Tentamen Operating Systems Security, 23 January 2017, 12:30-15:30

(tot 16:00 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. **(10 points)** Logging into a computer over SSH *from an untrusted machine* creates a security issue: the user cannot be sure that the untrusted machine does not, for example, run a keylogger in the background to obtain the user's password. One solution are one-time passwords, which can be used in PAM using the `pam_otpw.so` module. This module comes together with a one-time-password generator `otpw-gen`, which generates a list of one-time passwords and creates a file of the corresponding password hashes in the user's home directory.

   (a) What should the `/etc/pam.d/sshd` configuration file for the SSH server look like to allow authentication *only* by one-time password?

   (b) What should the `/etc/pam.d/sshd` configuration file for the SSH server look like to allow authentication by one-time password as a fallback if the usual password authentication fails (e.g., because the user decided to simply hit "enter" at the password prompt)?

   (c) What should the `/etc/pam.d/sshd` configuration file for the SSH server look like to use normal password authentication as fallback if authentication by one-time password failed (e.g., because the user forgot to generate sufficiently many one-time passwords)?

   (d) You wouldn't expect to see a line like the following in `/etc/pam.d/sshd`:

   ```
   auth      sufficient    pam_rootok.so
   ```

   Explain why.

   (e) In what PAM configuration file would you expect to see such a line?

2. **(15 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, which 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) Read file `/home/user/ossec/exam2015.tex` with level *secret*

   (b) Write file `/home/user/ossec/exam2016.tex` with level *top secret*

   (c) Read file `/home/user/notes.txt` with level *unclassified*

   (d) Write file `/home/user/notes.txt` with level *unclassified*

   (e) Read file `/etc/shadow` with level *top secret*

   (f) Read file `/var/log/syslog` with level *secret*

   (g) Read file `/home/user/ossec/exam2016.tex` with level *top secret*

3. **(25 points)** The "classical" way to exploit a buffer-overflow vulnerability is to inject shellcode on the stack and overwrite the return address with the address of that shell code. Consider the following vulnerable function:

```
int vuln() {
    char buf[150];
    ssize_t b;
    memset(buf, 0, 150);
    printf("Enter input: ");
    b = read(0, buf, 400);

    printf("Recv: ");
    write(1, buf, b);
    return 0;
}
```

(a) Which line contains the vulnerability?

(b) Assume that a program using the function `vuln()` is running with non-executable stack such that the shell-code-injection attack won't work. What kind of attack could you mount to exploit the buffer overflow instead?

(c) Assume that the program is running on an AMD64 machine. What addresses will you need to mount the attack from part b)?

(d) Explain in detail what data an attacker needs to place in `buf` in order to mount the attack. What address listed in part c) needs to overwrite the return address of `vuln()`?

(e) Address-space layout randomization (ASLR) makes this attack considerably more difficult. Explain why.

(f) What could an attacker do to circumvent ASLR and still mount the attack described in part b)?

4. **(15 points)** Consider the following program that only opens the file `/tmp/outfile.txt` for writing (appending), if the file exists and is not a symbolic link.

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

int main(int argc, char *argv[]) {
  struct stat st;
  FILE *fh;
  const char *path = "/tmp/outfile.txt";
  uid_t puid;
  int r;

  if(argc < 2) {
    fprintf(stderr, "Usage: %s FILE\n", argv[0]);
    return -1;
  }

  r = lstat(path, &st);
  if(r < 0) {
    fprintf(stderr, "lstat failed on %s\n", argv[1]);
    return -1;
  }

  puid = geteuid();

  // Check, whether file is a symbolic link
  if((st.st_mode & S_IFMT) != S_IFLNK) {
    fh = fopen(path, "a");
    if (!fh) {
      fprintf(stderr, "Opening file %s for appending failed\n", argv[1]);
      return -1;
    }

    fprintf(fh, "%s\n",argv[1]);

    fclose(fh);
  }
  else {
    fprintf(stderr, "Error: file %s is a symbolic link\n", argv[1]);
  }

  return 0;
}
```

Assume that an attacker has control over the command-line argument (`argv[1]`) passed to the program and that his target is to trick the program into writing that string to a symbolic link (for example, pointing to `/etc/shadow`).

(a) What is the vulnerability in the code that allows the attacker to achieve his goal?

(b) How would the attacker achieve his goal?

(c) Sketch how you would fix the vulnerability.

5. **(20 points)** Assume that you have one physical machine available that you want to use to run a web server (`apache2`) and a mail server (`qmail`). Consider the following different setups:

   (a) Both the `apache2` process and the `qmail` process run as user `root` natively on a Linux system.

   (b) The `apache2` process runs as user `web`, the `qmail` process runs as user `mail`. Each of those two users has read/write access only to the files that it needs to have access to to function (e.g., `qmail` has read/write access to mails; `apache2` has read access to html files that it serves)

   (c) The `apache2` and the `qmail` process each run as user `root` in separate chroot environments.

   (d) The `apache2` and the `qmail` process run in separate virtual machines controlled by a Xen hypervisor.

   Assume that an attacker has an exploit against the web server that gives a shell with the permissions of the web server (i.e., as the user that is running `apache2`). The target of the attacker is access to mails stored by the `qmail` process. For each of the scenarios above state what the attacker needs to do to obtain these mails and what additional exploits (if any) are required.

6. **(15 points)** One goal of a multiuser operating system is to protect each user's information and activity from damage caused by accidental or deliberate actions of other users of the system.

   (a) Describe two mechanisms that operating systems could use to reduce the opportunity for a user process to prevent another user's process from making progress. In your answer, include any particular OS features that are relied upon.

   (b) Describe how an operating system might attempt to ensure that long-term user information (i.e. information which exists beyond process execution) is not interfered with or misused by other users. Your answer should be clear about which actions are performed and the resources they consume.