CSE 610 Special Topics:
System Security - Attack and Defense for Binaries

Instructor: Dr. Ziming Zhao

Location: Frnczk 408, North campus

Time: Monday, 5:20 PM - 8:10 PM

Announcements

1. **Final Exam**: 12/14 2020 7:15PM-10:15PM. Same format as the mid-term. There will be ? challenges labelled with the vulnerability type.

2. **Take-home exam**. It will have ? offline challenges and multiple choices questions. Due on 12/21.

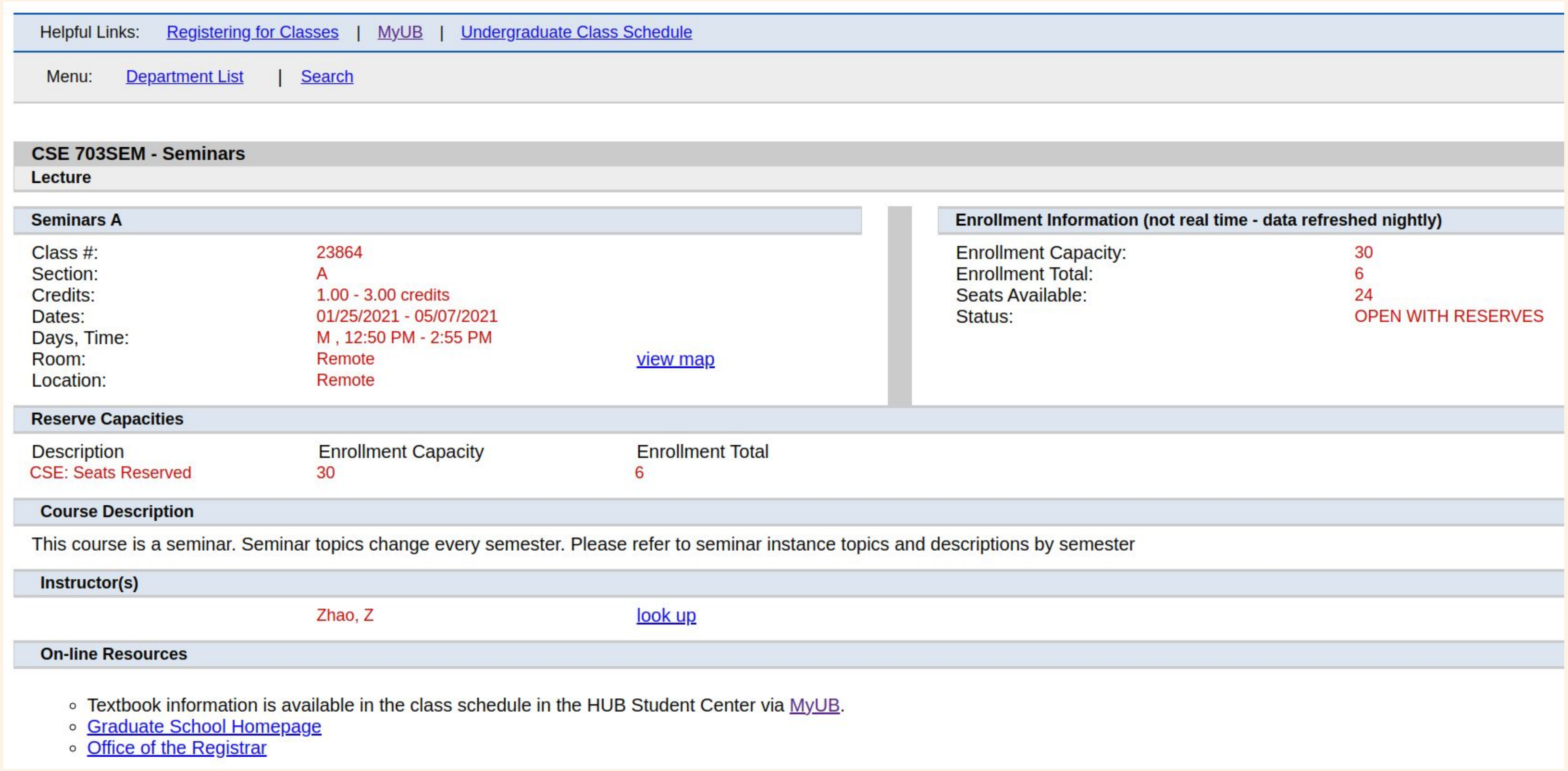
3. **HW-15.** Due on 12/21.

Guide to Prepare for the Final

1. Redo hw-12 where you develop a ROP shellcode to read from a file to print out to stdout. Get familiar with the steps to solve the homework and understand each gadget.

To incentivize you to evaluate the course, for the final evaluation if we get 100% response (all 13), each of you will get 45 bonus points. If we only get 12, no bonus points for anyone.

We are at 12/13.


Helpful Links: Registering for Classes | MyUB | Undergraduate Class Schedule

Menu: Department List | Search

CSE 703SEM - Seminars

Lecture

Seminars A

Class #:	23864	
Section:	A	
Credits:	1.00 - 3.00 credits	
Dates:	01/25/2021 - 05/07/2021	
Days, Time:	M , 12:50 PM - 2:55 PM	
Room:	Remote	view map
Location:	Remote	

Enrollment Information (not real time - data refreshed nightly)

Enrollment Capacity:	30
Enrollment Total:	6
Seats Available:	24
Status:	OPEN WITH RESERVES

Reserve Capacities

Description	Enrollment Capacity	Enrollment Total
CSE: Seats Reserved	30	6

Course Description

This course is a seminar. Seminar topics change every semester. Please refer to seminar instance topics and descriptions by semester

Instructor(s)

Zhao, Z look up

On-line Resources

- Textbook information is available in the class schedule in the HUB Student Center via MyUB.
- Graduate School Homepage
- Office of the Registrar

Today’s Agenda

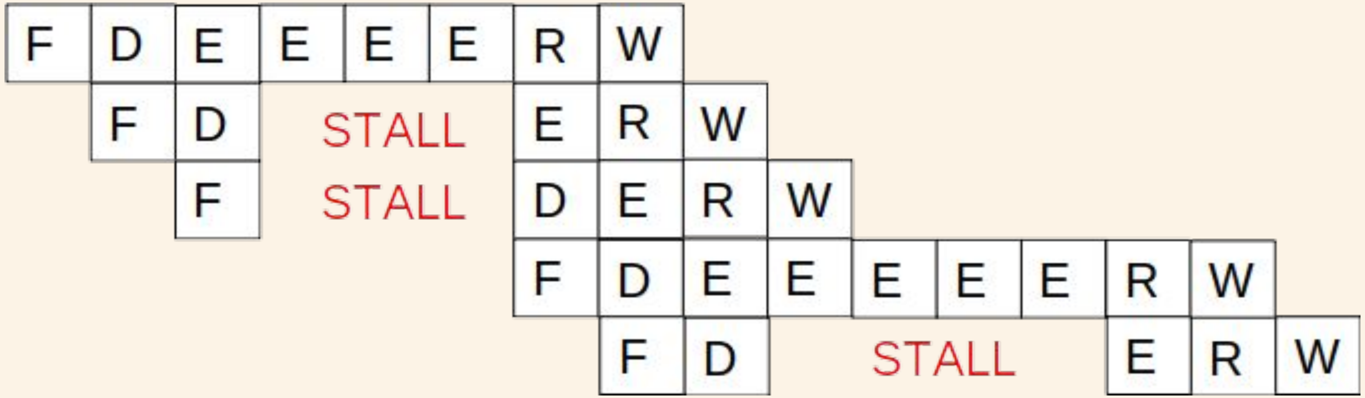
1. Spectre

Meltdown and Spectre

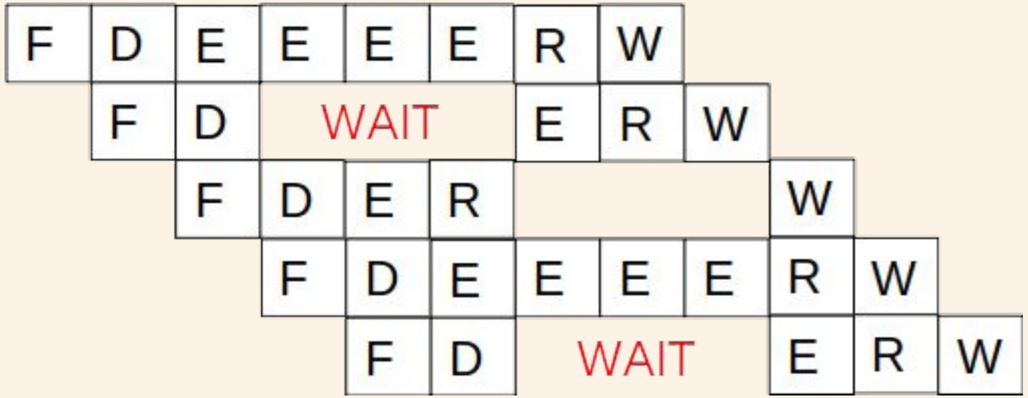
https://meltdownattack.com/

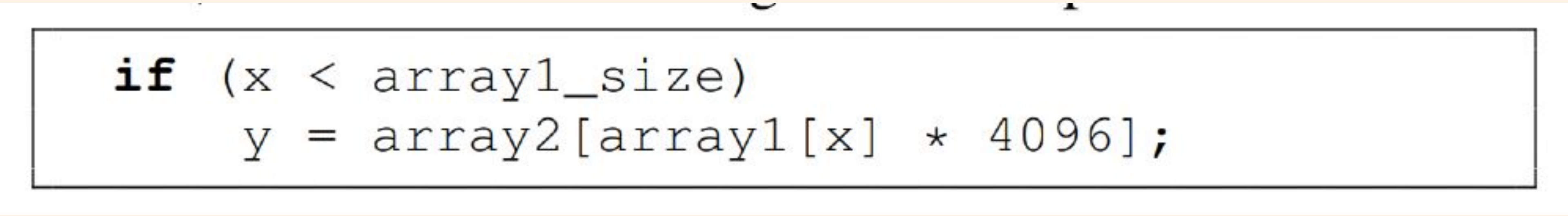
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5754

In-order vs. Out-of-order Dispatch



```
IMUL  R3 ← R1, R2
ADD   R3 ← R3, R1
ADD   R1 ← R6, R7
IMUL  R5 ← R6, R8
ADD   R7 ← R3, R5
```





```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Speculative Execution

The processor can preserve its current register state, make a prediction as to the path that the program will follow, and speculatively execute instructions along the path.

If the prediction turns out to be correct, the results of the speculative execution are committed (i.e., saved), yielding a performance advantage over idling during the wait.

Otherwise, when the processor determines that it followed the wrong path, it abandons the work it performed speculatively by reverting its register state and resuming along the correct path.

Speculative Execution

Speculative execution on modern CPUs can run several hundred instructions ahead.

Speculative execution is an optimization technique where a computer system performs some task that may not be needed. Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed.

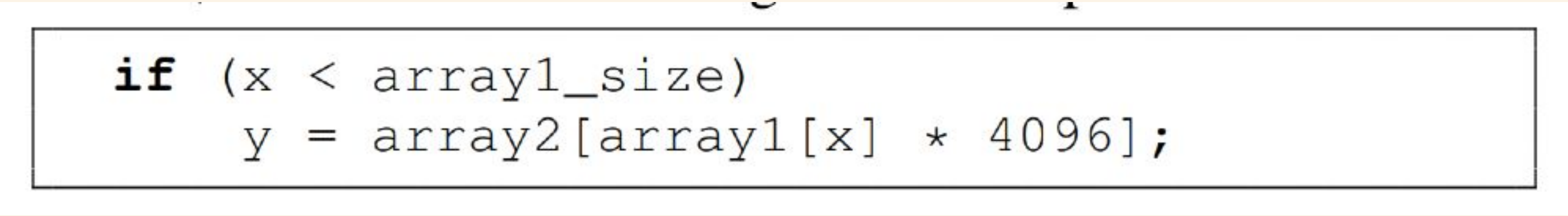
Branch Prediction

During speculative execution, the processor makes guesses as to the likely outcome of branch instructions.

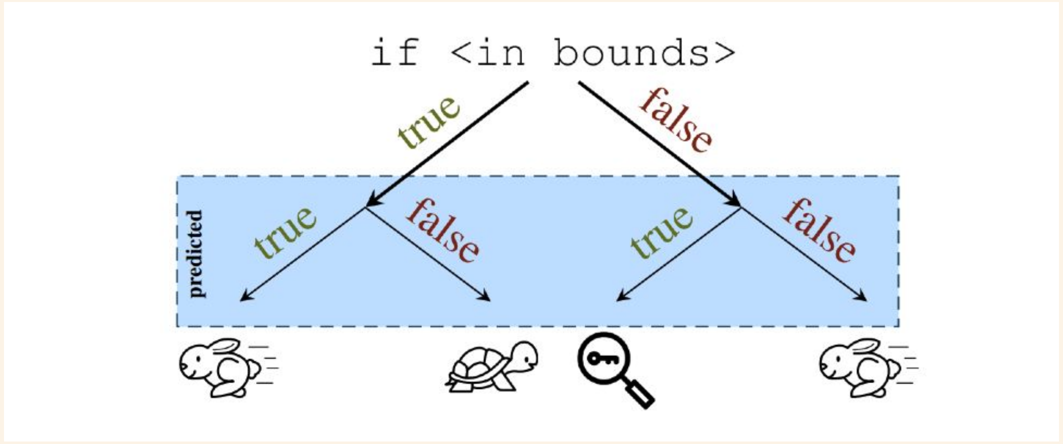
The branch predictors of modern Intel processors, e.g., Haswell Xeon processors, have multiple prediction mechanisms for direct and indirect branches.

Spectre V1

Conditional branch misprediction



```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```



Spectre V2

Indirect branches can be poisoned by an attacker and the resulting misprediction of indirect branches can be exploited to read arbitrary memory from another context.

Spectre vs. Meltdown

Meltdown does not use branch prediction. Instead, it relies on the observation that when an instruction causes a trap, following instructions are executed out-of-order before being terminated.

Second, Meltdown exploits a vulnerability specific to many Intel and some ARM processors which allows certain speculatively executed instructions to bypass memory protection.

Meltdown accesses kernel memory from user space. This access causes a trap, but before the trap is issued, the instructions that follow the access leak the contents of the accessed memory through a cache covert channel.