Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2012

Quiz I

You have 80 minutes to answer the questions in this quiz. In order to receive credit you must answer the question as precisely as possible.

Some questions are harder than others, and some questions earn more points than others. You may want to skim 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 and legible. If we can't understand your answer, we can't give you credit!

Write your name and submission website username (typically your Athena username) on this cover sheet.

This is an open book, open notes, open laptop exam.
NO INTERNET ACCESS OR OTHER COMMUNICATION.

Please do not write in the boxes below.

I (xx/20)	II (xx/16)	III (xx/12)	IV (xx/20)	V (xx/10)	VI (xx/16)	VII (xx/6)	Total (xx/100)

Name:

Submission website username:

I Buffer overflows

Consider the following C program, where an adversary can supply arbitrary input on `stdin`. Assume no compiler optimizations, and assume a 32-bit system. In this example, `fgets()` never writes past the end of the 256-byte `buf`, and always makes it NULL-terminated.

```
int main() {
  char buf[256];
  fgets(buf, 256, stdin);
  foo(buf);
  printf("Hello world.\n");
}
```

1. [12 points]: Suppose the `foo` function is as follows:

```
void foo(char *buf) {
  char tmp[200];  // assume compiler places "tmp" on the stack

  // copy from buf to tmp
  int i = 0;      // assume compiler places "i" in a register
  while (buf[i] != 0) {
    tmp[i] = buf[i];
    i++;
  }
}
```

Which of the following are true? Assume there is an unmapped page both above and below the stack in the process's virtual memory.

(Circle True or False for each choice.)

A. True / False An adversary can trick the program to delete files on a system where the stack grows down.

B. True / False An adversary can trick the program to delete files on a system where the stack grows up.

C. True / False An adversary can trick the program to delete files on a system using Baggy bounds checking with `slot_size=16`. (Stack grows down.)

D. True / False An adversary can prevent the program from printing "Hello world" on a system using Baggy bounds checking with `slot_size=16`. (Stack grows down.)

E. True / False An adversary can trick the program to delete files on a system using terminator stack canaries for return addresses. (Stack grows down.)

F. True / False An adversary can prevent the program from printing "Hello world" on a system using terminator stack canaries for return addresses. (Stack grows down.)

2. [8 points]: Suppose the `foo` function is as follows:

```
struct request {
  void (*f)(void);  // function pointer
  char path[240];
};

void foo(char *buf) {
  struct request r;
  r.f = /* some legitimate function */;
  strcpy(r.path, buf);
  r.f();
}
```

Which of the following are true?

(Circle True or False for each choice.)

A. True / False An adversary can trick the program to delete files on a system where the stack grows down.

B. True / False An adversary can trick the program to delete files on a system where the stack grows up.

C. True / False An adversary can trick the program to delete files on a system using Baggy bounds checking with `slot_size=16`. Assume `strcpy` is compiled with Baggy. (Stack grows down.)

D. True / False An adversary can prevent the program from printing "Hello world" on a system using Baggy bounds checking with `slot_size=16`. Assume `strcpy` is compiled with Baggy. (Stack grows down.)

II OS sandboxing

Ben Bitdiddle is modifying OKWS to use Capsicum. To start each service, Ben's `okld` forks, opens the service executable binary, then calls `cap_enter()` to enter capability mode in that process, and finally executes the service binary. Each service gets file descriptors only for sockets connected to `okd`, and for TCP connections to the relevant database proxies.

3. [6 points]: Which of the following changes are safe now that the services are running under Capsicum, assuming the kernel implements Capsicum perfectly and has no other bugs?

(Circle True or False for each choice.)

A. True / False It is safe to run all services with the same UID/GID.

B. True / False It is safe to run services without `chroot`.

C. True / False It is safe to also give each service an open file descriptor for a per-service directory `/cores/servicename`.

Ben also considers replacing the `oklogd` component with a single log file, and giving each service a file descriptor to write to the log file.

4. [5 points]: What should `okld` do to ensure one service cannot read or overwrite log entries from another service? Be as specific as possible.

5. [5 points]: What advantages could an `oklogd`-based design have over giving each service a file descriptor to the log file?

III Network protocols

Ben Bitdiddle is designing a file server that clients connect to over the network, and is considering using either Kerberos (as described in the paper) or SSL/TLS (without client certificates, where users authenticate using passwords) for protecting a client's network connection to the file server. For this question, assume users choose hard-to-guess passwords.

6. [6 points]: Would Ben's system remain secure if an adversary learns the server's private key, but that adversary controls only a single machine (on the adversary's own home network), and does not collude with anyone else? Discuss both for Kerberos and for SSL/TLS.

7. [6 points]: Suppose an adversary learns the server's private key as above, and the adversary also controls some network routers. Ben learns of this before the adversary has a chance to take any action. How can Ben prevent the adversary from mounting attacks that take advantage of the server's private key (e.g., not a denial-of-service attack), and when will the system be secure? Discuss both for Kerberos and for SSL/TLS.

IV Static analysis

Would Yichen Xie's PHP static analysis tool for SQL injection bugs, as described in the paper, flag a potential error/warning in the following short but complete PHP applications?

8. [10 points]:

A. True / False The tool would report a potential error/warning in the following code:

```
function q($s) {
  return mysql_query($s);
}

$x = $_GET['id'];
q("SELECT .. $x");
```

B. True / False The tool would report a potential error/warning in the following code:

```
function my_validate() {
  return isnumeric($_GET['id']);
}

$x = $_GET['id'];
if (my_validate()) {
  mysql_query("SELECT .. $x");
}
```

9. [10 points]:

A. True / False The tool would report a potential error/warning in the following code:

```
mysql_query("SELECT .. $n");
```

B. True / False The tool would report a potential error/warning in the following code:

```
function check_arg($n) {
  $v = $_GET[$n];
  return isnumeric($v);
}

$x = $_GET['id'];
if (check_arg('id')) {
  mysql_query("SELECT .. $x");
}
```

V Runtime instrumentation

10. [10 points]:

Consider the following Javascript code:

```
function foo(x, y) {
  return x + y;
}

var a = 2;
eval("foo(a, a)");

var p_c = {
  k: 5,
  f: function() { return a + this.k; }
};
var kk = 'k';
p_c[kk] = 6;
p_c.f();
```

Based on the description in the paper by Sergio Maffeis et al, and based on lecture 9, what will be the FBJS rewritten version of this code, assuming the application-specific prefix is `p_`?

VI Browser security

Ben Bitdiddle is taking 6.858. Once he's done with his lab at 4:55pm, he rushes to submit it by going to `https://taesoo.scripts.mit.edu/submit/handin.py/student`, selecting his `lab`*N*`-handin.tar.gz` file, and clicking "Submit". The 6.858 submission web site also allows a student to download a copy of their past submission.

For your reference, when the user logs in, the submission web site stores a cookie in the user's browser to keep track of their user name. To prevent a user from constructing their own cookie, or arbitrarily changing the value of an existing cookie, the server includes a signature/MAC of the cookie's value in the cookie, and checks the signature when a new request comes in. Finally, users can log out by clicking on the "Logout" link, `https://taesoo.scripts.mit.edu/submit/handin.py/logout`, which clears the cookie.

Alyssa P. Hacker, an enterprising 6.858 student, doesn't want to do lab 5, and wants to get a copy of Ben's upcoming lab 5 submission instead. Alyssa has her own web site at `https://alyssa.scripts.mit.edu/`, and can convince Ben to visit that site at any point.

11. [16 points]: How can Alyssa get a copy of Ben's lab 5 submission?

Alyssa's attack should rely only on the Same-Origin Policy. Assume there are no bugs in any software, Ben's (and Taesoo's) password is unguessable, the cookie signature scheme is secure, etc.

VII 6.858

We'd like to hear your opinions about 6.858. Any answer, except no answer, will receive full credit.

12. [2 points]: This year we started using Piazza for questions and feedback. Did you find it useful, and how could it be improved?

13. [2 points]: What aspects of the labs were most time-consuming? How can we make them less tedious?

14. [2 points]: Are there other things you'd like to see improved in the second half of the semester?

End of Quiz