链滴

KLEE 源码安装（Ubuntu 16.04 + LLVM 9）

作者：Hanseltu

原文链接：https://ld246.com/article/1599015720782

来源网站：链滴

原文链接： KLEE 源码安装（Ubuntu 16.04 + LLVM 9）

符号执行在近几年的软件测试及漏洞挖掘领域越来越火，其主要思想是把程序的执行路径转化为一个路径约束，然后使用约束求解器求解这些约束，从而生成特定覆盖路径的测试用例。其中最有名的工就属KLEE 了。KLEE是建立在 LLVM 编译器基础结构之上的符号执行虚拟机，从2008年提出以来，经在学术界和工业界得到了广泛的研究和应用。本文主要讨论 KLEE 安装，更多技术细节后面再细究。

KLEE 主要有两种安装方法，一是源码安装，可能比较费时；二是使用docker，这个会比较快。个人喜欢真实源码环境，在此记录一下源码安装过程，主要参考官方安装教程 Building KLEE with LLVM 9 如果你更喜欢docker环境，请移步 Our Docker images.

源码安装主要分为以下几步：

- 1 准备工作
- 2 安装 LLVM 9 (官方推荐使用LLVM 9版本)
- 3 安装约束求解器
- 4 编译 uClibc 和 POSIX 运行环境
- 5 编译KLEE

1 准备工作

环境：Ubuntu 16.04

编译器：clang clang++

```
$ clang --version
clang version 9.0.0 (tag/RELEASE_900/final)
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir /home/haoxin/corpos-compilers/llvm-9.0/llvm-src/build/bin
```

安装包依赖：

```
$ sudo apt-get install build-essential curl libcap-dev git cmake libncurses5-dev python-minim
l python-pip unzip libtcmalloc-minimal4 libgoogle-perftools-dev libsqlite3-dev doxygen
$ pip3 install tabulate wllvm
```

2 安装 LLVM 9

由于 LLVM 9 在 Ubuntu 16.04 上不能使用apt安装，这里需要自行源码编译安装LLVM。安装命令下 （在这里耗时较长，可能需要一个小时左右）

```
$ wget http://releases.llvm.org/9.0.0/llvm-9.0.0.src.tar.xz
$ wget http://releases.llvm.org/9.0.0/cfe-9.0.0.src.tar.xz
$ wget http://releases.llvm.org/9.0.0/clang-tools-extra-9.0.0.src.tar.xz
$ tar xvf llvm-9.0.0.src.tar.xz
$ tar xvf cfe-9.0.0.src.tar.xz
$ tar xvf clang-tools-extra-9.0.0.src.tar.xz
$ mv llvm-9.0.0.src llvm-src
$ mv cfe-9.0.0.src clang
$ mv clang llvm-src/tool/clang
```

```
$ my clang-tools-extra-9.0.0.src extra
$ mv extra llvm-src/tools/clang/tools/extra

$ cd llvm-src
$ mkdir build
$ cd build
$ cmake  -DLLVM_TARGETS_TO_BUILD=X86 -DCMAKE_BUILD_TYPE="Release"  -DCMAKE_IN
TALL_PREFIX=./ -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ ..
$ make -j8
$ make install
```

3 安装约束求解器

KLEE 支持多种约束求解器，如STP, Z3, meteSMT等。在此安装Z3 （耗时大约10分钟）

```
$ wget https://github.com/Z3Prover/z3/archive/z3-4.8.8.zip
$ unzip z3-4.8.8.zip
$ cd z3-4.8.8
$ mkdir build
$ cd build
$ cmake -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++
```

注意：这里使用的编译器最好换成clang，和后面编译 KLEE 保持一致，否则会因为在编译 KLEE 是不到Z3的某个头文件而编译失败

4 编译 uClibc 和 POSIX 运行环境

如果要分析的源代码中包含了C库函数，则需要安装POSIX运行环境，安装过程如下：

```
$ git clone https://github.com/klee/klee-uclibc.git
$ cd klee-uclibc
$ ./configure --make-llvm-lib
$ make -j2
$ cd ..
```

注意：默认情况下，环境中的 clang, llvm-config 都应该是9.0版本的，保持一致

5 编译KLEE

上面的步骤如果都没有问题，下面就可以源码安装KLEE了。命令如下

```
$ wget https://github.com/klee/klee/archive/v2.1.zip
$ unzip v2.1.zip
$ cd klee-v2.1
$ mkdir build
$ cd build
$ cmake -DENABLE_SOLVER_Z3=ON -DENABLE_POSIX_RUNTIME=ON -DENABLE_KLEE_UCLI
C=ON -DKLEE_UCLIBC_PATH=/home/haoxin/symbolic-execution/klee-uclibc-v1.2 -DLLVM_C
NFIG_BINARY=/home/haoxin/corpus-compilers/llvm-9.0/llvm-src/build/bin/llvm-config -DLL
MCC=/home/haoxin/corpus-compilers/llvm-9.0/llvm-src/build/bin/clang -DLLVMCXX=/home
haoxin/corpus-compilers/llvm-9.0/llvm-src/build/bin/clang++ -DCMAKE_C_COMPILER=clang
-DCMAKE_CXX_COMPILER=clang++ ..
$ make -j8
```

```
$ sudo make install
$ klee --version
KLEE 2.1 (https://klee.github.io)
Build mode: RelWithDebInfo (Asserts: ON)
Build revision: unknown

LLVM (http://llvm.org/):
LLVM version 9.0.0
Optimized build.
Default target: x86_64-unknown-linux-gnu
Host CPU: ivybridge
```

Ok, 到此 KLEE 已经安装完毕了，就可以正常使用了。

KLEE 测试案例

下面用过一个简单的例子说明KLEE的使用方法，主要的流程有两步，第一步将源码编程成 .bc 字节码
第二步使用 KLEE 生成测试用例。

get_sign.c

```
/*
 * First KLEE tutorial: testing a small function
 */

#include "klee/klee.h"

int get_sign(int x) {
  if (x == 0)
    return 0;

  if (x < 0)
    return -1;
  else
    return 1;
}

int main() {
  int a;
  klee_make_symbolic(&a, sizeof(a), "a");
  return get_sign(a);
}
```

第一步编译：

```
$ clang -I -emit-llvm -c -g -O0 -Xclang -disable-O0-optnone get_sign.c
```

第二步执行 KLEE:

```
$ klee get_sign.bc
KLEE: output directory = "/home/haoxin/symbolic-execution/klee-out-0"

KLEE: done: total instructions = 31
KLEE: done: completed paths = 3
```

```
KLEE: done: generated tests = 3
```

可以看到 KLEE 执行完以后报告了总共有多少条执行路径及覆盖各路径的测试用例。具体情况后面再述。

遇到的问题

问题1

```
$ klee get_sign.bc
KLEE: ERROR: Loading file /usr/local/lib/klee/runtime/libkleeRuntimeFreeStanding.bca failed: nvalid record
//or
KLEE: ERROR: Loading file get_sign.bc failed: Invalid record
```

这种错误是因为编译KLEE的编译器版本和编译源码的版本不一样，比如用 clang-9 编译的 KLEE 而用clang-10编译源码 get_sign.c，就会出现以上错误

问题二：

```
$ klee --version
klee: /usr/lib/x86_64-linux-gnu/libstdc++.so.6: version `GLIBCXX_3.4.26' not found (required b klee)
```

这种情况是因为在编译 KLEE 时没有指定 clang 编译器，而是用默认的编译器（可能是gcc）编译器，可能就会存在找不到库的情况。

以上即 KLEE 源码安装过程，后续会陆续更新一些 KLEE 更详细的介绍及使用。

参考：

1 KLEE 主页：https://klee.github.io

2 KLEE paper 2008 OSDI :

KLEE: Unassisted and Automatic Generation of High-CoverageTests for Complex Systems Programs