Debloating Address Sanitizer

Yuchen Zhang, *Stevens Institute of Technology;* Chengbin Pang, *Nanjing University;* Georgios Portokalidis, Nikos Triandopoulos, and Jun Xu, *Stevens Institute of Technology*

https://www.usenix.org/conference/usenixsecurity22/presentation/zhang-yuchen

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

This artifact provides the source code of ASan--, a tool assembling a group of optimizations to reduce (or "debloat") sanitizer checks and improve ASan's efficiency. It also provides a set of test cases and necessary dependencies. In principle, ASan-- has no special requirment of the hardware or operating system. For reproductive experiments, we recommend reviewers to build ASan-- on Ubuntu 18.04 LTS 64bit (a virtual machine is fine) and we suggest reviewers to install the desktop version of Ubuntu for Chromium evaluation. For software requirements, we separate into two parts. For runtime evaluation, we used SPEC CPU2006 and Chromium to test the performance of our tool. For bug detection capability evaluation, we utilized Juliet Test Suite and Linux Flaw Project to detect the vulnerabilities in software, and ASan-- should achieve identical results with ASan.

A.2 Artifact check-list (meta-information)

- **Algorithm:**
  Yes, we present six new algorithms. Details can be found in our paper.
- **Program:**
  SPEC CPU2006: is private, and it should be downloaded. The approximate size is 20GB.
  Chromium: is public, and version 58.0.3003.0 should be downloaded. The approximate size is 50GB.
  Juliet Test Suite: is public, and it is included in our artifact. The approximate size is 500MB.
  Linux Flaw Project: is public, and it is included in our artifact. The approximate size is 1.5GB.
- **Compilation:**
  The compiler we used is LLVM-4.0.0, which is public, and it is included in our artifact.
- **Run-time environment:**
  We recommend Ubuntu 18.04 LTS 64bit for testing, and you will need root access.
- **Execution:**
  The experiments will approximately run 5 hours.
- **Metrics:**
  Execution time and Vulnerabilities reproduction.
- **Output:**
  The outputs will be numerical results and error logs.
- **Experiments:**
  We provided both bash scripts and manual steps for users to reproduce results.
- **How much disk space required (approximately):**
  Approximately 100GB.
- **How much time is needed to prepare workflow (approximately):**
  The estimate time to prepare workflow will be 8 hours.
- **How much time is needed to complete experiments (approximately):**
  The estimate time to complete experiments will be 15 hours.
- **Publicly available:**
  Yes, our artifact is publicly available.
- **Archived (provide DOI or stable reference):**
  Yes, our artifact is publicly available on GitHub.

A.3 Description

A.3.1 How to access

Clone below repository from GitHub
https://github.com/junxzm1990/ASAN--/tree/f497310328fafddc7fe7993edb8befd4ab4d6393

A.3.2 Hardware dependencies

The approximate disk space required after unpacking our artifact is 100GB.

A.3.3 Software dependencies

For the OS, we recommend Ubuntu 18.04 LTS 64bit and we suggest reviewers to install the desktop version of Ubuntu. For software dependencies, we separate into two parts. For run-time evaluation, we used SPEC CPU2006 and Chromium. For bug detection capability evaluation, we utilized Juliet Test Suite and Linux Flaw Project.

A.4 Installation

You can get the artifacts from GitHub using the following command:

```
git clone https://github.com/junxzm1990/ASan--.git
```

A.5 Experiment workflow

The overall workflow consists of the following steps:

1. Install the dependencies;
2. Build tool ASan--;
3. Build and run the SPEC CPU2006 benchmarks;
4. Build and run the Chromium benchmarks;
5. Build and run the Juliet Test Suite benchmarks;
6. Build and run the Linux Flaw Project benchmarks.

We have provided scripts for each of the steps above.

A.6 Evaluation and expected results

We will go through the entire experiment workflow by describing all the commands in each step.

A.6.1 Install the dependencies

Run the following commands:

```
sudo apt-get install cmake
sudo apt-get install git
sudo apt-get install wget
sudo apt-get install tar
```

A.6.2 Build tool ASan--

Run the following commands:

```
git clone https://github.com/junxzm1990/ASan--.git
cd ASan--
cd llvm-4.0.0-project
mkdir ASan--Build && cd ASan--Build
cmake -DLLVM_ENABLE_PROJECTS="clang;compiler-rt" -G
"Unix Makefiles" ../llvm
make -j
```

A.6.3 Build fuzzing version ASan--

Run the following commands:

```
patch -p1 < patch_ASanASan--
cd llvm-4.0.0-project
mkdir ASanASan--Build && cd ASanASan--Build
cmake -DLLVM_ENABLE_PROJECTS="clang;compiler-rt" -G
"Unix Makefiles" ../llvm
make -j
```

A.6.4 Build and run the AFL Fuzzing

Run the following commands:

```
cd /fuzzing
bash set_ASan--.sh
cd binutils-2.32
bash auto_build_ASan--.sh
./afl-2.52b/afl-fuzz -S nm_afl -i
./afl-2.52b/testcases/others/elf/ -o ./eval/nm -m
none - ./binutils-2.32/ASan_Srk/binutils/nm-new @@
```

The result will be printed on the console.

A.6.5 Build and run the SPEC CPU2006

Run the following commands:

```
cd /spec
cp run_asan--.sh /cpu2006
cd cpu2006
CC=PATH/llvm-4.0.0-project/ASan--Build/bin/clang
CXX=PATH/llvm-4.0.0-project/ASan--Build/bin/clang++
./run_asan--.sh asan-- <test|train|ref> <int|fp>
```

The result will be printed on the console.

A.6.6 Build and run the Chromium

Run the following commands:

```
cd /chromium
git clone https://chromium.googlesource.com
/chromium/tools/depot_tools.git
```

Add depot_tools to the end of your PATH :

```
export PATH="$PATH:/path/to/depot_tools"
```

Create a chromium directory for the checkout and change to it

```
mkdir  /chromium && cd  /chromium
```

Run the fetch tool from depot_tools to check out the code and its dependencies.

```
fetch -nohooks chromium
git checkout tags/58.0.3003.0 -b 58
```

Check out a version of depot_tools from around the same time as the target revision.

```
# Get date of current revision:
/chromium/src $ COMMIT_DATE=$(git log -n 1
-pretty=format:%ci)
# Check out depot_tools revision from the same time:
/depot_tools $ git checkout $(git rev-list -n 1
-before="$COMMIT_DATE" <main | master>
/depot_tools $ export DEPOT_TOOLS_UPDATE=0
```

Checkout all the submodules at their branch DEPS revisions.

```
gclient sync -D -force -reset -with_branch_heads
```

To create a build directory, run:

```
gn args out/ASan--
```

Set build arguments.

```
is_clang = true
clang_base_path = "llvm-4.0.0-project/ASan--Build"
is_asan = true
is_debug = ture
symbol_level = 1
is_component_build = true
pdf_use_skia = true
```

Build Chromium (the “chrome” target) with Ninja:

```
ninja -C out/ASan-- chrome
```

Run benchmarks and Reproduce bugs:

```
# Sunspider:
./chrome https://webkit.org/perf/sunspider-0.9.1/
sunspider-0.9.1/driver.html
# Kraken:
./chrome https://mozilla.github.io/krakenbenchmark.
mozilla.org/index.html
# Lite Brite:
./chrome https://testdrive-archive.azurewebsites.net/
Performance/LiteBrite/
# Octane:
./chrome https://chromium.github.io/octane/
# Basemark:
./chrome https://web.basemark.com/
# WebXPRT:
./chrome https://www.principledtechnologies.com/
benchmarkxprt/webxprt/run-webxprt-mobile
```

#Issue 848914:

```
./chrome -disable-gpu /Issue_848914_PoC/gpu_freeids.html
```

#Issue 1116869:
```
./chrome /Issue_1116869_PoC/poc_heap_buffer_overflow
```
#Issue 1099446:
```
./chrome /Issue_1099446_PoC/poc_heap_buffer_overflow
```
The result will be printed on the console.

A.6.7 Build and run the Juliet Test Suite

Juliet contains different benchmarks, here we take CWE121 as an example. Run the following commands:
```
cd /juliet_test_suite
cd testcases
cd CWE121_Stack_Based_Buffer_Overflow
cd s01
make -j
export ASAN_OPTIONS=halt_on_error=0
./CWE121_s01
```
The result will be printed on the console.

A.6.8 Build and run the Linux Flaw Project

Linux Flaw contains different benchmarks, here we take CVE-2006-0539 as an example. Run the following commands:
```
cd /linux_flaw_project
```

Install the dependencies.
```
sudo apt-get install sendmail
sudo apt-get install vim
sudo apt-get install pkg-config
sudo apt-get install fontconfig
sudo apt-get install libfontconfig1-dev
export CC=$(readlink -f ../../llvm-4.0.0-project
/ASan--Build/bin/clang)
CXX=$(readlink -f ../../llvm-4.0.0-project
/ASan--Build/bin/clang++)
```

Build and run.
```
wget https://github.com/mudongliang/source-packages/
raw/master/CVE-2006-0539/fcron-3.0.0.src.tar.gz
tar -xvf fcron-3.0.0.src.tar.gz
cp configure ./fcron-3.0.0
cd fcron-3.0.0
sudo mkdir /var/spool/fcron
CC=$CC CXX=$CXX CFLAGS="-fsanitize=address -g"
CXXFLAGS="-fsanitize=address -g" ./configure
make -j
./convert-fcrontab `perl -e 'print "pi3"x600'`
```
The result will be printed on the console.