Demonstration of timing Attacks

By Uzan Moshé in the third year of Computer Science at Jerusalem College of Technologie (JCT)
Instructor: Heanel Arie

ABSTRACT:

In this project we want demonstrate the timing attack with three practical example. For this we will use an Arduino Uno as target and TI board as attacker.

1. Find a password keep by the target.
2. Find a password keep by the target with random delay as security against timing attack.
3. Find the private RSA key kept by the target.

In general when we want to protect informations in a program then we use some security techniques, but without to pay attention on time features of our algorithm.
Timing attacks in theory are a beautiful way to transcend the system, caused by the temporal restrictions imposed by an algorithm. In this project we will demonstrate that this not just theory thank to three practical examples. In first cracking of a password who is keep by a target, secondly we retake same example that the first but with additional behavior against timing attacks and thirdly cracking of RSA private key who is keep by a target.
To obtain practical result we will use two embedded chips in the present case, an Arduino[1] uno as target with Arduino IDE as C++ programing platform , a TI board[2] as attacker with Energia IDE as C++ programming platform and a led for times measures.

[1] Arduino Wikipedia

[2] Like Arduino

FIRST ATTACK: SIMPLE TIMING ATTACK

Theory:

A simple timing attack consist of some measure of the time to compute a comparaison of users inputs and the real password and deduce the real password.

Let's start with the properties of a comparison function as strcmp(password, userInput). This function compare each character one by one "password[i] == userInput[i]" and when the function find password[i] != userInput[i] the function stop the comparaison. In this example the comparison between real password and the user Inputs will be made by the strcmp function.

Therefore, for this example we can already see a track to follow for a timing attack.

For the example, let's take the comparison of a character by the strcmp function as a unit of time and assume that the passwords keep by the target is 9315.

The following table shows the comparison of different inputs with the real password and the time taken for each one.

9	3	1	5	:	times
1	🟥 1	🟥 1	🟥 1		1
9	1	🟥 1	🟥 1		2
9	3	2	🟥 1		3
9	3	1	1		4
9	3	1	5		4

Characters on a red background represent the characters not to be compared. We can easily observe that the closer you get to the real password, the longer the comparison time increases.

Practice:

As we explain above we will use an Arduino uno as target with Arduino IDE as C++ programing platform , a TI board as attacker with Energia IDE as C++ programming platform and a led for times measures.

In order to design the attack it is necessary a target, thus in first place it is necessary to design a target with the ability to keep a password (p1) and to manage the input of a password (p2) by a user which will be compared to p1 by a strcmp function. The goal being the demonstration of the attack timing, we just turn on a led before the strcmp function and turn it off immediately after. For the demo a simple 4-digit password is required, so we will use a simple "char p1 [4]" to keep p1. For the management of the inputs we will use a keypad with 12 keys which will be connected to the Arduino and will allow the input of passwords. Arduino provides the Keypad.h library that allows to use a keypad.

The design of the target does not pose any particular problem, so we go to the design of the attacker. Making this demonstration on embedded the design of the attacker is more complex, since to be able to input a password on the target we must emulate a keypad respecting the physical aspects of a keypad, for instance when push of a button and an error is quickly committed.

The goal of the attacker is to find the four digits of the password using an timing attack. For this purpose we vary the four numbers one after the other with the 12 keys composing the keypad, measuring each time the lighting time of the led we talked about when designing the target, either 12 * 4 = 48 measurements. We store these measurements in four arrays, one per digit making up the password. Finally, we only get the maximum value of the first table for the first digit, the second for the second digit and so on until the fourth digit. And gets four digits matching the password of the target.

When designing the attacker, having as a times measure only integers in the present case, there may be problems of accuracy in the calculated times. One of the possible solutions is to make an average of several inputs of the same key, which leads us to the second attack.

SECOND ATTACK: TIMING ATTACK AGAINST SIMPLE RANDOM TIME-DELAY INSERTION

As prevention against timing attacks some person will add a random time when comparing, but we will see here that this will just slow down the attacker.

Theory:

The theoretical part of the first attack is similar to this one with some difference.

We will therefore start directly with the example: Let's take the comparison of a character by the strcmp function as a unit of time and assume that the passwords keep by the target is 9315. But the difference with the precedent instance is that when the comparison the target add a random delay from 0 to 5.

The following table shows one possible result of the different inputs comparison with the real password , the time taken for each one, the random delay and the time + random delay .

9	3	1	5	:	times	random delay	time + random delay
1	1	1	1		1	5	6
8	1	1	1		1	3	4
9	1	1	1		2	3	5

Characters on a red background represent the characters not to be compared. We can observe that, unlike the previous example, the times are distorted by the random delay. An interesting solution is to make an average of several inputs of the same key as explain at the end of the practice section of the first attack, but in the present case number of inputs of the same key which let compute average it must greater.

Practice:

The practical part of the first attack is similar to this one with some difference:

1. When the comparison in the target need to add a random delay from 0 to 5.

2. Make an average of several inputs of the same key as explain at theory section of this part, with 80 entries by same input of the same key.

THIRD ATTACK: BASIC TIMING ATTACK DEMONSTRATION ON RSA

In this attack I assume that the reader has prerequisite knowledge of RSA.

Theory:

A basic Timing Attack demonstration on RSA consist of some measure of the time to deduce the rsa private key.

The modular exponentiation[3] in RSA consists of a large number raised to a large exponent which is a time consuming operation. A simple and efficient algorithm for computing $C^d \bmod N$ is the square and multiply algorithm.

For example, an exponent value of 10 is 1010 in binary. Without leading zeros, d0 is always 1. Notice that 20 can be built up one bit at a time from the left to right as (0, 1, 10, 101, 1010) = (0, 1, 2, 5, 10). In other words, the exponent 10 can be constructed by a series of steps, where in each step, we double the number by shifting one bit to the left, and add 1 if the next binary bit is 1.

When decrypting a message the value of the exponent is d (private keys), so in theory if we can measure the time taken by the condition when decrypting a message, we can determine if the next bit is a 1 or 0. In the present case if it is a 1 it will require more time because of the condition and the addition operation.

Example based on the algorithm used in the practice section:

```
1.     long long int modpow(long long int m, long long int k, long long int n)
2.     {
3.       long long int result = 1;
4.       while (k > 0)
5.       {
6.         if ((k & 1) > 0) result = (result * m) % n;
7.         k >>= 1;
8.         m = (m * m) % n;
9.       }
10.    }
```

In this example take d = 10, or binary 1010, the measurement of time will do on line 6.

[3] http://www.cs.sjsu.edu/faculty/stamp/students/article.html

Let us put t1> t0:

d in binary	times
1	t1
0	t0
1	t1
0	t0

From this table we can observe that by measuring time, we could theoretically easily demonstrated that an timing attack works.

Practice:

In practice the target implement the modpow() function as view in theoretical section. This function allows the target to be decrypted a message with the private key d.

To do this we generate keys from the outside using, for example, the openssl library. In the present case since the target is an Arduino Uno we generate a 32 bit RSA (for the modulo). Since it is here a demonstration, then we allow ourselves to use means not necessarily conventional, so we position the lighting of the led before the 6th line of the modpow() function and turn it off afterwards, we also add a delay of 10 milli seconds before turning on the led, To enable the attacker to be able to measure in a simpler way.

The attacker measures the times during which the led is lit when decrypting a message. We get results that allow us to define d in binary. In the present case the result is reversed, it is therefore sufficient to convert the times measured in 1 and 0 and finally inverted the whole.

CONCLUSION:

I chose this project because I thought I was going to learn a lot as much in security as in embedded programming. What was my pleasure when I realized that I was right. Doing programming embedded in this kind of project, we sometimes put in difficult positions, not having the habit of error due to delays or other problems specific to embedded programming.

This project allowed me to realize that sometimes a slower implementation may be preferable (eg square and multiply). It also allowed me to realize that the pirates could use unconventional means to succeed in their exploitation.

I thank my mentor Arie Heanel for his presence throughout this project.

HOW USE MY PROJECT:

For use my project you must execute the target before the attacker.

There are 6 project folder with attacker and the corresponding targets:

1. Target_1 vs Simple_Timing_Attack
2. Target_2 vs Random_Timing_Attack
3. Target_3 vs RSA_Timing_Attack