Efficient solving of string constraints for security analysis

[Poster Abstract]

Clark Barrett
New York University

Cesare Tinelli
The University of Iowa

Morgan Deters
New York University

Tianyi Liang,
The University of Iowa

Andrew Reynolds
The University of Iowa

Nestan Tsiskaridze
The University of Iowa

Motivation The security of software is increasingly more critical for consumer confidence, protection of privacy, protection of intellectual property, and even national security. As threats to software security have become more sophisticated, so too have the techniques developed to ensure security. One basic technique that has become a fundamental tool in static security analysis is symbolic execution. There are now a number of successful approaches that rely on symbolic methods to reduce security questions about programs to constraint satisfaction problems in some formal logic (e.g., [4, 5, 7, 16]). Those problems are then solved automatically by specialized reasoners for the target logic. The found solutions are then used to construct automatically security exploits in the original programs or, more generally, identify security vulnerabilities.

In the last few years, solvers based on Satisfiability Modulo Theories (SMT) techniques have become a natural choice in such approaches to security because of: (i) their superior performance and level of automation compared to more traditional theorem provers; and (ii) their greater generality with respect to ad-hoc tools and propositional satisfiability solvers. The most powerful SMT solvers integrate a fast propositional engine with several theory solvers, each specialized on a theory of interest such as, for instance, linear arithmetic, bit-vectors, or arrays.

Security analyses are frequently required to reason about constraints over character strings. Program inputs, especially in web-based applications, are often provided as strings which are then processed using string operations such as matching against regular expressions, string concatenation, and substring extractions or replacement. In general, both safety and security analyses can benefit from automatic solvers capable of checking the satisfiability of constraints over a rich set of data types that includes character strings.

Major challenges A major difficulty in reasoning about strings is that any reasonably comprehensive theory of character strings is undecidable [3]. However, several more restricted (but still quite useful) theories of strings do have a decidable satisfiability problem. These include any theories of fixed-length strings, which are trivially decidable for having a finite domain, but also some fragments over unbounded strings (e.g., word equations [15]). The satisfiability problem in most of these fragment has high worst-case time complexity (from NP to PSPACE [18]). Recent research has focused on identifying decidable fragments suitable for program analysis and, more crucially, on developing efficient solvers for them.

Previous state of the art Until recently, satisfiability solvers for strings were standalone tools that could reason only about some fragments of the theory of strings and regular expressions, sometimes with strong restrictions on the expressiveness of their input language such as, for instance, the imposition of exact length bounds on all string variables [10]. These solvers were based on reductions to satisfiability problems over other data types such as bit vectors [10], arrays [11], or integer arithmetic [3], or on reductions to automata decision problems [8, 9, 6]. General multi-theory SMT solvers have emerged as overall more powerful tools when reasoning over several datatypes is required. Thus incorporating native string support into an SMT solver has the potential to open a wide range of opportunities for security analyses, which normally need to consider programs with more than just string operations. Yet, so far SMT solvers have had minimal or no native support for handling strings, mostly due to the complexity of string solving.

Presented work This poster summarizes our recent work aimed at addressing the issues discussed above and published in a number of recent or upcoming papers [12, 14, 13]. The presented work has been developed with input and feedback from security experts at Carnegie-Mellon University. Our implementation within the SMT solver CVC4 [2] is being used at CMU for the analysis of Python programs. Several other users worldwide are now using the solver for a variety of safety and security analyses of programs that process strings in combinations with other datatypes.

Contribution and significance We present a set of algebraic techniques for solving constraints over a rich theory of unbounded strings natively, without reduction to other problems. These techniques can be used to expand SMT solvers with capabilities to reason over strings, in the context of the DPLL(T) architecture used by most state-of-the-art SMT solvers [17].

HotSoS '16, April 19–21, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2898375.2898393

Formally, we have devised a new calculus for reasoning about quantifier-free constraints over a theory of unbounded strings with length constraints, regular language membership constraints, and common string manipulating functions. We implemented a proof procedure for this calculus within CVC4 as a theory solver for strings, thus expanding CVC4's already large set of built-in theories. This work makes CVC4 the first SMT solver that is able to handle mixed constraints over strings, integers/reals, bit-vectors, arrays, and algebraic data types.[1]

Theoretical results We proved that our calculus is solution sound and refutation sound [13]. That is, when our solver returns a solution we can trust that it is indeed a solution (solution soundness). If our solver concludes there is no solution to an input instance we can trust this conclusion too (refutation soundness).

We obtained decidability results on some expressive fragments of our theory of strings by showing that our calculus can be turned into a decision procedure for those fragments. In particular, we devised a decision procedure for unbounded strings with regular expression membership and length constraints. [14]

Experimental evaluation We evaluated the the performance of our solver by comparing it against a legacy string solver, Kaluza [19], and more recent ones based on similar approached to ours: Z3-str [21], S3 [20], and Norn [1]. These solvers, which have also been used in security analysis applications, were chosen for being publicly available and having an input language that largely overlaps with that of our solver.

We ran experiments on a large set of about 50K benchmarks, coming from real-life web-security applications. They were generated by Kudzu, a symbolic execution framework for Javascript, and are available on the Kaluza web site[2]. The experimental evaluation shows that on string problems our approach is highly effective – overall it outperforms the other string solvers in terms of correctness, precision, and run time.

Acknowledgments This work was funded partially funded by grants 1228765 and 1228768 from the National Science Foundation.

Keywords

String solving, SMT, automated security analysis.

1. REFERENCES

[1] P. A. Abdulla, M. F. Atig, Y. Chen, L. Holík, A. Rezine, P. Rümmer, and J. Stenman. Norn: An SMT solver for string constraints. In D. Kroening and C. S. Pasareanu, editors, *Proceedings of the 27th International Conference on Computer Aided Verification*, volume 9206 of *Lecture Notes in Computer Science*, pages 462–469. Springer, July 2015.

[2] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. CVC4. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, CAV'11, pages 171–177. Springer-Verlag, 2011.

[1]CVC4 is publicly available at http://cvc4.cs.nyu.edu .

[2]http://webblaze.cs.berkeley.edu/2010/kaluza/ .

[3] N. Bjørner, N. Tillmann, and A. Voronkov. Path feasibility analysis for string-manipulating programs. In *Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009,*, pages 307–321. Springer-Verlag, 2009.

[4] D. Brumley, J. Caballero, Z. Liang, and J. Newsome. Towards automatic discovery of deviations in binary implementations with applications to error detection and fingerprint generation. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007.

[5] D. Brumley, H. Wang, S. Jha, and D. X. Song. Creating vulnerability signatures using weakest preconditions. In *20th IEEE Computer Security Foundations Symposium, CSF 2007, 6-8 July 2007, Venice, Italy*, pages 311–325, 2007.

[6] A. S. Christensen, A. Møller, and M. I. Schwartzbach. Precise analysis of string expressions. In *Proceedings of the 10th International Conference on Static Analysis*, SAS'03, pages 1–18. Springer-Verlag, 2003.

[7] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. Song. Dynamic spyware analysis. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ATC'07, pages 18:1–18:14, Berkeley, CA, USA, 2007. USENIX Association.

[8] P. Hooimeijer and W. Weimer. A decision procedure for subset constraints over regular languages. In *Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, pages 188–198. ACM, 2009.

[9] P. Hooimeijer and W. Weimer. Solving string constraints lazily. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 377–386. ACM, 2010.

[10] A. Kiezun, V. Ganesh, P. J. Guo, P. Hooimeijer, and M. D. Ernst. HAMPI: a solver for string constraints. In *Proceedings of the eighteenth international symposium on Software testing and analysis*, pages 105–116. ACM, 2009.

[11] G. Li and I. Ghosh. PASS: String solving with parameterized array and interval automaton. In V. Bertacco and A. Legay, editors, *Hardware and Software: Verification and Testing*, volume 8244 of *Lecture Notes in Computer Science*, pages 15–31. Springer International Publishing, 2013.

[12] T. Liang, A. Reynolds, C. Tinelli, C. Barrett, and M. Deters. A DPLL(T) theory solver for a theory of strings and regular expressions. In A. Biere and R. Bloem, editors, *Proceedings of the 26th International Conference on Computer Aided Verification*, volume 8559 of *Lecture Notes in Computer Science*. Springer, 2014.

[13] T. Liang, A. Reynolds, N. Tsiskaridze, C. Tinelli, C. Barrett, and M. Deters. An efficient smt solver for string constraints. *Formal Methods in System Design*, 2016. (To appear).

[14] T. Liang, N. Tsiskaridze, A. Reynolds, C. Tinelli, and C. Barrett. A decision procedure for regular

membership and length constraints over unbounded strings. In C. Lutz and S. Ranise, editors, *Proceedings of the 10th International Symposium on Frontiers of Combining Systems*, volume 9322 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2015.

[15] G. S. Makanin. The problem of solvability of equations in a free semigroup. *English transl. in Math USSR Sbornik*, 32:147–236, 1977.

[16] K. S. Namjoshi and G. J. Narlikar. Robust and fast pattern matching for intrusion detection. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 740–748, 2010.

[17] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: from an abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, Nov. 2006.

[18] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the ACM*, 51(3):483–496, May 2004.

[19] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, and D. Song. A symbolic execution framework for JavaScript. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 513–528. IEEE Computer Society, 2010.

[20] M.-T. Trinh, D.-H. Chu, and J. Jaffar. S3: A symbolic string solver for vulnerability detection in web applications. In M. Yung and N. Li, editors, *Proceedings of the 21st ACM Conference on Computer and Communications Security*, 2014.

[21] Y. Zheng, X. Zhang, and V. Ganesh. Z3-str: A z3-based string solver for web application analysis. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 114–124. ACM, 2013.