Raluca Ada Popa
Spring 2018

CS 161
Computer Security

Final Exam

PRINT your name: ______________________, ______________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct on this exam will lead to a "F"-grade for the course and that the misconduct will be reported to the Center for Student Conduct.

SIGN your name: ______________________

PRINT your class account login: `cs161-`______________________and SID: ______________________

Name of the person sitting to your left: ______________________

Name of the person sitting to your right: ______________________

Please read the following before starting the exam.

- You may consult three double-sided sheets of notes (or six single-sided sheets).
- You may not consult other notes, textbooks, &c. Calculators, computers and other electronic devices are not permitted without prior accomodation.
- Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page unless we are clearly told to look there.
- Before you turn in your exam, write your Student ID at the top of every page.
- Bubble every item completely! Avoid using checkmarks, Xs, writing answers on the side, &c. If you want to unselect an option, erase it completely and clearly.
- For questions with circular bubbles, you may select only one choice.
  - ○ Unselected option (completely unfilled)
  - ● Only one selected option (completely filled)
- For questions with square checkboxes, you may select any number of choices (including none or all).
  - ■ You can select
  - ■ multiple squares (completely filled).
- We reserve the right to deduct points from exams which do not follow the above directions. (Of course, we will make reasonable exceptions.)
- You have 170 minutes. There are 12 questions, of varying credit (0 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

SID: ______________________________

Problem 1 ***True or False*** **(0 points)**

Answer the following true or false questions.

(a) The Harvard Architecture separates the code and data of a program into two separate address spaces. This makes it impossible to treat data as code, or code as data. TRUE or FALSE: Buffer overflows are not exploitable on a Harvard Architecture.

○ TRUE ● FALSE

> **Solution:**
>
> False, as a buffer overflow could still overwrite local variables.

(b) TRUE or FALSE: Postconditions for a function are independent of implementation details of the function.

● TRUE ○ FALSE

> **Solution:**
>
> True, the postcondition deals with the final result of the function, not about how the function gets there.

(c) Consider the splitting problem: given a natural number n, find integers a and b such that $ab = n$ and $a, b > 1$. (Note that a and b need not be prime.) TRUE or FALSE: If we had a polynomial time solution to the splitting problem, we could create a polynomial time solution for factoring into *prime numbers*.

● TRUE ○ FALSE

> **Solution:**
>
> True, since we can factor a number by splitting it once, and then splitting a and b. (There is some work associated to show that this is still polynomial time.)

(d) Consider the function $H : \{0,1\}^{2b} \rightarrow \{0,1\}^b$. H takes a $2b$-bit string s, and splits it into two b-bit blocks k and x. It then computes $E_k(x)$, where E_k is a secure block cipher encryption using b-bit blocks and keys. TRUE or FALSE: H is a one-way function.

○ TRUE ● FALSE

> **Solution:**
>
> Let y be any arbitrary output. Then $H(k'||D_{k'}(y)) = y$ for any k'.

(e) TRUE or FALSE: If an arbitrary function is collision-resistant, then it is preimage resistant.

○ TRUE ● FALSE

Solution:

Consider the identity function. Then the function is collision-resistant (there are no two inputs such that $\mathrm{Id}(x) = \mathrm{Id}(y)$ with $x \neq y$), but it is clearly not preimage resistant (given y, its inverse is simply y).

(f) TRUE or FALSE: If an arbitrary function is collision-resistant, then it is second preimage resistant.

● TRUE ○ FALSE

Solution:

Say that we had a function f was not second preimage resistant. Then for any x we can find another preimage $x \neq x'$ with $f(x) = f(x')$. Therefore f is not collision-resistant.

(g) TRUE or FALSE: If it is possible, one way to prevent CSRF attacks is to use HTTP POST requests instead of HTTP GET requests, since this prevents an attacker from creating a request using an `img` tag.

○ TRUE ● FALSE

Solution: An attacker can still write Javascript to submit a POST request.

(h) TRUE or FALSE: If we do not use frames on our site, this prevents attackers from performing a clickjacking attack.

○ TRUE ● FALSE

Solution: A clickjacking attack often involves when another site frames you.

(i) TRUE or FALSE: If you know the IP addresses, ports, TCP sequence numbers and TCP acknowledgement numbers in a TCP connection, you can inject TCP traffic.

● TRUE ○ FALSE

Solution: TCP provides no authentication – this is all the information you need to spoof messages from either side.

(j) TRUE or FALSE: If we have two independent detectors, there is always a way to combine them such that the combined detector is more cost-effective than either detector alone. (Ignore the cost of the detectors.)

○ TRUE ● FALSE

Solution:

Intuitively: you don't know which detector is right so you might not be able to combine them to be more effective.

As a pathological counterexample: say we have a detector that always triggers and another detector that never triggers. Say the cost of a false-positive is $100, and the cost of a false-negative is $100. Furthermore say that the base-rate of attacks is 0. The detector that never triggers is more cost-effective, since it costs $0. Any combination of it with the detector that always triggers must be less cost-effective.

(k) TRUE or FALSE: HTTPS provides security even against attackers on the local network.

● TRUE ○ FALSE

Solution:

HTTPS provides end-to-end security.

(l) TRUE or FALSE: Even if you carefully inspect all URLs in the address bar to make sure they do not contain Javascript, you can still fall victim to a reflected XSS attack.

● TRUE ○ FALSE

Solution:

Example: you click an innocent URL, which contains a iframe with the reflected XSS as the source.

(m) TRUE or FALSE: Disabling Javascript in your browser prevents clickjacking attacks completely.

○ TRUE ● FALSE

Solution: There are CSS-based clickjacking attacks. Also, browser-in-browser still works.

(n) TRUE or FALSE: Disabling Javascript in your browser prevents CSRF attacks completely.

○ TRUE ● FALSE

Solution:

False, a CSRF attack can be performed by simply including the URL in the src tag of an image, as shown in HW4.

(o) TRUE or FALSE: If we hash the MAC of a message, this provides confidentiality.

○ True ● False

Solution:

False, as neither MACs nor hashes are designed for confidentiality. (As a simple example: both are deterministic and so sending the same message twice leaks.)

(p) True or False: If we have a secure MAC, it is computationally difficult to find two keys k and k', and a message m such that $\text{MAC}_k(m) = \text{MAC}_{k'}(m)$.

○ True ● False

Solution:

False, this is not a property that MACs aim to provide. In fact we can define a pathological MAC which does not have this property. For example, consider $\text{NEW-MAC}_{0||k}(m) = \text{NEW-MAC}_{1||k} = \text{HMAC}_k(m)$. Then this MAC is secure (in the unforgeability sense), but the given statement is false.

(q) True or False: Prepared statements are a possible defense for SQL injection attacks.

● True ○ False

Solution: This is the main purpose of prepared statements (in addition to improved efficiency).

(r) True or False: DNSSEC and TLS combined prevents eavesdroppers from seeing what sites we are visiting.

○ True ● False

Solution: False, DNSSEC does not provide confidentiality.

(s) True or False: ARP spoofing requires that the attacker has two devices on the network: one to send requests and the other to give fake answers.

○ True ● False

Solution: False, they only need one to give fake answers.

(t) True or False: Whitelist approaches are often more effective than blacklists in preventing injection attacks.

● True ○ False

Solution: Whitelisting is less error-prone.

SID: ____

Problem 2 ***Party, No Theme*** **(0 points)**

Answer the following questions about various course topics.

(a) (6 points) You have discovered a vulnerability in Snapitterbook which lets you create malicious posts. Whenever someone visits your Snapitterbook page, the evil post sends a request to the Snapitterbook webserver which causes the visitor to post a copy of the evil post to their own wall. Now their wall is also infected!

i. Which of the following concepts are relevant to this situation?

- ☐ CSRF
- ☐ Virus
- ■ Worm
- ☐ Trojan
- ☐ Reflected XSS
- ☐ Clickjacking
- ☐ SQL Injection
- ■ Stored XSS

Solution:

This is a worm (similar to the MySpace Sammy Worm) since it propogates by copying itself to another target. It is also a stored XSS, as that is the exploit it uses to make evil requests.

Note that this is *not* a CSRF: the origin is the same as the referring site.

ii. Which of the following technologies could help fix or detect the situation above?

- ☐ A strict X-Frame-Options
- ■ Input escaping
- ■ A strict Content-Security-Policy
- ☐ Prepared Statements
- ■ Anomaly-based detection
- ☐ Referer checking
- ☐ CSRF Tokens
- ☐ HTTPS

Solution:

Input escaping is an easy fix: if inputs were escaped then XSS could not occur.

A strong CSP would prevent inline and foreign scripts, preventing XSS from executing.

Anomaly-based detection could help detect this problem through the large volume of posts, or through the anomalous referers of HTTP requests to the post endpoint.

(b) (3 points) You are considering buying three detector solutions, with the following statistics:

1. Detector X: False positive rate 5%; False negative rate 2%
2. Detector Y: False positive rate 1%; False negative rate 5%
3. Detector Z: False positive rate 2%; False negative rate 1%

A false positive costs \$50, while a false negative costs \$100.

Answer the following questions which attempt to compare the cost effectiveness of each detector. If the detectors are equally effective, either choice will be accepted.

i. Compare X and Y.

○ X is better (or equal) ○ Y is better (or equal) ● Cannot say

ii. Compare Y and Z.

○ Y is better (or equal) ○ Z is better (or equal) ● Cannot say

iii. Compare X and Z.

○ X is better (or equal) ● Z is better (or equal) ○ Cannot say

Solution:

This is the base rate fallacy: without knowing how often attacks occur, you cannot determine which detector is better. The exception is X vs. Z, since Z has a lower false positive and false negative rate, it must be more effective.

(c) (3 points) What security principle explains why using proof of work to prevent email spam might work?

Solution: Security is economics: a spammer who has to spend money (as proof of work) in order to send spam is less incentivized to send spam.

(d) (3 points) A company decides to implement a complicated password policy for its employees. What security principle explains why the overall security of the system might go down?

Solution: Consider human factors: people will often write their passwords on post-it notes if they are too complicated.

(e) (3 points) For RSA signatures as discussed in lecture, why is verification faster than signing?

Solution: Verification requires raising to the 3rd power, can be done fast ($O(1)$ multiplications). Signing requires raise to the d, which is bigger than 3 and therefore slower ($O(\log d) = O(\log N)$ multiplications).

(f) (5 points) Bob allegedly posted a rude statement about Alice on `https://bob.com/alice-sux`. Alice decides to take Bob to court! As proof that the Bob's site had the statement at some point in time, Alice presents the entire HTTPS dialogue between her and the site. She also provides all the keys derived through the process. Should the judge be convinced? Explain your answer in 2–3 sentences.

Ignore the possibility that an attacker has compromised `bob.com` or Bob's private keys. Assume `bob.com` uses RSA TLS and has a certificate signed by a trusted certificate authority.

○ Judge should be convinced ● Judge should not be convinced

Explain:

Solution:

After Alice and Bob agree on a PS, both sides derive the TLS *symmetric* keys I_b, I_s, C_b, C_s. This means that Alice can easily forge messages which appear to come from Bob.

(g) (3 points) Which of the following attacks require an attacker to be on the same local network as their target?

- [ ] TCP Injection
- [x] ARP Spoofing
- [ ] DNS Spoofing
- [x] DHCP Spoofing
- [ ] Reflected XSS
- [ ] Stored XSS

Solution:

Both ARP and DHCP spoofing require that the attacker is on the same local network, since neither protocol actually leaves the local network.

(h) (3 points) Which property of the hash function does the hash chain in Bitcoin rely on? List one property alone.

Solution: Collision resistance: must prevent attackers from creating two blocks with the same hash.

(i) (3 points) Which of the following defenses are typically implemented using the compiler?

- [x] Position-Independent Executables
- [ ] NX bit
- [ ] ASLR
- [x] Stack Canaries

Solution:

PIE requires the compiler to output position-independent code which can execute in any location in memory.

Stack canaries are added by the compiler: it emits code to check that the canary does not change before leaving the function.

ASLR is implemented at the OS level, the NX bit is either implemented at the OS level or the hardware level.

(j) (3 points) Alice receives the following email:

```
From: Mallory <mallory@rsa.com.evil.org>
To: Alice <alice@rsa.com>
Hey Alice,

Your boss, Steven, wanted me to send you this link to those expense
reports for the Fall 2016 Quarter. He said that you would look at
them and give Evelyn the tax estimate she asked for earlier.
<http://taxes.fall.2016.rsa.com.evil.org/login.html>
```

What attack does this represent? (Be as specific as possible!)

Solution: Spearphishing: the attack contains specific information about Alice, but the site `rsa.com.evil.org` is trying to steal her login information. (Significant partial credit for phishing or social engineering.)

Problem 3 ***Déjà Vu*** **(0 points)**

The code below runs on a 32-bit Intel architecture. **ASLR is enabled**. There are no stack canaries, no position-independent executables and no NX bit. No padding is added by the compiler.

```
char *gets(char *s) { /* simple implementation of gets */
    char *s_ = s;
    while ((*s_++ = getchar()) != '\n');
    s_[-1] = '\0';
    return s;
}

void deja_vu() {
    char door[16];
    gets(door);
}
```

(a) What sort of exploit technique works by chaining execution of small blocks of code ("gadgets")?

Solution: Return-oriented programming

(b) After disassembling the code, you find the following gadgets.

```
Gadget 1:
   0x080484a2 <+30>:  sub    $0x14,%esp
   0x080484a5 <+33>:  ret
Gadget 2:
   0x080484fc <+30>:  add    $0x14,%esp
   0x080484ff <+33>:  ret
```

Which of the above gadgets was most likely generated intentionally by the compiler?

○ Gadget 1 ● Gadget 2

(c) On some older Intel processors, a single instruction could halt the entire system. The machine code for one such instruction is `\xf0\x0f\xc7\xc8`. Give an input which will cause the execution of this shellcode. Hint: We pushed `&door` onto the stack in order to call `gets`. This forms a "perfect pointer" for shellcode. Knowing that, how can you make the program start executing `door`?

Solution:

Main idea: earlier, we pushed `&door` onto the stack for the call to `gets`. We juggle `%esp` so that it points to `&door` right before a `ret`.

One possible input (with newlines for clarity):

- F00F Instruction: `\xf0\x0f\xc7\xc8`
- ret: `\xa5\x84\x04\x08`
- ret: `\xa5\x84\x04\x08`
- subtract 20 from esp, ret: `\xa2\x84\x04\x08`
- padding to overwrite ebp: `AAAA`
- subtract 20 from esp, ret: `\xa2\x84\x04\x08`

Other solutions exist.

Problem 4 ***Mystery Matrix Math*** **(0 points)**

Consider the following code which performs a mystery operation on the input matrix.

```
void mystery(int **A, size_t m, size_t n) {
    for (size_t i = 0; i < m; i++) {
        for (size_t j = 0; j < n; j++) {
            A[i][j] = A[j][i];
        }
    }
}
```

Assume that m and n are both non-zero. Ignore the possibility of negative indices.

(a) For each of the subparts below, mark necessary preconditions for `mystery` to be memory-safe.

i. ■ $A \neq$ `NULL`

□ `size`$(A) > n$

□ `size`$(A) > m$

■ `size`$(A) \geq n$

■ `size`$(A) \geq m$

Solution: Since we dereference A, we need to ensure that $A \neq$ `NULL`. Note that we access $A[i]$ from $0 \leq i < m$, so we need `size`$(A) \geq m$. We also access $A[j]$ from $0 \leq j < n$, so we need `size`$(A) \geq n$.

ii. □ $m \neq i, n \neq j$

□ $i, j < \max(m, n)$

□ $i < m$, $j < n$

□ $0 \leq i, j$

Solution: None of these are preconditions to the function, they are invariants.

iii. ■ $\forall i : i < m \implies A[i] \neq$ `NULL`

□ $\forall i \forall j : i < m \wedge j < n \implies A[i][j] \neq$ `NULL`

■ $\forall j : j < n \implies A[j] \neq$ `NULL`

□ $\forall i \forall j : i, j < \max(m, n) \implies A[i][j] \neq$ `NULL`

□ $\forall i : i < \min(m, n) \implies A[i] \neq$ `NULL`

Solution: We dereference $A[i]$ for $0 \leq i < m$, so we need to make sure these are not `NULL`. We also dereference $A[j]$ for $0 \leq j < n$, so we need to check those too.

Note that Option 3 is implied by Options 1 and 2, so it does not matter if you put it or not.

The other conditions are *not* valid, they check if the *integers* are null, which is not possible. (We only check if something is `NULL` when we are dereferencing it.)

iv. ■ $\forall i : i < m \implies$ `size`$(A[i]) \geq n$

□ $\forall j : j < m \implies$ `size`$(A[j]) \geq m$

□ $\forall i : i < n \implies$ `size`$(A[i]) \geq n$

■ $\forall j : j < n \implies$ `size`$(A[j]) \geq m$

Solution: This is just checking for valid accesses in the inner loop.

(b) For each of the following postconditions, indicate if they hold or not. Assume that the preconditions hold. Let A be the original matrix, and let A' be the transformed matrix.

i. All of the preconditions, applied to A'.

● True ○ False

Solution: All of the preconditions above apply to A'. All of the preconditions are about memory-safety, and we have not changed the allocation of A' itself.

ii. $\forall i : i < m \implies A[i] = A'[i]$

● True ○ False

Solution: This checks the pointers have not changed, which is true since (again) the allocation of A' has not changed.

Problem 5 ***Scared of Commitment*** **(0 points)**

Alice and Bob are playing a game which involves flipping a coin. To play, Alice begins by calling a side (Heads or Tails). Bob then flips the coin and announces the result to Alice. Alice wins if she correctly guesses the coin flip.

Alice and Bob do not trust each other. Alice is scared that Bob will lie about the result of the flip once he knows Alice's guess. Bob is scared that Alice will lie about her original guess once she knows the result of the flip. In order to solve this problem, we have Alice and Bob use a **commitment scheme**. A secure commitment scheme prevents both players from cheating the result of the coin flip.

Evaluate each of the proposed commitment schemes below, where a is a single bit representing Alice's guess and b is a bit representing the result of Bob's coin flip. Explain your answers.

You may assume that SHA256 is a random oracle. (Do not worry if you do not know what that means.)

(a)
1. Alice generates a random 128b string R.
2. Alice sends R and SHA256($a||R$) to Bob.
3. Bob sends b to Alice.
4. Alice reveals a.
5. Bob checks that SHA256($a||R$) equals the value Alice sent before.

○ Secure ● Insecure

Explain:

Solution: Insecure. Bob can try $a = 0$ and $a = 1$, and determine which one Alice chose.

(b)
1. Alice generates a random 128b string R.
2. Alice sends SHA256($a||R$) to Bob.
3. Bob sends b to Alice.
4. Alice sends a and R to Bob.
5. Bob checks that SHA256($a||R$) equals the value Alice sent before.

● Secure ○ Insecure

Explain:

Solution:

Secure. Once Alice has committed to $H(a||R)$, she cannot find another hash because the hash function is collision-resistant. Bob cannot find a because the hash function is one-way. (Technically we need the hash function to be a random oracle.)

(c)
1. Alice generates a 128b key k and a 127b random padding R.
2. Alice sends $C = E_k(a||R)$ to Bob, where E_k is the AES block cipher.
3. Bob sends b to Alice.
4. Alice reveals k and $a||R$.
5. Bob decrypts C to recover a and R, and checks that they match what Alice sent.

○ Secure ● Insecure

Explain:

Solution: Insecure. After Bob sends b, Alice can generate other keys k', and calculate decryptions of C until she gets one whose first bit is b. Say that the resulting plaintext is $M = b||R'$. Then at step 3, Alice can send k' and $b||R'$ and fool Bob.

Note that it is infeasible to fix b and R' in advance and then attempt to generate k' such that $AES_{k'}(b||R')$ matches.

Problem 6 ***Vaults*** **(0 points)**

A bank wants to set up a vault. They give all of their employees *badges*, which contain a microcontroller, a per-employee symmetric key k, the ElGamal public key K_V of the vault, and a password P which is needed to open the vault.

The bank knows that sending passwords in plaintext is bad, so they decide to encrypt it first. To send the password P to the vault, the badge performs the following protocol.

1. The badge uses the vault's ElGamal public key K_V to encrypt its employee symmetric key k.
2. The badge sends $E_{K_V}(k), \text{AES-CTR}_k(P)$.

The vault then decrypts the badge's message:

1. The vault decrypts $E_{K_V}(k)$ using its ElGamal private key. Since the key k is always 256 bits long, the vault ignores any excess bits (i.e., it takes the key $\bmod 2^{256}$).
2. The vault uses k to decrypt $\text{AES-CTR}_k(P)$.
3. If the decrypted password matches the correct password, the vault unlocks itself (if it is locked) or locks itself (if it is unlocked).

There is no limit to the number of times somebody can attempt to authenticate.

Mallory is an attacker with physical access to the vault who wants to break-in. Mallory manages to get a small "skimmer" device placed on the surface of the vault. The skimmer acts as a man-in-the-middle, which can spoof and intercept messages between a badge and the vault. The next day, the vault manager uses his badge to open the vault door.

(a) Explain how Mallory should program her skimmer in order to allow her to open and close the vault door again at some later time.

Solution: Simply replay the old ciphertext.

(b) Using your exploit in part (a), Mallory breaks open the vault. The room contains a post-it note with the password P. It also contains a flash drive of sensitive documents, encrypted with the vault manager's symmetric key k_m. Mallory needs to decrypt these documents!

i. ElGamal encryption is *malleable*: Given (1) the ElGamal encryption (c_1, c_2) of a message m and (2) an integer a, one can find the encryption of am without knowing m! Show how to do this. Recall that ElGamal encryption is $Enc(PK, m) = (g^r \mod p, m \cdot PK^r \mod p) = (c_1, c_2)$, for r chosen at random, and g, p public parameters. (No explanation needed. You can use this in later parts even if you do not show it here.)

ii. How can Mallory determine the least significant bit of the key k_m? Recall: From Mallory's skimmer, she knows $E_{K_V}(k_m)$ and $\text{AES-CTR}_{k_m}(P)$. From the post-it note, she knows P. Everyone knows the public key K_V of the vault. Hint: Mallory might have some success if she interacts with the vault again.

iii. Extend your attack above to determine the entire key k_m.

Solution:

Let $E_{K_V}(k_m) = (c_1, c_2)$ be the ElGamal ciphertext that the badge sends to the vault. Recall that we can generate an encryption of ak_m by sending (c_1, ac_2). Define $K_b = (c_1, 2^b c_2)$, i.e. the encryption of k_m shifted b bits to the left. Note that this makes the bottom b bits zero.

Mallory begins by sending sending $K_{255}, \text{AES-CTR}_{10\ldots00}(P)$. The key K_{255} has only the first bit set. If the vault door then open, Mallory knows that the least significant bit of the key k_m is 1. If remains locked, it was zero.

Mallory can repeat this process for each bit. For example, say that she determined before that the least significant bit of the key was 1. Then she can send $K_{254}, \text{AES-CTR}_{110\ldots00}(P)$. Again, if the vault opens, she knows the second least significant bit was 1. Otherwise, it was zero.

Using this, Mallory can wholly determine the key k_m.

This question is adapted from a real-world problem: https://arxiv.org/abs/1802.03367.

SID: ______________________________________

Problem 7 ***MS SQL*** **(0 points)**

Microsoft's MS SQL protocol allows a server and a client to negotiate an encryption protocol. Both sides begin the `PRELOGIN` stage by sending the one of the following flags in an unencrypted and unauthenticated format:

1. `NOT_SUPPORTED` (`NS`): "I do not support encryption. If you try to use it, I will abort this connection."
2. `SUPPORTED` (`S`): "I support, but do not require, encryption. Please use it if you support it."
3. `REQUIRED` (`R`): "I require encryption. If you cannot use it, I will abort this connection."

The server and client attempt to use the highest security that they can agree on. If they manage to agree on encryption, the rest of the conversation is encrypted using TLS with a server certificate, signed by a mutually trusted certificate authority. Assume that both sides of the connection properly implement the TLS protocol.

For each of the following scenarios below, determine if an active man-in-the-middle can compromise the confidentiality of the connection.

(a) Client sends `R`; Server sends `R`.

● Secure ○ Insecure

> **Solution:**
>
> Both sides require encryption, so TLS will be established and a man-in-the-middle cannot break the encryption.

(b) Client sends `S`; Server sends `S`.

○ Secure ● Insecure

> **Solution:**
>
> Insecure. A MITM can forge either side's message to say `NS`, therefore removing the encryption.

(c) Client sends `S`; Server sends `R`.

○ Secure ● Insecure

> **Solution:**
>
> Insecure. A MITM intercepts the server message saying `R`, and passes `NS` to the client. The MITM then performs the TLS handshake with just the server. It encrypts all the client's data to it, acting like a proxy.

(d) Client sends `R`; Server sends `S`.

● Secure ○ Insecure

> **Solution:**
>
> Secure. Note the attack described above does not work: the MITM cannot perform the TLS handshake with just the client because it does not the private key corresponding to the server certificate.

(e) Alice proposes a modification to the above protocol. Rather than sending its encryption level unauthenticated, the server will present a signed message of the encryption level it supports, along with its TLS certificate. The client verifies the certificate and the signed message. Reevaluate the above choices in this new protocol.

Assume the server never changes what encryption level it supports. Assume that all clients and servers (even if they do not support encryption) still follow Alice's protocol.

TRUE or FALSE: As long as both sides support encryption, the connection will be secure against a man-in-the-middle.

● TRUE ○ FALSE

Solution:

We only need to consider two cases above:

Client sends `S`; Server sends `S`: A MITM cannot change the server's message, so they must change the client's message. But if the attacker changes the client's message to read `NS`, the client will notice that the server is not using TLS (even though it supposedly supports it) and abort the connection.

Client sends `S`; Server sends `R`: A MITM cannot change the server's message, so the attack above does not work.

Problem 8 ***Fiat Tux (Let There Be DNS)*** **(0 points)**

You're in charge of manually running the DNS resolver for AirBears2. AirBears2 has a reputation for impeccable reliability among its users, and it's your job to make sure that nothing goes wrong on your end so that its users receive top-notch performance.

(a) You start up your DNS resolver for the first time, with no state in your cache. You do not currently make use of DNSSEC. You have the root name server's IP address hard-coded into your configuration (and nothing else). You receive your first DNS query from a user for `inst.eecs.berkeley.edu`. Where should you send your initial query?

> **Solution:** Your initial query goes to the hard-coded root server's IP address.

(b) You send your initial query and receive many responses, but only two of them have matching ID fields. You discard all responses that do not have matching ID fields, but you still have with two responses. The responses both contain records for the domain `.edu`, but the IP addresses in the A records are different. You also you notice that one of them arrived significantly before the other. Which of the answers should you trust?

○ Trust the faster one ○ Trust the slower one ● Trust neither

(c) Disregarding your decision in part (b), you choose to trust one of the two answers at random. You resolve the remainder of the requested domain name, caching all of the the records that you receive along the way, including any you may have received in part (b). For each remaining step, you receive a single response, and every response has a matching ID. Eventually, you receive the IP address of `inst.eecs.berkeley.edu`. You return this IP address to the user that requested it.

A few minutes later, you receive a complaint from this user claiming that when they attempted to visit `inst.eecs.berkeley.edu`, they instead received the webpage for `eecseecs.com`.

Assume that you are no longer under attack. What entries do you need to remove to ensure that future resolutions are safe? (You want to remove the minimal number of entries.)

● All records

○ Records ending with `.edu`

○ Records ending with `.berkeley.edu`

○ Records ending with `.eecs.berkeley.edu`

○ Records starting with `inst.`

○ Records for `inst.eecs.berkeley.edu`

○ Records ending with `.com`

○ Records for `eecseecs.com`

> **Solution:**
>
> You need to empty your cache completely, as the cached entries for `.edu`, `berkeley.edu`, and `eecs.berkeley.edu` could all be incorrect, as well as any potential additional records provided. Since you cached everything, any record in your cache could be poisoned.

(d) After the above incident, a disappointed Oski demands that you utilize DNSSEC. Assume that the rest of the world magically complies with your demands to support DNSSEC. All implementation details of DNS follow what was taught in lecture.

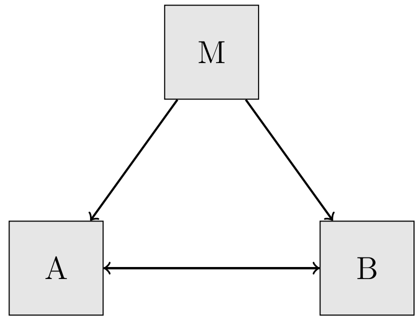
We replay the scenario up to part (b), greatly simplified, this time with DNSSEC. You start up your DNSSEC resolver for the first time, with no state in your cache. You have the root name server's IP address and public key hard-coded into your configuration (and nothing else).

You send your initial query and receive two responses. The responses both contain records for the domain `.edu`, but the IP addresses in the A records are different. At this given point in time, given zero tolerance for error, describe how you would determine which answer(s) to trust. If you cannot trust either answer, write **"cannot trust either"** and explain your answer briefly.

Solution:

You check for an RRSIG record over the A record for **.edu**, and if it exists, you validate it with the hard-coded public key of the root server. If the signature matches the provided A record, you trust that answer. If both signatures match, you trust both; if neither signature matches, you trust neither.

Problem 9 ***Networking Love Triangle*** **(0 points)**



Alice, Bob and Mallory are all on a local network. Mallory is an *off-path* attacker with a super-fast connection: her packets always arrive first.

Assume Mallory knows the MAC and IP addresses of Alice and Bob's computers. Assume that Alice will accept the first valid packet she receives as coming from Bob, and ignore what happens once Bob's real packet arrives.

(a) Alice and Bob are communicating through UDP on some unknown ports. What does Mallory need to guess to insert a fake UDP datagram from Bob to Alice?

> **Solution:** Bob's source port, Alice's destination port. (Or just "their ports".)

(b) Alice and Bob are communicating through TCP on some unknown ports. What does Mallory need to guess to insert a fake TCP packet from Bob to Alice?

> **Solution:** Bob's source port, Alice's destination port. Alice and Bob's sequence numbers.

(c) Alice is about to send a DNS request to Bob's resolver for `1.2.3.4.5.6.7.com`. What does Mallory need to guess to spoof the reply?

> **Solution:**
>
> Alice's (ephermal) source port. DNS transaction ID.
>
> Note we do not need to guess Bob's port: for DNS it will be port 53.

(d) Alice is about to send an ARP request for Bob's MAC address. What does Mallory need to guess to spoof the reply?

> **Solution:** Nothing. An off-path attacker can easily spoof ARP replies.

(e) True or False: If Mallory was on-path, she could spoof all of the above responses with a 100% success rate.

● True ○ False

> **Solution:** An on-path attacker can see all of the information above.

SID: ______________________________________

Problem 10 ***Tracking*** **(0 points)**

Let's say the web-page at `http://cute-puppies.com` looks like the following:

```
<html>
    <body>
        <p>Here is a GIF of puppies</p>
        <img src="https://image-host.com/yvxSMjw.gif">
        <script type="text/javascript"
                src="http://yahoo.com/analytics.js"></script>
        <script type="text/javascript"
                src="https://google.com/analytics.js"></script>
    </body>
</html>
```

Note that `google.com` is loaded over HTTPS, whereas `yahoo.com` is loaded over HTTP.

Alice uses Mozilla Firefox on her laptop running Microsoft Windows. In her first browser tab, she has `https://berkeley.edu` open. In a second tab, she opens `http://cute-puppies.com`. In a third tab, she opens `http://cute-puppies.com` once again.

Assume that no two entities share information out of band. Each of the parts below are independent.

(a) Assuming Alice does not use any tracking protection, which entities know that the same person visited `cute-puppies.com` twice?

■ `cute-puppies.com` operators	□ Microsoft
■ `yahoo.com` operators	□ Mozilla
■ `google.com` operators	■ Alice's ISP
■ `image-host.com` operators	□ UC Berkeley

> **Solution:**
>
> Most sites will see this, as Alice's browser makes requests to all of these sites twice.
>
> Alice's ISP sends off all of Alice's traffic, so of course they can see what sites Alice visits.
>
> Neither Microsoft nor Mozilla collect information about what sites you visit in your browser.
>
> UC Berkeley's site is in a different origin, so it cannot see information about the `cute-puppies.com` tab.

(b) Assume Alice opted in for a privacy service run by her ISP. This privacy service blocks analytics scripts based on a URL-based blocklist (not host-based). Which entities know that the same person visited `cute-puppies.com` twice?

■ `cute-puppies.com` operators	□ Microsoft
□ `yahoo.com` operators	□ Mozilla
■ `google.com` operators	■ Alice's ISP
■ `image-host.com` operators	□ UC Berkeley

Solution:

The ISP service can block the HTTP connection, but not the HTTPS connection since HTTPS hides the path of the URL we are visiting. Even though the ISP can see that the user is requesting `google.com`, they do not know if this is an analytics URL.

(c) Assume Alice uses a browser plugin. The browser-plugin blocks the analytics scripts based on a URL-based blacklist (not host-based). Which entities know that the same person visited `cute-puppies.com` twice?

- ■ `cute-puppies.com` operators
- ☐ `yahoo.com` operators
- ☐ `google.com` operators
- ■ `image-host.com` operators
- ☐ Microsoft
- ☐ Mozilla
- ■ Alice's ISP
- ☐ UC Berkeley

Solution:

The browser plugin sees all requests, so it can also block HTTPS requests.

(d) Assume Alice uses a VPN run by UC Berkeley. Which entities know that the same person visited `cute-puppies.com` twice?

- ■ `cute-puppies.com` operators
- ■ `yahoo.com` operators
- ■ `google.com` operators
- ■ `image-host.com` operators
- □ Microsoft
- □ Mozilla
- □ Alice's ISP
- ■ UC Berkeley

Solution:

Alice's connection to the VPN is encrypted, so her ISP does not know which sites she is visiting. However now UC Berkeley sees all of her traffic instead.

SID: ____

Problem 11 ***Boogle*** **(0 points)**

Boogle is a social networking website that's looking into expanding into other domains. Namely, they recently started a map service to try their hand at fusing that with social media. The URL for the main website is `https://www.boogle.com`, and they want to host the map service at `https://maps.boogle.com`.

(a) Describe how to make a cookie that will be sent to only Boogle's map website and its subdomains.

> **Solution:** Set the domain parameter of the cookie to .maps.boogle.com

(b) How can Boogle ensure that cookies are only transmitted encrypted so eavesdroppers on the network can't trivially learn the contents of the cookies?

> **Solution:** Set the secure flag on each cookie.

(c) Boogle adds the ability for users to check in to locations on maps.boogle.com, but discovers an XSS vulnerability that slipped through QA. Name a hotfix they can do to prevent scripts from stealing cookies using XSS.

> **Solution:** Set the HTTPOnly flag on cookies.

(d) Some of the XSS attacks are scraping sensitive information from the map site, like user emails. The security team wants to know the scope of the vulnerability. Can attackers use XSS to also scrape sensitive information from the main site, `https://www.boogle.com`? Explain why or why not.

> **Solution:**
>
> No, the two sites have different origins so `https://maps.boogle.com` cannot read anything from `https://www.boogle.com`.

(e) Boogle wants to be able to host websites for users on their servers. As we saw in HW4, it is not completely safe to host them on `https://[username].boogle.com`. Propose an alternate scheme so that Boogle can still host other users websites with less risk, and explain why this scheme is better.

Note: It is okay if the user sites interfere with each other, as long as they cannot affect official Boogle websites.

> **Solution:**
>
> Boogle should create a new domain exclusively for user hosted content, like `https://[username].boogleusercontent.com`. This way, user sites cannot set cookies that will affect all boogle domains due to the cookie setting policy. This is known as a cookie tossing attack, and is one of the reasons why github hosts user sites on github.io instead of github.com (see `https://blog.github.com/2013-04-09-yummy-cookies-across-domains/`).

Problem 12 ***Scripts and SQL*** **(0 points)**

Answer the following questions about Javascript and SQL.

(a) Oski Bank uses Javascript to deliver account information. The `https://oski.bank/account.js` file is generated server-side, and contains information for currently logged-in user. For example, here is how the file would look for a customer with name “John Doe” and 10232 dollars:

```
display({ name: "John Doe", money: 10232 });
```

i. Assume a victim user visits `evil.com` while this user is logged in with Oski Bank. How could `evil.com` use this to steal the user information provided as input to display for this victim user? Include approximate HTML in your explanation.

> **Solution:**
>
> An evil site just needs to create their own `display` function, and then include the script.
>
> ```
> <script>
>     function display(info) {
>         alert(info.name);
>     }
> </script>
> <script src="https://oski.bank/account.js"></script>
> ```

ii. Which of the following could be used to defend against this attack? Assume you can also update the rest of `oski.bank`.

- ☐ Strong Content-Security-Policy
- ■ CSRF Tokens
- ■ Strict Referer checking
- ☐ Whitelist user inputs
- ☐ `X-Frame-Options` header
- ☐ Only call `display` if `window == top`

> **Solution:** The attack essentially exploits the same idea as CSRF (although it is actually known as “XSS inclusion”). Therefore any defense to CSRF works.

(b) Oski Bank’s site also contains the following code:

```
name = request.form['username']
query = 'SELECT COUNT(*) FROM users WHERE name="{}"'.format(name)
found = database.execute(query).fetchone()[0]
if found: return 'User exists!', 200
else: return 'User not found!', 404
```

Assume that the `users` table also contains a `hash` column, which is a hexadecimal encoding of the hash of the user’s password. Explain in detail how you could determine `dirks`’s hash. (Hint: You might find the SQLite `substr(X, Y, Z)` command helpful. It returns the `Z` character substring of the string `X` starting at the `Y`th character.)

> **Solution:**
>
> Start by sending `dirks'' AND substr(hash, 1, 1) = ''0''--`. If the page prints “User exists!”, then you know the first hex digit of the hash is “0”. Otherwise, keep trying `1, 2, ..., a, ..., f` until you find the first digit.

Then retry using `substr(hash, 2, 1)` until you determine the second hex digit. Keep going for the entire length of the hash.

(Note that the `substr` command is 1-indexed, but we did not deduct for solutions which treated it as 0-indexed.)