ИСП РАН

Futag

Руководство пользователя

Содержание

I. Описание инструмента Futag

Futag – инструмент автоматической генерации фаззинг-оберток для функций, используемых в программных библиотеках, фаззинга сгенерированных оберток и обнаружения ошибок. Инструмент Futag предоставляет первичный анализ тестируемой библиотеки и не требует от пользователя особенных знаний об этой библиотеке.

Требования к системе:

- для системы Ubuntu 18.04, нужно установить следующие пакеты:

```
apt-utils libncurses5 libgcc-5-dev make gdb binutils fuse
```

- для системы Ubuntu 20.04, нужно установить следующие пакеты:

```
apt-utils libncurses5 gcc-multilib g++ make gdb binutils fuse
```

Структура пакета:

- futag/bin: содержит все компоненты, в том числе:
  1. Futag-gen4lib: инструмент для генерации фаззинг-оберток
  2. clang и инструменты llvm

- futag/futag: скрипты и шаблоны для запуска и сборки результатов работы инструмента

II. Подготовка к запуску

В качестве примера работы инструмента описан процесс фаззинга библиотеки curl версии 7.79.1. Для работы необходимо:

- установить необходимые пакеты для библиотеки curl:

libssl-dev zlib1g-dev wget libpsl-dev libgsasl7-dev libldap-dev

- скопировать Futag в домашнюю папку /home/futag/.

```
~$ cp -r futag /home/futag/
```

- скачать библиотеку curl версии 7.79.1 по ссылке: https://github.com/curl/curl/releases/download/curl-7_79_1/curl-7.79.1.tar.gz
- распаковать архив
- создавать папки для сборки библиотеки

```
~$ wget https://github.com/curl/curl/releases/download/curl-7_79_1/curl-7.79.1.tar.gz
~$ tar -xf curl-7.79.1.tar.gz
~$ cd curl-7.79.1/
~/curl-7.79.1$ mkdir build
~/curl-7.79.1$ mkdir build/local-install
~/curl-7.79.1$ cd build
~/curl-7.79.1/build$
```

- для того, чтобы получить подробную информацию при фаззинге, нужно скомпилировать библиотеку с санитайзерами и установить библиотеку в локальную папку

```
~/curl-7.79.1/build$ ../configure --with-openssl --prefix=/home/futag/curl-7.79.1/build/local-install CXX=/home/futag/futag/bin/clang++ CC=/home/futag/futag/bin/clang CFLAGS="-fsanitize=address -fprofile-instr-generate -fcoverage-mapping -g -O0" CXXFLAGS="-fsanitize=address -fprofile-instr-generate -fcoverage-mapping -g -O0" LDFLAGS="-fsanitize=address -g -O0"
~/curl-7.79.1/build$ make && make install
```

В результате библиотека будет установлена в папку /home/futag/curl-7.79.1/build/local-install.

Папка /home/futag/curl-7.79.1/build/local-install/include содержит все хедеры библиотеки curl.

Папка /home/futag/curl-7.79.1/build/local-install/lib содержит все статические и динамические библиотеки curl.

III. Запуск инструмента

Запустить скрипт futag-run внутри папки futag/futag-tools используется для того, чтобы автоматически генерировать обертки для функций библиотеки curl, и собрать результаты фаззинга. При запуске необходимо указать:

```
~$ cd curl-7.79.1/build/local-install
~/curl-7.79.1/build/local-install $ ~/futag/futag-tools/futag-run  -p ~/futag
-i ". include include/curl" -a lib -o targetscurl  -s "-lgsasl -lpsl -lssl -
lcrypto -lssl -lcrypto -lldap -llber -lz" -tt 300 -f 4 -m 8000
include/curl/curl.h
```

- ~/futag/futag-tools/futag-run
  скрип для автоматической генерации оберток, компиляции, фаззинга и сборки результатов
- -p ~/futag/
  опция -p указывает путь до папки futag
- -i ". include include/curl/"
  опция -i указывает какие пути нужно включить в процессе компиляции
- -a lib
  опция -a указывает путь, который содержит архив статических файлов тестируемой библиотеки (файлы с расширением **.a**)
- -o test-curl
  опция -o указывает папку, в которую будут создаваться все обертки
- -s "-lgsasl -lpsl -lssl -lcrypto -lssl -lcrypto -lldap -llber -lz "
  опция -s указывает какие системные библиотеки нужно включить в процессе компиляции.
- -tt 300
  Общее время для фаззинга одной обертки – 300 секунд
- -f 4
  Количество процессов при фаззингу
- -m 8000
  Максимальная память, используемая при фаззингу

При необходимости можно указывать еще опции:

- -gdb, --gdb_debug
  Опция для отладки с GDB
- -d, --debug
  Опция для просмотра информации отладки
- include/curl/curl.h
  заголовочный файл, для функций которого генерируются все фаззинг-оберток

Результат фаззинга записывается в файл **futag.svres**.

IV. Готовые скрипты

Папка Docker внутри папки docs содержит скрипты для автоматического запуска Futag с докером.

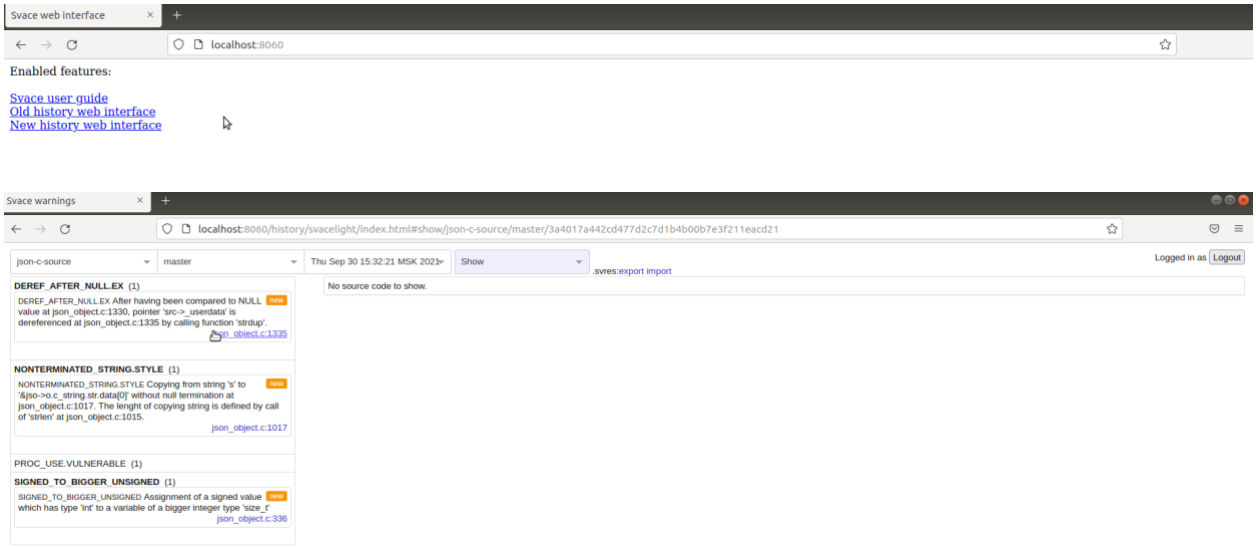
- build-docker: скрипт для построения докер-контейнера
- run-docker: скрипт для запуска докер-контейнера
- get_result: скрипт для получения в текущую папку файл результата после фаззинга (файл futag.svres)

V. Анализ результатов с помощью Svace

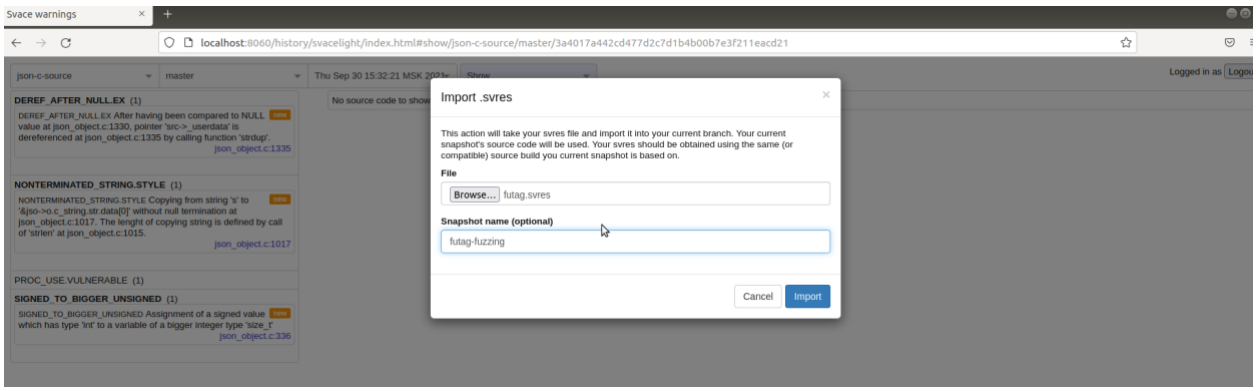
Для того, чтобы анализировать с помощью инструмента Svace, необходимо запустить в папке **build**:

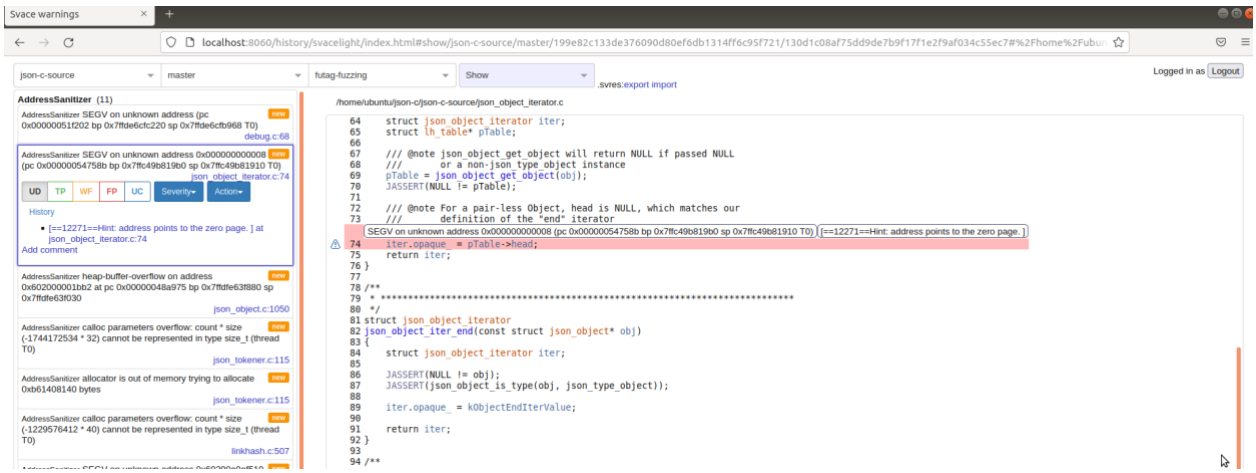
```
~/curl-7.79.1/build $ make clean
~/curl-7.79.1/build $ ~/svace/bin/svace init
~/curl-7.79.1/build $ ~/svace/bin/svace build --force-debug-info make
~/curl-7.79.1/build $ ~/svace/bin/svace analyze
~/curl-7.79.1/build $ ~/svace/bin/svace history import
~/curl-7.79.1/build $ ~/svace/bin/svace server single-start
You are starting Svace history server in single-project mode. In this mode
server accepts any credentials and gives unlimited access to one selected
project to anyone who can access your computer via network.
Remote analysis server started at port 55135
Web server started at port 8060
Press Ctrl+C to interrupt the process
```

Svace запустит локальный веб-сервер по адресу http://localhost:8060.



Далее результаты работы Futag импортируются в Svace:




Svace warnings
localhost:8060/history/svacelight/index.html#show/json-c-source/master/199e82c133de376090d80ef6db1314ff6c95f721/130d1c08af75dd9de7b9f17f1e2f9af034c55ec7#%2Fhome%2Fubun
json-c-source
master
futag-fuzzing
Show
.svres:export import
Logged in as Logout
AddressSanitizer (11)
AddressSanitizer SEGV on unknown address (pc 0x00000051f202 bp 0x7ffde6cfc220 sp 0x7ffde6cfb968 T0)
debug.c:68
AddressSanitizer SEGV on unknown address 0x000000000008 (pc 0x00000054758b bp 0x7ffc49b819b0 sp 0x7ffc49b81910 T0)
json_object_iterator.c:74
UD
TP
WF
FP
UC
Severity
Action
History
[==12271==Hint: address points to the zero page.] at json_object_iterator.c:74
Add comment
AddressSanitizer heap-buffer-overflow on address 0xb02000001bb2 at pc 0x00000048a975 bp 0x7ffdfe63f880 sp 0x7ffdfe63f030
json_object.c:1050
AddressSanitizer calloc parameters overflow: count * size (-1744172534 * 32) cannot be represented in type size_t (thread T0)
json_tokener.c:115
AddressSanitizer allocator is out of memory trying to allocate 0xb61408140 bytes
json_tokener.c:115
AddressSanitizer calloc parameters overflow: count * size (-1229576412 * 40) cannot be represented in type size_t (thread T0)
linkhash.c:507
/home/ubuntu/json-c/json-c-source/json_object_iterator.c
64 struct json_object_iterator iter;
65 struct lh_table* pTable;
66
67 /// @note json_object_get_object will return NULL if passed NULL
68 /// or a non-json_type_object instance
69 pTable = json_object_get_object(obj);
70 JASSERT(NULL != pTable);
71
72 /// @note For a pair-less Object, head is NULL, which matches our
73 /// definition of the "end" iterator
SEGV on unknown address 0x000000000008 (pc 0x00000054758b bp 0x7ffc49b819b0 sp 0x7ffc49b81910 T0) [==12271==Hint: address points to the zero page.]
74 iter.opaque_ = pTable->head;
75 return iter;
76 }
77
78 /**
79 * **
80 */
81 struct json_object_iterator
82 json_object_iter_end(const struct json_object* obj)
83 {
84 struct json_object_iterator iter;
85
86 JASSERT(NULL != obj);
87 JASSERT(json_object_is_type(obj, json_type_object));
88
89 iter.opaque_ = kObjectEndIterValue;
90
91 return iter;
92 }
93
94 /**