

ITIS 6200/8200 Principles of Information Security and Privacy

Homework 3

Please **briefly** explain your answer.

Question 1. Access Control (10 points)

Alice can read and write to the file x, can read the file y, and can execute the file z. Bob can read x, can read and write to y, and cannot access z.

Q 1.1: Write a set of access control lists for this situation. Which list is associated with while file?

Q 1.2: Write a set of capability lists for this situation. With what is each list associated?

Question 2. Cookies (20 points)

Q 2.1: For each of the following webpages, determine whether the webpage has the same origin as <https://cci.charlotte.edu>, and provide a **brief** justification.

- i. <https://cci.charlotte.edu/sis-faculty/>
- ii. <https://www.charlotte.edu>
- iii. <https://cci.charlotte.edu:443>
- iv. <http://cci.charlotte.edu/>

Q 2.2: Describe how to setup a cookie so it will be sent to only <https://cci.charlotte.edu> and its subdomains.

Q 2.3: How can <https://cci.charlotte.edu> ensure that cookies are only transmitted encrypted?

Q 2.4: How can <https://engr.charlotte.edu/> set a cookie it may affect <https://cci.charlotte.edu>?

Q 2.5: **Github** hosts user sites on **github.io** instead of **github.com**, i.e., **[username].github.io**, why do you think **Github** do that? Why don't Github host user sites as **[username].github.com**? How can it help defend against cookie related attacks?

Question 3. CSRF (25 points)

A CSRF attack exploits cookie-based authentication to perform an action as the victim. Consider the following example. Mallory posts the following in a comment on a chat forum:

``

To successfully conduct a transaction in *bank.com*, users need to authenticate first. Then, *bank.com* sets a cookie as session token.

Q 3.1: Explain what could happen when Alice visits the chat forum and views Mallory's comment.

Q 3.2: Suppose *bank.com* decides to defend against CSRF attacks with a cookie-based CSRF token, as follows:

- a. When a user logs in, *bank.com* sets a cookie `csrf_token` with domain attribute of *bank.com*.
- b. When the user sends a POST request, the value of the `csrf_token` is embedded as one of the form fields.
- c. On receiving a POST request, *bank.com* checks that the value of the `csrf_token` cookie matches the one in the form.

If the chat forum has domain *evil.com*, can the CSRF attack above succeed? If the chat forum has domain *evil.bank.com*, can the CSRF attack succeed?

Q 3.3: Suppose *bank.com* decides to defend against CSRF attacks by checking if the **Referer** header contains a string "bank.com". If the chat forum has domain *evil.com*, can the CSRF attack succeed? If the chat forum has domain *evil.bank.com*, can the CSRF attack succeed? Describe one way Mallory can modify her attack to always get around this check.

Q 3.4: Suppose *bank.com* decides to defend against CSRF attacks with an additional cookie field **SameSite**. When **SameSite**=strict, the browser will only send the cookie if the domain of the cookie exactly matches the domain of the origin. If the chat forum has domain *evil.com*, can the CSRF attack succeed? If the chat forum has domain *evil.bank.com*, can the CSRF attack succeed?

Question 4. XSS (25 points)

A website uses a Go handler is processing HTTP requests to URL **https://vulnerable.com/hello** that takes an argument of name. The website is vulnerable to XSS attacks.

```
func handleSayHello(w http.ResponseWriter, r *http.Request) {  
    name := r.URL.Query()["name"][0]  
    fmt.Fprintf(w, "<html><body>Hello %s!</body></html>", name)  
}
```

Q 4.1: Design a GET request that would run the JavaScript file **http://evil.com/hack.js**.

Q 4.2: When the script **hack.js** runs, what origin does it has?

Q 4.3: Can we use XSS to steal information in Cookies? If yes, how can we defend against that?

Q 4.4: Design a GET request that can launch a CSRF attack shown in Question 3, i.e., a CSRF attack to *bank.com* than can transfer \$1000 to Mallory.

Question 5. SQL injection (20 points)

A student forum stores its member information with the following schema. The server is vulnerable to SQL injection.

```
CREATE TABLE students (  
    StudentID INT,           -- member ID  
    Username VARCHAR(255),  -- User name  
    Password INT            -- member password  
);
```

When a new member signs up, the following code runs:

```
query := fmt.Sprintf(  
    "INSERT INTO students (StudentID, Username, Password)  
        VALUES ('%s', '%s', '%s');",  
        id, username, password  
)  
db.Exec(query)
```

Q 5.1: Design an input that would delete the table students. You can consider format of the inputs as: id = [some text]; username = [some text]; password = [some text].

Q 5.2: Assume that the forum lets users log in with a username and password. For example, if a user submits the username Alice and the password 123456, the application checks the credentials by performing the following SQL query:

```
SELECT * FROM users WHERE username = 'Alice' AND password = '123456'
```

If the query returns the details of a user, then the login is successful. Otherwise, it is rejected. Design an input that would bypass the password check.