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

www.uefi.org
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?
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

Fuzz Framework
(AFL/ LibFuzzer/ Peach)

1. **EntryPoint**
2. **Feed**
3. **Setup**
4. **Trigger**
5. **Invoke**

- TestXXXMain
- Function To Be Tested
- TestXXXStub
Host-based Firmware Analyzer – USB Test

- **Fuzz Framework (AFL/ LibFuzzer/ Peach)**
  - **Feed**
  - **Trigger**
  - **EntryPoint**
  - **TestUsbMain**
  - **UsbBuild DescTable**
  - **USB Desc Buf**
  - **Usb2Hc Stub**
  - **UsbIoPpiStub**

- **Setup**
- **Invoke**
### Example - AFL

**american fuzzy lop 2.52b (TestBmpSupportLib)**

<table>
<thead>
<tr>
<th>Process Timing</th>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run time: 0 days, 0 hrs, 2 min, 1 sec</td>
<td>Cycles done: 22</td>
</tr>
<tr>
<td>Last new path: none seen yet</td>
<td>Total paths: 42</td>
</tr>
<tr>
<td>Last uniq crash: none seen yet</td>
<td>Uniq crashes: 0</td>
</tr>
<tr>
<td>Last uniq hang: none seen yet</td>
<td>Uniq hangs: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle Progress</th>
<th>Map Coverage</th>
<th>Findings in Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Now processing: 25 (59.52%)</td>
<td>Map density: 0.04% / 0.26%</td>
<td>Favored paths: 25 (59.52%)</td>
</tr>
<tr>
<td>Paths timed out: 0 (0.00%)</td>
<td>Count coverage: 1.32 bits/tuple</td>
<td>New edges on: 27 (64.29%)</td>
</tr>
<tr>
<td>Stage progress</td>
<td></td>
<td>Total crashes: 0 (0 unique)</td>
</tr>
<tr>
<td>Now trying: splice 7</td>
<td></td>
<td>Total tmouts: 0 (0 unique)</td>
</tr>
<tr>
<td>Stage execs: 15/16 (93.75%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total execs: 1.11M</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exec speed:</strong> 9082/sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fuzzing Strategy Yields**

<table>
<thead>
<tr>
<th>Bit Flips</th>
<th>Byte Flips</th>
<th>Arithmetics</th>
<th>Known Ints</th>
<th>Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>20/64.1k, 4/64.1k, 0/64.0k</td>
<td>0/8018, 0/2781, 1/2707</td>
<td>5/157k, 0/102k, 0/59.9k</td>
<td>0/12.1k, 0/53.3k, 0/92.3k</td>
<td>0/0, 0/0, 0/24.8k</td>
</tr>
<tr>
<td>Havoc</td>
<td>2/185k, 0/215k</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trim</td>
<td>13.77%/3610, 64.04%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Path Geometry**

<table>
<thead>
<tr>
<th>Levels</th>
<th>Pending</th>
<th>Pend Fav</th>
<th>Own Finds</th>
<th>Imported</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>n/a</td>
<td>100.00%</td>
</tr>
</tbody>
</table>
Example – Peach + Sanitizer

```
==9294==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000032
at pc 0x000000052c278 bp 0x7fffffffa0 sp 0x7fffffffa98
READ of size 2 at 0x602000000032 thread T0
  #0 0x52c277 in UsbBuildDescTable /home/tiano/TEST/edk2/UefiHostTestPkg/TestCase/MdeModulePkg/Bus/Usb/UsbBusDxe/UsbDesc.c:830:64
  #1 0x5297ef in main /home/tiano/TEST/edk2/UefiHostTestPkg/TestCase/MdeModulePkg/Bus/Usb/UsbBusDxe/TestUsb.c:94:3
  #2 0x7ffffff6ee582f in __libc_start_main /build/glibc-Cl5G7W/glibc-2.23/csu/../csu/libc-start.c:291
  #3 0x41a728 in _start (/home/tiano/TEST/edk2/Build/UefiHostTestPkg/DEBUG_CLANG_BUILD=64/TestUsb+0x41a728)

0x602000000032 is located 1 bytes to the right of 1-byte region [0x602000000030, 0x602000000031)
allocated by thread T0 here:
  #0 0x4e945f in _interceptor_malloc /home/tiano/Downloads/llvm/projects/compiler-rt/lib/asan/asan_malloc_linux.cc:146
  #1 0x52d1e4 in AllocateZeroPool /home/tiano/TEST/edk2/UefiHostTestPkg/Library/AllocationLibHost/AllocationLibHost.c:37:12
  #2 0x52beb1 in UsbGetOneConfig /home/tiano/TEST/edk2/UefiHostTestPkg/TestCase/MdeModulePkg/Bus/Usb/UsbBusDxe/UsbDesc.c:744:9
  #3 0x52bfc9 in UsbBuildDescTable /home/tiano/TEST/edk2/UefiHostTestPkg/TestCase/MdeModulePkg/Bus/Usb/UsbBusDxe/UsbDesc.c:814:14
```