CES571S - Spectre - Final Report

467261 - Yifu Wang

2018 - 12 - 19

1 Introduction

Spectre is a vulnerability that affects modern microprocessors that perform branch prediction. On most processors, the speculative execution resulting from a branch misprediction may leave observable side effects that may reveal private data to attackers. For example, if the pattern of memory accesses performed by such speculative execution depends on private data, the resulting state of the data cache constitutes a side channel through which an attacker may be able to extract information about the private data using a timing attack.

Spectre attack can be conduct on browser victims. But this require attacker to send an attacker controled code to victims and execute it localy. Furthermore this attack can only retrieve the data from the process which runing the attacker controled code due to the seperation policy defined by mordern browser.

Howerver a derivation of spectre attack, netspectre, a generic remote Spectre variant 1 attack, could allow attackers to read arbitrary memory from the systems available on the network containing the required Spectre gadgets—a code that performs operations like reading through an array in a loop with bounds check on each iteration.

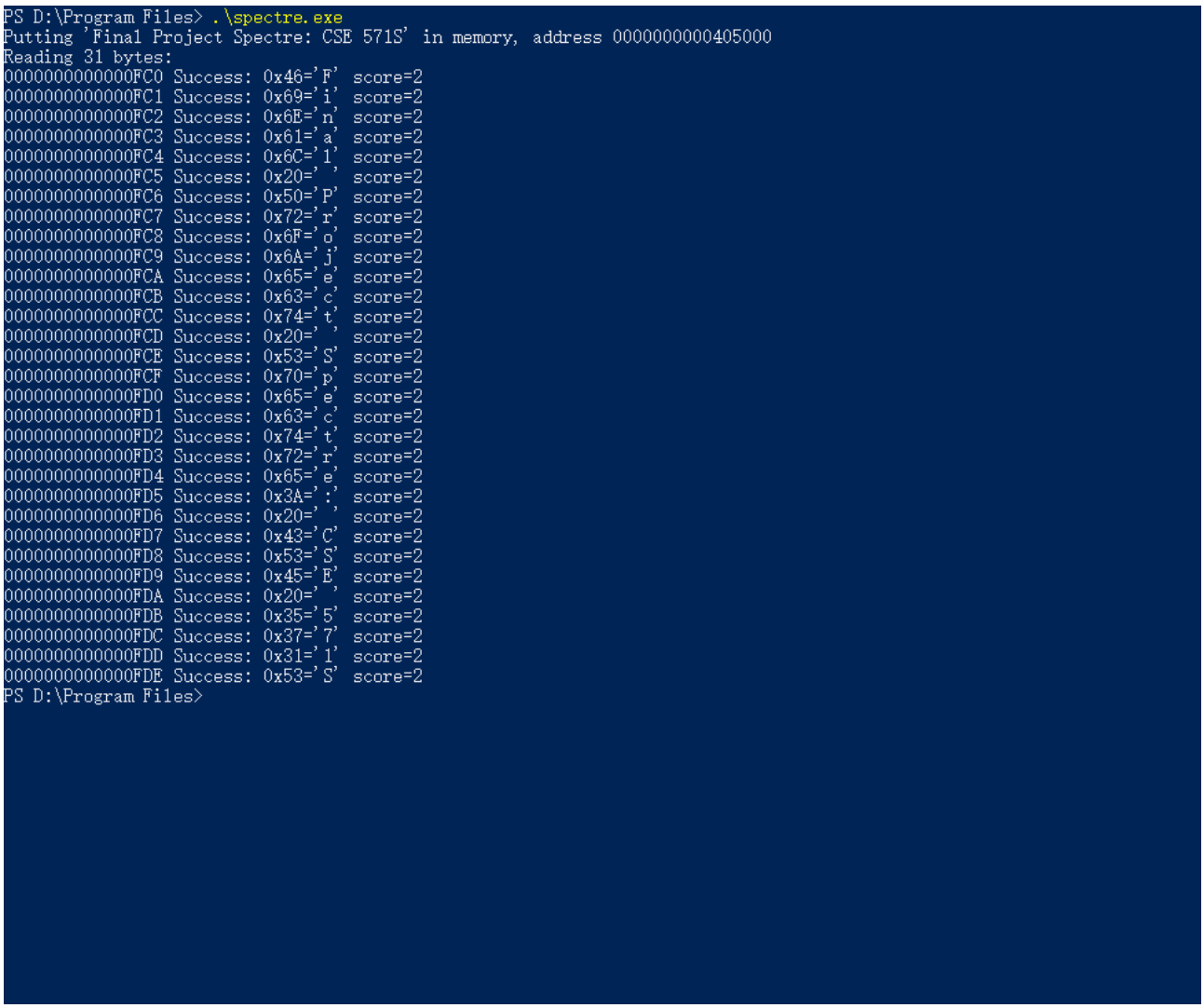
The original goal of this project is to setup a spectre attack cross processes, which now I knew impractical. So instead I implement a PoC of netspectre attack.

The project repository is held by this link.

2 Implement local attack

The local attack is fairly easy to implement. By using C/C++, you can directly execute assmembly language in your program, which allow to clear cache and measure system ticks manually, both of which are very importent for a successful spectre attack.

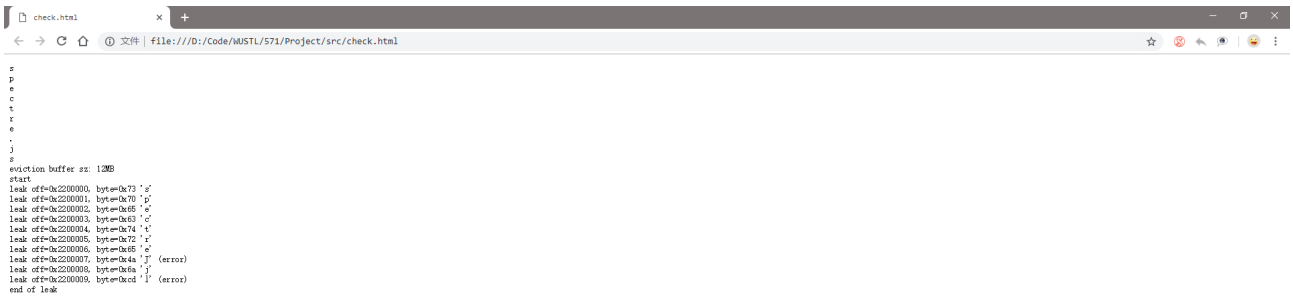
This program got auto-deleted all the time unless I turn off real time virus detection. But it still won't upload to github. So I compressed it. Basicly this program will retreive a build-in string array's value. Most of it was done by following the guide of SEED lab. But when runing on my machine. Some code will be optimized out sometimes. Those code is commened in source.



3 Implement browser attack

Since JavaScript couldn't manually control the caches. We have to evict cache before every iteration using some unrelavent junk data block. And since chrome has already patched this vulnerability by disabled SharedArray default. You need to enable it from setting and then you maight want to check if your browser is vulnerabel to spectre by this site.

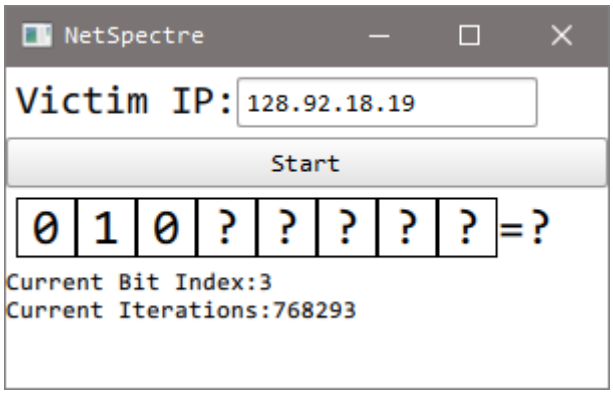
This site will retreive a build-in SharedArray data. But unlike local attack. Browser attack will sometimes be inaccurate due to the timer, and cache eviction. This is done by following the discreption in paper [1].

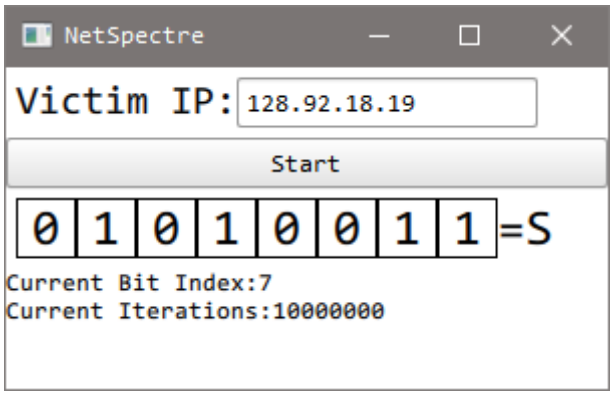


4 Implement netspectre attack

Basicly to complete a netspectre attack you need so call 'gadget' be pre-insert into user's machine. Which will listen to attacker's remote packet to mistraining the predictor and send back the bits value. Meanwhile the attacker side will measure the response timing and repeat this process as many as possible until the result is satisfide.

To get this attack work, firstly you should open netspectre.html and set up a single byte data. Then open NetSpectre.exe and set the victim ip, this program designed to be run in the attacker's machine, but victim and attacker should under the same subnet. By clicking the start button, the program should start to measure the response time for each bit. It's might take more than one hour to retreive one byte from netspectre.html.





5 Conclution

At present, spectre or netspectre is not a very serious problem that will endanger most common user. Since both of them need to pre-send some attacker code into user's machine. And even though netspectre attack claims to be able to retreive any data from a machine, but it's extremely slow. It's only reveal 15 bits per hour, namely 1 GB per 60822 years.

6 Reference

[1]:Spectre
[2]:NetSpectre