SMRe - Reversing binary codes

Some indications for Lab session 2

Before you start:

We are going to use an obfuscation tool called Tigress.

You can:

- either install this tool on your own machine by downloading it here:
  http://tigress.cs.arizona.edu/download.html
- or install it yourself on your ensimag account (from the same web page)
- or copy this archive on your home directory and add theses two lines in your .bashrc file:

```
export TIGRESS_HOME=$HOME/tigress-unstable
export PATH=$PATH:$HOME/tigress-unstable
```

We are also going to use again IDaPro. The simplest way is to dowload the tool from this page:
https://www.hex-rays.com/products/ida/support/download_freeware.shtml
and install it either on your own machine, or in your ensimag account.

Finally, you will need these files:
http://www-verimag.imag.fr/~mounier/Enseignement/SMRe/Lab2.tar

Exercice 0 (benchmarks)

Write 2 or 3 small C programs, e.g.:

- something closed to the fact42 example (you can take this example itself)
- a small "guess passwd" example (reading a string from the keyboard and comparint it with a "secret" string)
- another example, calling 2 or 3 intermediate functions

Compile your programs and check (with IDA) that you understand more or less the disassembled code.

Exercice 1 (obfuscate yourself)

Try to obfuscate your benchmarks by yourself :
Apply successively (by hand, at the source level) some transformations we saw during the lectures on copies of your benchmark, e.g.:

1. inserting opaque predicates
2. replacing integer constants by complex (equivalent) expressions
3. splitting or merging variable

Check if your transformations are correct (i.e., compile the transformed code and execute it).
Open the binary codes with IDA to see how "good" are your transformation (i.e., how much do they obfuscate ?)
You can also play with the optimisation options (-O1 or -O2) to see if there are some differences.

Exercice 2 (use Tigress !)

The purpose of this session is now to observe/understand the effects of some code transformations provided by Tigress on yopur benchmarks.

The principle is the following:

1. obfuscate each example using Tigress (for the given transformation listed below)
2. look at the source code produced (to understand the transformation)
3. run the executable code a.out (to check if the result produced is correct !)
4. disassemble a.out using IDA Pro to appreciate the quality of the obfuscation (from a disassembler point of view).

Examples of code transformation

Look at http://tigress.cs.arizona.edu/transformPage/index.html to

- understand the principle of each transformation
- know how to produce it (some examples of shell script are provided in, you can re-use and extend them)

Here is a first list of transformations you can play with:
Virtualize, AddOpaque, EncodeData, EndodeLiterals, EncodeArithmetics, Flatten, Merge, Split, etc.