Bibliography
Symbolic Executions Techniques for finding Bugs

Todo

October 1, 2018

References

[1] Romain Aissat, Frdric Voisin, and Burkhart Wolff. Infeasible paths elimination by symbolic execution techniques. In Jasmin Christian Blanchette and Stephan Merz, editors, *Interactive Theorem Proving*, pages 36–51, Cham, 2016. Springer International Publishing.

[2] Cristian Cadar and Koushik Sen. Symbolic execution for software testing: Three decades later. 56:82–90, 02 2013.

[3] Chen Chen, Baojiang Cui, Jinxin Ma, Runpu Wu, Jianchao Guo, and Wenqian Liu. A systematic review of fuzzing techniques. *Computers and Security*, 75:118 – 137, 2018.

[4] Manuel Costa, Miguel Castro, Lidong Zhou, Lintao Zhang, and Marcus Peinado. Bouncer: Securing software by blocking bad input. pages 117–130, Stevenson, Washington, USA, October 2007. Association for Computing Machinery, Inc.

[5] I. A. Dudina and A. A. Belevantsev. Using static symbolic execution to detect buffer overflows. *Programming and Computer Software*, 43(5):277–288, Sep 2017.

[6] Amal Khalil and Juergen Dingel. Chapter four - optimizing the symbolic execution of evolving rhapsody statecharts. volume 108 of *Advances in Computers*, pages 145 – 281. Elsevier, 2018.

[7] J.C. King. A new approach to program testing. 10:228–233, 06 1975.

[8] Dorel Lucanu, Vlad Rusu, and Andrei Arusoaie. A generic framework for symbolic execution: A coinductive approach. *Journal of Symbolic Computation*, 80:125 – 163, 2017. SI: Program Verification.

[9] M. Papadakis and N. Malevris. Automatic mutation test case generation via dynamic symbolic execution. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 121–130, Nov 2010.

[10] Vlad Rusu, Lydie du Bousquet, and Thierry Jeron. An approach to symbolic test generation. 09 2000.

[11] E. J. Schwartz, T. Avgerinos, and D. Brumley. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *2010 IEEE Symposium on Security and Privacy*, pages 317–331, May 2010.

[12] Asankhaya Sharma. Exploiting undefined behaviors for efficient symbolic execution. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 727–729, New York, NY, USA, 2014. ACM.

[13] David Trabish, Andrea Mattavelli, Noam Rinetzky, and Cristian Cadar. Chopped symbolic execution. In *International Conference on Software Engineering (ICSE 2018)*, 5 2018.

[14] Mathy Vanhoef and Frank Piessens. Symbolic execution of security protocol implementations: Handling cryptographic primitives. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, 2018. USENIX Association.

[15] Willem Visser, Corina S. Păsăreanu, and Sarfraz Khurshid. Test input generation with java pathfinder. In *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA '04, pages 97–107, New York, NY, USA, 2004. ACM.

[16] Guowei Yang, Suzette Person, Neha Rungta, and Sarfraz Khurshid. Directed incremental symbolic execution. 2014.

[17] Rui Zhang and Cynthia Sturton. A recursive strategy for symbolic execution to find exploits in hardware designs. pages 1–9, 06 2018.