ECE 458 – Computer Security (Spring 2013)

Instructor:	Mahesh V. Tripunitara, tripunit@uwaterloo.ca, x32864, DC 2532
Prerequisite:	Linux “capability” — you, as a team of 2, will need to install a version of Linux within a VM, and work on some hands-on assignments.
Textbook:	None. I will make chapters from books, notes and papers available to you.

We will consider various aspects of computer security. The split is roughly equally into 4 topics: (a) Authorization, (b) Programming with security in mind, (c) Non-cryptographic aspects of network security, and, (d) Cryptographic Engineering.

In (a), we will consider Access Control Lists (ACLs) and Unix user-ids, and why they are difcult to manage and intuit. In (b), we will consider buffer overows, format-string errors, SQL injection and cross-site scripting. In (c), we will consider Firewalls and Intrusion Detection. In (d), we will consider how to put cryptographic primitives such as public key encryption and cryptographic hashing together to build higher-level protocols that achieve some security objectives, such as key-exchange and authentication.

Grading: There will be three components to the marks. (1) Lecture attendance as measured by occasional reviews (15%), (2) Assignments (35%), (3) Final Exam (50%).

Sample Assignments: A collection of assignments from a previous offering follows. These serve as a good indication of the assignments and content of this course.

ECE 458, Winter 2012
Assignment 1 - Due Sat, Jan 14, 11:59:59 PM

- Your submissions need to be typeset and in pdf.
- Use the dropbox for Assignment 1 at Learn.
- Turn in only one submission for your group of 2.
- Note that the group is for the entire term. You are not allowed to change group memberships partway through the term.
- Put the names of both members of the group on the submission.
- If you prefer to work on the assignments for this course on your own (group of 1), that's fine as well.

1. From a news article in the context of protection:

> Pakistan's 60 plus [nuclear] warheads are believed to be stored separately from their delivery systems, with the nuclear cores removed from their detonators. The weapons are dispersed in as many as six separate locations, most to the South of the capital. And Pakistani officials say the weapons are fitted with code locks, with at least two people required to authenticate the codes before they can be released from storage.

(a) (5) Which of the principles of Saltzer & Schroeder does Pakistan use, according to the above passage? Provide a brief (max. 3 sentences) explanation for why you feel a particular principle is in use.

(b) (5) Which other principles(s) from S & S do you recommend that they use? Provide a brief (max. 3 sentences) explanation for each.

For the following questions, you need to work with the SEED Ubuntu VM installation. See `http://www.cis.syr.edu/~wedu/seed/lab_env.html`. We will use the Linux stuff only (and none of the Minix stuff). Perform the following:

- Download & install VMware Workstation/Fusion as appropriate to you.
- Download the `SEEDUbuntu9_August_2010.tar.gz` file from `http://www.cis.syr.edu/~wedu/seed/lab_env.html`. Open it with your VMware installation.
- Play around with a shell and other programs. Make sure you consult the `user manual` that is at `http://www.cis.syr.edu/~wedu/seed/lab_env.html` .

2. In your VM Ubuntu installation, do the following. As root, copy `/bin/zsh` and `/bin/bash` to `/tmp`. Then, again as root, issue the following commands so that the setuid bit gets set in each of their permission bits.

```
chmod u+s /tmp/zsh
chmod u+s /tmp/bash
```

So now, both `/tmp/zsh` and `/tmp/bash` are setuid-root. You need to first **(a)** (3 points) confirm the following:

> Running `/tmp/zsh` as a non-privileged user (e.g., `seed`) results in a shell with root privileges. But running `/tmp/bash` results in a shell that does not grant root privileges.

You then need to then **(b)** (10 points) figure out why this discrepancy exists between `/tmp/zsh` and `/tmp/bash`.

When you write your response for this question, make sure you include evidence (e.g., by copy-n-paste from your VM Ubuntu installation) for your responses to (a) and (b). For (a), the evidence would be how you confirmed the assertion.

For (b), the evidence would be how you determined the answer. Keep in mind that blind faith in what someone else says is not sufficient evidence. You need some evidence of a technical nature.

3. This question is about POSIX ACLs. You need to first remount the filesystem on your VM Ubuntu installation with ACL support. You do this as root with the following command:

```
mount -o remount,acl /
```

Now, the commands `setfacl` and `getfacl` should work on files.

The questions Suppose a user Alice has no write or execute access to a file named `data`, but has read access to it. Suppose the owner of `data` issues the following command:

```
setfacl -m u:alice:rw- data
```

(a) (3) Is it possible that Alice's current authorization to `data` does not change as a consequence of issuing the command? Why or why not (at most 3 sentences)?

(b) (5) Is it possible that another user, Bob's authorizations to `data` change as a consequence of issuing the command? Why or why not (at most 3 sentences)?

You can create new users such as alice and bob using the `adduser` utility.

ECE 458, Winter 2012

Assignment 2 - Due Sat, Feb 4, 11:59:59 PM

(Written portions must be typeset and in pdf. Use the dropbox in learn.)

1. (15) Consider the Finite State Automaton (FSA) in Figure 13 of the paper, "Setuid Demystified." There are 3 states: priv, unpriv_temp and unpriv_perm. There are 3 state-transitions: drop_priv_temp(), restore_priv() and drop_priv_perm().

The FSA shown in Figure 13 is not complete as it does not specify the new state for every ⟨state, state-transition⟩ pair. Draw a complete FSA.

Also, determine whether the specification for the state-transitions shown in Figure 14 is correct. (You could, for example, try them in your VM-Linux installation.) If yes, give a brief argument, and discuss any tests you tried. You may include the source code of your tests. If no, say why not and give the code and description for a new specification.

2. (15) Consider the following alternative to Fagin's approach in "On an Authorization Mechanism."

- A key in the authorization table is: table, right, grant-option, grantee. Each entry in the table contains the grantor and the timestamp as well, but those are not part of the key.
- If a grant is made that is already in the table, we adopt the latest (as opposed to the earliest) grant, and update the timestamp. For example, in Figure 3 in the paper, when A grants to C at time 40, the earlier grant from B to C at time 20 is overwritten by this new grant.

  An exception to this rule is if adding the edge results in a cycle in the graph. In such a case, we disallow the operation and make no changes to the authorization table. For example, in Figure 3 in the paper, if E attempts to grant to B, we detect that a cycle would result, and therefore disallow it and make no changes.
- Rather than "garbage collecting" downstream edges as mentioned in Page 315 of Fagin's paper, every time a grant is made, we walk forward in the graph and update the times to be the current time, which will be slightly greater than the time the grant was made. For example, in Figure 3 in the paper, when C grants D the privilege at time 60, we update the time at which D grants to E to 61 from 50.

Does the new scheme satisfy the correctness property as defined by Fagin? Does it satisfy the exercisability property? For each, if your answer is 'yes,' provide a proof. If your answer is 'no,' provide a counter-example.

3. (30) This question asks you to realize a protection mechanism from stack-smashing buffer overflow attacks.

Assume that every function invokes `protectMe_prologue()` as its first line of code (right after local variable declarations), and invokes `protectMe_epilogue()` before it returns. Implement these two functions with the signatures shown below to provide protection from stack-smashing buffer overflow attacks.

```
void protectMe_prologue();
void protectMe_epilogue();
```

You are allowed to:

- assume that a single global variable `void *s = NULL;` (i.e., `s` of type `void *`, initialized to `NULL`) is available for your use,

- define any data types (e.g., using `struct` and/or `typedef`) that you want, and,
- allowed to declare and use any local (within the two `protectMe_` functions you implement) variables you may need.

It suffices that you detect (e.g., by printing out a warning message and `exit()`-ing) when your functions detect that an overflow is happening. You will want to test that your solution works (no false negatives). And, it should be able to correctly identify when an attack is not happening (no false positives).

Notes about this question

- Use gcc as your compiler.
- You should set `kernel.randomize_va_space` to 0, `vm.legacy_va_layout` to 1, and `kernel.exec-shield` to 0 using `sysctl`. Also, use gcc with the options `-fno-stack-protector` and `-mpreferred-stack-boundary=2`. Of course, you may also want other options, such as `-ggdb`.
- You will want to reproduce a buffer-overflow problem as guided by the "Buffer-Overflow Vulnerability Lab" from the SEED project: `http://www.cis.syr.edu/~wedu/seed/Labs/Vulnerability/Buffer_Overflow/`. You can then test your solution to ensure that it works against this attack. (Hint: we will test your solution with something like this, in addition to other tests.)
- For a sample test case, see `stack.c` at `http://www.cis.syr.edu/~wedu/seed/Labs/Vulnerability/Buffer_Overflow/`.

Submission Instructions You should turnin a single `.zip` file named `ece458Assignment2.zip`. Within it, your responses should be organized as follows.

You should create a folder for each question with your response(s) for that question. Your folders should be named `1/`, `2/` and `3/` respectively.

`1/` This folder should contain a `.pdf` file for your written part, and any source code you've written. If you have written new versions of drop_priv_temp(), restore_priv() and drop_priv_perm(), then you should create a `Makefile` to compile them into a `.a` archive. The archive should be called `priv.a`. Here is how you create the `.a` file. Suppose that your implementation of the three functions is in the source file `priv.c`.

- `gcc -c priv.c` (This creates `priv.o`. Use whatever options you need to gcc.)
- `ar rvs priv.a priv.o` (This creates `priv.a`)

2/ Place a `.pdf` with your response for Question 2 in this folder.

3/ Include all your source-code, and a Makefile that creates a `.a` file named `protectMe.a`. This should contain (the compiled versions of) your protectMe_prologue() and protectMe_epilogue().

We will define the main() function (and any other functions we need). Make sure you have a `.h` file called `protectMe.h` that we can include in our code. Presumably, in `protectMe.h`, you will place any data type declarations you need for the global variable `s` that you are allowed to use.

Important NOTEs:

- You MUST provide Makefiles. All the TA will do is issue "make" from his script. If it does not work, you get an automatic 0. Your ultimate targets should be the `.a` files as discussed above.
- Your code should NOT contain a main() function. We will write that for both Questions 1 and 3. If we get a compilation error because of multiple main() functions, you will get an automatic 0.
- For Question 3, we will place an "`extern void *s;`" global variable declaration in our code. Make sure you place any data type declarations for `s` in `protectMe.h` as discussed above. We will `#include "protectMe.h"` in our code.

ECE 458, Winter 2012
Assignment 3 - Due Wed, Feb 29, 11:59:59 PM

(Written portions must be typeset and in pdf. Use the dropbox in learn.)

1. (15) Look at printRet.c that is on Learn. You need to edit it in two ways.

- Define `OFFSET_TO_RET_ADDR` (e.g., using `#define`) so it correctly prints out the return address from `f()` in the `printf()` statement that is already in printRet.c.
- Fill in code in the areas indicated to print out the return address using a format string to a call such as `printf()`.

Your output should look exactly like the following. (Of course, the return address that you print out may be different from what is shown here.)

```
Return address from looking into the stack: 0x0804855b
Return address using a format string: 0x0804855b
```

2. (30) An instructor uses the program addFeedback.c that is on Learn to leave feedback for her students. The instructor has root access, and each student has a non-root account. The instructor first runs the createFeedbackDir sequence of commands to create the `/tmp/feedback` directory with the appropriate permissions.

Look at the commands in createFeedbackDir to discern what it does. After you download it, you can simply `chmod u+x createFeedbackDir` and then `./createFeedbackDir` to run it.

She then uses addFeedback whenever she has some feedback for a student. As you can tell from addFeedback.c, the feedback for a student Alice is left in a file `alice`, where the filename `alice` is the same as Alice's username in the system. Alice may view the file, remove parts of feedback from the file, and even delete the file entirely.

You need to assess whether there is a security problem related to the above process; in particular, in addFeedback. Our security concern is: only Alice (somone that is able to authenticate herself as alice) and the instructor (i.e., root) should be able to view feedback that is intended for Alice. And, only Alice and the instructor should be able to edit feedback for Alice.

If you feel that there is a security issue, then you should edit addFeedback.c to address it. Your changes should be non-disruptive to the process that the instructor and the students follow, as discussed above. If you feel that there is no security issue, then write a brief paragraph to justify.

3. (30) For this question, you should set up the iptables development environment, and get the ipqTst sample program working as follows. The program ipqTst.c is on Learn.

1. `sudo vi /etc/apt/sources.list`
2. Add the following line to the file (say, near the top):
   `deb http://old-releases.ubuntu.com/ubuntu jaunty main universe restricted multiverse`
3. `sudo apt-get update`
4. `sudo apt-get install iptables-dev`
5. Compile `ipqTst.c`: `gcc ipqTst.c -o ipqTst -lipq`

6. Add kernel modules and an iptables entry as instructed in the man page for libipq, which is at `http://linux.die.net/man/3/libipq`:

   (a) `sudo modprobe iptable_filter`

   (b) `sudo modprobe ip_queue`

   (c) `sudo iptables -A OUTPUT -p icmp -j QUEUE`

7. Now run ipqTst: `sudo ipqTst`

8. Now, if you receive ICMP packets, you should see ipqTst printing out the first few bytes of the IP header. For example, if you issue `ping www.google.com`, you should see one line of output from ipqTst for every response packet. If you try `ping 127.0.0.1`, you should see one line of output for every request and response packet.

The objective of this exercise is to realize a "bump in the wire" protection mechanism from SQL injection and Cross-Site Scripting attacks. The bump in the wire will be a userspace program that uses libipq, iptables and the associated loadable kernel modules as indicated in the steps above.

(a) First reproduce some of the SQL injection and Cross-Site Scripting attacks against the phpbb that is part of your VM installation. That is, look at the pdf under "Lab Description and Tasks" at `http://www.cis.syr.edu/~wedu/seed/Labs/Attacks_SQL_Injection/` and `http://www.cis.syr.edu/~wedu/seed/Labs/Attacks_XSS/` .

For SQL injection, you should be able to reproduce at least the first attack, "Can you log into another person's account without knowing the correct password?" under "SQL Injection Attacks on Login" on the 4th page of the pdf. For Cross-Site Scripting, you should be able to reproduce Task 1, Task 2 and Task 3 under "Lab Tasks" on pages 3–4.

(b) Using iptables and libipq, realize a bump in the wire protection mechanism that detects (attempted) SQL injection and Cross-Site Scripting attacks against phpbb. It suffices if, when you detect an attack, you print out a warning message such as, "Warning: attempted SQL injection detected. Details: ..." (You could print out the source IP address of the attempted attack and what caused your code to raise the alarm under details.)

Submission Instructions You should turnin a single `.zip` file named `ece458Assignment3.zip`. Within it, your responses should be organized as follows.

You should create a folder for each question with your response(s) for that question. Your folders should be named `1/`, `2/` and `3/` respectively.

`1/` This folder should contain:

- Your (new) version of printRet.c, and,
- A makefile that compiles printRet.c into the executable printRet

`2/` This folder should contain a pdf with your justification if you feel that there is no security issue. Otherwise, it should contain:

- Your new version of addFeedback.c, and,
- A makefile that compiles addFeedback.c into the executable addFeedback

`3/` This folder should contain:

- A pdf that discusses which of the attacks for part (a) you were able to reproduce. You should also discuss briefly what you did to reproduce those attacks. (This is particularly important for the SQL injection attack(s) as I did not demonstrate them during the lectures.)
- The source-code for your bump in the wire detector.
- A makefile to create the executable for your bump in the wire detector. You should call the final executable "biwDetect."

ECE 458, Winter 2012
Assignment 4 - Due Fri, Mar 16, 11:59:59 PM
(Written portions must be typeset and in pdf. Use the dropbox in learn.)

1. Consider the problem of generating a TCP initial sequence number, and DNS query ID and port number that has been the subject of security issues. Suppose, to generate such an n-bit number, you simply ask a good random number generate to generate an n-bit number for you.

(a) (2) Would the approach work for TCP initial sequence numbers? ('Yes' or 'no' + 1 sentence justification.)

(b) (2) Would the approach work for DNS query ID + port number? ('Yes' or 'no' + 1 sentence justification.)

2. (7) Consider Figure 4.5 "Screened subnet architecture (using two routers)" and Figure 4.7 "Architecture using a merged interior and exterior router" from `http://oreilly.com/catalog/fire/chapter/ch04.html`.

(a) Provide an argument for the following perspective. "The approach of Figure 4.5 offers better security than the approach of Figure 4.7." You should specify, as part of your argument, what your notion of "security" is.

(b) Provide an argument for the following perspective. "The approach of Figure 4.7 offers better security than the approach of Figure 4.5." You should specify, as part of your argument, what your notion of "security" is. Your notion of security may be different from the one for (a) above.

(Hint: if your notions of security are from the Principles of Saltzer and Schroeder, that is ideal.)

3. (30) In this exercise, you want to protect your SQL server by using the intrusion detection system `snort` and the firewall `iptables` to first detect a SQL injection attack, and then program the firewall to prevent further TCP connection requests from the IP address from which we received the attempted SQL injection to our server. You should minimize false negatives and positives so that a legitimate client is allowed access, but a client that attempts SQL injection is blocked.

What you should do is use `snort` to detect whether a connection to your SQL server from a client with IP address i is associated with a SQL injection attack. If yes, then program `iptables` to block i from communicating with your server. You may have to write some code to link `snort` and `iptables`.

Note that you run `snort` only in non-inline mode. That is, it does detection only, and does not actively prevent anything. We use `iptables` for that.

Version 2.7.0 of `snort` may already be installed on your VM. Try typing `snort` at the shell prompt to check. You can also check for the `snort` executable in directories such as `/bin`, `/usr/sbin`, etc. If it is not installed, you should be able to install it via System → Administration → Synaptic Package Manager. In there, type "snort" in the search box. The manual for `snort` v. 2.7.0 is on Learn.

NOTE: you should use -I (insert) to add rules to `iptables`, and not -A (append). The reason is that we will add some rules that will follow your rules. Yould should be able to always insert your rules at the top of the chain. We will test your submission by mimicking SQL requests from various source IP addresses.

We will use source addresses from the 127.0.0.0/8 (loopback) subnet only. So none of your VM-related networking should be affected.

Submission Instructions You should turnin a single `.zip` file named `ece458Assignment4.zip`. Within it, your responses should be organized as follows.

You should create a folder for each question with your response(s) for that question. Your folders should be named `1/`, `2/` and `3/` respectively.

`1/` This folder should contain a pdf file with your response to Question 1.

`2/` This folder should contain a pdf file with your response to Question 2.

3/ This folder should contain:

- An executable shell-script named `protectSQLServer` that compiles all your programs and starts `snort` and your program that links `snort` and `iptables`. The TA will simply type `./protectSQLServer` and everything should happen, and the protection mechanism should start.
- All your source code, and any Makefiles and config files for `snort`.

ECE 458, Winter 2012

Assignment 5 - Due Mon, Apr 2, 11:59:59 PM

(Your solutions must be typeset and in pdf. Each question is worth 5 points.)

I refer to the following papers in this assignment. They are all up on Learn. As are my notes.

AN95 – Abadi and Needham, "Prudent Engineering Practice for Cryptographic Protocols."

BR93 – Bellare and Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols."

G96 – Gollmann, "What do we mean by Entity Authentication?"

L95 – Lowe, "An Attack on the Needham-Schroeder Public-Key Authentication Protocol."

1. The discussion in Section 4.2 of G96 demonstrates that the use of RSA for signatures in the manner discussed there is susceptible to selective forgery. Suppose Alice improves the scheme — her signature on m is $(h(m))^d \mod n$, where h is a hash function. Unfortunately, the particular h that she chooses has the following property: $(h(x) \cdot h(y)) \mod n = h(x + y) \mod n$. Show that the new scheme is susceptible to existential forgery.

2. In at most two brief sentences, justify the following assertion: an encryption scheme that is susceptible to a Chosen Plaintext adversary is susceptible to a Chosen Ciphertex adversary.

3. Suppose the encryption scheme is $E(x) = f(r) \,||\, G(r) \oplus x \,||\, H(r) \,||\, H(x)$, where r is a random number that is generated anew for each instance of encryption, $G(\cdot)$ is a random number generator that is a function of the input seed, $f(\cdot)$ is the public-key operation of an asymmetric block encryption algorithm with corresponding private-key operation f^{-1}, and $H(\cdot)$ is a hash function.

(a) What would be the corresponding decryption scheme, $D(y)$, of some string y?

(b) Is the scheme secure from a Chosen Plaintext adversary? Justify briefly.

4. Show that Cipher Block Chaining (CBC) mode for a block cipher is not secure from a Chosen Plaintext adversary if a fixed (publicly known) Initialization Vector (IV) is used. (Look up "block cipher modes of operation" online to find out what CBC mode is.)

5. In Section 3.1, BR93 propose the following encryption scheme. $E(x) = f(r) \,||\, G(r) \oplus x$. (See Question 3 above for an explanation of the components.) Then, in the following section, they assert, "It is easy to see that the scheme of the previous section is not secure against [a chosen ciphertext attack]." Show why it is not.

6. From Section 6, 4th paragraph of BR93, "...MD5(ax) cannot be used as a message authentication code of string x under key a." Given the property of MD5 that they discuss immediately prior, show why this is the case. (Hint: existential forgery.)

7. From Section 4, Example 3.4 of AN95: "We leave the construction of an attack as an exercise for the reader." Construct an attack. (You can use their proposed resolution as a hint.)

8. In Section 5, Example 5.2 of AN95, fix the message so that B can be convinced that A knows X. Your solution should have all the security properties of the existing solution plus this.

9. In Section 6.2, Example 9.1 of AN95, suppose that Message 5 is $A \longrightarrow B : \{N_b\}_{K_{ab}}$ (i.e., without

the "+1"). Show an attack on the protocol.

10. See the public-key version of the Needham-Schroeder authentication protocol in Section 2 of L95. Notice that in Message 7, A does not increment N_b by 1 before sending it back to B. Is your attack from the previous question possible on this version of the protocol? Why or why not?

11. Suppose that a malicious principal E acquires K_{ab} from a prior run of the Needham-Schroeder symmetric-key authentication protocol in Section 6.2, Example 9.1 of AN95. Show how E can compromise (or "deceive," as AN95 puts it) B as a consequence.

12. In Section 7, in the context of Example 10.1, AN95 asserts: "For example, a '+1' operation appears in Kerberos, where it is unnecessary." Assume that "Kerberos" refers to the protocol from Section 5, Example 4.1 of AN95. Argue briefly why the "+1" in Message 4 is unnecessary.

13. In Section 6.1, Example 6.1 of AN95, notice that the "M" is removed in their new version of the Otway-Rees protocol. Assume that the entities do not have any difficulty keeping instances of the protocol separate. (They may do this, for example, by choosing new ports for each instance of the protocol.) Identify a (security) consequence of removing the "M" from the protocol.

Submission Instructions Turnin a single `.pdf` file.