Dasharo User Group #6 and vPub 0xB

HBFA-FL: Host-based Firmware Analyzer-Fuzzing Lite

Brian Delgado, Erik C. Bjorge, Antonio Gomez-Iglesias, and Earl “Lynn” Tipton

Background

Thanks to many contributors for helping make HBFA-FL

- Disclaimer: Views expressed are my own. All opinions are my own. The opinions expressed here belong solely to me and do not reflect the views of my employer Intel.
- Original authors of HBFA and other maintainers/contributors:
  - Brian Richardson, Chris Wu, Jiewen Yao, and Vincent J. Zimmer
  - Wei Liu
- Others including:
  - Tamas Lengyel
  - Miki Demeter
  - Lelia Barlow

- In addition to HBFA-FL note the project **TSFFS by Rowan et al.** https://github.com/intel/tsffs
- My background:
  - Background in vulnerability research and reverse engineering software and hardware
  - Security researcher at Intel
    - Efforts spanning low-level up to cloud
    - Fuzzing of system software
    - Cloud security

Organization

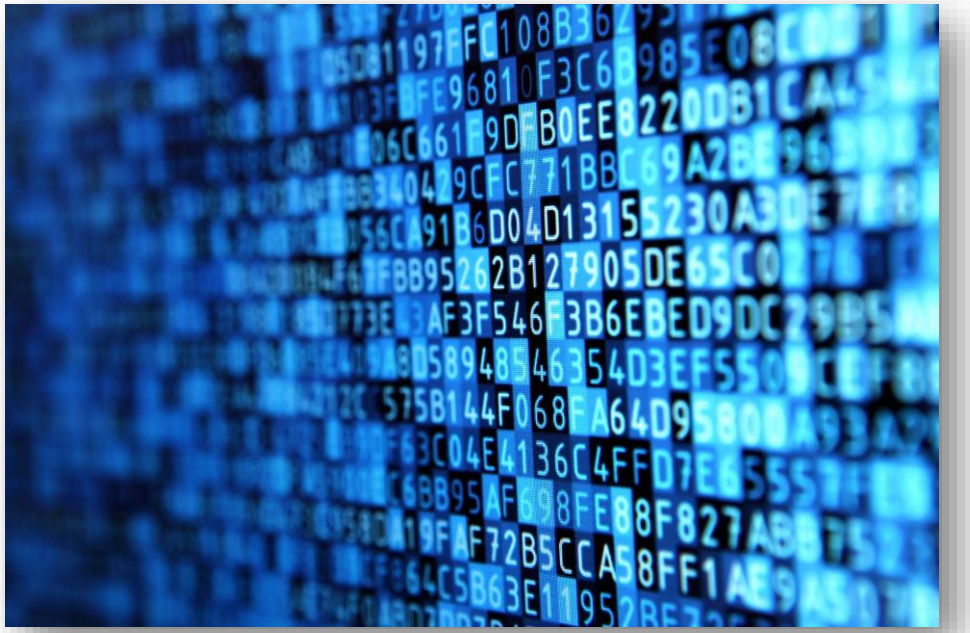
Host-based Firmware Analysis: Fuzzing-Lite

- Fuzz-testing for UEFI
- HBFA: TianoCore edk2-staging branch (2019)
- Motivation – Goals for enhancing HBFA
- Current efforts and features (workflow)
- Fuzzing Efforts - Issues found with HBFA-FL
- Demo

Fuzz Testing for UEFI

Fuzzing is a part of a suite of methods that should be used to enhance security

- A variety of validation should be used, including: †
  - Code review
  - Symbolic execution
  - Unit testing
  - Fuzz testing
- Pick one: Unit testing vs general fuzz testing - "Our opinion is that you don't pick just one testing method" ‡
- "Using fuzzers in unit testing is most effective on verifying low-level UEFI PI interfaces prior to integration" ‡
- Enhance and reduce challenges to leverage tools/frameworks for fuzzing in UEFI



† "Firmware Secure Coding Webinar" UEFI Forum 03/2019: https://uefi.org/node/3942

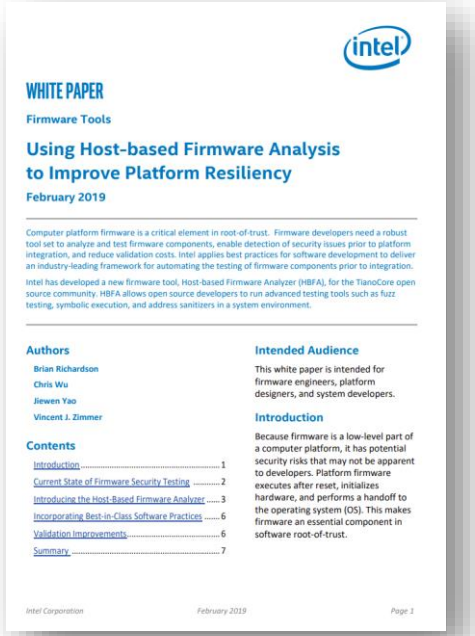
‡ UEFI Forum, 11/2019: https://uefi.org/node/4020

HBFA: Tianocore edk2-staging branch

Host-based Firmware Analyzer (HBFA)

- Contributed to edk2-staging branch by Intel in April 2019†
- Enables fuzzing/testing of UEFI drivers and Platform Initialization (PI) drivers
- Integrates several fuzzers & features
  - AFL, LibFuzzer, Peach
  - Simple GUI
  - Fault injection, KLEE/STP, Code coverage
  - Several unit/fuzz test cases included

- Original whitepaper 2019‡



WHITE PAPER

Firmware Tools

Using Host-based Firmware Analysis to Improve Platform Resiliency

February 2019

Computer platform firmware is a critical element in root-of-trust. Firmware developers need a robust tool set to analyze and test firmware components, enable detection of security issues prior to platform integration, and reduce validation costs. Intel applies best practices for software development to deliver an industry-leading framework for automating the testing of firmware components prior to integration.

Intel has developed a new firmware tool, Host-based Firmware Analyzer (HBFA), for the TianoCore open source community. HBFA allows open source developers to run advanced testing tools such as fuzz testing, symbolic execution, and address sanitizers in a system environment.

Authors

Brian Richardson
Chris Wu
Jiewen Yao
Vincent J. Zimmer

Intended Audience

This white paper is intended for firmware engineers, platform designers, and system developers.

Introduction

Because firmware is a low-level part of a computer platform, it has potential security risks that may not be apparent to developers. Platform firmware executes after reset, initializes hardware, and performs a handoff to the operating system (OS). This makes firmware an essential component in software root-of-trust.

Contents

† https://github.com/tianocore/tianocore.github.io/wiki/Host-Based-Firmware-Analyzer

‡ https://www.intel.com/content/dam/develop/external/us/en/documents/intel-usinghbfatoimproveplatformresiliency-820238.pdf

Motivation – Goals for Enhancing HBFA

Update, Improve, Enhance, and Reduce

- Need to support fuzzing efforts for UEFI at Intel
  - CI/CD integration is essential
- Original tool required a lot of manual setup to use
  - Automate and enhance useability
- Update to support
  - Leverage additional sanitizer options
- Add new features
- Focus not on GUI or Windows support
- Focus on:
  - Linux environment, CLI use
  - Fuzzing with AFL and LibFuzzer
- Upstream to community, ready code-base to enable OSS-Fuzz
- Find bugs ;)

Current efforts and features for HBFA-FL

Current focus/state

- Now works with newer versions: Clang/LibFuzzer
- Additional/finer granularity on selection of sanitizers (ASAN, MSAN, UBSAN)
- Updates to coverage data (LCOV and LLVM-tool output)
- Several of original HBFA unit/fuzzing test-cases build and working (**some fixes**)
- All new **documentation and training** materials
  - See: https://intel.github.io/HBFA-FL/src/index.html
- Intel Open-source Repository HBFA-FL: https://github.com/intel/HBFA-FL/tree/main
- Supports fuzzing of EDK2 per OSS-Fuzz: https://github.com/google/oss-fuzz/tree/master/projects/edk2

HBFA-FL: General Workflow

1. Build a suitable environment for EDK2 and HBFA-FL

- Easy approach: leverage container build from Tianocore:
  - https://github.com/tianocore/containers
  - E.g. ghcr.io/tianocore/containers/ubuntu-22-build
- Follow steps from our documentation:
  - https://github.com/intel/HBFA-FL/blob/main/docs/src/setup/linux.md
  - Take note of setting-up path and environmental variables

```
steps:
  - uses: actions/checkout@b4ffde65f46336ab88eb53be808477a3936bae11 # v4.1.1
  - name: Install Dependencies
    run: |
      sudo apt-get -y update && \
      sudo apt-get -y install \
        clang libclang-dev llvm
  - name: Retrieve and build EDK2
    run: |
      git clone https://github.com/tianocore/edk2.git --recursive && \
      pushd edk2 && make -C BaseTools && \
      source edksetup.sh && popd && \
      export -p > envsave
  - name: Setting up HBFA-FL and Build Environment
    run: |
      source envsave && \
      export WORKSPACE=$(pwd)/ && \
      export PACKAGES_PATH=$WORKSPACE/edk2:$WORKSPACE/HBFA/ && \
      python3 HBFA/UefiHostTestTools/HBFAEnvSetup.py && \
      export -p > envsave
  - name: Install AFL-2.52b
    run: |
      source envsave && \
      wget -q http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz && \
      tar xf afl-latest.tgz && rm afl-latest.tgz && \
      export AFL_PATH=$WORKSPACE/afl-2.52b && \
      export PATH=$PATH:$AFL_PATH && \
      export -p > envsave && \
      cd afl-2.52b && make && cd ..
  - name: Build Fuzzing Harnesses
    run: |
      source envsave && \
      cp HBFA/UefiHostFuzzTestPkg/Conf/build_rule.txt edk2/Conf/build_rule.txt && \
      cp HBFA/UefiHostFuzzTestPkg/Conf/tools_def.txt edk2/Conf/tools_def.txt && \
      build -p HBFA/UefiHostFuzzTestCasePkg/UefiHostFuzzTestCasePkg.dsc -a X64 -t AFL && \
      build -p HBFA/UefiHostFuzzTestCasePkg/UefiHostFuzzTestCasePkg.dsc -a X64 -t LIBFUZZER
```

HBFA-FL: General Workflow (continued)

2. Select or develop fuzzing harness & input corpus

- Option 1 – Use Existing Fuzzing Test Harness:
  - Original HBFA fuzzing test cases and corresponding test corpus included
  - See:

    https://github.com/intel/HBFA-FL/blob/main/docs/src/harness/includedfuzzharnesses.md

Fuzzing Test Case Name
TestTpm2CommandLib
TestBmpSupportLib
TestPartition
TestUdf
TestPeiUsb
TestVariableSmm
TestFmpAuthenticationLibPkcs7
TestFmpAuthenticationLibRsa2048Sha256
TestCapsulePei
TestFileName
TestTcg2MeasureGptTable
TestTcg2MeasurePeImage
TestVirtioPciDevice
TestVirtio10Blk
TestVirtioBlk
TestVirtioBlkReadWrite

Case Name	Seed Location (based from repository root)
TestTpm2CommandLib	HBFA/UefiHostFuzzTestCasePkg/Seed/TPM/Raw
TestBmpSupportLib	HBFA/UefiHostFuzzTestCasePkg/Seed/BMP/Raw
TestPartition	HBFA/UefiHostFuzzTestCasePkg/Seed/UDF/Raw/Partition
TestUdf	HBFA/UefiHostFuzzTestCasePkg/Seed/UDF/Raw/FileSystem
TestPeiUsb	HBFA/UefiHostFuzzTestCasePkg/Seed/USB/Raw
TestDxeCapsuleLibFmp	HBFA/UefiHostFuzzTestCasePkg/Seed/Capsule
TestVariableSmm	HBFA/UefiHostFuzzTestCasePkg/Seed/VariableSmm/Raw
TestFmpAuthenticationLibPkcs7	HBFA/UefiHostFuzzTestCasePkg/Seed/Capsule
TestFmpAuthenticationLibRsa2048Sha256	HBFA/UefiHostFuzzTestCasePkg/Seed/Capsule
TestCapsulePei	HBFA/UefiHostFuzzTestCasePkg/Seed/Capsule
TestUpdateLockBoxFuzzLength	HBFA/UefiHostFuzzTestCasePkg/Seed/LockBox/Raw
TestUpdateLockBoxFuzzOffset	HBFA/UefiHostFuzzTestCasePkg/Seed/LockBox/Raw
TestFileName	HBFA/UefiHostFuzzTestCasePkg/Seed/UDF/Raw/FileName

HBFA-FL: General Workflow (continued)

2. Select or develop fuzzing harness & input corpus

- Option 2 - Developing a fuzzing test harness
  - Need to write test harness:
    - C-source-code module and associated module description file (.inf)
    - Plumb in reference to the module in description file used by HBFA platform
  - See our tutorial (full-walkthrough): https://github.com/intel/HBFA-FL/blob/main/docs/src/tutorials/writingafuzzingharness.md
- Note, one may need to stub-out (e.g. mimic responses from hardware)
  - Several stub-libraries are included

```
VOID
FixBuffer (
  UINT8                 *TestBuffer
  )
{
}

UINTN
EFIAPI
GetMaxBufferSize (
  VOID
  )
{
  return TOTAL_SIZE;
}

VOID
EFIAPI
RunTestHarness(
  IN VOID  *TestBuffer,
  IN UINTN TestBufferSize
  )
{
  EFI_GRAPHICS_OUTPUT_BLT_PIXEL  *GopBlt;
  UINTN                          GopBltSize;
  UINTN                          PixelHeight;
  UINTN                          PixelWidth;
  FixBuffer (TestBuffer);
  GopBlt = NULL;
  TranslateBmpToGopBlt(
            TestBuffer,
            TestBufferSize,
            &GopBlt,
            &GopBltSize,
            &PixelHeight,
            &PixelWidth
            );
  if (GopBlt != NULL)
    FreePool (GopBlt);
}
```

```
## @file
# Component description file for TestBmpSupportLib module.
#
# Copyright (c) 2018, Intel Corporation. All rights reserved.<BR>
# SPDX-License-Identifier: BSD-2-Clause-Patent
#
##

[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = TestBmpSupportLib
  FILE_GUID                      = E911AB26-4741-4621-93EF-305FEA98A851
  MODULE_TYPE                    = USER_DEFINED
  VERSION_STRING                 = 1.0

#
# The following information is for reference only and not required by the build tools.
#
#  VALID_ARCHITECTURES           = IA32 X64
#

[Sources]
  TestBmpSupportLib.c

[Packages]
  MdePkg/MdePkg.dec
  MdeModulePkg/MdeModulePkg.dec
  UefiHostTestPkg/UefiHostTestPkg.dec


[LibraryClasses]
  BaseLib
  BaseMemoryLib
  MemoryAllocationLib
  DebugLib
  SafeIntLib
  BmpSupportLib
  ToolChainHarnessLib
```

HBFA-FL: General Workflow (continued)

3. Building and running fuzzing test-cases

A. EDK2 Build System

Filename	Description
/root/hbfa_workspace/edk2/BaseTools/BinWrappers/PosixLike/build	When invoking 'build' from the CLI in HBFA, this Bash script is ran and acts as a wrapper to invoke a Python-based build script 'build.py' for the HBFA environment in this Docker image.
/root/hbfa_workspace/edk2/BaseTools/Source/Python/build/build.py	Primary script used to orchestrate building a platform or a module for EDKII

When building a test module in HBFA, an invokation of the 'build' comand may be done similar to that shown in the following.

Platform

Test-case

```
build -p UefiHostFuzzTestCasePkg/UefiHostFuzzTestCasePkg.dsc -m
UefiHostFuzzTestCasePkg/TestCase/MdeModulePkg/Library/BaseBmpSupportLib/TestBmpSupportLib.inf -a X64 -b DEBUG -t AFL --conf /root/hbfa_workspace/edk2-
staging/HBFA/UefiHostFuzzTestPkg/Conf -t GCC5
```

The 'build' script has many options and features (e.g. see the output from `build -h`). Some useful flags used (and available in the HBFA environment are)

Build CLI option	Purpose	Notes/Available options
-p	To specify the platform (.dsc) file name	In this case, the platform should be the HBFA, UefiHostFuzzTestCasePkg.dsc file.
-m	To specify the test case the module specified by the INF file name argument	This should point to the test module file you have created and with to build/fuzz (or a pre-built testcase)
-a	To specify the target architecture	Per HBFA documentation, only 'X64' is supported for LibFuzzer. Use of 'IA32' is ok for AFL in HBFA.
-t	This is used to specify the toolchain	Note multiple targets can be specified (e.g. `-t AFL -t GCC5`)
--conf	the customized Conf directory	For HBFA, this should be set as '/root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestPkg/Conf'

HBFA-FL: General Workflow (continued)

3. Building and running fuzzing test-cases

B. **RunAFL.py** or RunLibFuzzer.py (recommended approach)

-a	Architecture: X64/IA32
-m	Test case module
-i	Seed corpus directory
-o	output directory

```
bash-5.1# RunAFL.py -a X64 -m /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/TestCase/MdeModulePkg/Library/BaseBmpSupportLib/TestBmpSupportLib.inf -i /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/Seed/BMP/Raw -o /tmp/fuzz_RunAFL_TestBmpSupportLib
Start build Test Module:
build -p UefiHostFuzzTestCasePkg/UefiHostFuzzTestCasePkg.dsc -m
UefiHostFuzzTestCasePkg/TestCase/MdeModulePkg/Library/BaseBmpSupportLib/TestBmpSupportLib.inf -a X64 -b DEBUG -t AFL --conf /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestPkg/Conf -t GCC5
Build Successfully !!!

Start run AFL test:
afl-fuzz -i /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/Seed/BMP/Raw -o /root/hbfa_workspace/tmp/fuzz_RunAFL_TestBmpSupportLib
/root/hbfa_workspace/Build/UefiHostFuzzTestCasePkg/DEBUG_AFL/X64/TestBmpSupportLib @@
```

HBFA-FL: General Workflow (continued)

3. Building and running fuzzing test-cases

B. RunAFL.py or **RunLibFuzzer.py** (recommended approach)

- BONUS: Supports additional options

-a	Architecture: X64/IA32
-m	Test case module
-i	Seed corpus directory
-o	output directory

```
bash-5.1# RunLibFuzzer.py -a X64 -m /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/TestCase/MdeModulePkg/Library/BaseBmpSupportLib/TestBmpSupportLib.inf -i
/root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/Seed/BMP/Raw -o /tmp/fuzz_RunLibFuzzer_TestBmpSupportLib
LibFuzzer output will be generated in current directory:/tmp/fuzz_RunLibFuzzer_TestBmpSupportLib
Start build Test Module:
build -p UefiHostFuzzTestCasePkg/UefiHostFuzzTestCasePkg.dsc -m UefiHostFuzzTestCasePkg/TestCase/MdeModulePkg/Library/BaseBmpSupportLib/TestBmpSupportLib.inf -a X64 -b
DEBUG -t LIBFUZZER --conf /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestPkg/Conf
Build Successfully !!!

Start run LibFuzzer test:
/root/hbfa_workspace/Build/UefiHostFuzzTestCasePkg/DEBUG_LIBFUZZER/X64/TestBmpSupportLib /root/hbfa_workspace/edk2-staging/HBFA/UefiHostFuzzTestCasePkg/Seed/BMP/Raw -
rss_limit_mb=0 -artifact_prefix=/tmp/fuzz_RunLibFuzzer_TestBmpSupportLib/
```

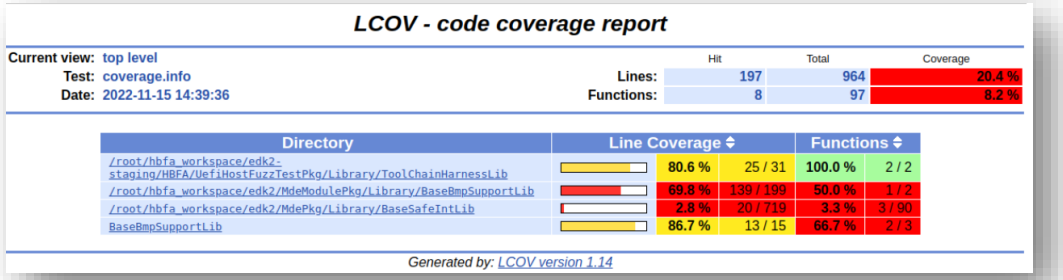
Other options include: “-s” for sanitizers (ASAN, MSAN, UBSAN) and “-p” if Profraw code coverage desired

HBFA-FL: General Workflow (continued)

4. Generate Fuzzing and Code Coverage Reports

Fuzzer	Coverage Format	Information generated/provided
AFL	GCOV	a summary of the AFL fuzzing session and gcov-based coverage data for the fuzzing run, in a HTML format
LibFuzzer	GCOV	a summary of the LibFuzzer fuzzing session and gcov-based coverage data for the fuzzing run, in a HTML format
LibFuzzer	PROFRAW	a summary of the LibFuzzer fuzzing session and source-based coverage data for the fuzzing run, in a text and HTML format

- Three steps for Code Coverage:
  1. Run the normal fuzzing case (e.g. RunAFL.py)
  2. Run ReportMain.py
  3. Run GenCodeCoverage.py



Bugs/Vulns/Issues Found

Fuzzing runs with HBFA-FL

- Fuzzing campaigns (2 x 2-months) and ongoing
- Initial campaign:
  - ~200 crashes distilled down to ~12 unique crashes/issues
  - 6 in EDK2 source
  - 6 in HBFA code (harnesses/support code)
  - Some bugs found with help of sanitizers (ASAN, UBSAN)
- Reported to Tianocore/EDK2
  - 3 new security bugs
    - 1 additional, found to be duplicate with one reported
    - 1 vulnerability resulting in code execution
    - Other 2 issues: a SIGSEG (read offset from Null page) and an integer overflow
  - 1 bug (Tianocore Bugzilla 4383)
    - check on max endpoints in Pei USB device descriptor only protected by assert
    - discovered by triggering assert fuzzing, then code review
  - 1 additional already reported (heap overflow)
    - Prior report over a year before our discovery
- OSS-Fuzz: Ongoing

Questions (?)

Ideas, Comments, Suggestions, …

- Intel Open-source Repository HBFA-FL:
  - https://github.com/intel/HBFA-FL/tree/main
- Email:
  - earl.lynn.tipton@intel.com


intel