Popa & Weaver
Spring 2019

CS 161
Computer Security

Homework 4

Due: Sunday, 28 April 2019, at 11:59pm

Instructions. This homework is due **Sunday, 28 April 2019, at 11:59pm**. No late homeworks will be accepted unless you have prior accomodations from us. This assignment must be done on your own.

Make sure you have a Gradescope account and are joined in this course. The homework *must* be submitted electronically via Gradescope (not by any other method). Your answer for each question, when submitted on Gradescope, should be written in the space provided on this PDF form. You may either use the LaTeX form provided to fill out your responses, use Adobe Acrobat to fill in this fillable PDF, or print this paper out and handwrite your solutions, but **please make sure your responses do not overflow the box provided before submitting to ensure that you get full credit for your response.**

REPEAT: DO NOT OVERFLOW THE BOXES. ONLY WHAT IS VISIBLE WILL BE GRADED.

Problem 1 ***True-or-False Questions*** **(9 points)**

Answer each question. You don't need to justify or explain your answer. Check the box if it is True and leave it blank if it is False.

(a) [X] Prepared statements are a good defense against SQL injection.

(b) [ ] Setting the "secure" flag on a cookie (so it will only be sent over HTTPS) is a good defense against CSRF.

(c) [ ] Setting the "secure" flag on a cookie (so it will only be sent over HTTPS) is a good defense against XSS cookie-leaking.

(d) [X] Setting the "HTTPOnly" flag on a cookie is a good defense against XSS cookie-leaking.

(e) [ ] Switching over all application requests to HTTP Post stops all CSRF attacks.

(f) [ ] SOP prevents XSS attacks.

(g) [ ] Two Javascript scripts embedded in pages running in two different tabs on a user's browser can never access the resources of each other.

(h) [ ] Browsers have a private browsing mode, which prevents websites from storing cookies on your computer altogether.

(i) [X] Because of the cookie policy, you cannot be tracked across domains by cookies.

Problem 2 ***Web Security Warm-Up*** **(15 points)**

(a) Oski owns a conglomerate, OskiBankAndServices.com. He hopes to compete with Google by combining online banking together with web services, such as web hosting. As part of his business plan, Oski decides to host a website creation service at `oskiwebhosting.com/[SITENAME]`. This service allows you to choose your own `SITENAME` and upload any script or HTML that you desire. Why is this a better design than putting user sites on `OskiBankAndServices.com/sites/[SITENAME]`?

By cookie origin policy, it can prevent malicious script from accessing cookies in OskiBankAndServices.com.

(b) Your friend Chad has decided to create a new microblogging service for aspiring presidential candidates but with the option to choose your intended audience. This way if you want to post something to pander to your base you can do so without offending another demographic! He informs you that he can handle the business side and tasks you with building the web-based sharing form, PresidentialTweets.gov. You have set up a simple form with just two fields, the text to share and the intended audience. When a user clicks submit, the following request is made:

```
https://www.presidentialtweets.gov/share?text=<the text to
    share>&audience=<the chosen demographic>
```

You show this to your bro Vladimir, and he thinks there is a problem. He later sends you this message:

```
Hey, check out this cute cat picture.  http://tinyurl.com/Cute161Kitty
```

You click on this link and later find out that you have created a post shared with "`voting-demographic`" with the text "I build the best aircraft carriers this country has ever seen, SAD". (TinyURL is a URL redirection service. Whoever creates the link can choose whatever URL it redirects to.)

How was this post created? What URL would cause this to happen? Write the link in your solution. *Hint: in URLs, spaces are encoded as* `%20`*.*

TinyURL can redirect me to the following URL.
https://www.presidentialtweets.gov/share?text=I%20build%20the%20best%20aircraft%20carriers%20this%20country%20has%20ever%20seen,SAD&audience=voting-demographic

(c) Continuing from part (b), what attack is this and how could you defend your form from the sort of attack listed in part (b)? Explain in 1–2 sentences.

This is a CSRF attack. I can use refer validation or csrf-token to defend it.

Problem 3 ***Attempted Web Security*** **(10 points)**

(a) When users of `bank.com` are logged in, a request to `bank.com/session.js` returns a Javascript file containing

```
let sessionID = "0123456789";
```

except that 0123456789 is replaced with the session ID for the user who made the request.

An attacker controls `evil.com` and would like to learn Alice's session ID for `bank.com`. How can the attacker do this? Explain why the same-origin policy doesn't stop this attack. (Assume the attacker can get Alice to visit `evil.com` .)

When Alice visit evil.com, the attacker can initiate a CSRF attack by placing a <img src="bank.com/session.js"> tag. Then use the script in evil.com to send sessionID to the attacker. Same-origin policy doesn't work because the seesionID being sent is set in the origin of evil.com, which doesn't violate the rule.

(b) When `bank.com` learns of this problem, they fix it by beginning all Javascript files with

```
if (!document.location.includes("http://bank.com")) {
    while (1) {} // infinite loop

}
```

Explain why this doesn't work. How could an attacker defeat this defense?

The location validation is not strong enough. The attacker can register an domain like this bank.com.evil.com and initiate the same attack as before.

Problem 4 ***SQL Injection*** **(15 points)**

You are discouraged to find the following Java code in the client login section of an online banking website:

```
/**
* Check whether a username and password combination is valid.
*/
ResultSet checkPassword(Connection conn, String username, String password)
        throws SQLException {
    String query = "SELECT userID FROM Customers WHERE username = '"
        + username + "' AND password = SHA256('" + password + "');";
    Statement s = conn.createStatement();
    return s.executeQuery(query);

}
```

Assume that before issuing a request, the bank's server calls `checkPassword` and ensures that the returned `ResultSet` contains exactly one `userID`. If this check fails, the bank fails the request. Otherwise the request is issued as the user represented by `userID`.

Note: if there are 0 user IDs in the ResultSet then the username and/or password are wrong. If there are more than one then something went wrong somewhere on the bank's end since usernames should be unique (and consequently limit results to at most one). For the purposes of this question, what's important is that the request goes through iff the ResultSet contains exactly one user IDs.

(a) What username could an attacker enter in order to delete the Customers table?

‘; DROP table Customers;--

(b) What username could an attacker enter in order to issue a request as user "Admin", without having to know the password?

Admin';--

(c) When you point this out to the development team, a junior developer suggests simply escaping all the single quotes with a backslash. For example, the following line could be added to the top of the function:

```
username = username.replaceAll("'","\\\\'");
```

This code replaces each ' in the username with \' before including it in the SQL query.

Modify your answer to part (b) above so it will work against this new code. Assume the database engine accepts either ' or " to enclose strings.

Admin“;--

Problem 5 ***True/False Miscellaneous*** **(6 points)**

Answer each question. You don't need to justify or explain your answer. Check the box if it is True and leave it blank if it is False.

(a) [ ] A virus is malware that propagates by copying itself into target systems, and a worm is malware that propagates by infecting other programs.

(b) [X] Rootkits are often used to conceal other malware.

(c) [X] Bob wants to prevent people from overwhelming his website, so he decides to implement proof-of-work. Let d be the number of days since January 1, 1900. The client must send $r, H(r||d)$ where r is some nonce chosen by the client. The hash must begin with 13 zero bits. If the nonce has been reused in the same day or if the hash does not begin with 13 zero bits, Bob's server ignores the request. This is a good proof-of-work scheme.

(d) [X] A Distributed Denial of Service (DDoS) attack is executed by a botnet overwhelming the victim with large amounts of traffic coming from many sources.

(e) [ ] Tor defends against adversaries who can view all network traffic.

(f) [ ] A malicious middle relay (non-exit node) can read and modify your unencrypted traffic.

Problem 6 ***Course Survey*** **(5 points)**

Fill out the course survey available at this link: https://forms.gle/tVSsa68kXbpR32aD9.

In order to receive credit for this question, you will need to finish the survey. However, your responses are anonymous and will not affect your grade in the class.