Homework 3: Caught in the Net

Due: Friday, April 14 @ 11:59 pm EDT

Overview and instructions

This homework has 5 problems:

- Problems 1–4 are required for all students
- Problems 5 is required for **CS1620/CS2660 students only**

Note on collaboration

You are welcome (and encouraged!) to collaborate with your peers, but the solutions you write down must be **your own work** (ie, written by you). You are responsible for independently understanding all work that you submit—after discussing a problem as a group, you should ensure that you are able to produce your own answers independently to ensure that you understand the problem. For more information, please see the course Collaboration Policy.

In your submission, we ask that you include a brief *collaboration statement* describing how you collaborated with others on each problem—see the next section for details.

How to submit

You will submit your work in PDF form on Gradesope. Your PDF should conform to the following requirements:

- **Do not** include any identifying information (name, CS username, Banner ID, etc.) in your PDF, since all homeworks are graded anonymously
- Each problem (where "problem" is one of the Problems 1–4 or 5) should start on a separate page. When you submit on Gradescope, you will be asked to mark which pages correspond to which problem
- At the start of each problem, write a brief *collaboration statement* that lists the names and CS usernames of anyone you collaborated with and what ideas you discussed together
- If you consulted any outside resources while answering any question, you should cite them with your answer

There are two separate Gradescope submissions for this assignment:

- All students should submit Problems 1–4 to the assignment labeled **"Homework 3: Problems 1–4"**
- CS1620/CS2660 students must also submit Problems 5 to the assignment labeled **"Homework 3: Problem 5"**. For this part, you can either make a separate PDF for it, or just have one PDF and then mark the pages for these problems. Submissions for Problem 5 from CS1660-only students will not be graded (ie, there is no extra credit).

Problem 1: Web security throwback: Cookies

Blue University runs a web server at `blue.university`, where it hosts different pages for students and courses. Here's what you know about it:

1. Each student and staff member can upload web pages to a directory on the webserver to host a personal website. For example, user `alice` can upload any files to the server and make them available at `http://blue.university/people/alice/`
2. The webserver is pretty old, and allows users to view pages using both HTTP and HTTPS, ie. at `http://blue.university` and `https://blue.university`, respectively. Data sent via HTTPS is *encrypted in transit*, whereas HTTP is a plaintext protocol.[1].

After the debacle with `cs666_handin`, the cs666 course staff have implemented an assignment submission system at `http://blue.university/courses/cs666/submissions`. Students 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 24 hours. You view the cookie in your browser and it looks like this:

```
Name:     sessionid
Value:    [random session ID...]
Domain:   blue.university
SameSite: Strict
```

Based on what you know about the Blue University website and the new cs666 submission system, consider the following questions. For each part, your answer should be around 100 words maximum.

Question a) Bob is a cs666 staff member in cs666. Eve manages to eavesdrop on Bob's network connection while Bob visits the site at `http://blue.university/courses/cs166/submissions`. Can Eve acquire Bob's `sessionid` cookie for the submission system? Explain why/why not.

Question b) Alice can upload webpages to `http://blue.university/users/alice/`; that is, Alice controls the content of the page but cannot change anything else on the `blue.university` webserver. Can Alice acquire Bob's `sessionid` cookie for the submission system? If so, outline an attack to steal the cookie; if not, explain why.

Question c) The Blue University administrators now consider various modifications to the cookie scheme.

(i) Suppose the administrators add the `Secure:TRUE` property to the cookie. First, identify what Bob 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 University 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.

Hint: For a concise description of these cookie attributes, we recommend this link: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies`. You can also find information in lectures 7–11.

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

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

(ii) Suppose the University decides to keep the domain of the system as `blue.university` (and keeps the `Domain:blue.university` property). Instead, they decide to put each users page on its own subdomain—for instance, Alice's user page is now located at `alice.blue.university`. Does this change your answer to part (b)? Explain.

[1] We'll learn more about HTTPS/TLS in a few lectures—for now, this is all we need to know about it.

Sample TA Solution

a) Eve can acquire the cookie. Bob is visiting the site using `http`, not `https`; as such, the network data to the website is sent unencrypted. A passive network attacker (like Eve) can then simply sniff the cookie data as it is transmitted, and can read its actual contents since the data is unencrypted.

The only thing left to do is see how the cookie data will actually be transmitted—this is easily done by virtue of the fact that Bob is viewing the site at the time Eve is sniffing the network. At some point during Bob's browsing, he will almost certainly make a request that causes the cookie to be transmitted.

b) Alice can acquire the cookie. The same-origin policy permits the same domain to access the same domain's cookies. As such, Alice can place some Javascript at `http://blue.university/staff/alice/` that simply reads from `document.cookie` and sends the cookie data to an endpoint she controls.

One last thing to consider is how Alice might get Bob to view her staff page. As discussed in Lecture 5, one of the attacker models in web application security is that we can assume that the web attacker can get a user to visit a particular website (perhaps either by a phishing attack, or social engineering, etc.). Alice and Bob are both staff members for Blue University, and therefore it's certainly within the realm of possibility that Alice could convince Bob to view his web page.

c) (i) Bob has to now visit the page via `https`, since the `Secure:TRUE` property means that the cookies are only transmitted via an `https` connection.

This changes the answer to part (a). Since Bob now must use `https`, Eve can no longer sniff the network connection for the cookie since it is sent encrypted. However, this *does not* change the answer to part (b). Alice can just make Bob visit her webpage via `https://blue.university/staff/alice/` (as opposed to the `http` version) and perform the same Javascript-based cookie extraction attack described in part (b).

(ii) This changes the answer to sub-part (b) but not subpart (a).

The naming of the `HttpOnly` property is a little misleading—while the `HttpOnly:TRUE` property suggests that the cookie must be sent via an `http` (as opposed to `https` connection), what it actually means is that the cookie *can only be accessed via an HTTP request* (as opposed to Javascript). This means that scripts (like the Javascript described in the attack for part (b)) cannot access the contents of the cookie via the `document.cookie` variable.

This implication changes Alice's abilities as an attacker. Alice is no longer able to extract Bob's session cookie, even if she can convince Bob to visit her webpage, since her attack relies on accessing the cookie via Javascript. However, this doesn't change Alice's abilities as an attacker, as long as Bob visits the submission site via `http`—even with `HttpOnly`, the cookie will still be sent in the clear and Alice will be able to sniff it from the network.

d) (i) The URL change to `cs166.blue.university` blocks Alice's attack. Due to the same-origin policy, cookies set for `cs166.blue.university` cannot be read by `blue.university` because these two domain names are not considered to be same-origin. This means that the Javascript `document.cookie` variable will not contain Bob's `sessionid` cookie if he visits Alice's staff webpage.

(ii) The `Domain` attribute specifies allowed hosts to receive the cookie. If `Domain` is *specified* (like it is in this updated scheme), then subdomains are always included in the cookie scope. As such, the subdomain change for the staff pages *do not block* Alice's attack. The same-origin policy allows for cookies set on `blue.university` to be read by `alice.blue.university`, and as such the Javascript `document.cookie` variable will contain Bob's `sessionid` cookie if she visits John's staff webpage.

Problem 2: SolarWinds

In lecture 15 ("OS Isolation and Malware"), we discussed the SolarWinds hack and its aftermath. Please take a look at the following which discuss liability in software systems.

- `https://cs.brown.edu/courses/cs166/files/docs/cyber_insurance.pdf`
- `https://queue.acm.org/detail.cfm?id=2030258`

The following questions are open-ended and will be graded based on the justification of your responses. Please reference and/or quote specific arguments from the readings (and/or lectures) in your answers. Your answers for each part should be a few sentences each.

Question a) During lecture, we have discussed the challenges surrounding user-delayed software updates. One solution proposed is forcing updates upon users in cases where an app, network device, or operating system company has discovered and patched a critical vulnerability (i.e., by giving users 24 hours to apply the update manually, before restarting the device and applying the updating automatically). Are you in favor of this solution, do you prefer the status quo, or would you propose a different solution? Justify your answer using concepts from lecture.

Question b) Imagine the U.S. Congress is considering implementing legislation that makes software companies liable for any and all damages resulting from vulnerabilities in their software *except* if the company followed a set of defined cybersecurity best practices leading up to the breach. What practices would you consider sufficient to exempt companies from this liability? Or should companies be liable regardless of their cybersecurity practices?

Question c) For the following questions, consider the CSO article about cyber insurance:

(i) How do the cascading effects of breaches interact with cyber insurance?

(ii) Do you believe that cyber insurance is a barrier to investment in cybersecurity? If so, what actions, if any, should be taken to mitigate this? If not, why not?

Question d) Consider the ACM Queue article above. What would you change, if anything, about the authors' proposal?

There are no recorded solutions for this problem. Answers will vary.

Problem 3: TryHackMe Lab: Wireshark 101

For this problem, visit `https://tryhackme.com/jr/cs166wiresharkyl` and complete the Wireshark 101 lab. This lab is designed to give you some practice with Wireshark.

As a reminder, these TryHackMe rooms are graded based on *completion*, not correctness. As long as you have answered all of the questions on TryHackMe, you will get full credit—you do not need to submit anything in the PDF you upload to Gradescope. This assignment should not take more than 1 hour, so if you are stuck or are dealing with technical issues, please post a question on Edstem.

Fun fact: It took two months, but we heard back from TryHackMe support and figured out what was wrong with the Burp Suite room! Hopefully, this lab will remain open throughout the assignment—if you run into issues, please post on Ed.

Problem 4: Local network eavesdropping

Note: We will have covered all of the material for this problem after the Networks II lecture on Tuesday, April 11.

Consider the network represented in Figure 1, a subnet whose addresses all take the form `192.168.1.*` (ie, the subnet 192.168.1.0/24.) Each router and host is labeled with its IP address and MAC address. In all parts of this problem, assume that all hosts (Host A, Host B, and Host C) and the router have entries for all other hosts on the subnet in their respective ARP tables, and thus no entity in Figure 1 is actively performing any ARP broadcasts.

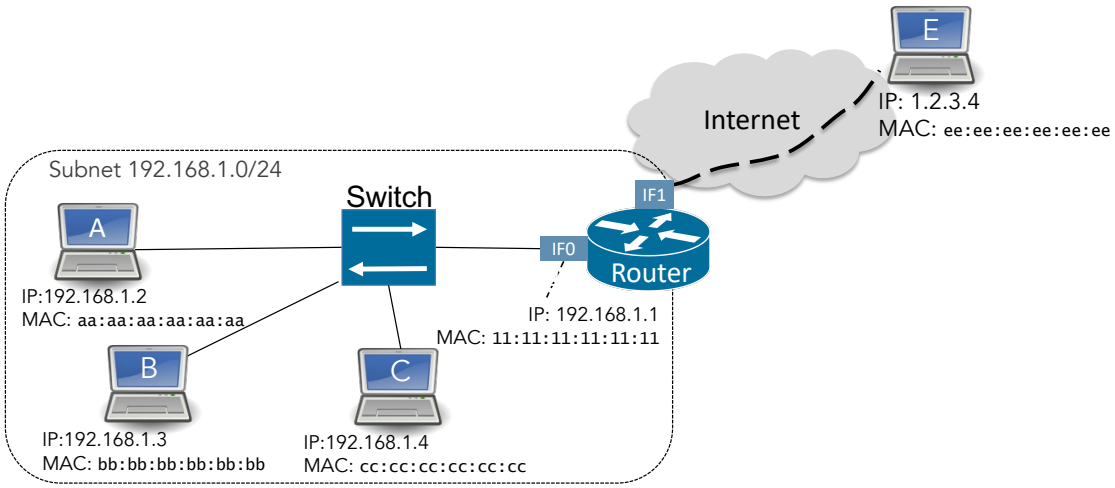

Figure 1: The network diagram for the "Network Switches" problem.

Question a) Host B wants to intercept traffic between Host A and Host C. What can B do to cause A to send their traffic to B instead of C? In a sentence or two, explain why your attack works.
Note: In carrying out this attack, B needs to be careful not to accidentally break other hosts' connections. How can B make sure that their attack only targets A? Make sure that you're specific about *where* any attack packets are being sent and the *content* of those packets.

Question b) Host B's attack is successful and it's now intercepting Host A's traffic, but this means that Host C isn't getting the traffic. As a result, A may soon notice that something is going wrong and stop sending data, which will limit the amount of information that B can intercept. How could B make sure that the communication between A and C is retained, while B can intercept it?

Question c) Assume that Host B's modified was successful and it is now intercepting Host A's traffic, and Host C is now receiving the traffic and responding properly—so A is not aware of the attack. However, B would also like to intercept **Host C's responses to Host A**. How can B accomplish this? Similar to part (a), explain your attack and why it works.

Question d) How would these techniques differ if Host B wanted to intercept Host A's communication with Host E, which is somewhere on the internet (ie, not on this subnet). Once again, we don't want to break the communication between Host B and Host E (ie, similar to part (b). In your response, try to be precise about the content of any attack packets you would send. **Hint:** Since Host E is not on B's subnet, it will not suffice to spoof E'S MAC address.

Sample TA Solution

a) Host B can repeatedly send ARP responses directly to A's MAC address (`22:22:22:22:22:22`), specifying that the MAC address corresponding to Host C's IP address (`192.168.1.4`) is Host B's MAC address (`33:33:33:33:33:33`). This way, only A's ARP cache is poisoned, while the ARP caches of other machines on the network remain unaffected. So when IP packets are sent from A's IP address to C's IP address, the resulting ethernet frames will be sent to B's MAC address.

   Note that gratuitous ARPs are usually sent to the broadcast address (`FF:FF:FF:FF:FF:FF`), but they don't have to be, as shown in the paragraph above. If sent to the broadcast address, all of the Hosts' ARP caches will associate B's MAC address with C's IP address, and so when any IP packets are sent to C's IP address, the resulting ethernet frames will be sent to B's MAC address.

   This sort of ARP response—one that is not in response to a request—is known as a "gratuitous ARP," and is a valid use of ARP (that is, the protocol specifies that hosts should update their ARP caches in response to gratuitous ARP replies). This technique is known as "ARP cache poisoning."

b) Host B can forward all of the IP packets that it receives from A (whose destination addresses specify that they were supposed to go to C) to Host C. In particular, this means that any IP packet that arrives in an ethernet frame whose source address is A's MAC address, and where the destination IP address of the IP packet is C's IP address, should be forwarded to C. B can do this by creating a new ethernet frame with the same IP packet and sending it to C's MAC address.

c) Host B can perform the same attack in reverse—poison C's ARP cache to believe that the MAC address corresponding to A's IP address is actually B's MAC address, and then forward any intercepted IP packets to A.

d) Since `1.2.3.4` is not in the local subnet, Host A (as described in problem 2) will need to learn the MAC address of the default gateway (`192.168.1.1`), and the IP packets will be encapsulated in ethernet frames and sent to that MAC address. Thus, B will need to instead poison A's ARP cache to associate the default gateway IP address with B's MAC address, and then forward any intercepted traffic to the gateway. In order to get the return traffic, B will need to poison the router's ARP cache to associate A's IP address with B's MAC address so that response IP packets (whose destination IP address will be A's) will also be sent to B.

   (*Aside:* If you're wondering how the machines on the LAN learn the address of the default gateway, see Section 7, in which we discuss the DHCP protocol.)

Problem 5: (CS1620/CS2660 only) Better Passwords via a Browser Extension

Only CS1620/CS2660 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.)

Consider the following questions. For each part, your answers should be around 50 words each.

Question a) Alice, a user with the PwdHash extension installed, likes using the same password "`balloons`" for every single website. Does PwdHash prevent Alice's password from being broken by a dictionary attack? Explain.

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

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

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

Sample TA Solution

a) In a dictionary attack, an attacker tries password possibilities that are likely to succeed (such as common words or short sequences of letters). PwdHash combines an easy password with a domain-specific string and then hashes it; this generates a pseudorandom string that is difficult to guess. If the attacker doesn't know about PwdHash, then this would be hard to guess. However, due to the open design principle, this is not a valid assumption to make. If the attacker knows that the user uses PwdHash, then a dictionary attack would be possible (since the hash function is public, the attacker knows the domain, and "`balloons`" is easy).

b) If the attacker does not know that Alice is using PwdHash, then the attacker will have to brute force long, random strings (assuming the output from PwdHash is long) which will be difficult. However, if the attacker does knows that Alice is using PwdHash, all they have to do is compute the brute force attack by running passwords through PwdHash before passing them to the database hash function, which will reduce the length of passwords needed to brute-force a small password like "`balloons`".

c) PwdHash is domain-specific, so it will generate a different hash of the password on the bank.co website; thus, `bank.co` will not get Alice's `bank.com` password.

d) The entropy of user's passwords either remains the same or decreases with PwdHash. This is because the entropy of the password is likely significantly lower than the entropy of the hash output space. If someone's password has more entropy than the hash output, it decreases entropy.