Master 2 SeCReTS 2010-2011
Module Sécurité Système
Examen - Partie 2

Sécurité des systèmes Unix

Consignes :
- 2h maximum ;
- tout document autorisé ;
- aucune communication ;
- aucun accès à un ordinateur, une station de travail, un téléphone portable, une calculatrice, un PDA ou tout autre dispositif électronique, connectable ou non.

Deux points sont attribués au soin apporté à la rédaction des réponses.

---

1. *(2 points)* Rappelez les deux principales raisons pour lesquelles on préfère installer un système minimal quand on veut obtenir un système plus sûr.
2. *(3 points)* Que signifie le sigle PAM ? Quel aspect de la sécurité des systèmes Unix est couvert par les PAM ?
   Les PAM constituent un mécanisme modulaire. Pourquoi ?
   Sur un système protégé par les PAM, on trouve le fichier `/etc/pam.d/login`. Que contient-il ?
3. *(7 points)* Observez le code suivant :

```c
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define LISTEN_PORT 11221

int main() {
  char buf[64];

  int sock;
  int peersock;
  struct sockaddr_in my_addr;

  sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  memset(&my_addr, 0, sizeof(my_addr));
  my_addr.sin_family = AF_INET;
  my_addr.sin_port = htons(LISTEN_PORT);
  bind(sock, (struct sockaddr *)&my_addr, sizeof(my_addr));
  listen(sock, 5);
  peersock = accept(sock, NULL, 0);
  recv(peersock, buf, 500, 0);
  return(0);
}
```

   (a) *(2 points)* Décrivez brièvement le fonctionnement du programme, et comment on peut lui communiquer des données.

   (b) *(3 points)* A partir des sources, on peut observer que ce programme présente une vulnérabilité de type *stack buffer overflow*.
   - Quelle est le nom du buffer dans lequel sont écrites les données reçues ? Quelle est sa taille ?
   - Quelle fonction écrit des données dans ce buffer ? Combien d'octets au maximum ?
   - En cas de tentative d'attaque par *buffer overflow*, à quel moment le programme va probablement faire une erreur de segmentation ?

   (c) *(2 points)* Un attaquant détermine qu'il faut fournir 96 octets de données avant de placer une adresse de retour. Décrivez le format des données à envoyer au programme pour réussir une attaque.

4. *(6 points)* Observez le listing suivant :

```
% ldd /usr/sbin/lighttpd
        linux-gate.so.1 =>  (0x008c5000)
        libpcre.so.3 => /lib/libpcre.so.3 (0x007dc000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0x00dcf000)
        libattr.so.1 => /lib/libattr.so.1 (0x00365000)
        libssl.so.0.9.8 => /lib/i686/cmov/libssl.so.0.9.8 (0x00110000)
        libcrypto.so.0.9.8 => /lib/i686/cmov/libcrypto.so.0.9.8 (0x0015a000)
        libfam.so.0 => /usr/lib/libfam.so.0 (0x0062b000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00eaa000)
        /lib/ld-linux.so.2 (0x002d5000)
        libz.so.1 => /lib/libz.so.1 (0x0075f000)
        libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0x00bdc000)
```

On souhaite faire fonctionner le programme `lighttpd` dans un environnement de type cage, aussi appelé `chroot()`.

(a) *(2 points)* Quel est l'intérêt de contruire un environnement restreint pour le programme lighttpd ?

(b) *(1 points)* Comment prenez-vous en compte le listing pour la construction de l'environnement du programme `lighttpd` ?

(c) *(3 points)* Citez deux autres mesures de précaution à appliquer à `lighttpd` pour minimiser ses privilèges d'exécution.

Question bonus *(2 points) :* L'administrateur observe le texte suivant dans les logs de l'application. Expliquez en quelques lignes ce qui s'est passé.

```
*** stack smashing detected ***: /sbin/vuln terminated
======= Backtrace: =========
/lib32/libc.so.6(__fortify_fail+0x50)[0xf7677aa0]
/lib32/libc.so.6(+0xe4a4a)[0xf7677a4a]
/sbin/vuln[0x8048628]
[0x41414141]
======= Memory map: ========
08048000-08049000 r-xp 00000000 08:05 67729  /sbin/vuln
08049000-0804a000 r--p 00000000 08:05 67729  /sbin/vuln
0804a000-0804b000 rw-p 00001000 08:05 67729  /sbin/vuln
09986000-099a7000 rw-p 00000000 00:00 0      [heap]
f7576000-f7590000 r-xp 00000000 fc:06 390934 /usr/lib32/libgcc_s.so.1
f7590000-f7591000 r--p 00019000 fc:06 390934 /usr/lib32/libgcc_s.so.1
f7591000-f7592000 rw-p 0001a000 fc:06 390934 /usr/lib32/libgcc_s.so.1
f7592000-f7593000 rw-p 00000000 00:00 0
f7593000-f76e7000 r-xp 00000000 fc:01 865    /lib32/libc-2.12.1.so
f76e7000-f76e9000 r--p 00154000 fc:01 865    /lib32/libc-2.12.1.so
f76e9000-f76ea000 rw-p 00156000 fc:01 865    /lib32/libc-2.12.1.so
f76ea000-f76ed000 rw-p 00000000 00:00 0
f7709000-f770b000 rw-p 00000000 00:00 0
f770b000-f770c000 r-xp 00000000 00:00 0      [vdso]
f770c000-f7728000 r-xp 00000000 fc:01 5314   /lib32/ld-2.12.1.so
f7728000-f7729000 r--p 0001b000 fc:01 5314   /lib32/ld-2.12.1.so
f7729000-f772a000 rw-p 0001c000 fc:01 5314   /lib32/ld-2.12.1.so
ffbb0000-ffbd1000 rw-p 00000000 00:00 0      [stack]
```

Fin de l'examen.