Seguridad Informática Fall 2023
Homework Module 4

Marco Guarnieri
IMDEA Software Institute

Guidelines

This homework is due by Tuesday, January 9th, 2024 23:59:59 GMT+1.

Please send your solutions in PDF format to `marco.guarnieri@imdea.org`. Any solution received after the deadline will receive a failing grade.

No collaboration is permitted on this assignment. Any cheating (e.g., submitting another person's work as your own or permitting your work to be copied) will automatically result in a failing grade.

Problem 1: Control-flow Side Channels *(5 points)*

Consider the following programs `A` and `B`, where `secret` and `input` are Boolean arrays of the same size `n`. The array `secret` contains an unknown secret and the array `input` contains user-provided input. Note that program `A` just ignores `input`. Assume that in both programs, each line of code takes *one unit of time* (1 cycle) to execute. In both cases, the adversary knows the size `n` of the arrays and its goal is inferring as much information as possible about `secret`.

```
Program A:

for j=0 to n-1
  if secret[j]==1
    do something
return
```

```
Program B:

for j=0 to n-1
  if secret[j]==input[j]
    do something
return
```

Consider an adversary `C` that provides `input` and measures the corresponding *absolute execution time* of the programs (i.e., `C` can measure the number of cycles taken to execute the program from beginning to termination).

(a) *(1 point)* What does adversary `C` learn about `secret` in *one* execution of program `A`? What does `C` learn in *multiple* executions?

(b) *(1 point)* What does adversary `C` learn about `secret` in *one* execution of program `B`? What does `C` learn in *multiple* executions?

Consider now an adversary `D` that can only *compare* the execution times of each program on different inputs. That is, the adversary can only observe whether the execution time on one input is bigger, smaller, or equal, to that on another input, but `D` cannot measure the absolute execution time.

(c) *(1 point)* What does adversary `D` learn about `secret` in multiple executions of program `A`?

(d) *(1 point)* What does adversary `D` learn about `secret` in multiple executions of program `B`?

(e) *(1 point)* Describe in words the attack of adversary `D` against program `B`.

Problem 2: Cache-based Timing Channels *(3 points)*

Consider a 1MB 4-way set associative cache with cache line size of 32B that is shared between an adversary and a victim process.

(a) *(1 point)* How many cache sets does this cache have?

(b) *(1 point)* What does the adversary learn about the victim's computation by flushing an address x from the cache at time t and measuring the time it takes to reload x at time $t' > t$? (Assume that there is no virtual memory.)

(c) *(1 point)* Give an example victim program that leaks the value of a Boolean variable `y` via this channel.

Problem 3: Constant-time *(2 points)*

The recommended countermeasure against timing attacks on cryptographic algorithms is to follow the "constant-time" coding discipline.

(a) *(1 point)* Describe the syntactic requirements that define constant-time code.

(b) *(1 point)* Which of these requirements are violated by the programs of Exercise 1?

Problem 4: Non-interference *(6 points)*

Consider a simple loop-free programming language and an attacker that can observe the initial and final values of all and only the public variables. In the following, we denote public variables as `pb_x`, `pb_y`, `pb_z`, ... and private variables as `pr_x`, `pr_y`, `pr_z`, ... Note that all variables store only natural numbers. Note also that `%` is the remainder operator, `==` is the equality operator, and =! is the inequality operator.

1. *(2 points)* Consider the following program:

```
pr_x = pr_x +((pr_x+1)%3);
pb_y = ((pr_x %3) == 0);
```

   Does the program satisfy non-interference? If yes, describe why non-interference holds. If no, provide an example illustrating a non-interference violation.

2. *(2 points)* Consider the following program:

```
if (pb_x == pr_y * 3)
  pb_x = 15;
else
  if (pb_x == pr_y * 2)
    pb_x = 42;
  else
    skip;
```

   Does the program satisfy non-interference? If yes, describe why non-interference holds. If no, provide an example illustrating a non-interference violation.

3. *(2 points)* Consider the following program:

```
pb_x = pr_y;
if (pr_y % 2 == 0)
  pb_x = pb_x +1;
else
  pb_x = pb_x +2;
pb_z = pb_x % 2;
pb_x = 0;
```

Does the program satisfy non-interference? If yes, describe why non-interference holds. If no, provide an example illustrating a non-interference violation.

Problem 5: Branch prediction *(6 points)*

Consider the following program:

```
int foo (int data[]){
  int sum = 0;
  for (int i=0, i < 8, i++) {
    if (data[i] < 50)
      sum += data[i];
  }
  return sum;
}
```

Consider two processors P_1 and P_2 that are identical except for their branch prediction unit. In particular, P_1 is equipped with a 1-bit dynamic branch predictor whereas P_2 is equipped with a 2-bit dynamic branch predictor. The internal counter of both predictors is initially set to 0. For simplicity, ignore the effects of branch prediction on the condition of the `for`-loop and focus only on branch prediction on the `if`-statement. Answer the following questions:

1. *(2 points)* For `data = {100, 62, 93, 42, 95, 13, 80, 65}`, which processor does result in less mispredictions and is, therefore, faster in computing `foo(data)`? Explain your answer.

2. *(2 points)* For `data = {10, 15, 60, 18, 70, 100, 11, 80}`, which processor does result in less mispredictions and is, therefore, faster in computing `foo(data)`? Explain your answer.

3. *(2 points)* Is there any preprocessing operation that you can do on `data` that preserves the result of `foo` and speeds up the computation (on average) for both P_1 and P_2? If yes, explain the pre-processing step and describe how it affects the computation. If no, describe why this is not possible.

Problem 6: Speculative execution attacks *(3 points)*

1. *(1 point)* Give an example of a snippet of code vulnerable to Spectre V1 attacks and describe how the attack works.

2. *(1 point)* Give an example of a hardware-level mitigation against Spectre V1 attacks.

3. *(1 point)* Give an example of a software-level mitigation against Spectre V1 attacks.