Midterm Exam

This exam is closed book and closed notes. You may use one double-sided 8.5"x11" cheat sheet. You may not use any electronic devices.

Write your answers in the spaces provided on the question sheets in the space provided. You may use the backs for scratch work.

The time limit is **one hour and 20 minutes**.

Name: _____

PID: _____

Question:	1	2	3	4	5	6	Total
Points:	10	10	14	10	6	0	50
Score:							

1. (10 points) **True or False (circle your answer).**

   (a) **True** or **False**: Stack canaries are used to detect stack buffer overflows.

   (b) **True** or **False**: The execution time of a program *always* leaks information about its inputs.

   (c) **True** or **False**: Cache timing attacks can be used to break process isolation.

   (d) **True** or **False**: Use after free vulnerabilities can be used to hijack control flow.

   (e) **True** or **False**: Using SameSite=Strict cookies is a countermeasure against XSS attacks.

   (f) **True** or **False**: IP is a packet-based protocol.

   (g) **True** or **False**: An air gapped machine would join a local WiFi network.

   (h) **True** or **False**: Virtual memory is a mechanism of isolating the memory address space of multiple processes.

   (i) **True** or **False**: Using `strncpy` instead of `strcpy` eliminates string-based buffer overflows.

   (j) **True** or **False**: Loading an address from main memory will usually take the same amount of time as loading from a CPU cache on any given modern computer.

2. **Short answer.**

(a) (2 points) What attacks does ASLR mitigate? How?

Buffer overflow | Attacks that overwrite jumps to an absolute address.

Adding random offsets to different memory regions.

(b) (2 points) What is the principle of least privilege? Give an example.

only just enough privilege to finish the task

(c) (2 points) What is a reference monitor? Give an example from class.

mitigates requests across security boundary.

seccomp bpt
virtual mem

(d) (2 points) What attack does constant-time programming mitigate?

Time-based side channel attacks.

(e) (2 points) Is the Unix filesystem more like an ACL or a capability-based system? Explain.

ACL
store permission info per file

(capabilities: store perm info per user)

3. **Application security.**

You have reverse-engineered a malware binary and discovered the following function.

```
1  void copy(char *arg) {
2          char buf[32];
3          strcpy(buf, arg);
4  }
```

You are developing proof-of-concept exploit code.

After trapping on a breakpoint at line 3, gdb yields the following:

```
(gdb) x/32xw $esp
0xbffff120: 0xbffff130 0xbffff3e5 0x00000000 0x00000000
0xbffff130: 0x00000000 0x00000000 0x00000000 0x00000000
0xbffff140: 0x00000000 0x00000000 0x00000000 0x00000000
0xbffff150: 0x00000002 0xbffff204 0xbffff178 0x080481a5
0xbffff160: 0xbffff3e5 0xbffff204 0xbffff210 0x080481a8
0xbffff170: 0x00000000 0x080ea00c 0x08049630 0x0804907a
0xbffff180: 0x00000002 0xbffff204 0xbffff210 0x00000000
0xbffff190: 0x00000000 0x080481a8 0x00000000 0x080ea00c
(gdb) info reg ebp
ebp            0xbffff158 0xbffff158
```

buf begins at 0xbffff130. The machine is a little-endian system that behaves like the VM from PA 1. There are no defenses such as ASLR, stack canaries, or W^X/DEP.

(a) (2 points) What is the address of the stack pointer?

0x bffff 120

(b) (2 points) What is the address of the return address?

0x bffff 158+4 = 0x bffff 15C

(c) (2 points) How many bytes away is buf from the return address?

0x bffff15C − 0x bffff 130 = 44

(d) (2 points) What is the address of the previous stack frame's base pointer?

0xbffff 158

(e) (4 points) You want to run some shellcode that is 24 bytes long and works like the shellcode provided in PA 1. Write the input bytes (in hex) to be copied to `arg` for the most concise possible exploit. For positions that will contain the payload, write "shellcode".

Shellcode											
41	41	.	.	.							
.	.	.	.			41	41	30	f1	ff	bf

bf ff f1 30

(f) (2 points) Describe how you could still exploit this vulnerability if W^X/DEP (non-executable stack) is enabled.

ROP / libc

4. **Privilege Escalation.**

You are consulting for a startup, and they have given you access to a shared developer machine. They wanted to let normal users install software packages, so a sysadmin wrote the following small utility, called `user-install`, which takes the name of a package as a command line argument and will install it on the system.

```
1  int main(int argc, char** argv) {
2      setuid(0);
3      char *cmd = malloc(strlen(argv[1]) + 100);
4      strcpy(cmd, "apt update -y && apt install -y ");
5
6      strcat(cmd, argv[1]);
7      system(cmd);
8  }
```

The `apt` program is a command-line interface for the Ubuntu package management. The `apt` utility normally requires root privileges to run, so the compiled program has the setuid bit set.

```
$ ls -l /usr/bin/user-install
-rwsr-xr-x 1 root root 2080 Jan 3  2022 /usr/bin/user-install
```

(a) (2 points) What purpose does setting the setuid bit serve here?

Change EUID to the owner of the file

allows you to run a program as the owner of the file

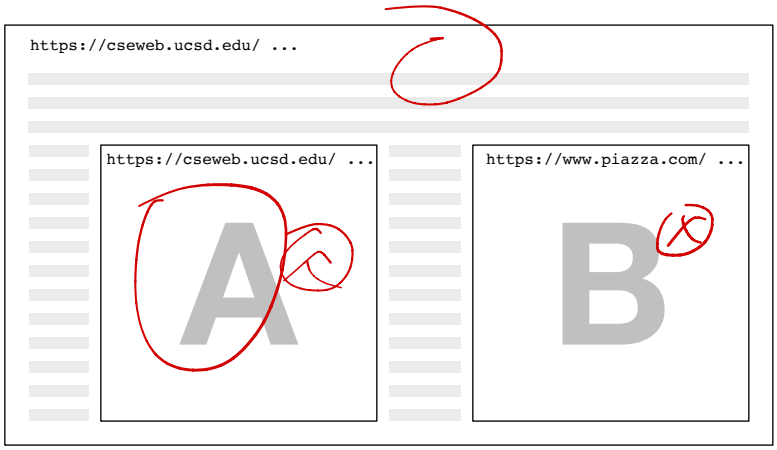
(b) (5 points) Describe how a normal user can exploit this program to gain a root shell on the machine.

Cmd: " apt update -y && apt install -y && vim "/bin/sh"

system(cmd)

(c) (3 points) How would you recommend the company fix this vulnerability?

5. **Same-origin policy.** Consider a Web page containing two `iframe` elements, illustrated below. The enclosing page has a URL of the form `https://cseweb.ucsd.edu/...`, where "..." denotes some URL path. This page contains two `iframe` elements, denoted *A* and *B*. The source of `iframe` *A* is of the form `https://cseweb.ucsd.edu/...`, but not necessarily the same as that of the enclosing page. The source of `iframe` *B* is of the form `https://www.piazza.com/...`.



Assume the browser cookie jar contains the following cookies:

X: `domain=piazza.com; path=/`

Y: `domain=ucsd.edu; path=/`

Z: `domain=cseweb.ucsd.edu; path=/; HttpOnly;`

(a) (1 point) **True** or **False**: A script running in the enclosing page can access the content of iframe (A) via the DOM.

(b) (1 point) **True** or **False**: A script running in iframe (A) can access the content of the sibling iframe (B) via the DOM.

(c) (1 point) Can a script in (B) *submit* an HTTP request (e.g., a form `POST` request) to `cseweb.ucsd.edu`? If so, which cookie(s) does the browser send to `cseweb.ucsd.edu`?

(d) (1 point) When iframe (A) makes a request to `cseweb.ucsd.edu`, which cookie(s) does the browser send?

(e) (1 point) When iframe (B) makes a request to `ucsd.edu`, which cookie(s) does the browser send?

(f) (1 point) **True** or **False**: A script running in iframe (A) can read the cookie for `ucsd.edu` (*Y*).

6. **Optional.** How are you feeling?