UTBot Simplifies Auto Test Generation
Samat Gaynutdinov, Saveliy Grigoryev, Pavel Iatchenii, Elena Ilina, Dmitry Ivanov,
Vladislav Kalugin, Aleksei Pleshakov, Pavel Ponomarev, Konstantin Rybkin, Svetlana Shmidt,
Vadim Volodin, Alexey Utkin

Generating C/C++ tests with KLEE is hard. UTBot does the work!

1. Prepare a project

<table>
<thead>
<tr>
<th>Project type</th>
<th>Project preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMake project</td>
<td>UTBot runs patched CMake to generate both compile_commands.json and link_commands.json</td>
</tr>
<tr>
<td>Make project</td>
<td>UTBot runs patched Bear to generate both compile_commands.json and link_commands.json</td>
</tr>
<tr>
<td>Other</td>
<td>UTBot asks user to run Bear $BUILD_COMMAND</td>
</tr>
</tbody>
</table>

**compile_commands.json** — compilation database

**link_commands.json** — linking commands for libraries and executables (specific to UTBot)

2. Prepare KLEE run

During the analysis phase, UTBot uses information about the declared top-level functions and types collected while opening the project. UTBot wraps the functions for KLEE.

```c
int klee_entry_main_abs_wrapped(int _argc,
    char **_argv, char **_envp)
{
    int val;
    klee_make_symbolic(&val, sizeof(val), "val");
    klee_prefer_cex(&val, val >= -10 & val <= 10);
    int _result;
    Make symbolic parameter
    _result = klee_symbolic(&_result, sizeof(_result), "_result");
    int tmp = abs(val);
    klee_assume(tmp == _result);
    return 0;
}
```

3. Google Test generation

KLEE returns test case descriptions for a function that was previously specified as an entry point. UTBot performs the transformation of KLEE output into the Google Test format.

4. Compile and run tests

There are plugins for VS Code and CLion, GitHub Actions.

UTBot allows compiling test files to link them with the rest of the code and to run the tests to get results and information about code coverage. UTBot can also be used as a zero false positive static analyzer by generating a SARIF report.

**Enhancements in KLEE**

- **Speed:**
  - Pruning the Recursive States*
  - Weakest Precondition in Symbolic Execution

- **Code coverage:**
  - Floating-point Support
  - Complex Test Input Generation*
  - Detection of Undefined Behavior*

* presented at the 3rd International KLEE Workshop, 2022

**References**

https://www.utbot.org
https://github.com/UnitTestBot/UTBotCpp
https://github.com/UnitTestBot/klee

3rd International KLEE Workshop on Symbolic Execution 15–16 September 2022 • London, UK