Timing Attacks

- TENEX Operating System:
 - Logging in was a system call
 - o Checking the password was done through a loop through characters that terminated early
 - Timing can be used to brute force the password
 - Noise:
 - You could run the test multiple times
 - Or you could align the string to a page boundary so that the character after the unknown character is on a new page
 - You then run the system out of memory to unload all pages from memory
 - Page fault will then take longer, so we can measure that very clearly
- dm-crypt timing attack
 - Had an encryption process called AES that used a key
 - o Combined the key with the user data to index into a table (1 kb) to get the encrypted data
 - What we could do:
 - Create a huge buffer B of 0s to fill up the entire cache
 - Ask for B to be encrypted
 - The AES algorithm will then go through and access certain array indices with indices equal to the key, evicting parts of B
 - We then scan through B and can use timing to find out which accesses are slow, telling us what the key is
 - Fixes:
 - Intel implemented instructions that directly perform AES operations that avoid the cache

Meltdown

- Exploits speculative execution
- Adversary is running in user space and kernel wants to keep some secret key safe
- Adversary allocates a buffer B
 - Again, we will time how long it takes to access locations in this buffer
- We will execute instructions that will try to load the key and then use it to index B
 - Speculative execution causes our buffer to be accessed even though the load key wil cause a trap and fail
 - o Our code will abort, but the speculative execution will load into the cache
- We then scan through B to see based on timings which bytes were loaded, allowing us to read the key
- Key characteristic: bypass permission checks with speculative execution
 - o CPUs don't fully rollback all of the sideeffects of the speculative execution
 - Some solutions: move the permission check earlier, but this causes a performance hit, flush parts of the CPU cache after the permission fails

Spectre

- Exploits branch prediction
- Example code for PHT variant of spectre:

```
if (i < sz) {
  char x = array1[i];
  char y = array2[x];
}</pre>
```

• We could supply a large value of i, and the CPU could speculatively guess it passes (even though it doesn't)

•	Even though this code should never	array2[x]	into the cache and we can ther	1
	do the same idea to time the cache			