TCB Safety

Elana Copperman, PhD – Mobileye / Intel
Rafi Davidovich – Intel

Linux Plumbers' Conference 2020
Kernel Dependability and Assurance Micro-conference

Agenda

- TCB and safety, use case for generalized application
  - Focus on software handling of memory errors
- Classification of TCB data
- Define some (not necessarily all) failure modes
- Define some protection mechanisms
- *Next steps …*

TCB – Thread Control Block

- Data structure in the Linux kernel which contains thread-specific information needed to manage threads.
- Manipulated by the kernel constantly, while the thread is being executed and while it is switched out of execution.
- Assuring the integrity of the TCB is critical to achieve safe thread life cycle management in Linux.

TCB – Data Classification

Fields	Safety critical	Examples
Scheduling information	Yes	counter nice *mm, *active_mm cpus_runnable, cpus_allowed sleep_time
Task state	Yes	state flags addr_limit exit_code, exit_signal Pointers to parent and siblings rt_priority
Process credentials	No	groups array kernel capabilities: cap_effective, cap_inheritable, cap_permitted pointer to user_struct
General	No	Limits File system info, open files information CPU specific sate of this task Thread group tracking
Signal handlers	Yes	pointer to signal_struct blocked sigset sigpending struct

Possible Failure modes

FM	Description	Effect
Bit flip	Corruption of TCB due to memory bit flip	Effect may vary, per corrupted field: • Crash of the thread (e.g. mm_struct corruption) • Wrong scheduling of the task (e.g. modification of scheduling priorities) • Wrong calculation of output (e.g. corruption of memory address space attributes) • Thread unable to respond to signal (e.g. signal set corruption)
Wild pointer	Corruption of TCB due to generation of erroneous write address by the kernel. TCB data may be corrupted or modified.	
Failure to update TCB	Failure to update a field (e.g. due to race condition)	Effect may vary, per required update. If failure is graceful, kernel will take the appropriate flow. If not, un-defined behavior may occur.

Mitigation strategies

- SW based mechanisms

- **Detection**:
  - Detect faults after they happen
  - Take appropriate action, based on policy

- **Prevention**:
  - Prevent the fault from happening

Possible Mitigations - Prevention

Mechanism	Description
Kernel configurations	Deploy existing kernel configurations to protect TCB: • For example: CONFIG_HARDENED_USERCOPY or disable CONFIG_DEVKMEM Define a customized configuration for protection of TCB data
memprotect(), on TCB pages	Use memprotect() to guard pages where TCB is stored
Make some data items R/O after init	Identify non-updateable fields (or, enforce some fields to be non-updateable) and make them read only
Grant write access only via hypervisor	Allow write only via hypervisor interface. Will block “wild pointer” failure mode

Possible Mitigations - Detection

Mechanism	Description
CRC	CRC/Checksum on safety critical data

Call For Action

- Complete TCB data classification
- Identify existing kernel configurations which can be used
- Define new mechanisms for prevention / detection
- Generalize safety mechanisms for protection of safety critical data, apply to additional use cases.