

Analyse dynamique depuis l'espace noyau avec Kolumbo

Julien Desfossez

Révolution Linux, Sherbrooke, Québec, Canada
Université de Sherbrooke, Sherbrooke, Québec, Canada

SSTIC 2009



Insertion de faux points d'arrêts

- *int3* déclenche le signal *SIGTRAP*
- Par défaut, un debugger va interpréter cette instruction comme un point d'arrêt
- Démo

Appels systèmes

Exemple

Exemple

```
SECTION .data
msg      db      "Hello, world!",0xa ;
len      equ     $ - msg

SECTION .text
mov      eax,4          ; write system call
mov      ebx,1          ; file (stdout)
mov      ecx,msg        ; string
mov      edx,len        ; strlen
int      0x80           ; call kernel
```

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Si processus suivi

- Traitement local
 - Lecture de la stack noyau du processus
 - Affichage des registres
 - Affichage de l'adresse de retour
- Traitement normal

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Si processus suivi

- Traitement local
 - Lecture de la stack noyau du processus
 - Affichage des registres
 - Affichage de l'adresse de retour
- Traitement normal

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Si processus suivi

- Traitement local
 - Lecture de la stack noyau du processus
 - Affichage des registres
 - Affichage de l'adresse de retour
- Traitement normal

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Si processus suivi

- Traitement local
 - Lecture de la stack noyau du processus
 - Affichage des registres
 - Affichage de l'adresse de retour
- Traitement normal

Suivi des appels systèmes

Fonctionnement

Interception de l'interruption 0x80 (*idtr*)

Si processus suivi

- Traitement local
 - Lecture de la stack noyau du processus
 - Affichage des registres
 - Affichage de l'adresse de retour
- Traitement normal

Sinon

- Traitement normal

Suivi des fork

Fonctionnement

- Interception de fork, vfork et clone
- Table de processus actifs
 - Si le parent est un processus suivi : ajout du PID du fils
 - Lors d'un appel à *exit* ou *exit_group* suppression du processus dans la table

Appels systèmes

Interception

Traitement dans le noyau

- Table syscall_table
- Index eax

Appels systèmes

Interception

Traitement dans le noyau

- Table syscall_table
- Index eax

Appels systèmes

Interception

Traitement dans le noyau

- Table `syscall_table`
- Index `eax`

Interception

```
asmlinkage long my_open(param, ...) {  
    traitement_special();  
    return old_open(param, ...);  
}
```

```
syscall_table[__NR_open] = (void *)my_open;
```

Affichage de la table de pages

Fonctionnement

- Déclenchement au démarrage du processus
- Déclenchement sur certains appels systèmes

Démo

Mode Dump

- Récupération du contenu de la mémoire d'un processus
- Déclenché lors d'un appel système configuré à l'avance
- Utile contre certains *packers* (UPX)

Gestion de la mémoire virtuelle

- Deux segments principaux : .text et .data
- Permissions sur chaque segments (RWX)
- Table des segments disponible dans `/proc/< pid >/maps`

Gestion de la mémoire virtuelle

Exemple

```
08048000-0804f000 r-xp      /bin/cat
0804f000-08050000 rw-p      /bin/cat
08050000-08071000 rw-p      [heap]
bffe0000-c0000000 rw-p      [stack]
```

Lien entre un ELF et son image mémoire

- Chaque segment dans une zone mémoire distincte
- Programme simple : en-têtes des sections sont en mémoire
- Programme chiffré : en-têtes des sections ne sont pas en mémoire

Échange de données avec le noyau

- Tampon circulaire (ring buffer)
- Périphérique de type périphérique de type caractère (/dev/kolumbo)
- Processus utilisateur pour lire le périphérique

Reconstruction d'un programme simple

Fonctionnement

- Récupération des en-têtes ELF
- Récupération des segments *text* et *data*
- Ajout de l'alignement entre les segments
- Programme exécutable

Reconstruction d'un programme simple

Démonstration

```
# ./hello
Hello, world!
# cat /dev/kolumbo > dump
# chmod +x dump
# ./dump
Hello, world!
```

Reconstruction d'un programme chiffré

- Récupération de l'en-tête ELF de base uniquement
- Récupération des pages contenant les segments *text* et *data*
- Les zones non utilisées des pages sont complétées par des 0
- Programme non exécutable mais déchiffré

Anti-anti-trace

- Permet l'analyse avec les outils habituels
- Invisible pour le programme

Anti-anti-trace sur ptrace

- Interception de l'appel système ptrace
- Changement de la valeur de retour

Démo avec le virus RST-B

- Scan SSH
- Infection de binaires
- Système anti-trace

Démo avec le virus RST-B (Murphy)

- Scan SSH
- Infection de binaires
- Système anti-trace

Étude des appels systèmes avec strace

Avec suivi des fork

```
$ strace -fF ./malware.exe
execve("./malware.exe", [ "./malware.exe" ], ) = 0
fork(Process 30588 attached)                = 30588
[pid 30588] ptrace(PTRACE_TRACEME, 0, 0x1, 0)
= -1 EPERM (Operation not permitted)
[pid 30588] _exit(0)                        = ?
Process 30588 detached
--- SIGCHLD (Child exited) @ 0 (0) ---
```

Démonstration du système anti-anti-trace

```
$ strace -fF ./malware.exe
execve("./malware.exe", [ "./malware.exe" ], ) = 0
fork(Process 30621 attached)                = 30621
[pid 30621] ptrace(PTRACE_TRACEME, 0, 0x1, 0) = 1

...
[pid 30621] open("grep", O_RDWR)           = 4
[pid 30621] lseek(4, 96104, SEEK_SET)       = 96104
[pid 30621] write(4, "SQRVW\353\vX\211\303\201
\353\27\0\0\0\377\340\350\360\377"... , 34) = 34

...
```