

## Homework #2

**Due date & time:** 1:30pm on October 11, 2007. Hand in at the beginning of class (preferred), or email to the TA (zmao@purdue.edu) by the due time.

**Late Policy:** Late homeworks will not be accepted.

**Additional Instructions:** The submitted homework must be typed. Using Latex is recommended, but not required.

**Problem 1 (10 pts)** Write a C function that contains a heap-overflow vulnerability, which is caused by an integer overflow vulnerability.

**Problem 2 (5 pts)** Compare the advantages and disadvantages of using the following three kinds of canaries in StackGuard: (1) a fixed canary value, (2) a random canary, and (3) a terminator canary.

**Problem 3 (5 pts)** Describe how a “return-to-libc” attack work.

**Problem 4 (5 pts)** Recall that a security label consists of a security level and a set of categories. Given the security levels *public*, *confidential*, and *strictly confidential*, and the categories FACULTY, STAFF, and STUDENTS. How many labels can be constructed? Which labels dominate the label (confidential, STUDENT)? More generally, how many security labels can be constructed given  $n$  security levels and  $m$  categories.

**Problem 5 (5 pts)** Explain the three classes of virtualization technologies discussed in class, and compare their advantages and disadvantages.

**Problem 6 (5 pts)** The Linux capability system labels each executable file with three sets of bitmaps: allowed, forced, and transfer. Suppose that a program is written without using the Linux capability system, and you need to give the program full root privilege (i.e., achieve the same effect as making the program `setuid root`). What do you have to assign to the three bitmaps.

**Problem 8 (10 pts)** Explain the following terminologies: Trusted Computer Base, Trusted Path, discretionary access control, mandatory access control, covert channels.

**Problem 9 (20 pts)** Read Part IA of the “The Protection of Information in Computer Systems” by Saltzer and Schroeder. For each of the eight principles listed there, write your understanding of the principle, give an instance of where it has been applied or where it should be applied.

**Problem 10 (25 pts)** The description of this problem is long, and the question you need to answer are in boldface.

In the Bell-LaPadula (BLP) model, a computer system is modeled as a state-transition system. To use the system model in BLP to describe a computer system, one first specifies  $S, S_T, A, L$ , which are explained as follows.

- $S$  is a set of subjects.  
The set of subjects in a system is fixed in the original description of BLP. This means that when a system is running, no new subject can be added and no subject can be removed. This certainly limits the kinds of systems that can be modeled in the BLP system model. It is possible to extend the BLP system model to allow a dynamic set of subjects.
- A set  $S_T \subset S$  of trusted subjects.
- A set of access modes  $A$ . For simplicity, we assume that  $A = \{read, write\}$ .
- A partially order set  $\langle L, \leq \rangle$  of security levels.

Each state is a tuple  $\langle O, b, F \rangle$ :

- $O$  is the set of the objects that currently exist in the state.
- $b \subseteq S \times O \times A$  is the current access set. Each element in  $b$  is a triple  $(s, o, a)$ , meaning that currently the subject  $s$  holds the access mode  $a$  over the object  $o$ .
- $F = \langle f_S, f_O, f_C \rangle$  is a triple of security level functions, where
  - $f_S : S \rightarrow L$  is the (maximum) security level function for subjects,
  - $f_O : O \rightarrow L$  is the object security level function, and
  - $f_C : S \rightarrow L$  is the current security level function for subjects.
 The idea is that while a subject does not always have to operate at its maximum security level. A subject can change its current security level.

A system is BLP-secure if and only if every reachable state is a secure state. A state  $\langle O, b, F = \langle f_S, f_O, f_C \rangle \rangle$  is a *secure state* if and only if it has the following two properties:

1. it satisfies *the simple-security property (ss-property)*; that is, for every  $(s, o, read) \in b$ ,  $f_S(s) \geq f_O(o)$ . In other words, if a subject can read an object, the subject's *maximal* security level must dominate the object's security level.
2. it satisfies *the \*-property*; that is, for every  $(s, o, a) \in b$  where  $s \notin S_T$ , the following conditions hold
  - if  $a = read$ , then  $f_C(s) \geq f_O(o)$
  - if  $a = write$ , then  $f_C(s) \leq f_O(o)$

Note that the current security level of a subject is used in the \*-property.

- **Several authors said that the \*-property implies the ss-property. Why is this wrong? Under what conditions would the \*-property imply the ss-property?**
- **The definition of the \*-property above has two conditions, one for reading and one for writing. Can we remove the restriction about reading? Explain why.**

Consider the following system, with  $S = \{s_1, s_2\}$ ,  $S_T = \emptyset$ ,  $L = \{\text{high}, \text{low}\}$ . The following are the only state transitions that are allowed:

- A subject  $s$  request to read an object  $o$ , which succeeds only if adding  $(s, o, \text{read})$  to  $b$  does not violate the ss-property and the \*-property.
- A subject  $s$  stops reading an object  $o$ , which succeeds if  $(s, o, \text{read})$  is in  $b$ , and results in  $(s, o, \text{read})$  being removed from  $b$ .
- A subject  $s$  request to write an object  $o$ , which succeeds only if adding  $(s, o, \text{write})$  to  $b$  does not violate the ss-property and the \*-property.
- A subject  $s$  stops reading an object  $o$ , which succeeds if  $(s, o, \text{write})$  is in  $b$ , and results in  $(s, o, \text{write})$  being removed from  $b$ .
- Change the current security level of a subject, which succeeds if after the change, the ss-property and \*-property are satisfied.

The initial state is specified as follows:

- $O = \{o_1, o_2\}$ ,
  - $b = \{(s_1, o_1, \text{read})\}$ ,
  - $f_S = \{s_1 \rightarrow \text{high}, s_2 \rightarrow \text{low}\}$ ,
  - $f_O = \{o_1 \rightarrow \text{high}, o_2 \rightarrow \text{low}\}$ ,
  - $f_C = \{s_1 \rightarrow \text{high}, s_2 \rightarrow \text{low}\}$ .
- **Prove, using mathematical induction, that the system is BLP-secure.**
  - **Provide a sequence of state transitions such that in the sequence, the subject  $s_1$  reads  $o_1$ , writes to  $o_2$ , which is then read by  $s_2$ . That is, we show that the system is intuitively insecure, even though it can proven secure under the BLP definition.**
  - **How to fix the BLP security definition so that system with the above weaknesses will not be proven secure?**