Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2010

Quiz I

All problems are open-ended questions. In order to receive credit you must answer the question as precisely as possible. You have 80 minutes to finish this quiz.

Write your name on this cover sheet.

Some questions may be harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Please do not write in the boxes below.

I (xx/16)	II (xx/15)	III (xx/17)	IV (xx/10)	V (xx/16)	VI (xx/6)	Total (xx/80)

Name:

I Baggy bounds checking

Suppose that you use Baggy bounds checking to run the following C code, where X and Y are constant values. Assume that *slot_size* is 16 bytes, as in the paper.

```
1  char *p = malloc(40);
2  char *q = p + X;
3  char *r = q + Y;
4  *r = '\0';
```

For the following values of X and Y, indicate which line number will cause Baggy checking to abort, or NONE if the program will finish executing without aborting.

1. **[2 points]:** $X = 45, Y = 0$

2. **[2 points]:** $X = 60, Y = 0$

3. **[2 points]:** $X = 50, Y = -20$

4. **[2 points]:** $X = 70, Y = -20$

5. **[2 points]:** $X = 80, Y = -20$

6. **[2 points]:** $X = -5, Y = 4$

7. **[2 points]:** $X = -5, Y = 60$

8. **[2 points]:** $X = -10, Y = 20$

II Control hijacking

Consider the following C code:

```
struct foo {
    char buf[40];
    void (*f2) (struct foo *);
};

void
f(void)
{
    void (*f1) (struct foo *);
    struct foo x;

    /* .. initialize f1 and x.f2 in some way .. */

    gets(x.buf);
    if (f1)   f1(&x);
    if (x.f2) x.f2(&x);
}
```

There are three possible code pointers that may be overwritten by the buffer overflow vulnerability: $f1$, $x.f2$, and the function's return address on the stack. Assume that the compiler typically places the return address, $f1$, and x in that order, from high to low address, on the stack, and that the stack grows down.

9. [5 points]: Which of the three code pointers can be overwritten by an adversary if the code is executed as part of an XFI module?

10. [5 points]: What code could the adversary cause to be executed, if any, if the above code is executed as part of an XFI module?

11. [5 points]: What code could the adversary cause to be executed, if any, if the above code is executed under control-flow enforcement from lab 2 (no XFI)?

III OS protection

Ben Bitdiddle is running a web site using OKWS, with one machine running the OKWS server, and a separate machine running the database and the database proxy.

12. [12 points]: The database machine is maintained by another administrator, and Ben cannot change the 20-byte authentication tokens that are used to authenticate each service to the database proxy. This makes Ben worried that, if an adversary steals a token through a compromised or malicious service, Ben will not be able to prevent the adversary from accessing the database at a later time.

Propose a change to the OKWS design that would avoid giving tokens to each service, while providing the same guarantees in terms of what database operations each service can perform, without making any changes to the database machine.

13. [5 points]: Ben is considering running a large number of services under OKWS, and is worried he might run out of UIDs. To this end, Ben considers changing OKWS to use the same UID for several services, but to isolate them from each other by placing them in separate `chroot` directories (instead of the current OKWS design, which uses different UIDs but the same `chroot` directory). Explain, specifically, how an adversary that compromises one service can gain additional privileges under Ben's design that he or she cannot gain under the original OKWS design.

IV Capabilities and C

Ben Bitdiddle is worried that a plugin in his web browser could be compromised, and decides to apply some ideas from the "Security Architectures for Java" paper to sandboxing the plugin's C code using XFI.

Ben decides to use the capability model (§3.2 from the Java paper), and writes a function `safe_open` as follows:

```
int
safe_open(const char *pathname, int flags, mode_t mode)
{
    char buf[1024];
    snprintf(buf, sizeof(buf), "/safe-dir/%s", pathname);
    return open(buf, flags, mode);
}
```

which is intended to mirror Figure 2 from the Java paper. To allow his plugin's XFI module to access to files in `/safe-dir`, Ben allows the XFI module to call the `safe_open` function, as well as the standard `read`, `write`, and `close` functions (which directly invoke the corresponding system calls).

14. [10 points]: Can a malicious XFI module access files (i.e., read or write) outside of `/safe-dir`? As in the Java paper, let's ignore symbolic links and ".." components in the path name. Explain how or argue why not.

V Browser security

15. [6 points]: In pages of a site which has enabled ForceHTTPS, `<SCRIPT SRC=...>` tags that load code from an `http://.../` URL are redirected to an `https://.../` URL. Explain what could go wrong if this rewriting was not performed.

Ben Bitdiddle runs a web site that frequently adds and removes files, which leads to customers complaining that old links often return a 404 File not found error. Ben decides to fix this problem by adding a link to his site's search page, and modifies how his web server responds to requests for missing files, as follows (in Python syntax):

```
def missing_file(reqpath):
    print "HTTP/1.0 200 OK"
    print "Content-Type: text/html"
    print ""
    print "We are sorry, but the server could not locate file", reqpath
    print "Try using the <A HREF=/search>search function</A>."
```

16. [10 points]: Explain how an adversary may be able to exploit Ben's helpful error message to compromise the security of Ben's web application.

VI 6.858

We'd like to hear your opinions about 6.858, so please answer the following questions. (Any answer, except no answer, will receive full credit.)

17. [2 points]: How could we make the ideas in the course easier to understand?

18. [2 points]: What is the best aspect of 6.858?

19. [2 points]: What is the worst aspect of 6.858?

End of Quiz