COSC 6385 - Computer Architecture
Programming Assignment

Submitted by:
Md Rafiqul Islam Rabin
myUH ID: 1797648
Email: mrabin@central.uh.edu

Date: March 21, 2019

This assignment is about the **Figure 3: Timing of the branch code block in the spy process, reflecting the data sent by the trojan**, based on the reference paper: **Covert Channels Through Branch Predictors** published in HASP'15 by D Evtyushkin, D Ponomarev and N Abu-Ghazaleh.

How to Run

I switched to the Intel asm syntax instruction and run the code on the **VirtualBoxVM Linux 18.04.2 LTS**.

Step #1

First, run the **asm_branch.py** file with following command to generate the asm branch for trojan/spy program.

```
# python asm_branch.py
```

Here, 500000 asm branch (taken and not-taken) for trojan and 30000 asm branch (only taken) for spy will be saved into **asm_branch.h** file.

Step #2

Then, run the **trojan.c** and **spy.c** file with the following commands to compile the trojan/spy program.

```
# gcc -masm=intel -ggdb trojan.c -o t.out -lpthread
# gcc -masm=intel -ggdb spy.c -o s.out -lpthread
```

In this stage, two executable files **t.out** and **s.out** will be generated.

Step #3

Now, run the **t.out** and **s.out** executables with following commands.

```
# ./t.out "msg"
# ./s.out
```

First, run the **t.out** with message as arguments. If no message (or empty message) has been passed as argument, it will show an error message saying that no/empty message found. Then, for any valid message, the message will be sent as a form of executing asm taken branch or not-taken branch. Here, a `pthread` trojan will be invoked from the main thread. Then main thread is running on cpu core 0 while the trojan thread is running on cpu core 1. This trojan thread will periodically execute asm taken branch and not-taken branch or do nothing based on the branch status set from main thread. On the other hand, the main thread will control the branch execution of trojan thread. The main thread will translate each character of message as 8-bit binary. Then, the main thread will tell the trojan thread whether the taken or not-taken branch to be executed. If bit=1/bit=0, the main thread will change the branch status to BRANCH_TAKEN/BRANCH_NOT_TAKEN. Based on the implemented code, each bit will be sent after 0.5 sec and after sending each character there will be a 1 sec do nothing; it seems on average 5 sec will be spent to send a character.

Now run the **s.out**. In similar way, here a `pthread` spy will be invoked from main thread. And main thread is running on cpu core 0 while the spy thread is running on cpu core 1. The spy thread will execute only the asm taken branch after each 0.25 sec interval for 200 iterations with saving the cycle information from rdtsc(), both before and after executing the branch. On the other hand, the main thread will calculate the number of cycles required by spy for each iteration and will save those data into **cycles_times.log** file. Note that the spy is always executing the asm taken branch. So, the execution time will be smaller if it gets the state from the trojan ‘taken’ branch and the execution time will be increased if it gets the state from the trojan ‘not-taken’ branch. Here, the `sleep` interval in the spy is used to make sure the state from trojan, also the `mutex_lock` is used to ensure that the main thread is actually calculating the cycles information after spy finished execution of the asm taken branch.

Step #4

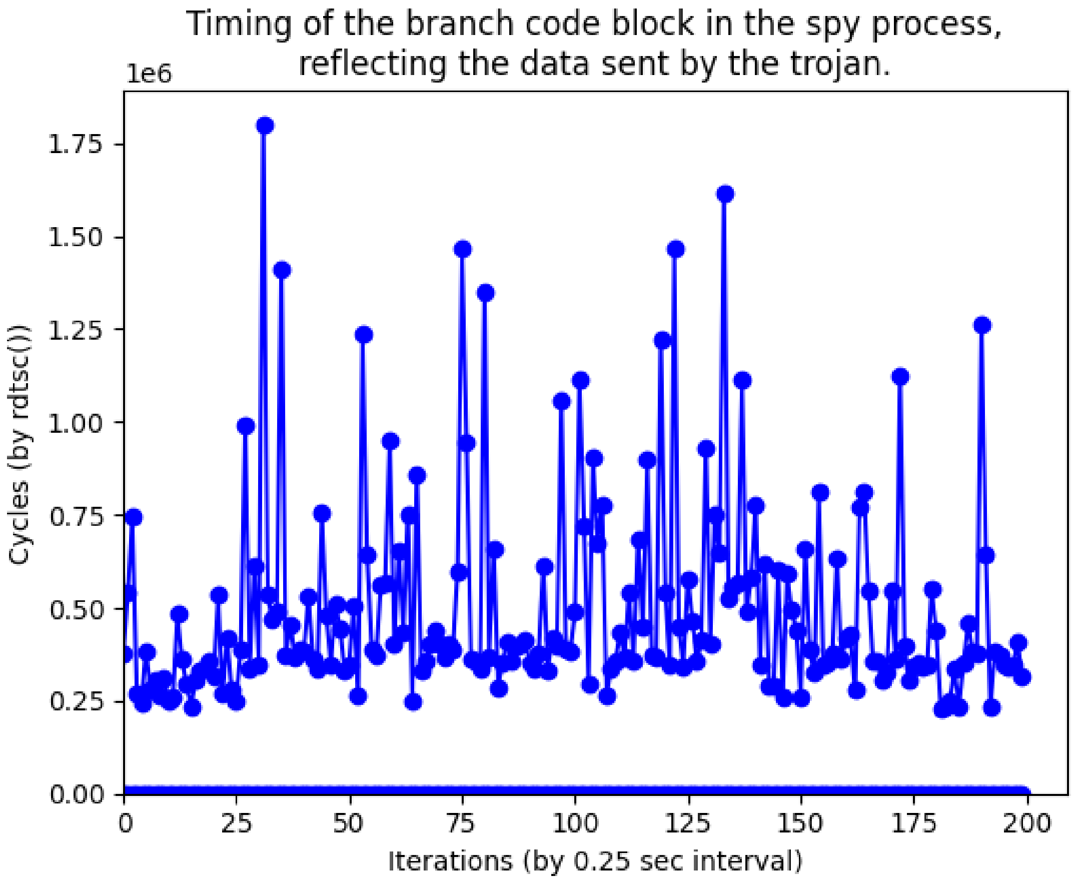
When executing the **s.out**, it will save the iteration times and cycles into **cycles_times.log** for MAX_PROBES. Finally, run the **draw_plot.py**, it will save the plot of cycles vs iteration times for latest run as **cycles_times.png**.

```
# python draw_plot.py
```

Note: I tested this program on the **VirtualBoxVM Linux 18.04.2 LTS**. I run the trojan with different messages and then check the timing of the branch code block in the spy process. The results almost reflect the data sent by the trojan, same as figure 3 of the reference paper. Although it has a little bit random effect for different runs, I believe this totally depends on the system under test and other running process.

Analysis of Result

Here, I will discuss my result with a sample message (i.e., username: '**mdrafiqulrabin**'). The data is also available in the log file of result folder. Note that, in our experiment, the trojan on average spends 5 sec to send a character and the spy calculates the execution cycle each 0.25 sec interval for MAX_PROBES=200 iterations.



Message: mdrafiqulrabin (length=14 chars)

Here, for message '**mdrafiqulrabin**', trojan will send the 112-bit (14 chars * 8 bits) signal in 70 sec (14 chars * 5 sec) and, in that time, spy will complete the MAX_PROBES=200 iterations. The figure shows the timing of the branch code block in the spy process, reflecting the data sent by the trojan, for full MAX_PROBES=200 iterations.