

Tools WG 2025 Update

Matt Kelly, The Boeing Company



ELISA
Enabling **Linux** in
Safety Applications

Aerospace · Automotive · Linux Features

Medical Devices · OS Engineering Process

Safety Architecture · Space Grade Linux · Systems · Tools

Tools WG

- WG Mission Statement

“The Tool Investigation and Code Improvement WG focuses on creation, analysis, and application of tools and techniques to contribute to the improvement of the kernel for use in safety cases.”

- Focus on tooling around the kernel, ELISA CI setups, and addressing static analysis issues against the kernel

2024 Recap

- 22 unique attendees across 17 meetings
- Broadened the WG's mission statement
- Pulled 4 different tool efforts under this umbrella
 - Ks-nav, BASIL, DeltaKernel, llvm-cov
- First cross-company contribution (ks-nav)
- First known outside users of a WG tool (BASIL)
- Engaged with several outside groups on possible collaboration
 - Cregit, stress-ng
- Still welcome discussion on kernel static analysis results and patching
 - 1 upstream kernel contribution on this last year

ks-nav

What is it?

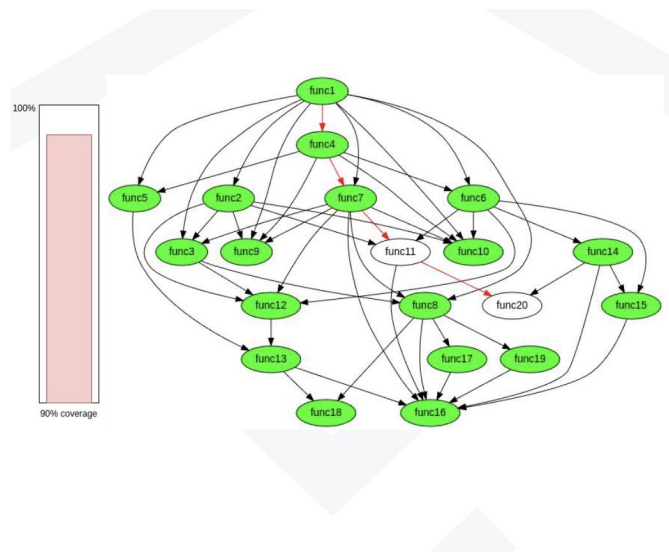
- Kernel binary analysis tool to assist with IS26262 FFI analysis
- Get it: <https://github.com/elisa-tech/ks-nav>

2024 Updates

- Improvements to the web interface
- Improvements to the way global variables are tracked and presented
- Integrated ftrace-based coverage analysis (https://youtu.be/ghUBAndh_uA)

2025 Plan

- Resolve the “indirect function calls” problem



BASIL

What is it?

- Lifecycle artifact traceability tooling with many excellent features!
- Get it: <https://github.com/elisa-tech/BASIL>
- Try it: <http://elisa-builder-00.iol.unh.edu:9056/>

2024 Updates

- Established a public instance for people to experiment
- Various new features linking test cases to requirements specs, etc.
- Triggering test in external CI systems and linking to results

2025 Plan

DeltaKernel

What is it?

- Provides a visual report of change impact between two kernel versions
- Get it: <https://github.com/elisa-tech/delta-kernel/>

2024 Updates

- Initial release of the tooling

2025 Plan

- No active plan for 2025 at this moment
- Extend analysis to the kernel configuration

Linux Kernel Git Diff Report

Comparing tags [v5.15](#) and [v5.15.100](#) in the [linux source code repository](#).

[Uncollapse All](#) [Collapse All](#) [Print PDF](#)

```
security ▲
security/device_cgroup.c Link ▼
security/integrity/integrity_audit.c Link ▼
security/keys/keyctl_pkey.c Link ▼
security/security.c Link ▲

diff --git a/security/security.c b/security/security.c
index 9ffa9e9c5c55..a97079e12c67 100644
--- a/security/security.c
+++ b/security/security.c
@@ -59,10 *59,12 @@ const char *const lockdown_reasons[LOCKDOWN_CONFIDENTIALITY_MAX+1] = {
 [59] [LOCKDOWN_DEBUGFS] = "debugfs access",
 [60] [LOCKDOWN_XMON_WRI] = "xmon write access",
 [61] [LOCKDOWN_BPF_WRITE_USER] = "use of bpf to write user RAM",
 [62] + [LOCKDOWN_DBG_WRITE_KERNEL] = "use of kgdb/kdb to write kernel RAM",
 [63] [LOCKDOWN_INTEGRITY_MAX] = "integrity",
 [64] [LOCKDOWN_KCORE] = "/proc/kcore access",
 [65] [LOCKDOWN_KPROBES] = "use of kprobes",
 [66] [LOCKDOWN_BPF_READ_KERNEL] = "use of bpf to read kernel RAM",
 [67] + [LOCKDOWN_DBG_READ_KERNEL] = "use of kgdb/kdb to read kernel RAM",
 [68] [LOCKDOWN_PERF] = "unsafe use of perf",
 [69] [LOCKDOWN_TRACEFS] = "use of tracefs",
 [70] [LOCKDOWN_XMON_RW] = "xmon read and write access",
@@ -747,25 +749,25 @@ static int lsm_superblock_alloc(struct super_block *sb)
 [749]
 [750] /* Security operations */
 [751]
 [750] -int security_binder_set_context_mgr(struct task_struct *mgr)
 [752] +int security_binder_set_context_mgr(const struct cred *mgr)
 [753] {
 [754] return call_int_hook(binder_set_context_mgr, 0, mgr);
 [755] }
```

llvm-cov

What is it?

- Joint effort between Boeing/UIUC to enhance llvm-cov to achieve MC/DC and object code coverage of the kernel
- Get it: LLVM 19.1 and later!

2024 Updates

- Demo of MC/DC coverage of the kernel at Linux Plumbers
- Started DO-330 qualification effort

2025 Plan

- Initial implementation of object code coverage tooling

```
if (pending && !ksoftirqd_running(pending))
MC/DC Decision Region (458:6) to (458:44)

Number of Conditions: 2
Condition C1 --> (458:6)
Condition C2 --> (458:17)

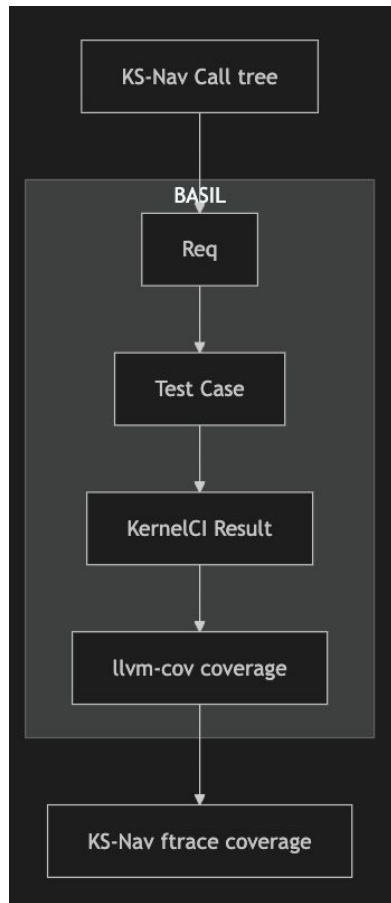
Executed MC/DC Test Vectors:

C1, C2    Result
1 { T, F  = F    }
2 { T, T  = T    }

C1-Pair: not covered
C2-Pair: covered: (1,2)
MC/DC Coverage for Expression: 50.00%
```

2025 Vision

- Present a use case which connects several existing tools/efforts
 - Leverage ks-nav call trees to inform requirements understanding
 - Kernel requirements / documentation effort: Create a req in the ftrace area against an existing test case
 - (NEW) Create CI which loads kernel data into a live instance of ELISA
 - (NEW) Work to get llvm-cov coverage data capture in KernelCI testing
 - Demonstrate linkage of Req □ Test Case □ Test Result □ Coverage
 - Demonstrate ftrace-based coverage on the same test
- Big Stretch: Demonstrate how this data can be used for change impact analysis



Get Involved

- Join our meetings!
 - 2nd Tuesday of the month @ 9:30 AM EST / 2:30 PM UTC
 - 4th Thursday of the month @ 11:00 AM EST / 4:00 PM UTC
- Hit the Mailing List!
 - <https://lists.elisa.tech/g/tool-investigation>
- Participate in our GitHub
 - Tools WG: <https://github.com/elisa-tech/wg-tools>
 - BASIL: <https://github.com/elisa-tech/BASIL>
 - ks-nav: <https://github.com/elisa-tech/ks-nav>
 - DeltaKernel: <https://github.com/elisa-tech/delta-kernel/>



ELISA

Enabling **Linux** in
Safety Applications

Thank you!