Assignment 2 — Fuzzing, Exponentiation and More

2018-09-04

This assignment consists of four parts. The submission is a single `.tar` or `.tgz` file.[1] Opening the file should create a single folder whose name is your a-number.[2] Inside the folder there is one PDF file[3] and 1–4 subfolders. The PDF file, `answers.pdf`, contains the answers to Parts 1–3 of this assignment. The subfolders, named `part-<n>` where `<n>` is a number between 1 and 4, contain additional information and solutions for Parts 1 to 4.

We will provide a test script that checks these and some other requirements. If your submission does not meet the requirements above we will not test it and your mark will be 0.

Part 1 (20%) — Getting Here.

Break the code that hides the URL of this assignment. This is a part of the assignment even if you reached here without breaking the code! Submit a description of how you broke the code. (Not more than one page of text.) If you used any software to break the code, submit it in the subfolder `part-1`. Such software can be in any programming langu
support your description.

Part 2 (20%) — Integers.

In this part, you are asked to analyse a few function and understand what they d
simple arithmetic and bitwise operations on their inputs. We do not ask you to d
but to explain what the outcome is. For example, consider the function:

```
int32_t example1(int32_t a) {
  return (a^0xFFFFFFFF)+1;
}
```

Saying that `example1` calculates the exclusive or of the input with the number `0xFFFFFFFF` and adds one is technically correct, but is not the expected answer. The correct answer is that the function computes the two's complement of the input. (Or any equivalent description.)

Similarly, for `example2` below, the expected answer is that the function returns bit `b` of `a`. Saying that it shifts the number 1 by `b` bits to the left and returns the result of anding that with `a` is not sufficient.

```
uint32_t example2(uint32_t a, uint32_t b) {
  return (1<<b)&a;
}
```

[1]The contents must match the extension. Files in zip, rar, cpio, ar, or other archive format are not accepted, even if their name ends with `.tar` or `.tgz`.
[2]Don't forget the 'a'. Also, make sure the number matches your student ID.
[3]It is not enough that the name ends with `.pdf`. The contents must be in PDF format. Submission of text, Microsoft Word, LibreOffice, or similar documents will be ignored.

For all examples, assume that signed numbers are represented using two's complement and that integer overflow wrap around. For example, MAX_INT32+1 results in MIN_INT32.

The functions to analyse are:

```
int32_t f1(int32_t a) {
  return a & -a;
}

int32_t f2(uint16_t a, uint16_t b) {
  return ((int32_t)a - (int32_t)b) >> 31;
}

int32_t f3(int32_t a) {
  return (a | -a)>>31;
}

uint32_t f4(uint32_t a, uint32_t b, int32_t c, int32_t d) {
  c ^= d;
  c = (c | -c) >> 31;
  return (a & ~c) | (b & c);
}
```

Part 3 (35%) — Fuzzing.

For this part your task is to fuzz t fuzz the bignum library or the develop your own tools. Two recomm ocs/LibFuzzer.html) and *american fuzzy lop*

The software to fuzz is included in this assignment description. To ex f assignment2.pdf bignums. This will create a folder called bignums that co gNum-0–BigNum-9, each of which contains a solution to Assignment 1. To determine which of thos least significant digits of your a number. Thus, student a gNum-0, BigNum-1, and BigNum-4.

In answers.pdf please submit a description of what you have done and what you have found. (Up to two pages of text.) Also, please submit any software or configuration files you created for this part of the assignment in the sub-folder part-3.

Part 4 (25%) — Modular Exponentiation.

Extend libbn to support modular exponentiation. The function signature is int bn_modexp(bn_t result, bn_t base, bn_t exp, bn_t modulus). It accepts three bignums, base, exp, and modulus, and computes $\texttt{base}^{\texttt{exp}}$ mod **modulus**.

You may assume that base and exp are non-negative, modulus is larger than 1, that result is not the same as any of the other arguments. You may use bn_div from the sample included in this assignment description. To extract use tar xf assignment2.pdf samples.

The function takes four arguments: quotient, remainder, numerator, and denominator. It divides numerator by denominator, storing the whole part of the division in quotient and the remainder in remainder. The function only works with a positive denominator and a non-negative numerator.

The source code of your solution should be in the sub-folder part-4. Typint make in this sub-folder should build libbn.a. it may also build calc, or other programs, libraries, etc., but these will not be tested. The test script verifies that the library builds and that it passes basic sanity tests. If the test script fails any of these tests, you will receive no marks for this part of the assignment.