PRINT your name: _____________________________________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will lead to a score of zero (0) on the examination and the misconduct will be reported to the Center for Student Conduct.

SIGN your name: _____________________________________________

PRINT your class account login: cs161-_____ and SID: ___________________________

Your TA’s name: ________________________________________________

Your section time: _______________________________________________

Name of the person sitting to your left: ________________ Name of the person sitting to your right: _______________________

You may consult one sheet of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators and computers are not permitted. Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

You have 80 minutes. There are 5 questions, of varying credit (200 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

<table>
<thead>
<tr>
<th>Question:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>36</td>
<td>42</td>
<td>46</td>
<td>36</td>
<td>40</td>
<td>200</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An astute attacker has just discovered a buffer overflow vulnerability present in a slightly outdated version of SSH. By sending well-crafted input at the password prompt, the exploit allows the attacker to gain a root shell.

Note that SSH servers run on TCP port 22 by default.

(a) The attacker searches for victims by connecting to TCP port 22 of the main web server for several sites whose servers he would like to compromise. Upon successfully connecting, the SSH server sends a string identifying what version it’s running, which enables the attacker to determine whether the server has the vulnerability.

After some time (and with fruitless results), the attacker realizes that the main web server of one of the sites, www.luser.com, doesn’t have an SSH server listening on port 22 at all. He “port scans” the target and finds that SSH is indeed running, but on a different, non-standard port. And that server is in fact running the vulnerable version of SSH!

Identify a relevant security principle and justify it in a sentence or two. (You only need to list one principle in cases where you think perhaps more than one applies.)

(b) The attacker sends the evil input and, lo and behold, obtains a root shell. As the attacker pokes around the compromised system, he realizes that www.luser.com is also used as a staging machine to deploy source code to several other web application servers. The deployment script copies the files to target machines via SSH. To keep the copying process quick and easy, the web application developer team has enabled password-less access to the other servers, even though the site’s security officers have set a policy that all access between machines must use two separate forms of authentication. The additional servers will accept any incoming SSH connection for user root that comes from www.luser.com without requiring authentication.

The attacker is delighted: his compromise of the staging server will now gain him root access to the several other servers as well!

Identify a relevant security principle and justify it in a sentence or two.
(c) The attacker now wants to ensure that he will continue to have access to the compromised systems in the future, even if the SSH version is upgraded. Since each machine runs a web server, the attacker decides to replace the web server binary (httpd) with a modified version that includes a backdoor. The backdoor grants access to a root shell whenever the web server receives a special URL.

However, it turns out that each of the compromised systems is running a detection tool called AIDE. AIDE is a program that periodically takes a snapshot of a machine’s file system by computing summaries of the files and recording their modification times. The site’s security staff has configured AIDE so that whenever it detects a change to the files that it monitors, it sends out an alert email to the machine’s administrator.

Thus, soon after adding the backdoor, the administrator receives a warning of a potential intrusion, since the modification time of the httpd binary has changed.

Identify a relevant security principle and justify it in a sentence or two.

(d) Unfortunately, the security staff set the rules for the files that AIDE tracks too broadly. Several web server log files fall under these rules, and since these files change frequently, the administrator receives one or two dozen emails about potential integrity violations every day, and ends up ignoring them all.

Identify a relevant security principle and justify it in a sentence or two.
Problem 2  \textit{Spoofing}\hfill (42 points)

The following figure shows a diagram of three networks and the Internet.

\textbf{Network Diagram}

The networks are configured as follows:

- Hosts A and B are laptops on the same open WiFi network, subnet 128.32.150.0/25.
- Host C is a wired desktop, subnet 128.32.150.128/25.
- R is the edge router for the Berkeley network and is on subnet 128.32.140.0/24.
- D and E are on separate networks run by different ISPs.

Assumptions for this problem are as follows:

- The networks do not employ any form of firewalling or filtering.
- All TCP implementations are modern.
- Attackers are lucky and will win timing races if they can act instantaneously upon observing an action of the victim.
- A “successful” attack requires that the attack will work without requiring more than a half dozen packets.
For each of the following attacks, **mark with an X** every host that can successfully carry out that attack against each specified victim. If no hosts can successfully carry out the attack, mark the entry for “None”. Ignore the entries in the table with “–”.

(a) (12 points) Attacker has the ability to successfully spoof DHCP offers directed at the victim, such that the victim will accept the offer as genuine:

<table>
<thead>
<tr>
<th>Attackers</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victims</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

(b) (12 points) Attacker can successfully spoof the IP source address of traffic sent to the victim to appear as though it comes from E:

<table>
<thead>
<tr>
<th>Attackers</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victims</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

(c) (12 points) Attacker can successfully initiate and complete a TCP handshake with the victim, such that to the victim it appears as though E initiated the connection:

<table>
<thead>
<tr>
<th>Attackers</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victims</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

(d) (6 points) Attacker can successfully spoof replies to the victim’s HTTP GET requests made to E:

<table>
<thead>
<tr>
<th>Attackers</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victims</td>
<td>A</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Problem 3  Web security  (46 points)

EasyWeb Inc.’s web server functions as follows: Whenever it receives a URL of the following form:

http://www.easyweb.com/login.do?user=\textit{username}\&pass=\textit{password}

it logs in users assuming that the provided password indeed matches the one associated with the provided username.

Note: for this problem, do not concern yourself with network eavesdroppers, nor with whether users pick strong passwords. In addition, you can do the last three parts, (c) through (e), separately from the first two, (a) and (b) (in case you get stuck on the first two).

(a) (8 points) Name and briefly describe a vulnerability in EasyWeb Inc.’s use of this approach. Briefly sketch a scenario in which an attacker would exploit the vulnerability.

(b) (10 points) Briefly describe a defense against the vulnerability identified in part (a).
An astute Berkeley student has submitted a bug report to EasyWeb Inc. and they have changed the way their website works. EasyWeb Inc. now requires that all login requests go through a specific login form page. However, to make the approach somewhat backwards-compatible, they designed the mechanism as follows. When a user of the service first surfs to the URL http://www.easyweb.com/login.do?user=\textit{username}, the website returns a web page that conveniently pre-fills part of the login form for the user, like this:

![Login Form]

The text \textit{username} will be replaced with the literal (exact) value from the URL. The form also contains \textbf{hidden fields} that will be sent to the web server upon clicking \textit{Sign in}. These fields contain, among other things, the \textit{username} from the URL.

(c) (8 points) In using this approach, EasyWeb Inc. has introduced a new vulnerability while fixing the previous one. \textbf{Name} the vulnerability and \textbf{briefly describe} it.

(d) (10 points) Explain how an attacker can use this new vulnerability to perform an attack. Briefly sketch how the attacker sets up the attack and what happens when the attack occurs.

(e) (10 points) Briefly sketch a potential defense that will prevent at least some instances of the attacks enabled by the vulnerability in part (d), even if not all of them. Discuss a drawback or limitation of the defense.
Problem 4  **Short Answers**  (36 points)

(a) (7 points) Suppose you implement a web service entirely in Java. For database access, you are careful to always use Prepared statements. Assume there are no flaws in the Java environment.

Because of the use of Java and Prepared statements, for the following circle all of the threats that you can confidently conclude the web service will not be vulnerable to:

- Buffer overflow
- CSRF
- TOCTTOU vulnerabilities
- SQL injection
- Reflected XSS
- Stored XSS
- None of the above

(b) (7 points) Suppose a developer implements a full-featured web browser entirely in Java. The browser does not use a database, and again assume there are no flaws in the Java environment.

Because of the developer’s use of Java, and the browser’s absence of any database usage, for the following circle all of the threats that the developer can confidently conclude the users of the browser will not be vulnerable to:

- Buffer overflow
- CSRF
- TOCTTOU vulnerabilities
- SQL injection
- Reflected XSS
- Stored XSS
- None of the above
(c) (10 points) Suppose an attacker wishes to disrupt access to a web-based service.

If the attacker finds they lack the resources to launch a successful network-layer DoS attack, they might still be able to launch a successful transport-layer DoS attack:

TRUE    FALSE

If the attacker finds they lack the resources to launch a successful network-layer DoS attack, they might still be able to launch a successful application-layer DoS attack:

TRUE    FALSE

If the attacker finds they lack the resources to launch a successful application-layer DoS attack, they might still be able to launch a successful network-layer DoS attack:

TRUE    FALSE

DoS attacks fundamentally concern exhausting some sort of resource required by the target:

TRUE    FALSE

Fuzz testing can be used to find some forms of DoS vulnerabilities:

TRUE    FALSE

(d) (12 points) Company A requires that its employees pick 6-character passwords made up of combinations of lowercase letters, uppercase letters, and digits. No other characters are allowed, and a given user’s password must not use any character twice.

Company B requires that its employees pick 12-character passwords, where each of the 12 can be any of 100 possible characters. Unlike for Company A, Company B’s employees can reuse characters in their passwords. However, Company B finds that users often make mistakes with these long passwords, so if an authentication attempt fails, the login server helps the user by telling them how many of the initial letters were correct. For example, if a password entered was "abcdefgij" and the server replies Wrong, but the first 4 letters were correct, then "abcd" are correct, ’e’ is wrong, and nothing is revealed about the correctness of the letters after ’e’.

Suppose an attacker is using exhaustive brute force to guess the password of user U1 at Company A, and user U2 at Company B. Both usernames are valid at the respective companies and the users have chosen passwords that conform with the policy.

1. Write down an expression for the maximum number of attempts the attacker needs for guessing the password of user U1 at Company A.

2. Write down an expression for the maximum number of attempts the attacker needs for guessing the password of user U2 at Company B.
Problem 5  **Software Vulnerabilities**  (40 points)

For the following code, assume an attacker can control the value of `basket` passed into `eval_basket()`. The value of `n` is constrained to correctly reflect the number of elements in `basket`.

The code includes several security vulnerabilities. **Circle three such vulnerabilities** in the code and **briefly explain** each of the three.

```c
struct food {
    char name[1024];
    int calories;
};

/* Evaluate a shopping basket with at most 32 food items. 
Returns the number of low-calorie items, or -1 on a problem. */
int eval_basket(struct food basket[], size_t n)
{
    struct food good[32];
    char bad[1024], cmd[1024];
    int i, total = 0, ngood = 0, size_bad = 0;

    if (n > 32)
        return -1;

    for (i = 0; i <= n; ++i ) {
        if (basket[i].calories < 100)
            good[ngood++] = basket[i];
        else if (basket[i].calories > 500) {
            size_t len = strlen(basket[i].name);
            snprintf(bad + size_bad, len, "%s ", basket[i].name);
            size_bad += len;
        }
    }

    total += basket[i].calories;

    if (total > 2500) {
        const char *fmt = "health-factor --calories %d --bad-items %s";
        fprintf(stderr, "lots of calories!");
        snprintf(cmd, sizeof cmd, fmt, total, bad);
        system(cmd);
    }

    return ngood;
}
```

Reminder: `strlen` calculates the length of a string, not including the terminating '\0' character. `snprintf(buf, len, fmt, ...)` works like `printf`, but instead writes to `buf`, and won’t write more than `len - 1` characters. It terminates the characters written with a '\0'. `system` runs the shell command given by its first argument.