UEFI and Security Development Lifecycle (SDL) – Unit Testing

Fall 2018 UEFI Plugfest
October 15 – 19, 2018
Presented by Trevor Western (Insyde Software)
Agenda

- SDL and Unit Testing
- Why Unit Test
- Real world example
- Recommendations
- Resources

www.uefi.org
The Security Development Lifecycle

- The SDL improves the capability to support, design, develop, test and release secure software.
  - Improved support through training and in-house security expertise.
  - Improved design using risk assessment and threat modeling.
  - Improved development with best practices that minimize chances of attacks.
  - Improved testing using tools to detect and test for vulnerabilities.
  - Improved response by root causing, deploying fixes, informing customers and updating tests.

- The Security Development Lifecycle promotes continuous improvement.
What are Unit Tests?

- Collections of test cases that verify the functionality and behavior of new code; and prevent “breaking” previously checked in code.

- The scope of a test case is limited to the smallest piece of testable code.

- Meant to run automatically and frequently.

- Can be used to guide software development (Test-Driven Development or Test-First-Code-After Development).
SDL and UEFI Unit Testing

• Why do we need Unit Tests?
  – Provide visible evidence that the new code functions and behaves correctly in the form of PASSED/FAILED report.
  – Prevent new code from “breaking” previously checked in code.
  – Provide reproducible and verifiable results for QA reports.
  – Promote good software development practices (SDL)
SDL and UEFI Unit Testing

• Many tests exist for the UEFI Runtime interfaces, such as the UEFI Self-Certification Test (SCT), and the Canonical Firmware Test Suite (FWTS)
• Start with tests for internal UEFI modules for pre-OS
• You can’t write Unit Tests for everything at once
  - Start with writing Unit Tests for bugs; or
  - Write Unit Tests for new code
• Keep the Unit Test code in same location as UEFI module code
  - They should be maintained together
  - Use a Unit Test Framework (Test Harness) to manage the tests
SDL and UEFI Unit Testing

• A Unit Test Harness provides the following capabilities:
  – A common language to express test cases (usually ‘C’) 
  – A common language to express the expected results
  – Access to the features of the production code
  – A place to collect the Unit Test cases for the project
  – A mechanism to run all the Unit Test cases
  – A small summary report of the test suite success or failure
  – A detailed report of any test failures

• The following slides show an example of Unit Testing for a bug
SDL and UEFI Unit Testing

• The ‘MacEmpty’ code below checks if a Mac Address is not null. It has a bug.

```c
MacEmpty( IN UINT8 *MacAddr ){
    UINTN Index;
    UINT8 TempValue = 0;
    For (Index=0; Index < 4; Index++) {
        TempValue = TempValue + UINT8(MacAddr[Index]);
    }
    If( TempValue == 0) return (TRUE)
    Else return (FALSE);
}
```

• Bug: TempValue overflows (sum of MacAddr[0,1,2,3] is a 32 bit value); but only causes a problem if TempValue % 0x100 = 0 (e.g., 0x200; 0x300; etc). This is a really strange bug.
SDL and UEFI Unit Testing

- Create a Unit Test for the MacEmpty() routine. Feed MacEmpty() test data to show the normal working case.

```c
UNIT_TEST_BEGIN (UT_IsMacZero?)
{
    UINT8 TestMacAddrs[4] = {0x00,0x00,0x00,0x00};

    if (MacEmpty(TestMacAddrs) != TRUE) {
        UNIT_TEST_RESULT("MacEmpty Failed Empty Mac test", FAILED)
    } else
        UNIT_TEST_RESULT("", PASS)
};
UNIT_TEST_END
```

- A simple test proves you did not break the working code
SDL and UEFI Unit Testing

• Now create a Unit Test to see if you can catch the bug

```c
UNIT_TEST_BEGIN (UT_Is8BitOverFlowBugFixed?)
{
    UINT8 TestMacAddrs[4] = {0x00, 0xFF, 0x01, 0x00};

    if (MacEmpty(TestMacAddrs) == TRUE) {
        UNIT_TEST_RESULT("MacEmpty has 8BitOverFlow bug", FAILED)
    } else
        UNIT_TEST_RESULT("", PASS)
};
UNIT_TEST_END
```

• A simple test proves the bug is fixed
  - But Unit Test on the unfixed code first
SDL and UEFI Unit Testing

• Collect the Unit Test together for the Test Harness:

```c
UNIT_TEST_GROUP_BEGIN ("MacEmpty")
    & UT_IsMacZero?()
    & UT_Is8BitOverflowBugFixed?()
UNIT_TEST_GROUP_END
```

<make all
Compiling “MacEmpty” ...
Running “MacEmpty” ...
OK (2 tests run, 0 failed)

• Setup the Test Harness to run these tests automatically when this code module changes
SDL and UEFI Unit Testing

Recommendations

• Test the Unit Test Code:
  – Make sure you test the Unit Test code with inputs designed to expose the bug in the unfixed code

• New product code has to be testable:
  – Modular design with well-defined API.
  – Separate functional code from UEFI framework details.

• Unit Tests are stored same place as code and managed by Test Harness
  – Update Unit Tests when code is expected to change.
  – Keep in a common code package (e.g. OurUnitTestPkg)
SDL and UEFI Unit Testing

Recommendations

• Don’t create a test framework or test harness from nothing
  • Several are available for free and easily adaptable
  • Some are designed to work in a UEFI environment
• SDL can now move to the **Response** step:

• Update the Unit Tests to catch the issue

---

**The Security Development Lifecycle**

- The SDL improves the capability to support, design, develop, test and release secure software.
  - Improved **support** through training and increased security expertise.
  - Improved **design** using risk assessment and threat modeling.
  - Improved **development** with best practices that minimize chances of attacks.
  - Improved **testing** using tools to detect and test for vulnerabilities.
  - Improved **response** by root causeing, deploying fixes, informing customers and updating tests.

- The Security Development Lifecycle promotes continuous improvement.
Resources:

• UEFI test tools - http://www.uefi.org/testtools

• “Implementing MicroPython as a UEFI Test Framework” - Spring 2018 UEFI Plugfest March 26-30, 2018 Presented by Chris McFarland (Intel)


• “Test-Driven Development for Embedded C” by James Grenning http://www.pragprog.com/titles/jgade

• Unity test framework / test harness http://unity.sourceforge.net
Thanks for attending the Fall 2018 UEFI Plugfest

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

Thanks to Tim Lewis and Tuan Vu for their contributions to this UEFI presentation