Fuzail & Peyrin
Summer 2022

CS 161
Computer Security

Midterm

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

PRINT your student ID: ____

You have 110 minutes. There are 9 questions of varying credit (150 points total).

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	1	30	14	20	9	13	13	14	36	150

For questions with **circular bubbles**, you may select only one choice.

○ Unselected option (completely unfilled)

● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

☐ You can select

■ multiple squares (completely filled)

Pre-exam activity (not graded, just for fun): Who is EvanBot?

Q1 *Honor Code* (1 point)

Read the following honor code and sign your name.

I understand that I may not collaborate with anyone else on this exam, or cheat in any way. 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 and may further result in, at minimum, negative points on the exam.

SIGN your name: ____

Q2 *True/False* (30 point)

Each true/false is worth 2 points.

Q2.1 True or False: In order for Joey charge his electric car, he gives his assistant a “charging key” that only allows his assistant to access the GPS, steering wheel, and nothing else. This best illustrates the principle of “Shannon’s Maxim.”

○ (A) True ○ (B) False

Q2.2 True or False: In little-endian x86, the least-significant byte of multi-byte numbers is placed in the highest memory address.

○ (A) True ○ (B) False

Q2.3 True or False: x86 registers are stored as part of memory.

○ (A) True ○ (B) False

Q2.4 True or False: A `%c` format string modifier moves `printf`’s argument pointer by a single byte.

○ (A) True ○ (B) False

Q2.5 True or False: If you don’t know the exact address of shellcode, you can still redirect execution to the shellcode.

○ (A) True ○ (B) False

Q2.6 True or False: In real-world systems, canary values often contain a null byte to mitigate string-based attacks.

○ (A) True ○ (B) False

Q2.7 True or False: Block ciphers, such as AES, are IND-CPA secure.

○ (A) True ○ (B) False

Q2.8 True or False: AES-CTR requires PKCS-7 padding to function correctly.

○ (A) True ○ (B) False

Q2.9 TRUE or FALSE: Given $\mathsf{SHA2}(M)$, an attacker cannot compute $\mathsf{SHA2}(M||M')$ for any M' of the attacker's choosing, since SHA2 is a secure cryptographic hash function.

○ (A) TRUE ○ (B) FALSE

Q2.10 xTRUE or FALSE: A MITM during the Diffie-Hellman key exchange can force Alice and Bob to derive a shared key that is different than the one they would have derived without the MITM.

○ (A) TRUE ○ (B) FALSE

Q2.11 TRUE or FALSE: Encrypting passwords before storing them is the best password storage scheme because the ciphertext does not leak any information about the password.

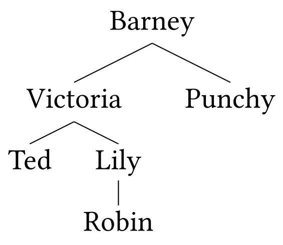
○ (A) TRUE ○ (B) FALSE

Q2.12 TRUE or FALSE: A password database stores passwords for 100 users. There are 5000 possible passwords.

If the database stores each password as $H(\text{password})$, an attacker has to compute 5000×100 hashes to learn every password.

○ (A) TRUE ○ (B) FALSE

Refer to the following certificate tree for the remaining questions.



Q2.13 TRUE or FALSE: If Victoria's secret key is compromised by an attacker, then no certificates generated by anyone in the tree can be trusted.

○ (A) TRUE ○ (B) FALSE

Q2.14 TRUE or FALSE: If Robin is trying to securely get Lily's public key, the only way to do so would be to obtain a certificate that was signed by Barney.

○ (A) TRUE ○ (B) FALSE

Q2.15 True or False: If Robin is trying to securely get Punchy's public key, the only way to do so would be to obtain a certificate that was signed by Barney.

○ (A) True ○ (B) False

Q3 *IV Been Running Through Your Mind All Day* (14 point)

For each of the following schemes, mark whether the scheme is IND-CPA secure or not, and why.

Assume that K_1 and K_2 are different symmetric keys not known to the attacker.

Clarification during exam: Unless otherwise specified, the IV is randomly generated per encryption.

Q3.1 (3 points) AES-CBC(K_1, M), where the IV is chosen as HMAC-SHA256(K_2, M), truncated to the first 128 bits.

- ○ (A) IND-CPA secure, because HMAC functions produce output that is indistinguishable from random.
- ○ (B) IND-CPA secure, because we avoid key reuse by ensuring that $K_1 \neq K_2$.
- ○ (C) Not IND-CPA secure, because an attacker can predict future IV values.
- ○ (D) Not IND-CPA secure, because HMAC functions are deterministic.

Q3.2 (3 points) AES-CBC(K_1, M)

PRNG is a secure, rollback-resistant PRNG that has been seeded once with some initial entropy. To generate an IV for each message, generate bytes from PRNG.

Assume that the attacker has compromised the internal state of the PRNG.

- ○ (A) IND-CPA secure, because the PRNG produces output that is indistinguishable from random.
- ○ (B) IND-CPA secure, because even if an attacker compromises the internal state of the PRNG, they cannot learn anything about future outputs.
- ○ (C) IND-CPA secure, because even if the IV is predictable, AES-CBC with predictable IVs does not provide any additional information to the attacker.
- ○ (D) Not IND-CPA secure, because compromising the internal state of the PRNG results in the attacker learning about previous IVs.
- ○ (E) Not IND-CPA secure, because a predictable IV in AES-CBC allows the attacker to learn additional information about the plaintexts.

Q3.3 (3 points) $\mathsf{AES\text{-}CTR}(K_1, M)$.

PRNG is a secure, rollback-resistant PRNG that has been seeded once with some initial entropy. To generate an IV for each message, generate bytes from PRNG.

Assume that the attacker has compromised the internal state of the PRNG.

- ○ (A) IND-CPA secure, because even if an attacker compromises the internal state of the PRNG, they cannot learn anything about future outputs.
- ○ (B) IND-CPA secure, because even if the IV is predictable, AES-CTR with predictable IVs does not provide any additional information to the attacker.
- ○ (C) Not IND-CPA secure, because compromising the internal state of the PRNG results in the attacker learning about previous IVs.
- ○ (D) Not IND-CPA secure, because a predictable IV in AES-CTR allows the attacker to learn additional information about the plaintexts.

For this question, all future subparts are independent of the previous subparts.

Alice and Bob come up with a new symmetric encryption scheme as follows:

$$M_0 = C_0 = IV$$
$$C_i = M_{i-1} \oplus E_K(M_i) \oplus C_{i-1}$$

E refers to AES-128 block cipher encryption.

Q3.4 (2 points) Write a decryption formula for M_i (for $i > 0$) using this scheme.

Q3.5 (3 points) Is the scheme IND-CPA secure?

- ○ (A) Yes, because C_i is the output of a block cipher that is indistinguishable from random.
- ○ (B) Yes, because AES-CBC is IND-CPA secure, and additionally XORing each C_i with M_{i-1} does not give the attacker any new information.
- ○ (C) No, because encrypting the same plaintext twice gives the same ciphertext twice.
- ○ (D) No, because the attacker can manipulate the ciphertext to learn a deterministic value.

Q4 *Slap Bet Commissioner* (20 point)

Alice and Bob are trying to communicate over an insecure communication channel without their messages getting tampered by Mallory, a MITM.

Assume **Mallory knows the 256-bit, one-block message,** M, that is being sent across the channel, in addition to the value of the ciphertext, C. For each of the following schemes, first mark whether Mallory can change the value M to be some value of her choosing, M'.

If you mark "Yes", provide a formula for $C' = (C'_1, C'_2)$ that, when decrypted by Bob, results in the message M'. Write your answer in terms of M, C_1, C_2, M', C'_1, and C'_2. You may also slice any of these values (for example, $M[0{:}127]$ returns the first 128 bits of M).

If you mark "No", write "Not Needed" in the textbox.

Assume that:

- K_1 and K_2 are different, symmetric keys known only to Alice and Bob.
- PK_B is Bob's public key that is certified by a trusted server.
- AES-CBC and AES-CTR are called with randomly generated IVs.
- H is SHA2-256, a secure, cryptographic, 256-bit hash function.

Q4.1 (4 points) Alice sends $C = (C_1, C_2)$

$$C_1 = \textsf{AES-CTR}(K_1, M)$$

$$C_2 = \text{not used}$$

○ (A) Yes, Mallory can modify the message. ○ (B) No, Mallory cannot modify the message.

Q4.2 (4 points) Alice sends $C = (C_1, C_2)$

$$C_1 = M$$

$$C_2 = \textsf{HMAC}(K_1, M)$$

○ (A) Yes, Mallory can modify the message. ○ (B) No, Mallory cannot modify the message.

Q4.3 (4 points) Alice sends $C = (C_1, C_2)$

$$C_1 = \mathsf{AES\text{-}CBC}(K_1, M)$$
$$C_2 = \mathsf{HMAC}(PK_B, C_1)$$

○ (A) Yes, Mallory can modify the message. ○ (B) No, Mallory cannot modify the message.

Q4.4 (4 points) Alice sends $C = (C_1, C_2)$

$$C_1 = M$$
$$C_2 = H(K_1 \mid\mid C_1)$$

○ (A) Yes, Mallory can modify the message. ○ (B) No, Mallory cannot modify the message.

Q4.5 (4 points) Alice sends $C = (C_1, C_2)$

$$C_1 = \mathsf{AES\text{-}CTR}(K_1, M)$$
$$C_2 = H(K_2) \mid\mid H(C_1)$$

○ (A) Yes, Mallory can modify the message. ○ (B) No, Mallory cannot modify the message.

Q5 *Lawyered!* (9 point)

Bob goes in front of Judge Marshal and needs to show proof that a message M was sent by Alice without any tampering. For each of the following schemes, select whether Judge Marshal should believe Bob based on his explanations and explain why or why not. Assume that Judge Marshal has the values of all secret keys. You can answer in 10 words or fewer.

Clarification during exam: Assume that Judge Marshal is not malicious.

Q5.1 (3 points) Bob shows Judge Marshal $\mathsf{MAC}(K, M)$. Bob argues that since the message is MACd using a secret, symmetric key, shared between only Alice and Bob, and MACs provide integrity, the message must have been sent by Alice.

○ (A) Yes, Judge Marshal can believe Bob. ○ (B) No, Judge Marshal cannot believe Bob.

Q5.2 (3 points) Bob shows Judge Marshal $\mathsf{Sign}(PK_A, M)$. Bob argues that since the message is signed by Alice's public key and digital signatures provide integrity and authenticity, the message must have come from Alice.

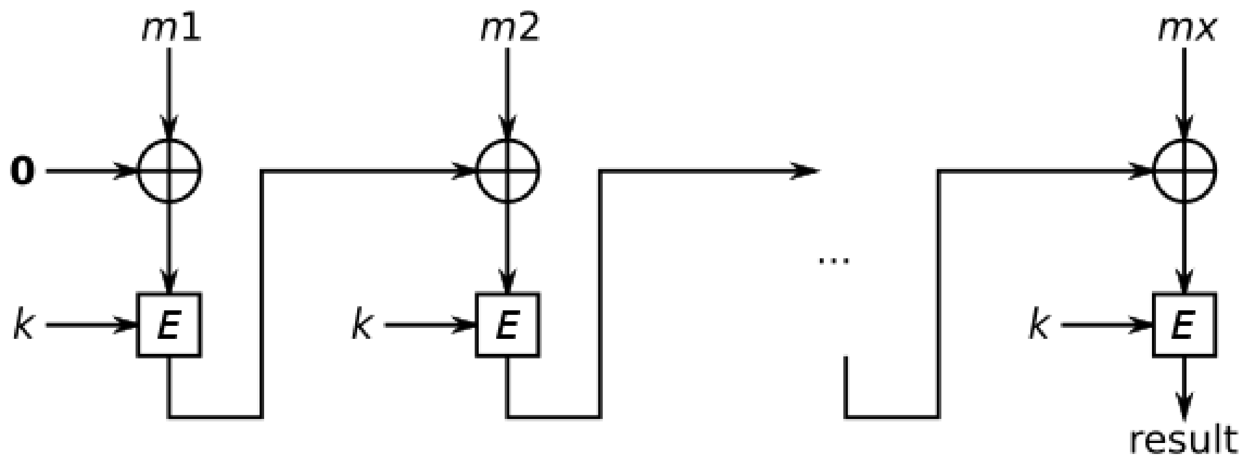
○ (A) Yes, Judge Marshal can believe Bob. ○ (B) No, Judge Marshal cannot believe Bob.

Q5.3 (3 points) Bob shows Judge Marshal $E(SK_A, M)$, where E is AES-128 block cipher encryption. Bob argues that since the message is encrypted with Alice's secret key, the message must have been sent by Alice.

○ (A) Yes, Judge Marshal can believe Bob. ○ (B) No, Judge Marshal cannot believe Bob.

Q6 *You sure he'll crack that code?* (13 point)

Consider the following scheme called CBC-MAC. CBC-MAC is very similar to AES-CBC. The IV is always 0 and the algorithm outputs the last block of the ciphertext as the MAC. The image below illustrates how blocks of a message are encrypted with E (AES-128 block cipher encryption) for a resulting MAC.



Alice and Bob are communicating over an insecure channel. K is a symmetric key known only to Alice and Bob.

For each message, Alice sends:

$$\text{AES-CBC}(K, M), \text{CBC-MAC}(K, M)$$

Clarification during exam: Unless otherwise specified, the IV is randomly generated per encryption.

Q6.1 (2 points) Select all true statements about this scheme.

☐ (A) Bob can always recover M.

☐ (B) A MITM can learn something about M.

☐ (C) A MITM can read some, but not all, of M.

☐ (D) A MITM can read all of M.

☐ (E) None of the above

Q6.2 (2 points) Alice sends a two-block message. Is Mallory able to tamper with the first block M_1 to some different value M_1' without being detected by Bob?

○ (A) Yes, without tampering with the CBC-MAC value

○ (B) Yes, by also tampering with the CBC-MAC value

○ (C) No

Q6.3 (2 points) Alice sends a two-block message. Is Mallory able to tamper with the second block M_2 to some different value M_2' without being detected by Bob?

- ○ (A) Yes, without tampering with the CBC-MAC value
- ○ (B) Yes, by also tampering with the CBC-MAC value
- ○ (C) No

Q6.4 (4 points) Alice sends an n-block message. Which blocks of the message is Mallory able to tamper with, without being detected by Bob?

Give your answer in terms of a series of inclusive ranges. For example $[1, 5]$, $[n - 3, n]$ refers to Mallory being able to tamper with the first five blocks and the last three blocks. If none of the blocks can be tampered with, write "None".

Clarification during exam: $[n - 3, n]$ refers to the last four blocks.

Q6.5 (3 points) How would you change CBC-MAC to prevent your attack from the previous subparts? If you said there was no attack, explain why CBC-MAC is secure. Limit your answer to 10 words or fewer.

Q7 *La Magle* (13 point)

Alice and Bob want to communicate over a public channel using El Gamal encryption. Unfortunately, they have forgotten the details of El Gamal and decide to devise their own encryption scheme they call La Magle. Below are the details for Alice and Bob's La Magle encryption scheme in the case where Alice is sending a message to Bob.

- **Key Generation:** Bob randomly picks b in the range $[2, p-2]$, and computes $B = g^b \bmod p$. Bob's public key is B and his private key is b.
- **Encryption:** Let Alice's message be some M between $[0, p-2]$. Alice randomly picks r in the range $[0, p-2]$. Alice then sends the following (C_1, C_2) as her ciphertext:

$$C_1 = g^r \bmod p$$
$$C_2 = g^M \times B^r \bmod p$$

Assume the parameters g and p are agreed upon and known by all parties involved.

Q7.1 (3 points) Bob has received the ciphertext $C = (C_1, C_2)$ from Alice which was encrypted using La Magle. Bob has access to the following magic functions and all magic functions have efficient runtimes.

- **Discrete Log Oracle:** A function $DL(y)$, that when given any integer y, returns an integer x such that $y \equiv g^x \bmod p$.
- **Diffie-Hellman Oracle:** A function $DH(x, y)$, that when given any two integers of the form $g^a \bmod p$ and $g^b \bmod p$, returns $g^{ab} \bmod p$.
- **RSA Oracle:** A function $RSA(x)$, that when given a large value, N, factors it into prime numbers p and q.

Provide a decryption formula, $D(C_1, C_2)$ for M in terms of C_1, C_2, b, g, p, and **at most one** of the above magic functions.

Q7.2 (4 points) Alice encrypts the plaintexts M_1 and M_2 and sends them to Bob in the form $C = (C_1, C_2)$ and $C' = (C'_1, C'_2)$. Bob wants to compute the sum of the plaintexts, but Bob is lazy and only willing to use the decryption function found above once.

Provide a formula that Bob could use to get $M_1 + M_2$ in terms of C_1, C_2, C'_1, C'_2, and the decryption function $D(C_1, C_2)$. You may use the decryption function $D(C_1, C_2)$ **at most once** in your formula. You **cannot** use Bob's private key b in your formula.

Q7.3 (3 points) Recall that in El Gamal, Mallory can tamper with the encryption of M so that Bob decrypts the ciphertext as $2 \times M$. Is the same attack possible in La Magle?

○ (A) Yes ○ (B) No

Q7.4 (3 points) Alice and Bob decide to further modify La Magle. Under this modified scheme, Alice sends the following (C_1, C_2) as her ciphertext:

$$C_1 = \textsf{AES-CTR}(K_1, g^r \bmod p)$$

$$C_2 = \textsf{AES-CTR}(K_2, g^m \times B^r \bmod p)$$

K_1 are K_2 are different symmetric keys known only to Alice and Bob.

Is this modified version of La Magle a malleable scheme? If yes, provide an example of how Mallory could forge a new ciphertext. If no, provide a brief justification about why it is not possible. You can answer in 10 words or fewer.

○ (A) Yes ○ (B) No

Q8 *How YOU Doin'?* **(14 point)**

Joey wrote some code to send messages to his friends, but got confused and he has some concerns. Can you help Joey fix his code?

```c
typedef struct {
    char my_lines[16];
} play;

typedef struct {
    char *my_scenes;
} script;

void memorize(int i, char *new_brain, char *al_pacino) {
    play *dool_play;
    script *joey_script;
    uint32_t flag = i;
    char buf[12] = "A MOO POINT";

    dool_play = (play *) malloc(sizeof(play));
    joey_script = (script *) malloc(sizeof(script));
    joey_script->my_scenes = (char *) malloc(128);

    memcpy(joey_script->my_scenes, buf, flag);
    strcpy(dool_play->my_lines, new_brain);

    if (strlen(al_pacino) >= 128) {
        printf("Your message is too long!\n");
        return;
    }

    strncpy(joey_script->my_scenes, al_pacino, strlen(al_pacino)+1);
}
```

For this question, assume the following:

- You are on a little-endian 32-bit x86 system.
- There is no other compiler padding or saved additional registers.
- `main` calls `memorize` with the appropriate arguments.
- **No memory safety defenses** are enabled.
- The address of the RIP of `memorize` is `0xffffff9c`.
- The address of `dool_play->my_lines` on the heap is `0x7ff3ec10`.
- For your inputs, you may use `SHELLCODE` as a 100-byte shellcode. **Note that there are fewer than 100 bytes above the RIP of `memorize`.**

The heap diagram when the program is paused at **Line 19** is:

Heap

[128] `*(joey_script->my_scenes)`
[4] `joey_script->my_scenes`
[16] `dool_play->my_lines`

Assume that there is no padding or any other values between blocks on the heap.

Q8.1 (2 points) Which of the following memory safety vulnerabilities exist in the code? Select all that apply.

☐ (A) Format string vulnerability

☐ (B) Signed/unsigned vulnerability

☐ (C) Heap overflow

☐ (D) Off-by-one vulnerability

☐ (E) None of the above

Q8.2 (10 points) Provide an input that would cause this program to execute shellcode. Write all your answers in Python 2 syntax (just like in Project 1). If you don't need a line for your exploit, you must write "Not Needed".

Provide an input to variable `i` (argument to `memorize`).

For this box only, you can write a decimal number instead of its byte representation.

Provide a string value for `new_brain` (argument to `memorize`):

Provide a string value for `al_pacino` (argument to `memorize`):

Q8.3 (2 points) Which of the following memory safety defenses would prevent an attacker from executing the exploit above? Select all that apply.

☐ (A) Stack canaries

☐ (B) ASLR

☐ (C) Using a memory safe language

☐ (D) Non-executable pages

☐ (E) None of the above

Q9 *The Way You Look Tonight* (36 point)

Consider the following vulnerable C code:

```c
typedef struct {
    char mon[16];
    char chan[16];
} duo;

void third_wheel(char *puppet, FILE *f) {
    duo mondler;
    duo richard;
    fgets(richard.mon, 16, f);
    strcpy(richard.chan, puppet);
    int8_t alias = 0;
    size_t counter = 0;

    while (!richard.mon[15] && richard.mon[0]) {
        size_t index = counter / 10;
        if (mondler.mon[index] == 'A') {
            mondler.mon[index] = 0;
        }
        alias++;
        counter++;
        if (counter == ___ || counter == ___) {
            richard.chan[alias] = mondler.mon[alias];
        }
    }

    printf("%s\n", richard.mon);
    fflush(stdout); // no memory safety vulnerabilities on this line
}

void valentine(char *tape[2], FILE *f) {
    int song = 0;
    while (song < 2) {
        read_input(tape[song]); //memory-safe function, see below
        third_wheel(tape[song], f);
        song++;
    }
}
```

For all of the subparts, here are a few tools you can use:

- You run GDB once, and discover that the address of the RIP of `third_wheel` is `0xffffcd84`.
- For your inputs, you may use `SHELLCODE` as a 100-byte shellcode.
- The number `0xe4ff` exists in memory at address `0x8048773`. The number `0xe4ff` is interpreted as `jmp *esp` in x86.
- If needed, you may use standard output as `OUTPUT`, slicing it using Python 2 syntax.

Assume that:

- You are on a little-endian 32-bit x86 system.
- There is no other compiler padding or saved additional registers.
- `main` calls `valentine` with the appropriate arguments.
- **Stack canaries** are enabled and no other no memory safety defenses are enabled.
- The stack canary is four completely random bytes (**no null byte**).
- `read_input(buf)` is a memory-safe function that writes to `buf` without any overflows.

Write your exploits in Python 2 syntax (just like in Project 1).

Q9.1 Fill in the following stack diagram, assuming that the program is paused at **Line 14**. Each row should contain a struct member, local variable, the SFP of `third_wheel`, or canary (the value in each row does not have to be four bytes long).

Stack
`RIP of third_wheel`
`SFP of third_wheel`
`STACK CANARY`
`mondler.chan`
`mondler.mon`
`richard.chan`
`richard.mon`
`alias`
`counter`

Q9.2 In the first call to `third_wheel`, we want to leak the value of the stack canary. What should be the missing values at line 21 in order to make this exploit possible?
Provide a decimal integer in each box.

Q9.3 Provide an input to each of the lines below in order to leak the stack canary value in the first call to `third_wheel`. If you don't need an input, you must write "Not Needed".

Provide a string value for `tape[0]`:

Provide an input to `fgets` in `third_wheel`:

Q9.4 Provide an input to each of the lines below in order to run the malicious shellcode in the second call to `third_wheel`. If you don't need an input, you must write "Not Needed".

Provide a string value for `tape[1]`:

Provide an input to `fgets` in `third_wheel`:

For the rest of the question, assume that **ASLR** is enabled in addition to stack canaries. Assume that the code section of memory has not been randomized.

Q9.5 Provide an input to each of the lines below in order to leak the stack canary in the first call to `third_wheel`. If you don't need an input, you must write "Not Needed".

Provide a string value for `tape[0]`:

Provide an input to `fgets` in `third_wheel`:

Q9.6 Provide an input to each of the lines below in order to run the malicious shellcode in the second call to `third_wheel`. If you don't need an input, you must write "Not Needed".

Provide a string value for `tape[1]`:

Provide an input to `fgets` in `third_wheel`:

Doodle

Nothing on this page will not affect your grade in any way.

Congratulations for making it to the end of the exam! Feel free to leave any final thoughts, comments, feedback, or doodles here: