Homework 4: The Final Homework

Due: Friday, April 28 @ 11:59 pm ET

Overview and instructions

This homework has 4 problems:

- Problems 1–4 are required for all students
- There is no extra CS1620/CS2660 part for this assignment. Have fun with Dropbox!

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) 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 is only one Gradescope submission for this assignment. All students should submit Problems 1–4 to the assignment labeled **"Homework 4"** on Gradescope.

Problem 1: Thinking about DNS (10 pts)

Question a) In the DNS lecture, we discussed how each DNS query has a `request identifier`, also called a transaction ID, to identify responses to individual queries. Consider the following questions about how DNS transaction IDs are used (your answers should be no more than 100 words for each):

(i) Why is using a transaction ID more secure than using *no* transaction ID?

(ii) Why is using *randomized* transaction IDs is more secure than having *sequential* transaction IDs.

Question b) Imagine you control the default DNS server for an Internet Service Provider (ISP). In 2–3 sentences, briefly explain how you can attempt to leverage DNS to block access to sites you don't want your customers to access.

Sample TA Solution

a) See subparts, 3pts each: 3/3 if answer is complete, 2/3 if minor details missing, 1/3 for multiple issues, 0/3 for major conceptual problem or answer missing.

(i) An attacker needs to guess the query ID that corresponds to a given DNS request in order to trick the requester.

Long-form explanation (more info than required): DNS query IDs ensure that DNS responses correspond to a given DNS request. In particular, they ensure that a malicious attacker cannot forge a malicious DNS response to a given DNS request—since a DNS ID takes on one of 65,536 distinct values (a 16-bit number), an attacker would have to guess the query ID that corresponds to a given DNS request in order to trick the requester into accepting their malicious DNS response. Without query IDs, an attacker would not have to go through this process.

(*Source:* `https://www.sans.org/reading-room/whitepapers/dns/achilles-heal-dns-565`)

(ii) Sequential IDs allow attackers to guess the ID number for subsequent queries if they can determine ID of a past query; randomized IDs prevent this.

(*Source:* `https://www.sans.org/reading-room/whitepapers/dns/achilles-heal-dns-565`)

b) If you are an ISP, you can do the following:

- Run your own DNS server that gets advertised to customers that blocks domains you don't like (ie, respond with error, or redirect to wrong server)
- Intercept/modify DNS traffic sent to your customers (ie, DNS hijacking, or outright blocking of DNS traffic)

4pts, total, graded on the following scale (adjust if necessary):

- 4/4 pts: Full credit: mentions at least one of these
- 2/4 points: Half-credit: Correct intuition, but something important missing
- 0/4 points: No credit/missing

Problem 2: TLS vs. Blue University (15 pts)

Blue University worked out a really good deal with a the CA AwesomeTrust to get a TLS certificate for its main website, `blue.university`. However, based on what you know about Blue University and the reputation of their business partners, you know something is going to go very wrong here...

Consider the following scenarios, which examine what happens when different parts of a public key infrastructure are affected if a certain key is compromised. For each part, write a brief explanation (1-2 sentences) of what capabilities you (as an attacker) can do in each scenario. Specifically, for each question, consider:

- Who could you impersonate?
- Who would believe you? (ie, For what set of users would you be able to impersonate X? All users in the world, or just a specific group?)

Note: For considering who you can impersonate, you don't need to worry about *how* you'd actually perform the attack at the network layer—instead, just **focus on how the TLS authentication process would be affected if you control the key in question**.

Question a) You compromise a University webserver and obtain the private key for `blue.university`. If you have this private key, what capabilities do you have? Give your answer by considering the two bullet points above and briefly explain your reasoning. You can impersonate `blue.university` for anyone who connects to the website. (2pts, +2 for identifying who to impersonate, +2 for identifying who would believe you; half credit if intuition is close but something significant is missing)

Question b) It turns out that the CA AwesomeTrust is pretty shady (big surprise...) and you're able to obtain AwesomeTrust's CA private key. If you have AwesomeTrust's CA private key, what capabilities do you have? Give your answer by considering the two bullet points above and briefly explain your reasoning. You can now sign your own certificates, so you can now impersonate any website for any user (at least until AwesomeTrust gets revoked or removed from browsers).

(4pts, +2 for identifying who to impersonate, +2 for identifying who would believe you; half credit if intuition is close but something significant is missing. Not required to state that AwesomeTrust would get revoked eventually.)

Question c) After you compromised AwesomeTrust, Blue University gives up on the idea of delegating trust to a third-party: it decides to *make its own CA* and issue its own certificate for `blue.university`. To make this work, the University's IT policy requires all users to install the Blue University CA certificate on their systems.

(i) Why do Blue University users need to install the CA certificate? What happens if a browser **doesn't** have the CA certificate installed and connects to `blue.university` anyway? If the CA certificate isn't installed, browsers will flag it as not trusted because they can't verify `blue.university`'s public key. If the CA isn't installed, this will display a warning when the user tries to connect.

(3pts, +2 for recognizing that certificate is not trusted, +1 for noting that browser will display warning; half-credit as applicable)

(ii) If you're able to obtain the Blue University CA private key, what capabilities do you have? Give your answer by considering the two bullet points above and briefly explain your reasoning. You can now sign your own certificates and thus impersonate any website, and any user who has the CA certificate installed (ie, all Blue University students/employees) will believe you.

(4pts, +2 for identifying who to impersonate, +2 for identifying who would believe you; half credit if intuition is close but something significant is missing.)

Problem 3: TryHackMe Lab: Nmap (10 pts)

Go to `https://tryhackme.com/jr/nmap02uv` to complete a lab using `nmap`, a port scanning tool. This lab will expand on what we learned in lecture about IPs and ports and show you how to scan for open ports and services using `nmap`.

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.

Problem 4: Onion-Flavored Handins (15 pts)

Note: We will have covered all of the material for this problem after the lecture on Anonymous Networking on Tuesday, April 25.

The CS666 course at Blue university has its own Tor network to allow students to submit their code "super-anonymously" to a custom testing server. Unlike Gradescope, the testing server doesn't require students log in and instead just sends back a test report so students can check their work against the autograder. This way, only students' final, best submissions need to be uploaded to Gradescope and recorded with their names.[1]

Figure 1 depicts the CS666 Tor network and a specific circuit established between you and the testing server. Lines depict the specific Tor circuit established by your client using the nodes highlighted in blue. In this network, each Tor node i has a public key PK_i, which is known to all other Tor nodes and the client[2].

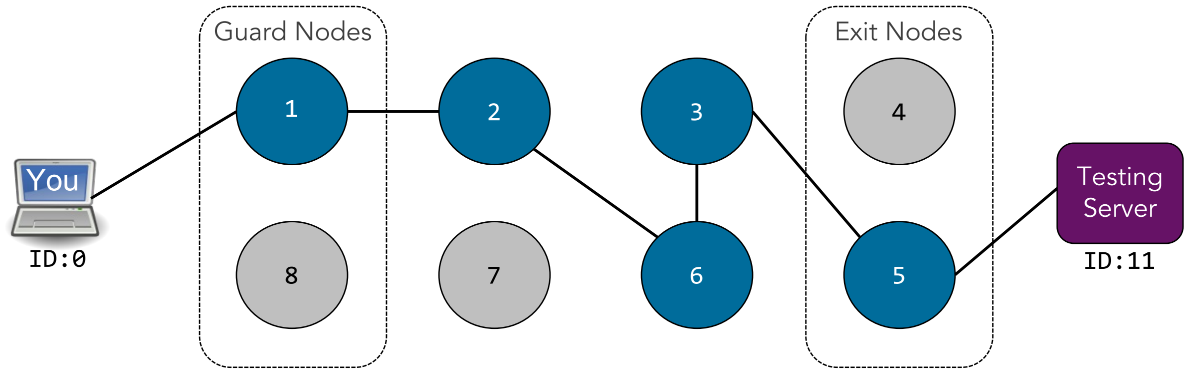

Figure 1: The CS666 Tor network. Note that the lines indicate connections in the Tor circuit, not individual network links.

Question a) To send a message M to the testing server using the circuit in Figure 1, what is the packet your Tor browser sends out?
You can express your answer in the format: `[dest-id, data]`
where `dest-id` is the ID of a node as labeled in Figure 1 and `data` is the packet content. For example, `[1, "Hello"]` sends the string "Hello" (in plaintext) to node 1. You can denote encryption of message M as $E(PK_i, M)$, where PK_i is the public key for a node with ID i. You can assume that each key is known to all Tor nodes and the client (this is established during circuit setup).

Question b) **True or False**: even if you connect to the testing server using plain HTTP (**not** HTTPS), no one can read M except the testing server—that is, TLS isn't required in order to have *confidentiality* in the presence of an eavesdropper in the network because the traffic is encrypted. **Explain your reasoning.**

(Continued on the next page)

[1] I promise you that we don't actually care about your non-final submissions and don't look unless we think there's an autograder bug. It makes a nice case study for Tor, though. ☺

[2] This is a simplification of the actual Tor protocol, which uses a combination of asymmetric and symmetric keys to set up circuits.

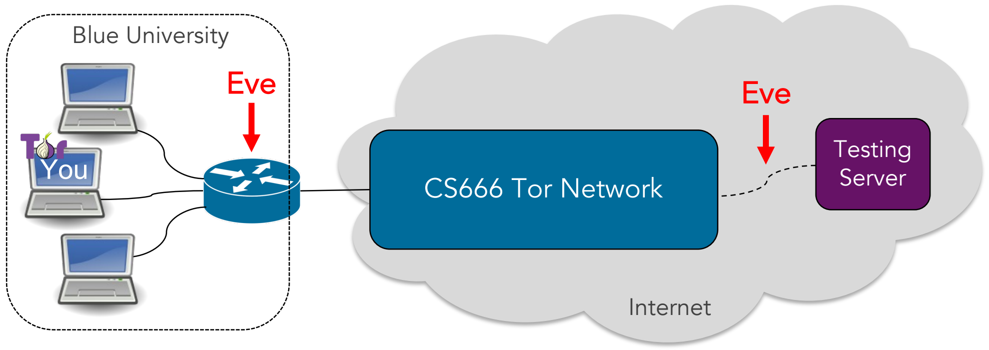

Figure 2: Eve's eavesdropping capabilities.

Question c) Suppose that Eve gains access to a bunch of Blue University systems and figures out how to view network traffic at the two points shown in Figure 2. Specifically, Eve can observe individual users' traffic leaving the Blue University network *and* traffic entering the testing server. Even if you use HTTPS, what, if anything, could Eve learn about your submissions to the testing server?
Explain your reasoning in 1–2 sentences—you don't need to provide specific details of an attack, just sketch what sort of information Eve could learn, and what is still hidden.

Sample TA Solution

a) The packet is:

$$[1, E(PK_1, [2, E(PK_2, [6, E(PK_6, [3, E(PK_3, [5, E(PK_5, [11, M])])])])])]$$

Note that the final message from the exit node to the testing server is not encrypted.

5pts total. +2 points for correct ordering/onion structure (except last hop), +3 points for recognizing that the last hop is not encrypted. Partial credit as applicable.

b) False. Based on our observation from part (a), we can see that confidentiality of M is not maintained as M is sent in plaintext (since HTTP is used, not HTTPS) between node 5 and the Gradescope server. Thus, a network eavesdropper between those two machines can read the contents of M.

5 pts total. For full credit, answer correctly reasons that HTTPS is required because data on the last hop is not encrypted using Tor. If answer for part (a) stated that the last hop was encrypted, full credit if answer states that HTTP is **not** required because last hop is encrypted (ie, no double jeopardy if last part is wrong). Partial credit as applicable if intuition is correct.

c) Eve knows that someone is using the super-anonymous service, and thus someone is using Tor. In general, Eve can perform a timing attack—by observing when student send Tor traffic and observing when traffic hits the Grading server, Eve might be able to correlate these, especially if only one user is using Tor at one period of time.

5pts total, graded on the following scale:

- 5/5: Full credit: answer mentions something about a timing attack, ie. correlating traffic seen at the testing server with tor traffic exiting the university network
- 4/5: Correct intuition, but some detail is missing
- 3/5: Correct intuition, but multiple things missing
- 2/5: Incorrect intuition or major things missing

- 1/5: Does not relate to an attack
- 0/5: Missing/incomplete

Answers do not need to be particularly detailed, so long as they mention something about using the timing of messages to gain information about use especially if there is only one, or a small number of users on the network.

More info: For a small number of users of the Tor network, Eve can then use her control of the router to see that our browser is the only one sending packets to Tor entry nodes (which are necessarily a public list of IPs in order for Tor browsers to find them—see the Tor directory servers discussed in the lecture slides). This ties with the issue of "we only have anonymity in a crowd" that we discussed in the *Applications I* lecture—if we're the only user using Tor, it doesn't stop Eve from just recognizing that all Tor connections are coming from a single IP address and thus she necessarily knows that we are the only IP address who could have possibly sent a given super-anonymous handin.

(*Aside:* This ties with this incident from 2013 where a Harvard student sent a bomb threat using Tor, but was able to be tracked because he sent it on the University network and thus the logs showed that his computer was sending data to Tor entry nodes (and no one else's computer was): `https://www.theverge.com/2013/12/18/5224130/fbi-agents-tracked-harvard-bomb-threats-across-tor`)