# Map2Check - Tutorial

Herbert Rocha
Rafael Menezes
Lucas Cordeiro

2018/12/21



Map2Check is a software verifier able to check for security properties in C programs. It currently supports the following properties (note: only one mode can be run at a time):

- Memsafety: default mode;
- Memcleanup: `--memcleanup-property`
- Signed Integer Overflow: `--check-overflow`;
- Reachability: `-f`, checks for function `__VERIFIER_error`;
- Asserts: `--check-asserts`;

- Ubuntu 16.04 or greater;
- Packages `libc6-dev` and `python-minimal`;
- RAM requirements depends on the input program that is being checked; 4GB should be enough in most cases.

Map2Check supports C and LLVM bytecode (experimental) programs as input, but some minor preparations on the input are needed to properly use it:

- Map2Check currently has no support to multi-threaded programs.
- The file extension should be: `.c`, `.i` (for C) or `.bc` (for LLVM bytecode)
- Make sure that your program can be compiled without any extra file (some `libc` headers such as `stdlib` are okay) and that it contains a `main` method.
- If you are using a LLVM bytecode you should use the `-g` flag when generating it (using `clang`).

```c
int main() {
  int a = __VERIFIER_nondet_int();
  int b = __VERIFIER_nondet_int();
  int c = a + b;

  __VERIFIER_assert(c != 42);
  return 0;
}
```

```
./map2check -t 60
--check-asserts
./input.c
```

```
State 0: file /home/map2check/devel_tool/mygitclone/release/./test.c
-----------------------------------------------------------
>Symbolic log
  Call Function  : __VERIFIER_nondet_int()
  Value          : -8
  Line Number    : 2
  Function Scope : main

State 1: file /home/map2check/devel_tool/mygitclone/release/./test.c
-----------------------------------------------------------
>Symbolic log
  Call Function  : __VERIFIER_nondet_int()
  Value          : 50
  Line Number    : 3
  Function Scope : main


-------------------------------------------------
Violated property:
        file map2check_property line 6 function main
        FALSE-ASSERT

VERIFICATION FAILED
```

The counterexample shows information about the program states such as: pointer tracking, non-deterministic calls and memory allocation/deallocation. From our previous example, we have

- In line 2, in the `main` function, a nondet call with value $-8$
- In line 3, in the `main` function, a nondet call with value 50
- Finally, in line 6 , in the `main` function, a violation of the assert statement
- We can manually validate (or using other tools) this violation by checking that $-8 + 50 = 42$.

- The `-t` flags specifies a timeout; you should always define one based on the input program. Map2Check iterates over two executors: LibFuzzer and Klee, this iteration is based on the set timeout (Libfuzzer: 20%, Klee: 80%). If no timeout is defined, Libfuzzer will run until it does not generate any new test case (which can take very long) or it can find an error.
- To systematically explore all paths, Klee is used, but the first executor is LibFuzzer; so sometimes a simple input program might take too long to report that there is no property violation.

- Memsafety checkings are based on tracking allocation/deallocation methods from *stdlib*. So make sure your program uses it.
- Map2Check functions and properties are based on SV-COMP rules, so it may be helpful to read it (`https://sv-comp.sosy-lab.org/2019/rules.php`)
- Map2Check does not support `__VERIFIER_atomic_begin`.

If you have any questions, or you would like to make a request or have found a bug, please send an e-mail to: map2check@gmail.com

Or if you want to have a look at the source code, it is available in GitHub: `https://github.com/hbgit/Map2Check/`