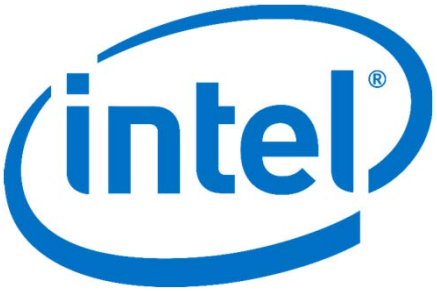
presented by



Hardening Firmware Components with Host-based Analysis

Spring 2019 UEFI Plugfest

April 8-12, 2019

Presented by Brian Richardson (Intel Corporation)

Agenda

- Today's Reality for Firmware Developers
- Examine Tools for Driver & Application Developers
- Host-based Firmware Analyzer
- Call to Action

Today's Reality for Firmware Developers

- Platform firmware is an essential component in software root-of-trust.
- Platform firmware is a low-level component, so potential security risks may not be apparent to users or developers.
- Most firmware validation is done via integration testing, which emphasizes functionality and stability.
- Integration testing is not ideal for detecting potential vulnerabilities in new firmware drivers.

Common Tools for Firmware Security Testing

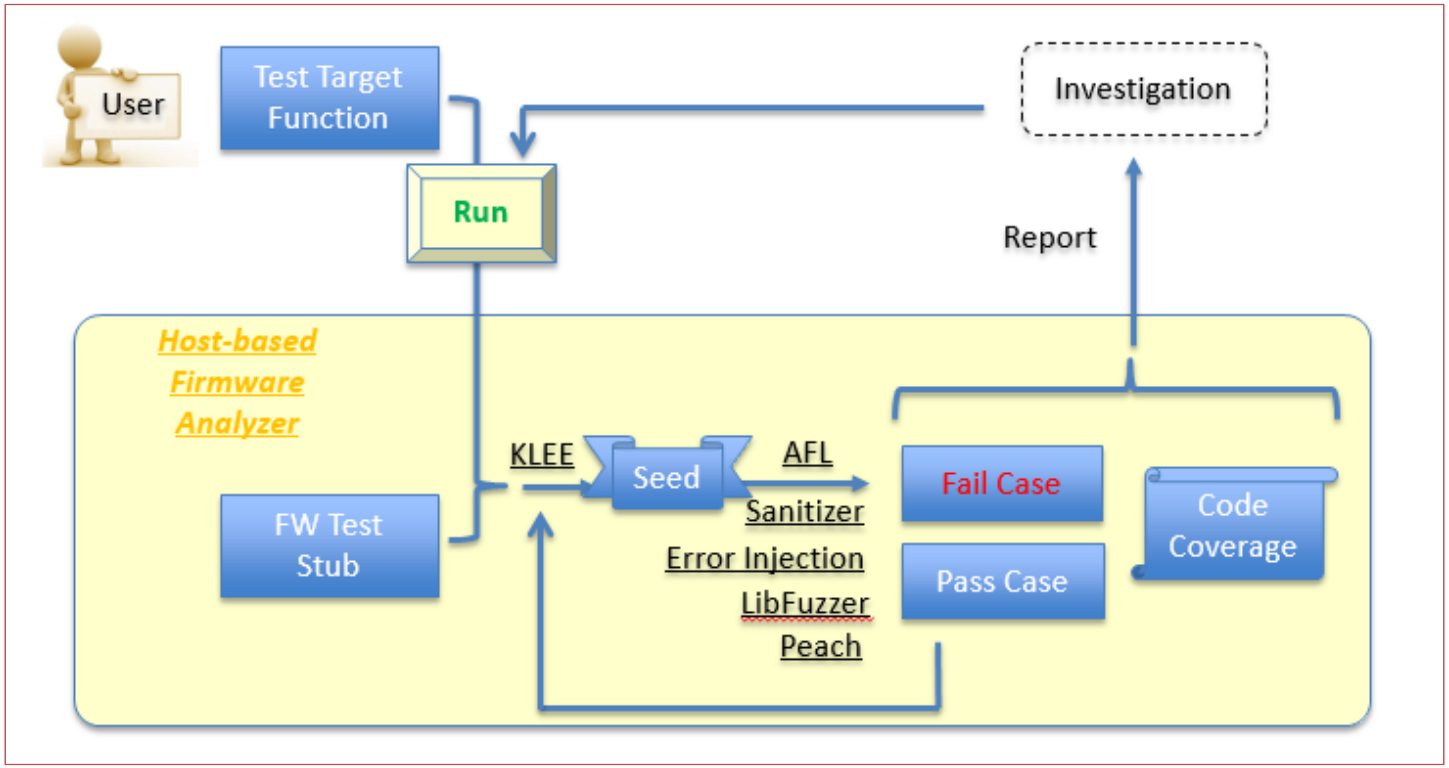
- CHIPSEC: open source framework for analyzing the security of platform firmware and hardware configuration at runtime, based on the Unified Extensible Firmware Interface (UEFI) specification.
- Code Coverage: Tools like the Intel® Intelligent Test System measure the amount of firmware code executed during test runs (high percentage is better).
- Symbolic Execution and Virtual Platforms: Intel's Excite project uses a combination of symbolic execution, fuzzing, and concrete testing to find vulnerabilities in firmware running on Wind River* Simics* virtual platforms.
- *Great tools… but they're based on integration testing, so issues are more expensive to detect and mitigate. Can we improve testing before integration?*

Examine Tools for Driver & Application Developers

- Fuzzing: Test application programming interfaces (APIs) by subjecting them to random, invalid, unexpected, or untrusted (potentially malicious) inputs.
- Address Sanitizing: Detect memory corruption issues such as heap buffer overflow, stack buffer overflow, and global buffer overflow.
- Code Coverage: Identify code paths not executed during validation so test scope can be increased to avoid corner cases.

- ***The Challenge:*** *How can firmware developers use OS-based test tools on isolated firmware components?*

Host-based Firmware Analyzer (HBFA)

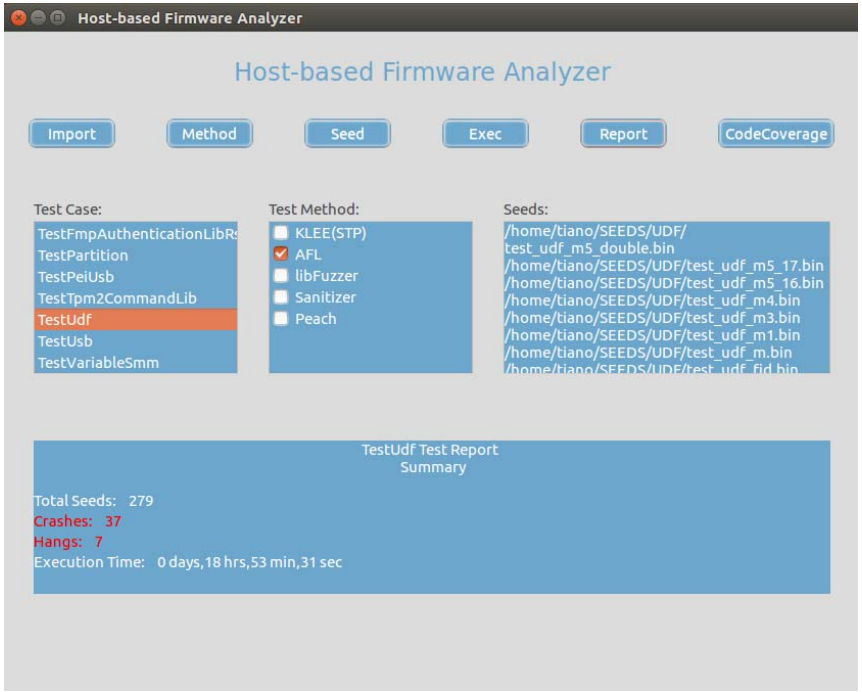


Testing UEFI firmware drivers in the developer's environment

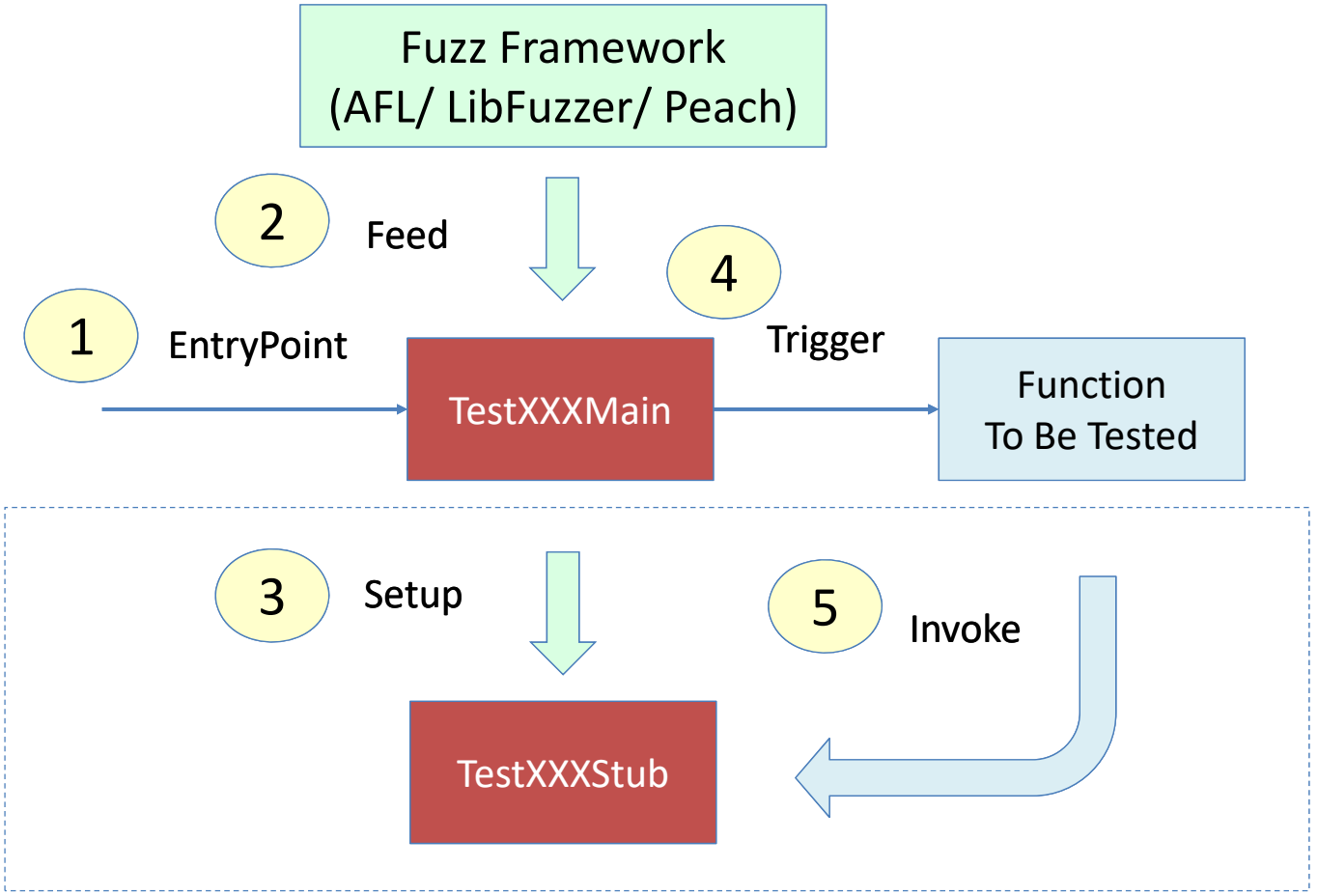
Host-based Firmware Analyzer

OS based environment utilizing best-in-class test tools

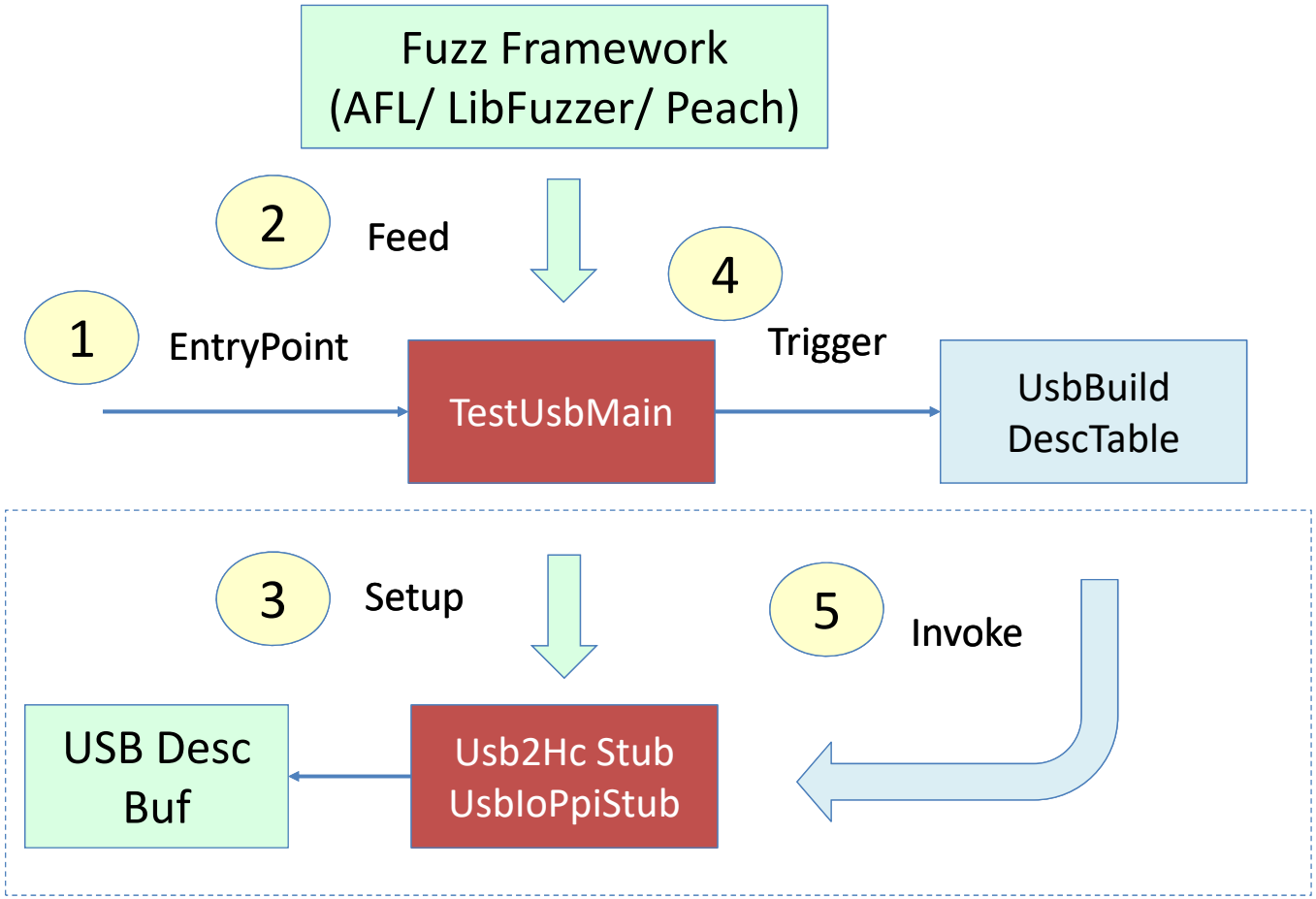
- GUI and command-line interfaces
- Fuzzing testing
  - AFL, libFuzzer, Peach
- Symbolic execution (KLEE/STP)
- Address Sanitizer & Code Coverage
- Automated unit test execution (Cunit)
- Instrumentation methods for fault injection and trace
- Database of unit test cases



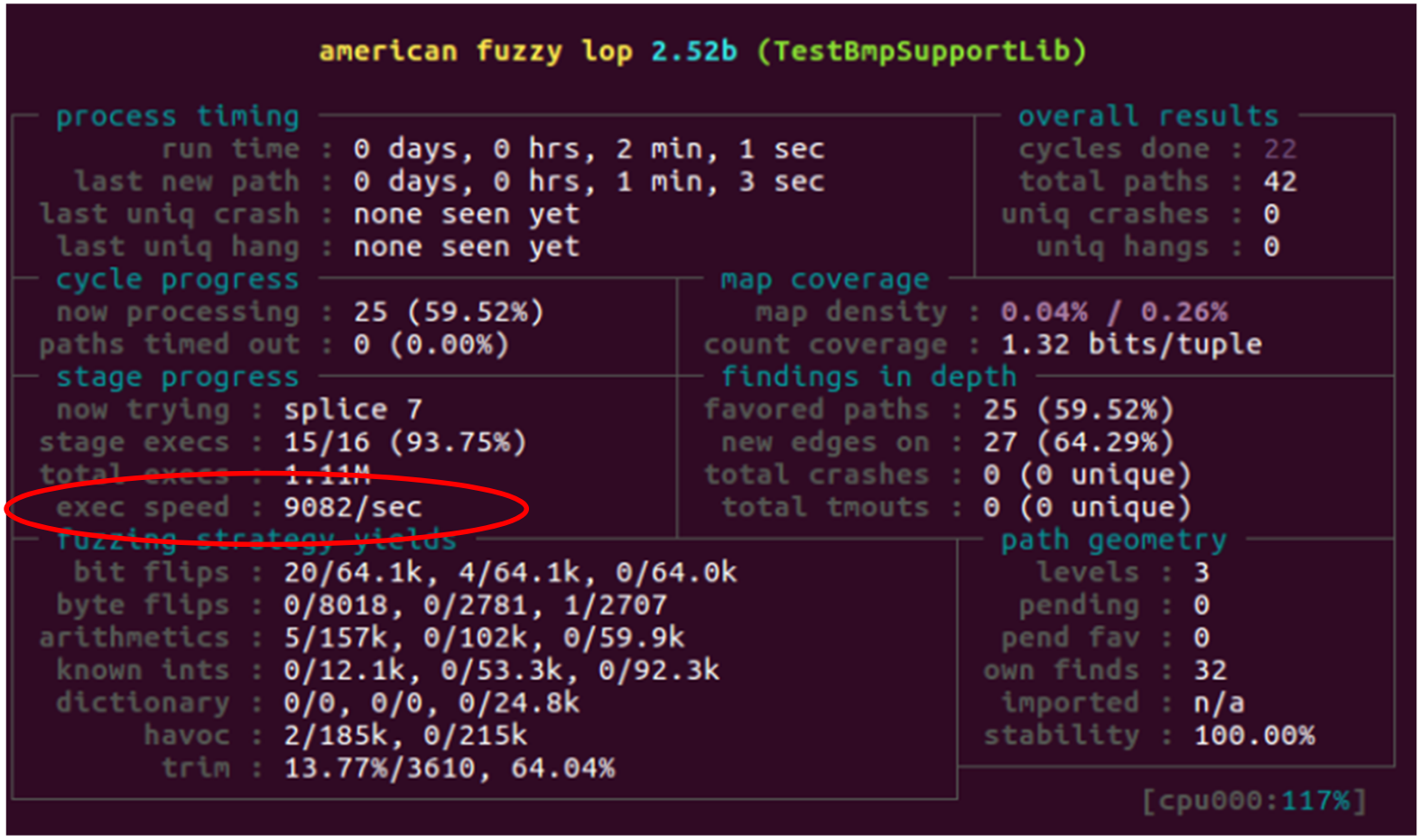
Host-based Firmware Analyzer - Case Design



Host-based Firmware Analyzer – USB Test



Example - AFL


american fuzzy lop 2.52b (TestBmpSupportLib)
process timing
run time : 0 days, 0 hrs, 2 min, 1 sec
last new path : 0 days, 0 hrs, 1 min, 3 sec
last uniq crash : none seen yet
last uniq hang : none seen yet
overall results
cycles done : 22
total paths : 42
uniq crashes : 0
uniq hangs : 0
cycle progress
now processing : 25 (59.52%)
paths timed out : 0 (0.00%)
map coverage
map density : 0.04% / 0.26%
count coverage : 1.32 bits/tuple
stage progress
now trying : splice 7
stage execs : 15/16 (93.75%)
total execs : 1.11M
exec speed : 9082/sec
findings in depth
favored paths : 25 (59.52%)
new edges on : 27 (64.29%)
total crashes : 0 (0 unique)
total tmouts : 0 (0 unique)
fuzzing strategy yields
bit flips : 20/64.1k, 4/64.1k, 0/64.0k
byte flips : 0/8018, 0/2781, 1/2707
arithmetics : 5/157k, 0/102k, 0/59.9k
known ints : 0/12.1k, 0/53.3k, 0/92.3k
dictionary : 0/0, 0/0, 0/24.8k
havoc : 2/185k, 0/215k
trim : 13.77%/3610, 64.04%
path geometry
levels : 3
pending : 0
pend fav : 0
own finds : 32
imported : n/a
stability : 100.00%
[cpu000:117%]

Example – Peach + Sanitizer