Testing Your Code for Security Issues With Automated Fuzzing

Albert **Astals Cid**

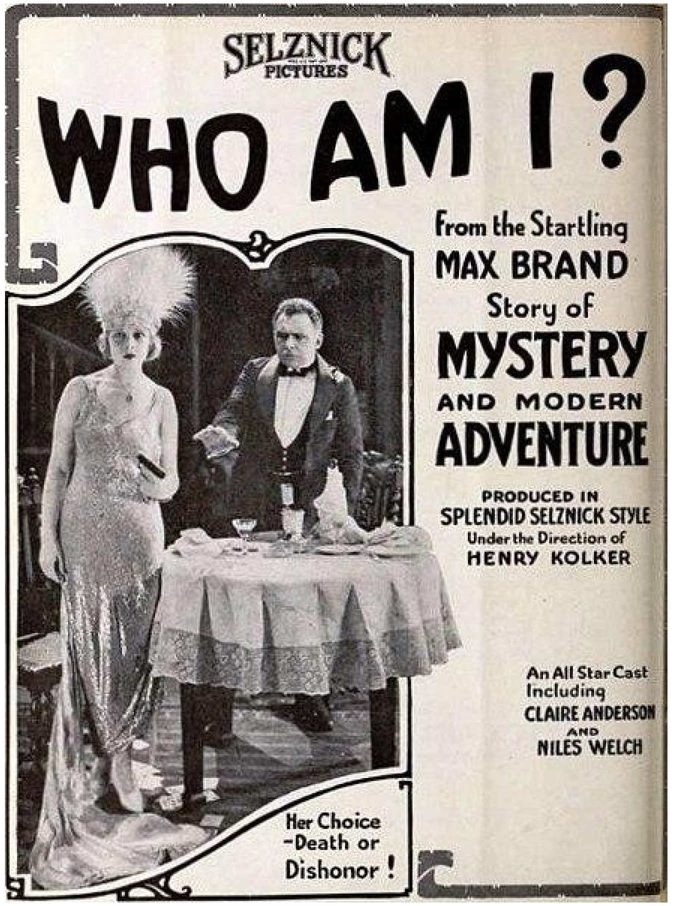
albert.astals.cid@kdab.com

Who is this?

Working with Qt since 2003

- KDE
- Canonical/Ubuntu Phone
- KDAB
- Contributed around 400 patches for Qt itself

NOT A SECURITY EXPERT


SELZNICK PICTURES
WHO AM I?
From the Startling MAX BRAND Story of MYSTERY AND MODERN ADVENTURE
PRODUCED IN SPLENDID SELZNICK STYLE
Under the Direction of HENRY KOLKER
An All Star Cast Including CLAIRE ANDERSON AND NILES WELCH
Her Choice -Death or Dishonor!

Security issues

Which security are we talking about?

The issues found by this kind of tools are generally related to wrong memory uses like uninitialized variables, using already freed memory or buffer overruns.

These errors typically mean that the application will behave incorrectly either by doing unexpected things or simply crashing.

People with enough experience are able to turn [some of] these memory related crashes into code execution exploits.

Which tools do we have?

- The Operating System

If your application uses memory incorrectly it will probably crash, making sure that doesn't happen is a good first step ;)

- Valgrind

Valgrind will help us find memory errors. Its main problem is the huge penalty paid regarding resource use

- ASAN/MSAN/UBSAN

The compiler sanitizers instrument the code at compile time. They have a functionality very similar to Valgrind but the resource usage is much smaller (though using them is from harder to way harder)

Fuzzing & oss-fuzz

What is fuzzing?

Fuzzing is a technique based in sending random/garbage values to a given application or function.

This way it tests the robustness of that code.

The most basic way is just calling a given binary with all possible inputs and make sure it doesn't explode.

```
echo "a" | pdfinfo -
echo "b" | pdfinfo -
echo "c" | pdfinfo -
…
echo "aa" | pdfinfo -
…
```

What to fuzz?

The most critical software to fuzz is the one exposed to the world.

Web-browser/e-mail client

Image reader

Your file-format parser

Your network service end-point

What is oss-fuzz?

oss-fuzz is a fuzzing engine developed by Google

[well the engine itself is called libFuzzer ;)].

It links with the code and is coverage based meaning it is able to learn and maximize coverage with the least number of “random” inputs.

```
void theFunction(int x) {
        if (x > 50) {
        } else {
        }
}
```

What is oss-fuzz? (II)

oss-fuzz is a set of docker images (with the last version of clang, libFuzzer, etc) and small test applications that exercise free software projects.

At this point there are around **240 projects**

https://github.com/google/oss-fuzz/tree/master/projects

What is oss-fuzz? (III)

oss-fuzz is a SAAS around libFuzzEngine + ASAN/MSAN/UBSAN + bug tracker

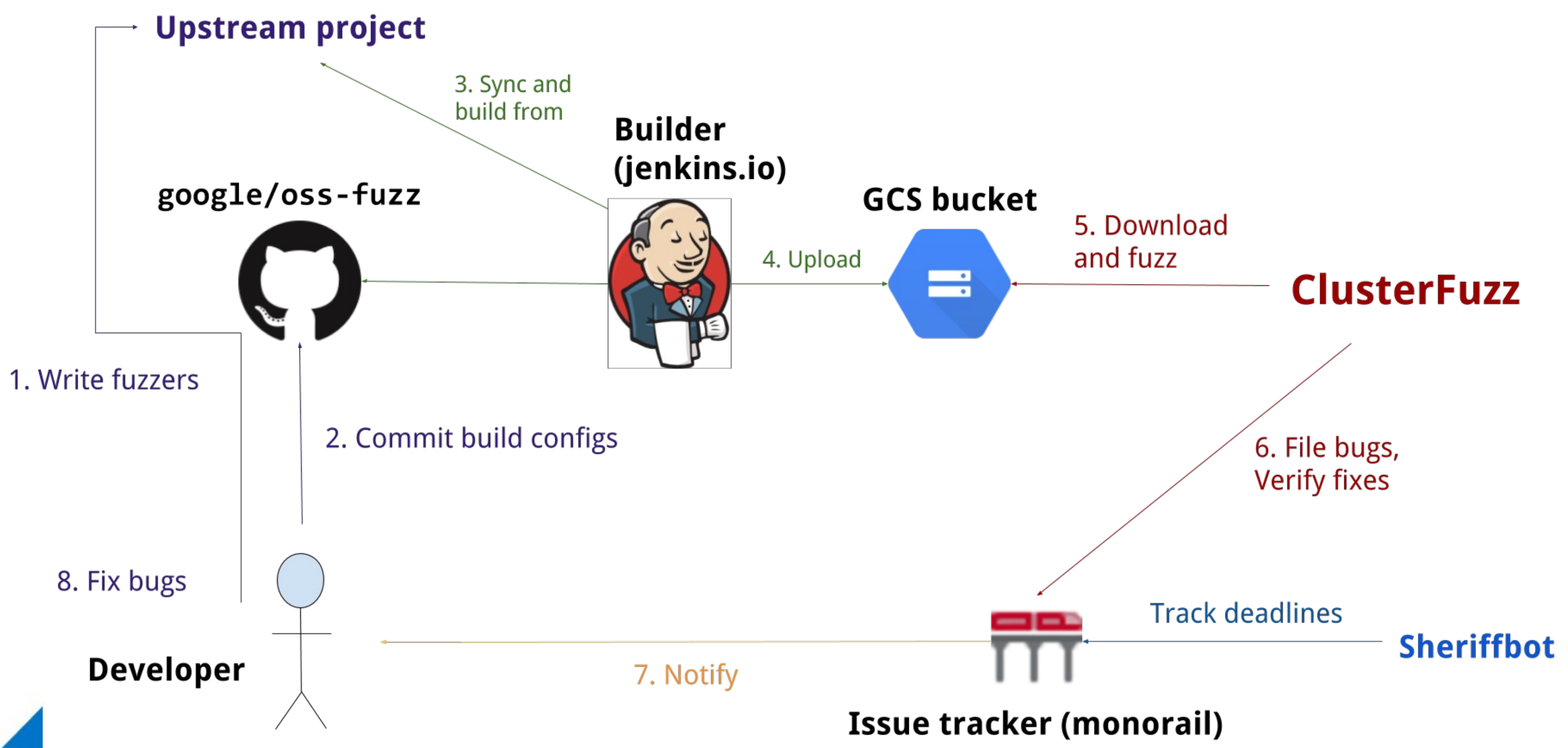
Strict policy on the bugs that are found:

- Mantainers are notified when issues are found
- Issue is made public:
  - **90 days** after being found

    or
  - **30 days** after being fixed

All the software to build the SAAS is free software in case you want to run one yourself

What is oss-fuzz? (IV)



Example

Simple fuzzer

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size)
{
    int argc = 0;
    QCoreApplication a(argc, nullptr);
    QImageIOHandler *handler = new TGAHandler();
    QImage i;
    QBuffer b;
    b.setData((const char *)data, size);
    b.open(QIODevice::ReadOnly);
    handler->setDevice(&b);
    handler->canRead();
    handler->read(&i);
    delete handler;
    return 0;
}
```

Simple Dockerfile

```
FROM gcr.io/oss-fuzz-base/base-builder
MAINTAINER your_email
RUN apt-get install --yes cmake
RUN git clone --depth 1 https://github.com/madler/zlib.git
RUN git clone --depth 1 https://github.com/nih-at/libzip.git
RUN git clone --depth 1 git://anongit.kde.org/extra-cmake-modules
RUN git clone --depth 1 --branch=5.15 git://code.qt.io/qt/qtbase.git
RUN git clone --depth 1 git://anongit.kde.org/kimageformats
COPY build.sh $SRC
COPY kimgio_fuzzer.cc $SRC
WORKDIR kimageformats
```

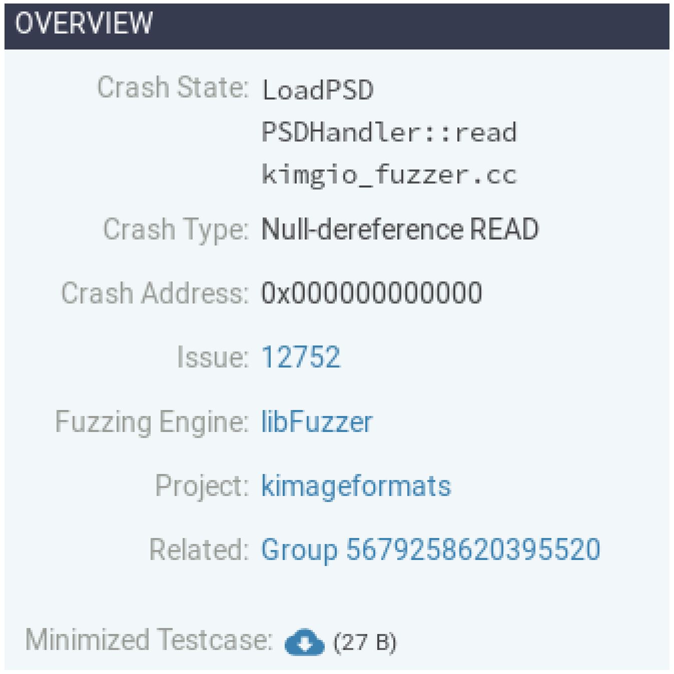
Simple build.sh

```
cd $SRC/zlib
./configure –static && make install -j$(nproc)

cd $SRC/libzip
cmake . -DBUILD_SHARED_LIBS=OFF && make install -j$(nproc)

cd $SRC/extra-cmake-modules
cmake . && make install -j$(nproc)

[build Qt]

cd $SRC/kimageformats
cmake . -DBUILD_SHARED_LIBS=OFF && make install -j$(nproc)

$CXX $CXXFLAGS -fPIC -std=c++11 $SRC/kimgio_fuzzer.cc -o $OUT/kimgio_tga_fuzzer ...

find . -name "*.tga" | zip -q $OUT/kimgio_tga_fuzzer_seed_corpus.zip -@
```

Maybe not so simple build.sh

```
cd $SRC

cd qtbase


# fix memory sanitizer build (3 sed lines right now)

sed -i -e "s/flags/other_flags/g"
mkspecs/linux-clang-libc++/qmake.conf


./configure --glib=no --libpng=qt -opensource -confirm-license -
static -no-opengl -no-icu -platform linux-clang-libc++ -v


cd src

../bin/qmake -o Makefile src.pro

make sub-gui -j$(nproc)
```

How does it look like?



OVERVIEW

Crash State:	`LoadPSD`
	`PSDHandler::read`
	`kimgio_fuzzer.cc`

Crash Type:	Null-dereference READ
Crash Address:	0x000000000000
Issue:	12752
Fuzzing Engine:	libFuzzer
Project:	kimageformats
Related:	Group 5679258620395520

Minimized Testcase: (27 B)

How does it look like? (II)

```
==1==ERROR: AddressSanitizer: SEGV on unknown address
0x000000000000 (pc 0x00000054f266 bp 0x7ffef040f0d0 sp
0x7ffef040f000 T0)

==1==The signal is caused by a READ memory access.

==1==Hint: address points to the zero page.SCARINESS: 10 (null-
deref)

    #0 0x54f265 in (anonymous namespace)::LoadPSD(QDataStream&,
(anonymous namespace)::PSDHeader const&, QImage&)
/src/kimageformats/src/imageformats/psd.cpp:206:51

    #1 0x54e90e in PSDHandler::read(QImage*)
/src/kimageformats/src/imageformats/psd.cpp:255:10

    #2 0x53a18f in LLVMFuzzerTestOneInput
/src/kimgio_fuzzer.cc:60:12
```

What about Qt?

Qt and oss-fuzz

Robert Löhning from The Qt Company has been working on it.

What we have so far:

- Fuzzing Qt with libFuzzer locally
- Configure switch for adding coverage info (since 5.13)
- Some fuzzers in qtbase, qtdeclarative, qtsvg
- Test files which can be used for initializing fuzzer
- Registered Qt project for oss-fuzz (but unused)

Questions?