Bibliography

Static Analysis

[1] Gary A. Kildall. "A Unified Approach to Global Program Optimization". In: *Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '73. Boston, Massachusetts: ACM, 1973, pp. 194–206. DOI: 10.1145/512927.512945. URL: http://doi.acm.org/10.1145/512927.512945.

[2] P. Cousot and R. Cousot. "Static determination of dynamic properties of programs". In: *Proceedings of the Second International Symposium on Programming*. Dunod, Paris, France, 1976, pp. 106–130.

[3] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. Los Angeles, California: ACM, 1977, pp. 238–252. DOI: 10.1145/512950.512973. URL: http://doi.acm.org/10.1145/512950.512973.

[4] Bjarne Steensgaard. "Points-to Analysis in Almost Linear Time". In: *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '96. St. Petersburg Beach, Florida, USA: ACM, 1996, pp. 32–41. ISBN: 0-89791-769-3. DOI: 10.1145/237721.237727. URL: http://doi.acm.org/10.1145/237721.237727.

[5] Manu Sridharan and Stephen J. Fink. "The Complexity of Andersen's Analysis in Practice". In: *Static Analysis: 16th International Symposium, SAS 2009, Los Angeles, CA, USA, August 9-11, 2009. Proceedings*. Ed. by Jens Palsberg and Zhendong Su. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 205–221. ISBN: 978-3-642-03237-0. DOI: 10.1007/978-3-642-03237-0_15. URL: https://doi.org/10.1007/978-3-642-03237-0_15.

[6] Ben Hardekopf and Calvin Lin. "The Ant and the Grasshopper: Fast and Accurate Pointer Analysis for Millions of Lines of Code". In: *Proceedings of the 28th ACM SIGPLAN Conference on Programming*

Language Design and Implementation. PLDI '07. San Diego, California, USA: ACM, 2007, pp. 290–299. ISBN: 978-1-59593-633-2. DOI: 10.1145/1250734.1250767. URL: http://doi.acm.org/10.1145/1250734.1250767.

[7] Ben Hardekopf and Calvin Lin. "Flow-sensitive Pointer Analysis for Millions of Lines of Code". In: *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*. CGO '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 289–298. ISBN: 978-1-61284-356-8. URL: http://dl.acm.org/citation.cfm?id=2190025.2190075.

[8] Jia Chen. "GSoC 2016: Improving alias analysis in LLVM". In: (2016). URL: http://dl.acm.org/citation.cfm?id=2190025.2190075.

[9] Maroua Maalej and Laure Gonnord. *Do we still need new Alias Analyses?* Research Report RR-8812. Université Lyon Claude Bernard / Laboratoire d'Informatique du Parallélisme, Nov. 2015. URL: https://hal.inria.fr/hal-01228581.

[10] Brian Hackett and Alex Aiken. "How is Aliasing Used in Systems Software?" In: *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. SIGSOFT '06/FSE-14. Portland, Oregon, USA: ACM, 2006, pp. 69–80. ISBN: 1-59593-468-5. DOI: 10.1145/1181775.1181785. URL: http://doi.acm.org/10.1145/1181775.1181785.

Program Verification

[11] Ralph L. London. "A View of Program Verification". In: *Proceedings of the International Conference on Reliable Software*. Los Angeles, California: ACM, 1975, pp. 534–545. DOI: 10.1145/800027.808477. URL: http://doi.acm.org/10.1145/800027.808477.

[12] Robert W. Floyd. "Assigning Meanings to Programs". In: *Program Verification: Fundamental Issues in Computer Science*. Ed. by Timothy R. Colburn, James H. Fetzer, and Terry L. Rankin. Dordrecht: Springer Netherlands, 1993, pp. 65–81. ISBN: 978-94-011-1793-7. DOI: 10.1007/978-94-011-1793-7_4. URL: https://doi.org/10.1007/978-94-011-1793-7_4.

[13] Thomas A. Henzinger et al. “Software Verification with BLAST”. In: *Model Checking Software: 10th International SPIN Workshop Portland, OR, USA, May 9–10, 2003 Proceedings*. Ed. by Thomas Ball and Sriram K. Rajamani. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 235–239. ISBN: 978-3-540-44829-7. DOI: 10.1007/3-540-44829-2_17. URL: https://doi.org/10.1007/3-540-44829-2_17.

[14] Shmuel M. Katz and Z Manna. *A Heuristic Approach to Program Verification*. Tech. rep. Stanford, CA, USA, 1973.

[15] Donald I. Good, Ralph L. London, and W. W. Bledsoe. “An Interactive Program Verification System”. In: *Proceedings of the International Conference on Reliable Software*. Los Angeles, California: ACM, 1975, pp. 482–492. DOI: 10.1145/800027.808472. URL: http://doi.acm.org/10.1145/800027.808472.

[16] Thomas Ball and Sriram K. Rajamani. “The SLAM Project: Debugging System Software via Static Analysis”. In: *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL ’02. Portland, Oregon: ACM, 2002, pp. 1–3. ISBN: 1-58113-450-9. DOI: 10.1145/503272.503274. URL: http://doi.acm.org/10.1145/503272.503274.

[17] Sagar Chaki et al. “Modular verification of software components in C”. In: *IEEE Transactions on Software Engineering* 30.6 (2004), pp. 388–402.

[18] Martin Leucker and Christian Schallhart. “A brief account of runtime verification”. In: *The Journal of Logic and Algebraic Programming* 78.5 (2009), pp. 293–303.

[19] David Evans et al. “LCLint: A Tool for Using Specifications to Check Code”. In: *Proceedings of the 2Nd ACM SIGSOFT Symposium on Foundations of Software Engineering*. SIGSOFT ’94. New Orleans, Louisiana, USA: ACM, 1994, pp. 87–96. ISBN: 0-89791-691-3. DOI: 10.1145/193173.195297. URL: http://doi.acm.org/10.1145/193173.195297.

[20] Edmund Clarke, Daniel Kroening, and Flavio Lerda. “A Tool for Checking ANSI-C Programs”. In: *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference,*

TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004. Proceedings. Ed. by Kurt Jensen and Andreas Podelski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 168–176. ISBN: 978-3-540-24730-2. DOI: 10.1007/978-3-540-24730-2_15. URL: https://doi.org/10.1007/978-3-540-24730-2_15.

Operating System Verification

[21] Alexandre Lissy, Stéphane Laurière, and Patrick Martineau. "Verifications around the Linux kernel". In: *Linux Symposium*. 2011, p. 37.

[22] Hendrik Post, Carsten Sinz, and Wolfgang Kuchlin. "Towards automatic software model checking of thousands of Linux modules-a case study with Avinux". In: *Software Testing, Verification and Reliability* 19.2 (2009), pp. 155–172.

[23] Hendrik Post and Wolfgang Küchlin. "Integrated static analysis for Linux device driver verification". In: *International Conference on Integrated Formal Methods*. Springer. 2007, pp. 518–537.

[24] Thomas Witkowski et al. "Model Checking Concurrent Linux Device Drivers". In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ASE '07. Atlanta, Georgia, USA: ACM, 2007, pp. 501–504. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321719. URL: http://doi.acm.org/10.1145/1321631.1321719.

[25] I. S. Zakharov et al. "Configurable Toolset for Static Verification of Operating Systems Kernel Modules". In: *Program. Comput. Softw.* 41.1 (Jan. 2015), pp. 49–64. ISSN: 0361-7688. DOI: 10.1134/S0361768815010065. URL: http://dx.doi.org/10.1134/S0361768815010065.

[26] Gerwin Klein et al. "seL4: Formal Verification of an OS Kernel". In: *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*. SOSP '09. Big Sky, Montana, USA: ACM, 2009, pp. 207–220. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629596. URL: http://doi.acm.org/10.1145/1629575.1629596.

Model Checking

[27] Ranjit Jhala and Rupak Majumdar. "Software model checking". In: *ACM Computing Surveys (CSUR)* 41.4 (2009), p. 21.

[28] Edmund Clarke et al. "SATABS: SAT-based predicate abstraction for ANSI-C". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2005, pp. 570–574.

[29] Thomas Ball et al. "Automatic Predicate Abstraction of C Programs". In: *Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation*. PLDI '01. Snowbird, Utah, USA: ACM, 2001, pp. 203–213. ISBN: 1-58113-414-2. DOI: 10.1145/378795.378846. URL: http://doi.acm.org/10.1145/378795.378846.

[30] Raivo Laanemets. "Software Model Checking and The BLAST Toolkit". In: (2009).

[31] Madanlal Musuvathi et al. "CMC: A pragmatic approach to model checking real code". In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 75–88.

Using Dynamic Information for Static Analysis

[32] "Static Exploration of Taint-Style Vulnerabilities Found by Fuzzing". In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. Vancouver, BC: USENIX Association, 2017. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/shastry.

[33] Markus Mock et al. "Dynamic Points-to Sets: A Comparison with Static Analyses and Potential Applications in Program Understanding and Optimization". In: *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*. PASTE '01. Snowbird, Utah, USA: ACM, 2001, pp. 66–72. ISBN: 1-58113-413-4. DOI: 10.1145/379605.379671. URL: http://doi.acm.org/10.1145/379605.379671.

[34] Markus Mock et al. “Improving Program Slicing with Dynamic Points-to Data”. In: *Proceedings of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering*. SIGSOFT ’02/FSE-10. Charleston, South Carolina, USA: ACM, 2002, pp. 71–80. ISBN: 1-58113-514-9. DOI: 10.1145/587051.587062. URL: http://doi.acm.org/10.1145/587051.587062.

[35] Axel Gross. “Evaluation of dynamic Points-To Analysis”. In: (2004).

Vulnerability Detection

[36] Al Bessey et al. “A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World”. In: *Commun. ACM* 53.2 (Feb. 2010), pp. 66–75. ISSN: 0001-0782. DOI: 10.1145/1646353.1646374. URL: http://doi.acm.org/10.1145/1646353.1646374.

[37] William R. Bush, Jonathan D. Pincus, and David J. Sielaff. “A Static Analyzer for Finding Dynamic Programming Errors”. In: *Softw. Pract. Exper.* 30.7 (June 2000), pp. 775–802. ISSN: 0038-0644. DOI: 10.1002/(SICI)1097-024X(200006)30:7<775::AID-SPE309>3.0.CO;2-H. URL: http://dx.doi.org/10.1002/(SICI)1097-024X(200006)30:7%3C775::AID-SPE309%3E3.0.CO;2-H.

[38] Nurit Dor, Michael Rodeh, and Mooly Sagiv. “CSSV: Towards a Realistic Tool for Statically Detecting All Buffer Overflows in C”. In: *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*. PLDI ’03. San Diego, California, USA: ACM, 2003, pp. 155–167. ISBN: 1-58113-662-5. DOI: 10.1145/781131.781149. URL: http://doi.acm.org/10.1145/781131.781149.

[39] Vinod Ganapathy et al. “Buffer Overrun Detection Using Linear Programming and Static Analysis”. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*. CCS ’03. Washington D.C., USA: ACM, 2003, pp. 345–354. ISBN: 1-58113-738-9. DOI: 10.1145/948109.948155. URL: http://doi.acm.org/10.1145/948109.948155.

[40] Yungbum Jung and Kwangkeun Yi. "Practical Memory Leak Detector Based on Parameterized Procedural Summaries". In: *Proceedings of the 7th International Symposium on Memory Management*. ISMM '08. Tucson, AZ, USA: ACM, 2008, pp. 131–140. ISBN: 978-1-60558-134-7. DOI: 10.1145/1375634.1375653. URL: http://doi.acm.org/10.1145/1375634.1375653.

[41] Yichen Xie and Alex Aiken. "Context-and path-sensitive memory leak detection". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 30. 5. ACM. 2005, pp. 115–125.

[42] Yichen Xie, Andy Chou, and Dawson Engler. "ARCHER: Using Symbolic, Path-sensitive Analysis to Detect Memory Access Errors". In: *Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ESEC/FSE-11. Helsinki, Finland: ACM, 2003, pp. 327–336. ISBN: 1-58113-743-5. DOI: 10.1145/940071.940115. URL: http://doi.acm.org/10.1145/940071.940115.

[43] Misha Zitser, Richard Lippmann, and Tim Leek. "Testing static analysis tools using exploitable buffer overflows from open source code". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 29. 6. ACM. 2004, pp. 97–106.

[44] Jiří Slabỳ. "Automatic Bug-finding Techniques for Linux Kernel". In: ().

[45] Yannick Moy, Nikolaj Bjørner, and David Sielaff. "Modular bug-finding for integer overflows in the large: Sound, efficient, bit-precise static analysis". In: *Microsoft Research* 11 (2009).

[46] Dipanwita Sarkar et al. "Flow-insensitive static analysis for detecting integer anomalies in programs". In: *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering*. ACTA Press. 2007, pp. 334–340.

[47] Xi Wang et al. "Improving Integer Security for Systems with KINT." In: *OSDI*. Vol. 12. 2012, pp. 163–177.

[48] *Verifying absence of integer overflow*. https://critical.eschertech.com/2010/06/07/verifying-absence-of-integer-overflow/. Accessed: 2017-08-30.

[49] Yichen Xie and Alexander Aiken. "Saturn: A SAT-Based Tool for Bug Detection." In: *CAV*. Vol. 3576. Springer. 2005, pp. 139–143.

[50] Dawson Engler et al. "Bugs As Deviant Behavior: A General Approach to Inferring Errors in Systems Code". In: *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*. SOSP '01. Banff, Alberta, Canada: ACM, 2001, pp. 57–72. ISBN: 1-58113-389-8. DOI: 10.1145/502034.502041. URL: http://doi.acm.org/10.1145/502034.502041.

[51] V. P. Ivannikov et al. "Static Analyzer Svace for Finding Defects in a Source Program Code". In: *Program. Comput. Softw.* 40.5 (Sept. 2014), pp. 265–275. ISSN: 0361-7688. DOI: 10.1134/S0361768814050041. URL: http://dx.doi.org/10.1134/S0361768814050041.

[52] Hao Chen and David Wagner. "MOPS: an infrastructure for examining security properties of software". In: *Proceedings of the 9th ACM conference on Computer and communications security*. ACM. 2002, pp. 235–244.

[53] *Current pattern based vulnerability detection tools: BOON, ITS4, RATS, PScan, Flawfinder, UNO, FlexeLint (PC-Lint), Viva64, Parasoft C++test*. https://www.viva64.com/en/a/0028. Accessed: 2017-08-30.

[54] *Current semantic based vulnerability detection tools: CodeSurfer, Frama-C, KlocWork K7, Coverity*. https://www.viva64.com/en/a/0028. Accessed: 2017-08-30.

Vulnerability Detection in Kernel

[55] Haogang Chen et al. "Linux Kernel Vulnerabilities: State-of-the-art Defenses and Open Problems". In: *Proceedings of the Second Asia-Pacific Workshop on Systems*. APSys '11. Shanghai, China: ACM, 2011, 5:1–5:5. ISBN: 978-1-4503-1179-3. DOI: 10.1145/2103799.2103805. URL: http://doi.acm.org/10.1145/2103799.2103805.

[56] Suhabe Bugrara and Alex Aiken. "Verifying the safety of user pointer dereferences". In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 325–338.

[57] Zachary R Anderson et al. "Beyond Bug-Finding: Sound Program Analysis for Linux." In: *HotOS*. 2007.

[58] Thomas Ball et al. “Thorough static analysis of device drivers”. In: *ACM SIGOPS Operating Systems Review* 40.4 (2006), pp. 73–85.

[59] Junfeng Yang et al. “Using model checking to find serious file system errors”. In: *ACM Transactions on Computer Systems (TOCS)* 24.4 (2006), pp. 393–423.

[60] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. “LOCKSMITH: Practical Static Race Detection for C”. In: *ACM Trans. Program. Lang. Syst.* 33.1 (Jan. 2011), 3:1–3:55. ISSN: 0164-0925. DOI: 10.1145/1889997.1890000. URL: http://doi.acm.org/10.1145/1889997.1890000.

[61] Yichen Xie and Alex Aiken. “Scalable Error Detection Using Boolean Satisfiability”. In: *Proceedings of the 32Nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL ’05. Long Beach, California, USA: ACM, 2005, pp. 351–363. ISBN: 1-58113-830-X. DOI: 10.1145/1040305.1040334. URL: http://doi.acm.org/10.1145/1040305.1040334.

[62] Sidney Amani et al. “Static Analysis of Device Drivers: We Can Do Better!” In: *Proceedings of the Second Asia-Pacific Workshop on Systems*. APSys ’11. Shanghai, China: ACM, 2011, 8:1–8:5. ISBN: 978-1-4503-1179-3. DOI: 10.1145/2103799.2103809. URL: http://doi.acm.org/10.1145/2103799.2103809.

User-annotation based Vulnerability Detection

[63] Ken Ashcraft and Dawson Engler. “Using programmer-written compiler extensions to catch security holes”. In: *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE. 2002, pp. 143–159.

[64] Junfeng Yang et al. “MECA: an extensible, expressive system and language for statically checking security properties”. In: *Proceedings of the 10th ACM conference on Computer and communications security*. ACM. 2003, pp. 321–334.

[65] Umesh Shankar et al. "Detecting Format String Vulnerabilities with Type Qualifiers". In: *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*. SSYM'01. Washington, D.C.: USENIX Association, 2001. URL: http://dl.acm.org/citation.cfm?id=1251327.1251343.

[66] Rob Johnson and David Wagner. "Finding User/Kernel Pointer Bugs with Type Inference". In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. San Diego, CA: USENIX Association, 2004, pp. 9–9. URL: http://dl.acm.org/citation.cfm?id=1251375.1251384.

[67] Xiaolan Zhang, Antony Edwards, and Trent Jaeger. "Using CQUAL for Static Analysis of Authorization Hook Placement". In: *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 33–48. ISBN: 1-931971-00-5. URL: http://dl.acm.org/citation.cfm?id=647253.720279.

[68] Ebrima N. Ceesay et al. "Using Type Qualifiers to Analyze Untrusted Integers and Detecting Security Flaws in C Programs". In: *Proceedings of the Third International Conference on Detection of Intrusions and Malware & Vulnerability Assessment*. DIMVA'06. Berlin, Germany: Springer-Verlag, 2006, pp. 1–16. ISBN: 3-540-36014-X, 978-3-540-36014-8. DOI: 10.1007/11790754_1. URL: http://dx.doi.org/10.1007/11790754_1.

Techniques to handle False alarms

[69] Ted Kremenek and Dawson Engler. "Z-ranking: Using statistical analysis to counter the impact of static analysis approximations". In: *International Static Analysis Symposium*. Springer. 2003, pp. 295–315.

[70] Ted Kremenek et al. "Correlation exploitation in error ranking". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 29. 6. ACM. 2004, pp. 83–93.

[71] Yungbum Jung et al. "Taming false alarms from a domain-unaware C analyzer by a bayesian statistical post analysis". In: *Static Analysis* (2005), pp. 203–217.

[72] David Larochelle, David Evans, et al. "Statically Detecting Likely Buffer Overflow Vulnerabilities." In: *USENIX Security Symposium*. Vol. 32. Washington DC. 2001.

[73] J Kosina. "Fighting Security Bugs in the Linux Kernel". In: *WDS*. Vol. 7, pp. 64–71.

[74] Fabian Yamaguchi, Markus Lottmann, and Konrad Rieck. "Generalized Vulnerability Extrapolation Using Abstract Syntax Trees". In: *Proceedings of the 28th Annual Computer Security Applications Conference*. ACSAC '12. Orlando, Florida, USA: ACM, 2012, pp. 359–368. ISBN: 978-1-4503-1312-4. DOI: 10.1145/2420950.2421003. URL: http://doi.acm.org/10.1145/2420950.2421003.

[75] Fabian Yamaguchi et al. "Automatic Inference of Search Patterns for Taint-Style Vulnerabilities". In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. SP '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 797–812. ISBN: 978-1-4673-6949-7. DOI: 10.1109/SP.2015.54. URL: http://dx.doi.org/10.1109/SP.2015.54.

[76] David Evans and David Larochelle. "Improving Security Using Extensible Lightweight Static Analysis". In: *IEEE Softw.* 19.1 (Jan. 2002), pp. 42–51. ISSN: 0740-7459. DOI: 10.1109/52.976940. URL: http://dx.doi.org/10.1109/52.976940.

[77] Julien Vanegue and Shuvendu K. Lahiri. "Towards Practical Reactive Security Audit Using Extended Static Checkers". In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy*. SP '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 33–47. ISBN: 978-0-7695-4977-4. DOI: 10.1109/SP.2013.12. URL: http://dx.doi.org/10.1109/SP.2013.12.

[78] Fabian Yamaguchi et al. "Modeling and Discovering Vulnerabilities with Code Property Graphs". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. SP '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 590–604. ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.44. URL: http://dx.doi.org/10.1109/SP.2014.44.

[79] Salva Peiró et al. “Detecting stack based kernel information leaks”. In: *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*. Springer. 2014, pp. 321–331.

[80] Fabian Yamaguchi et al. “Chucky: Exposing Missing Checks in Source Code for Vulnerability Discovery”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: ACM, 2013, pp. 499–510. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516665. URL: http://doi.acm.org/10.1145/2508859.2516665.

Vulnerability finding in Software Repositories

[81] Zhenmin Li et al. “CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code”. In: *IEEE Trans. Softw. Eng.* 32.3 (Mar. 2006), pp. 176–192. ISSN: 0098-5589. DOI: 10.1109/TSE.2006.28. URL: http://dx.doi.org/10.1109/TSE.2006.28.

[82] Jiyong Jang, Abeer Agrawal, and David Brumley. “ReDeBug: Finding Unpatched Code Clones in Entire OS Distributions”. In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 48–62. ISBN: 978-0-7695-4681-0. DOI: 10.1109/SP.2012.13. URL: http://dx.doi.org/10.1109/SP.2012.13.

[83] Benjamin Livshits and Thomas Zimmermann. “DynaMine: Finding Common Error Patterns by Mining Software Revision Histories”. In: *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ESEC/FSE-13. Lisbon, Portugal: ACM, 2005, pp. 296–305. ISBN: 1-59593-014-0. DOI: 10.1145/1081706.1081754. URL: http://doi.acm.org/10.1145/1081706.1081754.

[84] Shuvendu K. Lahiri, Kapil Vaswani, and C A. R. Hoare. “Differential Static Analysis: Opportunities, Applications, and Challenges”. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. FoSER '10. Santa Fe, New Mexico, USA: ACM, 2010, pp. 201–204. ISBN: 978-1-4503-0427-6. DOI: 10.1145/1882362.1882405. URL: http://doi.acm.org/10.1145/1882362.1882405.

[85] Shuvendu K. Lahiri et al. "SYMDIFF: A Language-agnostic Semantic Diff Tool for Imperative Programs". In: *Proceedings of the 24th International Conference on Computer Aided Verification*. CAV'12. Berkeley, CA: Springer-Verlag, 2012, pp. 712–717. ISBN: 978-3-642-31423-0. DOI: 10.1007/978-3-642-31424-7_54. URL: http://dx.doi.org/10.1007/978-3-642-31424-7_54.

Soundy analysis for Vulnerability Detection

[86] Benjamin Livshits et al. "In defense of soundiness: A manifesto". In: *Communications of the ACM* 58.2 (2015), pp. 44–46.

[87] V. Benjamin Livshits and Monica S. Lam. "Tracking Pointers with Path and Context Sensitivity for Bug Detection in C Programs". In: *Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ESEC/FSE-11. Helsinki, Finland: ACM, 2003, pp. 317–326. ISBN: 1-58113-743-5. DOI: 10.1145/940071.940114. URL: http://doi.acm.org/10.1145/940071.940114.

[88] Kihong Heo, Hakjoo Oh, and Kwangkeun Yi. "Machine-learning-guided Selectively Unsound Static Analysis". In: *Proceedings of the 39th International Conference on Software Engineering*. ICSE '17. Buenos Aires, Argentina: IEEE Press, 2017, pp. 519–529. ISBN: 978-1-5386-3868-2. DOI: 10.1109/ICSE.2017.54. URL: https://doi.org/10.1109/ICSE.2017.54.

[89] Aravind Machiry et al. "DR. CHECKER: A Soundy Analysis for Linux Kernel Drivers". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1007–1024. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/machiry.

Interactive analysis for Vulnerability Detection

[90] Ravi Mangal et al. "A User-guided Approach to Program Analysis". In: *Proceedings of the 2015 10th Joint Meeting on Foundations*

of Software Engineering. ESEC/FSE 2015. Bergamo, Italy: ACM, 2015, pp. 462–473. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786851. URL: http://doi.acm.org/10.1145/2786805.2786851.

[91] Xin Zhang et al. "Effective Interactive Resolution of Static Analysis Alarms". In: *Proceedings of the ACM on Programming Languages* 1.1 (2017).

Enforcing program properties

[92] Dinakar Dhurjati, Sumant Kowshik, and Vikram Adve. "SAFECode: Enforcing Alias Analysis for Weakly Typed Languages". In: *Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '06. Ottawa, Ontario, Canada: ACM, 2006, pp. 144–157. ISBN: 1-59593-320-4. DOI: 10.1145/1133981.1133999. URL: http://doi.acm.org/10.1145/1133981.1133999.

[93] Walter Chang, Brandon Streiff, and Calvin Lin. "Efficient and Extensible Security Enforcement Using Dynamic Data Flow Analysis". In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. CCS '08. Alexandria, Virginia, USA: ACM, 2008, pp. 39–50. ISBN: 978-1-59593-810-7. DOI: 10.1145/1455770.1455778. URL: http://doi.acm.org/10.1145/1455770.1455778.

Runtime Verification

[94] Reed Milewicz et al. "Runtime Checking C Programs". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 2107–2114. ISBN: 978-1-4503-3196-8. DOI: 10.1145/2695664.2695906. URL: http://doi.acm.org/10.1145/2695664.2695906.

[95] Harish Patil and Charles Fischer. "Low-cost, Concurrent Checking of Pointer and Array Accesses in C Programs". In: *Softw. Pract. Exper.* 27.1 (Jan. 1997), pp. 87–110. ISSN: 0038-0644. DOI: 10.1002/(SICI)1097-024X(199701)27:1<87::AID-SPE78>3.0.CO;2-P. URL: http://dx.doi.org/10.1002/(SICI)1097-024X(199701)27:1%3C87::AID-SPE78%3E3.0.CO;2-P.

[96] Olatunji Ruwase and Monica S. Lam. "A Practical Dynamic Buffer Overflow Detector". In: *In Proceedings of the 11th Annual Network and Distributed System Security Symposium*. 2004, pp. 159–169.

[97] Suan Hsi Yong and Susan Horwitz. "Protecting C Programs from Attacks via Invalid Pointer Dereferences". In: *Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ESEC/FSE-11. Helsinki, Finland: ACM, 2003, pp. 307–316. ISBN: 1-58113-743-5. DOI: 10.1145/940071.940113. URL: http://doi.acm.org/10.1145/940071.940113.

[98] Wei Xu, Daniel C. DuVarney, and R. Sekar. "An Efficient and Backwards-compatible Transformation to Ensure Memory Safety of C Programs". In: *Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering*. SIGSOFT '04/FSE-12. Newport Beach, CA, USA: ACM, 2004, pp. 117–126. ISBN: 1-58113-855-5. DOI: 10.1145/1029894.1029913. URL: http://doi.acm.org/10.1145/1029894.1029913.

Control Flow Integrity

[99] Martín Abadi et al. "Control-flow Integrity". In: *Proceedings of the 12th ACM Conference on Computer and Communications Security*. CCS '05. Alexandria, VA, USA: ACM, 2005, pp. 340–353. ISBN: 1-59593-226-7. DOI: 10.1145/1102120.1102165. URL: http://doi.acm.org/10.1145/1102120.1102165.

[100] John Criswell, Nathan Dautenhahn, and Vikram Adve. "KCoFI: Complete Control-Flow Integrity for Commodity Operating System Kernels". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. SP '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 292–307. ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.26. URL: http://dx.doi.org/10.1109/SP.2014.26.

[101] Lucas Davi, Patrick Koeberl, and Ahmad-Reza Sadeghi. "Hardware-Assisted Fine-Grained Control-Flow Integrity: Towards Efficient Protection of Embedded Systems Against Software Exploitation". In: *Proceedings of the 51st Annual Design Automation Conference*. DAC

'14. San Francisco, CA, USA: ACM, 2014, 133:1–133:6. ISBN: 978-1-4503-2730-5. DOI: 10.1145/2593069.2596656. URL: http://doi.acm.org/10.1145/2593069.2596656.

[102] Caroline Tice et al. "Enforcing Forward-edge Control-flow Integrity in GCC & LLVM". In: *Proceedings of the 23rd USENIX Conference on Security Symposium*. SEC'14. San Diego, CA: USENIX Association, 2014, pp. 941–955. ISBN: 978-1-931971-15-7. URL: http://dl.acm.org/citation.cfm?id=2671225.2671285.

[103] Yubin Xia et al. "CFIMon: Detecting violation of control flow integrity using performance counters". In: *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*. IEEE. 2012, pp. 1–12.

[104] Vishwath Mohan et al. "Opaque Control-Flow Integrity." In: *NDSS*. Vol. 26. 2015, pp. 27–30.

[105] Ali Jose Mashtizadeh et al. "CCFI: Cryptographically Enforced Control Flow Integrity". In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: ACM, 2015, pp. 941–951. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813676. URL: http://doi.acm.org/10.1145/2810103.2813676.

[106] Xinyang Ge et al. "Fine-grained control-flow integrity for kernel software". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016, pp. 179–194.

[107] Vasileios P Kemerlis, Georgios Portokalidis, and Angelos D Keromytis. "kGuard: Lightweight Kernel Protection against Return-to-User Attacks." In: *USENIX Security Symposium*. 2012, pp. 459–474.

Breaking Control Flow Integrity

[108] Enes Göktas et al. "Out of Control: Overcoming Control-Flow Integrity". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. SP '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 575–589. ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.43. URL: http://dx.doi.org/10.1109/SP.2014.43.

[109] Lucas Davi et al. “Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection.” In: *USENIX Security Symposium*. Vol. 2014. 2014.

[110] Isaac Evans et al. “Control Jujutsu: On the Weaknesses of Fine-Grained Control Flow Integrity”. In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. Denver, Colorado, USA: ACM, 2015, pp. 901–913. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813646. URL: http://doi.acm.org/10.1145/2810103.2813646.

[111] Mauro Conti et al. “Losing Control: On the Effectiveness of Control-Flow Integrity Under Stack Attacks”. In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. Denver, Colorado, USA: ACM, 2015, pp. 952–963. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813671. URL: http://doi.acm.org/10.1145/2810103.2813671.

[112] Nicholas Carlini et al. “Control-Flow Bending: On the Effectiveness of Control-Flow Integrity.” In: *USENIX Security Symposium*. 2015, pp. 161–176.

Data Flow Integrity

[113] Hong Hu et al. “Data-oriented programming: On the expressiveness of non-control data attacks”. In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 969–986.

[114] Miguel Castro, Manuel Costa, and Tim Harris. “Securing Software by Enforcing Data-flow Integrity”. In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. OSDI ’06. Seattle, Washington: USENIX Association, 2006, pp. 147–160. ISBN: 1-931971-47-1. URL: http://dl.acm.org/citation.cfm?id=1298455.1298470.

Instrumentation for specific program properties

[115] Kangjie Lu et al. "UniSan: Proactive kernel memory initialization to eliminate data leakages". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 920–932.

[116] Konstantin Serebryany et al. "AddressSanitizer: A Fast Address Sanity Checker." In: *USENIX Annual Technical Conference*. 2012, pp. 309–318.

[117] Konstantin Serebryany and Timur Iskhodzhanov. "ThreadSanitizer: Data Race Detection in Practice". In: *Proceedings of the Workshop on Binary Instrumentation and Applications*. WBIA '09. New York, New York, USA: ACM, 2009, pp. 62–71. ISBN: 978-1-60558-793-6. DOI: 10.1145/1791194.1791203. URL: http://doi.acm.org/10.1145/1791194.1791203.

[118] *LeakSanitizer*. https://github.com/google/sanitizers/wiki/AddressSanitizerLeakSanitizer. Accessed: 2017-08-30.

[119] Byoungyoung Lee et al. "Preventing Use-after-free with Dangling Pointers Nullification." In: *NDSS*. 2015.

[120] Juan Caballero et al. "Undangle: Early Detection of Dangling Pointers in Use-after-free and Double-free Vulnerabilities". In: *Proceedings of the 2012 International Symposium on Software Testing and Analysis*. ISSTA 2012. Minneapolis, MN, USA: ACM, 2012, pp. 133–143. ISBN: 978-1-4503-1454-1. DOI: 10.1145/2338965.2336769. URL: http://doi.acm.org/10.1145/2338965.2336769.

[121] Istvan Haller et al. "TypeSan: Practical Type Confusion Detection". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: ACM, 2016, pp. 517–528. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978405. URL: http://doi.acm.org/10.1145/2976749.2978405.

[122] Byoungyoung Lee et al. "Type Casting Verification: Stopping an Emerging Attack Vector." In: *USENIX Security Symposium*. 2015, pp. 81–96.

[123] *UndefinedBehaviorSanitizer*. Accessed: 2017-08-30. URL: https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html.

Dynamic Bounds Checking

[124] Santosh Nagarakatte et al. "SoftBound: Highly compatible and complete spatial memory safety for C". In: *ACM Sigplan Notices* 44.6 (2009), pp. 245–258.

[125] Joe Devietti et al. "Hardbound: architectural support for spatial safety of the C programming language". In: *ACM SIGARCH Computer Architecture News*. Vol. 36. 1. ACM. 2008, pp. 103–114.

[126] Dinakar Dhurjati and Vikram Adve. "Backwards-compatible Array Bounds Checking for C with Very Low Overhead". In: *Proceedings of the 28th International Conference on Software Engineering*. ICSE '06. Shanghai, China: ACM, 2006, pp. 162–171. ISBN: 1-59593-375-1. DOI: 10.1145/1134285.1134309. URL: http://doi.acm.org/10.1145/1134285.1134309.

[127] Zili Shao et al. "Efficient array & pointer bound checking against buffer overflow attacks via hardware/software". In: *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*. Vol. 1. IEEE. 2005, pp. 780–785.

[128] Periklis Akritidis et al. "Baggy Bounds Checking: An Efficient and Backwards-Compatible Defense against Out-of-Bounds Errors." In: *USENIX Security Symposium*. 2009, pp. 51–66.

[129] Gregory J. Duck and Roland H. C. Yap. "Heap Bounds Protection with Low Fat Pointers". In: *Proceedings of the 25th International Conference on Compiler Construction*. CC 2016. Barcelona, Spain: ACM, 2016, pp. 132–142. ISBN: 978-1-4503-4241-4. DOI: 10.1145/2892208.2892212. URL: http://doi.acm.org/10.1145/2892208.2892212.

[130] Gregory J Duck, Roland HC Yap, and Lorenzo Cavallaro. "Stack bounds protection with low fat pointers". In: *Symposium on Network and Distributed System Security*. 2017.