References

[1] B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar, "Native Client: A Sandbox for Portable, Untrusted x86 Native Code," in *2009 30th IEEE Symposium on Security and Privacy*, May 2009. doi: 10.1109/SP.2009.25. ISSN 2375-1207 pp. 79–93.

[2] A. Zakai, "Emscripten: An LLVM-to-JavaScript compiler," in *OOPSLA Companion*, 2011. doi: 10.1145/2048147.2048224

[3] Greggman, "Thoughts on asm.js vs PNaCl," 2013. [Online]. Available: https://games.greggman.com/game/thoughts-on-asm-js-vs-pnacl/

[4] O. Katz, Y. Olshaker, Y. Goldberg, and E. Yahav, "Towards Neural Decompilation," *arXiv:1905.08325 [cs]*, May 2019. [Online]. Available: http://arxiv.org/abs/1905.08325

[5] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*. USA: Addison-Wesley Longman Publishing Co., Inc., 1986. ISBN 978-0-201-10088-4

[6] "Introduction — WebAssembly 1.1." [Online]. Available: https://webassembly.github.io/spec/core/intro/introduction.html#wasm

[7] G. Neubig, "Neural Machine Translation and Sequence-to-sequence Models: A Tutorial," *arXiv:1703.01619 [cs, stat]*, Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.01619

[8] M. J. Harrold, G. Rothermel, and A. Orso, "Representation and Analysis of Software," p. 19.

[9] D. Chisnall, "Modern Intermediate Representations (IR)," p. 166, Jun. 2017.

[10] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien, “Bringing the web up to speed with WebAssembly,” *ACM SIGPLAN Notices*, vol. 52, no. 6, pp. 185–200, Jun. 2017. doi: 10.1145/3140587.3062363. [Online]. Available: https://doi.org/10.1145/3140587.3062363

[11] A. Hilbig, D. Lehmann, and M. Pradel, “An Empirical Study of Real-World WebAssembly Binaries,” p. 12, 2021.

[12] Y. Shi, K. Casey, M. A. Ertl, and D. Gregg, “Virtual machine showdown: Stack versus registers,” *ACM Transactions on Architecture and Code Optimization*, vol. 4, no. 4, pp. 1–36, Jan. 2008. doi: 10.1145/1328195.1328197. [Online]. Available: https://dl.acm.org/doi/10.1145/1328195.1328197

[13] S. Hochreiter, “Untersuchungen zu dynamischen neuronalen Netzen,” *Diploma, Technische Universität München*, vol. 91, no. 1, 1991.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” *arXiv:1409.3215 [cs]*, Dec. 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[15] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. doi: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” *arXiv:1706.03762 [cs]*, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[17] M.-T. Luong, H. Pham, and C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” *arXiv:1508.04025 [cs]*, Sep. 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[18] H. Xue, S. Sun, G. Venkataramani, and T. Lan, “Machine Learning-Based Analysis of Program Binaries: A Comprehensive Study,” *IEEE Access*, vol. 7, pp. 65 889–65 912, 2019. doi: 10.1109/ACCESS.2019.2917668

[19] M. Allamanis, H. Peng, and C. Sutton, “A Convolutional Attention Network for Extreme Summarization of Source Code,”

arXiv:1602.03001 [cs], May 2016. [Online]. Available: http://arxiv.org/abs/1602.03001

[20] P. Loyola, E. Marrese-Taylor, and Y. Matsuo, "A Neural Architecture for Generating Natural Language Descriptions from Source Code Changes," *arXiv:1704.04856 [cs]*, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1704.04856

[21] X. Chen, C. Liu, and D. Song, "Tree-to-tree Neural Networks for Program Translation," Feb. 2018. [Online]. Available: https://arxiv.org/abs/1802.03691v3

[22] U. Alon, R. Sadaka, O. Levy, and E. Yahav, "Structural Language Models of Code," *arXiv:1910.00577 [cs, stat]*, Jul. 2020. [Online]. Available: http://arxiv.org/abs/1910.00577

[23] B. Roziere, M.-A. Lachaux, M. Szafraniec, and G. Lample, "DOBF: A Deobfuscation Pre-Training Objective for Programming Languages," *arXiv:2102.07492 [cs]*, Feb. 2021. [Online]. Available: http://arxiv.org/abs/2102.07492

[24] Y. Yang, X. Xia, D. Lo, and J. Grundy, "A Survey on Deep Learning for Software Engineering," *arXiv:2011.14597 [cs]*, Nov. 2020. [Online]. Available: http://arxiv.org/abs/2011.14597

[25] M. Monperrus, "The Living Review on Automated Program Repair," Dec. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01956501

[26] W. A. Sassaman, "A computer program to translate machine language into FORTRAN," in *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, ser. AFIPS '66 (Spring). New York, NY, USA: Association for Computing Machinery, Apr. 1966. doi: 10.1145/1464182.1464211. ISBN 978-1-4503-7892-5 pp. 235–239. [Online]. Available: https://doi.org/10.1145/1464182.1464211

[27] C. Cifuentes, "Reverse compilation techniques," Ph.D. dissertation, 1994. [Online]. Available: /paper/Reverse-compilation-techniques-Cifuentes/2f25cb60ce7c12d067c0fe3dafef7e9a7896ea3c

[28] A. Mycroft, "Type-Based Decompilation (or Program Reconstruction via Type Reconstruction)," in *Proceedings of the 8th European Symposium*

on Programming Languages and Systems, ser. ESOP ’99. Berlin, Heidelberg: Springer-Verlag, Mar. 1999. ISBN 978-3-540-65699-9 pp. 208–223.

[29] M. Emmerik, “Static Single Assignment for Decompilation,” *undefined*, 2007. [Online]. Available: https://www.semanticscholar.org/paper/Static-Single-Assignment-for-Decompilation-Emmerik/6181cec77dc6fc26973f8d0386d587cf6978609a

[30] “Boomerang Decompiler.” [Online]. Available: http://boomerang.sourceforge.net/

[31] B. C. Housel, “A study of decompiling machine languages into high-level machine independent languages,” Ph.D. dissertation, Purdue University, USA, 1973.

[32] A. Fokin, E. Derevenetc, A. Chernov, and K. Troshina, “SmartDec: Approaching C++ Decompilation,” in *2011 18th Working Conference on Reverse Engineering*, Oct. 2011. doi: 10.1109/WCRE.2011.49. ISSN 2375-5369 pp. 347–356.

[33] I. Guilfanov, “Decompilation gets real – Hex Rays,” 2007. [Online]. Available: https://www.hex-rays.com/blog/decompilation-gets-real/

[34] D. Brumley, J. Lee, E. J. Schwartz, and M. Woo, “Native x86 Decompilation Using Semantics-Preserving Structural Analysis and Iterative Control-Flow Structuring,” in *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013. ISBN 978-1-931971-03-4 pp. 353–368. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/schwartz

[35] J. Lee, T. Avgerinos, and D. Brumley, “TIE: Principled Reverse Engineering of Types in Binary Programs,” p. 18, 2011.

[36] K. Yakdan, S. Eschweiler, E. Gerhards-Padilla, and M. Smith, “No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations,” in *NDSS*, 2015. doi: 10.14722/NDSS.2015.23185

[37] K. Yakdan, S. Dechand, E. Gerhards-Padilla, and M. Smith, “Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study,” *2016 IEEE Symposium on Security and Privacy (SP)*, 2016. doi: 10.1109/SP.2016.18

[38] J. Kˇroustek, P. Matula, and P. Zemek, “RetDec: An Open-Source Machine-Code Decompiler,” *Botconf 2017*, p. 152, Dec. 2017.

[39] “Avast open-sources its machine-code decompiler,” 2017. [Online]. Available: https://blog.avast.com/avast-open-sources-its-machine-code-decompiler

[40] “Avast/retdec,” Avast, Mar. 2021. [Online]. Available: https://github.com/avast/retdec

[41] “Ghidra,” 2019. [Online]. Available: https://ghidra-sre.org/

[42] Z. Liu and S. Wang, “How far we have come: Testing decompilation correctness of C decompilers,” in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. Virtual Event USA: ACM, Jul. 2020. doi: 10.1145/3395363.3397370. ISBN 978-1-4503-8008-9 pp. 475–487. [Online]. Available: https://dl.acm.org/doi/10.1145/3395363.3397370

[43] W. M. Khoo, “Decompilation as search,” Ph.D. dissertation, 2013.

[44] E. Schulte, J. Ruchti, M. Noonan, D. Ciarletta, and A. Loginov, “Evolving Exact Decompilation,” 2018. doi: 10.14722/BAR.2018.23008

[45] M. O. Myreen, M. J. C. Gordon, and K. Slind, “Decompilation into logic — Improved,” in *2012 Formal Methods in Computer-Aided Design (FMCAD)*, Oct. 2012, pp. 78–81.

[46] F. Verbeek, P. Olivier, and B. Ravindran, “Sound C Code Decompilation for a Subset of x86-64 Binaries,” in *Software Engineering and Formal Methods*, F. de Boer and A. Cerone, Eds. Cham: Springer International Publishing, 2020, vol. 12310, pp. 247–264. ISBN 978-3-030-58767-3 978-3-030-58768-0. [Online]. Available: http://link.springer.com/10.1007/978-3-030-58768-0_14

[47] N. Harrand, C. Soto-Valero, M. Monperrus, and B. Baudry, “Java Decompiler Diversity and its Application to Meta-decompilation,” *Journal of Systems and Software*, vol. 168, p. 110645, Oct. 2020. doi: 10.1016/j.jss.2020.110645. [Online]. Available: http://arxiv.org/abs/2005.11315

[48] D. S. Katz, J. Ruchti, and E. Schulte, “Using recurrent neural networks for decompilation,” in *2018 IEEE 25th International Conference on*

Software Analysis, Evolution and Reengineering (SANER), Mar. 2018. doi: 10.1109/SANER.2018.8330222 pp. 346–356.

[49] P. Koehn and R. Knowles, "Six Challenges for Neural Machine Translation," *arXiv:1706.03872 [cs]*, Jun. 2017. [Online]. Available: http://arxiv.org/abs/1706.03872

[50] C. Fu, H. Chen, H. Liu, X. Chen, Y. Tian, F. Koushanfar, and J. Zhao, "A Neural-based Program Decompiler," *arXiv:1906.12029 [cs]*, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.12029

[51] R. Liang, Y. Cao, P. Hu, and K. Chen, "Neutron: An attention-based neural decompiler," *Cybersecurity*, vol. 4, no. 1, p. 5, Mar. 2021. doi: 10.1186/s42400-021-00070-0. [Online]. Available: https://doi.org/10.1186/s42400-021-00070-0

[52] J. Caballero and Z. Lin, "Type Inference on Executables," *ACM Computing Surveys*, vol. 48, no. 4, pp. 65:1–65:35, May 2016. doi: 10.1145/2896499. [Online]. Available: https://doi.org/10.1145/2896499

[53] A. Jaffe, J. Lacomis, E. J. Schwartz, C. L. Goues, and B. Vasilescu, "Meaningful variable names for decompiled code: A machine translation approach," in *Proceedings of the 26th Conference on Program Comprehension*, ser. ICPC '18. New York, NY, USA: Association for Computing Machinery, May 2018. doi: 10.1145/3196321.3196330. ISBN 978-1-4503-5714-2 pp. 20–30. [Online]. Available: https://doi.org/10.1145/3196321.3196330

[54] "Hex Rays – State-of-the-art binary code analysis solutions." [Online]. Available: https://www.hex-rays.com/

[55] J. Escalada, T. Scully, and F. Ortin, "Improving type information inferred by decompilers with supervised machine learning," *arXiv:2101.08116 [cs]*, Feb. 2021. [Online]. Available: http://arxiv.org/abs/2101.08116

[56] J. Lacomis, P. Yin, E. J. Schwartz, M. Allamanis, C. L. Goues, G. Neubig, and B. Vasilescu, "DIRE: A Neural Approach to Decompiled Identifier Naming," *arXiv:1909.09029 [cs]*, Oct. 2019. [Online]. Available: http://arxiv.org/abs/1909.09029

[57] Y. David, U. Alon, and E. Yahav, "Neural Reverse Engineering of Stripped Binaries using Augmented Control Flow Graphs," *Proceedings*

of the ACM on Programming Languages, vol. 4, no. OOPSLA, pp. 1–28, Nov. 2020. doi: 10.1145/3428293. [Online]. Available: http://arxiv.org/abs/1902.09122

[58] T. Bao, J. Burket, M. Woo, R. Turner, and D. Brumley, "BYTEWEIGHT: Learning to Recognize Functions in Binary Code," in *USENIX Security Symposium*, 2014.

[59] E. C. R. Shin, D. Song, and R. Moazzezi, "Recognizing functions in binaries with neural networks," in *Proceedings of the 24th USENIX Conference on Security Symposium*, ser. SEC'15. USA: USENIX Association, Aug. 2015. ISBN 978-1-931971-23-2 pp. 611–626.

[60] Z. L. Chua, S. Shen, P. Saxena, and Z. Liang, "Neural Nets Can Learn Function Type Signatures From Binaries," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017. ISBN 978-1-931971-40-9 pp. 99–116. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/chua

[61] T. Ahmed, P. Devanbu, and A. A. Sawant, "Finding Inlined Functions in Optimized Binaries," *arXiv:2103.05221 [cs]*, Mar. 2021. [Online]. Available: http://arxiv.org/abs/2103.05221

[62] "Wasm-decompile." [Online]. Available: https://github.com/WebAssembly/wabt

[63] "Wasm2c." [Online]. Available: https://github.com/WebAssembly/wabt

[64] P. N. F. Software, "Reverse Engineering WebAssembly," Jul. 2018. [Online]. Available: https://medium.com/@pnfsoftware/reverse-engineering-webassembly-ed184a099931

[65] F. Ortin and J. Escalada, "Cnerator: A Python application for the controlled stochastic generation of standard C source code," *SoftwareX*, vol. 15, p. 100711, Jul. 2021. doi: 10.1016/j.softx.2021.100711. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235271102100056X

[66] X. Yang, Y. Chen, E. Eide, and J. Regehr, "Finding and understanding bugs in C compilers," *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 283–294, Jun. 2011. doi: 10.1145/1993316.1993532. [Online]. Available: https://doi.org/10.1145/1993316.1993532

[67] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-Source Toolkit for Neural Machine Translation," *arXiv:1701.02810 [cs]*, Mar. 2017. [Online]. Available: http://arxiv.org/abs/1701.02810

[68] Q. Chen, J. Lacomis, E. J. Schwartz, C. L. Goues, G. Neubig, and B. Vasilescu, "Augmenting Decompiler Output with Learned Variable Names and Types," *arXiv:2108.06363 [cs]*, Aug. 2021. [Online]. Available: http://arxiv.org/abs/2108.06363

[69] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan. 2009. doi: 10.1109/TNN.2008.2005605