# Raluca Ada Popa
# Spring 2018

# CS 161
# Computer Security

# Midterm 1

PRINT your name: _____, _____
(last) (first)

*I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct.*

SIGN your name: _____

PRINT your class account login: `cs161-`_____ and SID: _____

Name of the person sitting to your left: _____

Name of the person sitting to your right: _____

You may consult one sheet of paper of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

If you think a question is ambiguous, please come up to the front of the exam room to the staff. If we agree that the question is ambiguous we will add clarifying assumptions to the central document projected in the exam rooms. Write your student ID on the top of every page.

You have 80 minutes. There are 8 questions, of varying credit (155 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

| Do not turn this page until your instructor tells you to do so. |
|---|

SID _____

**Problem 1** ***True or False*** **(20 points)**

Bubble in True or False for each of the questions below. You do not need to explain your answers.

(a) If W^X is enabled and an attacker induces a buffer overflow that overwrites a local variable with attacker code, if afterwards the original program writes to the local variable, an error will occur flagging this situation.

○ True ● False

**Solution:** False, W^X prevents execution of writable parts of memory.

(b) If ASLR is enabled and we leak the address of a local variable, then we know the address of all local variables since ASLR does not randomize their relative offsets.

● True ○ False

**Solution:** True, ASLR only randomizes the start of segments, not the relative offsets of items within the segment. (Some implementations randomize the heap, but it's arguable whether that can be called ASLR. The question asks about local variables, which are on the stack.)

(c) If we have a stack canary and the attacker does not know its value, it is impossible (except with small probability) for an attacker to overwrite the return address of a function and cause the execution of other code.

○ True ● False

**Solution:**

False. It is possible to have a "write-around" vulnerability, where an index-out-of-bounds lets an attacker overwrite only specific parts of memory. As an example, the following code would overwrite the return address with `0xdeadbeef`.

```
1  void broken(void) {
2      int x[1];
3      x[-3] = 0xdeadbeef;
4  }
```

(Note that the question does not mention buffer overflows.)

(d) A larger TCB is more secure because it means that more components of the system must be broken in order to compromise the system.

○ True ● False

**Solution:** False. The TCB is the components of the system that we must trust, and the larger the TCB the worse it is.

(e) In the Diffie-Hellman protocol, one way to calculate $g^x \bmod p$ in polynomial-time is to multiply $g$ by itself $x - 1$ times, reducing modulo $p$ at every iteration.

○ True ● False

**Solution:**

False. This is exponential time. (We determine the speed of cryptographic algorithms by the time they take in terms of the bit length.)

(f) It is OK if the attacker knows the IV which will be used for AES-CTR in advance.

● True ○ False

**Solution:**

Yes, it is OK. The intuition is that $E_k(IV)$ is completely random to the attacker, even if they know what $IV$ will be. They do not have $k$, so they cannot find $E_k(IV)$. Therefore they cannot win the IND-CPA game.

(g) If one bit of a plaintext block is flipped in AES-CTR mode, that changes roughly half of the bits of that ciphertext block.

○ True ● False

**Solution:** False, it only flips the corresponding bit of that ciphertext block. (So flipping the 3rd bit in the 5th plaintext block also flips the 3rd bit in the 5th ciphertext block, without changing anything else.)

(h) If one bit of a plaintext block is flipped in AES-CBC mode, that changes roughly half of the bits of that ciphertext block.

● True ○ False

**Solution:**

True. Since CBC mode encryption is $E_k(P_i \oplus IV)$, we see that changing a bit of $P_i$ will change the ciphertext block completely as $E_k$ is a random permutation.

(i) Say that Alice and Bob have two pre-shared keys $k$ and $k'$. Alice can guarantee the integrity and confidentiality of a message $M$ if she sends AES-CBC$_k(M)$ and the tag MAC$_{k'}(M)$.

○ True ● False

**Solution:**

A leaky MAC like the one seen in discussion will compromise the confidentiality of the message. Recall that MACs **only** provide integrity, and make no attempt at confidentiality of their input.

(j) Say we have two similar messages $M$ and $M'$. We encrypt both messages in CBC mode, but accidentally reuse the same IV. Then we encrypt both messages in CTR mode, but accidentally reuse the same IV (but different from the one we used for CBC mode). CBC mode will compromise lesser or equal amounts of information compared to CTR mode.

● True ○ False

**Solution:** CBC encryption will reveal information up to the first difference in the messages, whereas CTR will reveal information anywhere that both messages are identical (including up to the first difference).

SID _____

**Problem 2** ***Short Answer*** **(15 points)**

Give **concise** answers to the following questions. Do not provide multiple answers to a question; you will not receive credit even if your answers contain a correct answer.

(a) Give one way an attacker could exploit a buffer overflow without executing their own shellcode.

> **Solution:** Overwrite local variables. Return-oriented programming. Arc injection.

(b) On some Linux systems, the least significant byte of a stack canary is always NUL (`0x00`). However, this decreases the entropy of the canary making it easier for an attacker to guess the canary randomly. Identify one **advantage** of having this NUL byte in stopping buffer overflow exploits.

> **Solution:** Harder to leak with not NUL terminated string. Cannot overwrite with functions like `strcpy`.

(c) Alice and Bob are thinking of using asymmetric encryption with a **single** public & private key that they both share. Explain one advantage of symmetric encryption over this asymmetric encryption model.

> **Solution:** Symmetric encryption is faster. Symmetric encryption has smaller key sizes. Symmetric encryption can be chained more easily.

(d) What is the computationally difficult problem that RSA digital signatures rely on?

> **Solution:** Finding prime factors of large integers.

(e) In all the confidentiality cryptography games we have discussed, the adversary gets to send a pair of messages $(m_0, m_1)$ to the challenger. The challenger only reveals the encryption of one of these messages. Explain why $m_0$ and $m_1$ need to be the same length.

> **Solution:**
>
> Encryption methods (to some extent) preserve the length of the original message, and so sending two messages of wildly different lengths would let the adversary distinguish.

SID _____

**Problem 3** ***Student Linked List*** **(20 points)**

Lord Dirks writes the following code below to manage the students of Leland Junior University:

```
1  struct student_node {
2      char name[8];
3      struct student_node *next;
4  };
5
6  typedef struct student_node student_node;
7
8  void add_student(student_node *head, char *student_name) {
9      student_node *new_student = calloc(1, sizeof(student_node));
10     while (head->next) head = head->next;
11     head->next = new_student;
12     strcpy(head->name, student_name);
13 }
14
15 student_node first;
16
17 int main() {
18     char *name_to_add;
19     first.next = NULL;
20     while (has_input()) {
21         name_to_add = safely_read_input();
22         /* esp = 0xbfff'f09c */
23         add_student(&first, name_to_add);
24     }
25 }
```

(a) Identify the line which causes the vulnerability. What vulnerability is this?

**Solution:**

The line with `strcpy`. Buffer overflow.

(b) Raluca needs your help to PwN Lord Dirks. To help you, she added some shellcode at the memory address `0xdeadbeef`. What names would you need to enter into the program in order to cause the execution of the shellcode? Note that the value of `esp` at line 21 is `0xbffff09c`. Assume that the compiler does not reorder any local variables or pad stack frames. Furthermore assume that the call to `strcpy` is inlined.

**Solution:**

The intuition for this problem is by overwriting the pointer `head->next`, we can ensure that the **next** name writes onto the stack. Note that solutions which attempted to do a buffer overflow in a single line do not work, because `first` is in static memory, not on the stack. It is also not possible to write all the way from static memory to stack memory.

In order to write `head->next` points to the stack, we need to enter a name which is greater than eight characters.

Since we aim to overwrite the return address at `0xbffff090`, we can try this:

`AAAAAAAA\x90\xf0\xff\xbf`

This successfully makes it so that the pointer `head->next` points onto the stack, and now our program thinks that this is a `student_node` struct. When we insert our next string, the program will "stop" at this memory address. Since the first member of a struct has offset 0, we see that our next `name` will be inputted at `0xbffff090`. Therefore, we can overwrite the return address of `add_student` with our next name:

`\xef\xbe\xad\xde`

However this has a subtle problem: there is no guarantee our program will not keep iterating once it hits `0xbffff090`, because it is likely that `((student_node *) 0xbffff090)->next` is not actually NULL. This would cause our program to keep iterating `head = head->next` and likely crash. (There will be significant credit for solutions which successfully set up a pointer to the stack and attempt to overwrite the return address of any function, even if they ignore this problem.)

The full solution requires us to write significantly below the stack pointer, where all of the memory is untouched zero bytes. Then, we can write our way up to the return address. The following sample solution works:

`AAAAAAAA\x90\xe0\xff\xbf`

`A` (`0x1000` times) `\xef\xbe\xad\xde`

Note that with this solution, we **must** overwrite the return address of `add_student`, since the program will crash before exiting from `main`.

SID _____

**Problem 4** ***New Operating Systems*** **(20 points)**

Answer the following questions about security principles as presented in class as **concisely** as you can.

(a) Alice is writing a new operating system TuxOS. She notices that the following lines of code appear many times:

```
1 FILE *fp;
2 if (!access_ok(filename)) exit(1);
3 fp = open_file(filename, "r");
```

i. Alice is scared that eventually she will forget to add an `access_ok` check. Which security principle is most relevant to this situation?

> **Solution:** Ensure complete mediation.

ii. How would you fix the security issue above?

> **Solution:**
>
> Create a single function `open_file_safely` which checks `access_ok`, and then use that instead. Make `open_file` private.
>
> (Basically any solution which ensures that `access_ok` is always called before `open_file` works.)

iii. What other security issue is relevant to this code?

> **Solution:** TOCTTOU. An attacker can create an "okay" file, and then swap it for a "bad" file before it is actually opened.

(b) Bob is frustrated with TuxOS and decides to develop a new open source OS, Noobuntu. The main feature of Noobuntu is that the user needs to specify the permissions with which each program runs. This means that if a user wants a video chat application to work properly, they would grant the application access to: a temporary directory to store temporary files, the internet, the camera, the microphone, and the speakers. Bob argues that this will secure users against malware from the internet.

i. What security principle does Bob have in mind with this feature?

> **Solution:** Least privilege

ii. Alice asks Bob how Noobuntu handles programs running other programs. He didn't think about that and runs home to patch Noobuntu before anyone notices. He specifies that programs run other programs with the same set of permissions that were granted to the first program. Is this design secure? Why or why not?

> **Solution:** No, if a user runs a vulnerable program that requires a lot of permissions, an attacker could exploit the vulnerability to execute their own malicious program with all of the permissions of the original program.

iii. Bob, frustrated with all the people trying to attack his system, decides to rewrite his code in Penguin++, a new programming language he wrote up overnight. Bob, proud of his knowledge

of cryptography implements cryptographic primitives in his language to add extra security. He reasons that because nobody knows Penguin++ they won't be able to understand what he is doing, and definitely won't be able to attack his system. What are two reasons why Bob's new system may be insecure?

**Solution:** He used security through obscurity (violated Shannon's Maxim) and he wrote his own crypto.

SID _____

**Problem 5** ***Selection*** **(20 points)**

Evelyn is working on some code to sort the prefix of an array. Here is her code so far:

```
1  /** Sort the first n integers of the array a */
2  void sort(int a[], size_t n) {
3      for (size_t i = 0; i < n; i++) {
4          size_t idx = i;
5          for (size_t j = i; j < n; j++)
6              if (a[j] > a[idx])
7                  idx = j;
8
9          /* part (b) */
10
11         int tmp = a[idx];
12         a[idx] = a[i];
13         a[i] = tmp;
14     }
15 }
```

(a) What are the preconditions needed for memory safety of the above code?

**Solution:** `a != NULL AND n <= size(a)`

(b) What are the invariants which hold at the line denoted part (b)? Express your answer mathematically or in very precise English.

**Solution:** A full list of invariants follows:

1. `0 <= i < n`
2. `for all x, y such that 0 <= x <= y < i, a[x] >= a[y]` (The first `i` elements of `a` are sorted in descending order.)
3. `for all x, y such that 0 <= x < i and i <= y < n, a[x] >= a[y]`
4. `i <= idx < n`
5. `for all y such that i <= y < n, a[idx] >= a[y]` (`a[idx]` is the largest element among the remaining `n - i` elements.)
6. The first n elements of a is a permutation of the original first n elements of a.

In addition, all of the preconditions hold as invariants. It is not required to list all of the invariants in order to receive full credit on this part.

(c) What are the postconditions ensured by this function? Express your answer mathematically or in very precise English.

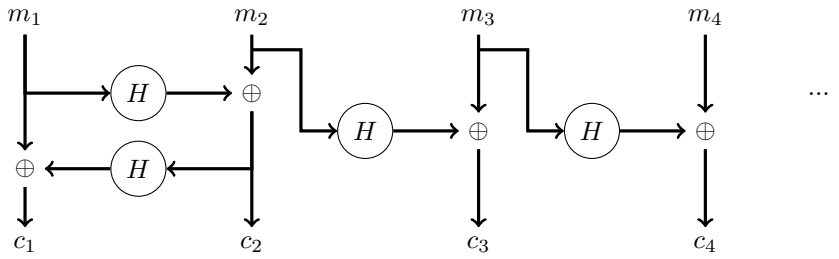
**Solution:**

1. `for all x, y such that 0 <= x <= y < n, a[x] >= a[y]` (The first `n` elements of the array are sorted in descending order.)
2. The first n elements of a is a permutation of the original first n elements of a.

SID _____

**Problem 6** ***Hashing a Scheme Out*** **(15 points)**

Alice doesn't trust block ciphers because she feels that rectangular shapes can't be trusted, so she has created her own encryption scheme using hashing. She's enlisted your help in analyzing the security of her scheme; just don't tell her that this is a "block" diagram.




(a) Now Alice needs to choose the hashing function $H$. Assume that key is a 128-bit key (generated randomly once) and IV is a 128-bit initialization vector (generated randomly for every encryption). Furthermore we define $a||b$ as the concatenation of $a$ and $b$. For example if $a =$ "Hello" and $b =$ "World", then $a||b =$ "HelloWorld".

If $H$ was defined as

$$H(x) = \text{SHA256}(\text{IV}||\text{SHA256}(x||\text{key}))$$

would this make scheme IND-CPA? Explain your reasoning.

○ Yes ● No

**Explain:**

**Solution:** The scheme cannot be secured with this hashing function. The key makes the output of the hash function unique to us, and the IV makes it so that encrypting the same message twice will generate different outputs. The problem is that the hashing function never "increases" in entropy as we work our way through the message; what this means is that if a single block is repeated in a message, then we can have repeating blocks in the ciphertext.

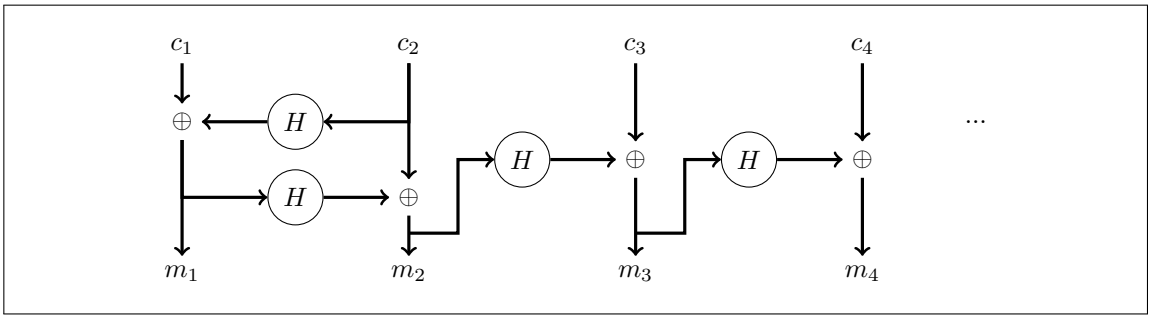
$$m_1 = m_2 = m_3 \implies c_2 = c_3$$
$$c_1 = H(m_1) \oplus m_2$$
$$c_2 = H(m_2) \oplus m_3$$

(b) What would decryption look like in this scheme? Listing decryption for $m_1$ and $m_i$ where $i > 1$ is sufficient.

**Solution:**

$$m_1 = c_1 \oplus H(c_2)$$
$$m_2 = c_2 \oplus H(m_1)$$
$$m_i = c_i \oplus H(m_{i-1})$$

SID _____



SID _____

**Problem 7** ***Protocols*** **(25 points)**

Alice wants to send a message to her friend Bob. Evaluate each of the cryptographic protocols below on confidentiality, integrity, and authenticity by marking all that apply. **Confidentiality** means that an eavesdropper cannot learn any information about the contents of the message. **Integrity** means that a man-in-the-middle cannot modify any messages sent without Bob eventually detecting it. **Authenticity** means that Bob can tell that the message came from Alice. (Assume that everyone knows Alice and Bob's public keys.)

Mark **Broken** if it is impossible for Bob to decrypt the message, or if it is impossible to perform the protocol. If a protocol is **Broken**, you do not need to determine its other properties.

(a) 1. Alice sends Bob a key $k$, encrypted with Bob's public key.

2. Alice sends Bob AES-CBC$_k(M)$.

■ Confidentiality ☐ Authenticity

☐ Integrity ☐ Broken

**Solution:** Mallory cannot learn the message because it is encrypted with the key $k$. She cannot learn $k$ because it is encrypted with Bob's public key. However, she could change the message simply by changing the ciphertext, so this provides neither integrity nor authenticity.

(b) 1. Alice sends Bob two keys $k$ and $k'$, both encrypted with Bob's public key.

2. Alice sends Bob AES-CBC$_k(M)$, MAC$_{k'}$(AES-CBC$_k(M)$).

■ Confidentiality ☐ Authenticity

☐ Integrity ☐ Broken

**Solution:** Mallory cannot learn the message because it is encrypted with the key $k$. She cannot learn $k$ because it is encrypted with Bob's public key. However, **anyone** can encrypt with Bob's public key. Mallory can encrypt her own keys $k$ and $k'$ and drop all of Alice's messages.

(c) 1. Alice sends Bob a key $k$, encrypted with Bob's public key.

2. Alice sends Bob a signature on $k$ using her RSA key.

3. Alice sends AES-CBC$_k(M)$ to Bob.

■ Confidentiality ☐ Authenticity

☐ Integrity ☐ Broken

**Solution:** Mallory cannot learn the message because it is encrypted with the key $k$. She cannot learn $k$ because it is encrypted with Bob's public key. She cannot change $k$, because it is signed with Alice's RSA private key. However, Mallory can change the encrypted message since it has no integrity checks at all, so this only provides confidentiality.

(d) 1. Alice sends Bob a key $k$, encrypted with Bob's public key.

2. Alice sends Bob a signature on $k$ using her RSA key.

3. Alice sends $M$, MAC$_k(M)$ to Bob.

☐ Confidentiality ■ Authenticity

■ Integrity ☐ Broken

**Solution:** Mallory cannot change the key $k$ because it is protected by the RSA signature. She cannot change the message because it is protected by the MAC. The message is sent in plaintext, so no confidentiality is provided.

(e)
1. Alice breaks her message up into small blocks $B_1, B_2, \ldots, B_n$. These blocks are small enough to be encrypted and signed asymmetrically (say 1024 bits).
2. Alice encrypts each block $B_i$ with Bob's public key to produce a ciphertext $C_i$. Then, Alice signs $C_i$ to produce a signature $s_i$.
3. Alice sends $(C_1, s_1), \ldots, (C_n, s_n)$ to Bob.

■ Confidentiality ☐ Authenticity

☐ Integrity ☐ Broken

**Solution:**

Only Bob can decrypt the message, since only he has the private key. Note that this **does** provide confidentiality, even in the case of repeated blocks of plaintext message, since public-key encryption schemes are non-deterministic. However, Mallory can drop or rearrange ciphertexts, and so this scheme does not provide integrity or authenticity.

SID _____

**Problem 8** ***Food!*** **(20 points)**

Suppose there is a database where each entry is a name of a person and the person's favorite food, all encrypted. Mallory is an "honest but curious" employee at the company who knows the names of every person in the database but wants to know about their favorite food. However, she does not have the private keys to any encryption scheme the database uses.

Note that in this particular database, the names do not repeat, but the foods may repeat. Also assume that when encrypting, padding is used so the length is the same.

(a) Suppose there is a request made to the database to fetch all the names. Each name is encrypted and sent out, and Mallory can see all of these encrypted names. More concretely, she sees $E_k(\text{name}_1), E_k(\text{name}_2), \ldots$ If the encryption scheme was deterministic could Mallory learn anything new? Why or why not?

**Solution:**

Mallory will learn nothing new since the names are unique (and their encryptions are of equal length)

(b) Could Mallory learn anything new if the encryption scheme was IND-CPA? Why or why not?

**Solution:**

Same answer as above

(c) Suppose a new request is made to obtain all the foods. As previous, Mallory can see all of these encrypted values: $E_k(\text{food}_1), E_k(\text{food}_2), \ldots$ If the encryption scheme was deterministic, could Mallory learn anything new? Why or why not?

**Solution:**

Yes. This time, Mallory can learn the relative distribution of favorite foods.

(d) Could Mallory learn anything new if the encryption scheme was IND-CPA? Why or why not?

**Solution:**

No, since encryption that is IND-CPA ensures that the same message will output different ciphertexts.

SID _____