Fully Automated Static Analysis of Fedora Packages

Red Hat
Kamil Dudka
August 9th, 2014

Abstract

There are static analysis tools (such as Clang or Cppcheck) that are able to find bugs in Fedora packages before they are noticed by users or even Security Response Team. We are announcing an easy to use tool – csmock – that makes it possible to run static analysis tools on RPM packages fully automatically. csmock is given an SRPM and simply returns a list of defects found in the package.

fedora

Definition of Static Analysis

- Static analysis is an analysis of code without executing it.
- Already done by the compiler (optimization, warnings, . . .).
- This presentation focuses on using static analysis tools to find bugs in Fedora packages.

fedora

Agenda

fedora

Analyzers of C/C++ Code Available in Fedora

- Compilers:
  - GCC – produces various kinds of useful warnings
- Analyzers applicable on vast majority of Fedora Packages:
  - Cppcheck – based on pattern matching
  - Clang – static checkers built on top of LLVM
- Analyzers with limited scope of use:
  - sparse – used mainly by kernel developers
  - ...
- Research prototypes with limited level of automation:
  - frama-c – uses CIL as the parser (written in OCaml)
  - ...

fedora

GCC Warnings

- Taken seriously by upstream developers (of some projects), but usually ignored by Fedora package maintainers.
- Enforcing -Werror=... to get them fixed is a good way to introduce new bugs (see `https://bugzilla.redhat.com/1025257#c5`).
- GCC warnings are difficult to process in an automated way:
  - GCC does not use absolute paths in diagnostic messages.
  - The format of diagnostic messages is not parser-friendly.
  - Multi-line GCC warnings are difficult to collect consistently when building in parallel.

fedora

Cppcheck

- Based on pattern matching (very lightweight static analysis).
- Can be run on a directory with sources, but then it:
  - tries several combinations of `-D` flags,
  - ignores missing include files in this mode,
  - causes false positives and false negatives (loss of precision),
  - reports bugs in unrelated code (e.g. `#ifdef WIN32` sections),
  - there is no easy way to predict the time needed for the analysis (e.g. to limit the time per compilation unit).
- Can be run per compilation unit with correct `-D` and `-I` flags.
  - Not so easy to automate when analyzing arbitrary SRPMs.
  - `cscppc` is a new package in Fedora that runs Cppcheck fully transparently during the build (in parallel with the compiler).

fedora

Clang Analyzer

- A set of static analysis-based checkers built on top of LLVM (Low Level Virtual Machine).
- There is a Perl script named `scan-build` that runs Clang during the build somewhat transparently:

```
scan-build rpmbuild --rebuild krb5-1.11.5-7.fc20.src.rpm
```

- Uses HTML or plist format for the results.
- Known to fail when analyzing certain packages:
  - `krb5` (because it overrides `$CC` by the `%{__cc}` RPM macro)
  - `pidgin` (because libtool did not recognize the faked compiler)
  - ...

fedora

Static Analyzers in Fedora – Summary

- There are ready to use static analyzers in Fedora, which provide useful results...
- ...but there is no common interface to run them,
- ...and there is no common format for the results.

fedora

Agenda

fedora

Fully Automated Static Analysis of SRPMs

- How to easily analyze a given SRPM by Cppcheck and Clang?

```
csmock -t cppcheck,clang krb5-1.11.5-7.fc20.src.rpm
```

- How to get csmock running on Fedora?

```
sudo yum install csmock
```

```
sudo gpasswd -a $USER mock
```

fedora

Results of Static Analysis (HTML format)

krb5-1.11.5-7.fc20

List of Defects

Error: CLANG_WARNING: [#def1]
krb5-1.11.5/src/lib/krad/remote.c:118:9: **warning**: Access to field 'tqh_last' results in a dereference of a null pointer
```
#        TAILQ_REMOVE(&req->rr->list, req, list);
#        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```
krb5-1.11.5/src/include/k5-queue.h:547:20: **note**: expanded from macro 'TAILQ_REMOVE'
```
#                (head)->tqh_last = (elm)->field.tqe_prev;               \
#                ~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~
```

Error: CLANG_WARNING: [#def2]
krb5-1.11.5/src/lib/rpc/xdr_rec.c:165:9: **warning**: Potential leak of memory pointed to by 'rstrm'
```
#                (void)fprintf(stderr, "xdrrec_create: out of memory\n");
#                      ^~~~~~~
```

Error: CPPCHECK_WARNING: [#def3]
krb5-1.11.5/src/lib/rpc/xdr_rec.c:166: error[**memleak**]: Memory leak: rstrm

Error: COMPILER_WARNING: [#def4]
krb5-1.11.5/src/lib/rpc/xdr_rec.c: **scope_hint**: In function 'xdrrec_getbytes'
krb5-1.11.5/src/lib/rpc/xdr_rec.c:258:18: **warning**: comparison between signed and unsigned integer expressions [-Wsign-compare]
```
#   current = (len < current) ? len : current;
#                  ^
```

fedora

What else can we do with csmock?

- Capture GCC warnings (by `-t gcc`).
- Change the GCC warning level (by `-w[0-2]`).
- Check for downstream-only bugs (by `--diff-patches`).
- Analyze upstream tarballs instead of SRPMs (by `-c CMD`).
- Get the results in a predictable amount of time (by default).
- You can easily write plug-ins for additional static analyzers.

fedora

How is csmock implemented?

- It uses mock (chroot-based tool for building RPMs):
  - to use the same build environment as for production builds,
  - to be able to do destructive changes in the chroot.
- It uses cswrap (a compiler/analyzer wrapper):
  - to capture the results consistently when building in parallel,
  - to translate relative to absolute paths in GCC output,
  - to transparently add/remove compiler flags,
  - to limit the time spent by analysis per compilation unit.

fedora

Processing the Results of Static Analysis

The csdiff Fedora package provides command-line utilities for processing the results:

- `csdiff` – matches defects introduced by an update
- `csgrep` – filters the list of defects by various predicates
- `cshtml` – generates an HTML report from the list of defects

fedora

Future Plans

- Write plug-ins for additional static analyzers.
- Support additional languages – Python, Java, . . .
- Expose csmock as a service – hook a task on Taskotron?

fedora

Conclusion

- There are static analyzers in Fedora that are able to automatically find bugs in our packages.
- csmock makes it easy to run them on a given SRPM.
- csdiff utilities can be used to efficiently process the results.
- These packages are now available in Fedora!