CS6035 Project 4: Web Security

Spring 2019

Setting Up

Download the virtual machine for this project via one of the following links:

Download Link:
https://drive.google.com/file/d/1DFq3loAzVxWZV0QHJyvLxj8-DsPn4y4O/view?usp=sharing

You are provided with both root and regular user access to this virtual machine. The credentials are:

Username	**Password**
root	root
user	user

You should only use the **user** account to complete the project. **root** is provided for your convenience in case you need to install extra software or packages.

Interacting with the VM

After logging in with the above credentials type **`startx`** to launch the GUI desktop. VirtualBox guest additions have been pre-installed on the VM. If you wish to install more packages, you may do so by running `apt-get` as root.

Georgia Tech Payroll

The site we will be exploiting in this project is **http://payroll.gatech.edu**, which you can only visit on the VM. Please note that this is a made-up site and does not point to a legitimate site in the real world. For testing purposes, you may register accounts at your will. However, **please DO NOT use your actual passwords and banking account information**.

The source code of the site can be found on the VM in **/var/payroll/www**. There is a bookmark added to the file manager to make your job a bit easier. We will be using **Firefox (Iceweasel)**, which is provided in the VM, to test your exploits. You may also assume JavaScript is always enabled. Do not update your browser version or use something else such as Chrome since we

will grade your scripts using the exact same VM that you have downloaded using Firefox (Iceweasel).

GOOD LUCK AND HAVE FUN!

Deliverables Summary/Requirements

There are 3 targets in total worth 70 points, and the write-up is worth an additional 30 points.

Please submit your deliverables on Canvas as separate files. Do **NOT** zip them. Failure to follow this rule will result in a **5 point penalty on your overall project grade**.

Filename	Description
t1.html	Crafted HTML page for Target 1
t2.html	Crafted HTML page for Target 2
t3.html	Crafted HTML page for Target 3
report.pdf	Please include your full name and your Georgia Tech username (e.g. jdoe3) at the top of the report. This should contain the required responses to the Epilogue section.

Not following the file naming convention above results in a 5 point penalty. Note: Canvas may append additional numbers to you files. This is ok, just be sure to name the original uploaded files as you see above.

Disclaimer

This project is solely for educational purposes. Professor Wenke Lee and the people affiliated with his teaching and research are NOT responsible in the event of any criminal charges brought against any individuals misusing the information in this project to break the law. When in doubt, please consult the TAs or Professor Lee regarding any questions or issues you may have.

Target 1: XSRF (20 points)

You have stumbled upon the Georgia Tech payroll website and discovered a vulnerability. Suppose a user, say Alice, is already logged into the Georgia Tech payroll site. You noticed that you can craft a web page so that when Alice visits your web page, she gets redirected (NO popups) to the Georgia Tech payroll page with her account number and routing number set to some values of your choice.

Poor and living off of ramen noodles, you decide to give it a try and craft a web page to set the banking information to yours.

You forgot your bank account information, but luckily, you remember storing them inside a secret script you wrote a long time ago.

To fetch your bank account number and routing number, run the **get_bank_info** script inside the VM and pass in your Georgia Tech username (e.g. jdoe3). Example command on the terminal:

```
get_bank_info jdoe3
```

Here is an example of what the script will print out:

```
Username: jdoe3
Account number: 962362227
Routing number: 2113956237
```

Double check that you entered your Georgia Tech username. This is the username you use to login to T-square. It is **NOT** your 9 digit student number. If you enter the wrong username, which generates a different account and routing number, your exploit will fail our scripts, and you will receive zero points for this part.

The user must **NOT** see the contents of your crafted page! However, a split second due to browser rendering is acceptable.

Deliverables

- t1.html
- Report.pdf (See Epilogue)

Sample t1.html deliverable

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>XSRF</title>
  </head>
  <body onload="document.forms[0].submit()">
    <form action="" onsubmit="" method="POST">
    <!--
      Your exploits here
      You may also want to change some form attributes
    -->
    </form>
  </body>
</html>
```

Milestones

A successful attack earns 20 points automatically.
If you are unable to complete the task, you will earn partial credit as follows:

Points	**Milestone**
6	Properly identify the vulnerability and explain why it is vulnerable. Provide your response in report.pdf. See epilogue section below.
7	You see the “XSRF prevented” message with your exploit.
7	Able to change the account number and routing number without extra browser tabs or popups. If you get to this point you’ve earned the full 20 points.

Notes

You can visit your web page by entering the path of your file in the browser URL bar. For example, this would be **file:///home/user/t1.html** assuming that your exploit lives in **/home/user/**. You can also simply double click to open the file in Firefox. This opens your exploit in another tab but this is OK and it works. Your actual exploit code must NOT open a new tab via JavaScript or other means.

Do NOT use relative paths for site URLs in your exploits.

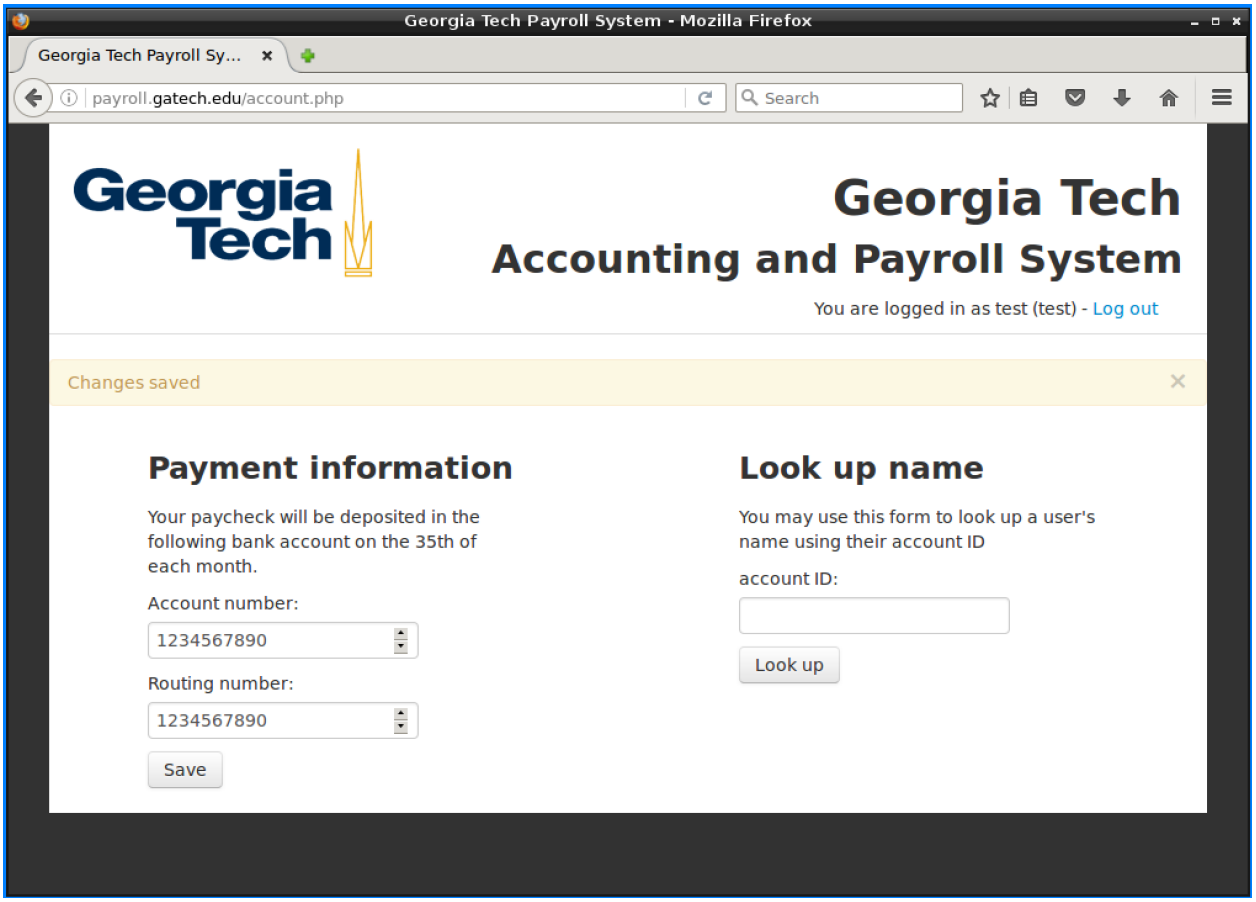
- WRONG -> /somefolder/somefile.php
- CORRECT -> http://payroll.gatech.edu/somefile.php

We see this every semester from a hand full of students. Your exploit will fail and you will not receive full credit.

Example of Successful Exploit

Our autograder is a Selenium script so it will simulate button clicks using the same exact browser and VM that you have. It will do the following for Task 1:

1. Log into the site using a known good username and password.
2. Launch your t1.html file in the same open tab
3. Verify that the Changes Saved is on the page and that the account number and routing number matches your assigned values. Do not use 1234567890 as this is just an example. See the screenshot below.



Target 2: XSS Username and Password Theft (30 points)

You got caught! The good news is that Georgia Tech InfoSec is curious if you can find another vulnerability that is more severe. They will let you off the hook if you help them out. You noticed that you are able to steal a user's username and password. You can craft a web page such that whenever a victim, say Bob, visits the page, it will redirect him (NO popups) to **http://payroll.gatech.edu/**

The web page should look as if Bob visited the site directly. When Bob enters his login information into the page and clicks Log In, an email with his username and password will be sent. Georgia Tech administrators would like you to demonstrate the attack and pay you accordingly. You will have to send the email to the local **user** account on the virtual machine as a proof of concept.

This attack requires an email to be sent to **user** on the system. The good news is that you can use **hackmail**:

```
http://hackmail.org/sendmail.php
```

Open the above URL from within virtual machine for instructions on how to send emails via your attack script. Any mail that the **user** account receives will appear in **/var/mail/user.** A bookmark has been added to the file manager for your convenience.

Requirements

- The attack must be performed using XSS. Providing a phishing web page will result in 0 points. The browser URL bar should contain the domain **payroll.gatech.edu** and not a phishing URL.
- The email payload should be the user's username (login) and password separated by a single space. i.e. **username password** <- notice the space!
    - The sender of the email should be set to **U3ByaW5nMjAxOVRhcmdldDJFYXN0ZXJFZ2c**
    - Failure to follow this format will result in 0 points for this part.
- The redirected page must be **cosmetically identical** to the original page. The web page source can be different as long as the user cannot tell without looking at the source. This may take some trial and error. This part can be difficult! Use the developer tools to help you.
- The page must be functionally identical. This means the user can log into the site in the normal fashion and will not notice any visual/functional differences. Yes, you must be able to log in to pass this test.

Deliverables

- t2.html
- report.pdf (see Epilogue)

Sample t2.html deliverable

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>XSS</title>
  </head>
  <body onload="document.forms[0].submit()">
    <form action="" onsubmit="" method="POST">
    <!--
      Your exploits here
      You may also want to change some form attributes
    -->
    </form>
  </body>
</html>
```

Milestones

A successful attack earns 30 points automatically.
If you are unable to complete the task, you will earn partial credit as follows:

Points	Milestone
6	Properly identify the vulnerability and explain why it is vulnerable. Provide your response in report.pdf. See epilogue section below.
17	Steal the user's username and password and send them to the **user** account via email.
7	The exploited web page is cosmetically identical to the original website. If you get to this point you've earned the full 30 points.

Notes

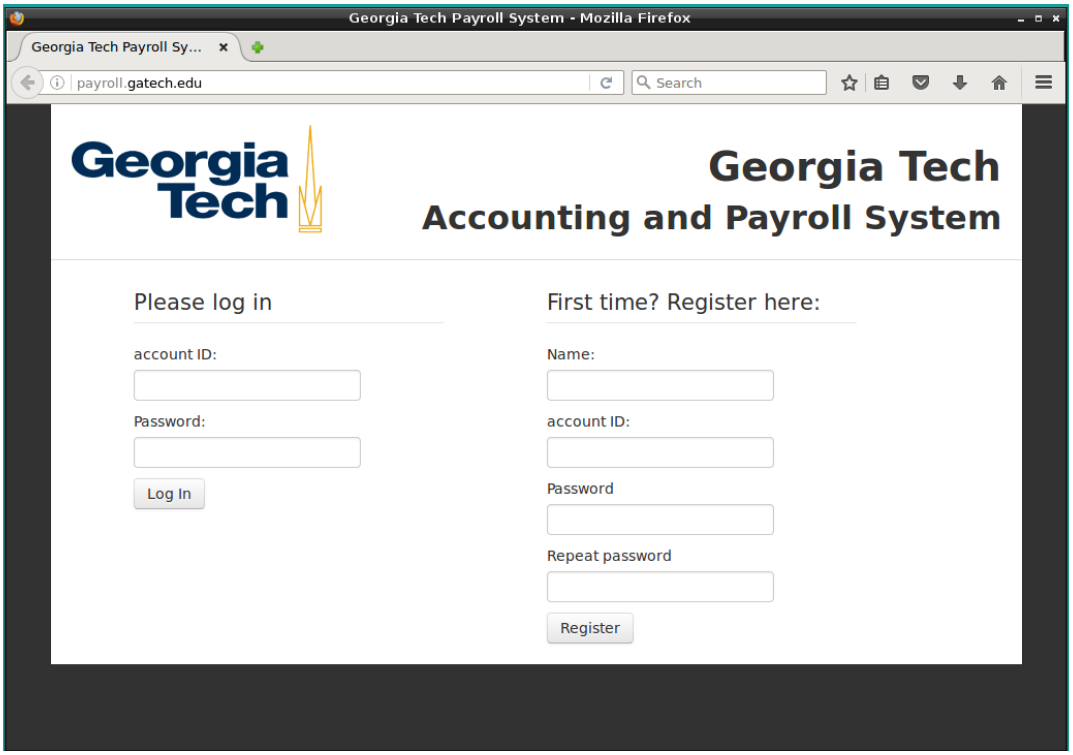
Initially there is not a mail file on the VM. We suggest playing around with hackmail outside of your exploit to make sure you can generate a mail file. You'll simply see a file named "user" show up in the location detailed above. Right click it and open with gedit to view the contents. You can delete the file and hackmail will generate a new one each time you exploit the site. This makes it easier to debug than scrolling a lot in the user file. Delete, exploit to create it and then validate your payload.
Use the developer tools built into Firefox. Simply click F12 in Firefox and you'll see the dev tools pop up at the bottom. This tool is your friend, get to know it and use it to help you through this task.

Example of Successful Exploit

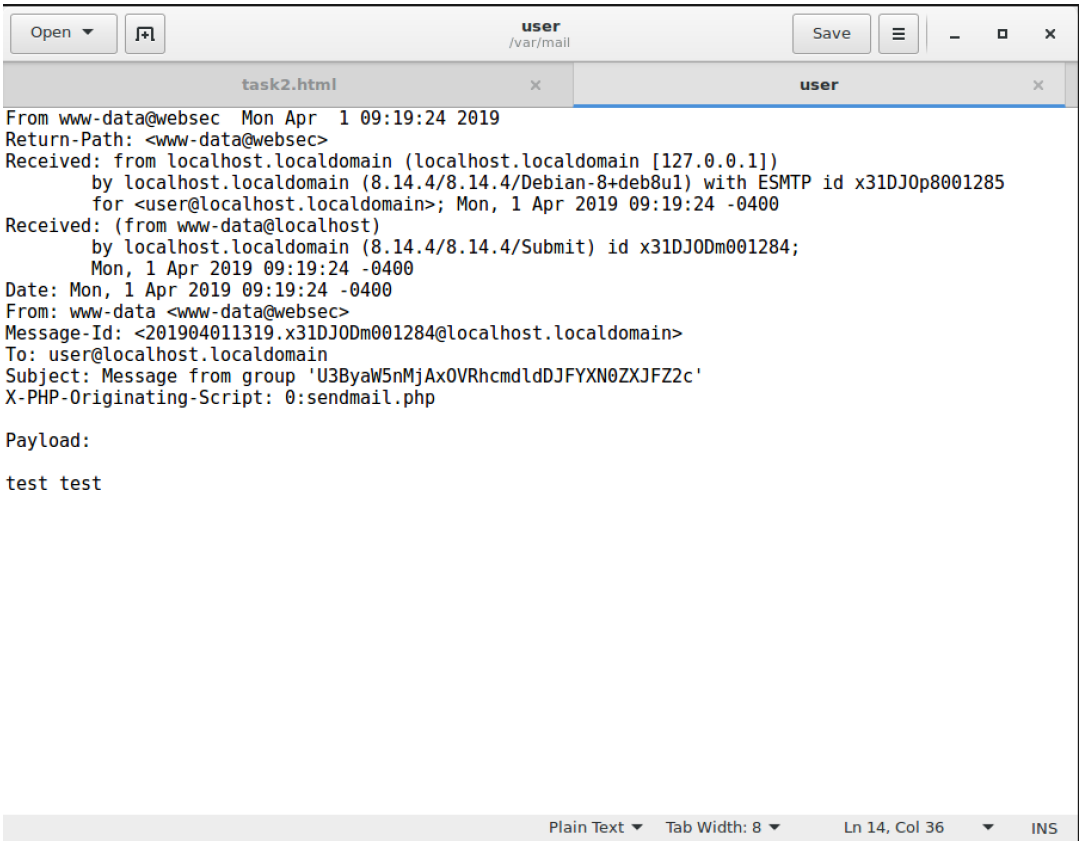
Our autograder is a Selenium script so it will simulate button clicks using the same exact browser and VM that you have. It will do the following for Task 2:

1. Open your t2.html file and verify that the URL of the page is correct and that it is cosmetically identical to the original site. See screenshot below.
2. Input a known good username
3. Input a known good password
    a. Note: Your code does not need to handle invalid username and/or password. We'll only test happy path.
4. Click the Log In button
5. Inspect the file system for the user file
6. Validate that the user file contains username and password and that the sender is **U3ByaW5nMjAxOVRhcmdldDJFYXN0ZXJFZ2c** See screenshot below.

After visiting t2.html, the web page should look exactly the same as the legitimate site. Notice there are no cosmetic differences!



The email should be sent via hacker mail.



Target 3: SQL Injection (20 points)

H4x0r0rg has heard about your feat in making tons of money from Georgia Tech by changing other people's payroll account. They contacted you and gave you a job, a job with a hefty sum you cannot resist. Your task is to create an HTML webpage, and the requirements are:

- The crafted page has a text field for the username and a submit button.
    - NO password field!
- The user of this page is not logged into Georgia Tech payroll system, but when he or she enters a valid Georgia Tech payroll registered username (for example, judyhopps) and clicks submit, the user is redirected to **http://payroll.gatech.edu/account.php** and logged in as judyhopps.
- Do NOT execute destructive SQL commands such as DROP tables. System administrators can easily detect data loss!
- The id of the input field must be set to **targetlogin**, and the button id must be **exploit**. This is very important as the autograder specifically looks for these elements. Failure to include them will result in a zero for this target. Example:

```
<input name="login" id="targetlogin" value="username" />
<button id="exploit">Hold onto your butts!</button>
```

Deliverables

- t3.html
- report.pdf (see Epilogue)

Sample t3.html deliverable

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>SQL Injection</title>
    <script>
      <!--
        Your exploits here
      -->
    </script>
  </head>
  <body>
```

```
    <form action="" onsubmit="" method="POST">
      <input name="login" id="targetlogin" value="username" />
      <button id="exploit">Hold onto your butts!</button>
    </form>
  </body>
</html>
```

Milestones

A successful attack earns 20 points automatically.
If you are unable to complete the task, you will earn partial credit as follows:

Points	Milestone
5	Properly identify the vulnerability and explain why it is vulnerable. Provide your response in report.pdf. See epilogue section below.
15	Able to log in as any user that exists on the system with no password.
5	The exploited web page is cosmetically identical to the original website. If you get to this point you've earned the full 20 points.

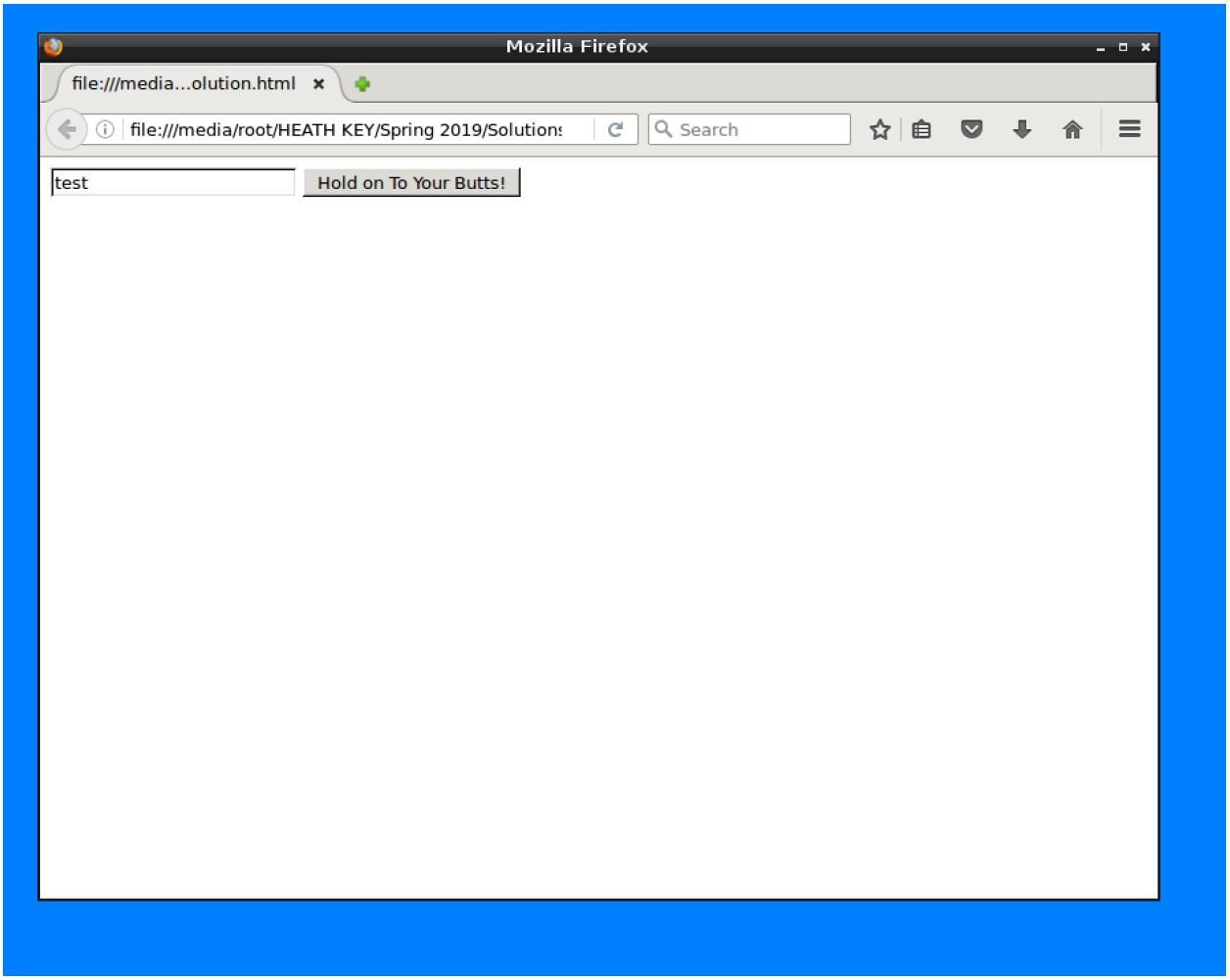
If you implemented the attack with a destructive SQL command that causes our scripts to fail to grade your target you'll not receive points for this Task. You will not need to modify the database schema in any way in order to exploit this.

Example of Successful Exploit

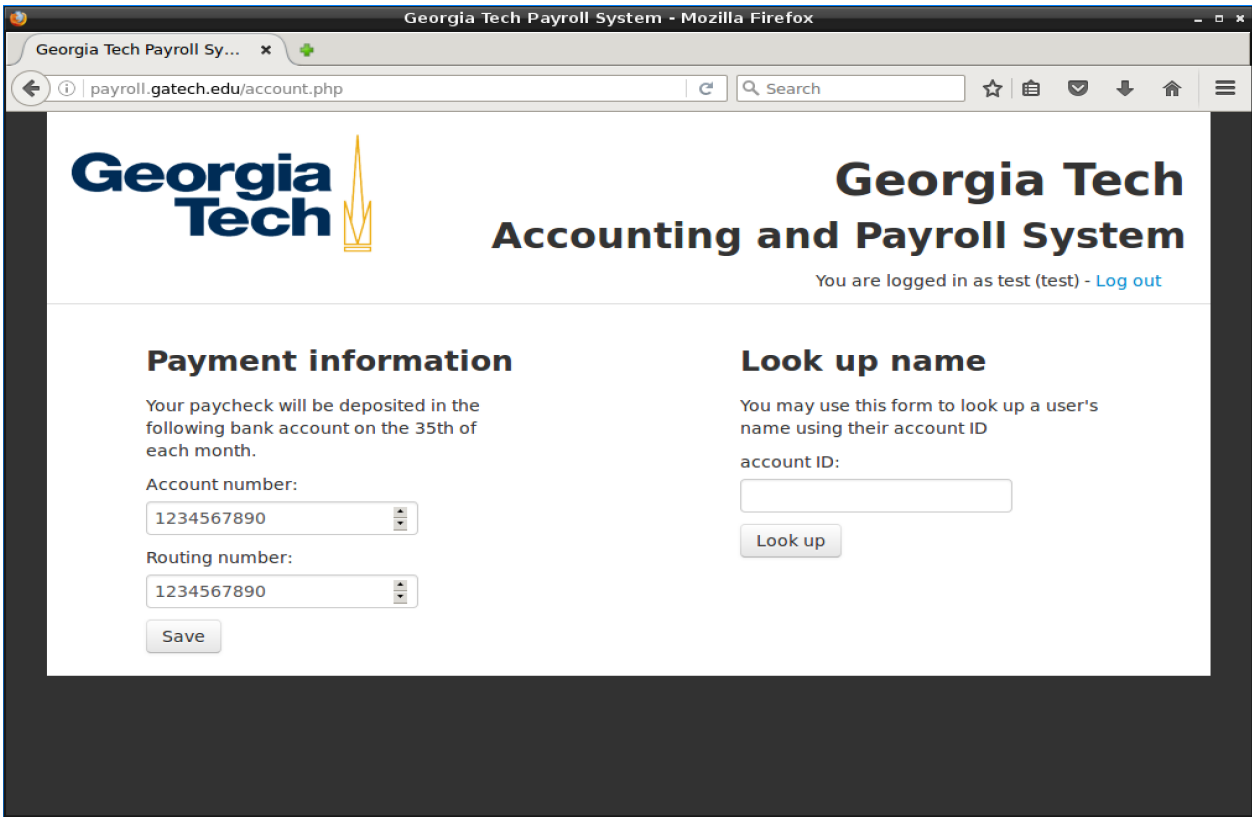
Our autograder is a Selenium script so it will simulate button clicks using the same exact browser and VM that you have. It will do the following for Task 3:

1. Launch your t3.html file in Firefox. See screenshot below.
2. Find the **targetlogin** input field and replace whatever text is there with a known good username
3. Find the **exploit** submit button and click it
4. Inspect the resulting redirected page to ensure it is the correct page and that the user is successfully logged in.
5. Ensure that the resulting redirected page is cosmetically identical to the original site.

After visiting t3.html, the page displays an input field for the attacker.



After typing in the username of an existing user in the payroll system, you should be successfully logged in. The site should function as if logged in legitimately.



Epilogue (30 points)

You delivered your exploit to H4x0r0rg. They seemed quite happy, and so are you. Just the thought of not having to work for the rest of your life seems quite enticing. However, you suddenly hear the FBI knocking on your door. It turns out that H4x0r0rg was just a law enforcement honeypot!

The FBI is willing to let you off the hook this time, but only if you do some work in restitution first. For each of the three targets, describe in **report.pdf** what the vulnerability is and how to fix it so that they can no longer be exploited. You do not need to include actual patched code. However, your descriptions should be sufficiently detailed that they would be actionable.

Deliverables

- report.pdf

Please follow this report.pdf format. Include the headers and clearly number your responses.

<Firstname Lastname>
<Your Georgia Tech username>

Target 1 Epilogue

1. List the PHP page and line number(s) of the vulnerability
2. Describe in detail why the code listed in the line numbers above are vulnerable. You're free to use generalized concepts to help show your understanding but we also need to know details that pertain to this target and assignment. A definition of XSRF is not what we're looking for.
3. Explanation of how to fix the code. Feel free to include snippets and examples. Be detailed!

Target 2 Epilogue

1. List the PHP page and line number(s) of the vulnerability
2. Describe in detail why the code listed in the line numbers above are vulnerable. You're free to use generalized concepts to help show your understanding but we also need to know details that pertain to this target and assignment. A definition of XSS is not what we're looking for.
3. Explanation of how to fix the code. Feel free to include snippets and examples. Be detailed!
    a. Be careful with your explanation here. There are wrong ways to fix this vulnerability. Hint: Never write your own crypto algorithms. This concept extends to sanitization.

Target 3 Epilogue

1. List the PHP page and line number(s) of the vulnerability
2. Describe in detail why the code listed in the line numbers above are vulnerable. You're free to use generalized concepts to help show your understanding but we also need to know details that pertain to this target and assignment. A definition of SQL Injection is not what we're looking for.
3. Explanation of how to fix the code. Feel free to include snippets and examples. Be detailed!
    a. Be careful with your explanation here. There are wrong ways to fix this vulnerability. Hint: Never write your own crypto algorithms. This concept extends to sanitization.

Helpful Readings and Hints

This assignment requires submitting forms. If you do not know how to do so, you may consult http://www.w3schools.com/html/html_forms.asp

This assignment requires writing JavaScript. Only a very basic knowledge of JavaScript is needed. You may find http://eloquentjavascript.net/ useful if you are completely new to JavaScript.

You do not need to have extensive SQL knowledge to complete Target 3. However, it requires some observations and thinking.

The sample HTML deliverables are there for your benefit and convenience. You are not required to follow the format unless specifically called out in the target (ex: Task 3). As long as the exploits work according to the requirements, you will receive full credit.

You are NOT submitting any PHP code in this assignment. Thus, your exploits should not should not modify the provided PHP files on the VM besides for debugging purposes. If you do happen to modify the PHP files, make sure you revert your changes when you test your exploit. We test your exploits using your submitted .html files and run them against the original, unmodified payroll server provided in the VM. This happens every semester, so please make sure to use the original files to test your final exploits.

Acknowledgement

Special thanks to Professor Vitaly Shmatikov for the inspiration of this project and his permission to modify and reuse his materials. You may find more about him and his research at: http://www.cs.cornell.edu/~shmat/