The exam will be held in class on 10/24. It will cover the material from Lectures 1-10 (see lecture notes here: https://user.eng.umd.edu/~danadach/Security_Fall_22/lectures.html).

1. **What’s the difference?**
   For each pair of items/concepts below, explain in two sentences or less how they are different.

   (a) stack vs. heap

   (b) dangling pointer vs. memory leak

   (c) encryption vs. message authentication code

   (d) reflected XSS vs. persistent (stored) XSS

   (e) terminator canaries vs. random canaries
2. Memory safety

Consider each code snippet below and decide whether there is a memory safety vulnerability. Justify your answer by providing a **one-sentence** explanation.

(a)```c
int add (int n) {
    int *array = (int *) malloc(n*sizeof(int));
    if(array == NULL) { return ERROR; }
    for(int i=0; i<n; i++) {
        array[i] = n;
    }
    free (array);
}
```

(b)```c
int print (int n) {
    int *array = (int *) malloc(n);
    if(array == NULL) { return ERROR; }
    for(int i=0; i<n; i++) {
        array[i] = i;
    }
    for(i=0; i<n; i++) {
        printf(''%i
'', i);
    }
    free(array);
}
```

(c)```c
char* copy10 (char *inStr) {
    int max = 10;
    char * outStr = malloc(max);
    if(outStr == NULL) { return ERROR; }

    int i = 0;
    while(i <= max) {
        outStr[i] = inStr[i];
        i++;
    }
    return outStr;
}
```
3. Stack overflows

(a) The following code is compiled for 32-bit x86:

```c
void badbuf(int *input) {
    int array[2];
    for(int i=0; i<30; i++) {
        array[i] = input[i];
    }
}
```

This code contains a buffer overflow vulnerability.

Using assembly code, the `exit` system call can be invoked by placing 0x01 into %eax and placing the error code value into %ebx, then invoking the `int 0x80` instruction. Using the following assembly code snippets (with given addresses), construct an input to `badbuf` that uses ROP to cause the program to exit with error code 5.

<table>
<thead>
<tr>
<th>Address</th>
<th>Assembly Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x801011aa</td>
<td>pop %eax; ret</td>
</tr>
<tr>
<td>0x80105bbb</td>
<td>pop %ebx; ret</td>
</tr>
<tr>
<td>0x801027cc</td>
<td>pop %ecx; ret</td>
</tr>
<tr>
<td>0x801049dd</td>
<td>pop %edx; ret</td>
</tr>
<tr>
<td>0x8010c580</td>
<td>int 0x80</td>
</tr>
</tbody>
</table>

Use the diagram below to show what you would include in your input (one 32-bit word at a time). In the left column, show the value; in the right column, give a short explanation for why this value. There may be more space in the diagram than you need – leave anything unused blank. If a value is unknowable or doesn’t matter, you can mark it as ?. Note that input reads from the top to the bottom of the page, as though you were reading memory in gdb.
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>input + 0</td>
<td>...</td>
</tr>
<tr>
<td>input + 4</td>
<td></td>
</tr>
<tr>
<td>input + 8</td>
<td></td>
</tr>
<tr>
<td>input + 12</td>
<td></td>
</tr>
<tr>
<td>input + 16</td>
<td></td>
</tr>
<tr>
<td>input + 20</td>
<td></td>
</tr>
<tr>
<td>input + 24</td>
<td></td>
</tr>
<tr>
<td>input + 28</td>
<td></td>
</tr>
<tr>
<td>input + 32</td>
<td></td>
</tr>
<tr>
<td>input + 36</td>
<td></td>
</tr>
<tr>
<td>input + 40</td>
<td></td>
</tr>
<tr>
<td>input + 44</td>
<td></td>
</tr>
<tr>
<td>input + 48</td>
<td></td>
</tr>
<tr>
<td>input + 52</td>
<td></td>
</tr>
<tr>
<td>input + 56</td>
<td></td>
</tr>
</tbody>
</table>

(b) Would making the stack non-executable prevent the attack you just set up? Why or why not? Answer in 3 sentences or less.
4. **Web security**

The CS department is holding an online poll to determine what to name the cafe in the new Iribe Center. Every student is allowed to vote once, from among the following choices:

1. KFC++
2. Dining Philosophers
3. Snack Overflow
4. Big Bowl Notation
5. TGI Fortran
6. Standard Query Lunch

You have very strong feelings about this choice, and you suspect you may be able to influence the outcome more than just by voting.

The voting website includes a form with three fields: a username, a password, and a field to select your choice. On the backend, the website executes this query:

```sql
UPDATE Votes SET Preference='$choice'
WHERE User='$user' AND Password='$password';
```

to record a vote. You can vote as many times as you like, but your choice is always overwritten (so only one vote counts). `$choice` is formatted as an integer from 1-6.

Here are some other syntax hints that might be useful:

- ```sql
  SELECT [ColumnName1, ColumnName2] FROM [TableName]
  WHERE [ColumnName] = '[Selector]'; -- This is a comment
  ```
- ```sql
  UPDATE [TableName] SET [ColumnName] = '[Value]'  
  WHERE [ColumnName1] = '[Selector1]' AND [ColumnName2]='Selector2';
  ```

(a) Briefly explain how you could vote fraudulently to ensure a unanimous vote for your favorite choice. Provide the syntax of any command(s) you might use.

(b) The department catches on to your exploit and updates the backend processing to escape single quotes: replacing each input instance of ' with \ (i.e., placing a single backslash in front of it). Can you still rig the vote? If yes, provide the syntax of any command(s) you might use.
(c) The department once again catches on to your exploit and redesigns the voting server to prevent any similar exploit. However, all hope is not lost. The voting submission form site also includes a comments section, in which you can attempt to persuade your fellow students to agree with your choice. After experimenting with the comments section, you realize that you can post a comment to rig the vote in your favor. Explain how (briefly). What is the name of the attack you are using (be specific)? Why does it work?

(d) In one sentence, describe a countermeasure the department could take to prevent the exploit you described in part (c).
5. Symmetric Key Cryptography

(a) Let \((\text{Gen}, \text{Enc}, \text{Dec})\) be a CPA-secure symmetric key encryption scheme. Consider the following modification to the encryption algorithm: On input message \(m\), the encryption algorithm outputs \((c, H(m))\), where \(c \leftarrow \text{Enc}(k, m)\) is computed by running the old encryption algorithm and \(H(m)\) is the output of \(H\) on \(m\), where \(H\) is a cryptographic hash function that maps messages of any length to hashes of length 256. Is the modified encryption scheme still CPA-secure? Justify your answer.

(b) Assume an attacker sees a CBC-Encryption of a message \(m = m_1 || m_2 || m_3\) under a randomly generated key \(k\), denoted by ciphertext \(c = IV || c_1 || c_2 || c_3\). Show that the attacker can use \(c\) to construct a new ciphertext \(c' = IV' || c'_1 || c'_2 || c'_3\) that decrypts to \(m' = m'_1 || m'_2 || m'_3\) under \(k\), where \(m'_2\) is all 0's (denoted \(0^n\)), and \(m'_1, m'_3\) are arbitrary. You answer should describe the new ciphertext \(c' = IV' || c'_1 || c'_2 || c'_3\) in terms of \(c\) and \(m\). Can the attacker get both \(m'_1\) and \(m'_2\) to decrypt to \(0^n\)? What about \(m'_1\) and \(m'_3\)? Why or why not?

(c) Let \((\text{Gen}, \text{Mac}, \text{Vrfy})\) be a secure MAC. Consider the following modification to the MAC tag generation algorithm: On input message \(m\), the encryption algorithm outputs \((t, H(m))\), where \(t \leftarrow \text{Mac}(k, m)\) is computed by running the old Mac algorithm and \(H(m)\) is the output of \(H\) on \(m\), where \(H\) is a cryptographic hash function that maps messages of any length to hashes of length 256. Is the modified MAC still secure? Justify your answer.
6. Incorrect usage of cryptography

Alice is a customer with a bank account at Bank Bob (just “Bob” for short). She goes in person to the bank and generates a single key $k_{AB} \leftarrow \text{Gen()}$ to be used for authentication and encryption. Both Alice and Bob store the key securely. Bob also stores Alice’s unique username (“alice”) together with $k_{AB}$.

When Alice wants to log in to Bank Bob under her username, she computes $t \leftarrow \text{Mac}_{k_{AB}}(alice)$ and sends the message $(alice, t)$ to Bob. Bob verifies the Mac and lets Alice know if the verification succeeded. If yes, then Alice can subsequently send encrypted messages (under $k_{AB}$) to the Bank to specify a transaction she would like to perform. If the transaction is invalid, Bob notifies Alice that it is invalid. Otherwise, Bob notifies Alice that the transaction succeeded. The messages returned from Bob to Alice are unencrypted.

List three security problems stemming from incorrect usage of cryptography in the above method of implementing an online banking application.