Année Universitaire 2021-2022 session 1 d'automne

Parcours : TCY Master CSI **UE :** 4TCY904U
Épreuve : Examen Sécurité Système
Date : 16 décembre **Heure :** 14h **Durée :** 3h
Modalités : Épreuve sans document, calculatrice autorisée
Épreuve de : Philippe Virouleau, Samuel Thibault 6 pages

université de BORDEAUX

Collège Sciences et technologie

Exercice 1. Compilation

Le programme ci-dessous vise à implémenter un simple programme calculant une puissance :

```c
#include <math.h>

int main(int argc, char **argv) {
  (void)argv;
  printf("pow: %f", pow(argc, 3));
  return 0;
}
```

Voici un extrait de la documentation de la fonction *pow* :

```
double pow(double x, double y);
This function return the value of x raised to the power of y.
```

J'essaye de compiler ce programme, et voici ce que j'obtiens :

```
$ clang pow.c -o pow
pow.c:5:3: warning: implicitly declaring library function 'printf' with type 'int (const char *, ...)'
  printf("pow: %f", pow(argc, 3));
  ^
1 warning generated.
/usr/bin/ld: /tmp/pow-b981e1.o: in function `main':
pow.c:(.text+0x24): undefined reference to `pow'
clang-12: error: linker command failed with exit code 1 (use -v to see invocation)
```

Q1.1 Que signifient l'avertissement et l'erreur affichés ?

Q1.2 Quelles modifications devriez-vous faire au programme et à la ligne de commande pour que le programme fonctionne comme attendu ?

Q1.3 En supposant que je corrige les points ci-dessus, quelle est la sortie du programme si je lance la commande suivante ?

```
./pow 3
```

Exercice 2. Représentation intermédiaire LLVM

Étant donné le programme suivant :

```c
int main(int argc, char **argv) {
  size_t sum = 0;
  for (size_t i = 0; i < strlen(argv[0]); i++) {
    sum += argv[0][i];
  }
  printf("sum: %zu\n", sum);
  return 0;
}
```

Une compilation avec *clang* donne la représentation intermédiaire LLVM suivante :

```
@.str = constant [10 x i8] c"sum: %zu\0A\00"
define i32 @main(i32 %0, i8** %1) {
  %3 = load i8*, i8** %1
  %4 = load i8, i8* %3
  %5 = icmp eq i8 %4, 0
  br i1 %5, label %9, label %6
6:                                                ; preds = %2
  %7 = load i8*, i8** %1
  %8 = call i64 @strlen(i8* nonnull dereferenceable(1) %7)
  br label %12
9:                                                ; preds = %12, %2
  %10 = phi i64 [ 0, %2 ], [ %19, %12 ]
  %11 = call i32 (i8*, ...) \
    @printf(i8* getelementptr inbounds ([10 x i8], [10 x i8]* @.str, i64 0, i64 0), i64 %10)
  ret i32 0
12:                                               ; preds = %6, %12
  %13 = phi i8* [ %3, %6 ], [ %7, %12 ]
  %14 = phi i64 [ 0, %6 ], [ %20, %12 ]
  %15 = phi i64 [ 0, %6 ], [ %19, %12 ]
  %16 = getelementptr inbounds i8, i8* %13, i64 %14
  %17 = load i8, i8* %16
  %18 = sext i8 %17 to i64
  %19 = add i64 %15, %18
  %20 = add nuw i64 %14, 1
  %21 = icmp ult i64 %20, %8
  br i1 %21, label %12, label %9
}
```

Q2.1 Si *@main* est une fonction et *%3* est une instruction, quel type d'élément est *%6* ?

Q2.2 Dans quel bloc se trouve le corp de la boucle ?

Q2.3 Le code a été généré sans activer toutes les optimisations. La valeur *%14* est utilisée comme variable d'itération de la boucle (i dans le code original), serait-il possible de remplacer l'instruction suivante :

```
%20 = add nuw i64 %14, 1
```

par l'instruction suivante :

```
%14 = add nuw i64 %14, 1
```

Q2.4 L'instruction suivante est un noeud *phi* :

```
%14 = phi i64 [ 0, %6 ], [ %20, %12 ]
```

Dans quel cas *%14* peut-il prendre la valeur *0* ?

Q2.5 Rappelez l'utilité de cette instruction dans le contexte de la représentation intermédiaire LLVM.

Exercice 3. Questions de cours sur l'obfuscation

Pour les questions suivantes, une et une seule réponse est correcte. Une mauvaise réponse ne retire pas de point.

Q3.1 Laquelle de ces affirmations concernant l'obfuscation est vraie ?

1. Obfusquer un programme peut en changer la sémantique.
2. L'obfuscation offre une protection absolue contre un attaquant.
3. Obfusquer un programme peut permettre de corriger des bugs.
4. L'obfuscation est un compromis entre protection et performance du programme.

Q3.2 Dans le contexte de ce cours, que signifie MBA ?

1. Mississipi Beekeepers Association.
2. Mixed Boolean-Arithmetic.
3. Master of Business Administration.
4. Mechanical Brake Assist.

Q3.3 Quel outil de tests avez-vous utilisé dans les tps ?

1. GoogleTest.
2. Lit.
3. SpeedTest.
4. JUnit.

Q3.4 Quel type de protections protège contre l'instrumentation de code ?

1. Les protections statiques.
2. Les protections dynamiques.
3. Les protections mécaniques.
4. La réponse D.

Q3.5 Sur quelle propriété de ptrace s'appuie-t-on pour détecter si l'on s'est fait attaché par gdb ?

1. On ne peut appeler ptrace qu'une seule fois sur un processus donné.
2. ptrace retourne une valeur spécifique lorsqu'il sait que gdb est lancé.
3. Il y a une fonction ptrace spécifique pour détecter gdb.
4. ptrace fait partie de gdb.

Exercice 4. Protection d'un programme

Le *Cyclic Redundancy Check* est une manière de pouvoir détecter si certains blocs de données ont changés (par exemple lors d'un transfert). Cette vérification consiste à implémenter une fonction de hashage, dont une implémentation possible est donnée ci-dessous :

```
unsigned int crc32(const unsigned char *message) {
  int i, j;
  unsigned int byte, crc, mask;

  i = 0;
  crc = 0xFFFFFFFF;
  while (message[i] != 0) {
    byte = message[i]; // Get next byte.
    crc = crc ^ byte;
    for (j = 7; j >= 0; j--) { // Do eight times.
      mask = -(crc & 1);
```

```
        crc = (crc >> 1) ^ (0xEDB88320 & mask);
      }
      i = i + 1;
    }
    return ~crc;
}
```

Je souhaite protéger cet algorithme, de manière à ce qu'un attaquant utilisant une analyse statique du binaire ne puisse pas facilement comprendre que cette fonction effectue un *CRC32*.

Q4.1 Quels sont les élements du code que vous jugez utile de protéger ?

Q4.2 Quelles protections / transformations pouvez-vous suggérer d'appliquer, et dans quel ordre (pensez à justifier vos choix) ?

Il y a une valeur *signante* [1] dans le code. Même si sa valeur n'apparait pas statiquement dans le binaire une fois les protections appliquées, elle pourrait toujours apparaître dans les registres lors de l'exécution.

Q4.3 Quel(s) type(s) d'attaque pourrait révéler cette valeur à l'attaquant ?

Q4.4 Quel(s) type(s) de protections pourriez-vous suggérer pour la contrer ?

Exercice 5. Reflexion sur un choix de protection

Voici une citation extraite d'un article [2] :

"Afin de garantir la protection de son jeu Devil May Cry 5, Capcom avait opté pour le DRM Denuvo. Un système de protection assez décrié, puisqu'il est accusé d'avoir un impact non négligeable sur les performances. Assez ironiquement, le DRM avait été craqué le jour même de la sortie du jeu, le 8 mars 2019. Le studio se décide finalement à retirer Denuvo de son titre via une mise à jour."

Ce n'est pas la première fois que le DRM implémenté par Denuvo est mis en cause dans des problèmes de performances, et ce n'est pas non plus la première fois que Capcom utilise la stratégie d'utiliser une protection pour les premiers mois de commercialisation de leur jeu, pour ensuite la retirer dans un patch ultérieur.

Q5.1 En supposant que Capcom utilise une protection pour garantir l'utilisation d'une license valide, quel(s) type(e)s de protection(s) (comme par exemple vu en cours et TPs) pensez-vous que le DRM de Denuvo contient ?

Q5.2 Pouvez-vous emettre une ou plusieurs hypothèses sur la source des problèmes de performances ?

Q5.3 Quelles solutions techniques pourriez-vous proposer aux développeurs pour rendre la protection plus pertinente et limiter les problèmes de performances ?

Q5.4 En plus de la protection de license, le DRM de Denuvo propose une protection "anti tricheur" : l'objectif est d'empêcher les utilisateurs d'obtenir un avantage inéquitable (voir à travers les murs, viser automatiquement, etc...). Ce type d'avantage peut être obtenu en changeant le binaire initialement distribué par l'éditeur du jeu. Quel(s) type(s) de protection(s) peu(ven)t bien être implémenté(s) dans cette protection "anti tricheur" ?

1. caractérisque de l'algorithme
2. `https://www.tomshardware.fr/capcom-supprime-le-drm-denuvo-de-devil-may-cry-5/`

Important : pour les exercices à partir de celui-ci, composer sur une autre copie

Exercice 6. Questions de cours

Q6.1 Pourquoi vaut-il mieux que le noyau évite de montrer les adresses de ses données ou fonctions à l'espace utilisateur ?

Q6.2 Pourquoi est-ce une mauvaise idée de brancher sur son ordinateur une clé USB trouvée dans la rue ? Expliquer 3 types d'attaques possibles (on ne parlera pas d'attaque physique du genre électrique, seulement des attaques logicielles).

Q6.3 Expliquer le principe général d'un rootkit.

Exercice 7. SMEP/SMAP

Q7.1 Rappelez le rôle des macros `get/put_user`.

Intel a introduit il y a une dizaine d'année un drapeau SMAP ("Supervisor Mode Access Prevention"). LWN [3] le décrit ainsi :

« This extension defines a new SMAP bit in the CR4 control register ; when that bit is set, any attempt to access user-space memory while running in a privileged mode will lead to a page fault. »

Q7.2 Quel est l'intérêt d'un tel drapeau ?

Intel a également introduit un drapeau AC dans le registre EFLAGS. La protection SMAP n'est active que lorsque ce drapeau est activé. Deux nouvelles instructions, `STAC` et `CLAC`, permettent de très efficacement activer et désactiver ce drapeau.

Q7.3 Pourquoi, lorsque SMAP est activé dans CR4, ces instructions doivent être utilisées au sein des macros `get/put_user` ?

On pourrait penser que pour le cas où l'on a plusieurs appels à `get/put_user` dans une boucle, il vaudrait mieux factoriser l'utilisation de `STAC` et `CLAC`, pour ne les utiliser qu'avant et après la boucle plutôt qu'à chaque itération.

Q7.4 Pourquoi est-ce une mauvaise idée, à quoi cela vous fait-il penser ?

Netlink est une famille de protocoles de socket qui est utilisée pour discuter de manière efficace entre différents services du noyau et l'espace utilisateur, par exemple pour charger dans le noyau des règles de firewall. Le principe est de créer une socket de type `PF_NETLINK`, puis d'utiliser `sendmsg` dessus pour envoyer au noyau une requête contenant le détail de ce que l'espace utilisateur souhaite faire.

Différents services noyaux vont donc enregistrer des fonctions pour réceptionner ces messages.

Q7.5 À quel appel système vu en cours cela vous fait-il penser, pourquoi ce genre d'appel est en général dangereux ?

Notamment le service `sock_diag` enregistre la fonction suivante :

3. `https://lwn.net/Articles/517475/`

```
int __sock_diag_rcv_msg(struct sk_buff *skb, struct nlmsghdr *nlh)
{
        int err;
        struct sock_diag_req *req = nlmsg_data(nlh);
        const struct sock_diag_handler *hndl;

        if (nlmsg_len(nlh) < sizeof(*req))
                return -EINVAL;

        hndl = sock_diag_lock_handler(req->sdiag_family);
        if (hndl == NULL)
                err = -ENOENT;
        else
                err = hndl->dump(skb, nlh);
        sock_diag_unlock_handler(hndl);

        return err;
}
```

Ici `req` est une copie du message qui a été envoyé par l'espace utilisateur, que le service peut interpréter comme il le souhaite.

La fonction `sock_diag_lock_handler` s'occupe de charger automatiquement un module pour traiter la demande :

```
struct sock_diag_handler *sock_diag_lock_handler(int family)
{
        if (sock_diag_handlers[family] == NULL)
                request_module("net-pf-%d-proto-%d-type-%d", PF_NETLINK,
                               NETLINK_SOCK_DIAG, family);

        mutex_lock(&sock_diag_table_mutex);
        return sock_diag_handlers[family];
}
```

Q7.6 Pourquoi est-ce risqué de charger automatiquement un module ainsi ?

Q7.7 Quel est la faille béante visible dans le code de cette fonction `sock_diag_lock_handler` ?

Intel a aussi introduit un drapeau SMEP (Supervisor Mode Execution Protection) qui, lorsqu'il est activé dans le registre CR4, empêche le noyau d'exécuter du code utilisateur.

Q7.8 Pour l'instant on suppose que SMAP n'est pas activé. Pourquoi ce drapeau SMEP peut ici éviter que la faille soit exploitée vraiment très facilement ?

Q7.9 Avec SMEP activé, que peut-on faire tout de même grâce à la faille ? (bien préciser les contraintes de ce que l'on peut faire)

Q7.10 Pourquoi SMAP rend l'exploitation plus difficile ?

Q7.11 Avec SMAP activé mais SMEP non activé, que peut-on faire tout de même plutôt facilement grâce à la faille ?

Q7.12 Avec à la fois SMAP et SMEP activés, que peut-on faire tout de même grâce à la faille ? (bien préciser les contraintes de ce que l'on peut faire)

`iret`