Contrôle continu écrit numéro 2 de sécurité informatique

M1 d'informatique

27 avril 2011 – durée : 2 heures

Les notes de cours sur papier sont autorisées. Les machines sont interdites.

1 Buffer overflow (10 points)

On suppose que l'utilisateur root a écrit le programme C :

```
// Programme vulnerable.c
#include<string.h>
int main(int argc, char *argv[]) {
   char buffer[512];
   if (argc > 1) strcpy(buffer,argv[1]);
   return 0;
}
```

et qu'il a ensuite tapé dans un terminal :

```
# gcc-3.4 vulnerable.c -o vulnerable
# chmod u+s vulnerable
# echo 0 > /proc/sys/kernel/randomize_va_space
```

1. (3 points) Expliquez précisément ce que fait chacune des commandes ci-dessus.
2. (4 points) Quelle faille de sécurité l'exécutable `vulnerable` présente t-il dans ce contexte ? Décrivez un mode opératoire permettant à n'importe quel utilisateur d'obtenir un shell en mode root à partir de `vulnerable`. Vous illustrerez votre description par un schéma de la pile d'exécution.
3. (3 points) Certains systèmes proposent des piles non-exécutables qui font échouer la technique d'exploitation de buffer overflow décrite en cours et en TP. La technique "return to libc" permet de contourner cette protection de la pile et de lancer un shell à partir du programme attaqué. L'idée est d'utiliser des fonctions de la bibliothèque C standard, plutôt qu'un shellcode, sachant que ces fonctions ne sont pas stockées dans la pile : on remplit le buffer attaqué avec des caractères `A` pour le faire déborder et on s'arrange pour qu'il se termine par l'adresse de la fonction `system` (qui permet de lancer une commande shell à partir d'un programme C), suivie de l'adresse de la fonction `exit` (qui permet de quitter proprement un programme C), suivie de l'adresse de la chaîne de caractères `/bin/sh`. L'objectif est de sauter à l'adresse de `system` lorsque la fonction dans laquelle se trouve le buffer attaqué se termine.
   (a) En vous aidant de la manière dont on remplit le buffer attaqué, dites quel(s) argument(s) et quelle adresse de retour sont destinés à `system`.
   (b) On suppose que l'on a réussi à remplir le tableau `buffer` du programme `vulnerable` comme indiqué ci-dessus. Faites un schéma illustrant l'état de la pile d'exécution juste avant que la fonction `main` ne saute à l'adresse de `system`.

(c) On suppose qu'à partir de 520 caractères `A` injectés dans `buffer`, l'adresse de retour de `main` est écrasée complèment. On suppose que l'adresse de `system` est `0xb7ea88b0`, celle de `exit` est `0xb7eadb30` et celle de `/bin/sh` est `0xbffff76d`. Que doit taper un utilisateur dans un terminal pour obtenir un shell root à partir de l'exécutable `vulnerable` ?

2 Sécurité en Java 2 (6 points)

On suppose installée une version de Java supérieure ou égale à la 6.0. Donnez les commandes à taper dans un terminal, ou les instructions à enregistrer dans un fichier de configuration (précisez lequel), pour effectuer chacune des opérations suivantes.

1. (0.5 point) Générer un fichier keystore contenant une paire de clés RSA.
2. (0.5 point) Exporter un certificat auto-signé contenant la clé publique RSA générée.
3. (0.5 point) Créer une archive `jar` exécutable contenant les fichiers `A.class` et `B.class` sachant que le point d'entrée du programme se trouve dans `A.class`.
4. (0.5 point) Créer une version signée de l'archive exécutable en utilisant la paire de clés RSA générée plus haut.
5. (0.5 point) Visualiser le contenu du fichier `MANIFEST.MF` situé dans le répertoire `META-INF` de l'archive signée.
6. (1 point) Mettre en place une politique de sécurité pour que les classes signées avec la paire de clés RSA générée plus haut aient le droit de lire les fichiers placés dans toute l'arborescence du répertoire `home`.
7. (0.5 point) Exécuter l'archive signée en tenant compte de la politique de sécurité.
8. (2 points) Vérifier la validité de l'archive `truc.jar` reçue par email et signée par l'expéditeur au moyen d'un certificat auto-signé. En supposant que `truc.jar` est correctement formée, qu'elle n'a pas été altérée pendant son téléchargement et que le destinataire ne possède aucune information sur l'expéditeur dans son fichier keystore, que répond la commande de vérification ? Pourquoi ?

3 XSS (4 points)

Expliquez précisément les termes "XSS non-permanent" et "XSS permanent".