We'll post an outline of each recitation after the fact so you know what was covered in case you had to miss class that day. The recitation outlines are not a full replacement for recitation!

This week's recitation was largely a problem-solving session (good for exam prep). These are the problems we went over; some sections may not have gotten to every problem.

Meltdown

Question 1: Sadhana is experimenting with the toy example in Listing 1 of the Meltdown paper. She uses a system that has 4096-byte pages, just like in the paper. However, in her experiences, she uses the number 1024 where the example uses 4096. Sadhana's code is below. Each element in probe_array is one byte.

```
1: raise_exception();
2: // the line below is never reached
3: access(probe_array[data * 1024])
```

Sadhana runs the example on her system, and observes that the access time for Page 63 is around 100 milliseconds; for every other page, the access time is around 400 milliseconds. Assume that pages are numbered starting at 0. Given Sadhana's observations, what is the value of data? If there is more than one possible value of data, list them all.

Question 2: Jay is also experimenting with the toy example in Listing 1. He is using exactly the code from the paper:

```
1: raise_exception();
2: // the line below is never reached
3: access(probe_array[data * 4096])
```

However, unlike the paper, Jay is using a system that has 2048-byte pages. Jay runs the example on his system and observes that the access time for Page 64 is around 200 milliseconds; for every other page, the access time is around 400 milliseconds. Assume that pages are numbered starting at 0.

Given Jay's observations, what is the value of data? If there is more than one possible value of data, list them all.

Stack Smashing

Consider the code below.

```
void win() { printf("code flow successfully changed\n"); }
```

```
int main(int argc, char **argv) {
      char x;
      char buffer[32];
      char y;
      gets(buffer);
      int i;
      int j;
}
```

Sam runs this code on a machine that has no protection against stack-sashing. On his machine, integers are four bytes long and characters are one byte long. Any pointers — e.g., the base and instruction pointers, function pointers — are also four bytes long. When a function is called, the following happens (in this order):

1. Any arguments to the function are pushed onto the stack
2. The base pointer (BP) is pushed onto the stack
3. The instruction pointer (IP) is pushed onto the stack
4. Local variables to the function are pushed onto the stack

The list above describes **everything** that happens on the stack. There aren't, e.g., any other pointers pushed onto the stack between BP and IP. Sam's goal is to overwrite the saved instruction pointer and force the function win() to be called, by inputting a string into this program.

- Draw the stack
- How long should Sam's string be, in bytes?
- Where should the address of win appear in the string?
- (Bonus?) Draw the stack *after* Sam has input the string.

Authentication

Amelia is building a system that authenticates users via usernames and passwords. She knows that the correct way to do this is to store the following in a table on the server, where salt is a randomly-generated salt (each user gets their own salt) and H is a "slow" hash function:

username, salt, H(password | salt)

In this system, Amelia would authenticate users by taking their inputted password, concatenating it with the stored salt, and checking whether the hash of that string matched what is stored in the table.

Amelia wants to mix things up, and tries implementing her system by storing different information in the table. Below, we propose four schemes. Each scheme stores **only** the three given pieces of information about each user.

For each of the proposed schemes, decide whether Amelia could use this information to properly authenticate users without opening her system up to additional attacks from adversaries with read-access to her system. Explain why or why not. You can assume for all parts of this question that any salts are generated properly (e..g, they are random in the way that we need them to be) and that H is a proper slow hash function (cryptographically secure, etc.).

Scheme 1: username, salt, H(password) | salt
Scheme 2: username, salt, H(password | H(salt))
Scheme 3: username, H(salt), H(password | salt)
Scheme 4: username, H(salt), H(password | H(salt))

Secure Channels

Alice and Bob are communicating via what they hope is a secure channel. They've exchanged a symmetric key, *k*, which they use to encrypt, decrypt, and MAC (in the real world we'd use a different key for the MAC than encryption/decryption). You can assume that only Alice and Bob know k.

To send a message m to Bob, Alice does the following:

1. Computes c = encrypt(k, m)
2. Computes h = MAC(k, c)
3. Sends c | h | seq to Bob, where seq is a sequence number. Alice's first message to Bob has sequence number 1, and she increments the sequence number for each message (i.e., her second message has sequence number 2, the message after that has sequence number 3, etc.). You can assume that the sequence number space is infinite (i.e., sequence numbers will not "wrap").

Question 1: This approach isn't quite secure yet, but let's ignore that for now. What does Bob need to do when he receives a message from Alice?

Now assume Bob is doing that. Communication between Alice and Bob is always one-way: Alice sends data to Bob, but Bob never sends data to Alice. There are **no** network failures between them, no packets are ever dropped, and the route between Alice and Bob never changes.

Consider an on-path attacker, Eve. Eve happens to know that Bob runs a bank, and each of the messages that Alice is sending to him initiate a bank transfer that takes money out of Alice's account. Even would like to drain Alice's bank account. She is **not** interested in launching DDoS attacks on Alice or Bob, stealing their passwords, etc.

Question 2: As an on-path attacker, what could Eve do to achieve her goal?

Question 3: What could Alice and/or Bob do to prevent Eve from launching this attack? Assume that Alice still wants to be able to communicate with Bob using the network (i.e., “stop sending messages” is not a correct answer)

Question 4: Suppose Alice and Bob now want to allow for communication in both directions. What would they need to do to keep things secure?