WARGAME

***

Ayitic – Port au Prince

11 – 16 Août 2014

***

Instructeur: LOISEAU Lucien

Ce document est une traduction basée sur les documents libres d'accès

https://www.pentesterlab.com/

Objectifs du Wargame

L'objectif de ce wargame est de se familiariser avec quelques vulnérabilités et mauvaises pratiques connues sous GNU/Linux. Vous aurez à faire face entre autre aux choses suivantes :

- Permissions
- Binaire SUID
- La manipulation des variables d'environnement
- Crontab
- Vulnérabilités logicielles

À la fin de ce wargame, vous aurez un aperçu des techniques d'exploitation locales d'un système Linux et une idée des attaques possibles à distances.

Préparation

Les épreuves sont disponible sous la format d'une image système que vous allez télécharger puis lancez localement dans une machine virtuelle l'aide de la commande `qemu`. Vous vous connecterez ensuite à ce système via SSH avec l'utilisateur `level1`. Chaque niveau est récompensé par un mot de passe permettant de vous connecter au niveau suivant. L'objectif est bien sur d'arriver au `level10`.

Téléchargement de l'image

L'instructeur va vous donner l'adresse IP où vous pourrez téléchargez l'image du wargame. Créez un répertoire avec au moins `520M` de libre que vous appelerez « `wargame-ayitic` ». Déplacez vous dans ce répertoire et téléchargez l'iso en ligne de commande avec l'outil `wget` :

```
$ wget <IP donné par l'instructeur>/wargame.qcom2
```

Vous pouvez maintenant lancer l'image

Lancement de l'image

Nous allons utiliser l'outil qemu pour lancer l'image disque dans une machine virtuelle. Vérifiez d'abord que qemu est bien installé :

```
$ qemu-system-i386 --version
QEMU emulator version 2.0.0, Copyright (c) 2003-2008 Fabrice Bellard
```

Si c'est le cas, vous pouvez alors lancer l'image de la façon suivante :

```
$ qemu-system-i386 -hda wargame.qcow2 -redir tcp:2222::22 -boot d
```

Le système virtualisé

Le système virtualisé est une distribution Debian standard. Il y a 10 utilisateurs, vous pouvez vous connecter avec l'utilisateur `level1` et le mot de passe `level1` de la façon suivante :

```
$ ssh -p 2222 level1@127.0.0.1
password : level1
```

Chaque utilisateur représente un niveau qu'il faut résoudre pour passer au niveau suivant. Les épreuves

se trouvent dans le répertoire home de chaque utilisateur. Résoudre une épreuve consiste généralement (mais pas toujours) à réussir à lire un mot de passe permettant ainsi de se connecter en ssh au niveau suivant. Les mots de passe de chaque utilisateurs se trouvent en clair dans le répertoire `/etc/wargame-password/`, mais comme vous vous en doutez les permissions ne permettent pas de le lire directement, il faudra pour cela exploiter des vulnérabilités :)

Avant de vous lancer

- Si vous bloquez sur une épreuve ne perdez pas espoir, faites preuve de patience et dans le pire des cas, appelez l'instructeur pour qu'il vous donne le petit coup de pouce nécessaire pour terminer l'épreuve.

- N'oubliez pas de consulter le manuel dès que vous ne comprenez pas une commande ou une fonction. Ayez le reflexe de faire `man`, par exemple pour la commande `find` :

```
$ man find
```

  La plupart des fonctions de la libc ont également leur page de manuel, par exemple on peut regarder la page de manuel de la fonction access de la façon suivante :

```
$ man access
```

  Il arrive qu'un nom de fonction de la libc ait le même nom qu'une commande. Il existe ainsi plusieurs manuel que l'on peut différencier avec un chiffre. Par exemple pour `printf` :

```
$ man printf       <- affiche le manuel de la commande printf (shell)
$ man 3 printf     <- affiche le manuel de la fonction printf (en C)
```

- Appelez l'instructeur à chaque fois que vous terminez une épreuve, un point supplémentaire sera donné au premier groupe qui termine une épreuve.

Les Épreuves

Level 1

Le mot de passe du Level 1 est level1. Connectez vous de la façon suivante :

```
$ ssh -p 2222 level1@127.0.0.1
password : level1
```

Dans cette épreuve vous devez trouver ce qui est caché ;) regardez bien !

Level 2

Dans ce niveau vous trouverez un programme SUID nommé `unknown` ainsi que son code source `unknown.c` Déterminez ce que fait ce programme et utilisez le à votre avantage.

```
~~~~ unknown.c ~~~~
#include <stdio.h>
#include <unistd.h>

int main(int argc, char** argv)
{
  if(argc < 2)
  {
    printf("*** FILE PRINTER ***\n");
  }

 if(access(argv[1], F_OK) != -1)
 {
   char command[100];
   snprintf(command, 100, "/bin/cat %s", argv[1]);
   system(command);
 }
 return 0;
}
```

Level 3

Ce niveau est similaire au précédent excepté qu'une protection a cette fois mis en place. Cependant elle n'est pas suffisante, trouvez comment détourner la protection (indice : jouez avec les noms de fichiers)

~~~~ unknown.c ~~~~

```c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char** argv)
{
    if(argc < 2)
    {
        printf("*** FILE PRINTER ***\n");
    }

    if(access(argv[1], R_OK) != -1)
    {
        char command[100];
        snprintf(command, 100, "/bin/cat %s", argv[1]);
        system(command);
    }
    else
    {
        printf("you are not authorized\n");
    }
    return 0;
}
```

Level 4

Dans ce niveau une vulnérabilité permet d'exécuter n'importe quel programme, saurez vous trouver comment ?

~~~~ vuln.c ~~~~

```c
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv, char **envp)
{
  gid_t gid;
  uid_t uid;
  gid = getegid();
  uid = geteuid();

  setresgid(gid, gid, gid);
  setresuid(uid, uid, uid);

  system("/usr/bin/env echo and now what?");
}
```

Level 5

Regardez le répertoire utilisateur de `level5`. Un crontab est appelé par l'utilisateur `level6` toute les

quelques minutes.

Level 6

L'utilisateur `level7` a écrit son premier programme en C permettant de pinger un hôte fournis en paramètre et d'afficher un message indiquant si l'hôte est up ou non. Malheuresement ce programme présente une vulnérabilité, à vous de l'exploiter !

```
~~~~ ping.c ~~~~
#include <stdio.h>
#include <unistd.h>

int main(int argc, char** argv)
{
  if(argc < 2)
  {
    printf("il faut donner un hote a pinger\n");
    return 0;
  }

  char command[500];
  snprintf(command,500,"ping -c 3 -W 5 -n %s 2>&1 >/dev/null",argv[1]);

  int ret = system(command);

  printf("code de retour: %d -- ",ret);
  if(ret == 0){
    printf("ca ping !\n");
  } else {
    printf("ca ne ping pas!\n");
  }

  return 0;
}
```

Level 7

Un programme en C a été écrit afin de faciliter le transfert de fichier vers un serveur. Pour ne pas transférer n'importe quel fichier le programmeur a fait attention à vérifier les permissions (à l'aide de la méthode `access` déjà vu dans les niveaux précédent) mais une vulnérabilité dans le code permet de contourner cette vérification.

```
~~~~ tocttou.c ~~~~
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <fcntl.h>
```

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

int main(int argc, char **argv)
{
  char *file;
  char *host;

  if(argc < 3) {
    printf("USAGE: %s <FILE> <HOST>\n\tEnvoie le fichier FILE à HOST sur le port 18211\n", argv[0]);
    exit(1);
  }

  file = argv[1];
  host = argv[2];

  // on vérifie que l'utilisateur ait le droit de lire le fichier
  if(access(argv[1], R_OK) == 0) {
    int fd;
    int ffd;
    int rc;
    struct sockaddr_in sin;
    char buffer[4096];

    fd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&sin, 0, sizeof(struct sockaddr_in));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = inet_addr(host);
    sin.sin_port = htons(18211);

    // on se connecte à HOTE sur le port 18211
    printf("Connexion à %s:18211 .. ", host); fflush(stdout);
    if(connect(fd, (void *)&sin, sizeof(struct sockaddr_in)) == -1) {
      printf("La connexion a échoué %s\n", host);
      exit(EXIT_FAILURE);
    }

    // on patiente 100 millisecondes
    usleep(100000);

    printf("Connecté!\nOn envoie le fichier .. "); fflush(stdout);

    // on ouvre le fichier en lecture
    ffd = open(file, O_RDONLY);
    if(ffd == -1) {
      printf("Impossible d'ouvrir le fichier\n");
      exit(EXIT_FAILURE);
    }

    // on lit les données du fichier
    rc = read(ffd, buffer, sizeof(buffer));
    if(rc == -1) {
      printf("Erreur lors de la lecture du fichier: %s\n", strerror(errno));
      exit(EXIT_FAILURE);
```

```
    }

    // on les envoies sur le réseau
    write(fd, buffer, rc);
    printf("succes\n");
  } else {
    printf("Vous n'avez pas les permissions pour ce fichier %s\n", file);
  }
}
```

Level 8

Un programme nommé `unknow` se trouve dans le repertoire utilisateur. Essayez de comprendre ce que ce programme fait et détournez le à votre avantage (indice : regarder la page de manuel de la commande `ltrace`).

Level 9

Dans ce niveau, un programme vous demande de lui donner en paramètre le mot de passe du `level9` et du `level10`. Si un des deux mots de passe est faux, c'est perdu et sinon c'est gagné. Vous avez déjà le mot de passe du `level9` mais comment faire pour trouver le mot de passe du `level10` ?:)

Level 10

Félicitation vous êtes arrivé au dernier niveau ! Cette dernière épreuve est un peu similaire à la précédente dans la mesure ou le programme attend cette fois un code (et pas une chaîne de caractère). Courage !