Nicholas & Peyrin Summer 2021 — CS 161 Computer Security — Midterm

For questions with **circular bubbles**, you may select exactly *one* choice on Examtool.

- ◯ Unselected option
- ● Only one selected option

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

- ■ You can select
- ■ multiple squares

For questions with a **large box**, you need to write your answer in the text box on Examtool.

There is an appendix at the end of this exam, containing descriptions of all C functions used on this exam.

You have 110 minutes, plus a 10-minute buffer for distractions or technical difficulties, for a total of 120 minutes. There are 10 questions of varying credit (150 points total).

The exam is open note. You can use an unlimited number of handwritten cheat sheets, but you must work alone.

Clarifications will be posted on Examtool.

Q1 *MANDATORY – Honor Code* **(5 points)**

Read the following honor code and type your name on Examtool.

> 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 and a corresponding notch on Nick's Stanley Fubar demolition tool.

Q2 *True/false* **(28 points)**

Each true/false is worth 2 points.

Q2.1 TRUE or FALSE: If the attacker can only overwrite a function's SFP but not the RIP, the attacker cannot cause shellcode to execute.

◯ TRUE ◯ FALSE

Q2.2 TRUE or FALSE: ECB mode only leaks information if you encrypt two identical messages.

◯ TRUE ◯ FALSE

Q2.3 TRUE or FALSE: If a cryptographic hash is collision-resistant, a pair of two different inputs that hash to the same output does not exist.

◯ TRUE ◯ FALSE

Q2.4 TRUE or FALSE: A common approach to communicating securely and quickly is first using symmetric-key cryptography to send a key, then using public-key cryptography to send messages.

◯ TRUE ◯ FALSE

Q2.5 TRUE or FALSE: Enabling ASLR prevents all memory attacks on the stack.

◯ TRUE ◯ FALSE

Q2.6 TRUE or FALSE: In x86 calling convention, the SFP is located at a higher address than the RIP.

◯ TRUE ◯ FALSE

Q2.7 TRUE or FALSE: Using El Gamal together with Diffie Hellman to encrypt messages provides both confidentiality and integrity.

◯ TRUE ◯ FALSE

Q2.8 Alice obtains a copy of a digital certificate for Bob from an untrustworthy source. She trusts the certificate authority (CA) who signed Bob's certificate.

TRUE or FALSE: It is safe for Alice to trust the certificate after she verifies the signature.

◯ TRUE ◯ FALSE

Q2.9 TRUE or FALSE: Stack canaries that include a fixed NULL byte are easier to brute-force than stack canaries with 4, completely random bytes.

◯ TRUE ◯ FALSE

Q2.10 TRUE or FALSE: One problem with the Trusted Directory (TD) model discussed in lecture is that users have no way of reliably determining the TD's public key.

◯ TRUE ◯ FALSE

Q2.11 TRUE or FALSE: Certificate authorities solve the problem of scalability by allowing delegated trust.

○ True ○ False

Q2.12 True or False: Storing the hash of the passwords prevents any attacker from learning passwords.

○ True ○ False

Q2.13 True or False: Rollback resistance is a required property for a secure PRNG.

○ True ○ False

Q2.14 True or False: MACs are a symmetric-key protocol.

○ True ○ False

Q3 *Security Principles* (15 points)

For each scenario, select the most relevant security principle. Each option is used exactly once.

Q3.1 (3 points) To prevent memory safety vulnerabilities, a programmer enables ASLR, non-executable pages, and stack canaries.

- ◯ (A) Defense in depth
- ◯ (B) Detect if you can't prevent
- ◯ (C) Separation of privilege
- ◯ (D) Consider human factors
- ◯ (E) Ensure complete mediation
- ◯ (F) ——

Q3.2 (3 points) A bank installs alarms to alert the security guards in case intruders break in.

- ◯ (G) Defense in depth
- ◯ (H) Detect if you can't prevent
- ◯ (I) Separation of privilege
- ◯ (J) Consider human factors
- ◯ (K) Ensure complete mediation
- ◯ (L) ——

Q3.3 (3 points) To access top-secret CS 161 data, Nicholas must enter a password that only he knows, and Peyrin must enter a second password that only he knows.

- ◯ (A) Defense in depth
- ◯ (B) Detect if you can't prevent
- ◯ (C) Separation of privilege
- ◯ (D) Consider human factors
- ◯ (E) Ensure complete mediation
- ◯ (F) ——

Q3.4 (3 points) When writing C code, a programmer decides to leave stack canaries disabled, because they forgot the name of the compiler flag for enabling canaries.

- ◯ (G) Defense in depth
- ◯ (H) Detect if you can't prevent
- ◯ (I) Separation of privilege
- ◯ (J) Consider human factors
- ◯ (K) Ensure complete mediation
- ◯ (L) ——

Q3.5 (3 points) In an airport, every passenger must pass through the security checkpoint.

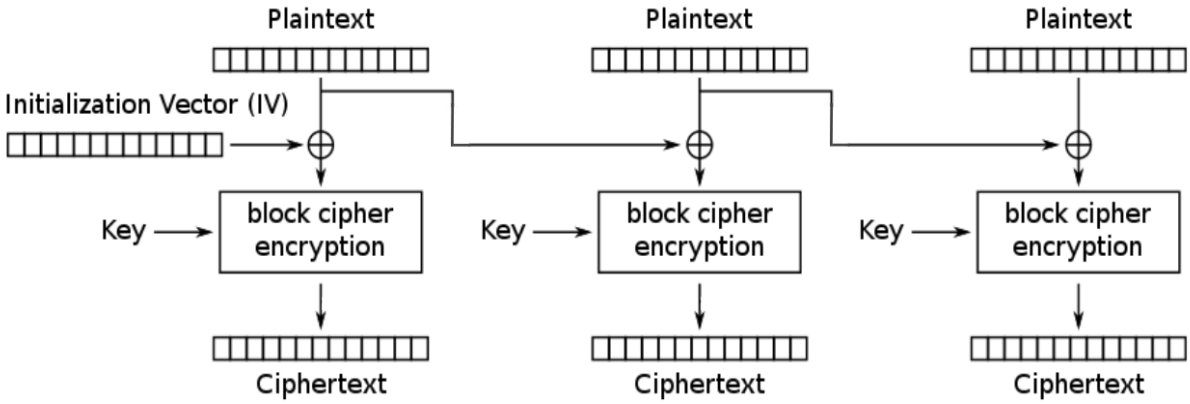
- ◯ (A) Defense in depth
- ◯ (B) Detect if you can't prevent
- ◯ (C) Separation of privilege
- ◯ (D) Consider human factors
- ◯ (E) Ensure complete mediation
- ◯ (F) ——

Q4 *Block Ciphers* **(15 points)**

Consider the following block cipher mode of operation.

M_i is the ith plaintext block. C_i is the ith ciphertext block. E_K is AES encryption with key K.

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



Q4.1 (5 points) Which of the following is true about this scheme? Select all that apply.

☐ (A) The encryption algorithm is parallelizable

☐ (B) If one byte of a plaintext block M_i is changed, then the corresponding ciphertext block C_i will be different in exactly one byte

☐ (C) If one byte of a plaintext block M_i is changed, then the next ciphertext block C_{i+1} will be different in exactly one byte

☐ (D) If two plaintext blocks are identical, then the corresponding ciphertext blocks are also identical

☐ (E) The encryption algorithm requires padding the plaintext

☐ (F) None of the above

Q4.2 (4 points) TRUE or FALSE: If the IV is always a block of all 0s for every encryption, this scheme is IND-CPA secure. Briefly justify your answer.

◯ (G) True ◯ (H) False ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Q4.3 (6 points) TRUE or FALSE: If the IV is randomly generated for every encryption, this scheme is IND-CPA secure. Briefly justify your answer.

◯ (A) True ◯ (B) False ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Q5 *Certificates* **(10 points)**

You are working as a software engineer for an online discussion forum called Piazzzzza, which uses the following certificate hierarchy:

1. Everyone has access to the public key of a trusted root certificate authority (CA)
2. The root CA uses its private key to sign a certificate C for Piazzzzza's public key
3. Piazzzzza uses its private key to sign a certificate for each user's public key

Q5.1 (2 points) TRUE or FALSE: An attacker who steals the private key of the root CA can forge C.

◯ (A) True ◯ (B) False ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Q5.2 (2 points) TRUE or FALSE: An attacker who steals the private key of Piazzzzza can forge C.

◯ (G) True ◯ (H) False ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Q5.3 (2 points) TRUE or FALSE: An attacker who steals the private key of a user can forge C.

◯ (A) True ◯ (B) False ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Q5.4 (4 points) Suppose you are talking with someone claiming to be Jinan. Assume you have Jinan's public key.

Which of the following pieces of information on its own can prove that you are really talking with Jinan? Select all that apply.

☐ (G) The root certificate

☐ (H) Jinan's certificate

☐ (I) A message "You are talking to Jinan" signed by Jinan's private key

☐ (J) A message "You are talking to Jinan" signed by the root CA's private key

☐ (K) None of the above

☐ (L) ——

Q6 *Password Storage* **(12 points)**

Consider a website that needs to securely store the filename-password pairs in a database.

Notation:

- pwd is the password that we are storing in the database.
- salt is a randomly generated 256-bit string that is different for each password in the database.
- Hash is a secure cryptographic hash function. Hash is not vulnerable to length extension attacks. The attacker knows the hash function being used.

Assumptions:

- Every password is exactly 10 characters.
- The attacker has a precomputed table of the hash of every possible password.
- The attacker will not compute any hashes unless otherwise stated.
- The attacker can read all the records in the database.

For each password storage scheme, select all true statements.

Clarification during exam: For schemes involving a salt, assume each salt is randomly generated per user and stored in a row with the username and hashed password.

Clarification during exam: Assume that the attacker may compute as many XOR operations as they want.

Q6.1 (3 points) $\mathsf{Hash}(\mathsf{pwd}\|\mathsf{salt})$ and salt

☐ (A) The attacker can learn every user's password

☐ (B) The attacker can verify that a given password for a particular user is correct by computing at most one hash

☐ (C) The attacker can determine if two users have the same password without using the precomputed table

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Q6.2 (3 points) $(\mathsf{Hash}(\mathsf{pwd}) \oplus \mathsf{salt})$ and salt

☐ (G) The attacker can learn every user's password

☐ (H) The attacker can verify that a given password for a particular user is correct by computing at most one hash

☐ (I) The attacker can determine if two users have the same password without using the precomputed table

☐ (J) None of the above

☐ (K) —

☐ (L) —

Q6.3 (3 points) Hash(pwd)

☐ (A) The attacker can learn every user's password

☐ (B) The attacker can verify that a given password for a particular user is correct by computing at most one hash

☐ (C) The attacker can determine if two users have the same password without using the precomputed table

☐ (D) None of the above

☐ (E) —

☐ (F) —

Q6.4 (3 points) Suppose that Piazzzzza limits users to only be able to try inputting a password three times per minute. Which of the following attacks does this defend against?

◯ (G) Online brute-force attacks

◯ (H) Offline brute-force attacks

◯ (I) Eavesdropping

◯ (J) Format string vulnerability

◯ (K) —

◯ (L) —

Q7 *Encryption and Authentication* **(15 points)**

Alice wants to send messages to Bob, but Mallory (a man-in-the-middle attacker) will read and tamper with data sent over the insecure channel.

- Alice and Bob share two secret keys K_1 and K_2
- K_1 and K_2 have not been leaked (Alice and Bob are the only people who know the keys)
- Enc is an IND-CPA secure encryption scheme
- MAC is a secure (unforgeable) MAC scheme

For each cryptographic scheme, select all true statements.

Clarification during exam: For the answer choice "Bob can always recover the message M," assume that Mallory has not tampered with the message.

Clarification during exam: The answer choice "Bob can guarantee that M has not been changed by Mallory," this should say "Bob can guarantee that M has not been changed by Mallory without detection."

Q7.1 (4 points) $\mathsf{Enc}(K_1, M), \mathsf{MAC}(K_2, M)$

- ☐ (A) Bob can guarantee M is from Alice
- ☐ (B) Bob can guarantee that M has not been changed by Mallory
- ☐ (C) Mallory cannot read M
- ☐ (D) Bob can always recover the message M
- ☐ (E) None of the above
- ☐ (F) ——

Q7.2 (4 points) $\mathsf{Enc}(K_1, M), \mathsf{MAC}(K_2, \mathsf{Enc}(K_1, M))$

- ☐ (G) Bob can guarantee M is from Alice
- ☐ (H) Bob can guarantee that M has not been changed by Mallory
- ☐ (I) Mallory cannot read M
- ☐ (J) Bob can always recover the message M
- ☐ (K) None of the above
- ☐ (L) ——

Q7.3 (4 points) $\mathsf{Hash}(M), \mathsf{MAC}(K_1, M)$

- ☐ (A) Bob can guarantee M is from Alice
- ☐ (B) Bob can guarantee that M has not been changed by Mallory
- ☐ (C) Mallory cannot read M

☐ (D) Bob can always recover the message M

☐ (E) None of the above

☐ (F) ——

Q7.4 (3 points) To simplify their schemes, Alice and Bob decide to set $K_1 = K_2$. (In other words, K_1 and K_2 are the same key.) Does this affect the security of their cryptographic schemes?

◯ (G) Yes, because they should always use a different key for every algorithm

◯ (H) Yes, because they should always use a different key for every message

◯ (I) No, because the encryption and MAC schemes are secure.

◯ (J) No, because the keys cannot be brute-forced.

◯ (K) ——

◯ (L) ——

Q8 *PRNGs and Diffie-Hellman Key Exchange* **(15 points)**

Eve is an eavesdropper listening to an insecure channel between Alice and Bob.

1. Alice and Bob each seed a PRNG with different random inputs.
2. Alice and Bob each use their PRNG to generate some pseudorandom output.
3. Eve learns both Alice's and Bob's pseudorandom outputs from step 2.
4. Alice, without reseeding, uses her PRNG from the previous steps to generate a, and Bob, without reseeding, uses his PRNG from the previous steps to generate b.
5. Alice and Bob perform a Diffie-Hellman key exchange using their generated secrets (a and b). Recall that, in Diffie-Hellman, neither a nor b are directly sent over the channel.

For each choice of PRNG constructions, select the minimum number of PRNGs Eve needs to compromise (learn the internal state of) in order to learn the Diffie-Hellman shared secret $g^{ab} \bmod p$. Assume that Eve always learns the internal state of a PRNG between steps 3 and 4.

Q8.1 (3 points) Alice and Bob both use a PRNG that outputs the same number each time.

◯ (A) Neither PRNG ◯ (C) Both PRNGs ◯ (E) ——

◯ (B) One PRNG ◯ (D) Eve can't learn the secret ◯ (F) ——

Q8.2 (3 points) Alice uses a secure, rollback-resistant PRNG. Bob uses a PRNG that outputs the same number each time.

◯ (G) Neither PRNG ◯ (I) Both PRNGs ◯ (K) ——

◯ (H) One PRNG ◯ (J) Eve can't learn the secret ◯ (L) ——

Q8.3 (3 points) Alice and Bob both use a secure, rollback-resistant PRNG.

◯ (A) Neither PRNG ◯ (C) Both PRNGs ◯ (E) ——

◯ (B) One PRNG ◯ (D) Eve can't learn the secret ◯ (F) ——

For the rest of the question, consider a different sequence of steps:

1. Alice and Bob each seed a PRNG with different random inputs.
2. Alice uses her PRNG from the previous step to generate a, and Bob uses his PRNG from the previous step to generate b.
3. Alice and Bob perform a Diffie-Hellman key exchange using their generated secrets (a and b).
4. Alice and Bob, without reseeding, each use their PRNG to generate some pseudorandom output.
5. Eve learns both Alice's and Bob's pseudorandom outputs from step 2.

As before, assume that Eve always learns the internal state of a PRNG between steps 3 and 4.

Q8.4 (3 points) Alice and Bob both use a secure, but not rollback-resistant PRNG.

◯ (G) Neither PRNG ◯ (I) Both PRNGs ◯ (K) ——

◯ (H) One PRNG ◯ (J) Eve can't learn the secret ◯ (L) ——

Q8.5 (3 points) Alice and Bob both use a secure, rollback-resistant PRNG.

◯ (A) Neither PRNG ◯ (C) Both PRNGs ◯ (E) ——

◯ (B) One PRNG ◯ (D) Eve can't learn the secret ◯ (F) ——

Q9 *Memory Safety Mitigations* **(12 points)**

Suppose we are on a 64-bit system, and we have an address space of 2^{50} bytes.

Q9.1 (3 points) How many unused bits are available for pointer authentication in each address?

◯ (A) None ◯ (B) 4 ◯ (C) 11 ◯ (D) 14 ◯ (E) 17 ◯ (F) 32

Q9.2 (3 points) Regardless of your answer to the previous part, for the rest of the question, assume that 10 bits are used for pointer authentication in each address.

Additionally, for the rest of the question, assume that 64-bit stack canaries are enabled. The first byte of the stack canary is always a null byte.

Assume the attacker does not have the ability to create their own pointer authentication codes (PACs). How many bits does the attacker have to guess correctly to guess the stack canary and the PAC?

◯ (G) 0 ◯ (H) 10 ◯ (I) 56 ◯ (J) 64 ◯ (K) 66 ◯ (L) 74

Q9.3 (3 points) Now assume that the attacker has a format string vulnerability that lets them read any part of memory while the program is running.

Assume the attacker does not have the ability to create their own PACs. How many bits does the attacker have to guess correctly to guess the stack canary and the PAC?

◯ (A) 0 ◯ (B) 10 ◯ (C) 56 ◯ (D) 64 ◯ (E) 66 ◯ (F) 74

Q9.4 (3 points) Assume the attacker is interacting with a remote system. Provide one defense that would make brute-force attacks infeasible for the attacker. (Please answer in 10 words or fewer.)

Q10 *Memory Safety Vulnerabilities* **(23 points)**

Note: This is the hardest question on the exam. We recommend trying the other questions on the exam before this one.

Consider the following vulnerable C code:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct packet {
5      char payload[300];
6      char format[300];
7  };
8
9  void deploy(struct packet *ptr) {
10     printf(ptr->format, ptr->payload);
11 }
12
13 int main(void) {
14     struct packet p;
15     do {
16         strcpy(p.format, "%s\n");
17         gets(p.payload);
18         deploy(&p);
19     } while (strcmp(p.payload, "END") != 0);
20     // Assume loop always exits for subpart 3.
21     return 0;
22 }
```

Assume you are on a little-endian 32-bit x86 system. Assume that there is no compiler padding or additional saved registers in all subparts. For the first 3 subparts, assume that **no memory safety defenses** are enabled.

Fill in the following stack diagram, assuming that execution has entered the call to `printf`:

RIP of `main`
SFP of `main`
(1a)
(1b)
(1c)
(2a)
(2b)
(2c)
(2d)
RIP of `printf`
SFP of `printf`

Q10.1 (3 points) For (1a), (1b), and (1c):

◯ (A) (1a) - `p.format`; (1b) - `p.payload`; (1c) - `ptr`

◯ (B) (1a) - `p.payload`; (1b) - `p.format`; (1c) - `ptr`

◯ (C) (1a) - `ptr`; (1b) - `p.payload`; (1c) - `p.format`

◯ (D) (1a) - `ptr`; (1b) - `p.format`; (1c) - `p.payload`

◯ (E) ——

◯ (F) ——

Q10.2 (3 points) For (2a), (2b), (2c), and (2d):

◯ (G) (2a) - RIP of `deploy`; (2b) - SFP of `deploy`; (2c) - `&ptr->format`; (2d) - `&ptr->payload`

◯ (H) (2a) - SFP of `deploy`; (2b) - RIP of `deploy`; (2c) - `&ptr->format`; (2d) - `&ptr->payload`

◯ (I) (2a) - `&ptr->payload`; (2b) - `&ptr->format`; (2c) - RIP of `deploy`; (2d) - SFP of `deploy`

◯ (J) (2a) - `&ptr->payload`; (2b) - `&ptr->format`; (2c) - SFP of `deploy`; (2d) - RIP of `deploy`

◯ (K) (2a) - RIP of `deploy`; (2b) - SFP of `deploy`; (2c) - `&ptr->payload`; (2d) - `&ptr->format`

◯ (L) ——

Q10.3 (3 points) For this subpart only, assume that you may only execute one iteration of the while loop and that the call to `printf` will not segfault. For this subpart, assume that no memory safety defenses are enabled.

If the address of `p` is `0x7ff3ec10`, construct an input at line 18 that would cause the program to execute malicious shellcode. You may reference `SHELLCODE` as a 30-byte malicious shellcode. Write your answer in Python 2 syntax (just like in Project 1).

Clarification during exam: Instead of "Line 18," the question should say "Line 17."

For the remaining subparts, assume that **stack canaries are enabled**. Note that this changes the stack diagram!

Q10.4 (5 points) For your exploit, construct a one-line Python helper function `write_byte(addr, byte)` that returns an input for line 17 of the vulnerable C code. This input should ensure that `byte` is written to the address at `addr`. This function may change bytes **above** `addr` (but not below), as long as the correct byte is written at `addr` itself. **The returned input only needs to work for values of `byte` greater than 8.**

Assume that `addr` is given as a 4-byte Python string containing the bytes of the address in little-endian, and assume that `byte` is given as a Python integer between 9 and 255. For example, `write_byte('\xef\xbe\xad\xde', 128)` would be a valid call to this function. Write your answer in Python 2 syntax (just like in Project 1).

```
1 def write_byte(addr, byte):
2     return # Your answer here
```

Hint: You may find the `%c` format specifier useful: Read 4 bytes off the stack and print as a single character.

Q10.5 (5 points) If the address of `p` is `0x7ff3ec10` and the address of the RIP of `main` is `0x7ff3ee68`, construct a series of inputs for repeated calls at line 18 that would cause the program to execute malicious shellcode. Assume that `write_byte` is implemented correctly, and you may call `write_byte` for as many inputs as you would like. Write your answer as a series of `print` statements, all in Python 2 syntax (just like in Project 1).

Hint: You may write hex literals to represent integers in Python, such as `0x36`.

Clarification during exam: Instead of "Line 18," the question should say "Line 17."

Q10.6 (4 points) Which of the following changes, if made on their own, would prevent the attacker from executing malicious shellcode (not necessarily using your exploit above)?

☐ (G) Enabling non-executable pages in addition to stack canaries

☐ (H) Enabling ASLR in addition to stack canaries

☐ (I) Rewriting the code in a memory-safe language

☐ (J) Using `fgets(p.payload, 300, stdin)` instead of `gets(p.payload)` on line 17

☐ (K) None of the above

☐ (L) ——

C Function Definitions

```
int printf(const char *format, ...);

       printf() produces output according to the format string format.

char *strcpy(char *dest, const char *src);

       The strcpy() function copies the string pointed to by src, including
       the terminating null byte ('\0'), to the buffer pointed to by dest.
       The strings may not overlap, and the destination string dest must be
       large enough to receive the copy.

char *gets(char *s);

       gets() reads a line from stdin into the buffer pointed to by s until
       either a terminating newline or EOF, which it replaces with a null byte
       ('\0').

int strcmp(const char *s1, const char *s2);

       The strcmp() function compares the two strings s1 and s2. It returns
       an integer less than, equal to, or greater than zero if s1 is found,
       respectively, to be less than, to match, or be greater than s2.

char *fgets(char *s, int size, FILE *stream);

       fgets() reads in at most one less than size characters from stream and
       stores them into the buffer pointed to by s. Reading stops after an
       EOF or a newline. If a newline is read, it is stored into the buffer.
       A terminating null byte ('\0') is stored after the last character in
       the buffer.
```