Problem 3

Spectre

Solution

Part A

Timing attacks work by exploiting the variable speed at which operations occur. Cache timing attacks (what Meltdown exploits) apply this to the cache. This can be exploited by running 2 steps:

1. Clearing CPU cache
2. Attempt to read desired memory address

This will result in an exception as the attacker won't have access to the desired memory address. However, as the regular program runs, it will read memory addresses which will then be placed in the cache. The attacker can then see which memory addresses have been read into the cache as the exception will be raised faster. This can be used by the attacker to determine where sensitive data may live.

Part B

Spectre takes advantage of speculative execution to cause the CPU to execute branches with code that should/may not have executed if the branch condition was evaluated fully before execution. Spectre takes advantage of this to then leak memory or register contents. An example of this is shown here (taken from the Spectre paper):

```
if (x < array1_size):
    y = array2[array1[x] * 256];
```

This can be exploited as the CPU can speculatively execute this branch even though x may be larger than then size of array1. Spectre can then use by indexing past the end of array1 and then leaking the information.

Part C

Disable indirect branch prediction in all processors.

1. Would protect against Spectre
2. Pros: fully protects against Spectre, not difficult for customers
3. Cons: huge performance hit to AWS, no option to opt out
4. Don't recommend

Give individual processes the ability to disable indirect branch prediction when they are running.

1. This would protect against Spectre if used correctly. This would be alright if the processes dealing with code that could access addresses had indirect branch prediction disabled.
2. Pros: Fine grain, safe is used well, performance friendly if used well
3. Cons: difficult for customers, can be used poorly both for security and performance
4. Recommend

Given individual customers a choice of whether to disable indirect branch prediction for their virtual machines.

1. This would protect against Spectre if a user opted in. I would recommend this approach as well as it gives the customers the option to be safe.
2. Pros: Not much work on customer (more likely to be adopted if it's easier)
3. Cons: Not every user is safe, customers who do opt in may take a large performance hit
4. Recommend

Do nothing.

1. Does not mitigate Spectre in any new way
2. Pros: easy
3. Cons: does not mitigate anything, customers may switch to different platform that does protect them, ignoring the problem
4. Don't recommend