In order to better understand the broader landscape of TEE vulnerabilities and identify issues that are persistent through all models, I am starting this TEE vulnerability database. I certainly won't be able to complete it myself so please add to or correct it!

A starting point for characterisation (partly taken from this SoK):

- Remote adversary:

any adversary who is not involved in the manufacturing process and does not have physical access to the trusted hardware at any point in time. Think cloud tennants or cloud sysadmin.

- Fabric adversary

: "has the ability to introduce special hardware such as fabric interposers to launch man-in-the-middle attacks. The fabric adversary can also directly access data-at-rest such as the disk or external memory. However, this adversary cannot breach the SoC package: everything within the package remains out-of-scope"

- Invasive adversary

: "This adversary can launch invasive attacks such as de-layering the physical chip, manipulating clock signals and voltage rails to cause faults, etc., to extract secrets or force a different execution path than the intended one."

- Malicious Manufacturer Adversary

: This adversary probably needs to be broken down into several adversaries, but I include it for now to capture broadly the attacks that can be pulled off by any collection of entities involved in the manufacturing process.

To the best of our knowledge, no TEE defends against the latter two adversaries. They are listed for completeness. Hence, we are trying to categorise the attacks between physical adversaries and remote adversaries. Within remote adversaries we want to know if there exists a software based mitigation or if changes in hardware are required.

Failure mode

Attacker model

Mitigations

Models with vulnerability (can be mitigated)

Models with vulnerability (cannot be mitigated)

Models without vulnerability

Bus snooping

Confidentiality: Access pattern leaks

Physical

- Encrypt memory locations with specialised DRAM (e.g. invisimem, obfusmem)
- ORAM
- use "in package"/on-chip memory

All major (?)

Memory replay

Integrity

Physical

- store some info in on-chip meory (e.g. version counters or merkle root)
- use exclusively on-chip memory
- poison bits (software)

TDX (& SGX?)

Keystone

Interrupt: page fault (e.g. SGX step)

Access pattern

Software

- do not allow unexpected page faults (e.g. abort on fault)

SGX, TDX

Interrupt: timing-based (e.g. SGX step)

Cache side channel