Efficient Solving of String Constraints for Security Analysis

Clark Barrett[2], Cesare Tinelli[1], Morgan Deters[2], Tianyi Liang[1], Andrew Reynolds[1], Nestan Tsiskaridze[1]
[1] The University of Iowa, [2] New York University

AUTOMATED REASONING IN SECURITY

A recent successful approach to security analysis reduces security questions about programs to constraint satisfaction problems in some formal logic. Automatic reasoners for that logic can then be used to solve those problems.

Example: *A code for a password verification with a buffer overflow attack vulnerability is reduced to the set of constraints, solution to which provides an instance of a buffer overflow attack – i.e. proof of the vulnerability.*

```
char buff[15];
char pass;
std::cout << "Enter the password :";
gets(buff);
if (std::regex_match(
      buff,
      std::regex("([A-Z]+)") ) ) {
  if(strcmp(buff, "PASSWORD")) {
    std::cout << "Wrong Password";
  }
  else {
    std::cout << "Correct Password";
    pass = 'Y';
  }
}
if(pass == 'Y') {
  /* Grant the root permission*/
}
```

Encode

```
(set-logic QF_S)
(declare-const input String)
(declare-const buff String)
(declare-const pass0 String)
(declare-const pass1 String)
(declare-const mem String)
(assert (= (str.len buff) 15))
(assert (= (str.len pass1) 1))
(assert (str.prefix (str.++ buff pass0) mem)))
(assert (str.prefix (str.++ input "\0" ) mem)))
(assert (str.in.re buff (re.+ (re.range "A" "Z"))))
(assert (ite (str.prefix "PASSWORD\0" buff)
             (= pass1 "Y")
             (= pass1 pass0)))
(assert (not (str.prefix "PASSWORD\0" input)))
(assert (= pass1 "Y"))
```

Extract

Solve

```
ajreynol@serv16102[~]% cvc4 strings-ex-security.smt2
sat
(model
(define-fun input () String "AAAAAAAAAAAAAAY")
(define-fun buff () String "AAAAAAAAAAAAAA")
(define-fun pass0 () String "Y")
(define-fun pass1 () String "Y")
(define-fun mem () String "AAAAAAAAAAAAAAY\0")
)
```

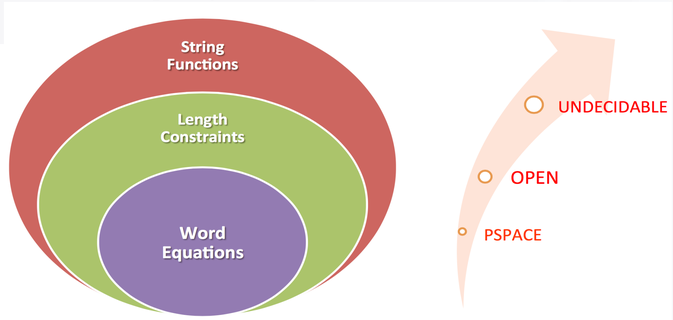
SATISFIABILITY MODULO THEORIES

Solvers based on *Satisfiability Modulo Theories* (SMT) techniques have become a natural choice in such approaches to security. The most sophisticated SMT solvers integrate a fast propositional engine with several theory solvers, each specialized on a theory of interest such as, linear integer arithmetic (LIA), uninterpreted functions (EUF), bit vectors, arrays, etc. The need for powerful automated solvers that can handle string constraints is increasingly important in security analysis.

SOLVING STRING CONSTRAINTS

A major difficulty is that, in general, the satisfiability problem of any reasonably comprehensive theory of character strings is undecidable. However, one can identify several restricted, but still quite useful, fragments of the theory of strings that are decidable.

Theoretical Complexity Challenges



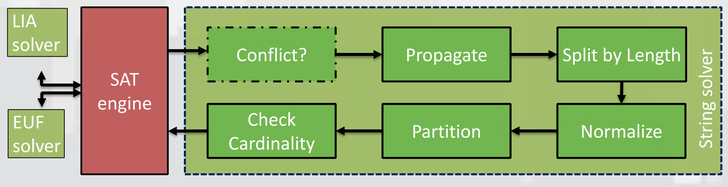
Recent research has focused on identifying decidable fragments suitable for program analysis and developing efficient solvers for them. Most of the string solvers are stand-alone tools that can reason only about small fragments of the theory of strings. These solvers are based on reductions to the satisfiability problems over other data types, such as bit-vectors, or to decision problems over automata. At the same time, they are either unsound, or lack expressiveness.

CVC4 STRING SOLVER

We developed a new algebraic approach for solving constraints over:

- a theory of unbounded strings,
- length constraints,
- extended regular expression membership,
- common string manipulating functions

directly without reducing to other problems. We implemented an automated string solver based on this approach, and incorporated it into the state-of-the-art SMT solver CVC4.



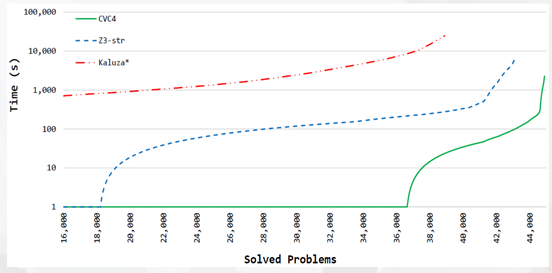
EXPERIMENTAL RESULTS

Our experimental results show that, on string problems, our CVC4 string solver is highly effective – it outperforms other existing specialized string solvers in terms of correctness, precision, and run time, when evaluated on comparable input fragments.

We compared our string solver against solvers: Kaluza (UC Berkley) and Z3-str (Purdue) on about 50,000 Kudzu benchmarks from real-world web security applications:

	CVC4	Z3-str		Kaluza	
Result		Incorrect[2]		Incorrect[2]	
unsat	11,625[1]	317	11,769	7,154	13,435
sat	33,271	1,583	31,372	n/a	25,468
unknown	0		0		3
timeout	2,388		2,123		84
error	0		120		1,140

1. CVC4 answers UNSAT, but neither Z3-str nor Kaluza answers SAT.
2. We verified the errors by asserting the model back to the assertions. Later versions of Z3-str have eliminated those errors at the cost of some loss in performance



THEORETICAL RESULTS

We have a general proof of correctness for our string solver in terms of refutation soundness and solution completeness. We have also identified two expressive fragments for which it is a decision procedure: one with unbounded strings and length constraints and one with regular language membership and length constraints.

The Science of Security Initiative is funded by the National Security Agency