Raluca Ada Popa
Spring 2018

CS 161 Computer Security — Homework 4

Due: Sunday, April 29th, at 11:59pm

Instructions. This homework is due Sunday, April 29th, at 11:59pm. It *must* be submitted electronically via Gradescope (and not in person, in the drop box, by email, or any other method). This assignment must be done on your own.

Please put your answer to each problem on its own page, in the order that the problems appear.

Problem 1 ***Web Security Warm-Up*** **(20 points)**

(a) In class we learned about the Same-Origin Policy and the cookie policies for the browser and the server. Nicholas Dirks says that you should be careful of visiting any untrusted site, because the site's owners can read cookies from *any* other websites they want. Is Nick right? Explain in 1–2 sentences why or why not. For this question, only focus on cookies.

(b) 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]?

(c) 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-demographics" 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?[1] What URL would cause this to happen? Write the link in your solution.

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

[1] A reminder: in URLs, spaces are encoded as `%20`.

Problem 2 *XSS* (15 points)

(a) After the sound security beating Snapitterbook took in project 3, Lord Dirks has given up on traditional security means. In an attempt to remedy all of Snapitterbook's XSS vulnerabilities, he proposes adding a new HTML tag called `<NOXSS>`. In between `<NOXSS>` and `</NOXSS>`, JavaScript and plugin content (such as Flash or Java) is disabled: browsers will ignore (and in particular, not execute) any such content that appears between a `<NOXSS>` tag and its matching `</NOXSS>` tag. Within both tags, a random value is placed, and this random value must match in the opening and closing tags. So `<NOXSS1bc625a9>` can only be closed by `</NOXSS1bc625a9>`

For instance, consider the following vulnerable code:

```
w.write("Hello, " + name + "! Welcome back.\n");
```

Because `name` comes from user input, the above code has a XSS vulnerability. Lord Dirks proposes that the above code should be rewritten as:

```
Random rng = new Random();
int random_int = rng.nextInt();
w.write("Hello, <NOXSS" + random_int + ">" + name
    + "</NOXSS" + random_int + ">! Welcome back.\n");
```

Why is this random token necessary? If the random token wasn't used, how could an XSS attack execute arbitrary javascript code if their only access point was in between a `NOXSS` tag?

(b) Snapitterbook's mysterious co-founder, Zark Muckerberg, proposes an optimization: instead of generating random numbers for each untrusted content block, the whole page can share the same random number. (This still forces different re-loadings of the same page to have different random numbers.) Zark is particularly concerned with potential security scandals; does this new optimization affect the security of the scheme, and why or why not?
Hint 1: Consider the case where there are two or more nested NOXSS tags.
Hint 2: How does the browser pair NOXSS opening tags and closing tags together?

Problem 3 *Access Control Policies* (20 points)

An organization has been serving users' websites from

`https://users.wile-e-sites.com/[USERNAME]/`

Anyone can sign up with a chosen username and build their own websites and applications with arbitrary HTML and JavaScript. Currently, they're discussing changing their service to serve the websites instead from

`https://[USERNAME].users.wile-e-sites.com/`

(a) Under the **current** system, a malicious site may be able to **read** some cookies from another user's site. Describe a cookie that a site could set such that another user's site would **not** be able to read it.

(b) Under the **proposed** system, could a site still prevent other users' sites from reading its cookies? If so, how?

(c) Under the **current** system, a malicious site can **write** a cookie that will be sent to other users' sites. Describe a cookie that a malicious site could set such that it would be sent to other users' sites as well.

(d) Under the **proposed** system, could a malicious site still write cookies for other users' sites? If so, how?

(e) One user, who runs a password-protected blog, claims that, under the **current** system, readers who visit other users' sites while signed in to the blog can cause the secret content to be disclosed to an attacker.

They report that malicious site open a window with the blog, access the blog's DOM with JavaScript, and send the results to its own server. State why browsers allow this to happen.

(f) Under the **proposed** system, could a malicious site still steal secret content this way?

Problem 4 *SQL Injection* (25 points)

You join as a software developer at *Bittr*, a fancy new Web 2.0 startup that aims to gastronomically empower communities by enabling disintermediated folksonomies of vegetables.[2] Freshly equipped with your CS 161 knowledge, you are horrified to find the following pseudocode snippet:

```
getProfile(username, connection):
    query = "SELECT profile FROM Users WHERE username ='" +
        username + "' ;"
    statement = connection.createStatement()
    return statement.executeQuery(query)
```

(a) You quickly send a email pointing out the issue to your boss, Arthur, telling him that the code is vulnerable to SQL injection. Arthur doesn't agree with your assessment. You decide that the only thing to do is to demonstrate a working SQL injection attack. Craft a value for `username` that will result in the table `Users` being dropped (deleted).

(b) On seeing your demonstration, Arthur apologizes and agrees that the issue is serious. Since he did not take CS 161, he decides to fix the code by backslash-escaping all single quotes in `username`. (Assume their SQL engine uses backslashes to escape single quotes.)

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

This replaces all instances of `'` with `\'`. The code is still vulnerable to SQL injection. Make appropriate modifications to your answer above, so that the table `Users` will still be deleted. Explain why you made the modifications.

(c) Finally, Arthur sees the light and requests that help him solve this issue. Armed with your CS 161 knowledge, you realize that parameterized queries are the way to go here. Explain what parameterized queries are, and why they would solve this problem.

(d) To get Arthur started with prepared statements, you decide to demonstrate the concept with an example. Rewrite the `getProfile` function using parameterized queries in an understandable pseudocode.

[2] No, the Bittr folks don't quite know what that buzz-phrase means, either. But they're pretty sure that people want it!

Problem 5 *Payment Fraud* (20 points)

You are the developer for a new fancy payments startup, and you have been tasked with developing the web-based payment interface. You have set up a simple payment page, with two inputs: the amount to be paid and the recipient of the payment. When a user clicks a "Pay now" button, the following request is made to complete the transfer:

```
GET https://www.cashbo.com/payment?amount=[dollars]&recipient=[name]
```

The server then transfers the specified amount from the logged in user (determined by a session cookie) to the specified recipient.

(a) You show this to your friend Eve, and she thinks there is a problem. She later sends you this message:

```
Hey, check out this funny cat picture.  tinyurl.com/z2k52gs
```

You click on this link, and later find out that you have paid Eve $1. (TinyURL is a url redirection service, and whoever creates the link can choose whatever url it redirects to.)

What kind of attack did Eve use to steal $1 from you? What did the tinyurl redirect to? Write the link in your solution.

(b) Web browsers send a `Referer` [sic] HTTP header when navigating (such as submitting a form), which tells the server the URL from which the user came from.

Describe a way to use this feature to prevent Eve from stealing any more money from you in this way.

(c) A defense using the Referer header is promptly implemented.

Another developer adds a new feature: the payments page now shows a list of the user's friends, with custom avatars. A user sets up an avatar by specifying a URL, which will be put in an `<img>` tag next to their name. (This fancy startup only hires the best, so there are no XSS vulnerabilities with the implementation.)

You later find out that all of Eve's friends have paid her $1. Describe how Eve used this new feature to bypass the defense and steal money from her friends.

(d) You decide to add a defense that uses secret tokens to stop Eve's latest attack: the payments page includes a secret value in a hidden field that must be sent with the payment request. That way, fixed URLs from unscrupulous users won't result in a fraudulent payment.

Around the same time, the startup rolls out new "Pay me" buttons that can be embedded on other sites. These are implemented as `<iframe>`s which load a small payment page from

```
https://www.cashbo.com/payme/[recipient]/[dollars]
```

This small payment page only has a button; the recipient and dollar amount are taken from the `<iframe>`'s URL.

As you tell Eve about these recent updates, she proudly announces her own new website, which displays two random cat pictures and which invites the visitor to click on which one is cuter. You check it out—just for a few minutes, you swear. But an hour later you realize you're still clicking away when you receive a message from your bank saying your account is overdrawn.

You find that you've sent Eve all your money in $1 increments. What kind of attack did Eve use to steal all your money? Describe her attack.

Problem 6 ***True/False*** **(20 points)**

Answer true or false and explain your reasoning.

(a) True or False: Two Javascript scripts embedded in pages running in two different tabs on a user's browser can never access the resources of each other.

(b) True or False: The httpOnly flag of a cookie mitigates XSS attacks because it ensures the browser sends the cookie only over https.

(c) True or False: A website that rejects all user input that contains `<script>` tags may still be vulnerable to XSS attacks.

(d) True or False: The same-origin policy does not prevent `https://abc.com` from loading an image from `http://xyz.com` onto its page.

(e) True or False: Javascript running on the outer frame of a website can always access the resources of an inner frame in the page.

(f) True or False: Browsers have a private browsing mode, which prevents websites from storing cookies on your computer altogether.

(g) True or False: Because of the cookie policy, you cannot be tracked across domains by cookies.

(h) True or False: A virus is malware that propagates by copying itself into target systems, and a worm is malware that propagates by infecting other programs.

(i) True or False: An SMS two-factor-authentication is an effective defense against phishing attacks.

(j) 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. True or False: This is a good proof-of-work scheme.

(k) True or False: A Distributed Denial of Service (DDoS) attack is executed by a botnet overwhelming the victim with large amounts of traffic coming from many sources.