OT Lab 1 - Assembly Patterns

> Individual assignment

In this lab, you will learn the basic concepts of binary file disassembly, memory protection mechanisms and program debugging techniques. Then you will learn various methods of compiling executable files, learn binary linking, learn how to recreate assembly code or use reverse engineering techniques to crack the simplest program.

Part 1 - Understanding Assembly

Task 1 - Preparation

1. Choose any debugger/disassembler what supports the given task architecture (32bit & 64bit). Some of options:
    - IDA
    - Ghidra
    - Hopper
    - Radare2
    - gdb
    - OllyDbg (for Windows OS)
    - Immunity Debugger (for Windows OS)
2. You have to check and analyze what does a binary do before running it. You have no idea what file we are working with, even if it is initially considered as legitimate application.
3. Prepare a Linux VM for this lab, as you might need to disable ASLR.

Task 2 - Theory

1. What is the difference between `mov` and `lea` commands?
2. What is `ASLR`, and why do we need it?
3. How can the debugger insert a breakpoint in the debugged binary/application?

Task 3 - Disassembly

1. Disable ASLR in your Linux VM using the following command:

    ```
    sudo sysctl -w kernel.randomize_va_space=0
    ```

    or

    ```
    echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
    ```

2. Load the binaries (`sample 32` **and** `sample 64` **from part1.zip**) into a disassembler/debugger.
3. Does the function prologue and epilogue differ in 32bit and 64bit? What about calling conventions?
4. Does function calls differ in 32bit and 64bit? What about argument passing?
5. What does the command `ldd` do? “ `ldd BINARY-NAME` ”.

Part 2 - Reverse engineering

Task 1 - Theory

1. What kind of file did you receive (which arch? 32bit or 64bit?)?
2. What do stripped binaries mean?
3. What are `GOT` and `PLT`? Describe on the practical example.
4. What are binary symbols in reverse engineering? How does it help?

Task 2 - Reverse & Cracking

Inside the ZIP file (**part2.zip**), you will have multiple binaries (`bin 1-6`), try to reverse them by recreating
them using any programming language of your choice (C is more preferred). You are given two simple PE files as well:

- task1.exe (it was found that this binary is dynamically linked to the debug version of Microsoft Visual C++ redistributable 2015. You can put this package together with Visual Studio installation, or add dll file that is inside the zip archive)
- task2.exe

Your task is to crack the program and find the correct password.

> *Tasks distribution:*
>
> - *if your **st** number is even: you have to work with* `bin 1-6`
> - if your **st** number is odd: you have to work with `bin 1-4` + `task1.exe` + `task2.exe`

Bonus

Describe the difference between PE and ELF executable files. Dhow it by a practical example.