OSADL: Safety Critical Linux Working Group

Project # 2: LOPA tooling and infrastructure development

Rationale:

Currently only manual methods of proof of concept are available. However, first results look principally usable and may provide significant architectural flexibility as well as some level of modularity (notably for hardening). The LOPA technologies currently studied also provide significant security related capabilities that will be relevant to all practical systems and are of relevance in the context of IEC 61508 Ed 2 (see part 1 7.4.2.3). Security is though explicitly not considered in this proposal for LOPA (see possible extensions).

The formal basis for LOPA at the system level can be found in 61508-1 7.6.2.7 as well as 61508-2 Ed 2 7.4.3.1-4, respectively by reference in 61508-3 Ed 2 7.4.2.11 that addressed the coverage of systematic capabilities by elements that them selves provide a lower SC than the target safety function. Note that this differs significant from ISO 26262 Ed 1 ASIL decomposition. Software LOPA it self is not a concept established in the context of IEC 61508 Ed 2 directly but is utilized in the process industry sector standard IEC 61511 Ed 1 Clause 9 (main reference) which is intended as guidance for this LOPA interpretation though with the significant difference that LOPA in 61511 is referring to low-complexity elements while we are intending to apply the concept to high-complexity elements with the intent to cover residual analytical uncertainty as well as incompleteness issues.

Overview of development:

- *Phase 1:*	Define role of LOPA in overall software safety plan: • Selection • Verification • Qualitative/semi-quantitative/quantitative handling • Integration into Annex-A,B,C -> Annex-QR
- Phase 2:	61508-7 Ed 2 extension for SW LOPA (proposal) • Formal mapping into 61508-1,2,3,7 • Formal definition of LOPA to cover 61508-2 Ed 2 7.4.3
- *Phase 3:*	Process and tools requirements - mapping to standard (based on review of current results) • Process specification: work-flow and artifacts (61508-2 7.4.3 as well as 61508-3 7.4.2) • Tools specification (T1/T2) (61508-3 7.4.4 and 7.9) • Initial presentation/discussion with CA
- *Phase 4:*	General Investigation • Call-Tree vs Ftrace/GCOV interfaces and pre-processing

	• NCC analysis (how it worked up to 4.0 and why it stopped working) • GCC analysis (what is reusable what must be implemented) • Learn FTrace usage • Data formatting and structuring analysis
- *Phase 5:*	GCC Investigation • Specify initial test-cases (POSIX PSE 51 subset) • FTrace (manual) results for initial test-cases • Call tree generation with gcc experiments and prototyping • Verification of call trees, mitigation of ftrace to call tree differences (e.g. due to auto-inline)
- *Phase 6:*	Caller/Callee relation • Automated function classification (e.g. based on Coccinelle) • Kernel layering analysis (fuzzy-set) • Data formatting and structuring (analysis and definition) • Verification procedures (statistic e.g. SRS based)
- *Phase 7:*	Implementation of complete prototype tool set (it most likely will not be a single tool but a layered tool set). • Discussion with GCC/FTrace developers • Initial test on available test-cases (e.g. LTP subset) • Integration of function classifier and layering • Subset extraction (e.g. via minimize/PIT TBD) • Manual verification of results (30 samples) • Public verification by submission to appropriate kernel discussion lists.
- *Phase 8:*	Specific testing - e.g. x86_64 arch4 • Artificial (simple) Use-Case (e.g. PLC in a container) • Container configuration (cg/ns, seccom, cpu shielding, fs) • Initial element analysis (non-exhaustive) -> potentials • Manual analysis of layer of protection capabilities (qualitative) based on documentation and bug reports security documentation etc.
- *Phase 9:*	Application to prototype (not yet a full use-case - but basically something like arch4 - 3 containers of different isolation profiles) • Effort recording • Result recording • V&V efforts and recording of generated artifacts • Presentation/discussion to CA
- *Phase 10:*	Setup public accessible server and provide continuous data for a small set of configuration (e.g. arch4 on x86_64)

	• Feedback and incident reporting facilities • Development and user mailing lists
- Phase 11:	Formal acceptance for use in safety related systems • Formalized Acceptance and Test Criteria (ATC) • Assessment Report (AR) • Presentation at TueV Rheinland • Formal submission for acceptance as a basic safety mechanism in complex safety related systems. Formal proposal of LOPA for SIL decomposition (61508-3 Ed 2 7.4.2.11, 61508-2 Ed 2 7.4.3)
- Phase 12:	Dissemination and community integration (time-line of this phase would need to be beyond project end depending on how relevant tool data is found to be to kernel developers) • OSADL/LF communication channels to industry • FLOSS/Linux community events and community channels (LWN) • Safety community conferences
- Possible extensions	• Analytical and technical extension to address system security issues.
Participation form:	
	• Project development and data repository • Project mailing lists (development, review) • Dissemination workshops (may be in context of other events)
Responsibilities:	
	• Project legal entity: OSADL • Project resource maintenance OSADL Safety Critical Linux WG • Project Management: OSADL • Technical lead: Andreas Platschek ARTech (proposed), Nicholas Mc Guire • Implementation: Markus Kreidl, Nicholas Mc Guire, TBD • Certification Body: TueV Rheinland (proposed)
Effort estimation:	
	TODO
Timeline:	
Start	March 1 2019
Milestone 1	Formal integration into 61508 Ed 2
Milestone 2	Requirements and Design workshop
Milestone 3	Prototype presentation and discussion

Milestone 4	Initial release version (public/community)
Milestone 5	CA Report and all related artifacts published
Maintenance phase	
Project format:	
	Partner funded project (OSADL members)