# Homework 2: Tangled in the Web

*Due: Tuesday, March 15 @ 11:59 pm ET*

Do not include any identifying information (name, login, Banner ID, etc.) on your handin. When you're done, please submit a PDF file to Gradescope where your answer to each problem (not each individual question) is on a separate page. Make sure to assign each problem on Gradescope to the correct pages in your writeup (or you may receive a deduction).

Unless otherwise stated, you should justify and explain all of your proposed attacks, schemes, protocols, defenses, etc., referencing specific properties when necessary. Precise, effective communication and presentation of your ideas and solutions in your writeups is something we highly value when grading. Similarly, be concise, but make sure that you don't elide intermediate details, take subtleties for granted, or make any assumptions without justification.

While we encourage everyone to discuss and work through the problems on the homeworks, please remember that your homework solutions must be written up *entirely independently*, in your own words. You may not share your solutions with anyone (or read solutions written by others). You should not write your solutions while working with other students, and when you're writing your solutions, you should ensure that you independently understand and can reproduce your answers without referring to notes from collaboration sessions and consulting with other students.

**Reminder:** At the top of every problem (not question), you should write a *collaboration statement* that states who you discussed the problem with, who contributed major intellectual insights to your understanding of the problem, and any external resources you referenced while solving the problem. See the *Collaboration Policy* for examples. If you did not collaborate with anyone and did not refer to any sources, your collaboration statement should state as much.

## Homework 2 Instructions

Answer Problems 1–4 and do the TryHackMe lab. If you're a CS162 student, you must also answer Problems 5 and 6.

When you're ready to submit, you should hand in your answers to Problems 1–4 to the "CS166: Problems 1–5" drop on Gradescope. CS162 students should also hand in their answers to Problems 5–6 to the "CS162: Problems 5–6" drop on Gradescope in a separate PDF. (Answers submitted by CS166-only students to the CS162 questions will not be graded.)

Questions start on Page 2.

# TryHackMe Lab: Burp Suite

For this homework go to `https://tryhackme.com/jr/cs1660burpra` and complete the Burp Suite room we created.

As a reminder, these TryHackMe rooms are graded based on *completion*, not correctness. As long as you have answered all of the questions, you will get full credit. This assignment should not take more than 45 minutes so if you are stuck or are dealing with technical issues, make sure to post a question on Edstem.

## Problem 1: Crewmate Computer Cookies

> *Victor says: "Heads up! You might find it helpful to do some additional research (in addition to what was covered in lecture) to complete this problem."*

As part of their training, the crewmates at the CREWMATE ACADEMY are required to take a course called `cs666`: "Secure Computer Systems". The TAs of `cs666` have implemented an assignment submission system on the course website at `http://crewmate.academy/courses/cs666/submissions`.

Crew members taking `cs666` provide an email and password to log in to the submission site. The website then issues authenticated users a cookie which contains a session ID that allows the user to authenticate themselves without having to type in their password again for twenty-four hours:

```
Name:     sessionid
Value:    [random session ID...]
Domain:   crewmate.academy
SameSite: Strict
```

Authenticated users can then upload their assignments (or download assignment handins that they've previously uploaded). There's two other things worth mentioning about the ACADEMY's website—first, each crewmate-in-training has full control over their personal web page on the `crewmate.academy` domain (i.e. `http://crewmate.academy/staff/jbernar3/`); second, the ACADEMY's website allows for access via both `http` and `https` (similar to the `cs.brown.edu` website).

For **all parts** (or, for parts (c) and (d), subparts), limit each of your answers to **100 words maximum**—longer responses will receive no credit.

**Question a)** Consider Jake, a staff member taking the `cs666` course, and Kaki, an adversary who eavesdrops on Jake's network connection while Jake visits the site at `http://crewmate.academy/courses/cs666/submissions`. Can Kaki acquire Jake's `sessionid` cookie for the submission system? If so, outline an attack to steal the cookie; if not, explain why.

**Question b)** Consider Ben S., who controls the page at `http://crewmate.academy/staff/bsilverm/`; that is, he controls the content of the page but he does not have control over or access to the entire `crewmate.academy` server. Can Ben S. acquire Jake's `sessionid` cookie for the submission system? If so, outline an attack to steal the cookie; if not, explain why.

**Question c)** The ACADEMY administrators now consider various modifications to the cookie scheme.

(i) Suppose the ACADEMY administrators add the `Secure:TRUE` property to the cookie. First, identify what Jake has to change in the way he visits the submission site. Then, answer: does this change any of your answers to part (a) or part (b)? Explain.

(ii) Suppose the ACADEMY adds the `HttpOnly:TRUE` property to the cookie *instead of the `Secure` property*. Does this change any of your answers to part (a) or part (b)? Explain.

**Question d)** For this part, only consider Ben S., the adversary from part (b), and ignore the modifications made to the cookie scheme in part (c).

(i) Suppose the ACADEMY administrators change the URL of the submission system to `cs666.crewmate.academy/handin` and change the `Domain` property of the cookie to `cs666.crewmate.academy`. Does this change your answer to part (b)? Explain.

(ii) Suppose the ACADEMY decides to keep the domain of the system as `crewmate.academy` (and keeps the `Domain:crewmate.academy` property). Instead, they decide to put each staff page on its own subdomain—for instance, Ben S.'s staff page is now located at `bsilverm.crewmate.academy`. Does this change your answer to part (b)? Explain.

(iii) Suppose the ACADEMY changes the staff page URLs described in sub-part (ii), but, in the process, they accidentally remove the `Domain` property; that is, the `Domain` property isn't specified in the cookie. Does this change your answer to sub-part (ii)? Explain.

## Problem 2: CSRF-ing Around Tokens

*Jefferson says: "Heads up! Start working on this problem early—it's a little tricky. Also, you might need to do some additional research (in addition to what was covered in lecture) to complete part (c)."*

Recall the *Cross-Site Request Forgery* (CSRF) attack discussed in lecture. One possible mitigation for CSRF is a family of techniques known as CSRF tokens.

For **parts (a) and (b)**, limit each of your answers to **100 words maximum**—longer responses will receive no credit. Part (c) has no word limit.

**Question a)** One might implement such a technique via cookies. In addition to a session ID cookie, every user gets a cookie containing a random CSRF token when they log in. Then, whenever a sensitive request is made (i.e. via a `<form>`), before sending off the final request, JavaScript executing in the webpage reads this cookie, and copies it into the body of the request. The website then only accepts the request as valid if the CSRF token cookie matches the CSRF token in the body of the request.

Does this CSRF prevention technique work? If so, explain why and reference the web / browser security features that are relied upon in this technique; if not, provide an example of a CSRF attack that works while the user is visiting a malicious website.

**Question b)** The Crewmate Academy's assignment submission web interface at implements CSRF tokens in a different way from part (a). When the website (`https://crewmate.academy`) displays a sensitive `<form>` to the user, it generates a CSRF token, associates the value of this token with the user's session on the server, then loads the CSRF token into a hidden field on the form before sending it to the user. On form submit, the server checks that the submitted token from the form is the same as the token stored in the session.

Does this CSRF prevention technique work? If so, explain why and reference the web / browser security features that are relied upon in this technique; if not, provide an example of a CSRF attack that works while the user is visiting a malicious website.

**Question c)** Assume that the Academy's CSRF prevention technique works regardless of your answer to part (b); that is, assume that there is no way for an adversary to learn the CSRF token that must be submitted as part of the `<form>`, and that the website will only accept requests that contain a valid CSRF token.

It's clear what to do if a given CSRF token was valid—simply process the form action as intended. It's less clear what to do when the token is invalid, but here's what the Academy's website does: it takes the fields in the form submission (except the CSRF token) and uses them to populate a new form (which points to the same endpoint as the form submission that was just rejected) along with the correct CSRF token. This new form is then sent to the user as the response to the form submission that had the invalid CRSF token. In this way, if the CSRF token had expired, the user won't need to reenter all the information in the form—they can just check it for correctness and resubmit.

Why would this strategy for handling incorrect CSRF tokens be a bad idea? That is, provide a concrete attack against this scheme that allows a web attacker to submit arbitrary form contents with a valid CSRF token. You can assume that the Academy's website has no defenses against *Clickjacking / UI Redress* attacks and does not set security headers such as `Content-Security-Policy`.

(*Hint:* Consider the `target` attribute on HTML `<form>` tags.[1] This attribute is commonly used to designate that the response to a form submission should be opened in a new tab, but what else can you do with this attribute?)

[1] `https://www.w3schools.com/tags/att_form_target.asp`

## Problem 3: Vulnerability Disclosure

Please read the following *vulnerability equity processes*:

- U.S. Vulnerability Equities Process (VEP): `https://drive.google.com/file/d/1npUqw5ELRvV4m063fF3YA2eOrE_6x-Gs/view?usp=sharing`
- Summary of the Equities Process in the UK: `https://www.gchq.gov.uk/information/equities-process`

The following questions are open-ended and will be graded for thought and justification. Please reference and/or quote specific arguments from the readings in your answers.

**Question a)** What are the differences between responsible disclosure, full disclosure, and private disclosure? Describe one advantage and disadvantage of each. (about 5-7 sentences)

**Question b)** Suppose you are a security researcher who has discovered a vulnerability leaking financial data. Although you reported to the company 90 days ago (and you gave them a 90-day deadline), they have still not fixed the vulnerability and have been unresponsive. Knowing that this could lead to a large data breach, would you follow responsible disclosure guideline of publicizing the vulnerability? (about 4-5 sentences)

**Question c)** Companies' sensitivity towards having their vulnerabilities publicized has contributed towards adding conditions to their bug bounty terms limiting hackers' legal liability in exchange for non-disclosure. If you were a security researcher, would you accept these bug bounty terms? (about 4-5 sentences)

**Question d)** Do you prefer the UK Equities Process or the US Vulnerabilities Equities Process? Refer to what you like/dislike specifically about both processes when justifying your answer. (about 4-5 sentences)

## Problem 4: Redirecting To Crypto and Back

Many sites use *redirect scripts* that take a URL as a parameter and send the browser to that URL. Sites do this instead of simply linking directly to a particular site because it allows them to collect analytics on which external links users of their site are clicking on. That is, instead of linking directly to external websites, they provide links on the same domain where the external link is exposed via a "redirect parameter".

The course staff is developing a new startup, `http://pizza.brown`, a platform where students ask questions about course assignments. Staff members have the ability to link to external websites on `pizza.brown` (i.e., Victor's `pizza.brown` profile page links to `http://cs.brown.edu/vkalev/`). However, instead of linking directly to the external website, `pizza.brown` transforms the link into a link that uses the `/redirect` endpoint and displays the redirect-based link instead. For example, this link redirects users to the specified `url`:

`http://pizza.brown/redirect?url=http://cs.brown.edu/vkalev/`

However, this kind of "open redirect" poses a significant security risk to `pizza.brown` users. For example, consider an adversary that creates a phishing page at `evil.com`, then sends a user the following link:

`http://pizza.brown/redirect?url=http://evil.com/`

Since the beginning of this link appears to be a normal `pizza.brown` page, users who aren't looking closely at the link may think are clicking on a regular link to `pizza.brown`—however, the redirect script will cause them to be directed to the phishing page.

For **parts (a) and (b)**, limit each of your answers to **75 words maximum**—longer responses will receive no credit. Part (c) has no word limit.

**Question a)** One possible mitigation to the phishing attack is for the `pizza.brown` developers to maintain a *safelist* database of all allowed redirect targets. Under this system, if the `url` is not on the safelist, the redirect endpoint returns an error. Why might this be an unreasonable form of security? (*Hint:* Section Practice 3.2.)

**Question b)** Another possible mitigation to the phishing attack is for the `pizza.brown` developers to check that the domain name in the `Referer` header (or, even better, the `Origin` header[2]) is `pizza.brown`. Under this system, if the url is not `pizza.brown`, the redirect endpoint returns an error. Why might this be an unreasonable form of security?

**Question c)** Using cryptography, design a new redirect system that prevents the phishing attack and explain why it works. Then, explain why your protocol avoids the issues described in parts (a) and (b). Finally, describe why your protocol satisfies the following properties:

- P1. It must be possible for the website to "authorize" new redirect links when they are posted by staff members. As discussed in part (a), your protocol should not rely on maintaining a set of allowed links (e.g. in a database), but the website must be able to tell whether or not a given redirect URL has been authorized by the website.
- P2. Users cannot *arbitrarily* craft redirect links to non-authorized domains.
- P3. There must be some way for the website to "revoke" access to a particular redirect URL, though the permission revocation does not have to be instantaneous.
- P4. For user experience purposes, the stakeholders of `pizza.brown` have stated that they do *not* want an interstitial approach where the user is prompted to decide whether or not they want to visit the external link—the redirect script must be fully transparent to the end-user.

(*Hint:* You can assume a user who's visiting a profile page on `pizza.brown` won't view it for longer than 15 minutes. Also, be careful to make sure your scheme doesn't impose so

[2] The `Origin` header is exactly like the `Referer` header, but it only includes the website domain, not the entire URL.

much computational overhead that it leaves your website vulnerable to a denial-of-service attack—some of the Section material may be helpful here.)

## Problem 5: Better Passwords via a Browser Extension

*Only CS162 students are required to complete this problem.*

PwdHash (`https://crypto.stanford.edu/PwdHash/`) is a browser extension that transparently converts a user's password into domain-specific values. For example, if a user visits `bank.com` and types in the plaintext password `iampassword`, when the user submits the login form, PwdHash instead causes the browser to send hash(`iampassword` ++ `bank.com`) as the password, where ++ is the string concatenation operator and hash is some cryptographic hash function. (PwdHash knows the domain that the user is visiting on because, as a browser extension, it has direct access to the URLs a user visits.)

For **all parts**, limit each of your answers to **50 words maximum**—longer responses will receive no credit.

**Question a)** Sierra, a user with the PwdHash extension installed in her browser, likes using the same password "`balloons`" for every single website. Does PwdHash protect her password from being broken by a dictionary attack? Explain.

(*Hint:* For this part—and all other parts—think Kerckhoffs's principle.)

**Question b)** Sierra uses `bank.com` for her online banking operations. Suppose `bank.com`'s database is stolen. Does PwdHash protect Sierra's password from being cracked by a brute-force attack? Explain.

**Question c)** Suppose Sierra visits a fake website set up by Ocean, a web attacker that controls a website that looks identical to `bank.com` but is actually hosted at `bank.co`. However, Sierra doesn't notice the phishing scam and, through her browser, submits her real password for `bank.com` to the fake website. Does PwdHash protect Sierra's password from being stolen and used by Ocean? Explain.

**Question d)** How does PwdHash affect the overall entropy of its users' passwords? Explain.

## Problem 6: SOSQLI (or Unnecessarily Complicated Acronym)

*Only CS162 students are required to complete this problem.*

The CREWMATE ACADEMY's bookstore uses a SQL backend to manage its logic for shopping carts and order purchases.

Consider a user, Raj, who uses the ACADEMY bookstore. Whenever Raj adds an item to his cart (via making a POST request to the bookstore website that contains a cookie named `sessionToken` and a variable named `item`), the ACADEMY backend runs the following code, which stores each item as a separate record in the table named `carted_items`:

```
cart_add := fmt.Sprintf(
    "INSERT INTO carted_items (session, item) VALUES ('%s', '%s')",
    sessionToken, item)
db.Exec(cart_add)
```

You can assume that the ACADEMY's website will first check that a given `sessionToken` is valid before executing any queries with it.

**Question a)** The new influx of ACADEMY crewmates has increased domain for the popular `cs666` textbook, "Introduction to Secured Computers" by Toberto Ramassia). To deal with this increased demand, the ACADEMY's website blocks users from adding more than 1 copy of the textbook to their cart.

However, Raj is a huge fan of Ramassia's work and wants to buy multiple copies of the textbook that he can give out as gifts to his friends and relatives. Explain how Raj can perform an attack that allows him to add 10 copies of the textbook to his cart.

**Question b)** When a user visits the cart page, the ACADEMY populates the the cart webpage with links to the items. Given that most users only buy one book from the store at any given time, the ACADEMY administrators implemented an optimization: *if a user only has one item in their cart*, they improve performance by executing the following code instead:

```
cart_query := fmt.Sprintf(
    "SELECT item FROM carted_items WHERE session='%s' LIMIT 1",
    sessionToken)
item := db.Query(cart_query)

link_query = fmt.Sprintf(
    "SELECT link FROM store_items WHERE item='%s'",
    item)
db.Query(link_query)
```

(This is faster than what's normally done, which is to execute a SQL `JOIN` statement between the `carted_items` table and the `items` table.)

Unfortunately, the ACADEMY adminstrators caught wind of Raj's exploits from part (a) and fixed the vulnerability by changing the `cart_add` statement into a *parameterized query.*

TRUE or FALSE: Even though the ACADEMY changed the query used for the `cart_add` statement, Raj is still able to add 10 copies of the textbook to his cart. Explain.