2nd Laboratory

Deadlock

In an operating system a deadlock occurs when a process P1 or a thread T1 tries to take a resource R1 taken by another process P2 or another thread T2 which waits to take another resource R2 already taken by process P1 or thread T1.

P1/T1 code:	P2/T2 code:
Lock R2	Lock R1
P1/T1 is here-> LockR1	P1/T1 is here-> LockR2

Please open the archive laboratory2.zip and you will find in it the folder deadlock. From a command line console, please enter in it and run the command:

gcc -o deadlock deadlock.cpp

Run the compiled executable, does it hang? Is the deadlock perfect (meaning does it happen for sure all the time)?

Practic 1: Create a perfect deadlock by adding a barrier after each thread takes the first resource.

Practic 2: Lets go back to the initial deadlock. Compile the executable with the following command:

gcc -O0 -g -o deadlock deadlock.cpp

Run the command:

gdb ./deadlock

When the gdb starts run the command run.

Your programme most probably is already hanged in this moment. To stop it and analyze what is happening send to the running process a SIGINT signal by using CTRL+C in the console where gdb runs. In this moment the process is stopped, and you can analyze it.

Type this command:

info threads

how many threads does it display? Why?

Do the same steps for the modified deadlock from exercise Practic 1. Does info threads shows you now 3 threads? Why?

Type the commands:

thread 2

info stack

thread 3

info stack

After the commands from above you should have been able to identify where is hanged each thread. By analysing like that an executable you are able to identify after each resource is waiting each thread hanged and knowing this you should be able to fix the deadlock.

Library hijacking

Please enter in the folder loading_so. You will find the files library.c and main.c . Open them and understand the code. From library.c we will build a library using this command:

gcc -o library.so -shared -fPIC library.c

From main.c we will build the executable which will load the library:

gcc -o main main.c

Run the following command:

./main

After the process ends the result should be the display of the word library from library library.so

Practic 3: Today we will take a look at one of the vulnerabilities used pretty often by attackers – library hijacking. The goal is to change the library library.so with a library which will help the attacker to run his commands.

Please enter in the folder client-server. You will find 2 files server.c and client.c. Please do compile them:

gcc -o server server.c

gcc – o client client.c

Run the 2 executables from 2 different consoles and see how messages are exchanged between client and server.

Now replace the content of function MyFunction from library library.so with a code which will connect to a server and will receive commands from that server which will be run using function system:

http://man7.org/linux/man-pages/man3/system.3.html

Please observe how by doing this an attacker can obtain full access to the targeted machine. A real life scenario is when you are replacing files from an application with the cracked once to use an application.