Summary of Spectre Attacks: Exploiting Speculative Execution

1. SUMMARY

Spectre attacks undermine inherit security assumptions made while writing softwares and give attacker an opportunity to read entire address space of the user via micro-architectural traces left by uncommitted "transient instructions" due to speculative execution.

- Address spaces of different processes are isolated to ensure safety and avoid access to address space of other process directly.
- While executing the program, branches may have data dependency to resolve the direction and data may be unavailable. In such cases processor goes for Speculative Execution to save on time instead of idling.
- If the predicted path (upon arrival of actual value) turns out to be correct, the changes are committed else discarded.
- But if prediction was wrong (even though CPU reverts to it's checkpoint state) it leaves many micro-architectural traces which can be exploited
- Relies on mistraining branch predictors, BTBs and force speculative execution using Flush+Reload or Evict +Reload and later probing the data or timing the access to particular monitored line.

2. ATTACK OVERVIEW

Setup:

- The adversary performs operations that mistrain the processor so that it will later make an exploitably erroneous speculative prediction.
- The attacker performs targeted memory reads that cause the processor to evict from its cache a value that is required to determine the destination of a branching instruction.
- The adversary also prepares the side channel that will be used for extracting the victims information, e.g. by performing the flush or evict portion of a flush+reload or evict+reload attack.

Speculative execution:

- The processor speculatively executes instruction(s) that transfer confidential information from the victim context into a microarchitectural side channel.
- This is triggered by having the attacker request that the victim to perform an action (e.g., via a syscall, socket, file, etc.). The attackers may leverage the speculative (mis-)execution of its own code in order to obtain sensitive information from the same process.

Probing:

- For the final phase, the sensitive data is recovered. For Spectre attacks using flush+reload or evict+reload, the recovery process consists of timing how long reads take from memory addresses in the cache lines being monitored.

3. POSITIVES

- Simple idea, yet powerful implications. Speculative execution in today's processors provides indispensable boost to performance but compromises security and privacy of user.
- Points out the need to shift to "Security first" approach instead of "Performance first"

4. NEGATIVES

- Patching up older machines may mean severe performance degradation and lower performance achievable on newer designs

5. IMPLICATIONS

- Opens up new research avenue in design of systems keeping security in mind.
- May provide pointers to discovering newer bugs or altogether new methods to fight existing bugs while searching for solution to Spectre.