Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.893 Fall 2009

Quiz II

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/10)	II (xx/30)	III (xx/10)	IV (xx/10)
V (xx/10)	**VI (xx/20)**	**VII (xx/10)**	**Total (xx/100)**

Name:

I KeyKOS

1. [10 points]: Bob is running the privilege-separated Zoobar web site on a KeyNIX system, using code from lab 3. Suggest a way in which Bob can modify the Zoobar server-side code to take advantage of KeyKOS capabilities to improve the security of his site, in a way that he wouldn't be able to do on Linux.

II Network protocols

Bob logs into an Athena workstation, which uses Kerberos to obtain a ticket for bob@ATHENA.MIT.EDU, and then runs Bob's mail client, which contacts Bob's post office server to fetch new messages.

2. [10 points]: Alice doesn't want Bob to know about an upcoming event, which was announced to Bob via email. To this end, Alice plans to intercept Bob's communication with his post office server, and to pretend that Bob has no new mail. Alice can observe and modify all network packets sent by Bob. How does Kerberos prevent Alice from impersonating Bob's mail server? Be as specific as possible; explain how Bob can tell between Alice and the real mail server in terms of network packets.

Now, Alice wants to read Bob's email, and intercepts all network packets ever sent and received by Bob's workstation (which is the only computer that Bob uses). However, Alice does not know Bob's password to access Bob's post office server, and Bob's packets to and from the post office server are protected by Kerberos.

3. [10 points]: Suppose that after Bob reads and deletes all of his mail, Alice learns what Bob's password was. Describe how Alice can obtain Bob's past messages.

4. [10 points]: To prevent Alice from reading any more messages, Bob ensures that Alice cannot intercept any subsequent network traffic, and changes his Kerberos password. Could Alice still read Bob's mail after this? Explain why not or explain how.

III ForceHTTPS

5. [10 points]: Bob is developing a new web site, and wants to avoid the problems described in the ForceHTTPS paper. He uses HTTPS for all of his pages and marks all of his cookies "Secure". Assuming Bob made no mistakes, is there any reason for Bob's users to install the ForceHTTPS plugin and enable it for Bob's site? Explain why or why not.

IV BitLocker

6. [10 points]: Alice wants to make BitLocker run faster. She decides that computing a different IV_s for each sector (pg. 13 in the BitLocker paper) is needlessly expensive, and replaces it with the fixed value $E(K_{\text{AES}}, e(0))$ instead. Explain how an attacker may be able to leverage this change to obtain data from a stolen laptop that uses BitLocker in TPM-only mode.

V Tor

7. [10 points]: Bob is running a hidden service on top of Tor, and wants to know how frequently he should choose new introduction points. Bob cares about his identity not being exposed, and about the availability of his service. Help Bob make an informed choice by explaining the costs and benefits of rotating introduction points either more or less frequently.

VI BackTracker

8. [10 points]: Alice is using the VM-based BackTracker to analyze a compromised server, where an attacker obtained some user's password, logged in via SSH, exploited a buffer overflow in a setuid-root program to gain root access, and trojaned the login binary, all using high-control events that are still stored in the event logger. How can Alice figure out what *specific* vulnerability in the setuid-root program the attacker exploited? In what situations would this be possible or not possible?

9. [10 points]: The VM-based BackTracker system requires no modifications to the guest OS, but nonetheless makes assumptions about the guest OS. List assumptions that are critical to back-tracking attacks that used high-control events.

VII 6.893

We'd like to hear your opinions about 6.893, so please answer the following questions. (Any answer, except no answer, will receive full credit.)

10. [10 points]: If you could change one thing in 6.893, what would it be?

End of Quiz

6.858 Computer Systems Security

Fall 2014

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

6.858 Computer Systems Security

Fall 2014

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2011

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 on this cover sheet.

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Please do not write in the boxes below.

I (xx/20)	II (xx/10)	III (xx/16)	IV (xx/22)	V (xx/10)	VI (xx/16)	VII (xx/6)	Total (xx/100)

Name:

Username from handin site:

I XFI

Consider the following assembly code, which zeroes out 256 bytes of memory pointed to by the EAX register. This code will execute under XFI. XFI's allocation stack is not used in this code.

You will need fill in the verification states for this code, which would be required for the verifier to check the safety of this code, along the lines of the example shown in Figure 4 of the XFI paper. Following the example from the paper, possible verification state statements include:

```
valid[regname+const, regname+const)
origSSP = regname+const
retaddr = Mem[regname]
```

where `regname` and `const` are any register names and constant expressions, respectively. Include all verification states necessary to ensure safety of the subsequent instruction, and to ensure that the next verification state is legal.

```
1   x86 instructions        Verification state
2
3   mrguard(EAX, 0, 256)
4                           (1)
5   ECX := EAX      # current pointer
6   EDX := EAX+256  # end of 256-byte array
7
8 loop:
9                           (2)
10  Mem[ECX] := 0
11  ECX := ECX+4
12                          (3)
13  if ECX+4 > EDX, jmp out
14                          (4)
15  jmp loop
16
17 out:
18  ...
```

1. **[5 points]:** What are the verification states needed at location marked (1)?

2. **[5 points]:** What are the verification states needed at location marked (2)?

3. **[5 points]:** What are the verification states needed at location marked (3)?

4. **[5 points]:** What are the verification states needed at location marked (4)?

II ForceHTTPS

5. [10 points]: Suppose `bank.com` uses and enables ForceHTTPS, and has a legitimate SSL certificate signed by Verisign. Which of the following statements are true?

A. True / False ForceHTTPS prevents the user from entering their password on a phishing web site impersonating `bank.com`.

B. True / False ForceHTTPS ensures that the developer of the `bank.com` web site cannot accidentally load Javascript code from another web server using `<SCRIPT SRC=...>`.

C. True / False ForceHTTPS prevents a user from accidentally accepting an SSL certificate for `bank.com` that's not signed by any legitimate CA.

D. True / False ForceHTTPS prevents a browser from accepting an SSL certificate for `bank.com` that's signed by a CA other than Verisign.

III Zoobar security

Ben Bitdiddle is working on lab 2. For his privilege separation, he decided to create a separate database to store each user's zoobar balance (instead of a single database called `zoobars` that stores everyone's balance). He stores the zoobar balance for user `x` in the directory `/jail/zoobar/db/zoobars.x`, and ensures that usernames cannot contain slashes or null characters. When a user first registers, the login service must be able to create this database for the user, so Ben sets the permissions for `/jail/zoobar/db` to 0777.

6. [4 points]: Explain why this design may be a bad idea. Be specific about what an adversary would have to do to take advantage of a weakness in this design.

Ben Bitdiddle is now working on lab 3. He has three user IDs for running server-side code, as suggested in lab 2 (ignoring transfer logging):

- User ID 900 is used to run dynamic python code to handle HTTP requests (via zookfs). The database containing user profiles is writable only by uid 900.
- User ID 901 is used to run the authentication service, which provides an interface to obtain a token given a username and password, and to check if some token for a username is valid. The database containing user passwords and tokens is stored in a DB that is readable and writable only by uid 901.
- User ID 902 is used to run the transfer service, which provides an interface to transfer zoobar credits from user *A* to user *B*, as long as a token for user *A* is provided. The database storing zoobar balances is writable only by uid 902. The transfer service invokes the authentication service to check whether a token is valid.

Recall that to run Python profile code for user *A*, Ben must give the profile code access to *A*'s token (the profile code may want to transfer credits to visitors, and will need this token to invoke the transfer service).

To support Python profiles, Ben adds a new operation to the authentication service's interface, where the caller supplies an argument `username`, the authentication service looks up the profile for `username`, runs the profile's code with a token for `username`, and returns the output of that code.

7. [4 points]: Ben discovers that a bug in the HTTP handling code (running as uid 900) can allow an adversary to steal zoobars from any user. Explain how an adversary can do this in Ben's design.

8. [8 points]: Propose a design change that prevents attackers from stealing zoobars even if they compromise the HTTP handling code. Do not make any changes to the authentication or transfer services (i.e., code running as uid 901 and 902).

IV Baggy bounds checking

Consider a system that runs the following code under the Baggy bounds checking system, as described in the paper by Akritidis et al, with slot_size=16:

```
1  struct sa {
2    char buf[32];
3    void (*f) (void);
4  };
5
6  struct sb {
7    void (*f) (void);
8    char buf[32];
9  };
10
11 void handle(void) {
12   printf("Hello.\n");
13 }
14
15 void buggy(char *buf, void (**f) (void)) {
16   *f = handle;
17   gets(buf);
18   (*f) ();
19 }
20
21 void test1(void) {
22   struct sa x;
23   buggy(x.buf, &x.f);
24 }
25
26 void test2(void) {
27   struct sb x;
28   buggy(x.buf, &x.f);
29 }
30
31 void test3(void) {
32   struct sb y;
33   struct sa x;
34   buggy(x.buf, &y.f);
35 }
36
37 void test4(void) {
38   struct sb x[2];
39   buggy(x[0].buf, &x[1].f);
40 }
```

Assume the compiler performs no optimizations and places variables on the stack in the order declared, the stack grows down (from high address to low address), that this is a 32-bit system, and that the address of `handle` contains no zero bytes.

9. [6 points]:

A. True / False If function `test1` is called, an adversary can construct an input that will cause the program to jump to an arbitrary address.

B. True / False If function `test2` is called, an adversary can construct an input that will cause the program to jump to an arbitrary address.

C. True / False If function `test3` is called, an adversary can construct an input that will cause the program to jump to an arbitrary address.

D. True / False If function `test4` is called, an adversary can construct an input that will cause the program to jump to an arbitrary address.

For the next four questions, determine what is the minimum number of bytes that an adversary has to provide as input to cause this program to likely crash, when running different test functions. Do not count the newline character that the adversary has to type in to signal the end of the line to `gets`. Recall that `gets` terminates its string with a zero byte.

10. [4 points]: What is the minimum number of bytes that an adversary has to provide as input to likely cause a program running `test1` to crash?

11. [4 points]: What is the minimum number of bytes that an adversary has to provide as input to likely cause a program running `test2` to crash?

12. [4 points]: What is the minimum number of bytes that an adversary has to provide as input to likely cause a program running `test3` to crash?

13. [4 points]: What is the minimum number of bytes that an adversary has to provide as input to likely cause a program running `test4` to crash?

V Browser security

The same origin policy generally does not apply to images or scripts. What this means is that a site may include images or scripts from any origin.

14. [3 points]: Explain why including images from other origins may be a bad idea for user privacy.

15. [3 points]: Explain why including scripts from another origin can be a bad idea for security.

16. [4 points]: In general, access to the file system by JavaScript is disallowed as part of JavaScript code sandboxing. Describe a situation where executing JavaScript code will lead to file writes.

VI Static analysis

Consider the following snippet of JavaScript code:

```
1  var P = false;
2
3  function foo() {
4    var t1 = new Object();
5    var t2 = new Object();
6    var t = bar(t1, t2);
7    P = true;
8  }
9
10 function bar(x, y) {
11   var r = new Object();
12   if (P) {
13     r = x;
14   } else {
15     r = y;
16   }
17
18   return r;
19 }
```

A flow sensitive pointer analysis means that the analysis takes into account the order of statements in the program. A flow insensitive pointer analysis does not consider the order of statements.

17. [4 points]: Assuming no dead code elimination is done, a flow-insensitive pointer analysis (i.e., one which does not consider the control flow of a program) will conclude that variable `t` in function `foo` may point to objects allocated at the following line numbers:

A. True / False Line 1

B. True / False Line 4

C. True / False Line 5

D. True / False Line 11

18. [4 points]: Assuming no dead code elimination is done, a flow-sensitive pointer analysis (i.e., one which considers the control flow of a program) will conclude that variable `t` in function `foo` may point to objects allocated at the following line numbers:

A. True / False Line 1

B. True / False Line 4

C. True / False Line 5

D. True / False Line 11

19. [2 points]: At runtime, variable `t` in function `foo` may only be observed pointing to objects allocated at the following line numbers:

A. True / False Line 1

B. True / False Line 4

C. True / False Line 5

D. True / False Line 11

20. [2 points]: Do you think a sound analysis that supports the `eval` construct is going to be precise? Please explain.

21. [4 points]: What is one practical advantage of the bottom-up analysis of the call graph described in the PHP paper by Xie and Aiken (discussed in class)?

VII 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.)

22. [2 points]: How could we make the ideas in the course easier to understand?

23. [2 points]: What is the best aspect of 6.858 so far?

24. [2 points]: What is the worst aspect of 6.858 so far?

End of Quiz

6.858 Computer Systems Security

Fall 2014

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

6.858 Computer Systems Security

Fall 2014

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:

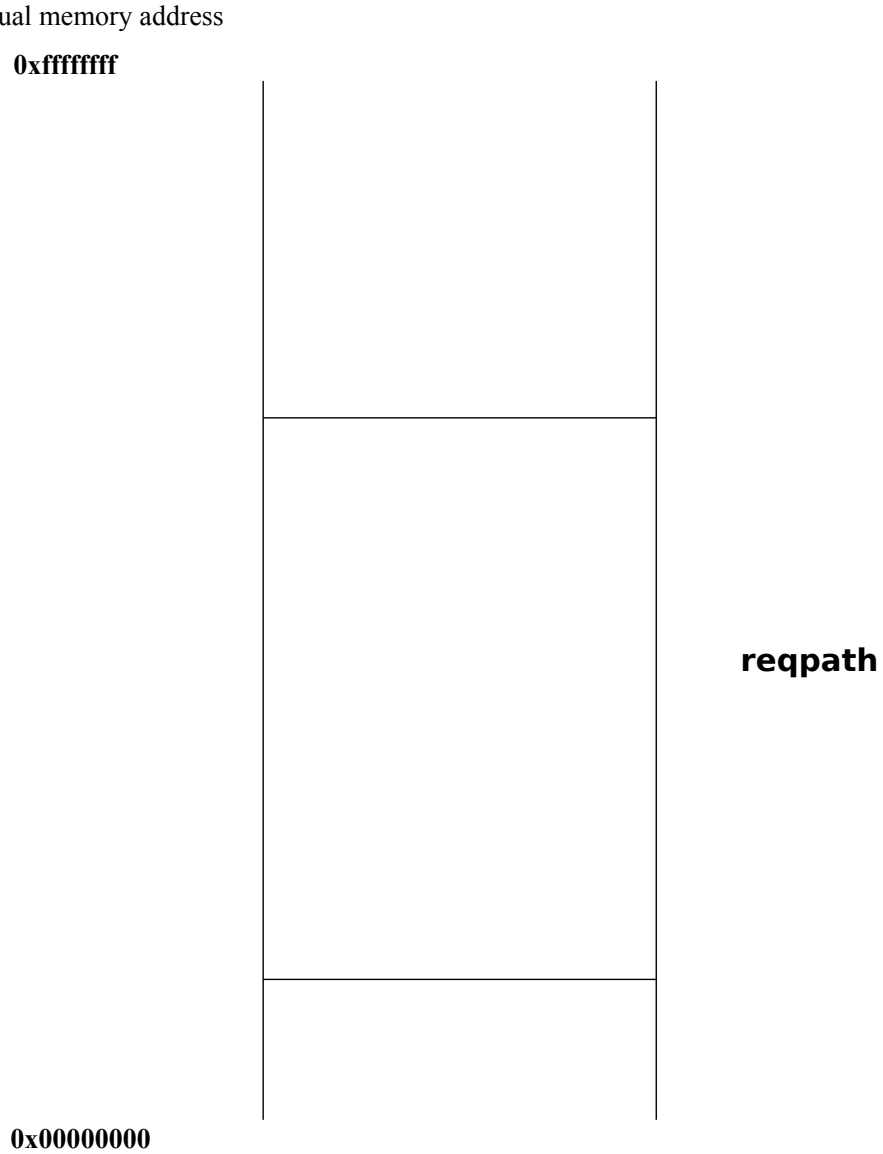
```
static void process_client(int fd)
{
    static char env[8192];  /* static variables are not on the stack */
    static size_t env_len;
    char reqpath[2048];
    const char *errmsg;
    int i;

    /* 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.

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:

```
 1.  def buy_handler(cookie, param):
 2.    print "Content-type: text/html\r\n\r\n",
 3.
 4.    user = check_cookie(cookie)
 5.    if user is None:
 6.      print "Please log in first"
 7.      return
 8.
 9.    book = param['book']
10.    if in_stock(book):
11.      ship_book(book, user)
12.      print "Order succeeded"
13.    else:
14.      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

6.858 Computer Systems Security

Fall 2014

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2014

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/12)	II (xx/14)	III (xx/14)	IV (xx/12)	V (xx/6)	VI (xx/22)	VII (xx/14)	VIII (xx/6)	Total (xx/100)

Name:

Submission website email address:

I Baggy Bounds and Buffer Overflows

1. [6 points]: At initialization time, a baggy bounds system on a 32-bit machine is supposed to set all of the bounds table entries to 31. Suppose that, in a buggy implementation with a `slot_size` of 32 bytes, bounds table initialization is improperly performed, such that random entries are incorrectly set to 1.

Suppose that a networked server uses an uninstrumented library to process network messages. Assume that this library has no buffer overflow vulnerabilities (e.g., it never uses unsafe functions like `gets()`). However, the server *does* suffer from the bounds table initialization problem described above, and the attacker can send messages to the server which cause the library to dynamically allocate and write an attacker-controlled amount of memory using uninstrumented code that looks like this:

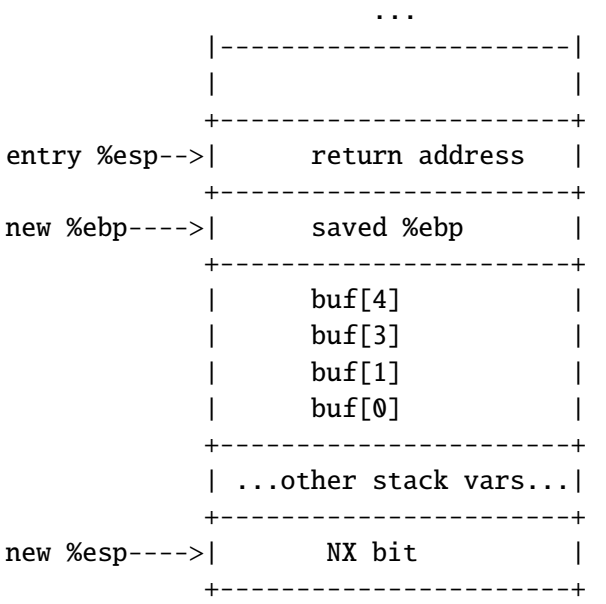
```
// N is the buffer size that the
// attacker gets to pick.
char *p = malloc(N);
for (int i = 0; i < N/4; i++, p += 4) {
  *p = '\a';
  *(p+1) = '\b';
  *(p+2) = '\c';
  *(p+3) = '\d';
}
```

Assume that the server uses a buddy memory allocator with a maximum allocation size of 2^{16} (i.e., larger allocations fail). What is the smallest N that the attacker can pick that will definitely crash the server? Why will that N cause a crash?

2. [6 points]: Modern CPUs often support NX ("no execute") bits for memory pages. If a page has its NX bit set to 1, then the CPU will not run code that resides in that page.

NX bits are currently enforced by the OS and the paging hardware. However, imagine that programs execute on a machine whose OS and paging hardware do not natively support NX. Further imagine that a compiler wishes to implement NX at the software level. The compiler associates a software-manipulated NX bit with each memory page, placing it at the bottom (i.e., the lowest address) of each 4KB page.

The compiler requires that all application-level data structures be at most 4095 bytes large. The compiler allocates each stack frame in a separate page, and requires that a stack frame is never bigger than a page. A stack frame might look like the following:



such that, as shown in the sample code above, an overflow attack in the frame will not overwrite the frame's NX bit.

The compiler also associates NX bits with each normal code page. The NX bit for a stack frame is set to "non-executable", and the NX bit for a normal code page is set to "executable".

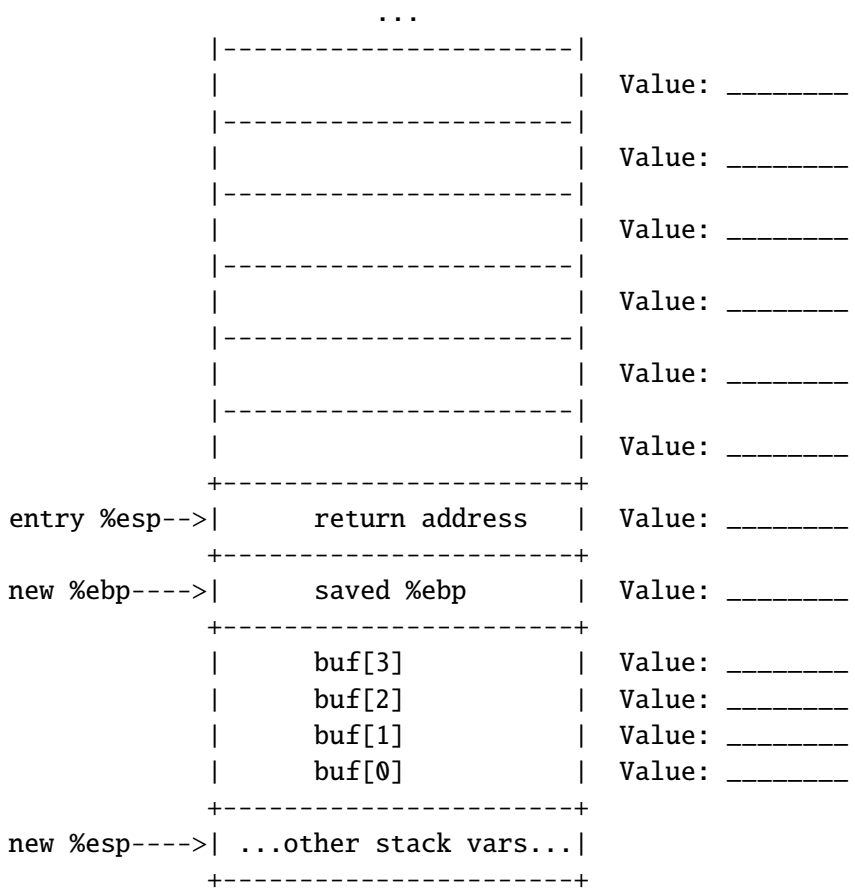
The compiler instruments updates to the program counter such that, whenever the PC migrates to a new page, the program checks the NX bit for the page. If the bit indicates that the page is non-executable, the program throws an exception.

Describe how a buffer overflow attack can still overwrite NX bits.

II Stack Canaries and Return-Oriented Programming

3. [4 points]: Stack canaries live in an area of memory that programs can read as well as write. In the typical buffer overflow attack (e.g., via the `gets()` function), what prevents an attacker from simply reading the canary value and then placing that canary value in the overflow payload?

4. [10 points]: In the first part of a BROP attack, the attacker must find gadgets that pop entries from the stack and store them into attacker-selected registers. Suppose that the attacker has already found the address of a stop gadget and a trap value (i.e., a memory value which, if accessed, causes a fault). In the stack diagram below, depict what a buffer overflow should write on the stack to identify pop gadgets which pop exactly two things from the stack e.g., `pop rdi; pop rsi; ret;`. If it doesn't matter what goes in a particular memory location, put "Doesn't mattter". To represent the values for the stop gadget and the trap, simply write "stop" or "trap". To represent the address of a candidate pop gadget, write "probe".



III OKWS and OS Security

Suppose Unix did not provide a way of passing file descriptors between processes, but still allowed inheriting file descriptors from a parent on `fork` and `exec`.

5. [4 points]:

What aspects of the OKWS design would break without file descriptor passing?

(Circle True or False for each choice.)

A. True / False It would be impossible for services to send messages to `oklogd`.

B. True / False It would be impossible for services to get a TCP connection to a database proxy.

C. True / False It would be impossible for services to get a TCP connection to the client web browser.

D. True / False It would be impossible for `okd` to run as a non-root user.

Consider the following Python code for a program that might run every night as root on a Unix machine to clean up old files in `/tmp`. The Python function `os.walk` returns a list of subdirectories and filenames in those subdirectories. It ignores `"."` and `".."` names. As a reminder, a Unix filename cannot contain / or NULL bytes, and `os.unlink` on a symbolic link removes the symbolic link, not the target of the symbolic link.

```
def cleanup():
  ## Construct a list of files under /tmp that are over 2 days old.
  files = []
  for (dirname, _, filenames) in os.walk('/tmp'):
    for filename in filenames:
      fn = dirname + '/' + filename
      if os.path.getmtime(fn) < time.time() - 2 * 86400:
        files.append(fn)

  for fn in files:
    os.unlink(fn)
```

6. [10 points]:

Explain how an adversary could take advantage of this program to delete `/etc/passwd`.

IV Native Client

Answer the following questions about how Native Client works on 32-bit x86 systems, according to the paper "Native Client: A Sandbox for Portable, Untrusted x86 Native Code."

7. [6 points]: Which of the following statements are true?

(Circle True or False for each choice.)

A. True / False The Native Client compiler is trusted to generate code that follows Native Client's constraints.

B. True / False The Native Client validator ensures that no instruction spans across a 32-byte boundary.

C. True / False The Native Client service runtime is checked using the validator to ensure its code follows the constraints.

D. True / False Native Client requires additional instructions before every direct jump.

E. True / False Native Client requires additional instructions before every indirect jump.

F. True / False Native Client requires additional instructions before every memory access.

8. [6 points]:

For the following x86 code, indicate whether Native Client's validator would allow it (by writing **ALLOW**), assuming the parts after `...` are valid, or circle the *first* offending instruction that causes the validator to reject the code.

```
   10000:       83 e0 2e              and    $0x2e,%eax
   10003:       40                    inc    %eax
   10004:       01 ca                 add    %ecx,%edx
   10006:       4a                    dec    %edx
   10007:       eb fa                 jmp    0x10003
   10009:       b9 ef be ad de        mov    $0xdeadbeef,%ecx
   1000e:       8b 39                 mov    (%ecx),%edi
   10010:       8b 35 ef be ad de     mov    0xdeadbeef,%esi
   10016:       8b 66 64              mov    0x64(%esi),%esp
   10019:       5b                    pop    %ebx
   1001a:       8b 58 05              mov    0x5(%eax),%ebx
   1001d:       83 e0 e0              and    $0xffffffe0,%eax
   10020:       ff e0                 jmp    *%eax
   10022:       f4                    hlt
...
```

V Symbolic execution

Consider the following Python program running under the concolic execution system from lab 3, where `x` is a concolic integer that gets the value 0 on the first iteration through the loop:

```
def foo(x):
  y = x + 7
  if y > 10:
    return 0
  if y * y == 256:
    return 1
  if y == 7:
    return 2
  return 3
```

9. [6 points]:

After running `foo` with an initial value of `x=0`, what constraint would the concolic execution system send to Z3 for the second `if` statement?

VI Web security

10. [8 points]: Suppose that a user visits a mashup web page that simultaneously displays a user's favorite email site, ecommerce site, and banking site. Assume that:

- The email, ecommerce, and banking sites allow themselves to be placed in iframes (e.g., they don't prevent this using `X`-Frame-Options headers).
- Each of those three sites is loaded in a separate iframe that is created by the parent mashup frame.
- Each site (email, ecommerce, banking, and mashup parent) come from a different origin with respect to the same origin policy. Thus, frames cannot directly tamper with each other's state.

Describe an attack that the mashup frame can launch to steal sensitive user inputs from the email, ecommerce, or banking site.

11. [8 points]: Each external object in a web page has a type. That type is mentioned in the object's HTML tag (e.g., an image should have an <img> tag like <img src="http://x.com/x.gif">). An object's type is also described as a MIME type in its HTTP response (e.g., Content-type: "image/gif").

These two kinds of type specifications can mismatch due to programmer error, misconfiguration, or malice. For example, for the tag <img src="http://x.com/x.gif">, the server might return the MIME type "text/css".

Suppose that, in the case of a type mismatch, the browser uses the MIME type in the HTTP response to determine how to interpret an object. For example, if X's frame tries to load the MIME-type-less tag <img src="http://Y/file">, and Y's server returns a MIME type of "text/css", the browser will interpret the fetched object as CSS in X's frame, even though the object is embedded in X's frame as an <img> tag.

Why is this a bad security policy?

12. [6 points]: In a SQL injection attack, attacker-controlled input is evaluated in the context of a SQL query, resulting in malicious SQL statements executing over sensitive data. Ur/Web allows web applications to directly embed SQL queries in a page; furthermore, those queries may contain information that originates from the user or an untrusted source. Why is this safe in Ur/Web?

VII Network security and Kerberos

Ben Bitdiddle is concerned about the sequence number guessing attack that Steve Bellovin described in section 2 of his paper, where an adversary can spoof a TCP connection to a server from an arbitrary source IP address, and send data on that connection.

Ben implements the following strategy that his server will use for choosing the initial sequence number ISN_s of an incoming TCP connection:

$$\text{ISN}_s = \text{ISN}_{\text{original}} \oplus \text{IP}_{\text{src}} \oplus \text{IP}_{\text{dst}} \oplus (\text{Port}_{\text{src}} || \text{Port}_{\text{dst}}) \tag{1}$$

where $\oplus$ refers to the XOR operation and $||$ refers to concatenation; the IP fields being XORed refer to the 32-bit IP addresses of the source and destination of the TCP connection; and the Port fields refer to the 16-bit source and destination ports. Assume $\text{ISN}_{\text{original}}$ increments by 64 for each new incoming connection, and initially starts at some random value.

13. [8 points]:

Explain how an adversary could still launch a sequence-number-guessing attack against Ben's server with a small number of tries.

Suppose the KDC server at MIT developed a subtle hardware problem, where the random number generator became highly predictable (e.g., it would often produce the same result when asked for a "random" number).

14. [6 points]:

How could an adversary leverage this weakness to access some user's data on a file server that uses Kerberos for authentication? Describe the minimal amount of additional access the adversary might need to mount such an attack. Assume the file server ignores IP addresses in Kerberos tickets, and that the keys of all principals were generated *before* the server developed this hardware problem.

VIII 6.858

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

15. [2 points]: We introduced a new lab on symbolic execution this semester (lab 3). How would you suggest improving this lab in future semesters?

16. [2 points]: Are there other things you'd like to see improved in the second half of the semester?

17. [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

6.858 Computer Systems Security

Fall 2014

Quiz 1: Next Wednesday, Walker, during regular class time

- L1: Security Basics
- L2: Baggy Bounds
- L3: Hacking Blind
- L4: OKWS
- L5: Guest Lec.
- L6: Capsicum & Conf. Deputy
- L7: NaCl
- L8: OWASP & Tangled Web
- L9: Django & CSRF
- L10: Klee & Symbolic Exec.
- L11: Ur/Web
- L12: TCP/IP
- L13: Kerberos
- L14: ForceHTTPs
- --
- Lab1: Buffer Overf.
- Lab2: Priv. Sep.
- Lab3: Concolic Exec.

How to study for Quiz 1

- Do old exams.
- Review papers and lecture notes together.
- Youtube videos available for each lecture.
- Student questions and lecture questions for each lecture are available on the submission page.

L1: Intro

Policy

- the goal you want to achieve
- negative goals (e.g. “system should not crash”)
- CIA (confidentiality, integrity, availability)

Threat model

- attacker’s capabilities (access to source? physical access? etc.)

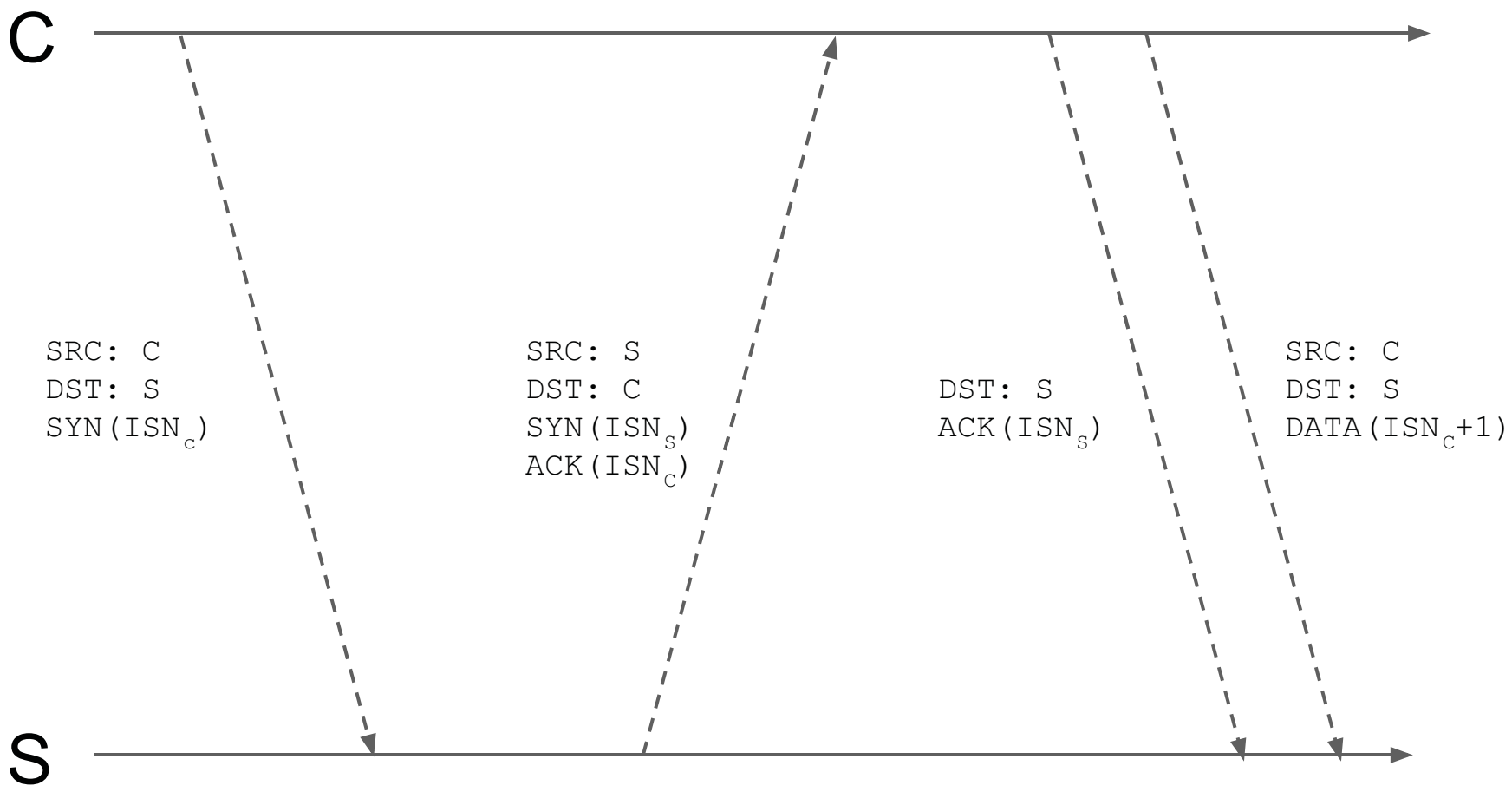
Mechanism

- technology to enforce policy (unix permissions, capabilities, etc.)

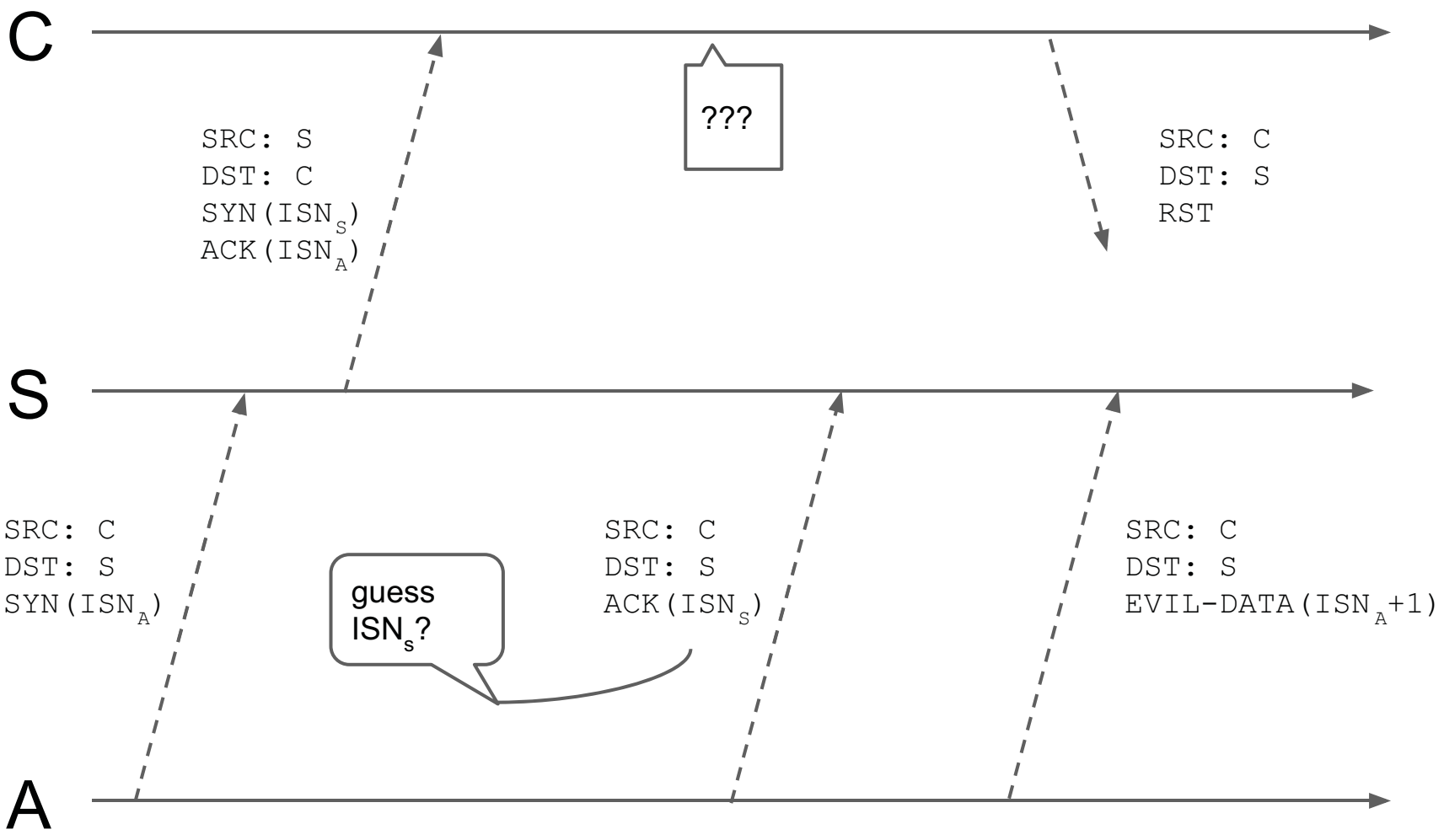
L12: TCP/IP and Network Security

1. TCP Handshake
2. Spoofing connections
3. DOS attacks (RST, SYN Flooding)
4. DNS, ARP, DHCP, BGP, ...
5. Old Quiz Problems

[L12, 1/5] TCP Handshake



[L12: 2/5] Connection Spoofing



[L12: 2/5] Connection Spoofing

Problem

- Easy for A to guess ISN_s
  - $ISN_s \leftarrow (ISN_s)_{old} + 250000 * \Delta t_{sec}$
  - A can get $(ISN_s)_{old}$ by sending a legal $SYN(ISN_A)$ packet to S.

Solution

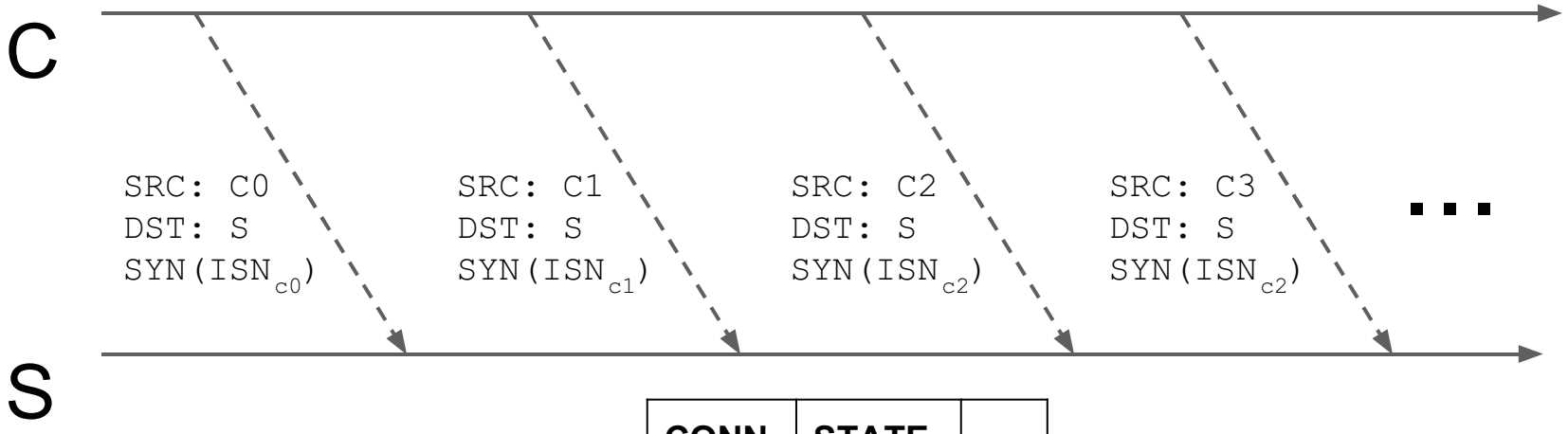
- Make it hard for A to guess ISN_S
  - $ISN_s \leftarrow ISN_{oldstyle} + F(ipaddr_{src}, portn_{src}, ipaddr_{dst}, portn_{dst}, key)$
  - F a cryptographic hash function (e.g. SHA1)

[L12: 2/5] Connection Spoofing

- If A knows SN_C:
  - Reset connection: RST($SN_{C\mp}$)
  - Inject data into existing stream: DATA(SN_C)

[L12: 3/5] DOS Attacks

SYN Floods



CONN	STATE	...
C0	waiting	
C1	waiting	
C2	waiting	
...	...	

[L12: 3/5] DOS Attacks

Problem with SYN Floods

- Server must keep state for every incoming SYN packet

Solution: SYN Cookies

- Don’t keep state for new incoming SYN packets!
    - Reply w/ SYN-ACK as normal.
- $ISN_s \leftarrow F_{key}(ipaddr_{src}, portn_{src}, ipaddr_{dst}, portn_{dst}, timestamp) \,||\, timestamp$
    - timestamp on scale of minutes

[L12: 4/5] DNS, ARP, DHCP, BGP

See lecture notes and paper for:

- DNS response spoofing
- DNS amplification
- ARP spoofing
- DHCP spoofing
- others

[L12 5/5] Quiz 1, 2013, Problem 6

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 ISNs 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?

L13: Kerberos

1. Overview
2. Practice problems

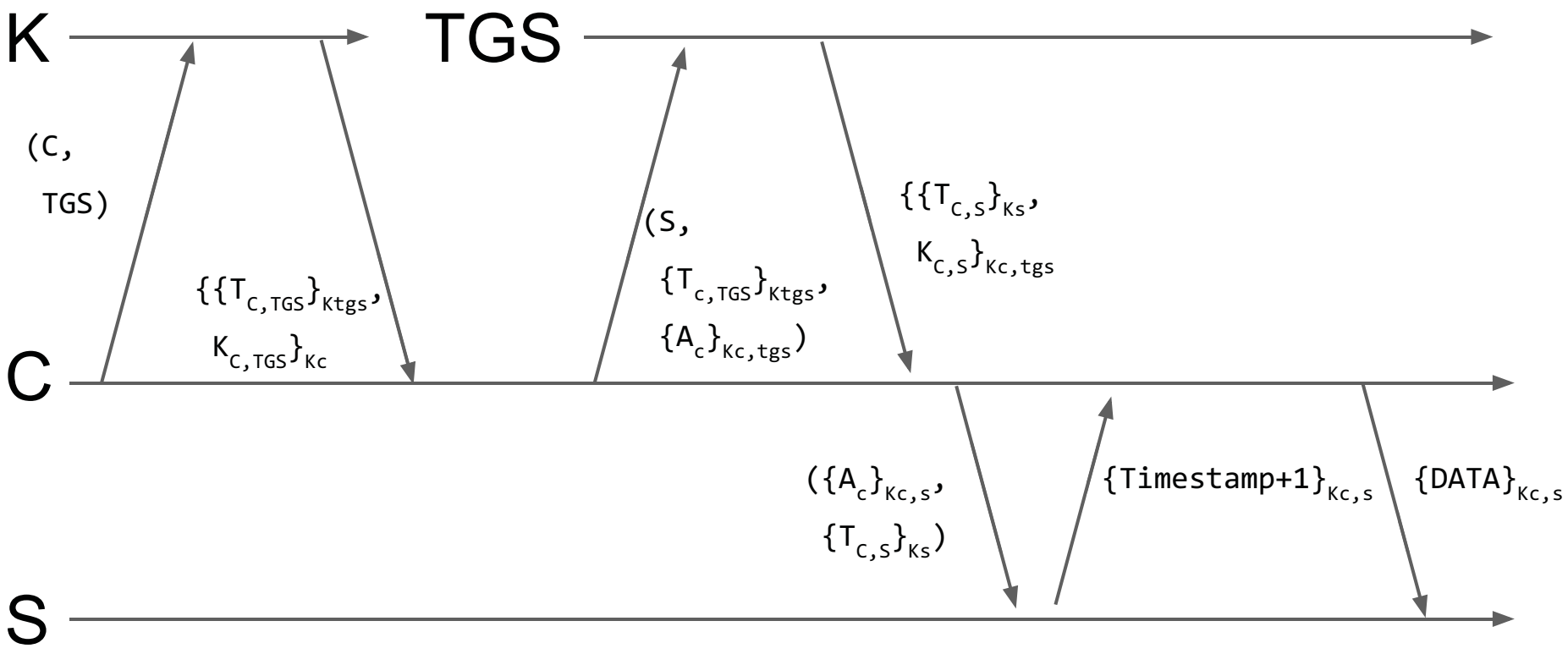
For fun, see:

http://web.mit.edu/kerberos/dialogue.html

L13: Kerberos Overview

$T_{c,s}$ = {s, c, addr, timestamp, life, $K_{c,s}$ }

A_c = {c, addr, timestamp}$_{Kc,s}$



L13: 2009 Quiz 2 Problem 2

Bob logs into an Athena workstation, which uses Kerberos to obtain a ticket for bob@ATHENA.MIT.EDU, and then runs Bob's mail client, which contacts Bob's post office server to fetch new messages.

2. Alice doesn't want Bob to know about an upcoming event, which was announced to Bob via email. To this end, Alice plans to intercept Bob's communication with his post office server, and to pretend that Bob has no new mail. Alice can observe and modify all network packets sent by Bob. How does Kerberos prevent Alice from impersonating Bob's mail server? Be as specific as possible; explain how Bob can tell between Alice and the real mail server in terms of network packets.

L13: 2009 Quiz 2 Problem 3

Now, Alice wants to read Bob's email, and intercepts all network packets ever sent and received by Bob's workstation (which is the only computer that Bob uses). However, Alice does not know Bob's password to access Bob's post office server, and Bob's packets to and from the post office server are protected by Kerberos.

3. Suppose that after Bob reads and deletes all of his mail, Alice learns what Bob's password was. Describe how Alice can obtain Bob's past messages.

L13: 2012 Quiz 1 Problem 6

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. 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~~.

L13: 2013 Quiz 1 Problem 7

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. 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.

L13: 2013 Quiz 1 Problem 8

8. 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.

[L13] SSL/HTTPs/ForceHTTPs

- Covered in lecture next Monday

- HTTPs handles authentication and encryption for web

- Still problematic in some cases
  - ForceHTTPs fixes some browser quirks

6.858 Quiz Review

Blind ROP

6.858 Quiz Review

OWASP/Tangled Web

Security Risks

- Open Redirectors
- Vulnerable Software
- CSRF/XSRF
- Function Level Access Control
- Data Exposure

Security Risks

- Misconfigurations
- Insecure Direct Object Reference
- XSS
- Broken Session Management
- Injections

Same Origin Policy

- Dependent on Protocol and Host
- Blocks Execution and Access
- Not foolproof
  - Java
  - Flash

Same Origin Policy

- Dependent on Protocol and Host
- Blocks Execution and Access
- Not foolproof
  - Java
  - Flash

Content Security Policy

- Specify Constraints on Loading
- Prevent “malicious” scripts
- Still Vulnerable

OKWS

- Separate each service into a distinct process
- Each service runs as a sepreate UID and GID
- Each service is chrooted
- Database proxy is used to enforce query structure

OKWS

- Okld launches all services, creates sockets, run as root
- No protection against XXS
- Pubd stores all templates
- Oklogd keeps log of all activities, chroot to its own sandbox

DB Proxy

- Each service is given a subset of possible queries it is allowed to make
- Each service passes a token to the db proxy with it's query that proves its identity

DAC

- Each object has a set of permissions
- Applications set permissions on objects
- Privileges are checked when a program access an object

DAC Problems

- Only Root can create new users
- Sometimes hard to customize permissions
- Some objects don’t have clear configurable access control list

MAC

- Users can’t change the policy
- Tries to enforce military classified levels
- Policy is separated from application code

MAC Problems

- The system controls who can access a newly created object
- Hard to customize permissions

Capabilities

- Token based access
- Can have different tokens for read vs write
- Each object accessed by the file handler
- Tokens are passed down to forked processes

Caps Mode

- Once a process enters it can’t leave caps mode
- Name space is restricted, can’t use “..”
- Unix permissions still apply
- Allowed operations stored in file descriptor

Capabilities

Advantages:

- Any process can create a new sandbox (even a sandbox)
- Very easy to customize access

Disadvantages:

- No global namespace
- Hard to keep track of who has access to persistent files

UrWeb

- Only one programming language is used for the entire application
- All objects passed between different parts of the application are strongly typed
- Instead of Http requests, clients call typed functions that are run on the server atomically

Strongly Typed

Strongly typed objects are objects that can only be used in specific functions of the application. This prevents XXS because string input from the user cannot be transformed into HTML or javascript unless the author of the application explicitly allows it.

6.858 Computer Systems Security

Fall 2014