CSCI 476: Computer Security
Spring 2023
Final Lab

Due Wednesday May 10th @11:59 PM

Print your name: _____

Print your Net ID: _____

Instructions:

- You may print this out, fill it out by hand, scan it when finished, and submit to D2L **as a pdf**
- You may fill this out digitally and save and submit to D2L **as a pdf**
- You may put your answers in a separate word document, and save and submit to D2L **as a pdf**

Logistics

- This is an individual assignment. You are not allowed to work with other students. You are welcome to post questions in the Discord, or DM/email Reese
- You cannot use a late pass on this assignment, nor can you submit it past the deadline. Due to grading turnarounds, I will not be able to accept late work.
- You may use any resources that you'd like (slides, lecture videos, labs, etc). You shouldn't *need* to use google for anything, but you are welcome to use external resources as well
- Make sure you answer each question. Some questions have multiple parts

1. What operating system mechanism must be used by software writers to access computing resources, such as memory or disk? Please briefly describe how this mechanism work.

2. What is a **Set-UID** program? Why do we need them?

3. While studying Set-UID programs, we looked at two functions used to invoke and execute external programs: **`system()`** and the **`exec()`** family of functions. Why is **`system()`** considered unsafe, while **`execve()`** is considered to be safe?

4. We studied environment variables as an attack vector for exploiting Set-UID programs. Describe *one* of these approaches that uses environment variables to exploit a Set-UID program (Hint: we looked at two methods: one used PATH and the other one used LD_PRELOAD)

5. What is the Shellshock attack? What vulnerability is this attack exploiting?

6. What is a reverse shell? Why did we need a reverse shell during the shellshock attack rather than a normal shell?

7. What is a **buffer overflow attack**? What is the goal of a buffer overflow attack?

8. What is the stack? What data gets put onto the stack during execution?

9. What is **shellcode**? Why did we inject shellcode rather than our own malicious program that we compiled?

10. What is address space layout randomization (**ASLR**)? Why does ASLR make buffer overflow more difficult? How did we bypass ASLR?

11. Suppose that Santa Claus uses an SQL database to keep track of all the nice and naughty children. Santa uses SQL queries in order to figure out which children he needs to deliver to (children that are "naughty" do not receive any presents). Consider the following SQL table:

Table Name: `Santa_List`

Name	Naughty_or_Nice	Age	Wishlist
Mary	Nice	8	Xbox
Eve	Naughty	12	Dolls
John	Nice	4	Train Toys
Steve	Nice	7	Bike

Each child has a name, whether they are naughty or nice, their age, and what toy(s) they would like for Christmas.

Santa can run a SQL query to find the names of the children he needs to deliver presents to. If a child is "Nice" they will get presents this year. If they are "naughty" they do not get any presents:

```
SELECT Name FROM Santa_List WHERE Naughty_or_Nice = 'Nice';
```

Name
Mary
John
Steve

Santa also has a program that children can use to update their wishlist. The child enters their name, and the toy they would like to put on their wishlist. The program updates the SQL table using the UPDATE keyword:

```
Enter your name: Mary
What do you want for christmas? Nintendo Switch
```

```
UPDATE Santa_List SET Wishlist='Nintendo Switch' WHERE Name='Mary';
```

Eve discovers that this program does not use `prepare()` statements and has no separation of code and data. Eve is currently "naughty" in the database, but she wants to perform an SQL injection attack to update her information so that she appears on nice list (so that she can get presents this year).

Knowing that there is an SQL injection vulnerability, and how the backend SQL query is operating, **your goal is to determine the correct inputs for the vulnerable program so that the SQL Query updates the `Naughty_or_Nice` field to be "Nice" for Eve.**

```
UPDATE Santa_List SET Wishlist='          ' WHERE Name='          ';
```

Provide the input below to perform the SQL injection:

```
Enter your name:
What do you want for christmas?
```

12. What is a Cross-Site-Scripting (XSS) attack?

13. During the Cross-Site-Scripting attack lecture, we found a method for stealing another person's browser cookies. How did we get their cookies sent back to us (the attacker) ?

14. What is the TCP/SYN flooding attack?

15. In order to prevent SYN flooding attacks, a security engineer proposes the following solution:

    "We will collect the source IP addresses of all SYN packets, and after 5 retries we will assume that IP address was used for the SYN flooding attack and mark that IP address as "bogus". For all bogus IP addresses we find, we will add a rule to the firewall that will block all incoming traffic from that IP address."

    Will this solution help prevent SYN flooding attacks? Why or why not?

16. In the spoofed packet for the TCP Reset and TCP Hijack attacks, why was the sequence number required for the packet?

17. What is the goal of a DNS cache poisoning attack?

18. ECB was one mode of encryption we looked at during symmetric key cryptography. Why is ECB unsecure? How do other modes of encryption (CBC, CFB, CTR) fix this?

19. What are the three important properties for a cryptographic hash function?

20. How can hashing be used for message integrity?

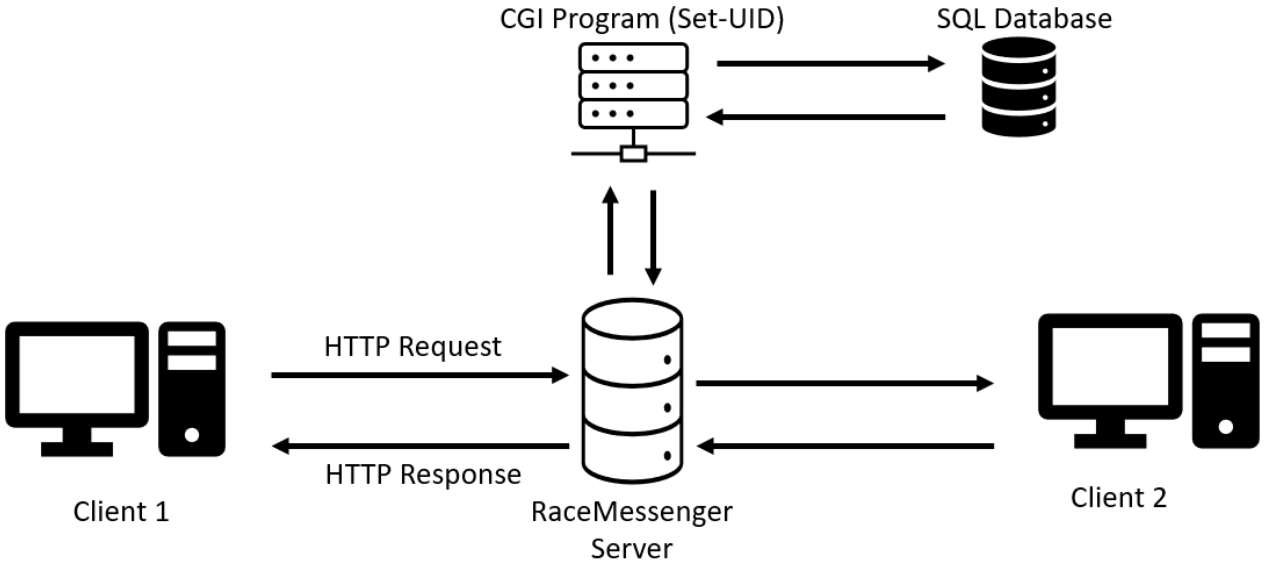
21. From a high level, please describe why RSA / Diffie-Hellman Key Exchange is considered to be secure. (i.e Why is it difficult for a malicious third party to crack the secret key, but not for the two hosts?

22. In RSA, both a public key and private key are used in the cryptosystem. What is the difference between them, and when is each key used?

23. As a future computer scientist, why is it important to learn about security?

24. Consider the following software system.

RaceMessenger is brand new messaging software being released soon. This is a client-server application where clients can send messages to each other via the internet (think of it as something like Discord or Facebook messenger). Clients do not directly interact with each other. Instead, they communicate with a middleman server that is run by RaceMessenger. If the server needs to access user information (for validating log in, or pulling/modifying user information) the server will contact a CGI Program that will handle the request, and do the appropriate operation. This CGI program is a bash shell that is running as a Set-UID program. The CGI Program will contact an SQL database, and pull any needed information from the database. Once the Server gets the information it needs, it will send the information back to Client 1 (if the user is updating/checking their own information) or send the information to Client 2 (if they are sending a message to client 2). All communication is done with HTTP requests/responses, which uses TCP as the transport layer protocol. The client-side messaging software for RaceMessenger is written in the super safe language of C. Users can log in with a username and password, and have the ability to update their own user information.



You've been hired as a security consultant for RaceMessenger, and your task is to develop a **threat model** for this system. You can use whatever threat modeling technique you'd like:

- Brainstorming/bullet points (like we did in class)
- Mind Map
- Attack Tree
- STRIDE

You threat model should at least answer the following questions:

1. What are the assets?
2. What can go wrong? What are the threats? Think about all the attacks/security principles that we've talked about
3. What mechanisms can we implement to prevent things from going wrong? How can we recover if things go wrong? Think about all the countermeasures/security mechanisms we've talked about