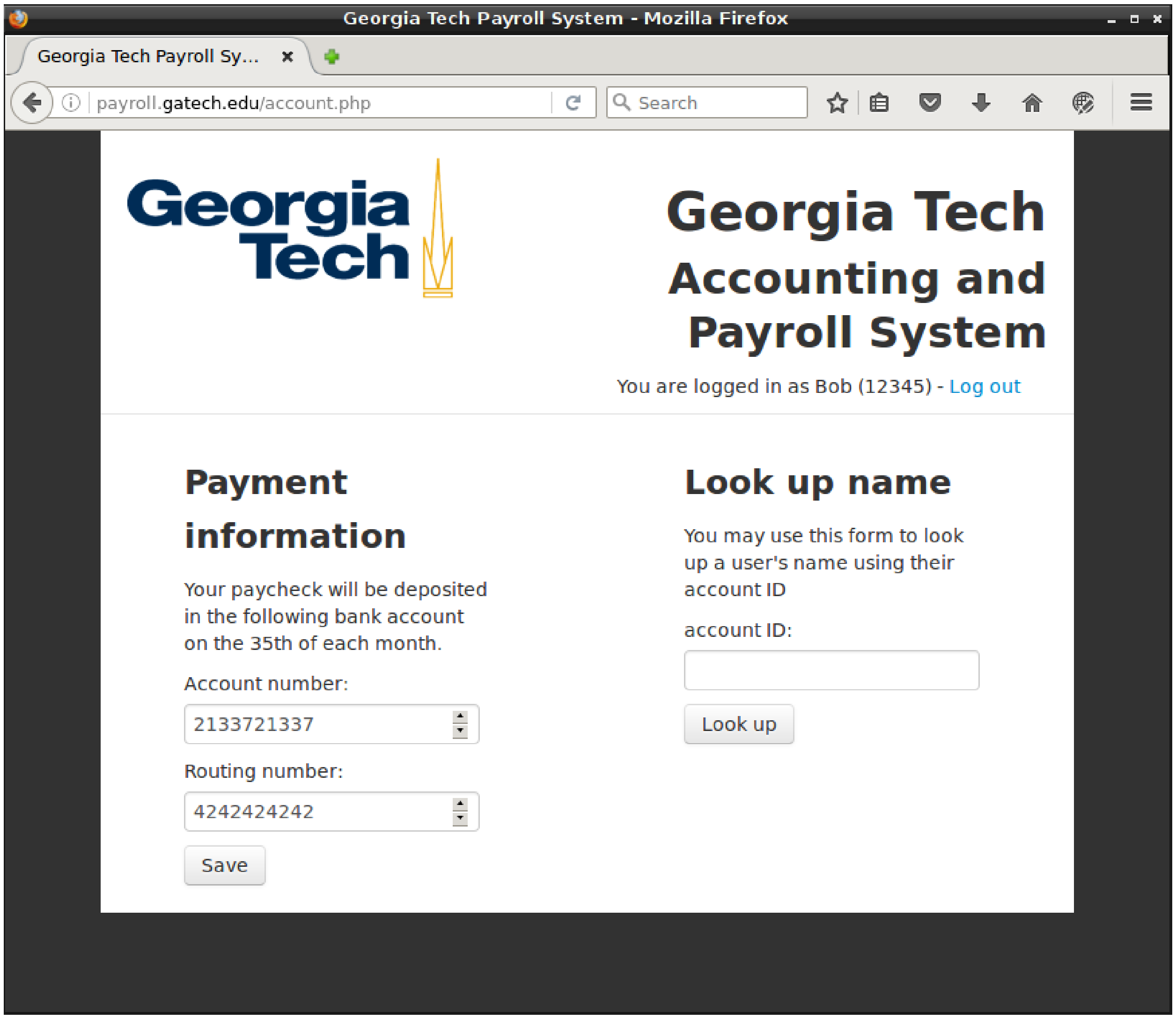

Mozilla Firefox
file:///home/...c/A4/t3.html
file:///home/user/Documents/OMSCS/IntroS
Search
12345
Submit

Figure 5: Attack webpage with existing username in input field


Georgia Tech Payroll System - Mozilla Firefox

payroll.gatech.edu/account.php

Georgia Tech Accounting and Payroll System

You are logged in as Bob (12345) - Log out

Payment information

Your paycheck will be deposited in the following bank account on the 35th of each month.

Account number: 2133721337

Routing number: 4242424242

Look up name

You may use this form to look up a user's name using their account ID

account ID:

Figure 6: After clicking submit, successfully log into account for Bob

In general, OWASP is a great resource to consult on how to prevent these attacks.

- https://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet