University of San Francisco

CS686: Adv. Topics in Computer Security
Exam Solution

Student's Name: _____

Student's ID: _____

Professor: Dr. Vahab Pournaghshband

Question	Score
Problem 1 (1.5)	
Problem 2 (2)	
Problem 3 (1.5)	
Problem 4 (1)	
Problem 5 (2)	
Problem 6 (2)	
Total	

Q.1) Security Principles

1.5 points

Select the best answer:

a) Select the best answer: A company requires that employees change their work machines' passwords every 30 days, but many employees find memorizing a new password every month difficult, so they either write it down or make small changes to existing passwords. Which security principle does the company's policy violate?

A. Defense in depth

B. Ensure complete mediation

C. Consider human factors

D. Fail-safe defaults

Solution: C. Here is an article that discusses why password rotation should be phased out in practice, if you're interested in reading more.

b) Select the best answer: In the midst of a PG&E power outage, Carol downloads a simple mobile flashlight app. As soon as she clicks a button to turn on the flashlight, the app requests permissions to access her phone's geolocation, address book, and microphone. Which security principle does this violate?

A. Security is economics

B. Least privilege

C. Separation of responsibility

D. Design in security from the start

Solution: B. A flashlight application does not actually need these permissions in order to execute its functionality. It is over-permissioning its access to sensitive resources, violating the principle of least privilege.

c) Select the best answer: A security consultant is hired by a high school's principle and notices that a single admin password provides access to all of the school's funds and advises the principal that this is dangerous. What principle would the consultant argue the school is violating?

A. Don't rely on security through obscurity
B. Design security in from the start
C. Separation of responsibility
D. Fail-safe defaults

Solution: C.

For parts d-e, let's think about how fare gates are implemented in subway systems around the world. Select the security principle most relevant to each story.

d) In Staten Island, there are only fare gates at the first stop. Some passengers avoid paying by walking to the second stop and getting on the train there.

A. Security is economics
B. Least privilege
C. Separation of responsibility
D. Defense in depth
E. Consider human factors
F. Ensure complete mediation
G. Know your threat model
H. Detect if you can't prevent
I. Don't rely on security through obscurity
J. Design security in from the start

Solution: F. This is a failure of ensuring complete mediation; not all ways to enter the system are equally secured.

e) London's system was built and originally operated by many different companies. Even though the systems have been unified now, it's still hard to standardize fare gates at all the stations

A. Security is economics
B. Least privilege
C. Separation of responsibility
D. Defense in depth
E. Consider human factors
F. Ensure complete mediation
G. Know your threat model
H. Detect if you can't prevent
I. Don't rely on security through obscurity
J. Design security in from the start

Solution: J. Security was not designed in from the start, which makes it difficult to build security on top of an existing system.

f) Caltrain doesn't have fare gates, but the conductor on the train will occasionally check that you have a valid ticket.

A. Security is economics
B. Least privilege
C. Separation of responsibility
D. Defense in depth
E. Consider human factors
F. Ensure complete mediation
G. Know your threat model
H. Detect if you can't prevent
I. Don't rely on security through obscurity
J. Design security in from the start

Solution: H. Detect if you can't prevent, Even though there are no fare gates to prevent passengers from taking the train without a ticket, they try to detect passengers without tickets by checking tickets sometimes.

Q.2) Hash Functions 2 points

Bob proposes two new hash functions, E and B:

$$E(x) = H(x_1, x_2, \ldots, x_{M-1})$$

$$B(x) = H(x_1, x_2, \ldots, x_M \| 0)$$

where H is a preimage-resistant (one-way) and collision-resistant hash function, $x = x_1, x_2, \ldots, x_M, x_i \in \{0, 1\}$ and $\|$ denotes concatenation.

In other words, $E(x)$ calls H with the last bit of x removed, and $B(x)$ calls H with a 0 bit appended to x.

a) Is $E(x)$ preimage-resistant? Provide a counter-example if it is not.

A. Yes

B. No

Solution: A.

b) Is $E(x)$ collision-resistant? Provide a counter-example if it is not.

A. Yes

B. No

Solution: B. $E(x)$ is preimage-resistant. Suppose not, i.e., given $E(x)$ we could find an x' such that $E(x) = E(x')$. We will argue this means that H is not preimage-resistant, either. Suppose we are given $H(y)$. Let $x = y_0$, so that $E(x) = H(y)$. By assumption, we can find x' such that $E(x) = E(x')$. Let $y' = x'_1 \ldots x'_{M-1}$. Then it follows that $H(y) = E(x) = E(x') = H(y')$, so given $H(y)$ we can find y' such that $H(y) = H(y')$. This implies that H is not preimage resistant. That is a contradiction, so our assumption that E was not preimage-resistant must have been wrong.

$E(x)$ is not collision-resistant. Counter example: E(1...010) = E(1...011)

c) Is $B(x)$ preimage-resistant? Provide a counter-example if it is not.

A. Yes

B. No

Solution: A.

d) Is $B(x)$ collision-resistant? Provide a counter-example if it is not.

A. Yes

B. No

Solution: A. $B(x)$ is preimage resistant, using the same reasoning as E(x). (If there is an attack B's preimage resistance, then we can construct an attack against H's preimage-resistance that succeeds half as often, which is often enough to show that H is not preimage-resistant - but we werepromised that H is preimage-resistant, so it follows that B must be preimage-resistant, too.)

$B(x)$ is collision-resistant. If $B(x)$ was not collision resistant, then we can find $x \neq y$ such that $B(x) = B(y)$. This can be rewritten as $H(x\|0) = H(y\|0)$. Letting $x' = x'\|0$ and $y' = y'\|0$, this means we found $x' \neq y'$ such that $H(x') = H(y')$, which proves that $H()$ is not collision-resistant, which is a contradiction, Thus $B(x)$ must be collision-resistant.

Q.3) Web Security 1.5 points

califlower.com decides to defend against CSRF attacks as follows:

1. When a user logs in, **califlower.com** sets two 32-byte cookies session_id and csrf_token randomly with domain **califlower.com**.
2. When the user sends a POST request, the value of the csrf_token is embedded as one of the form fields.
3. On receiving a POST request, **califlower.com** checks that the value of the csrf_token cookie matches the one in the form.

Assume that the cookies don't have the secure, HTTPOnly, or Strict flags set unless stated otherwise. Assume that no CSRF defenses besides the tokens are implemented. Assume every subpart is independent.

a) Suppose the attacker gets the client to visit their malicious website which has domain **evil.com**. What can they do?

A. CSRF attack against **califlower.com**
B. Change the user's csrf_token cookie
C. Learn the value of the session_id cookie
D. None of the above

Solution: D. The attacker's website is of a different domain so they are not able to change/read any cookies for califlower.com. As such, they not able to execute a CSRF attack since they can't guess the value ofcsrf_token.

b) Suppose the attacker gets the client to visit their malicious website which has domain **evil.califlower.com**. What can they do?

A. CSRF attack against **califlower.com**
B. Change the user's csrf_token cookie
C. Learn the value of the session_id cookie
D. None of the above

Solution: ABC. Since the attacker's website is a subdomain for califlower.com, it can read/set cookies. The attacker can embed Javascript in their page to extract csrf_token and form a malicious POST request.

c) Suppose the attacker gets the client to visit a page on the website **xss.califlower.com** that contains a stored XSS vulnerability (the website **xss.califlower.com** is not controlled by the attacker). What can they do?

A. CSRF attack against **califlower.com**

B. Change the user's csrf_token cookie

C. Learn the value of the session_id cookie

D. None of the above

Solution: ABC. Utilizing the XSS vulnerability, the attacker can extract the csrf_token cookie and cause the user's browser to make a malicious POST request.

d) Suppose the csrf_token and session_id cookies have the HTTPOnly flag set. Suppose the attacker gets the client to visit a page on the website **xss.califlower.com** (the website **xss.califlower.com** is not controlled by the attacker) that contains a stored XSS vulnerability. What can they do?

A. CSRF attack against **califlower.com**

B. Change the user's csrf_token cookie

C. Learn the value of the session_id cookie

D. None of the above

Solution: D. The HTTPOnly flag renders the XSS attack useless for a CSRF attack since Javascript can't extract the value of csrf_token or session_id.

e) Suppose the attacker is on-path and observes the user make a POST request over HTTP to **califlower.com**. What can they do?

A. CSRF attack against **califlower.com**

B. Change the user's csrf_token cookie

C. Learn the value of the session_id cookie

D. None of the above

Solution: ABC. The attacker can observe session_id and csrf_token in plaintext and forge a POST request. Also, they can spoof a response to the POST request, and include a Set-Cookie header in the response to change the csrf_token cookie.

f) Suppose the attacker is a MITM and observes the user make a POST request over HTTPS to **califlower.com**. What can they do?

A. CSRF attack against **califlower.com**

B. Change the user's csrf_token cookie

C. Learn the value of the session_id cookie

D. None of the above

Solution: D. Nothing, a MITM can't break learn/change the cookie values without breaking TLS.

Q.4) Transport Layer Security

1 point

a) TRUE or FALSE: By default, in a TLS connection, both the server and client are authenticated to each other.

A. TRUE

B. FALSE

Solution: B. False. TLS only authenticates the server by default.

b) TRUE or FALSE: TLS has end-to-end security, so it is secure against an attacker who steals the private key of the server.

A. TRUE

B. FALSE

Solution: B. False. An attacker who's stolen the private key of the server could impersonate theserver to the victim.

c) TRUE or FALSE: An on-path (MITM) attacker can learn the request parameters of a GET request loaded over HTTPS.

A. TRUE

B. FALSE

Solution: B.

d) Consider a modified version of Diffie-Hellman TLS where the server does not include the signature when sending $g^a \bmod p$. Does this version of TLS provide confidentiality against a MITM? Explain.

Solution: Without the signature, we lose protection against a Man-in-the-Middle since we are now effectively using standard Diffie-Hellman key exchange. An adversary can now pretend to be the client to the server and pretend to be the server to the client, and therefore we lose all security guarantees because we now trust the Man-in-the-Middle.

Q.5) Encryption

2 points

Alice wants to create a secure channel of communication with a server. Alice knows that the best way of communicating is to somehow end up with a shared, symmetric key, but has no idea how this process works.

Assume that there exists a certificate authority (CA) actively sending certificates to many clients. Assume that Mallory, a Man-in-the-Middle attacker, and Eve, an eavesdropper, can both exist in all communication channels for all subparts unless otherwise specified.

Notation: PK: Public Key, SK: Private key.

a) Alice first wants to authenticate the CA before authenticating the server. She remembers that certificates provide authenticity, so she exchanges the following messages with the CA:

1. Alice queries the CA for the CA's public key and receives PK_{CA}.
2. Alice queries the CA for the server's public key and receives $\{\text{The server's public key is } PK_S\text{"}\}_{SK_{CA}}$.

Can Mallory trick Alice into accepting a different public key PK' of Mallory's choosing as the server's public key without being detected? **Explain**.

A. Yes

B. No

Solution: A. A MITM attacker could be present within the Alice-CA channel and therefore provide a public key (PK_{CA}) that is different from the CA's actual public key, to Alice. Since there is no root of trust, there is no way for Alice to know that she received an incorrect key.

For the rest of this question, assume that instead of querying the CA for their public key, Alice has the CA's correct public key, PK_{CA}, hardcoded into her computer.

b) When Alice queries the CA for the server's public key, the CA sends $\{\text{The server's public key is } PK_S\}_{SK_{CA}}$.

Can Mallory trick Alice into accepting a different public key PK'_S, **not necessarily of Mallory's choosing**, as the server's public key without being detected?

If you mark "Yes", provide an attack that would accomplish this goal. If you mark "No", explain why not in 2 sentences or fewer.

A. Yes

B. No

Solution: A. Mallory can replay any certificate that has been sent over the channel previously.Two potential examples include a certificate of a different server that has previously been sent over the channel by the CA, or sending an old public key of this server.

c) Alice has received some public key PK_S from the CA, but she doesn't trust that PK_S belongs to the server. Which of the following messages can the server send to convince Alice that she is talking to the legitimate server? Select all that apply.

A. The server sends $H(PK_S)$

B. The server sends $Sign(SK_S, H(PK_S))$

C. The server sends $H(SK_S||PK_S)$

D. The server randomly generates a symmetric key K and sends $Sign(SK_S, K), HMAC(K, PK_S))$

E. None of the above

Solution: E. This question is an issue of the server trying to authenticate themselves to Alice. Ifthe server sends $H(PK_S)$, it doesn't authenticate themselves to Alice since any MITM could replace the hash with their own hash. If the server signs the hash of their public key, since Alice still does not have the server's actual public key, this doesn't work either. For example an attacker who has sent Alice PK'_S could replace the signature with Sign$(SK'_S, H(PK'_S))$Finally, if the server generates a symmetric key, the same issue (and attack) with option 2 still applies.

d) Alice wants to try a modified version of the Diffie-Hellman key exchange which is as follows:

- Alice sends $(g^a, Sign(SK_A, g^a))$
- The server sends $(g^s, H(g^s))$
- The shared key is derived as $K = g^{as}$

Can Mallory trick the either Alice or the server into deriving a key that is different from the one they would have derived if Mallory had not existed?

If you mark "Yes", provide an attack that would accomplish this goal. If you mark "No", explain why not in 2 sentences or fewer.

A. Yes

B. No

Solution: A. Alice signs her section of the exchange, but the server doesn't, meaning that a MITM attacker could replace whatever the server sends to be $(g^{s'}, H(g^{s'}))$ for an s' of the attacker's choosing.

Q.6) Memory Safety

2 points

The following code allows you to print characters of your choice from a string. It runs on a 32-bit x86 system with **stack canaries enabled**, but no other memory defense methods in use. Assume local variables are pushed onto the stack in the order that they are declared, and there is no extra padding saved registers, or exception handlers. Note that `scanf("%d", &offset)` reads a number from the input, converts it to an integer, and stores it in the offset variable.

```
 1   void foo() {
 2       char buf[300];
 3       gets(buf);
 4   }
 5
 6   int main() {
 7       char *ptr;
 8       int offset = 0;
 9       char important[12] = "sEcuRitY!!!";
10       while (offset >= 0) {
11           scanf("%d", &offset);
12           ptr = important + offset;
13           printf("%c\n", *ptr);
14       }
15       foo();
16       return 0;
17   }
```

a) Draw the stack, when at the point in time when line 12 of the code is executing, by filling in the diagram below. Label the location of sfp, rip (saved return address), stack canary, and the ptr, offset, and important variables, for main's stack frame. Each empty box represents 4 bytes of stack memory. If a value spans multiple boxes, label all of them.

Solution:

rip
sfp
canary
ptr
offset
important
important
important

b) Someone informs you that this code contains a vulnerability which leaks the value of main's stack canary. Which sequence of inputs to the program would leak this information? Fill in the blanks below.

_____\n _____\n _____\n _____\n

Solution: Since bounds aren't checked, use ptr to read off the stack canary: 20\n 21\n 22\n23\n

c) Next, suppose you want to develop a reliable arbitrary-code-execution exploit that works by overwriting foo's entire return address, so that when foo returns, your shell code will be executed. You first supply the string from part (b) to learn the value of the stack canary, followed by the string '-1\n', followed by a carefully chosen third string of some length. Write the **minimum** possible length of the third string, to achieve this. Assume your shell code is 100 bytes long and it cannot be shortened.

Solution: 312 bytes. The stack frame for foo looks like

rip
sfp
canary
buf
...
buf

We're going to overflow buf, so we need 300 bytes for buf, plus 4 bytes for foo's canary, plus4 bytes for foo's sfp, plus 4 bytes for overwriting foo's rip. We can store the shell code within buf, so we don't need another 100 bytes for it. Notice that the canary is the same for every function, so after learning main's canary, we know that the same value will be used for foo as well.

313 is correct, in case you assume that you need a newline at the end (gets doesn't actually require a newline-you could actually omit the newline and substitute it with end of file).

d) The developers propose to fix the program by replacing lines 12-13 with the following code. Fill in the blank inside the if-statement to make the fix correct.

```
12    if (_____________________________________){
13        ptr = important + offset;
14        printf("%c\n", *ptr);
15    }
```

Solution:

```
0 <= offset && offset < 12
or
0 <= offset && offset < siezeof(important)
```