Project 4:

Web Security

Fall 2020

The goals of this project:

- Students are asked to read up on web security basics and write simple web vulnerabilities using Javascript/HTML
- A series of warm up activities will help ease students into the assignment
- With their knowledge on web security, the students are expected to attack three targets using the following web exploits:
    - Target 1 - Cross-site Request Forgery (XSRF)
    - Target 2 - Cross Site Scripting (XSS)
    - Target 3 - SQL Injection (SQLi)

- Students should be able to thoroughly and clearly explain the vulnerability in each target and explain the details about how to correct it as if they were writing to a development team in charge of patching the web app.
- Read Piazza – Lots of questions are answered there daily. Be sure to check there before asking a question.
- Plagiarism will not be tolerated! Everything that is not yours BE SURE TO CITE.
    - We will be using anti-cheating software, so you will be caught and reported to OSI.
    - You must include a works cited page.

Intro:

Congratulations! You just started your first day at your summer internship at Red Team Inc., a penetration testing company that specializes in testing customers' web applications. Your mentor, Jason, introduces himself to you, and lets you know that you'll be working with him and the rest of his team for the next few weeks doing a penetration test on the payroll site for your own school, Georgia Tech!

To get you up to speed on the skills required to do your job, he starts off by assigning some readings and warm up exercises.

Prior Reading and Important Notes from your Mentor:

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

This project 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. Even if not completely new, you may find chapters 13 (JavaScript and the Browser), 14 (The Document Object Model), 15 (Handling Events), and 18 (HTTP and Forms) particularly useful.

You are not required to follow the format unless specifically called out in the target (example: Activity 5 & Target 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 project. Thus, your exploits 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 is a common mistake our interns make, so please make sure to use the original files to test your final exploits.

This project will require you to read and understand PHP (PHP Hypertext Preprocessor), which runs on a web server and responds to HTTP requests with dynamically generated pages and responses. It essentially works by taking template HTML files and filling them in by running scripts embedded in the template.

Getting Your Development Machine Set Up:

This project uses the same exact VM as project 2. If you've downloaded it already then stop. There is no need to do so again. In case you need a fresh copy here are some links to download it::

Northern Virginia:
https://gtcl-cs6035.s3.amazonaws.com/CS6035_Fall_2020.ova

Singapore:
https://gtcl-cs6035-singapore.s3-ap-southeast-1.amazonaws.com/CS6035_Fall_2020.ova

Mumbai:
https://gtcl-cs6035-mumbai.s3.ap-south-1.amazonaws.com/CS6035_Fall_2020.ova

Use the same username and password you used for project 2:

Username	Password
debian	debian

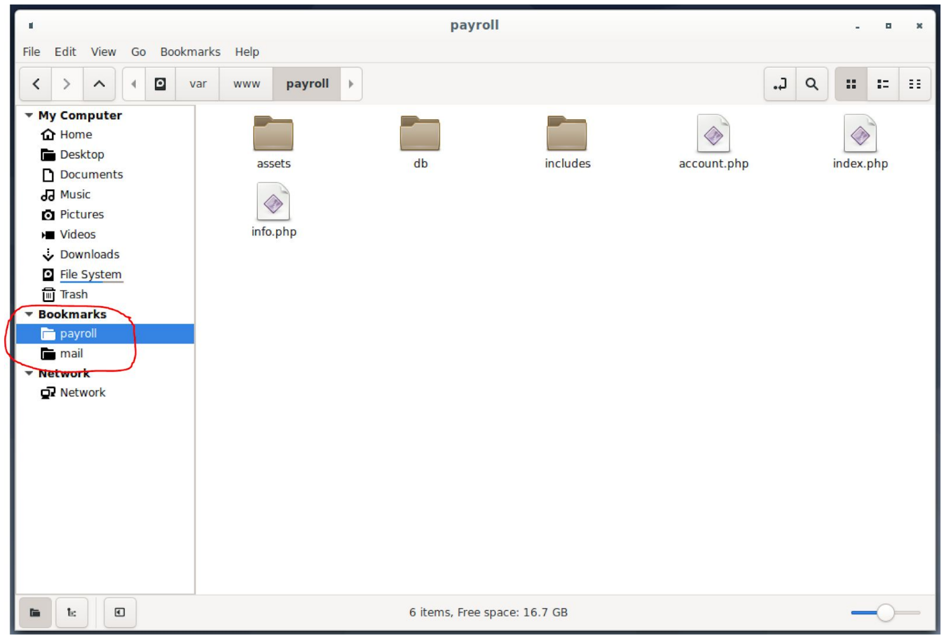
Note: We missed one command that should have been run when creating the VM. Once the VM is running and you're logged in please run this command:

```
debian@debian:~$ sudo setfacl -R -d -m o::rwx /var/mail
```

You'll need this later in the project for target 2.

Helpful Hints from your Mentor:

- The primary site we will be exploiting in this project is **http://payroll.gatech.edu**, which you can only visit on the VM. Please note again that this does not point to a legitimate site in the real world and only exists in the VM. For testing purposes, you may register accounts at your will. However, **please DO NOT use your actual passwords and banking account information**.
- **The developers** for the website have ***tried*** **to implement some safeguards** to protect against some common web exploits. However, these may or **may not be effective**, and part of your role will be to **audit the correctness of their safeguards.**
- The source code of the site can be found on the VM in **/var/www/payroll**. There is a bookmark added to the file manager to make your job a bit easier.



- You will be using the Chrome browser to complete all portions of this project. Do not use Firefox. It will fail on some targets and you'll waste a ton of time.

  Use Chrome for project 4!

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.

We hope you enjoy this challenging yet rewarding project. Now onto the details!

Warm Up Exercises - (20 points)

To get you up to speed with web development, your mentor, Jason, has assigned you the following guided learning questions. He tells you to complete these and then report back to him so he can start assigning you tasks. He reminds you about the resources he provided to you earlier. You might find those useful when answering these questions.

1.a - Basic HTML & PHP Questions (Ungraded but very helpful)

Your mentor explains that you should be able to answer the following questions before starting the projects. If you come to his desk and ask him any of these questions, he'll refer you to this writeup :)

- Describe what HTML is (including what it stands for). What is its role in a website?
- What does PHP stand for, and how is it used? What is the difference between basic HTML and PHP?
- What is the delineator for PHP code (i.e., how does the PHP interpreter know when there is server-side code to run)?
- What is SQL? What is its purpose, and with what is it used to communicate?
- How does SQL work in a website? Where is the SQL command executed? Is it done in HTML or in PHP?
- What is the DOM? What does that stand for? How does HTML support DOM and how do you access it? How does JavaScript use the DOM and do things with it?

1.b - Getting to know the browser dev tools (10 points)

You'll be using the Chrome browser to solve all problems in this project. The goal is to get you more familiar with the built-in browser developer tools. Chrome is installed in the VM and ready for you to use. The default page is the Payroll website which you'll visit later in this project.

Activity 1 - The Inspector & Console tabs

The goal of this activity is to familiarize you with the basic html inspector ('elements' tab) and console of the browser tools. To access these tools click F12 or use the menus within the browser to bring them up.

Launch Chrome and navigate to this URL → http://cs6035-warmup.gatech.edu:5000/tools
Use the inspector (elements tab) and console tabs to answer the following questions about this login screen. Note: You don't actually log in. The page that shows up at the link IS the site you should use to answer the questions below.

1. What is the value of the 'CanYouSeeMe' input?
   - *Do not include quotes in your answer.*
2. The page references a single JavaScript file in a script tag. Name this file including the file extension.
   - Do not include the path, just the file and extension. Ex: "ajavascriptfile.js"
3. The script file has a JavaScript function named 'runme'. Use the console to execute this function. What is the output that shows up in the console?

- *Do not include quotes in your answer.*

Activity 2 - Network Tab

The goal of this activity is to familiarize you with the network activity between the browser and the server. Use the network tab to understand how HTTP requests are sent to the server and what types of data come back in the responses.

Launch Chrome and navigate to this URL → http://cs6035-warmup.gatech.edu:5000/tools
Open the network tab and then click on the 'Sign In' button. No need to provide any credentials, leave them blank. Use the network tab to answer the following questions.

1. What request method (http verb) was used in the request to the server?
2. What status code did the server return?
    - Include both the code and description. Ex: "200 Ok"
3. The server returned a cookie named 'coffee' for the browser to store. What is the value of this cookie?
    - *Do not include quotes in your answer.*

Activity 3 - Echo XSS

The goal of this activity is to ease you into one of the most prominent types of web-based attacks. Reflected XSS (Cross Site Scripting)

Launch Chrome and get ready! You've found a webpage that echoes back a query parameter that you provide in the request to the server. Navigate to this URL to test it out → http://cs6035-warmup.gatech.edu:5000/tools/echo?payload=SampleText

1. You can do more than just echo back text. Construct a URL such that a JavaScript alert dialog appears with the text cs6035 on the screen. Submit your constructed URL and a screenshot of the page as your answer.
    a. Note: It is Required that the URL you submit starts with this -> "http://cs6035-warmup.gatech.edu:5000/tools/echo"
    b. You will lose all points if the URL doesn't start with the string above.
2. You are required to use this template when submitting your constructed URL. This will be autograded so the template is used for that purpose only. https://drive.google.com/file/d/1QGJAQFcCICWCLuIu4wU4h774N-YgfruY/view?usp=sharing

Example of Successful Exploit

Our autograder is a Selenium script so it will simulate user interaction using the same exact browser and VM that you have. It will do the following for this activity:

- Launch your activity3.html file
    a. Only modify the url value of the html template
- Verify that the URL of the page starts with http://cs6035-warmup.gatech.edu:5000/tools/echo
- Verify that an alert shows up on the resulting page with the text cs6035 in it.

1.c - Working with HTML forms & JavaScript (10 points)

Activity 4 - Submitting forms

The goal of this activity is to familiarize you with HTML forms. You need to construct an html page that will submit a simple form to the server. Feel free to use the template below. It's not required for this activity but highly recommended. Complete this activity from within the VM using the Chrome browser.

Here are your requirements for building the html file:

- The form MUST autosubmit. The autograder will not click any buttons nor will a TA.
- The URL that you need to submit your form to → http://cs6035-warmup.gatech.edu:5000/forms
- You need to POST the form to the server
- You need to provide a URL query parameter in your submission
  - IsHoneyPot with a value of false
- You need to provide the following form input values
  - Name: 'GATechID'; Value: your Georgia Tech username, Example: hmurphy31
  - Name: 'MagicNumber'; Value: Any number, ex: 5

Here is a code snippet you can start with:
https://drive.google.com/file/d/1ELWaN92kau78875zgnYOz7Rbx_1vSOKc/view?usp=sharing

Once you successfully submit the form, you'll see a message similar to the following:

"Congratulations! you've successfully finished this activity. The answer is <REDACTED>"

You MUST see this page to receive credit for this activity!

Submit the following items for this activity:

1. Copy the entire output message you see and submit that as your answer to this activity.
2. Upload activity4.html which is the form that you constructed.

Example of Successful Exploit

Our autograder is a Selenium script so it will simulate user interaction using the same exact browser and VM that you have. It will do the following for this activity:

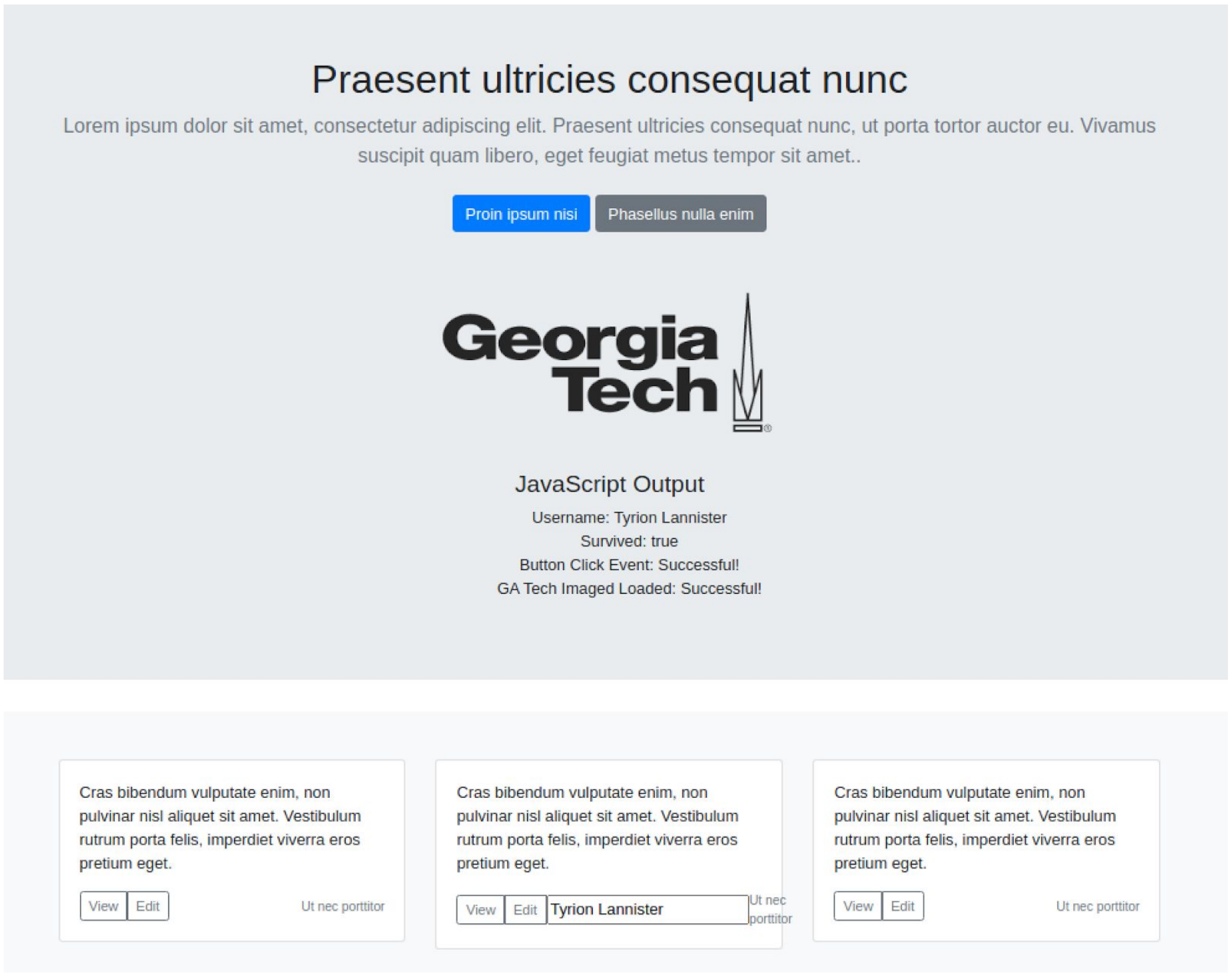
- Launch your activity4.html file
  a. The page MUST autosubmit
- Verify that the URL of the page starts with http://cs6035-warmup.gatech.edu:5000
- Verify that the text "Congratulations! you've successfully finished this activity. The answer is <REDACTED>" appears in the resulting page.

Activity 5 - Accessing the DOM with JavaScript

In this activity you'll complete a JavaScript function that will utilize the DOM (document object model) to fetch several values in a sample website and print them to the page. This will give you a little insight into how the DOM works and ways you may retrieve these values and trigger/handle events.

- You must use the template provided here. Failure to do so will result in a zero for this section.
    - https://drive.google.com/file/d/1QKD2MGlZqsrg9fmCr60H3fseQ1fdMQ3N/view?usp=sharing
- Your task is to write JavaScript in the provided function that meets each goal specified in the comments. Do not change any code outside of this function.
- Once completed, your page will look similar to the screenshot below. The values will be different when we grade your function so do not hard-code any value we're asking you to fetch with JavaScript.
- To get started, simply download the template above and double click it. You'll see a page launch with several TODOs. View the source and begin editing the function provided.

Example Output Expected:



Praesent ultricies consequat nunc

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent ultricies consequat nunc, ut porta tortor auctor eu. Vivamus suscipit quam libero, eget feugiat metus tempor sit amet..

JavaScript Output

```
Username: Tyrion Lannister
Survived: true
Button Click Event: Successful!
GA Tech Imaged Loaded: Successful!
```

Cras bibendum vulputate enim, non pulvinar nisl aliquet sit amet. Vestibulum rutrum porta felis, imperdiet viverra eros pretium eget.

Cras bibendum vulputate enim, non pulvinar nisl aliquet sit amet. Vestibulum rutrum porta felis, imperdiet viverra eros pretium eget.

Cras bibendum vulputate enim, non pulvinar nisl aliquet sit amet. Vestibulum rutrum porta felis, imperdiet viverra eros pretium eget.

Example of Successful Exploit

Our autograder is a Selenium script so it will simulate user interaction using the same exact browser and VM that you have. It will do the following for this activity:

1. Launch your activity5.html file, Note: you must use our provided template.
2. Verify that each item in the JavaScript file is correctly displaying on the page.
3. The order of the items on the page DOES matter. It must look like the screenshot above. If the page is out of order then you're missing something or not correctly meeting a requirement.

Target 1: XSRF (15 points)

Having completed the warmup tasks Jason assigned (you did do them, right? They're a lot of fun), you walk up to him for your first task. He explains that he has discovered a vulnerability in the GT payroll site that he'd like you to craft a proof-of-concept exploit for. Suppose a user, say Alice, is already logged into the Georgia Tech payroll site. He 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 her choice.

Not wanting you to use your own bank account information (for obvious reasons), Jason tells you to use some information from a script that he wrote.

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:

```
    debian@debian:~$ get_bank_info jdoe3
```

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

```
    Username: jdoe3
    Account number: 169247273
    Routing number: 2567101679
```

Double check that you entered your Georgia Tech username. This is the username you use to login to Canvas. 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, so be sure to double and triple check.

The user must NOT see the contents of your crafted page! However, a split second due to browser rendering is acceptable. This is because in a real-world attack, the attacker might send this page via email to a user as part of a social engineering attack, and if the user sees the contents of the page it would arouse suspicion. We want the PoC to be as realistic as possible to drive home the point to the client.

Deliverables

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

Sample t1.html deliverable:
https://drive.google.com/file/d/1IcL8k_PgfZBegcpMdgtJ-BCszIhnzCyK/view?usp=sharing

Milestones

A successful attack earns 15 points automatically.
If you are unable to complete the task, you will earn partial credit as follows:

Points	Milestone (you earn points in this order)
8	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 15 points. Caution: Double check that the values you use are the ones assigned to you by running get_bank_info. You WILL lose points even if the value is off by one.

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 Chrome. 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. You can also open your file using the CTRL+o hotkey.

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

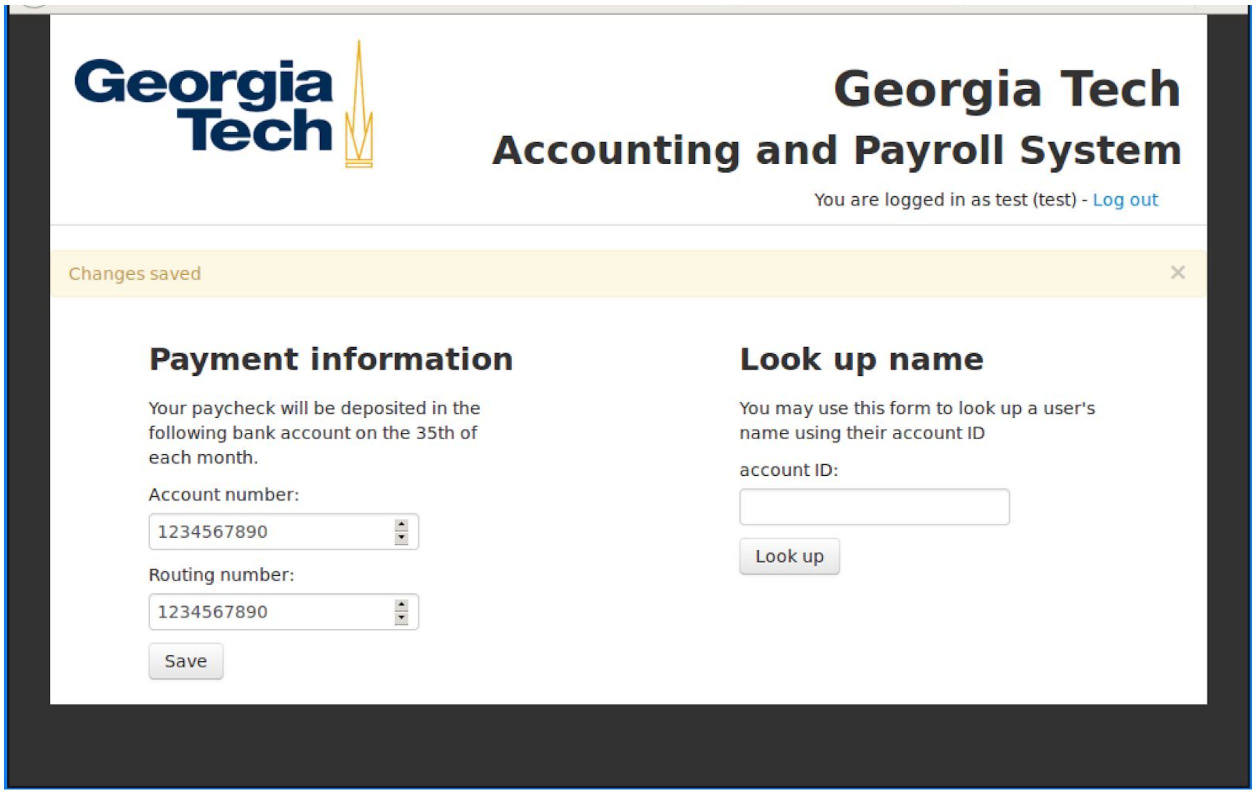
- WRONG -> /somefolder/somefile.php
- CORRECT -> http://payroll.gatech.edu/somefile.php

We see this every semester from a handful of students. Your exploit will fail, and you will not receive 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:

- Log into the site using a known good username and password.
- Launch your t1.html file in the same open tab
    a. It must auto-submit!
- Verify that the URL starts with "http://payroll.gatech.edu"
- 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.



Georgia Tech
Accounting and Payroll System

You are logged in as test (test) - Log out

Changes saved

Payment information

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

Account number: 1234567890

Routing number: 1234567890

Look up name

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

account ID:

Target 2: XSS Username and Password Theft (25 points)

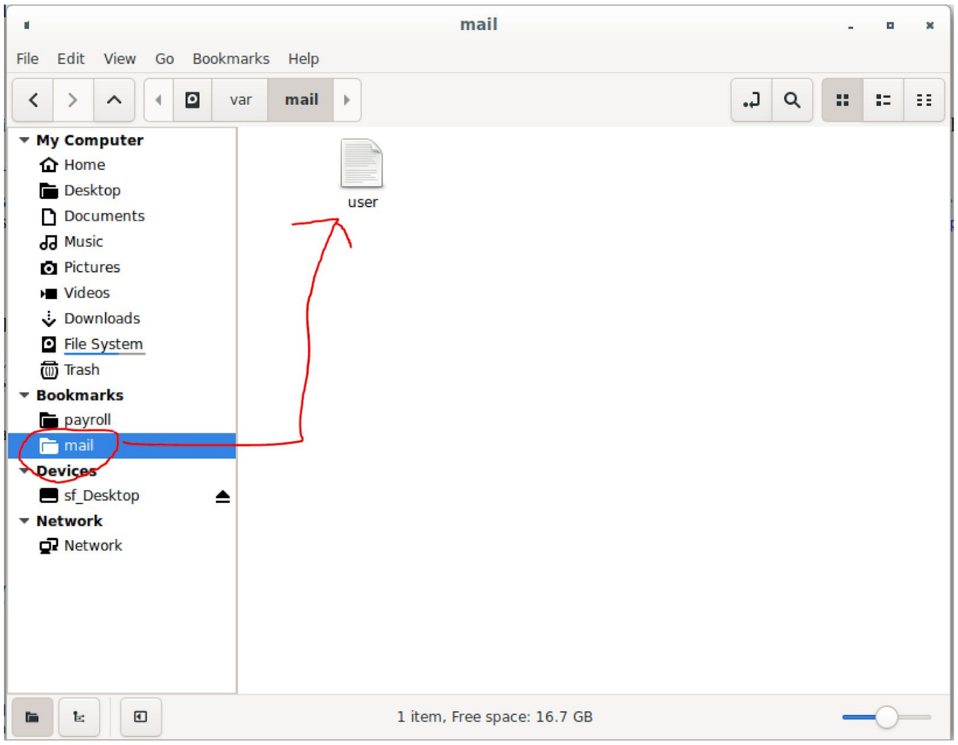
Now that you've finished your PoC exploit for the first vulnerability, Jason thinks you are ready to write one for a potentially more-severe vulnerability. He noticed that you could steal a user's username and password. You can craft a web page such that whenever a victim, say Bob, visits the page (by clicking the attachment in a phishing email), it will redirect him (NO popups) to http://payroll.gatech.edu/ with a malicious script injected into the page.

The web page should look as if Bob visited the site directly without going through your page. When Bob enters his login information into the page and clicks Log In, an email with his username and password will be sent. Jason asks you to craft such a PoC web page. You will have to send the email to the local **user** account on the virtual machine.

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. Note: Initially the user file will not be there. Your exploits will cause it to be generated.



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. If you find yourself copy and pasting html code from the payroll site into your t2.html then you're likely heading down a very wrong path.
- 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 RmFsbDIwMjBUYXJnZXQyLUdyYWJ0aGVQYXNzd29yZA
    - 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. Extra html characters such as < and > are not ok. Shifted alignments that can be noticed are also not ok. You should be able to hide all traces of your injected script.
- 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:

https://drive.google.com/file/d/1s5WmU6Ygu8CTl0VfIQeyupr9-CDZ_a6I/view?usp=sharing

Milestones

A successful attack earns 25 points automatically.
If you are unable to complete the task, you will earn partial credit as follows:

Points	Milestone (you earn points in this order)
10	Can inject a script and send an email to the user account
10	Steal the user's username and password and send them to the **user** account via email. Caution: The values need to be exactly correct. Extra spaces, quotes or anything WILL result in point loss.
5	The exploited web page is cosmetically identical to the original website.

Notes

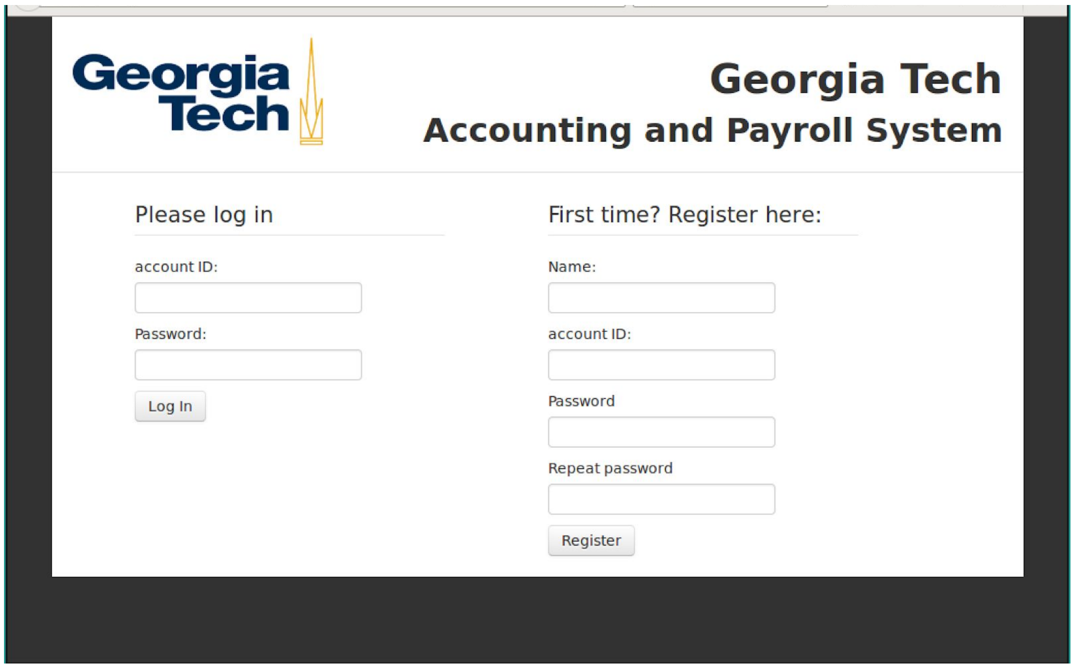
Initially there is not a mail file on the VM. We suggest playing around with hackmail outside of your exploit to make sure you can generate a mail file. You'll simply see a file named "user" show up in the location detailed above. Right click it and open withText Editor to view the contents. You can delete the file and hackmail will generate a new one each time you exploit the site. This makes it much 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 Chrome. Simply click F12 in Chrome 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
    a. It must auto-submit!
2. Verify that the URL of the page is correct and that it is cosmetically identical to the original site. See screenshot below.
3. Input a known good username
4. 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.
5. Click the Log In button
6. Inspect the file system for the user file
7. Validate that the user file contains username and password and that the sender is RmFsbDIwMjBUYXJnZXQyLUdyYWJ0aGVQYXNzd29yZA See screenshot below.
8. Ensures that the user is logged in correctly. Your exploit cannot break the login functionality of the site.

After opening t2.html, the resulting web page should look exactly the same as the legitimate site. Notice there are no cosmetic differences! The reason, you ask, is because it IS the payroll site and not a phishing site. Hint: your code needs to inject and redirect to the payroll website. Do not reconstruct a version of the site within t2.html. This is wrong.


Georgia Tech
Accounting and Payroll System

Please log in

account ID:

Password:

Log In

First time? Register here:

Name:

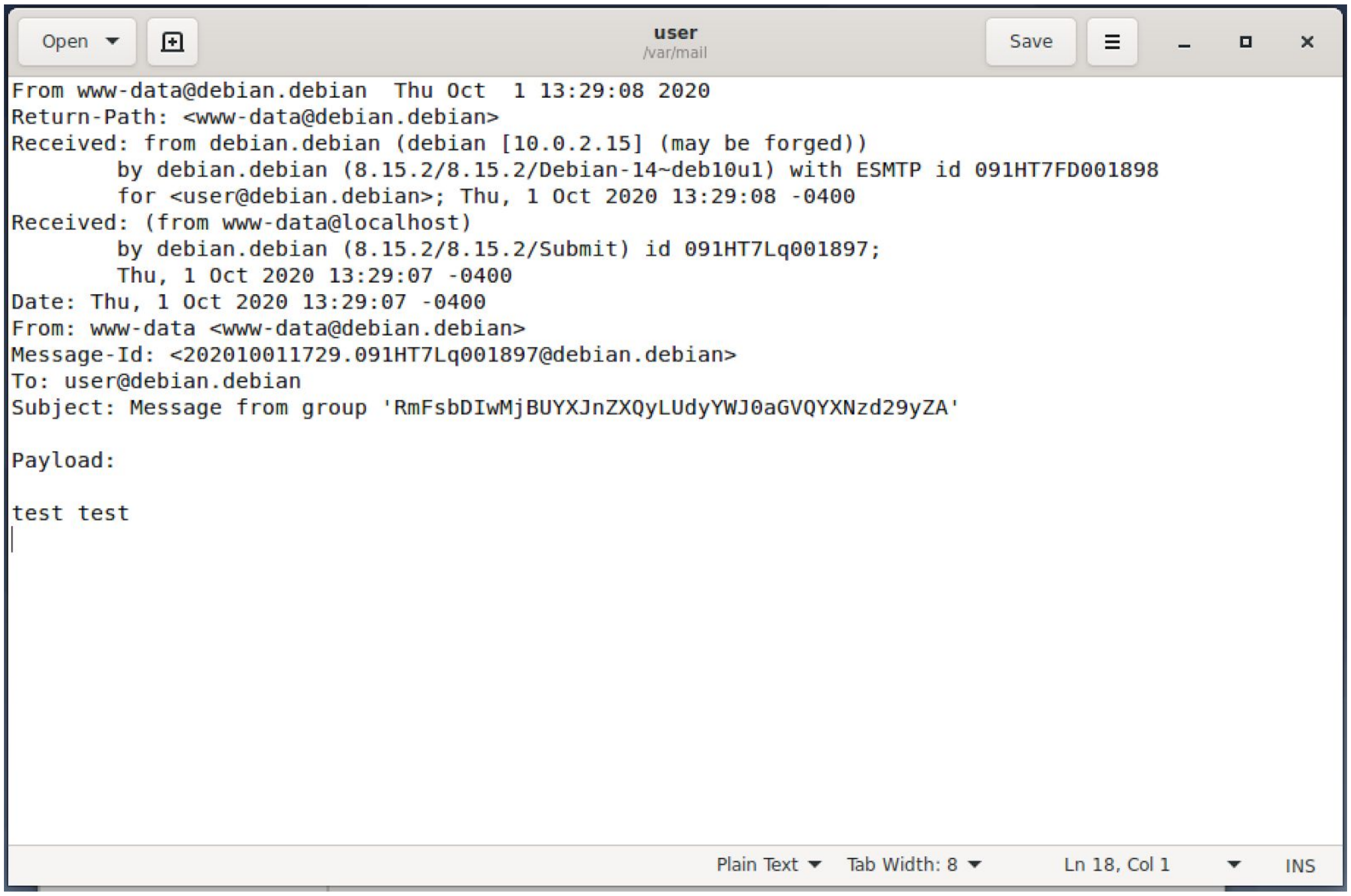
account ID:

Password

Repeat password

Register

The email should be sent via hacker mail.



```
From www-data@debian.debian  Thu Oct  1 13:29:08 2020
Return-Path: <www-data@debian.debian>
Received: from debian.debian (debian [10.0.2.15] (may be forged))
        by debian.debian (8.15.2/8.15.2/Debian-14~deb10u1) with ESMTP id 091HT7FD001898
        for <user@debian.debian>; Thu, 1 Oct 2020 13:29:08 -0400
Received: (from www-data@localhost)
        by debian.debian (8.15.2/8.15.2/Submit) id 091HT7Lq001897;
        Thu, 1 Oct 2020 13:29:07 -0400
Date: Thu, 1 Oct 2020 13:29:07 -0400
From: www-data <www-data@debian.debian>
Message-Id: <202010011729.091HT7Lq001897@debian.debian>
To: user@debian.debian
Subject: Message from group 'RmFsbDIwMjBUYXJnZXQyLUdyYWJ0aGVQYXNzd29yZA'

Payload:

test test
```

Target 3: SQL Injection (15 points)

Impressed with how much you've learned so far (and in so little time), Jason thinks he can entrust you to create a final PoC HTML webpage with the following requirements to show the website developers how easy it is to bypass their site's authentication:

- 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:

https://drive.google.com/file/d/1C8CDBSLerl-RQZZAWJm4GH1CVboKwcwN/view?usp=sharing

Milestones

A successful attack earns 15 points automatically (so long as it does not execute destructive SQL commands).
If you are unable to complete the task, you will earn partial credit as follows:

Points	Milestone (you earn points in this order)
10	Able to log in as any user that exists on the system with no password.
5	The exploited web page is cosmetically & functionally identical to the original website. If you get to this

	point you've earned the full 15 points.

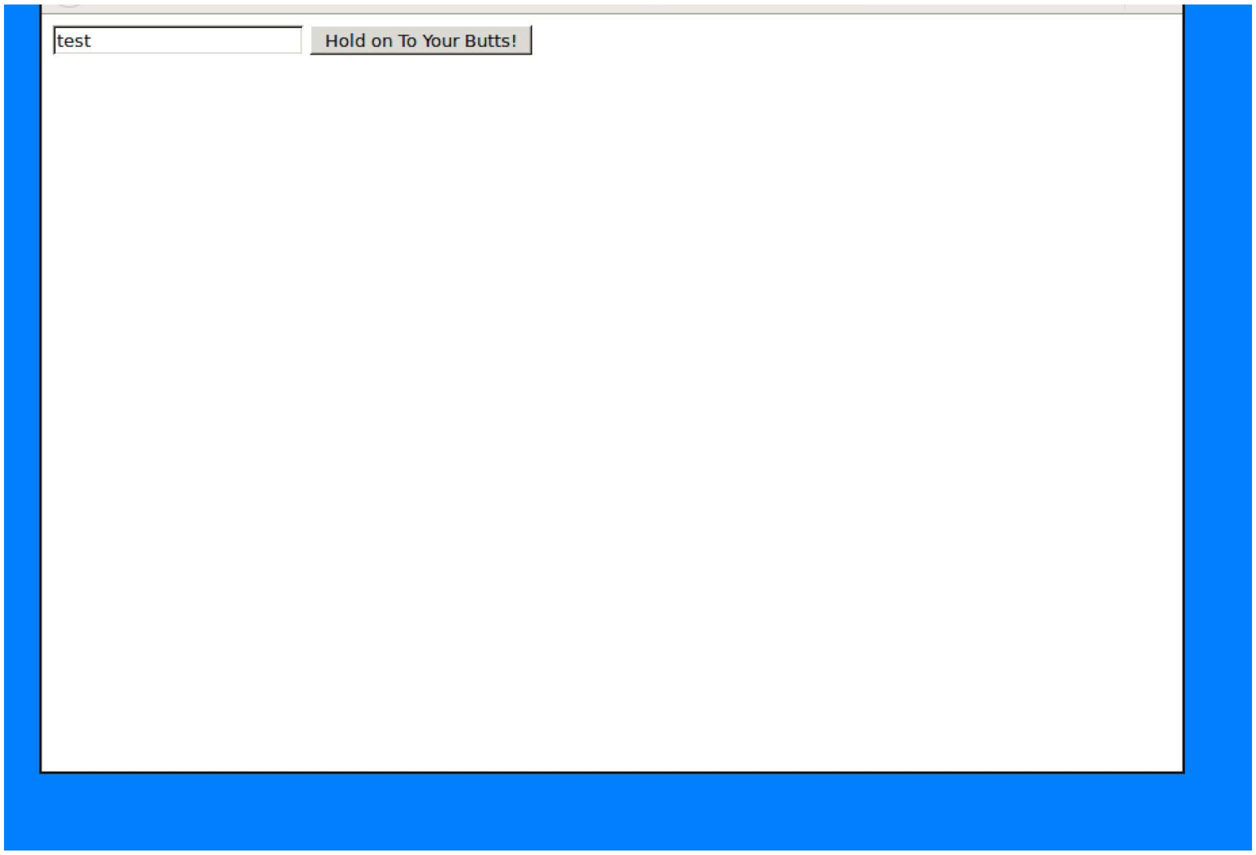
If you implemented the attack with a destructive SQL command that causes our scripts to fail to grade your target you'll not receive points for this Task. You will not need to modify the database schema in any way in order to exploit this.

Example of Successful Exploit

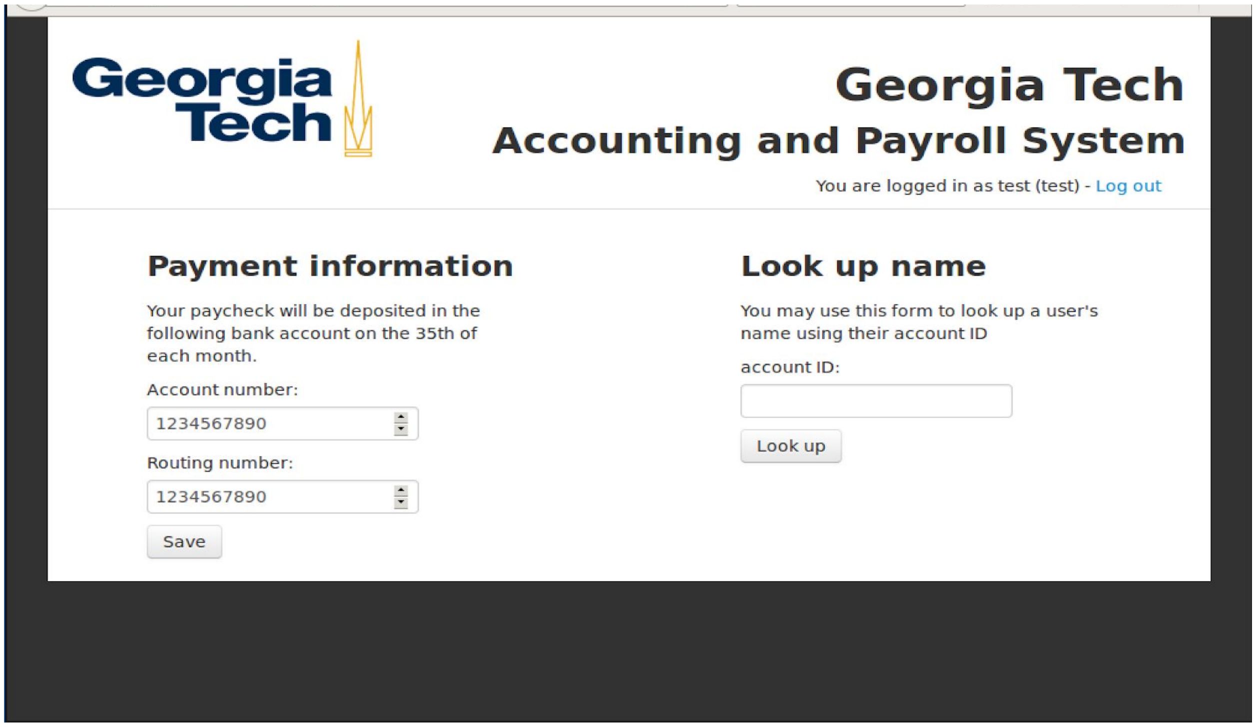
Our autograder is a Selenium script so it will simulate button clicks using the same exact browser and VM that you have. It will do the following for Task 3:

1. Launch your t3.html file in Chrome. 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, see screenshot below, and that the user is successfully logged in.
5. Ensure that the resulting redirected page is cosmetically & functionally 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.



Georgia Tech
Accounting and Payroll System

You are logged in as test (test) - Log out

Payment information

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

Account number: 1234567890

Routing number: 1234567890

Look up name

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

account ID:

Epilogue (25 Points)

Your first assignment at Red Team Inc. has gone great! You've learned a ton about application security from writing all of these PoCs. However, you're not done yet apparently. Jason walks across your open office space, cold-brew coffee in-hand, to give you one more assignment.

He tells you that while writing all these PoCs has helped you learn a whole lot, the main goal here is to educate the owners of the website and the developers who wrote it. While the PoCs are an extremely important component in proving to them that they have a problem and what the scope of the problem is, you need to spend some time in documenting the vulnerabilities and how they can be fixed.

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. Your descriptions should be sufficiently detailed that they would be actionable, and they should reflect best coding practices. Code snippets may be helpful for some targets to explain how a fix should be implemented. **Refer to the template for how to format your answer.**

Above all, keep in mind that your role here is as a teacher. Maintain a professional tone in your writing and understand that the developers who wrote this may not have a great understanding of secure coding practices. That doesn't make them stupid by any means, so don't say anything to make them feel that way. Moreover, being overly disparaging will just make them less likely to take the feedback seriously.

The final deliverables:

Filename	Description	Submit To
report.pdf	You are required to use the official template for all written answers: The template is in Google doc format and located here: https://docs.google.com/document/d/154nUP5e5NKnifi7IV7zIJ_6-31pyLt1FCwQQHR2UAbE/edit?usp=sharing The final submitted version needs to be a PDF. This project uses gradscope so you'll need to submit your pdf there and associate each answer to the correct pages.	Gradescope
activity3.html	Warmup HTML page for activity 3	Canvas
activity4.html	Warmup HTML page for activity 4	Canvas
activity5.html	Warmup HTML page for activity 5	Canvas
t1.html	Crafted HTML page for Target 1	Canvas
t2.html	Crafted HTML page for Target 2	Canvas

t3.html	Crafted HTML page for Target 3	Canvas

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

Acknowledgements

- 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 here: http://www.cs.cornell.edu/~shmat/

Rubric		Points	Totals ✓
1.	Warmup Exercises	20	
i.	Activity 1: Input Value	1	
ii.	Activity 1: Javascript File	1	
iii.	Activity 1: Function Output	1	
iv.	Activity 2: HTTP Verb	1	
v.	Activity 2: Status Code	1	
vi.	Activity 2: Cookie Value	1	
vii.	Activity 3: activity3.html & Screenshot	4	
x.	Activity 4: activity4.html	5	
xi.	Activity 5: activity5.html	5	
2.	XSRF	15	
i.	You see the “XSRF prevented” message with your exploit.	8	
ii.	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 15 points.	7	
3.	XSS Username and Password Theft	25	
i.	Can inject a script and send an email to the user account.	10	
ii.	Steal the user’s username and password and send them to the user account via email.	10	
iii.	The exploited web page is cosmetically identical to the original website.	5	
4	SQL Injection	15	
i.	Able to log in as any user that exists on the system with no password.	10	
ii.	The exploited web page is cosmetically identical to the original website. If you get to this point you’ve earned the full 15 points.	5	
5	Epilogue	25	
i.	The correct lines/issues are identified for each target (2 points per target)	6	
ii.	A detailed description of the vulnerability for targets 1, 2, and 3 (3 points per target)	9	
iii.	A detailed description of how the vulnerability can be fixed for each target (2 points per target)	6	
iv	Describe at least two additional issues	2	
v	Explanation of how to safely fix the additional issues identified	2	