
EPITECH.
L'ECOLE DE L'EXPERTISE INFORMATIQUE
Quand un passionné devient un expert reconnu
European Institute of Information Technology
Titre homologué par l'Etat - Niveau I (CNCP)
medega_j

LE MODULE

Déroulement :
- 2 Cours
- 2 TP
- Deux projets (Strace & Ftrace)

Objectifs :
- Approfondir la connaissance du format ELF
- Comprendre les mécanismes d'éditions de liens dynamique
- Mieux comprendre comment fonctionnent les instructions x86_64
- Comprendre le fonctionnement interne de GDB
- Approfondir votre connaissance des mécanismes du C & des processeurs

Environnement :
- Architecture x86_64 (64 bits)
- Linux

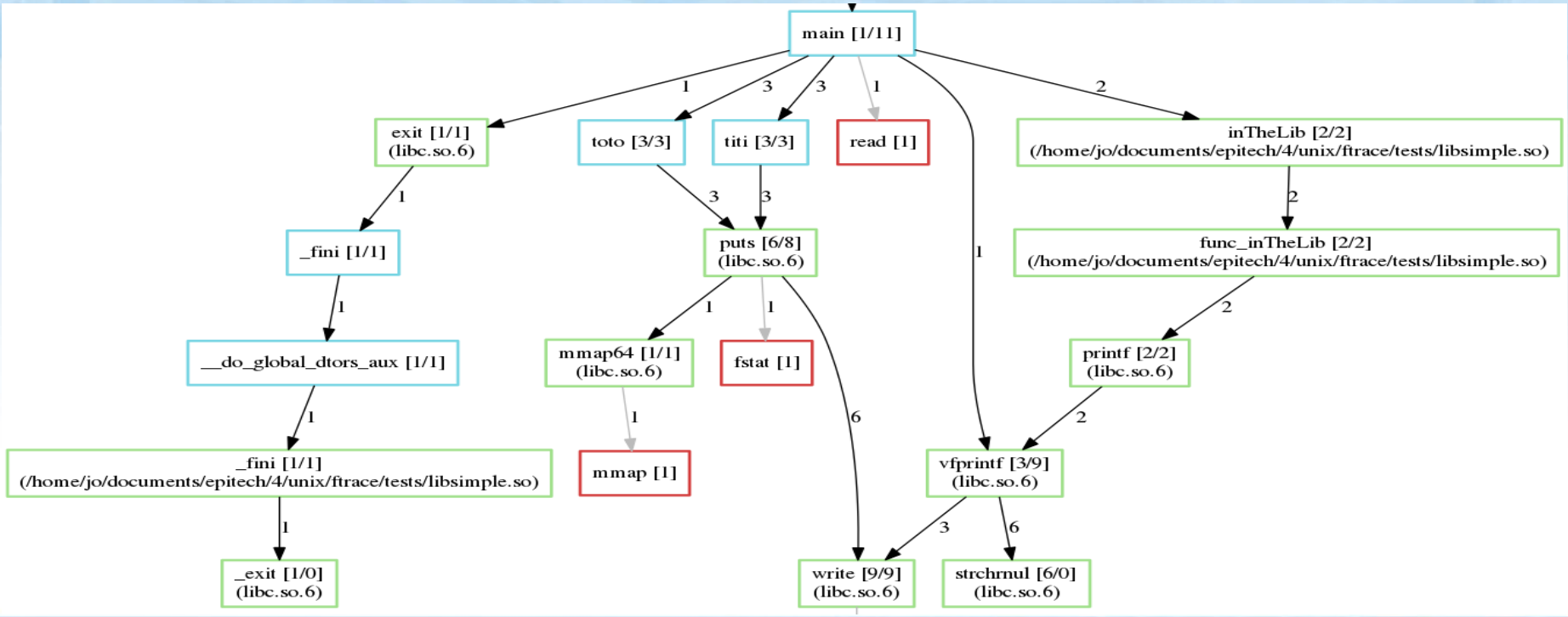
LES PROJETS : STRACE

- Traceur d'appels systèmes (syscall)
- Utilisé à des fins de débuggage
- Facile, ce projet permet de se familiariser avec l'interface de *tracing* des processus.

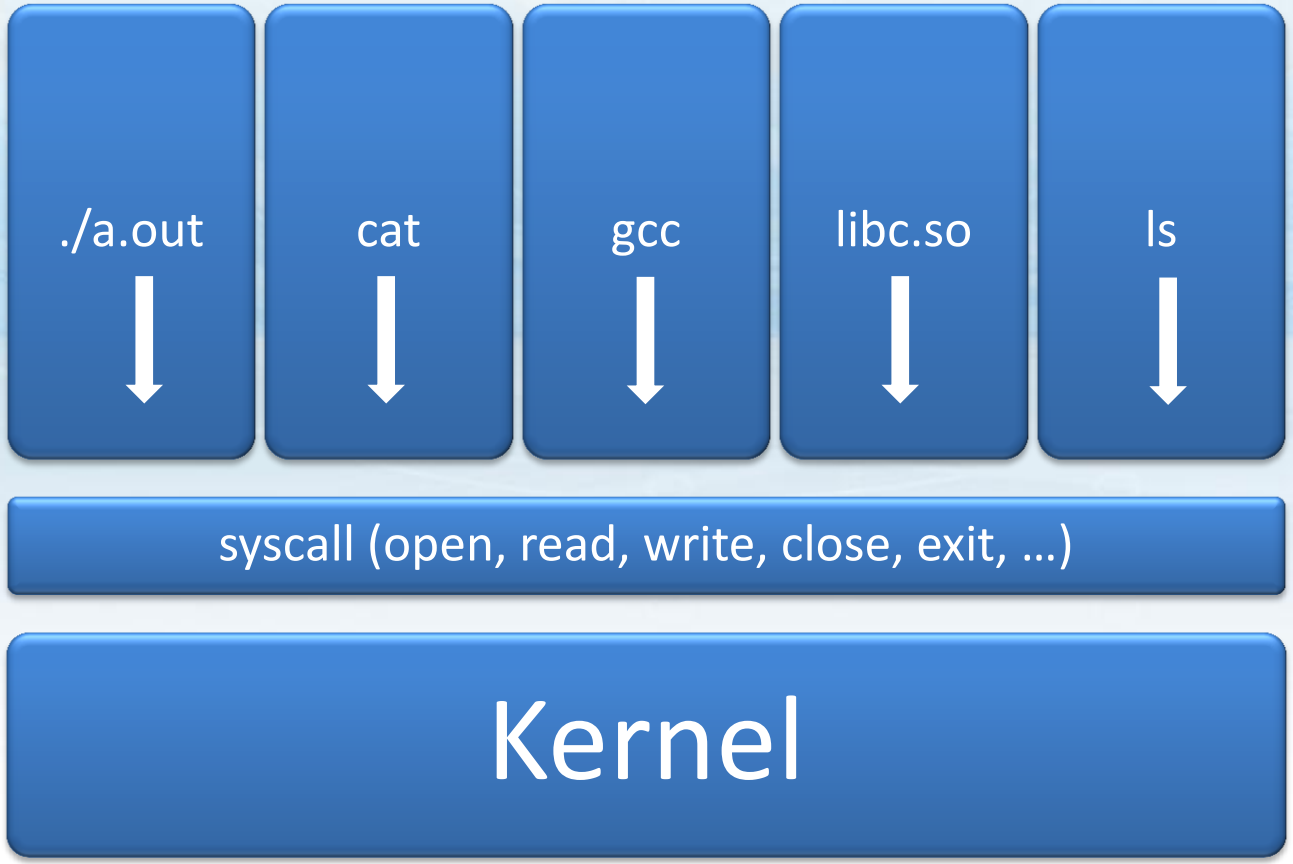
```
medega_j@ubuntu:~$ strace ./rtv0
execve("./rtv0", ["./rtv0"], [/* 42 vars */]) = 0
brk(0)                                  = 0x8bb3000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77bf000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=75571, ...}) = 0
mmap2(NULL, 75571, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77ac000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/i386-linux-gnu/libXext.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\260%\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=55888, ...}) = 0
mmap2(NULL, 59092, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x893000
mmap2(0x8a0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc) = 0x8a0000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/i386-linux-gnu/libX11.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0 \21\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=1151776, ...}) = 0
```

LES PROJETS : FTRACE

- Générateur de CallGraph
- Permet d'avoir une vue d'ensemble du flux d'exécution d'un processus (appels de fonctions -statiques & dynamiques-, syscalls, signaux, ..)
- Bien plus complexe que le Strace, demande une parfaite compréhension des mécanismes d'édition de lien dynamique & des différents appels.



Kernel & Appels systèmes



- Liste des syscalls : /usr/include/asm/unistd_64.h

PTRACE

ptrace est un appel système qui permet de monitorer & de contrôler le flux d'exécution d'un processus, sur un modèle « *debugger/debuggee* ».

ptrace fourni différentes primitives, dont :

- *PTRACE_TRACEME, PTRACE_ATTACH*
- *PTRACE_CONT*
- *PTRACE_SINGLESTEP*
- *PTRACE_SYSCALL*
- *PTRACE_PEEKTEXT, PTRACE_POKETEXT*
- *PTRACE_GETREGS, PTRACE_SETREGS*

- « man ptrace »
- « man wait4 »

STRACE : HOW TO ?

fork (si le processus à tracer n'existe pas)

PTRACE_TRACEME dans le fils (ou PTRACE_ATTACH si le processus existe déjà)

wait4 dans le père (pour attendre les changements de statut du fils)

execve dans le fils (pour lancer le binaire à tracer)

Tracer à l'aide de PTRACE_SINGLESTEP

Vérifiez à chaque instruction si on a pas un syscall

Le cas échéant, récupérer le nom du syscall appelé & ses paramètres ainsi que sa valeur de retour

RAPPEL SUR LES REGISTRES

Nous avons appris que lors de l'exécution d'un processus sur une architecture x86_64/Linux :

- Le registre *rip* contient l'adresse de la prochaine instruction qui va être exécutée par le processeur.

- Avant l'exécution d'un syscall, *rax* contient le numéro du syscall (voir *orig_rax* si on utilise PTRACE_SYSCALL).

- Avant l'exécution d'un syscall, les paramètres de l'appel sont placés par ordre dans *rdi*, *rsi*, *rdx*, *r10*, *r8* et *r9.* Au delà d'un 6ème paramètre, ils sont placés sur la pile (rsp..).

- Après l'exécution d'un syscall, la valeur de retour est placée dans *rax.*

SYSCALL

- A l'échelle de l'assembleur, un appel système est déclenché généralement, sous une architecture x86_64/Linux, par l'instruction *« syscall »*.
- Man intel (*http://download.intel.com/products/processor/manual/325383.pdf*) :

SYSCALL—Fast System Call

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F 05	SYSCALL	NP	Valid	Invalid	Fast call to privilege level 0 system procedures.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

- Voir aussi *« int 0x80 »* & *« SYSENTER »*.
- Attention à l'endianess.

Questions ?