Module SEL

Editions de liens

- Edition de liens statique
  - Fonctionnement
  - Limitations
- Edition de liens dynamique

istic Informatique Électronique

Introduction aux systèmes d'exploitation

Edition de liens statique et dynamique

istic Informatique Électronique

Rappels sur la liaison

- Objets logiques (variables, procédures, fichiers)
- Objets physiques (valeurs, emplacements mémoire)
- Liaison
  - Passage du nom de l'objet logique à sa représentation concrète
- Remarque
  - Dans les systèmes à pagination, on considérera uniquement le passage identificateur → adresse virtuelle, le passage @-virtuelle à @- physique étant pris en charge par le mécanisme de pagination

Rappels sur la liaison

- Logiciels contribuant à la liaison
  - Traducteur (compilateur ou assembleur) : traduction de code source en code machine
    - Traduction des identificateurs d'objets dans une représentation interne
  - Editeur de liens : regroupement du code et des données de plusieurs modules en résolvant les références externes
  - Chargeur : initialisation de la machine (processeur +mémoire) pour que le programme puisse être exécuté

Rappels sur la liaison

- Moment de la liaison
  - Edition de liens statique : tous les identificateurs ont été traduits avant exécution
  - Edition de liens dynamique : il reste des identificateurs de références externes non résolus au début de l'exécution
    - Utilisation typique : fonctions de bibliothèque

Edition de liens statique

- Tous les identificateurs ont été transformés en adresses avant exécution
- Fonctions appelées incorporées dans l'exécutable
- Un exemple :

```
int main () {
      int x = 0;
      int y = foo(x);
}
```

main.c

```
int foo (int x) {
      return x+1;
}
```

foo.c

```
gcc -O0 -c main.c
gcc -O0 -c foo.c
gcc -o main main.o foo.o
```

Edition de liens statique

- Un exemple :

Adresses ne sont encore que des offsets

```
0:   55                      push   %rbp
1:   48 89 e5                mov    %rsp,%rbp
4:   48 83 ec 10             sub    $0x10,%rsp
8:   c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
f:   8b 45 f8                mov    -0x8(%rbp),%eax
12:  89 c7                   mov    %eax,%edi
14:  e8 00 00 00 00          callq  19 <main+0x19>
19:  89 45 fc                mov    %eax,-0x4(%rbp)
1c:  c9                      leaveq
1d:  c3                      retq
```

Adresse de l'appel pas encore rempli

Code objet main.o (gcc –c –O0, désassemblé – objdump –d main.o)

```
U foo
0000000000000000 T main
```

foo U (Undefined)

Table des symboles main.o (nm main.o)

Edition de liens statique

- Un exemple :

Adresses finalisées (adresses virtuelles)

```
00000000004004ed <main>:
4004ed:  55                      push   %rbp
4004ee:  48 89 e5                mov    %rsp,%rbp
4004f1:  48 83 ec 10             sub    $0x10,%rsp
4004f5:  c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
4004fc:  8b 45 f8                mov    -0x8(%rbp),%eax
4004ff:  89 c7                   mov    %eax,%edi
400501:  e8 05 00 00 00          callq  40050b <foo>
400506:  89 45 fc                mov    %eax,-0x4(%rbp)
400509:  c9                      leaveq
40050a:  c3                      retq
000000000040050b <foo>:
40050b:  55                   push   %rbp
40050c:  48 89 e5             mov    %rsp,%rbp
40050f:  89 7d fc             mov    %edi,-0x4(%rbp)
400512:  8b 45 fc             mov    -0x4(%rbp),%eax
400515:  83 c0 01             add    $0x1,%eax
400518:  5d                   pop    %rbp
400519:  c3                   retq
```

Adresse (offset) de l'appel rempli

Code de foo dans l'exécutable

Code binaire main (objdump –d main)

Edition de liens statique

- Limites
  - Gestion des évolutions difficile (versions, corrections de bugs).
  - Nécessité de refaire l'édition de liens pour bénéficier d'une nouvelle version d'une bibliothèque
  - Consommation d'espace disque et mémoire inutiles pour les copies des objets liés

Edition de liens dynamique

- Propriétés
  - On ne connaît pas l'adresse finale du symbole avant exécution
  - Fonctions de bibliothèques dynamiques non incorporées à l'exécutable
- Instants possibles de la liaison
  - Au chargement du module contenant une référence externe ou
  - A la première référence à un objet externe

Edition de liens dynamique

- LD_LIBRARY_PATH
  - Où chercher les bibliothèques (statiques ou dynamiques)
- Remarques
  - Edition de liens dynamique va de pair avec partage des bibliothèques ⇒ les bibliothèques partagées doivent être réentrantes (supporter d'être exécutées par plusieurs processus simultanément)
  - Edition de liens dynamique ⇒ les symboles non résolus doivent être conservés plus longtemps qu'avec une édition de liens statique
  - Résolution des liens inconnus plus tardive : portabilité, écrasement de bibliothèques

Edition de liens dynamique

- Un exemple (foo est maintenant une librairie dynamique)

```
int main () {
      int x = 0;
      int y = foo(x);
}
```

main.c

```
int foo (int x) {
      return x+1;
}
```

foo.c

```
gcc -O0 -c main.c
gcc -O0 -fPIC -c foo.c
gcc -shared -o libfoo.so foo.o
gcc -o main main.o -L. -lfoo.o
```

Recherche librairie dans .

On lie avec libfoo

Création lib dynamique (.so)

Edition de liens dynamique

- Un exemple :

Adresses finalisées pour main (adresses virtuelles)

```
0000000000400590 <foo@plt>:
400590:    ff 25 92 0a 20 00      jmpq   *0x200a92(%rip)      # 601028 <_GLOBAL_OFFSET_TABLE_+0x28>
400596:    68 02 00 00 00         pushq  $0x2
40059b:    e9 c0 ff ff ff         jmpq   400560 <_init+0x20>
000000000040068d <main>:
40068d:    55                     push   %rbp
40068e:    48 89 e5               mov    %rsp,%rbp
400691:    48 83 ec 10            sub    $0x10,%rsp
400695:    c7 45 f8 00 00 00 00   movl   $0x0,-0x8(%rbp)
40069c:    8b 45 f8               mov    -0x8(%rbp),%eax
40069f:    89 c7                  mov    %eax,%edi
4006a1:    e8 ea fe ff ff         callq  400590 <foo@plt>
4006a6:    89 45 fc               mov    %eax,-0x4(%rbp)
4006a9:    c9                     leaveq
4006aa:    c3                     retq
```

Adresse (offset) de l'appel rempli - appel à foo@plt

Code de foo n'est pas dans l'exécutable, juste foo@plt

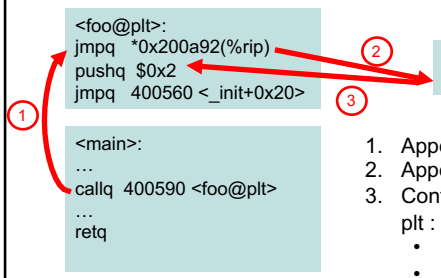
Code binaire main (objdump –d main)

Edition de liens dynamique Fonctionnement Linux

- Structures de données
  - Région *plt* (procedure lookup table)
    - Une par processus
    - Une entrée de 16 octets par fonction appelée
    - Entrée 0 : permet d'appeler le linker dynamique
  - Région *got* (global offset table)
    - Une fois le linker dynamique appelé, contient l'adresse des fonctions de lib dynamiques
  - Liées à des adresses fixes lors du link
    - (gcc -o main main.o -L. -lfoo.o)
  - Source :
    - http://www.segmentationfault.fr/linux/role-plt-got-ld-so/

Edition de liens dynamique
Fonctionnement Linux

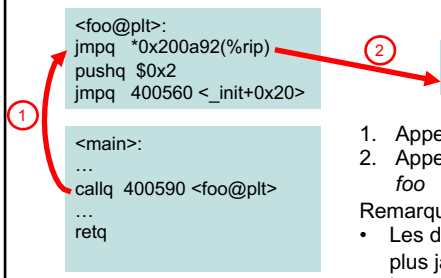
- En images
  - Etat avant liaison dynamique
    (juste un peu schématisé)



1. Appel via plt (jamais changé)
2. Appel indirect via got (jamais changé)
3. Contenu de base de la got entraîne retour à la plt :
   - Empilement de 0x2 (idf de foo)
   - jmp entrée 1 de plt (linker dynamique !)

Edition de liens dynamique
Fonctionnement Linux

- En images
  - Etat après liaison dynamique
    - Le linker a mis l'adresse de foo dans la got !



1. Appel via plt (jamais changé)
2. Appel indirect via *got*, qui branche directement à *foo*

Remarques :
- Les deux instructions suivantes de la plt ne sont plus jamais exécutées !
- Les appels coûtent un peu plus cher que normalement (surcoût = branchement indirect)

Edition de liens dynamique

- Spécificités édition de lien dynamique Linux
  - Par défaut, liaison à la première utilisation d'un symbole
  - Variable d'environnement LD_BIND_NOW : si chaîne non vide, on résout tous les symboles au chargement
  - Fonctions pour incorporer et lier dynamiquement des bibliothèques : dlopen, dlclose, dlsym, dlerror
    - N'ont pas à être connues avant exécution

Commandes relatives à d'édition de liens

- Commandes relatives à l'édition de lien
  - ld : éditeur de lien (linker)
    - Option par défaut = dynamique
  - nm : visualisation de la table des symboles
  - objdump : visualisation exécutable (dé-ssemblage, table des symboles)
  - readelf : visualisation format elf (sections, etc)
  - strip : suppression sections et symboles d'un exécutable

Notes diverses

- Ecrasement binaire pendant exécution
  - Erreur : text file busy (system dependent)
- Ecrasement .so pendant utilisation
  - Pas d'erreur et l'exécution continue (au moins sur mon cas très simple)
- Variable dans .so
  - Pas partagée entre les processus, dupliquée (mais même adresse virtuelle, semble similaire au fork)
- Bib non thread_compliant: à creuser