Tentamen Operating Systems Security, 18 January 2016, 12:30-15:30

(tot 16:30 voor studenten met extra tijd)

Any material other than a pen is not allowed; in particular no books, notes, or calculator.

Write clearly and answer short and concise. Je mag gewoon in het Nederlands antwoorden. Succes!

1. **(15 points)** A company uses two-factor authentication at the login to their Linux computers. The users have to authenticate by USB token (PAM module `pam_usbtoken.so`) and additionally authenticate either by password (PAM module `pam_unix.so`) or by fingerprint (PAM module `pam_fprint.so`).

   (a) What does the file `/etc/pam.d/login` need to look like to implement the policy described above?

   (b) What does the file `/etc/pam.d/login` need to look like to enforce three-factor authentication (all three methods are required).

   (c) Can you think of a way to implement a policy that allows a user to log in with any two out of the three methods?

2. **(20 points)** Consider the following access matrix for 2 files and 4 users:

	user1	user2	user3	user4
file1	read,write	read	read	
file2	read,write,execute	read,execute	execute	read

   (a) For each of the files state, whether it is possible to implement the policy given by the access matrix only with classical UNIX access rights.
   If it is possible, give a sequence of commands that assigns ownership and group to files, assigns group memberships to users, and sets the respective access rights.
   If it is not possible, give a brief explanation why not and give a list of commands that sets UNIX Access Control Lists (ACLs) for the files to accomplish the policy stated by the access matrix.

   (b) Can any of the four users increase his/her own access rights for any of the two files with the setup from part a)? Explain why or why not.

3. **(20 points)** Consider the follwing obviously vulnerable piece of code:

```
int bad(const char *str, size_t len) {
  char buf[256];
  int i;
  for(i=0;i<len;i++)) {
    buf[i] = str[i];
  }
  ...
}
```

   Assume that the address of `buf` is `0x7fffffffe0c0` and assume that the return address of the function `bad` is stored at `7fffffffe1c8`.
   Assume furthermore that the program is running with non-executable stack on an AMD64 (also known as x86_64 or x64) machine and that an attacker controls the arguments to the function.

(a) What additional addresses does an attacker need to know *at least* to mount a return-to-libc attack with the goal to launch a shell through a call to the `system` function from the standard library?

(b) With the knowledge of the addresses from part a), the attacker now crafts an input to the function `bad`. What exactly does this input need to look like to obtain a shell?

(c) Now assume that the function is changed slightly to the following (using the `strlen` function defined in `string.h`):

```
int bad(const char *str) {
  char buf[256];
  int i;
  for(i=0;i<strlen(str);i++)) {
    buf[i] = str[i];
  }
  ...
}
```

Why does the attack from part b) not work anymore?

(d) How can an attacker mount the attack against the modified version of the function from part c)? What additional information is required? How does the attack string need to change?

(e) What mechanism do you know that the operating system can use to make such an attack considerably harder?

4. **(20 points)** Consider an operating system with a reference monitor that is using mandatory access control to implement the Bell-LaPadula security model with weak tranquility. Assume that a user logs in with clearance *secret* and starts a program `userprog`. This program is attempting to perform a certain sequence of operations on files. For each of those operations state whether it is allowed or forbidden. If an operation is forbidden, briefly explain why. Additionally, state the security level of the program before it performs any operations and every time the security level changes.

   Note: The program tries to perform the operations in the order given below. If an operation is forbidden, assume that the program simply continues without performing this operation.

   (a) Write to file `file1` with level "top secret"

   (b) Read the file `file2` with level "confidential"

   (c) Write to file `file2` with level "confidential"

   (d) Write to file `file3` with level "unclassified"

   (e) Read file `file2` with level "top secret"

   (f) Read file `file4` with level "secret"

   (g) Write to file `file3` with level "confidential"

5. **(15 points)** Answer the following questions about race conditions:

   (a) What are race conditions and how can they be prevented?

   (b) Consider the piece of code below, which contains a race condition. Explain why.

   (c) How would you change to code to eliminate the race condition?

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>

static void charatatime(char * str){
  char * ptr;
  int c;
  for(ptr = str; c = *ptr++; ){
    putc(c, stdout);
    fflush(stdout);
  }
}

int main(void){
  pid_t pid;
  if((pid = fork()) < 0) {
    perror("Fork error");
    exit(1);}
  else if( pid == 0) {
    charatatime("This is the output of the original process\n");}
  else {
    charatatime("This is the output of the forked process\n");}
  exit(0);
}
```

6. **(20 points)** Consider the following three different scenarios of protecting a system against the compromise of a mail server:

   - The mail server is running with non-priviledged (i.e., non-root) user-ID.
   - The mail server is running with non-priviledged user-ID in a `chroot` environment.
   - The mail server is running in a virtual machine (using Xen), which is running its own Linux operating system.

   (a) Explain why the second protection measure is stronger than the first by describing a type of malware that would compromise the system in the first scenario, but not in the second scenario. Describe what kind of vulnerability or vulnerabilities this malware would need to exploit.

   (b) Explain why the third protection measure is stronger than the second by describing a type of malware that would compromise the system in the second scenario, but not in the third scenario. Describe what kind of vulnerability or vulnerabilities this malware would need to exploit.