Examen de Sécurité Informatique

21 avril 2009

Chaque exercice vaut 5 points.

1 Cryptographie RSA

Vous incarnez Bob, un célèbre agent secret. Un jour, vous voyez passer ce message crypté :

$$\underbrace{0001010101101}_{C_0}\ \underbrace{100010110001011}_{C_1}\ \underbrace{101110110010001}_{C_2}\ \underbrace{1011010110001101}_{C_3}$$

(une suite de 4 blocs de bits). Vous vous doutez immédiatement qu'Alice (votre ennemie jurée) a envoyé ce message à Kate (une vilaine espionne infiltrée, à la solde des méchants). La clé publique RSA de Kate est

$$(e, n) = (31, 52447)\ .$$

« Attends un peu, ma gaillarde ! », maugréez-vous en silence. « Grâce à Python, ma redoutable arme secrète, je vais de ce pas décrypter ton message ! »

1. Écrivez un programme en Python permettant de retrouver la clé privée (d, n) de Kate à partir de sa clé publique (e, n). Donnez la valeur de (d, n).
2. Le bit de poids fort de chaque C_i est celui le plus à droite. Convertissez chaque C_i en décimal et calculez le message en clair M_i (un nombre en décimal) correspondant à C_i. Sachant que chaque M_i résulte de la juxtaposition de deux codes ASCII, retrouvez le message (texte) qu'Alice a envoyé à Kate.

2 Vulnérabilités du réseau

1. Expliquez en détail en quoi consiste une attaque par ARP spoofing. Vous prendrez soin de définir clairement chacun des mécanismes sous-jacents (protocole ARP, adresse MAC, ...)
2. Détaillez deux types d'attaque par fragmentation permettant d'outrepasser les règles de filtrage d'un firewall. Comment peut-on se prémunir contre chacun de ces types d'attaque ?

3 Vulnérabilités applicatives : race condition

On considère le code donné à la figure 1. Quel est le danger de ce code sachant que l'exécutable correspondant appartient à l'utilisateur `root` et possède le droit SUID (flag `s`) ? Que faudrait-il faire pour éliminer ce danger (on ne demande pas de code, seulement une explication) ?

4 Sécurité en Java 2

Supposons que vous disposiez de la version 6 du JDK. Donnez les commandes exactes répondant aux questions suivantes.

1. Générez un fichier keystore de nom `.keystore` contenant une paire de clés RSA.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
int main(int argc, char **argv) {
   struct stat st;
   FILE *fp;
   if (argc != 3) {
       fprintf(stderr,"usage : %s fichier message\n", argv[0]);
       exit(EXIT_FAILURE);
   }
   if (stat(argv[1], &st) < 0) {
       fprintf(stderr,"%s introuvable\n", argv[1]);
       exit(EXIT_FAILURE);
   }
   if (st.st_uid != getuid ()) {
       fprintf(stderr,"%s ne vous appartient pas !\n", argv[1]);
       exit(EXIT_FAILURE);
   }
   if (! S_ISREG (st.st_mode)) {
       fprintf(stderr,"%s n'est pas un fichier normal\n", argv[1]);
       exit(EXIT_FAILURE);
   }
   if ( (fp = fopen(argv[1],"w")) == NULL) {
       fprintf(stderr,"Ouverture impossible\n");
       exit(EXIT_FAILURE);
   }
   fprintf(fp,"%s\n",argv[2]);
   fclose(fp);
   fprintf(stderr,"Ecriture OK\n");
   exit(EXIT_SUCCESS);
}
```

FIG. 1 –

2. Créez une archive jar exécutable de nom `Truc.jar` contenant les fichiers `Machin.class` et `Main.class` (la méthode `main` se trouve dans `Main.class`).
3. Signez l'archive `Truc.jar` en utilisant votre paire de clés RSA.
4. Vérifiez la validité de votre archive. Qu'aurait répondu votre commande si votre keystore ne contenait pas la paire de clés ayant servi à signer ? Dites pourquoi.
5. Créez votre fichier de politique de sécurité de sorte que les archives exécutables signées par vous (en utilisant votre paire de clés RSA) aient le droit de lire les fichiers placés dans votre répertoire `home` (mais pas dans ses sous-répertoires). Quelle commande faut-il taper pour exécuter votre archive `Truc.jar` en tenant compte de votre politique de sécurité ?