Fully Automated Dynamic Analysis of RPM Packages

Kamil Dudka <kdudka@redhat.com>

February 20th 2021

Abstract

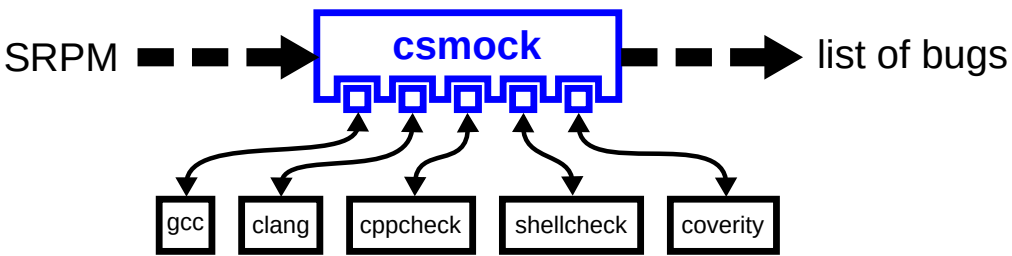
It is easy to statically analyze source code of RPM packages. Fedora contains a tool named csmock, which takes a source RPM package, runs static analyzers on it, and returns a list of potential programming mistakes detected in the package. We are extending this fully automated solution for dynamic analyzers, such as valgrind or strace. Using this extension, one can easily get a list of bugs detected by valgrind in the regression tests embedded in a source RPM package. Thanks to our innovative approach, the results do not contain unrelated reports that would otherwise be produced by bash, make, python interpreter, and all the external testing frameworks.

Static vs. Dynamic Analysis

- Static analyzers:
  - do not execute programs.
  - embedded in compilers: `clang --analyze`, `gcc -fanalyzer`
  - standalone tools: cppcheck, shellcheck, coverity, ...
- Dynamic analyzers:
  - execute code in a modified run-time environment.
  - embedded in compilers: address sanitizer, thread sanitizer, ...
  - standalone tools: valgrind, strace, ...

Analysis of RPM Packages

- Command-line tool to run static analyzers on RPM packages.
- One interface, one output format, plug-in API for (static) analyzers.
- Can we implement plug-ins for dynamic analyzers?



Tests Embedded in RPM Packages

```
$ fedpkg clone -a logrotate
$ cd logrotate
$ grep -A8 '%build' logrotate.spec
%build
mkdir build && cd build
%global _configure ../configure
%configure --with-state-file-path=%{_localstatedir}/lib/logrotate/logrotate.status
%make_build

%check
%make_build -C build -s check

$ fedpkg srpm
$ rpmbuild --rebuild *.src.rpm
```

Dynamic Analysis of RPM Packages – Simple Approach

- Dynamic analyzers usually support tracing of child processes.
- Let's combine it together:
  - `valgrind --trace-children=yes rpmbuild --rebuild *.src.rpm`
  - `strace --follow-forks rpmbuild --rebuild *.src.rpm`
- But did we want to dynamically analyze rpmbuild, bash, make, etc.?
  - This makes the analysis extremely slow.
  - We get reports unrelated to `*.src.rpm`.

Dynamic Analysis of RPM Packages – Better Approach

- Produce binaries that will launch a dynamic analyzer for themselves.
- We can use a compiler wrapper to instrument the build of an RPM package:

```
$ export PATH=$(cswrap --print-path-to-wrap):$PATH
$ export CSWRAP_ADD_CFLAGS=-Wl,--dynamic-linker,/usr/bin/csexec-loader
$ export CSEXEC_WRAP_CMD=valgrind
$ rpmbuild --rebuild *.src.rpm
```

- Only binaries produced in `%build` will run through valgrind in `%check`.

Red Hat

Program Interpreter

- Program interpreter specified by shebang:

```
$ head -1 /usr/bin/yum
#!/usr/bin/python3
```

```
$ /usr/bin/yum [...]  ⟶  /usr/bin/python3 /usr/bin/yum [...]
```

- Program interpreter specified by ELF header:

```
$ file /sbin/logrotate
/sbin/logrotate: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=...
```

- ELF interpreter can be set to a custom value when linking the binary:

```
$ file ./logrotate
./logrotate: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /usr/bin/csexec-loader, BuildID[sha1]=...
```

Wrapper of Dynamic Linker – Implementation

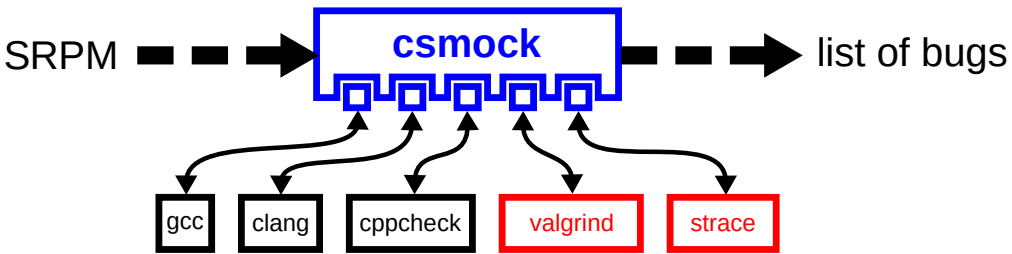
- `csexec` works as a wrapper of the system dynamic linker.
- `$CSEXEC_WRAP_CMD` can specify a dynamic analyzer to use.
- csexec runs the system dynamic linker explicitly (to eliminate self-loop):
  `./logrotate [...]` ⟶ `valgrind /lib64/ld-linux-x86-64.so.2 ./logrotate [...]`
- csexec uses the `--argv0` option of the system dynamic linker if available:
  `https://sourceware.org/git/?p=glibc.git;a=commitdiff;h=c6702789`
- csexec emulates the original target of the `/proc/self/exe` symlink.

Wrapper of Dynamic Linker – Evaluation

- No completely unrelated bug reports.
- Minimal performance overhead.
- Minimal interference with commonly used testing frameworks.
- Able to successfully run upstream test-suite of GNU coreutils (without valgrind).
- Some tests fail if we wrap them by valgrind though:
  - a test that verifies the count open file descriptors
  - a test that intentionally sets non-existing `$TMPDIR`
  - ...

Dynamic Analysis of RPM Packages with csmock

- Experimental csmock plug-ins for valgrind and strace:




```
$ sudo yum install csmock-plugin-valgrind
$ csmock -t valgrind -r fedora-rawhide-x86_64 *.src.rpm
```

- Extended demo: https://github.com/kdudka/cswrap/wiki/csexec

Future Work

- Implement parser of valgrind's XML output.
- Port csexec to more architectures (x86_64 only for now).
- Support use of multiple dynamic analysis plug-ins in a single run of csmock.
- Integrate formal verification tools (Symbiotic, Divine, CBMC) as csmock plug-ins as part of the AUFOVER (Automation of Formal Verification) project, supported by Technology Agency of the Czech Republic:
  https://starfos.tacr.cz/en/project/TH04010192
- Improve the formal verification tools to handle more RPM packages (task for developers of the verification tools).