Automatic Test Generation for String Manipulation Programs using Symbolic Execution

Anitha Gollamudi, Zehra Naz
Harvard University

Abstract

Symbolic execution of string manipulation programs is challenging as the constraint solvers do not typically support logic over strings and non-string operations. KLEE [1] is a symbolic execution tool used to generate test cases with high coverage. It uses Simple Theorem Prover (STP) as its constraint solver. STP encodes constraints only as bit-vector logic and solves the constraints. It has no direct representation of strings. Strings and their operations (like concatenation, substring) have to be encoded in bit-vector logic and then passed onto STP for a solution. This renders STP unsuitable for analyzing arbitrary length strings.

KLEE has some basic support for handling symbolic strings. However, the quality of test cases generated quickly deteriorates when string operations are used. Our initial analysis shows that it cannot handle arbitrary-length strings. The symbolic value is always bounded by its size. Clearly, this is undesirable in applications where the size of string is relatively unknown. The limitations are particularly relevant when testing applications for SQL injections in web servers like Apache or Nginx.

We therefore aim to achieve the following:

1. Clearly identify the weaknesses of KLEE, as is, when handling symbolic strings on applications that use string operations
2. Compare and contrast the above results by porting KLEE to other string solver (CVC or Z3-str [2])
3. Evaluate the above ported KLEE to detect SQL injections.

Related Work

Symbolic execution for string manipulation is popular in web testing frameworks especially to detect cross-site scripting (XSS) and SQL injections. Several popular tools applied this technique to detect vulnerabilities in python, PHP and javascript libraries. However our literature survey shows that there exist no such tools for testing C or C++ applications, to our knowledge. We think this is particularly important given that performance oriented web servers like Apache [9], Nginx [8] or browsers like Firefox or Chrome are developed in C/C++. These programs heavily rely on string manipulations for forming SQL queries and carrying out other tasks.

Our literature survey pointed us to Klover [3] - a KLEE based coverage tool for C/C++. The tool handles string operations. However the tool is proprietary and we unfortunately could not use its source code for our project.

KLEE-MultiSolver [4] is a framework (it evolved into metaSMT [6]) that targets KLEE to several popular constraint solvers. Z3 is one of the supported constraint solvers. Though Z3 is supported, no effort has been made to integrate Z3-str which augments Z3 constraint solver to use strings. Also, metaSMT does not support string theory. [7]

Implementation

We will be using KLEE as our base symbolic execution tool. The plan is to make KLEE able to use string solvers.

1. Design a suite of string manipulation programs in C. We will run tests on these. Development of this suite will be done incrementally as we add complexity to the project.
2. Use KLEE + STP framework to evaluate the coverage. This will be our baseline.
3. Support basic string theory in metaSMT.
4. Integrate Z3-str into metaSMT.
5. Evaluate KLEE + (metaSMT + Z3-str) against KLEE + STP.
6. (Ambitious) Integrate HAMPI with KLEE so as to constrain SQL queries generated. An altered form
of this work has been presented in [5]. We plan to use integrated HAMPI+KLEE to test sql injections in simple web server programs.

**Deliverables**

We hope that this project will give us necessary insight into the dependency of symbolic execution on underlying constraint solvers. At minimum, we aim to deliver an evaluation of the tool we develop by combining KLEE with metaSMT + Z3-str against existing KLEE + STP in the context of running symbolic execution on strings. An ambitious goal will be to present the evaluated results of SQL injections.

Our one week project will be to write simple string manipulations to start the development of our test suite. We will run KLEE on these and see how far we can get in getting KLEE to identify buggy test cases in these. Once we establish a baseline of and exhaust KLEE’s powers with regards to string manipulation, we will extend KLEE with simple logic from Z3-str (such as its substring or concatenation-handling ability). We will test this to see if improvement in KLEE is observed and by how much. This will allow us to not only learn about the inner workings of KLEE, but also of the string solver Z3-str which will be very useful later in the project. It will additionally help us to see how much room there is for improvement in KLEE with regards to string handling and what it takes to build extensions into KLEE.

**Testing**

Testing will be done as follows:

1. **Bare minimum:** we aim to write simple string manipulation programs not involving any libraries outside of uclibc. We will run KLEE + STP on these to establish baseline of how KLEE, without modification, performs on strings. If we find existing test suites for C dealing with SQL injection or string manipulations, we will use those instead.

2. We can extend the programs in (1) with test cases that resemble real string manipulations that web servers such as microhttpd and Nginx perform to generate SQL queries.

3. Run our tool on a web server like microhttpd which comprises of a single C file.

4. **Ambitious:** Run our tool on Nginx to see how many SQL injection vulnerabilities or string manipulation errors we can identify.

**Concerns**

Our main concern is to make KLEE work with Z3-str using metaSMT. We are also new to SMT-LIB v2 format so there is a steep learning curve involved with changing metaSMT.

**References**


