Tentamen Operating Systems Security, 23 January 2015, 8:30-11:30

(tot 12:30 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. For computers with a fingerprint reader it is possible to add fingerprint authentication through the `pam_fprint` module.

   (a) What does the file `/etc/pam.d/login` need to look like to allow *only* authentication by fingerprint?

   (b) What does the file `/etc/pam.d/login` need to look like to allow authentication *only* by fingerprint *and* password?

   (c) What does the file `/etc/pam.d/login` need to look like to allow authentication by either by fingerprint *or* password?

   Note: The PAM module for password authentication is called `pam_unix`.

2. Consider a Linux system with four users, `luke`, `han`, `leia`, and `chewie`. Each user is member of only one group, namely his personal group. In other words, `luke` is member of the group `luke`; `han` is member of the group `han` etc.

   (a) Give a set of files together with access-control lists for the 4 users, such that the same access rights *could not* be achieved only by assigning ownership, group, and traditional UNIX access rights to the files.

   (b) Give a sequence of commands that sets the access-control lists from part a).

   (c) Explain why the sticky bit is typically set for the `/tmp/` directory on UNIX systems.

3. Assume that the following is a code snippet from a program that is running with the `suid` bit set:

```
char buffer[100];
strncpy(buffer, argv[1],100);
if (access("/tmp/mysuidprogram.log", W_OK) != 0) {
  exit(1);
}
int fd = open("/tmp/mysuidprogram.log", O_WRONLY);
write(fd, buffer, strlen(buffer));
```

   (a) Explain line by line what this code does.

   (b) Explain what the security problem in this code is.

   (c) How would an attacker exploit this vulnerability? Give one example of what he could achieve.

   (d) Explain how the vulnerability could be removed in this piece of code.

   Note: `argv[1]` is the first command line argument passed to the process.

4. Assume that the reference monitor of a security-enhanced operating system implements Bell-LaPadula with categories. Bell-LaPadula uses the typical confidentiality levels **unclassified** $\leq$ **confidential** $\leq$ **secret** $\leq$ **top secret** $\leq$. Categories are **financial**, **hr**, **sciences**, **medicine**, **languages**, and **arts**. A user has clearance up to level **secret** and compartment {**financial**, **sciences**, **medicine**}. This user starts a process and this process attempts to perform the following tasks in the given order:

   (a) Read the file `file1` with label **unclassified** and compartment {}.

   (b) Read the file `file2` with label **confidential** and compartment {**financial**}.

   (c) Read the file `file3` with label **confidential** and compartment {**hr**}.

   (d) Read the file `file4` with label **confidential** and compartment {**hr,sciences**}.

   (e) Write the file `file5` with label **secret** and compartment {**hr,sciences**}.

   (f) Write the file `file6` with label **secret** and compartment {**financial,sciences**}.

   (g) Write the file `file7` with label **unclassified** and compartment {}.

   (h) Read the file `file8` with label **top secret** and compartment {**medicine**}.

   (i) Write the file `file9` with label **top secret** and compartment {**medicine**}.

   (j) Write the file `file10` with label **secret** and compartment {**medicine**, **sciences**}.

   Answer for each of these tasks whether it is allowed by the reference monitor and if it is not allowed, explain why.

   Note: Assume that if a task fails, because it is stopped by the reference monitor, the process will just continue without this task as if it had never attempted to perform this task.

5. Consider this snippet of code from the Sniffit network traffic sniffer:

```
char *clean_string (char *string) {
   char help[20];
   int i, j;
   j=0;
   for (i=0;i<strlen(string);i++) {
      if( (isalnum(string[i]))||(string[i]=='.') ) {
         help[j]=string[i];
         help[j+1]=0;
      }
      j++;
   }
   strcpy(string, help);
   return string;
}
```

   Note: Sniffit is typically running with `root` permissions, e.g., through `suid`; the input `string` can be controlled by the user through a configuration file.
   Answer the following questions:

   (a) Where is the critical vulnerability in this code?

   (b) How would an attacker typically exploit this vulnerability and what would he achieve?

(c) Does a non-executable stack help against attacks that exploit this kind of vulnerability? Explain your answer.

(d) Does address-space-layout randomization help against attacks that exploit this kind of vulnerability? Explain your answer.

6. Recent versions of the Windows operating system have a feature called Volume Snapshot Service (VSS) or "Shadow Copy" that allows the administrator to save the system state and later restore this system state. VSS works on the block level of the file system.

(a) In principle, a VSS restore can be used to restore a clean system state after a malware infection; however, this is safe only for certain types of malware. Give one example of malware which is safely removed by a VSS restore and one example of malware which is not removed by a VSS restore.

(b) Also most virtual-machine managers (e.g., VMWare) allow to take snapshots of a virtual machine (VM) and later restore this state. Explain why this provides a better way to restore a clean system state after a malware infection. Give one example of malware which is eliminated by a VM restore but not by a VSS restore.