CS336 Homework #4

Due: Friday, Oct 8th, 2021, 5:00 pm
Points: 80 pts

1. (10 points) Suppose that someone suggests the following way to confirm that the two of you are both in possession of the same secret key. You create a random bit string the length of the key, XOR it with the key, and send the result over the channel. Your partner XORs the incoming block with the key (which should be the same as your key) and sends it back. You check, and if what you receive is your original random string, you have verified that your partner has the same secret key, yet neither of you has ever transmitted the key. Is there a flaw in this scheme?
    a. There is a flaw in this scheme. Take this example:
        i. User 1 generates random key and XORs it with secret key
            1. 101101 <- Random String
            2. 010110 <- Secret Key
            3. **111011** <- This is sent to User 2
        ii. Now User 2 XORs the result with the secret key:
            1. 111011 - From User 1
            2. 010110 - Secret
            3. **101101** - Result gets sent back to User 1
        iii. User 1 confirms the secret keys match—However:
        iv. Even though the secret key itself has never been sent, all an attacker needs to do is XOR the two strings that *did* get sent to find the secret key.
            1. 111011
            2. 101101
            3. **010110 <- Secret Key**

2. (10 points) Explain why hash collisions occur. That is, why must there always be two different plaintexts that have the same hash value? What property of a hash function means that collisions are not a security problem? That is, why can an attacker not capitalize on collisions and change the underlying plaintext to another form whose value collides with the hash value of the original plaintext?
    a. The hash value is a set length, so there are only so many possible outputs. The inputs, however, are for all intents and purposes infinite. Hash collisions occur because they assign an infinite set of inputs to a finite set of outputs—there will eventually *have* to be repeats. A good hash function has collision resistance; that is, it is computationally infeasible to find another file that will generate a collision.

3. (20 points) In this problem we will compare the security services that are provided by digital signatures (DS) and message authentication codes (MAC). We assume that Oscar is able to observe all messages sent from Alice to Bob and vice versa. Oscar has no knowledge of any keys but the public one in case of DS. State whether and how (i) DS and (ii) MAC protect against each attack. The value auth(x) is computed with a DA or a MAC algorithm, respectively.
    a. (Message integrity) Alice sends a message x = “Transfer $1000 to Mark” in the clear and also sends auth(x) to Bob. Oscar intercepts the message and replaces “Mark” with “Oscar”. Will Bob detect this in either case (i) DS and (ii) MAC?

    a. (DS) This will be detected. The decrypted signature will not match the hash of the new message.
    b. (MAC) This will be detected. The computed MAC will not match the MAC attached to the message.
b. (Replay) Alice sends a message x = “Transfer $1000 to Oscar” in the clear and also sends auth(x) to Bob. Oscar observes the message and signature and sends them 100 times to Bob. Will Bob detect this in either case (i) DS and (ii) MAC?
    a. (DS) This will be detected. The receiver will see a reuse of the same signature and know something is wrong.
    b. (MAC) This might not be detected. With a MAC, the receiver is only assured that the message has not been altered, and that it was signed by the sender.
c. (Sender authentication with cheating third party) Oscar claims that he sent some message x with a valid auth(x) to Bob but Alice claims the same. Can Bob clear the question in either case (i) DS and (ii) MAC?
    a. I’m not sure I understand exactly what this question is asking. It seems to me that Oscar and Alice are both claiming to have sent the same message. In this case:
    b. (DS) Bob can use Oscar and Alice’s public keys to decrypt the hash, whoever’s key results in a matching hash is the original sender.
    c. (MAC) Similarly, Bob should have a secret key for both Oscar and Alice. Whichever secret key results in the correct MAC is the original sender.
d. (Authentication with Bob cheating) Bob claims that he received a message x with a valid signature auth(x) from Alice (e.g., “Transfer $1000 from Alice to Bob”) but Alice claims she has never sent it. Can Alice clear this question in either case (i) DS and (ii) MAC?
    a. (DS) If Bob does not have Alice’s private key, then her public key will not be able to decrypt the hash. Alice can decrypt the signature with her public key, calculate the message hash and compare them to show that it wasn’t her that sent the message.
    b. (MAC) Bob and Alice share the same secret key, so Alice will not be able to show that the message was not from her. She will not be able to prove it wasn’t her from just the MAC.

4. (10 points) Explain what PKI (Public Key Infrastructure) is in your words. Does a PKI perform encryption? Explain your answer. Does a PKI use symmetric or asymmetric encryption? Explain your answer. Why does a PKI need a means to cancel or invalidate certificates? Why is it not sufficient for the PKI to stop distributing a certificate after it becomes invalid?
    a. One of the problems with public key cryptography is that anyone can claim to be anyone else, and give out a public key that decrypts the information they send out. However, how do we know that the person is who they claimed they are? A Public Key Infrastructure is a solution to this. A PKI serves as a central information system that allows people to verify the public keys for authentication purposes. Then to verify that someone is who they say they are, you just make sure the public key you have for that person matches the one in the PKI. The PKI needs to be able to invalidate certificates for reasons like if the certificate expires or if someone gets ahold of the person’s private key. It’s not sufficient to just stop distributing the certificate, as the systems using the

certificates might not be checking with the PKI every time it tries to verify someone's identity.

5. (10 points) Investigate the details of the **integer overflow attack**, how it works, and how the attack string it uses is designed. Then experiment with implementing this attack (write a piece of code which has integer overflow vulnerability, take a screenshot of running it and show me that you have successfully exploited an integer overflow vulnerability).
    a. Data types generally allocated a fixed size of memory depending on the type. For example, in C a character can range in value from 0 to 255. If the program increases that value past the upper range, it will cause an overflow, which whill cut off the overflowed data, leaving you back at the lower range. Or, the value could go from the lower range to the upper end.

b.

integerOverflow.c

hw > hw4 > integerOverflow.c

```c
#include <stdio.h>

int main() {
    short balance = -32768;
    short choice = 0;
    short input = 0;
    printf("Your balance is $%hi\n", balance);
    printf("Would you like to (1) Deposit or (2) Withdraw: ");
    scanf("%hi", &choice);
    switch(choice) {
        case 1:
            printf("How much would you like to deposit? ");
            scanf("%hi", &input);
            balance = balance + input;
        break;
        case 2:
            printf("How much would you like to withdraw? ");
            scanf("%hi", &input);
            balance = balance - input;
        break;
        default:
        break;
    }
    printf("Your balance is $%hi\n", balance);
}
```

TERMINAL PROBLEMS

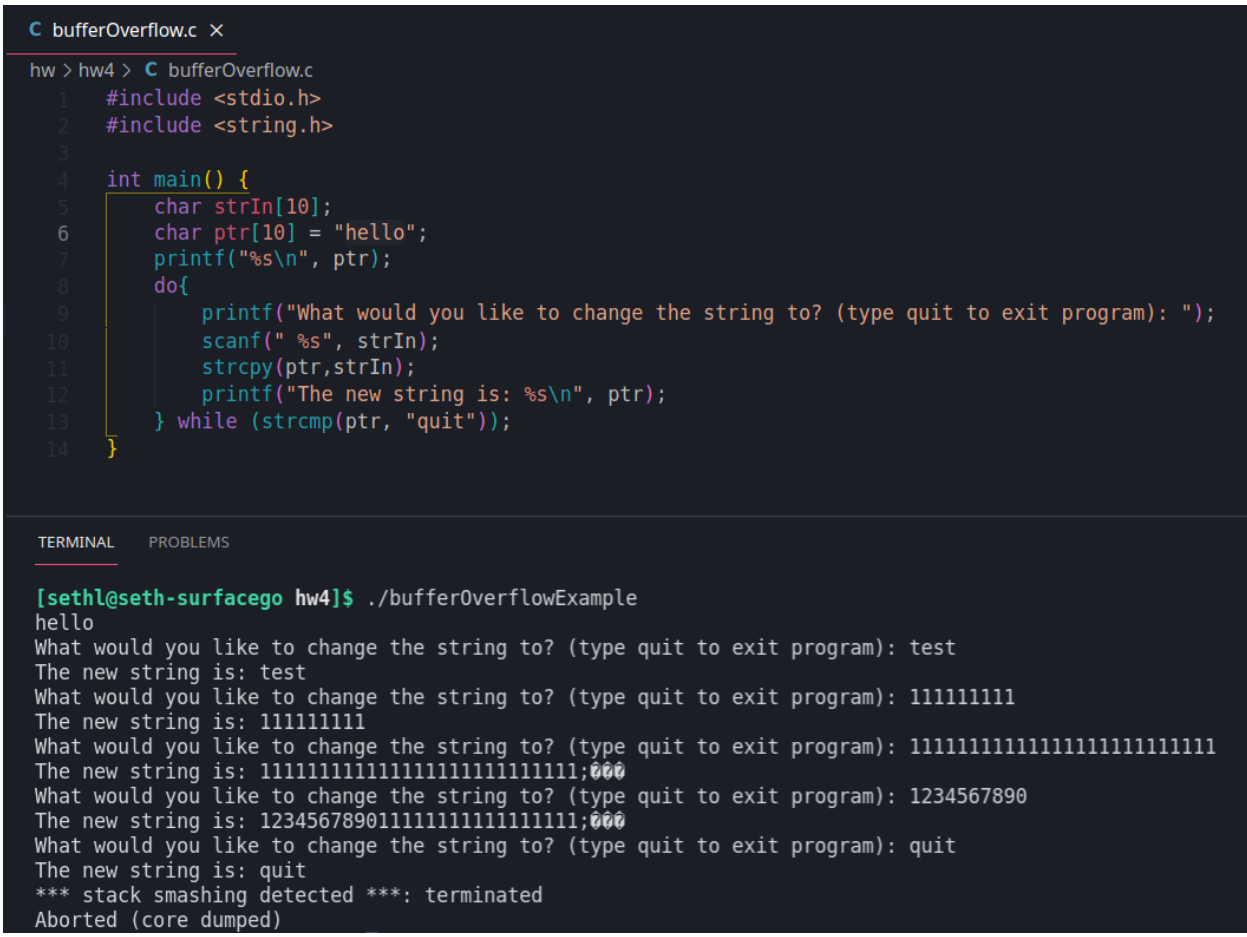
```
[sethl@seth-surfacego hw4]$ ./integerOverflowExample
Your balance is $-32768
Would you like to (1) Deposit or (2) Withdraw: 1
How much would you like to deposit? 32000
Your balance is $-768
[sethl@seth-surfacego hw4]$ ./integerOverflowExample
Your balance is $-32768
Would you like to (1) Deposit or (2) Withdraw: 2
How much would you like to withdraw? 1
Your balance is $32767
```

i. In this example, the user “withdraws” past the limit, and the program loops their balance back to the max positive number.

6. (10 points) Investigate the details of the **buffer overflow attack**, how it works. Then experiment with implementing this attack against a suitably vulnerable test program (write a piece of code

which has buffer overflow vulnerability, take a screenshot, and show me you successfully exploited a buffer overflow vulnerability).

a. A buffer overflow attack occurs when an input exceeds the size of the memory allocated for it and is written into the subsequent memory. If the attacker could, say, write into the program counter, they could change where the program goes next.

b. 

```
C bufferOverflow.c ×
hw > hw4 > C bufferOverflow.c
1   #include <stdio.h>
2   #include <string.h>
3
4   int main() {
5       char strIn[10];
6       char ptr[10] = "hello";
7       printf("%s\n", ptr);
8       do{
9           printf("What would you like to change the string to? (type quit to exit program): ");
10          scanf(" %s", strIn);
11          strcpy(ptr,strIn);
12          printf("The new string is: %s\n", ptr);
13      } while (strcmp(ptr, "quit"));
14  }

TERMINAL  PROBLEMS

[sethl@seth-surfacego hw4]$ ./bufferOverflowExample
hello
What would you like to change the string to? (type quit to exit program): test
The new string is: test
What would you like to change the string to? (type quit to exit program): 111111111
The new string is: 111111111
What would you like to change the string to? (type quit to exit program): 11111111111111111111111111
The new string is: 11111111111111111111111111;
What would you like to change the string to? (type quit to exit program): 1234567890
The new string is: 1234567890111111111111111111;
What would you like to change the string to? (type quit to exit program): quit
The new string is: quit
*** stack smashing detected ***: terminated
Aborted (core dumped)
```

   i. In this example, the strcpy function allows me to write into memory past the limits of my ptr[10] array. This has poor security as it allows me to write to memory I shouldn't have access to.

7. (5 points) Explain why genetic diversity is a good principle for secure development. Cite an example of lack of diversity that has had a negative impact on security.
   a. Genetic diversity is good because it means even if an attacker finds an exploit that works on one system, it won't be easily spread to other systems.
   b. The Spectre attack is one that comes to mind here. According to an Ars Technica article, Spectre "got its name for its abuse of speculative execution, a feature in virtually all modern CPUs". I don't know a lot about the technical details, but since "virtually all modern CPUs" use this system, this had a very wide effect.

8. (5 points) List three controls that could be applied to detect or prevent off-by-one errors.
   a. Use a for-each loop if the programming language supports it, instead of manually iterating through an array

b. Remember that arrays generally start at array[0], and double-check code to make sure it accounts for the last member of the array being array[length - 1].

c. Run tests on your code, paying close attention to boundary cases, before putting it into production.