Bardin, Sébastien; **Herrmann, Philippe**; **Védrine, Franck**

Refinement-based CFG reconstruction from unstructured programs. (English) Zbl 1317.68028

Jhala, Ranjit (ed.) et al., Verification, model checking, and abstract interpretation. 12th international conference, VMCAI 2011, Austin, TX, USA, January 23–25, 2011. Proceedings. Berlin: Springer (ISBN 978-3-642-18274-7/pbk). Lecture Notes in Computer Science 6538, 54-69 (2011).

Summary: This paper addresses the issue of recovering a both safe and precise approximation of the Control Flow Graph (CFG) of an unstructured program, typically an executable file. The problem is tackled in an original way, with a refinement-based static analysis working over finite sets of constant values. Requirement propagation allows the analysis to automatically adjust the domain precision only where it is needed, resulting in precise CFG recovery at moderate cost. First experiments, including an industrial case study, show that the method outperforms standard analyses in terms of precision, efficiency or robustness.

For the entire collection see [Zbl 1206.68013].

MSC:

68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)
68Q55 Semantics in the theory of computing

Software:

Jakstab; OSMOSE; SLAM

Full Text: DOI

References:

[1] Balakrishnan, G., Gruian, R., Reps, T.W., Teitelbaum, T.: CodeSurfer/x86–A platform for analyzing x86 executables. In: Bodik, R. (ed.) CC 2005. LNCS, vol.˜3443, pp. 250–254. Springer, Heidelberg (2005) · Zbl 1081.68604 · doi:10.1007/978-3-540-31985-6_19

[2] Baufreton, P., Heckmann, R.: Reliable and precise wcet and stack size determination for a real-life embedded application. In: ISoLA, Workshop On Leveraging Applications of Formal Methods, Verification and Validation, Poitiers-Futuroscope, France, December 12-14 (2007)

[3] Bardin, S., Herrmann, P.: Structural Testing of Executables. In: IEEE ICST 2008. IEEE Computer Society, Los Alamitos (2008)

[4] Bardin, S., Herrmann, P.: OSMOSE: Automatic Structural Testing of Executables. International Journal of Software Testing, Verification and Reliability (STVR), doi 10.1002/stvr.423

[5] Balakrishnan, G., Reps, T.W.: Analyzing memory accesses in x86 executables. In: Duesterwald, E. (ed.) CC 2004. LNCS, vol.˜2985, pp. 5–23. Springer, Heidelberg (2004) · Zbl 1125.68345 · doi:10.1007/978-3-540-24723-4_2

[6] Ball, T., Rajamani, S.: The SLAM project: Debugging system software via static analysis. In: POPL 2002. ACM, New York (2002)

[7] Balakrishnan, G., Reps, T.W.: DIVINE: DIscovering Variables IN Executables. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol.˜4349, pp. 1–28. Springer, Heidelberg (2007) · Zbl 05259422 · doi:10.1007/978-3-540-69738-1_1

[8] Balakrishnan, G., Reps, T.W.: Analyzing Stripped Device-Driver Executables. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol.˜4963, pp. 124–140. Springer, Heidelberg (2008) · Zbl 05262365 · doi:10.1007/978-3-540-78800-3_10

[9] Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: POPL 1977. ACM, New York (1977) · Zbl 1149.68389

[10] Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counter Example-Guided Abstraction Refinement for Symbolic Model Checking. Journal of the ACM˜50(5) (2003) · Zbl 1325.68145 · doi:10.1145/876638.876643

[11] Dhurjati, D., Das, M., Yang, Y.: Path-Sensitive Dataflow Analysis with Iterative Refinement. In: Yi, K. (ed.) SAS 2006. LNCS, vol.˜4134, pp. 425–442. Springer, Heidelberg (2006) · Zbl 05528268 · doi:10.1007/11823230_27

[12] Ferdinand, C., Heckmann, R.: aiT: worst case execution time prediction by static program analysis. In: IFIP Congress Topical Sessions 2004. Kluwer, Dordrecht (2004)

[13] Guyer, S.Z., Lin, C.: Client-driven pointer analysis. In: Cousot, R. (ed.) SAS 2003. LNCS, vol.˜2694. Springer, Heidelberg

(2003) · Zbl 1067.68543 · doi:10.1007/3-540-44898-5_12

[14] Godefroid, P., Levin, M.Y., Molnar, D.: Automated Whitebox Fuzz Testing. In: NDSS 2008. The Internet Society, San Diego (2008)

[15] Henzinger, T.A., Jhala, R., Majumbar, R., Sutre, G.: Lazy Abstraction. In: POPL 2002. ACM, New York (2002) · Zbl 1323.68374

[16] Handjieva, M., Tzolovski, S.: Refining static analyses by trace-based partitioning using control flow. In: Levi, G. (ed.) SAS 1998. LNCS, vol.˜1503, pp. 200–214. Springer, Heidelberg (1998) · doi:10.1007/3-540-49727-7_12

[17] Jeannet, B., Halbwachs, N., Raymond, P.: Dynamic partitioning in analyses of numerical properties. In: Cortesi, A., Filé, G. (eds.) SAS 1999. LNCS, vol.˜1694, p. 39. Springer, Heidelberg (1999) · doi:10.1007/3-540-48294-6_3

[18] Kinder, J., Veith, H.: Jakstab: A Static Analysis Platform for Binaries. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol.˜5123, pp. 423–427. Springer, Heidelberg (2008) · Zbl 05301130 · doi:10.1007/978-3-540-70545-1_40

[19] Kinder, J., Zuleger, F., Veith, H.: An Abstract Interpretation-Based Framework for Control Flow Reconstruction from Binaries. In: Logozzo, F., Peled, D.A., Zuck, L.D. (eds.) VMCAI 2008. LNCS, vol.˜4905. Springer, Heidelberg (2008) · Zbl 1206.68091

[20] Mauborgne, L., Rival, X.: Trace Partitioning in Abstract Interpretation Based Static Analyzers. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol.˜3444, pp. 5–20. Springer, Heidelberg (2005) · Zbl 1108.68427 · doi:10.1007/978-3-540-31987-0_2

[21] Myers, E.W.: Efficient Applicative Data Types. In: POPL 1984. ACM, New York (1984)

[22] Shivers, O.: Control-Flow Analysis in Scheme. In: PLDI 1988. ACM, New York (1988)

[23] Thakur, A.V., Lim, J., Lal, A., Burton, A., Driscoll, E., Elder, M., Andersen, T., Reps, T.W.: Directed Proof Generation for Machine Code. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol.˜6174, pp. 288–305. Springer, Heidelberg (2010) · Zbl 05772640 · doi:10.1007/978-3-642-14295-6_27

[24] IDA Pro homepage, http://www.hex-rays.com/idapro