Testing your code for security issues with automated fuzzing

Albert Astals Cid
aacid@kde.org
@tsdgeos
Akademy2019
Who is this?

In KDE for a while:
- KPDF/Okular
- Translations
- Releases
- kdegames
- kdeedu
- KDE España
- KDE e.V.
- A little bit of everything :)

NOT A SECURITY EXPERT
Which kind of security issues are we talking about?

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

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

People with more experience than me are able to turn this memory related crashes into code execution exploits.
Which tools do we have to detect such issues?

- The Operating System
  If your application uses memory incorrectly it will probably crash, making sure that doesn’t happen it’s a good first step ;)

- Valgrind
  valgrind will help us find memory errors. It’s 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)
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 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 lean and maximize coverage with the least number of “random” inputs.

```c
void theFunction(int x) {
    if (x > 50) {
    } else {
    }
}
```
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
- Mantainers have 30 days to fix a given issue
- If not fixed the issue is made public after 30 days

All the software to build the SAAS is free software in case you want to run one yourself
What is oss-fuzz? (IV)

1. Write fuzzers
2. Commit build configs
3. Sync and build from
4. Upload
5. Download and fuzz
6. File bugs, Verify fixes
7. Notify
8. Fix bugs

Upstream project
Builder (jenkins.io)
GCS bucket
ClusterFuzz
Sheriffbot
Issue tracker (monorail)

Developer
oss-fuzz and KDE

kimageformats (January)
kcodec (February)
karchive (April)

poppler (May 2018)
libical (April)

What do they have in common?
KArchive en oss-fuzz

KArchive y ficheros laaaaaaaaaaaaaaaaaaaargos

KImageFormats en oss-fuzz

Bugs ya resueltos de KImageFormats

https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=12752

#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 0x54e90e in LLVMFuzzerTestOneInput /src/kimgio_fuzzer.cc:60:12

git log

Parche para qpnghandler.cpp en Qt
Future

What else should be fuzz?
baloo
kfilemetadata
more pim stuff?

Aaaaaaaand who’s going to work in it ;)
Questions

Who?

When?

How?

Why?