Examen de Sécurité Informatique

Session 1

20 avril 2010

1 Cryptographie RSA

On considère le message crypté :

$$\underbrace{111110101000100001}_{C_0}\ \underbrace{111011111101111001}_{C_1}\ \underbrace{00100000111010011}_{C_2}$$

(une suite de 3 blocs de bits) envoyé par un espion A à un espion B. La clé publique RSA de B est

$$(e, n) = (53, 246\ 973)\ .$$

1. (3 points) Écrivez un programme permettant de retrouver la clé privée (d, n) de B à partir de sa clé publique (e, n). Donnez la valeur de (d, n).
2. (4 points) 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) que A a envoyé à B.

2 Vulnérabilités du réseau

(3 points) Expliquez en détail en quoi consiste une attaque par vol de connexion TCP.

3 Vulnérabilités applicatives

On considère le programme C suivant :

```
#include <string.h>
int main(int argc, char *argv[]) {
  char buffer[8];
  if (argc > 1) strcpy(buffer, argv[1]);
  return 0;
}
```

1. (4 points) Quel danger ce code présente t-il sachant que l'exécutable correspondant appartient à l'utilisateur `root`, possède le droit SUID (flag `s`) et peut être lancé par n'importe qui ? Vous détaillerez une attaque possible en étayant votre explication par un schéma de la pile d'exécution.
2. (2 points) Que faudrait-il faire pour éliminer ce danger (on ne demande pas de code, seulement une explication) ?

4 Sécurité en Java 2

(4 points) On considère le programme Java suivant. Détaillez précisément, ligne par ligne, ce que fait la portion de code située dans le bloc `try { ... }`.

```
import java.io.*;
import java.security.*;

class GenererSignature {
    public static void main(String args[]) {
        if (args.length != 1) {
            System.out.println("Il faut 1 argument sur la ligne de commande!");
        }
        else try {
                KeyPairGenerator keyGen =
                    KeyPairGenerator.getInstance("DSA", "SUN");

                SecureRandom random =
                    SecureRandom.getInstance("SHA1PRNG", "SUN");
                keyGen.initialize(1024, random);

                KeyPair pair = keyGen.generateKeyPair();
                PrivateKey priv = pair.getPrivate();
                PublicKey pub = pair.getPublic();

                Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");

                dsa.initSign(priv);

                FileInputStream fis = new FileInputStream(args[0]);
                BufferedInputStream bufin = new BufferedInputStream(fis);
                byte[] buffer = new byte[1024];
                int len;
                while (bufin.available() != 0) {
                    len = bufin.read(buffer);
                    dsa.update(buffer, 0, len);
                }
                bufin.close();

                byte[] realSig = dsa.sign();

                FileOutputStream sigfos = new FileOutputStream("sig");
                sigfos.write(realSig);
                sigfos.close();

                byte[] key = pub.getEncoded();
                FileOutputStream keyfos = new FileOutputStream("clepub");
                keyfos.write(key);
                keyfos.close();
            } catch (Exception e) {
                System.err.println("Exception " + e.toString());
            }
    }
}
```