Name: Truc Ho Nguyen

NetID: thv20002

Section: 8

VM IP Address: 172.16.49.54

1A. For Question 1A, I opened the file Q1login to see the login information for users AQuinnita54 and BNeena54. Then, I logged into BNeena54 on the bank website. To find the cookies, I right-clicked on the webpage, clicked on “Inspect”, clicked on “Application”, and the appropriate cookie was associated with the name “LOGIN\_INFO” (eac7ccb170414fd1990186409a7c9d58b4652db8fe37cb0aabed24583417e515).

1B. For Question 1B, I wrote Q1B.JS as a template for a simple website that sets 3 different cookies, then used Q1B.py to render the template. To verify that the cookies were set, I once again used my browser’s “Inspect” option. I made sure to add Q1B2 under the path “/Q1B2” and Q1B3 with the “Secure” attribute to help prevent exposure to a rogue cross-site script.

2. For Question 2, I wrote Q2.JS and Q2.py to issue a request to the banking site that would transfer $5 from the B account to the A account. To test the attack, I repeatedly loaded my website, taking note of the balance beforehand, then compared it with the balance afterwards to make sure $5 was transferred every time.

3. For Question 3, I wrote Q3.JS and Q3.py to embed the banking website into my own, with the query variables filled into the source url. Then, my scripts sent an alert with all of the site’s cookies. I only needed to find the cookie paired with the key “Q3Cookie” (46858).

4. For Question 4, I wrote Q4FindMagicNumber.py to repeatedly post requests to the bank’s Q4 page and return the correct magic number after scanning the response text. Then, I wrote Q4.JS and Q4.py to embed the Q4 page into my own website, with the query variables filled into the source url. My scripts sent an alert with all of the site’s cookies. I only needed to find the cookie paired with the key “magicCookie” (51397).