Securing Low-latency Hardware Designs

Adam Sherer, Account Technical Executive, Cadence Design Systems

STAC Spring 2022 Summits

cadence®

Cyber Security: Who Cares?

Aero/Def, auto, medical, industrial, comms, IoT, semi, robotics, …

Financial Technology

Knight Capital Group

SOFTWARE BUGS IN HISTORY

Losing $460m in 45 minutes

Source: https://www.bugsnag.com/blog/bug-day-460m-loss

SECURITY HARDWARE BRANCHSCOPE

BranchScope attack successfully demonstrated on several Intel CPUs

BranchScope exploits another weakness in CPU branch prediction

By William Gayde on Mar 27, 2018, 10:26

Security researchers at several American univers
attack that can expose sensitive system data by
as BranchScope, uses some of the same predict
and shows just how problematic they can be.

TECH CHANGING FACE OF SECURITY

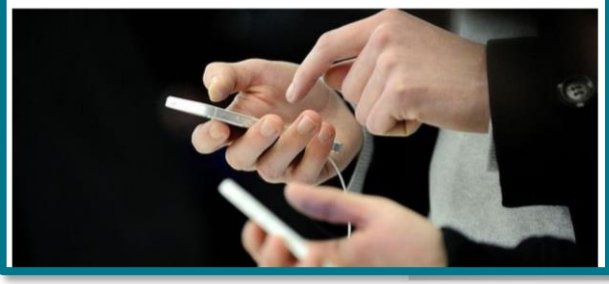
How Qualcomm Flaws Left 900 Million Android Devices Vulnerable to Spies

by David Meyer AUGUST 8, 2016, 9:59 AM EDT

Technology

Hardware hack defeats iPhone passcode security

19 September 2016 | Technology Share



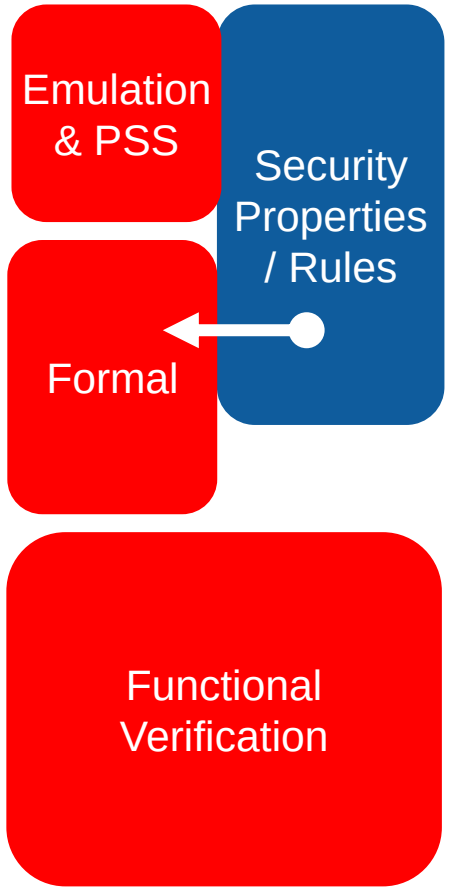
- Impact: $460M loss in 45 minutes
- Multiple factors in loss
  - Improperly set flag put system into test mode
  - Improperly configured production env
  - Dead code in production env (“Power Peg”)
  - Lack of formal QA process
- Bug could have been a security issue
  - “Power Peg” was essentially a trojan

Build To Objective Security Analysis

More Objective

Property / Rules Driven Security	• Comprehensive asset analysis via security properties / rules • System level security scenario verification
Asset-Specific Security	• Formal property analysis guided by rules/SME analysis • Comprehensive Mitre CWE analysis
Security Aware Verification	• Mitre CWE (ex. 1193 power-on, 1276 connectivity, 1242 undocumented features, 1245 improper state machines)
Comprehensive Verification	• Automated metrics analysis to verification plan • Safety and rad-hard verification as needed
Essential Verification	• 100% code coverage w/ dead code and waiver analysis • Comprehensive lint analysis including coding weakness

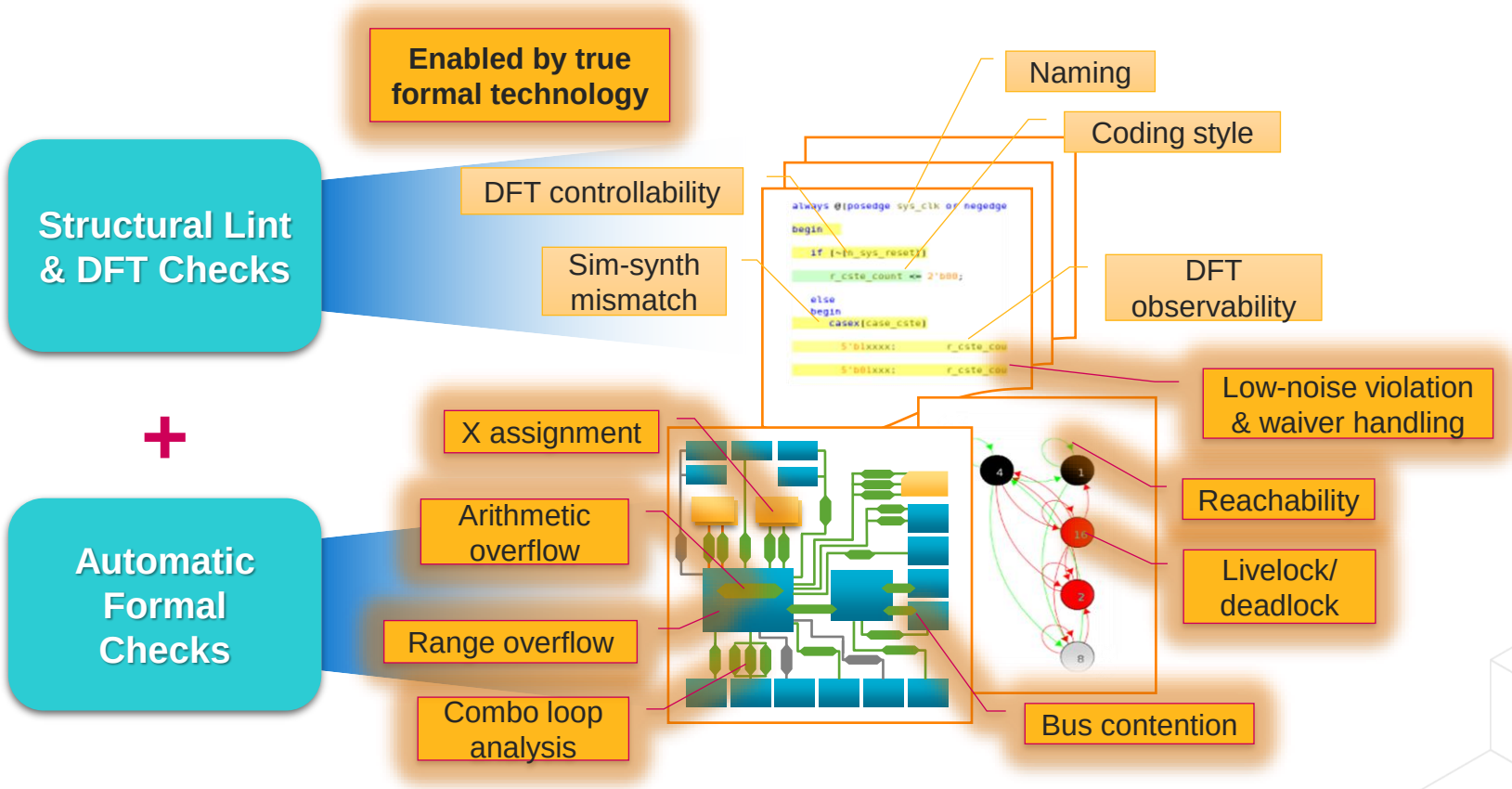
More Subjective



- Security starts with a foundation of comprehensive verification
- All projects should be security aware, adding technology to support by security requirements

Using Lint Checks to Reduce Potential Attack Surfaces

- Coding style can leave a design exposed
  - Ex: side-channel exploit could be forcing data input that causes a register overflow resulting in a denial of service
  - Ex: trojan hiding in statemachine
- Examples coding issues
  - Undefined states in explicit and implicit statemachines
  - Incomplete if-then-else statements
  - Uninitialized variable/signal states
  - Livelock/deadlock states
  - Unguarded overflow/ underflow registers/ queues/arrays, etc.
- Dead code elimination
  - Code coverage analysis



Common Weakness Enumeration

A Community-Developed List of Software & Hardware Weakness Types

- CWE: list of known weaknesses that a secure system should not have
- Mitre Corp maintains on-line db
- Started with SW CWE, HW added about a year ago

1194 - Hardware Design

- Manufacturing and Life Cycle Management Concerns - *(1195)*
- Security Flow Issues - *(1196)*
- Integration Issues - *(1197)*
- Privilege Separation and Access Control Issues - *(1198)*
- General Circuit and Logic Design Concerns - *(1199)*
- Core and Compute Issues - *(1201)*
- Memory and Storage Issues - *(1202)*
- Peripherals, On-chip Fabric, and Interface/IO Problems - *(1203)*
- Security Primitives and Cryptography Issues - *(1205)*
- Power, Clock, and Reset Concerns - *(1206)*
- Debug and Test Problems - *(1207)*
- Cross-Cutting Problems - *(1208)*

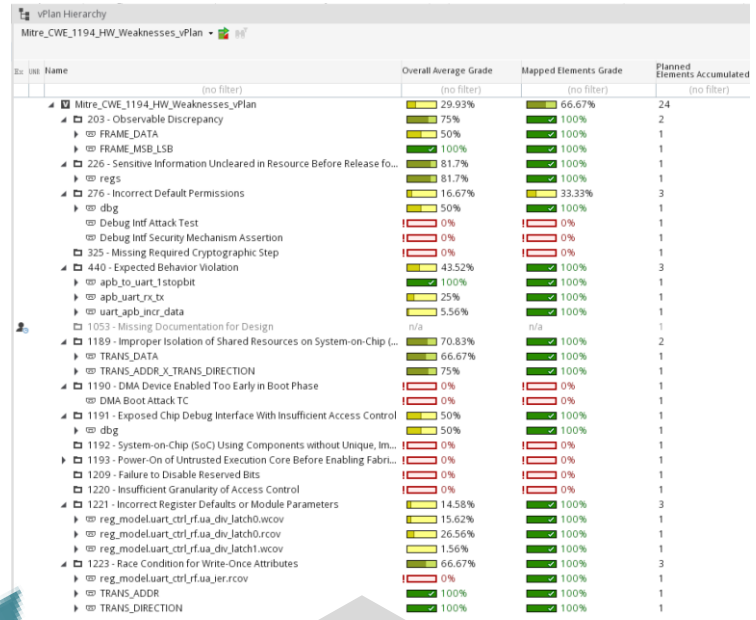
View Metrics

	CWEs in this view		Total CWEs
Weaknesses	96	out of	922
Categories	12	out of	316
Views	0	out of	44
Total	**108**	out of	1282

Security Verification Plan – Central Aggregation Point for Security Data

Security Requirements (from req. mgmt.)

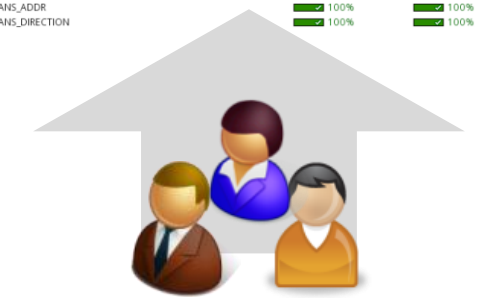
Security vPlan



Link to Security Specs

Identify Security Metrics

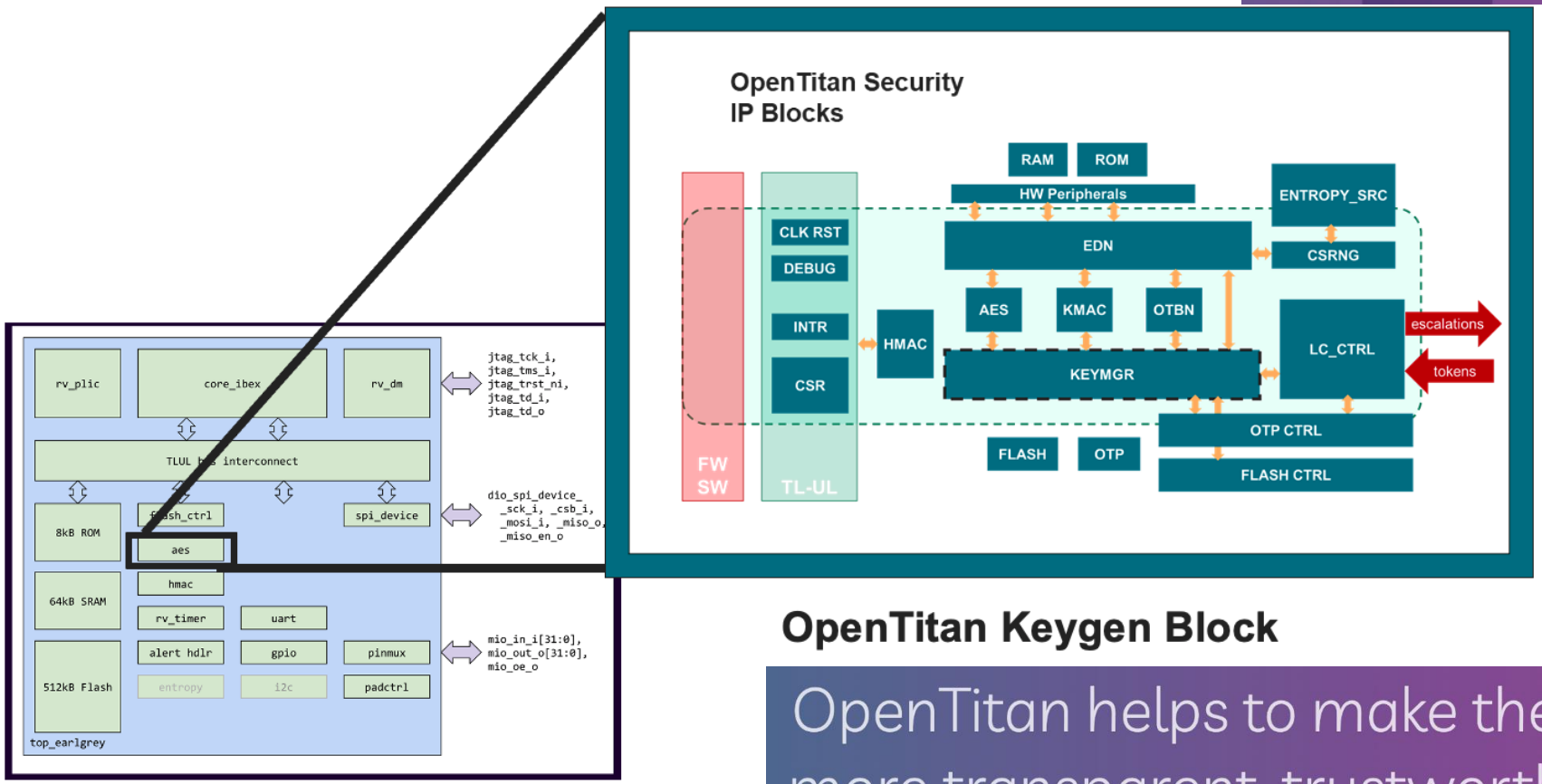
Import and Reuse Security Plans



Collaborate on common Security Plan

Security Reference Platform

OpenTitan is the first open source project building a transparent, high-quality reference design and integration guidelines for silicon root of trust (RoT) chips.



OpenTitan EarlGrey SoC

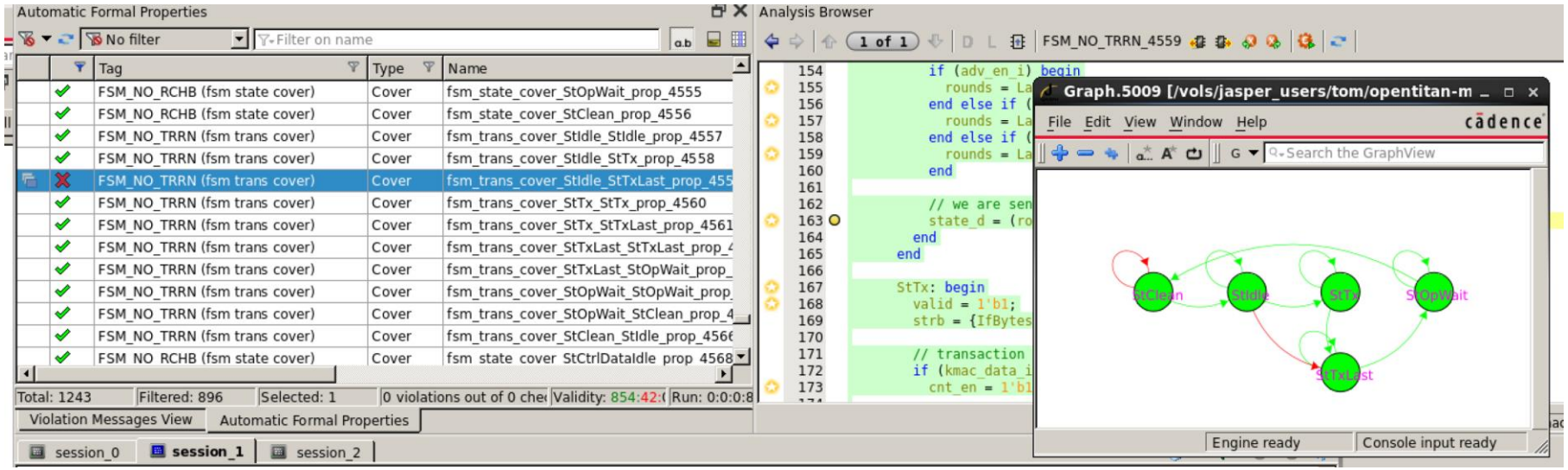
OpenTitan Keygen Block

OpenTitan helps to make the silicon root of trust (RoT) more transparent, trustworthy, and ultimately, secure.

Improper Finite State Machines in HW Logic with SuperLint

CWE	Description	Formal Application
1245	Improper Finite State Machines (FSMs) in Hardware Logic	SuperLint

- Faulty finite state machines (FSMs) in HW logic allow attacker to put system in undefined state causing denial of service (DoS) or gain privileges to system
- Formal analysis automatically extracts all states and transitions for each FSM in design
- Unreachable states/transitions indicate potential weaknesses where faults can lead the design into unknown behaviors



cadence

Key Overwrite in Wipe Mode Found with Formal Property Verification

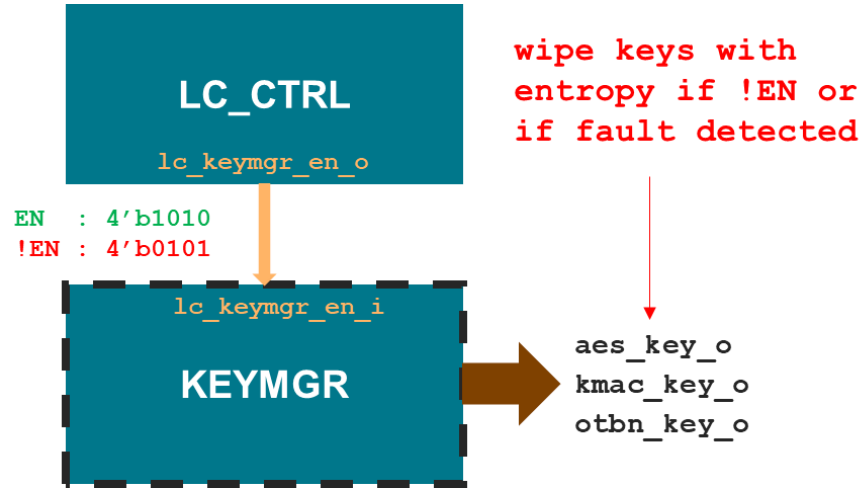
CWE	Description	Formal Application
1258	Exposure of Sensitive System Information Due to Uncleared Debug Information	Formal Property Verification

Disabled

Disabled is a state where the key manager is no longer operational. Upon Disabled entry, the working state is updated with KMAC computed random values; however, sideload keys are preserved. This allows the software to keep the last valid sideload keys while preventing the system from further advancing the valid key.

When advance and generate calls are invoked from this state, the outputs and keys are indiscriminately updated with randomly computed values. Key manager enters disabled state based on direct invocation by software:

- Advance from OwnerRootKey
- Disable operation



LFSR must be enabled in key wipe state

```
assert {u_ctrl.state_q == StCtrlWipe |-> (ctrl_lfsr_en && wipe_key)}

# key values must be filled with random data
assert {u_ctrl.state_q == StCtrlWipe |-> kmac_key.key[0] == {8{ctrl_rand[0]}} }
assert {u_ctrl.state_q == StCtrlWipe |-> kmac_key.key[1] == {8{ctrl_rand[1]}} }

# software operations are forbidden when keymanager is disabled during key wipe
assert {!u_ctrl.en_i |-> u_ctrl.disable_sel && stage_sel == Disable}
```

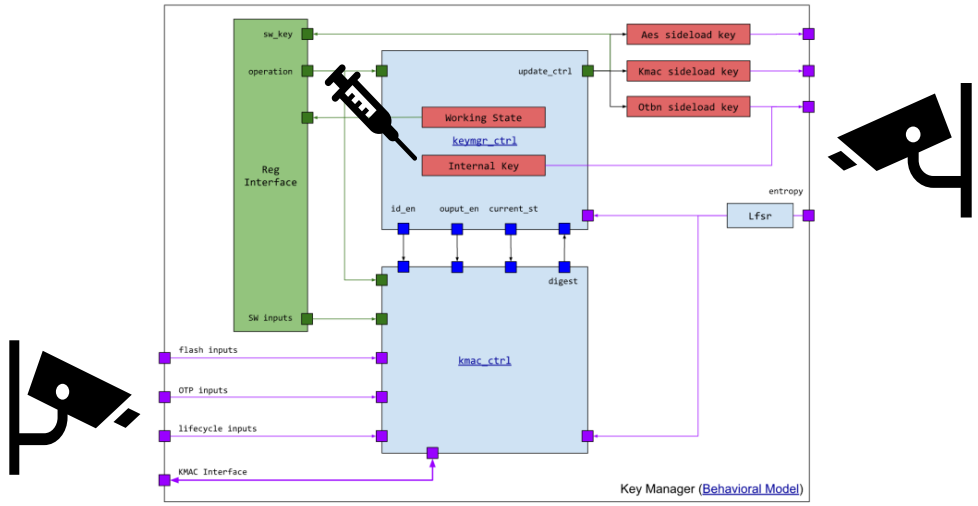
	Type	Name	Engine	Bound	Time	Task
✗ !	Assert	in_StCtrlWipe_wipe_lfsr_en	L	336 - 374	436.1	fpv_cwe1258
✗	Cover (relat...	in_StCtrlWipe_wipe_lfsr_en:witness1	N (4)	Infinite	0.0	fpv_cwe1258
✓	Cover (relat...	in_StCtrlWipe_wipe_lfsr_en:precondition1	L	330 - 374	436.1	fpv_cwe1258

Key Verification with Security Path Verification (SPV)

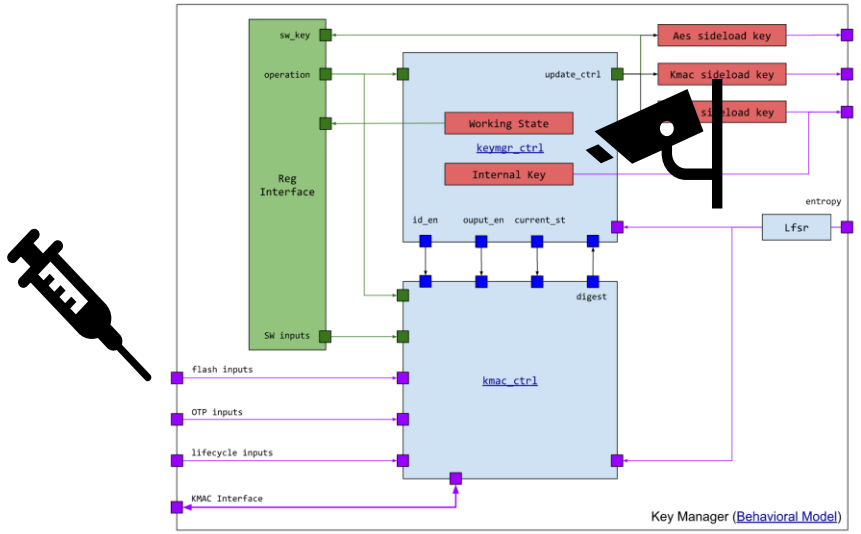
CWE	Description	Weakness
1263	Improper Physical Access Control	Data Confidentiality and integrity
1282	Assumed-Immutable Data is Stored in Writable Memory	Data integrity
1258	Exposure of Sensitive System Information Due to Uncleared Debug Information	Data confidentiality
1330	Remnant Data Readable after Memory Erase	

- Internal key is maintained inside of the keymgr_ctrl block

Confidentiality (leakage) – inject taint (unique tag) at the key and look for propagation at block outputs



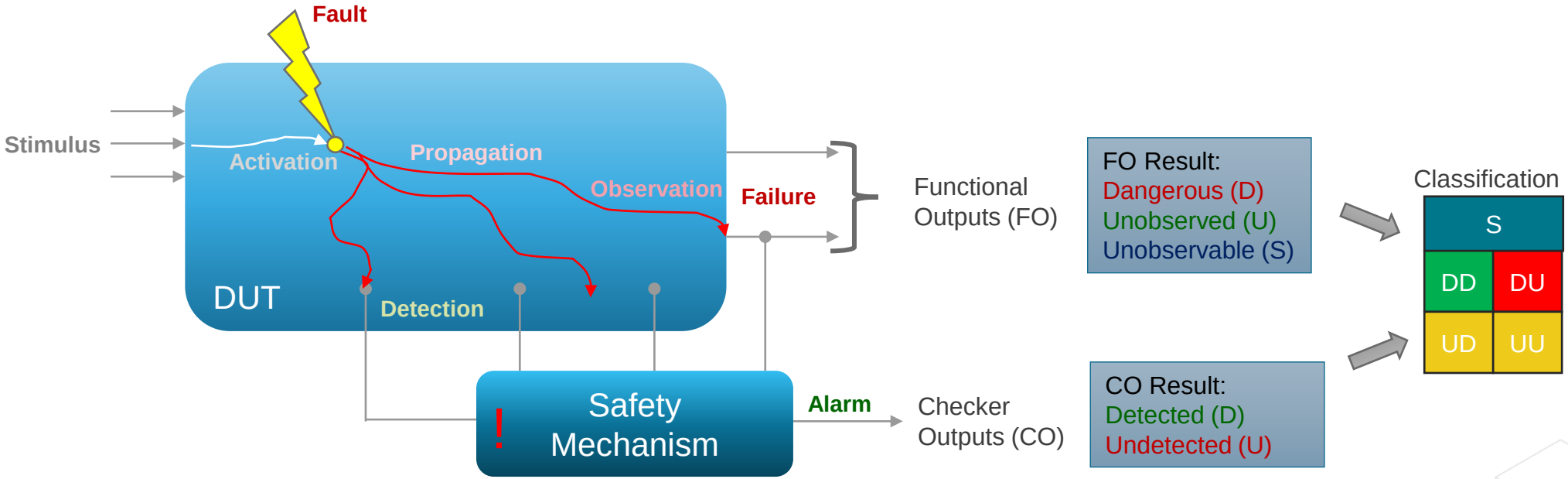
Integrity (corruption) – inject taint at the inputs and look for propagation to key



Formal Safety Verification Exposes Security Vulnerability Analysis

CWE	Description	Formal Application
1261	Improper Handling of Single Even Upsets	Formal Safety Verification (FSV)

- Check ability to detect or eliminate hacker attacks in secure subsystems
  - IP is protected against hacker attacks
    - by sensors and checkers - attack raises alarm
    - by error correction mechanism - attack is eliminated
  - Goal is to detect or correct all attacks



Negative Testing Exposes Illegal Scenarios Using PSS*

PSS

```
action generate_key {
...
  share sml_core_r core;
  output sml_data_buff token_out;

  // API Parameters
  rand crypto_algo_e algo;
  rand bool extractable;
  rand crypto_key_usage_e key_usage;

  constraint crypto_mem_blks {
    !allow_illegal_mem ->
        token_out.mem_seg.mem_block in [MEM0, MEM1];
  }


  rand bool allow_illegal_mem;

  constraint default allow_illegal_mem == false;

}
```

LEGAL Spec – Crypto accesses only MEM0 and MEM1

Attribute to allow illegal memories

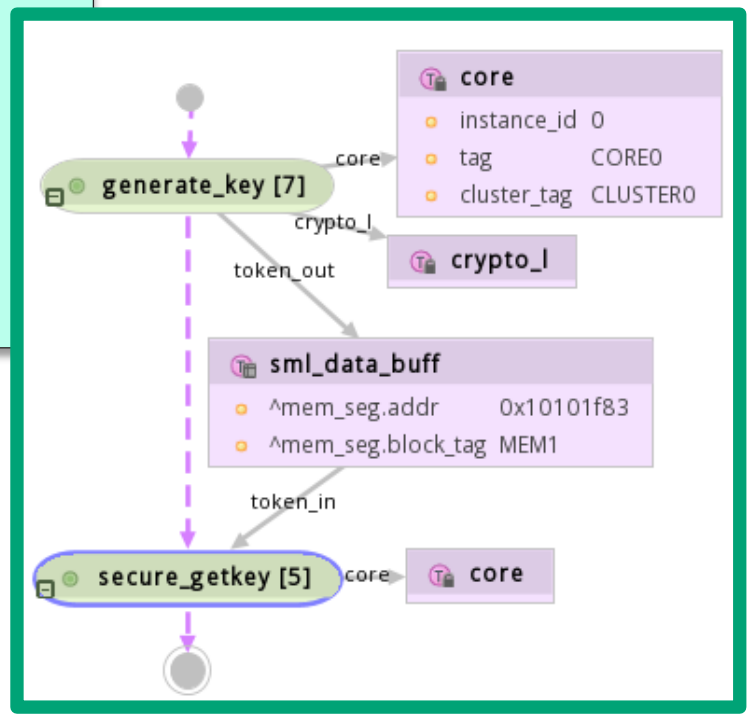
By default, allow only legal values

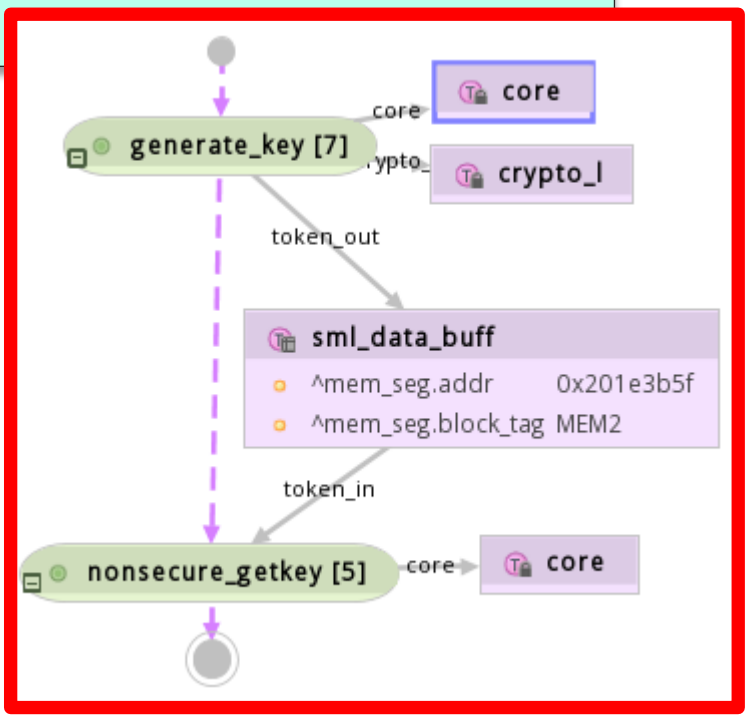
PSS

```
action crypto_illegal_mem_test {
  activity {
    do crypto_c::generate_key with {
      default disable allow_illegal_mem;
      allow_illegal_mem != false;
      token_in.mem_seg.mem_block == MEM2;
}
    do security_mod_c::nonsecure_getkey;
  }
}
```

ILLEGAL – Scenario to force crypto to write keys to illegal memory

Disable default (legal) rule





Concept is applicable to any SoC resources, state variables, protocol/memory interfaces, etc.

*PSS = Portable Stimulus Standard (Accellera) defines stimulus usable in simulation, emulation, and ASIC/FPGA

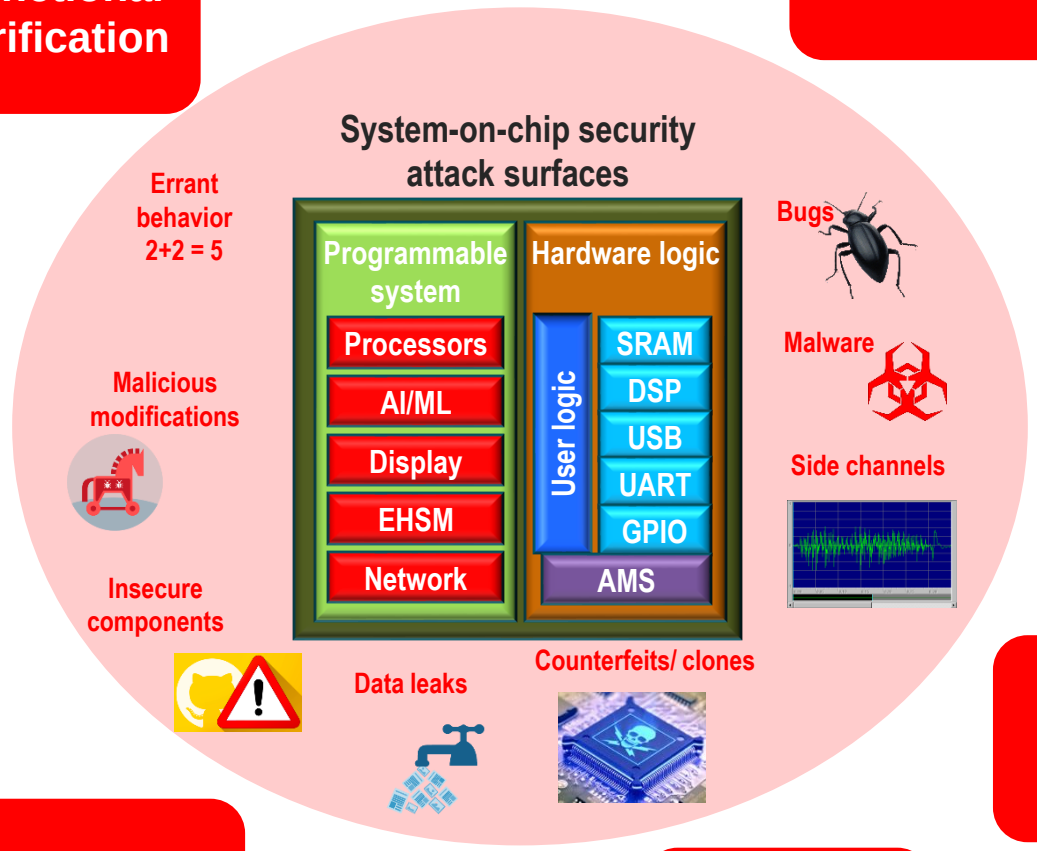
Broad Array of Security Solutions is Needed

Functional verification
Multi engine
Planning & management
Portable stimulus gen
HW/SW verification/debug

Formal Equivalence
Logical equivalence
Sequential equivalence
Power management verification

Formal Verification
Security path verification
Formal property verification
Register verification
Fault/metastability injection

Traceability tracking
vPlan/tracking/mgt
Interface to reqts mgt



Processor IP
Encryption
Authentication
HROT compatible
Secure RTOS compatible
Auto qualified safety

Design IP
Auto qualified
Adv node
Standards interfaces
Converters/mixed-signal

SW dev/debug
HW verif links to SW envs
Accurate HW/SW interaction
Blended abstraction w/ real devices

Physical analysis
Power
Thermal
Reliability/aging
EMIR

Security design services
Security function implementation
Advanced node expert team
Cleared personnel

Security Call to Action

- Assess current functional and security verification methodology
  - More comprehensive verification reduces security risk
- Discuss how security levels of assurance are applied on projects
  - Set security objectives
- Execute additional security verification
  - Discuss additional security verification goals
  - Identify joint engineering team to bring-up and apply new tools/methodology
  - Measure results from application of new tools/methodology
  - Document results and support materials/training to bring-up new projects

Cadence Security Verification Solution and Partners