# CS 161 Fall 2024 — Introduction to Computer Security — Final Exam

Name: ______________________________

Student ID: ______________________________

This exam is 110 minutes long.

| Question: | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Points: | 0 | 6 | 15 | 16 | 6 | 14 |

| Question: | 7 | 8 | 9 | 10 | 11 | Total |
|---|---|---|---|---|---|---|
| Points: | 8 | 10 | 11 | 7 | 7 | 100 |

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (completely unfilled)
- ● Only one selected option (completely filled)
- Don't do this (it will be graded as incorrect)

For questions with **square checkboxes**, you may select one or more choices.

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

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we may grade the worst interpretation.

**Pre-exam activity** (0 points): Staff tried to draw EvanBot!



Draw your own EvanBot in the blank space above! (or, try to guess which staff drew which one!)

## Q1 *Honor Code* (0 points)

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.

Read the honor code above and sign your name: ______________________________

## Q2 *True/False* (6 points)

Each true/false is worth half a point.

Q2.1 EvanBot's new AI startup ClosedAI hides their secret model weights in a normal-looking `testing.txt` file in their server.

TRUE or FALSE: This is an example of security through obscurity.

○ TRUE ○ FALSE

Q2.2 CodaBot brings a thousand-dollar safe to work to store their lunch in, so EvanBot can't steal their food.

TRUE or FALSE: This is an example of least privilege.

○ TRUE ○ FALSE

Q2.3 TRUE or FALSE: The reason off-by-one attacks work is that the SFP is stored in big-endian.

○ TRUE ○ FALSE

Q2.4 TRUE or FALSE: We can likely find a collision in a cryptographic hash function with a 256-bit output using somewhere between $2^{120}$ to $2^{136}$ hash computations.

○ TRUE ○ FALSE

Q2.5 TRUE or FALSE: Hashing a message before signing it significantly reduces the security of a digital signature.

○ TRUE ○ FALSE

Q2.6 TRUE or FALSE: The `Secure` cookie flag prevents JavaScript from accessing the cookie.

○ TRUE ○ FALSE

Q2.7 TRUE or FALSE: CSRF attacks can be prevented using a content security policy (CSP).

○ TRUE ○ FALSE

Q2.8 TRUE or FALSE: Using TLS can often prevent reflected XSS attacks.

○ TRUE ○ FALSE

Q2.9 TRUE or FALSE: DHCP spoofing allows one machine in a local area network to become a MITM attacker against another machine on the same network.

○ TRUE ○ FALSE

Q2.10 TRUE or FALSE: Firewalls prevent ARP spoofing attacks.

○ TRUE ○ FALSE

Q2.11 True or False: With DNSSEC, it is possible for a MITM between the recursive resolver and the rest of the Internet to view all DNS records sent to the recursive resolver.

- ◯ True
- ◯ False

Q2.12 True or False: Today, we typically defend against attacks on BGP using defenses from higher layers.

- ◯ True
- ◯ False

Q2.13 (0 points) True or False: `EvanBot is a real bot.`

- ◯ True
- ◯ False

## Q3 *Secret Formula — Memory Safety* **(15 points)**

EvanBot stores their secret pancake recipe in the following password-protected program:

```c
1  void login(char* correct_password) {
2      int x = 0xAABBCCDD;
3      char buf[16];
4
5      printf("Enter password:\n");
6      fread(buf, 1, 64, stdin);
7
8      if (strcmp(buf, correct_password) != 0) {
9          printf("%s is incorrect.", buf);
10     } else {
11         // Recipe not shown
12         printf("EvanBot's secret recipe: ...");
13     }
14 }
15
16 int main() {
17     char password[8] = ...; // Password not shown
18
19     login(password);
20     return 0;
21 }
```

**Stack at Line 4**

| |
|---|
| RIP of `main` |
| SFP of `main` |
| (1) |
| (2) |
| RIP of `login` |
| (3) |
| x |
| buf |

Assumptions:

- All memory safety defenses are disabled.
- There is no compiler padding.
- The address of `buf` is `0xFFFFFFA0`.

Q3.1 (1 point) Fill in the stack diagram, assuming the program is paused on Line 4.

- ○ (1) `password` (2) `correct_password` (3) `SFP of login`
- ○ (1) `password` (2) `SFP of login` (3) `correct_password`
- ○ (1) `buf` (2) `SFP of login` (3) `RIP of fread`
- ○ (1) `correct_password` (2) `buf` (3) `SFP of login`

Q3.2 (4 points) Which of the following inputs to `fread` on Line 6 would cause the program to leak the value of `password` with high probability?

*HINT: By "leak the value of `password`", we mean the value in `password` should be somewhere in the program output. The program is allowed to crash as long as it prints the password first.*

- ○ `'A' * 20 + '\x00\x01\x02\x03'`
- ○ `'\x00' * 16`
- ○ `'A' * 64`
- ○ `'A' * 16`

Q3.3 (2 points) Let `OUTPUT` be the value printed by the program in bytes using the correct option in the previous subpart. Which slice of `OUTPUT` gives the value of `password`?

*HINT: For example, `[0:8]` means the first eight bytes of `OUTPUT`.*

- ○ `[0:8]`
- ○ `[16:24]`
- ○ `[32:40]`
- ○ `[8:16]`
- ○ `[24:32]`
- ○ `[40:48]`

Q3.4 (2 points) Which of the following memory safety defenses would prevent the correct exploit in Q3.2 without modifications from leaking the value of `password`? Select all that apply.

- ☐ Non-executable pages
- ☐ ASLR
- ☐ None of the above

For the remaining subparts, assume that the `fread(buf, 1, 64, stdin)` on Line 6 is replaced with `gets(buf)`.

Q3.5 (2 points) After the change to Line 6, does the exploit in Q3.2 without modifications still leak `password`? Select the best option.

- ○ Yes, because `gets` allows us to write arbitrarily many bytes.
- ○ Yes, because `gets` still overwrites the RIP of `login`.
- ○ No, because `gets` will add a null terminator at the end of the input.
- ○ No, because `gets` will overwrite the value in `correct_password`.

Rather than leaking the value of `password`, Eve wants to access the recipe directly.

Q3.6 (4 points) Give an input to `gets` following the below pattern that causes the program to print the secret recipe, i.e., causes the `printf` on Line 12 to run.

`'A' *` ____ + __________

Provide an input for the first blank.

- ○ 0
- ○ 8
- ○ 20
- ○ 28
- ○ 4
- ○ 16
- ○ 24
- ○ 32

Provide an input for the second blank.

| |
|---|
| |

## Q4 *Last-Minute Replacement — Memory Safety* (16 points)

EvanBot writes the following program:

```
1  void foo() {
2      char buf[16];
3
4      fgets(buf, 25, stdin);
5  }
6
7  int main() {
8      int x = 0;
9      int* x_ptr = &x;
10
11     foo();
12     return 0;
13 }
```

**Stack at Line 3**

| |
|---|
| RIP of `main` |
| SFP of `main` |
| `x` |
| (1) |
| RIP of `foo` |
| (2) |
| (3) |

This is the result of running `disas fgets` in GDB:

```
0x08076030: push %ebp
0x08076034: mov %esp, %ebp
...
0x08076054: pop %ebp
0x08076058: ret
```

Assumptions:

- You have access to a **20-byte** `SHELLCODE`.
- ASLR is enabled, **the addresses of the code section of memory are not randomized**, and all other memory safety defenses are disabled.
- There is no compiler padding.

Q4.1 (1 point) Fill in the stack diagram, assuming the program is paused on Line 3.

- ○ (1) `SFP of foo` (2) `buf` (3) `RIP of gets`
- ○ (1) `x_ptr` (2) `SFP of foo` (3) `buf`
- ○ (1) `x_ptr` (2) `buf` (3) `SFP of foo`
- ○ (1) `SFP of foo` (2) `SFP of foo` (3) `buf`

Q4.2 (4 points) Provide an input to `fgets` on Line 4 that executes `SHELLCODE` with probability $\geq \frac{1}{256}$.

| |
|---|
| |

Q4.3 (2 points) For which of the following addresses of `buf` does the correct exploit from Q4.2 **succeed**?

- ☐ 0xFFFFFE00
- ☐ 0xFFFFFEF0
- ☐ 0xFFFFFF04
- ☐ 0xFFFFFF1C
- ☐ 0xFFFFFEE0
- ☐ 0xFFFFFF00
- ☐ 0xFFFFFF0C
- ☐ 0xFFFFFF20

Q4.4 (1 point) Which of the following memory safety defenses would prevent the correct exploit from Q4.2, without modifications, from executing `SHELLCODE`?

☐ Stack canaries ☐ Non-executable pages ☐ None of the above

For the remaining subparts, suppose that `foo` was replaced with the following:

```c
void foo() {
    char buf[64];

    fgets(buf, 73, stdin);
}
```

Also, assume that the instruction for `NOP` is `0x90` and `NOP` is a one-byte instruction.

Q4.5 (5 points) Give an input that executes `SHELLCODE` with maximum probability.

Q4.6 (2 points) For which of the following addresses of `buf` does the correct exploit from Q4.5 **succeed**?

☐ `0xFFFFFE00` ☐ `0xFFFFFEF0` ☐ `0xFFFFFF04` ☐ `0xFFFFFF1C`

☐ `0xFFFFFEE0` ☐ `0xFFFFFF00` ☐ `0xFFFFFF0C` ☐ `0xFFFFFF20`

Q4.7 (1 point) Which of the following memory safety defenses would prevent the corect exploit from Q4.5, without modifications, from executing `SHELLCODE`?

☐ Stack canaries ☐ Non-executable pages ☐ None of the above

This page is intentionally left (mostly) blank.

The exam continues on the next page.

## Q5 *Certainly Intelligent, Alice — Cryptography* (6 points)

For each of the following block cipher modes of operation, answer whether the scheme has confidentiality (IND-CPA), integrity, or both.

Q5.1 (1.5 points) $\mathsf{Enc}(K_1, K_2, M) = (C, T)$, where

$$C = \mathsf{ECB}(K_1, M) \quad T = \mathsf{HMAC}(K_2, C)$$

- ☐ This scheme has confidentiality
- ☐ None of the above
- ☐ This scheme has integrity

Q5.2 (1.5 points) $\mathsf{Enc}(K_1, K_2, M) = (C, T)$, where

$$C = \mathsf{CTR}(K_1, M) \quad T = \mathsf{HMAC}(K_2, C)$$

- ☐ This scheme has confidentiality
- ☐ None of the above
- ☐ This scheme has integrity

Q5.3 (1.5 points) $\mathsf{Enc}(K_1, K_2, M) = (C, T)$, where

$$C = \mathsf{CTR}(K_1, M) \quad T = \mathsf{HMAC}(K_2, C_0 \oplus \cdots \oplus C_n)$$

- ☐ This scheme has confidentiality
- ☐ None of the above
- ☐ This scheme has integrity

Q5.4 (1.5 points) $\mathsf{Enc}(K_1, SK, M) = (C, S)$, where $SK$ is an RSA private key and

$$C = \mathsf{CTR}(K_1, M) \quad S = \mathsf{Sign}(SK, C)$$

Assume that $SK$ is the sender's secret key, and that the receiving party trusts the sender's corresponding public key.

- ☐ This scheme has confidentiality
- ☐ None of the above
- ☐ This scheme has integrity

## Q6 *Shibboleth — Password Storage* **(14 points)**

Consider a password storage server, with the following assumptions:

- There are $U$ users who each choose a password uniformly at random from a common pool of $N$ possible alphanumeric passwords.
- Usernames are unique. Passwords are not necessarily unique.
- The attacker knows the set of possible usernames and passwords.

Consider an attacker who is able to:

- Read and modify what is stored in the password database.
- Perform offline brute-force attacks (but not online attacks).

**Scheme 1**: For each user, the server stores in the password database the username and:

$$\mathsf{H}(\text{username}) \,\|\, \mathsf{H}(\text{password})$$

Q6.1 (1 point) In Scheme 1, is the attacker able to determine all pairs of users who share the same password, by computing at most $N + 1$ hashes (i.e., running the hash function on at most $N + 1$ inputs)?

- ○ Yes, without computing any hashes
- ○ Yes, it requires computing at least one and at most $N + 1$ hashes
- ○ No (it is impossible, or it would require computing at least $N + 2$ hashes)

Q6.2 (2 points) In Scheme 1, how many users' passwords is the attacker able to learn, if the attacker is limited to computing at most $N + 1$ hashes?

○ None ○ Only one ○ All of them

Q6.3 (1 point) In Scheme 1, what is the attacker able to do to a specific user's password?

○ Nothing ○ Change the password to any value

**Scheme 2**: For each user, the server stores the username and:

$$\mathsf{H}(\texttt{username} \,\|\, \texttt{password})$$

Q6.4 (1 point) In Scheme 2, is the attacker able to determine all pairs of users who share the same password, by computing at most $N + 1$ hashes?

- ○ Yes, without computing any hashes
- ○ Yes, by computing at least one and at most $N + 1$ hashes
- ○ No (it is impossible, or it would require computing at least $N + 2$ hashes)

Q6.5 (2 points) In Scheme 2, how many users' passwords is the attacker able to learn, if the attacker is limited to computing at most $N + 1$ hashes?

○ None ○ Only one ○ All of them

Q6.6 (1 point) In Scheme 2, what is the attacker able to do to a specific user's password?

○ Nothing ○ Change the password to any value

**For all remaining subparts**, assume that the attacker is able to ask the server to create new users, provided that the username does not already exist. In other words, the attacker can give the server `username, password` and the server will add a corresponding entry if that username is not already present.

**Scheme 3**: For each user, the server stores the username and

$$\mathsf{HMAC}(K, \texttt{username} \parallel \text{"abc"} \parallel \texttt{password})$$

$K$ is a key known only by the server.

Q6.7 (1 point) In Scheme 3, is the attacker able to determine all pairs of users who share the same password, by computing at most $N + 1$ hashes?

- ○ Yes
- ○ No

Q6.8 (5 points) Provide an attack where the attacker picks a new password for `evanbot` (known to the attacker) and chooses new values to store in the password database, to replace `evanbot`'s password with that new password.

Pick a new password for `evanbot`:

| |
|---|
| |

The attacker asks the server to create a new user with the following username/password:

Username of new user:

| |
|---|
| |

Password of new user:

| |
|---|
| |

The attacker reads the HMAC tag stored in the database for the new user. Let $x$ be this HMAC tag.

Give a new value to replace the HMAC tag for `evanbot` with.

New HMAC tag stored for `evanbot`:

| |
|---|
| |

At this point, the attacker should be able to log in to the `evanbot` account using the new password picked above.

## Q7 *Query Sanitization — SQL Injection* **(8 points)**

EvanBot attempts to write their own application using a SQL database. They do not believe in parameterization, and are confident in their ability to properly sanitize the code.

```
CREATE TABLE users (
    uuid      INT,
    fullname  STRING,
    username  STRING,
    password  STRING,
    secret    CHAR(9)
);
```

Suppose that EvanBot has two SQL queries that tell you what the `secret` attribute of a particular user is. Query 0 accepts a `UUID` as input, and Query 1 accepts two inputs: `username` and `password`.

**Query 0**:

```
SELECT fullname, secret FROM users WHERE uuid = $0
```

**Query 1**:

```
SELECT fullname, secret FROM users WHERE username = '$0' AND password = '$1'
```

Q7.1 (3 points) Which inputs would leak all users' `secret` values?

Query 0:

- ☐ `uuid: 1234 UNION SELECT fullname, secret FROM users --`
- ☐ `uuid: 1234 AND 1=1 --`
- ☐ `uuid: 1234' UNION SELECT * FROM users --`
- ☐ None of the above

Query 1 :

- ☐ `username: EvanBot`
  `password: UNION SELECT fullname, secret FROM users`
- ☐ `username: Evan' OR '1'='1`
  `password: Bot' OR 1<=1 --`
- ☐ `username: 1234`
  `password: 1234' UNION SELECT * FROM users --`
- ☐ None of the above

EvanBot decides to start sanitizing the inputs to the SQL queries by removing all single quotes from the input value.

Assumptions:

- The ASCII encoding for a single quote ( `'` ) is 39.
- Whenever `CHAR(x)` is processed by SQL, it is first replaced with the ASCII decoding of *x*.
  For instance, `CHAR(65)bc` becomes `abc` during query processing.
- There is no implicit type conversion.
- Sanitization happens **before** `CHAR` is resolved.

Q7.2 (3 points) Which of the following inputs would leak all users' `secret` values? Select all that apply.

Query 0:

- ☐ `uuid: 1234' UNION SELECT * FROM users --`
- ☐ `uuid: 1234 OR 1=1`
- ☐ `uuid: CHAR(39) UNION SELECT * FROM users`
- ☐ None of the above

Query 1:

- ☐ `username: 1234 UNION SELECT fullname, secret FROM users --`
  `password: null`
- ☐ `username: Ali''ice'`
  `password: '''' OR 1=1 --`
- ☐ `username: CHAR(39) UNION SELECT * FROM people --`
  `password: ' OR 1 = 0`
- ☐ None of the above

*For your convenience, Query 0 is repeated here.*
**Query 0**:

```
SELECT fullname, secret FROM users WHERE uuid = $0
```

Q7.3 (2 points) Which of the following inputs to Query 0 would leak the `secret` value for `username = 'Bob'`? Select all that apply.

- ☐ `uuid: BOB AND 1=1`
- ☐ `uuid: CHAR(39)BobCHAR(39)`
- ☐ `uuid: 1234 'O'R 'username'=CH'AR(39)Bob'CHAR(39)`
- ☐ `uuid: 123 AND username='Bob' OR username=Bob`
- ☐ None of the above

## Q8 *The Cookies Zone - Web Security* **(10 points)**

Ben runs the website `ben.com`. He posts information about the location where he is traveling.

Q8.1 (1 point) Assume a cookie has the fields `Domain=ben.com`, `Path=/location`. Which of the following pages, if visited by a browser with this cookie, would receive the cookie? Select all that apply.

- ☐ `http://ben.com/`
- ☐ `https://ben.com/`
- ☐ `http://ben.com/location`
- ☐ `https://ben.com/location`
- ☐ `https://ben.com/home/location`
- ☐ `http://ben.com/location/california`
- ☐ `https://reviews.ben.com/location`
- ☐ None of the above

Q8.2 (1 point) Ben adds some cookies to his reviews site, located at `reviews.ben.com`. Which of the following pages can set cookies with `Domain=reviews.ben.com`? Select all that apply.

- ☐ `https://ben.com`
- ☐ `https://reviews.ben.com`
- ☐ `https://reviews.ben.com/home`
- ☐ `https://admin.reviews.ben.com/home`
- ☐ `https://views.ben.com/`
- ☐ None of the above

Suppose that anyone who visits the website can only see Ben's location if they are logged in with the proper session token.

Q8.3 (2 points) For this subpart only, assume Ben designs his session tokens to contain only the username of the logged in user and the date at which they logged in (e.g., "`ben-17december2024`").

Since he sets both the `HttpOnly` and `Secure` flags, he believes that this is a secure system which will only allow people who can log in to access his location. Is he correct?

- ◯ Yes, because the `Secure` flag means that the cookie cannot be tampered with.
- ◯ Yes, because the `HttpOnly` flag means that the cookie cannot be tampered with.
- ◯ No, because cookies can be modified by the user, making it possible for them to delete valid session tokens at will.
- ◯ No, because cookies can be modified by the user, making it possible for them to create a valid session token from scratch.

For the remaining subparts, assume Ben uses session-based authentication with randomly-generated session tokens, as seen in lecture. Mallory wants to learn Ben's location from his website, but she does not have an account she can log into.

Q8.4 (1 point) Mallory leaves a comment on Ben's website saying `<script>alert(1)</script>` and gets a popup saying `"1"` when she posts it. She notices that anyone else visiting the comments page gets the same popup as a result of her comment. What kind of vulnerability is this an example of?

- ○ Stored XSS
- ○ CSRF
- ○ Reflected XSS
- ○ Clickjacking

Q8.5 (1 point) Mallory then visits the following URL:

```
https://ben.com/search?query=<script>alert(1)</script>
```

and also sees the `"1"` popup. What kind of vulnerability is this an example of?

- ○ Stored XSS
- ○ CSRF
- ○ Reflected XSS
- ○ Clickjacking

Now, Ben makes a code change that prevents all user input containing `<script>` tags from working.

Q8.6 (4 points) Mallory notices that image tags have an `onerror` attribute, which will execute the provided line of JavaScript code only if the image fails to load. For instance, `<img src="invalid_url" onerror="alert(1)"/>` will cause a popup window showing 1.

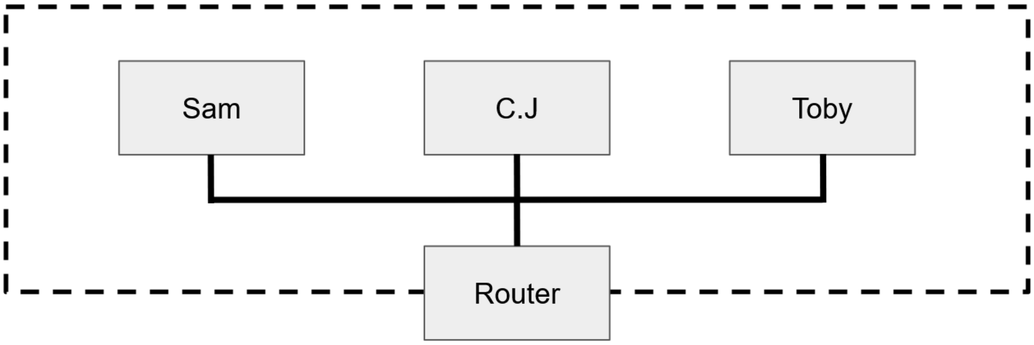
Create an input that Mallory could post as a comment on the site that would send Ben's session token to her. More precisely, when Ben visits ben.com and views Mallory's comment, Ben's browser should make a POST request to the URL `https://mallory.com/logger?value=____`, where the blank should be filled in with something that contains or reveals Ben's session token.

Assumptions:

1. JavaScript can make POST requests with the code `post(URL)`.
2. Assume Ben's session tokens do not have the `HttpOnly` attribute.
3. JavaScript can access cookies by reading the global variable `document.cookie`.

## Q9 *The West Wing – Networking* **(11 points)**

Sam has recently joined a broadcast LAN (local area network) with two others: C.J. and Toby.




Q9.1 (0.5 point) Sam wants to send a packet to Toby over the local network. What information does Sam's computer need to obtain in order to successfully send the message? Select all that apply.

- ☐ Toby's MAC address
- ☐ The router's MAC address
- ☐ C.J.'s MAC address
- ☐ The router's IP address

Q9.2 (0.5 point) Which protocol would Sam's computer follow to learn this information?

○ ARP ○ DHCP ○ BGP ○ DNS

Q9.3 (1 point) Suppose C.J. wishes to eavesdrop on the message between Sam and Toby. What **must** be true for this to occur? Select all that apply.

- ☐ C.J. is a man-in-the-middle attacker between Sam and Toby
- ☐ C.J. controls the router
- ☐ The LAN is wireless
- ☐ None of the above

Q9.4 (1 point) Sam now wishes to connect to a server outside the LAN. Which information does Sam's computer need to obtain or learn in order to successfully connect using TCP?

Assume that Sam's computer has just joined the network and only knows the server URL. Select all that apply.

- ☐ The server's MAC address
- ☐ Sam's computer's IP address
- ☐ The server's IP address
- ☐ The router's MAC address
- ☐ None of the above

Q9.5 (1 point) Which protocol could be used to look up the server's IP address?

○ DHCP ○ BGP ○ DNS ○ DoS

Q9.6 (1 point) Toby sends a sensitive message to a server outside the LAN using TLS. Select all true statements.

- ☐ The router can see Toby's IP address.
- ☐ The router can see the IP address of the server.
- ☐ The router can see the message contents.
- ☐ The router can modify the message without being detected.
- ☐ None of the above.

Toby starts looking into SYN cookies, firewalls, and intrusion detection systems.

Q9.7 (2 points) Which of the following best explains the motivation and design for SYN cookies?

- ◯ To prevent SYN flooding by enforcing cookie policy.
- ◯ To prevent SYN flooding by putting server state in the initial sequence number.
- ◯ To prevent denial of service attacks by requiring a valid CSRF token.
- ◯ None of the above.

For the next two subparts, select whether the intended rule is possible with stateful firewalls only, both stateful and stateless firewalls, or not possible for any firewall. Assume that the firewall is not given access to TLS keys from client machines.

Q9.8 (1 point) Prevent all outgoing TCP packets to a specific IP address.

◯ Both ◯ Stateful only ◯ Not possible

Q9.9 (1 point) Drop all TLS connections when the word "cheese" is included in a message.

◯ Both ◯ Stateful only ◯ Not possible

Q9.10 (1 point) A computer virus that has existed for many years attempts to infect Toby's computer. Which intrusion detection paradigm works best to defend against this attack?

◯ Signature ◯ Anomaly ◯ Specification

Q9.11 (1 point) Toby wishes to flag any TLS-encrypted messages his computer receives that contain the word "block". Which type of intrusion detection system works best to implement this policy?

- ◯ HIDS, because a NIDS would have difficulty unambiguously interpreting the message.
- ◯ HIDS, because a NIDS would cost too much.
- ◯ NIDS, because a HIDS would have difficulty unambiguously interpreting the message.
- ◯ NIDS, because a HIDS would cost too much.

## Q10 *Wish we used Public-key Authentication – WPA* (7 points)

Q10.1 (1 point) True or False: During the WPA2-PSK handshake, the client sends the password to the access point.

○ True ○ False

Q10.2 (1 point) True or False: When clients send a data packet to the router in a WPA2-Enterprise network, other network clients cannot decrypt the packet.

○ True ○ False

Q10.3 (1 point) True or False: An attacker who records a WPA2-PSK handshake can attempt to brute-force the password offline (i.e., without needing to attempt a connection for each guess).

○ True ○ False

Q10.4 (1 point) Suppose that the **server nonce** (ANonce) was no longer used or sent in the WPA2-PSK handshake. Select all true statements.

☐ Attackers could record a WPA2 connection and successfully replay the client messages to the access point in a new connection.

☐ Attackers could record a WPA2 connection and successfully impersonate the access point by replaying the access point's recorded messages to a new client.

☐ None of the above

Q10.5 (1 point) Suppose that the **client nonce** (SNonce) was no longer used or sent in the WPA2-PSK handshake. Select all true statements.

☐ Attackers could record a WPA2 connection and successfully replay the client messages to the access point in a new connection.

☐ Attackers could record a WPA2 connection and successfully impersonate the access point by replaying the access point's recorded messages to a new client.

☐ None of the above

Q10.6 (2 points) Which option best explains the most important difference between WPA2-Enterprise compared to WPA2-PSK?

○ WPA2-Enterprise uses AES-256 bit encryption; WPA2-PSK uses AES-128 bit encryption.

○ WPA2-Enterprise provides a key to the client and access point after the client authenticates over TLS to a central authentication server.

○ WPA2-Enterprise involves the client sending a username and password to the access point to log in.

## Q11 *DNS + Dan Kaminsky + Tor = DaNS Torinsky* **(7 points)**

Suppose you want to visit `evanbot.berkeley.edu` and `cooked.berkeley.edu`, but need to find their IP addresses by performing DNS queries.

Q11.1 (1 point) For this subpart only, assume that DNS caching is disabled. How many DNS requests does a recursive resolver need to make to find the IP addresses corresponding to the two domains?

- ○ 4
- ○ 5
- ○ 6
- ○ 8

Q11.2 (1 point) Assume that the DN S cache is initially empty, but every subsequent DNS response is cached. How many non-cached DNS requests does a recursive resolver need to make to find the IP addresses corresponding to the two domains?

- ○ 3
- ○ 4
- ○ 5
- ○ 6

Q11.3 (3 points) Suppose that no subdomain exists under `cooked.berkeley.edu`, and all possible subdomains under `evanbot.berkeley.edu` exist (e.g., `1.evanbot.berkeley.edu`, `2.evanbot.berkeley.edu`, etc.).

What sequence of DNS queries would allow an off-path attacker to perform a Kaminsky attack to poison `berkeley.edu`?

Assume that DNS responses are cached, the DNS cache is initially empty, and bailiwick checking is enabled. Select all that apply.

- ☐ Repeatedly query for `cooked.berkeley.edu`
- ☐ Query for **`N`**`.google.com` for **N** = 1,2,3,…
- ☐ Query for **`N`**`.evanbot.berkeley.edu` for **N** = 1,2,3,…
- ☐ Query for **`N`**`.cooked.berkeley.edu` for **N** = 1,2,3,…
- ☐ None of the above.

Q11.4 (2 points) Assume that a recursive resolver starts performing DNS queries (without DNSSEC) over TCP. Also, assume that Mallory is a MITM between all DNS nameservers and the rest of the Internet.

If the recursive resolver uses Tor, does this limit Mallory's ability to tamper with DNS traffic?

- ○ Yes, since Tor uses multiple layers of encryption, preventing a MITM between the exit node and the DNS nameserver from tampering with the message undetected.
- ○ Yes, since Tor uses randomized relay selection, ensuring that Mallory does not know who to forward the tampered DNS response to.
- ○ No, since DNS cannot work over Tor at all due to the exit node sending encrypted packets to the DNS nameserver, who expects plaintext packets.
- ○ No, Mallory could still tamper with TCP packets coming from the DNS nameserver.

## Post-Exam Activity

Help EvanBot decorate the tree! (Or cut it down using our rightfully-owned axe.) Your art won't affect your grade.


## Comments/Assumptions Box

Congratulations for making it to the end of the exam! Feel free to leave any thoughts, comments, feedback, or doodles here. These comments won't affect your grade.

If there's anything else you want us to know, or you feel like there was an ambiguity in the exam, please put it in the box below. For ambiguities, you must qualify your answer and provide an answer for both interpretations. For example, "if the question is asking about A, then my answer is X, but if the question is asking about B, then my answer is Y". You will only receive credit if it is a genuine ambiguity and both of your answers are correct. We will only look at ambiguities if you request a regrade.