A Formal Model of Cache Speculation Side-Channels

Catalin Marinas – TLA+ Community Event, October 2020

While side-channels have been formalised before, including confidentiality properties, most models do not cover speculative instruction execution and the information leakage is derived only from data explicitly accessed by the victim program. This presentation will introduce the work on formalising cache speculation side-channels using TLA⁺, with the model checker providing a counterexample to the confidentiality property: a form of Spectre-v1.

In this model, the abstract machine consists of a *privilege mode* (user/kernel), *registers, memory and cached state* (true/false) together with a set of actions modifying such state:

$$\text{Instructions} \triangleq Havoc \cup Move \cup Load \cup Store \cup Op \cup Exception$$

$$\begin{aligned}\text{Init} \triangleq\ & \wedge mode = \text{“}user\text{”} \\ & \wedge regs = [\text{r} \in REGS \mapsto \text{”}zero\text{”}] \\ & \wedge mem \in [ADDRS \to VALUES] \\ & \wedge cached = [a \in ADDRS \mapsto FALSE]\end{aligned}$$

$$\text{Next} \triangleq \exists instr \in Instructions : \text{Execute}(instr)$$

$$\text{Spec} \triangleq \text{Init} \wedge \Box[Next]_{\langle vars \rangle}$$

The confidentiality property is equivalent to an attacker execution trace being deterministic of only its initial state, its input (or the output of the victim program) and instructions executed under a given observation function (view of the system state). In other words, the attacker does not observe anything other than what the victim explicitly provides in its output. A timing side-channel is a difference in the execution time of the attacker or victim operations (for example a memory access). Abstractly, the observation function includes additional state that would cause such execution time difference (*cached* in this presentation):

$$\text{Obs}(\text{mem, cached}) \triangleq [addr \in ADDRS \mapsto \langle mem[addr], cached[addr] \rangle]$$

Let *ConfSpec* be the conjoined specification of two systems producing identical victim program output for the same input under the above observation function. The confidentiality property is then expressed as:

$$\text{THEOREM ConfSpec} \Rightarrow \Box(Obs(mem1, cached1) = \text{Obs}(mem2, cache2))$$

For certain sensitive algorithms, programmers have limited the information leakage through side-channels by carefully crafting the data access patterns so that that the attacker observations are identical. This presentation will show, however, that under speculative execution the confidentiality property no longer holds.