CS457 - Introduction to Information Security Systems

Tutorial for Assignment 2
Assignment 2

- In this assignment, you have to solve 2 main tasks
  - Implementation of an access control system
  - Implementation of a password sharing algorithm
Access Control

Access control aims for the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.

Exercise 1
You are required to use the following techniques to build a basic access control system that prohibits an executable from invoking the “fork” system call (see linux man page for `fork(2)`):

- Technique 1: Implement a custom library and load it using `LD_PRELOAD`
- Technique 2: Use the `ptrace` system call
**LD_PRELOAD**

LD_PRELOAD instructs the linker to load the library that contains your own “fork” implementation, before any other library.

Example of LD_PRELOAD that replaces `rand()`, the pseudo-random number generator, with a custom `rand()`.

```c
/* run_rand.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int i = 0;
    srand(time(NULL));
    for (; i < 3; i++)
        printf("%d\n", rand() % 1000);
    return 0;
}
```

```bash
epapado@adam $ gcc run_rand.c -o run_rand
epapado@adam $ gcc -shared -fPIC my_rand.c -o my_rand.so
```

```bash
epapado@adam $ ./run_rand
542
699
159
```

```bash
epapado@adam $ LD_PRELOAD=$PWD/my_rand.so ./run_rand
666
666
666
```
The ptrace system call

- The ptrace() system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers.
- It is primarily used to implement breakpoint debugging and system call tracing.
The ptrace system call

Example of ptrace use that counts the number of system calls that are invoked by the “date” command.

```c
/* ptrace.c */
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/ptrace.h>

int main()
{
    pid_t pid;
    int count = 0;
    int wait_status;
    int enter_syscall = 1;
    pid = fork();
    if (pid == 0) {
        ptrace(PTRACE_TRACEME, NULL, NULL, NULL);
        execl("/bin/date", "date", NULL);
    } else if (pid > 0) {
        wait(&wait_status);
        while (1) {
            if (enter_syscall == 1) {
                count++;
                enter_syscall = 0;
            } else
                enter_syscall = 1;
            ptrace(PTRACE_SYSCALL, pid, NULL, NULL);
            wait(&wait_status);
            if (WIFEXITED(wait_status))
                break;
        }
    } else
        printf("fork error\n");
    printf("Total system calls executed: %d\n", count);
    return 0;
}
```

epapado@adam $ gcc ptrace.c -o ptrace
epapado@adam $ ./ptrace
Sun Apr 12 21:06:50 EEST 2020
Total system calls executed: 48
Access Control

Exercise 2

1. You are required to use the ptrace system call to build an access control system that prohibits an executable from invoking a list of system calls. This is an extension of Exercise 1b.

2. Write a blacklist with the prohibited system calls.

3. [Hint] On your working machine, locate the file that contains system call information (contents should look like the picture).

4. Generalize your Exercise 1b solution and compute the required statistics.

```c
#define __NR_read 0
#define __NR_write 1
#define __NR_open 2
#define __NR_close 3
#define __NR_stat 4
```
Ten Friends with a Bank Account

- **Ten friends** decided to put their life savings in a bank account
- The account is anonymous and can be accessed **only with one password**
  - The password is an integer number
- They decided to let the money stay in the bank
- They will withdraw it on January 2nd 2050

- So, they decided to **split the password into 10 pieces**

- However:
  - One of them may not want the rest to have their share of the money
  - He **may simply not appear** on January 2nd 2050
Ten Friends with a Bank Account

- They decided that **at least nine of them** should be present to withdraw the money.

- To solve the problem:
  - They developed an 8th degree polynomial: \( f(x) = a_8 \cdot x^8 + a_7 \cdot x^7 + \ldots + a_1 \cdot x + a_0 \)
    - Where **\( a_0 \text{ is the password} \)** and \( a_i \) are numbers chosen randomly.
  - They decided to give **one point of the polynomial to each friend**
    - The first friend would take \( f(1) \), the second \( f(2) \), the third \( f(3) \), and the last one \( f(10) \)
  - **9 of the 10 points will be used** at January 2nd 2050 to **reconstruct the password**
Ten Friends with a Bank Account

- They decided that **at least nine of them** should be present to withdraw the money.

- To solve the problem:
  - They developed an 8th degree polynomial: \( f(x) = a_8 \cdot x^8 + a_7 \cdot x^7 + \ldots + a_1 \cdot x + a_0 \)
    - Where \( a_0 \) is the password and \( a_i \) are numbers chosen randomly.
  - They decided to give **one point of the polynomial to each friend**
    - The first friend would take \( f(1) \), the second \( f(2) \), the third \( f(3) \), and the last one \( f(10) \)
  - 9 of the 10 points will be used at January 2nd 2050 to **reconstruct the password**

- The main idea is based on that it takes \( k \) points to define a polynomial of degree \( k-1 \)
- Given any subset of \( k \) of these pairs, we can find the coefficients of the polynomial using interpolation\(^1\)

---

1. **Interpolation** is the process of estimating unknown values that fall between known values
Exercise 3

At this exercise, you will have to implement this password sharing solution

- You will implement a “split” function that creates 10 polynomial points
  - Given an integer password
- You will implement a “join” function that recreates the password
  - Given 9 of these 10 polynomial points
Exercise 3

- You are going to create a polynomial with a degree of 8
- You will need to generate random coefficients using /dev/random
- You are going to need some basic knowledge of linear algebra
  - Solving a system of linear equations
  - **Gaussian elimination**\(^1\) will do the job
    - Transformation to an upper triangular matrix (reduced row echelon form)
    - Using elementary row operations

---

The initial set of linear equations is:

\[ \begin{align*}
X + Y + Z &= 5 \\
2X + 3Y + 5Z &= 8 \\
4X + 0Y + 5Z &= 2
\end{align*} \]

The solution is:

\[ X = 3, \; Y = 4, \; Z = -2 \]