A Bibliography of Publications about the Rust programming language

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA

Tel: +1 801 581 5254
FAX: +1 801 581 4148

E-mail: `beebe@math.utah.edu`, `beebe@acm.org`,
`beebe@computer.org` (Internet)
WWW URL: `https://www.math.utah.edu/~beebe/`

11 April 2024
Version 1.21

Title word cross-reference

2018 [Joh18, Kol19, Mat19a, Mat19b, SK19].

7 [BHM19].

Abstractions [HPC+13]. **Action** [McN21]. **Actor** [OOI+17]. **Advanced** [SKM19]. **Advantages** [BBS23]. **Aeneas** [HP22]. **Agent** [ACA+19, CDP16]. **Agent-Based** [ACA+19, CDP16]. **Algorithms** [Mat19a]. **Aliasing** [EBP+23, JDKD20]. **Alignment** [CXH23]. **Alzheimer** [CDP16]. **am** [Ter16]. **Analysis** [CSI19, CCXZ23, ABC+20, LAT+18]. **analyze** [ABC+20]. **Any** [KOSK19, PXA+21]. **APIs** [MKW18]. **Appendix** [Bee19]. **Applications** [Bha20, Egu18, GB18, Joh18, Lyu21, RW20]. **Applied** [Har22]. **Approach** [TOK20a, LFEL19]. **Apps** [Mat19b, Nus20]. **Architecture** [Har22, Ter16]. **Assessing** [CXH23]. **Assurance** [Har22]. **Atomics** [Bos23]. **automatic** [PXA+21]. **Avoid** [SST19]. **aware** [PXA+21].

B [BHM19]. **B-DTN7** [BHM19]. **Based** [ACA+19, CDP16, DP18, LKBP20, Lyu21, MTK20, OOI+17, YC22, BHM19, KTK19, LAT+18, LFEL19, MTK21]. **Beautiful** [Bla15]. **Beginning** [Mil18]. **Beispiel** [Ter16]. **between** [CXH23]. **Beyond** [BBB+17]. **blockchain** [NQ20]. **Borrowing** [Pea21]. **borrows** [JDKD20]. **Bounded** [TPT15]. **Bring** [WW20]. **Browser**

[Egu18, LH20]. **permissions** [YDJD21]. **Physical** [Lyu20a]. **Pi** [Nus20]. **Population** [CDP16]. **Practical** [Bha20, Lyu20b, Lyu21]. **Practice** [Bos23]. **Prediction** [CDP16]. **Preserving** [Riv21]. **Primer** [YC22]. **Processing** [SNPH18, SNGH20]. **Professional** [Mil18]. **Profile** [SNGH20]. **Profile-Guided** [SNGH20]. **programmers** [AMP^{+}20]. **Programming** [ACA^{+}19, BBB^{+}17, Bla15, BO17, BOT21, CSI19, Cha18, CXH23, EEB^{+}19, GKLS18, GB18, Har22, HPC^{+}13, Joh18, Jun20, KN17, Lun18, Mat19a, Mat19b, McN21, SKM19, SNPH18, TOK20b, JJKD18, JJKD21, KN19, Ua17]. **Programs** [BHR18, CCXZ23, LHC^{+}23, LAL18, LZH20, MTK20, MTK21]. **Project** [Dre2x, YC22]. **Project-Based** [YC22]. **Projects** [Lyu20b, Lyu21]. **Protocol** [BHM19]. **Protocols** [HKC^{+}20]. **Python** [LH20].

Q [OOI^{+}17]. **Q-Network** [OOI^{+}17].

R [Bee19]. **Raspberry** [Nus20]. **Reactive** [GB18, Joh18, Kol19]. **Readable** [Mat19a]. **Reconciling** [LHJ^{+}18]. **Reference** [Pea21, SKM19]. **Registry** [LCB23]. **relaxed** [DJKD20]. **Releases** [LCB23]. **Relevance** [MZH22]. **remote** [Ter16]. **repackaging** [KTK19]. **report** [Flu22]. **Resilient** [Cha18]. **Rethinking** [TGS18]. **Revisiting** [PP19]. **Risks** [ZWZ^{+}24]. **Robust** [GB18, PP19]. **RTL** [TOK20a, TOK20b]. **Rust** [HP22, SKM19, Ter16, ABC^{+}20, CDP16, ABG^{+}16, ACA^{+}19, ABC^{+}20, ABCV21, AMPS19, AMP^{+}20, BBS23, BBB^{+}17, Bal15, BHR18, Bee19, Bha20, Bla15, BO17, BOT21, Bos23, Cha18, CXH23, Cou15, CCXZ23, DRH15, Egu18, ESDH21, EBP^{+}23, Eva13, ECS20, FOPS20, Flu22, GB18, Har22, HPC^{+}13, Joh18, JJKD18, JDKD20, Jun20, JJKD21, KN17, KN19, Kol19, KTK18, KTK19, LNY20, LKBP20, LHC^{+}23, LCB23, LAL18, LFEL19, LZH20, Lun18, LH20, Lyu20a, Lyu20b, Lyu20d, Lyu20c, Lyu21, MK14, MTK20, MTK21, Mat19a, Mat19b, McN21, Mil18, MKW18, Mod19, NQ20, Nus20, Pea21, PCO19, PXA^{+}21, Riv21, SKM19, SK19, SST19, SNGH20, TOK20a, TOK20b, Ter16, TPT15, TPD20, Tro18, Ua17, WSZ^{+}18, WBM^{+}21, XCS^{+}22, YDJD21, YC22, ZWZ^{+}24, vO23]. **Rust-Based** [LKBP20]. **rust-code-analysis** [ABC^{+}20]. **Rust/Node.js/WebAssembly** [KTK19]. **RustBelt** [DJKD20, Dre2x, JJKD18]. **rusted** [BBS23]. **RustHorn** [MTK20, MTK21]. **RustyHermit** [LKBP20].

Safe [Bla15, BOT21, EBP^{+}23, JJKD21, Lun18, SNPH18, SNGH20, Tro18, Riv21]. **SafeDrop** [CCXZ23]. **Safely** [ECS20]. **Safer** [PXA^{+}21, ESDH21]. **Safety** [BBB^{+}17, LFEL19, Riv21, SK19, SST19, XCS^{+}22, vO23, Cha18, PXA^{+}21]. **Scalable** [Kol19, LKBP20]. **Secure** [HKC^{+}20, Mat19b, PP19]. **Securing** [LZH20, JJKD18]. **Security** [ZWZ^{+}24]. **Semantics** [WSZ^{+}18]. **semesters** [Flu22]. **Sensor** [OOI^{+}17]. **separating** [YDJD21]. **Servers** [Cha18]. **Servo** [ABG^{+}16]. **Session** [LNY20]. **SIMD** [KOSK19]. **Simulation** [CDP16, OOI^{+}17]. **Simulations** [ACA^{+}19]. **Skyrocket** [Egu18]. **SMACK** [BHR18]. **Software** [ECS20, Har22, KTK18, PP19, SKM19, Tro18, NQ20, Ter16]. **Softwarearchitektur** [Ter16]. **Solved** [MZH22, XCS^{+}22]. **Source** [KTK18, ABC^{+}20]. **specification** [AMPS19, WBM^{+}21]. **Specifications** [FOPS20]. **Speed** [WW20, PXA^{+}21]. **SPMD** [KOSK19]. **Stacked** [JDKD20]. **Static** [CCXZ23]. **Stream** [SNPH18, SNGH20]. **streaming** [Cou15]. **Structure** [KGDP17]. **Structures** [Mat19a]. **Stuck** [NQ20]. **Stuck-me-not**

References

Ardito:2020:RCA

[ABC+20] Luca Ardito, Luca Barbato, Marco Castelluccio, Riccardo Coppola, Calixte Denizet, Sylvestre Ledru, and Michele Valsesia. rust-code-analysis: a Rust library to analyze and extract maintainability information from source codes. *SoftwareX*, 12:100635, 2020. CODEN ???? ISSN 2352-7110. URL `https://www.sciencedirect.com/science/article/pii/S2352711020303484`.

Ardito:2021:ERC

[ABCV21] Luca Ardito, Luca Barbato, Riccardo Coppola, and Michele Valsesia. Evaluation of Rust code verbosity, understandability and complexity. *PeerJ Computer Science*, 7:e406:1–e406:33, February 2021. ISSN 2167-8359.

Anderson:2016:ESW

[ABG+16] B. Anderson, L. Bergstrom, M. Goregaokar, J. Matthews, K. McAllister, J. Moffitt, and S. Sapin. Engineering the Servo web browser engine using Rust. In *2016 IEEE/ACM 38th International Conference on Software*

Engineering Companion (ICSE-C), pages 81–89. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2016.

Antelmi:2019:ERP

[ACA+19] Alessia Antelmi, Gennaro Cordasco, Matteo D. Auria, Daniele De Vinco, Alberto Negro, and Carmine Spagnuolo. On evaluating Rust as a programming language for the future of massive agent-based simulations. In *Methods and Applications for Modeling and Simulation of Complex Systems: 19th Asia Simulation Conference, AsiaSim 2019, Singapore, October 30–November 1, 2019, Proceedings*, pages 15–28. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2019. URL `http://link.springer.com/chapter/10.1007/978-981-15-1078-6_2`.

Astrauskas:2020:HDP

[AMP+20] Vytautas Astrauskas, Christoph Matheja, Federico Poli, Peter Müller, and Alexander J. Summers. How do programmers use unsafe Rust? *Proceedings of the ACM on Programming Languages (PACMPL)*, 4(OOPSLA): 136:1–136:27, November 2020. URL `https://dl.acm.org/doi/10.1145/3428204`.

Astrauskas:2019:LRT

[AMPS19] Vytautas Astrauskas, Peter Müller, Federico Poli, and Alexander J. Summers. Leveraging Rust types for modular specification and verification. *Proceedings of the ACM on Programming Languages (PACMPL)*, 3 (OOPSLA):147:1–147:30, October 2019. URL `https://dl.acm.org/doi/abs/10.1145/3360573`.

Balbaert:2015:RE

[Bal15] Ivo Balbaert. *Rust Essentials*. Packt Publishing, Birmingham, UK, 2015. ISBN 1-78528-213-1. x + 161 pp. LCCN QA76.73.R87 B35 2015. URL `http://proquest.safaribooksonline.com/9781785285769`.

Balasubramanian:2017:SPR

[BBB+17] Abhiram Balasubramanian, Marek S. Baranowski, Anton Burtsev, Aurojit Panda, Zvonimir Rakamari, and Leonid Ryzhyk. System programming in Rust: Beyond safety. *Operating Systems Review*, 51(1): 94–99, August 2017. CODEN OSRED8. ISSN 0163-5980 (print), 1943-586X (electronic).

Bagnara:2023:CRA

[BBS23] Roberto Bagnara, Abramo Bagnara, and Federico Serafini. C-rusted: The advantages of Rust, in C, without the disadvantages. *arXiv.org*, ??(??):1–12, February 10, 2023. URL `https://arxiv.org/abs/2302.05331`.

Beebe:2019:ARR

[Bee19] Nelson H. F. Beebe. Appendix R: Rust interface. Technical report, University of Utah, Department of Mathematics, Salt Lake City, UT 84112-0090, USA, December 13, 2019. R-1–G-11 [994/995] pp.

Bhattacharjee:2020:PML

[Bha20] Joydeep Bhattacharjee. *Practical Machine Learning with Rust: Creating Intelligent Applications in Rust*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. ISBN 1-4842-5120-2, 1-4842-5121-0 (e-book). xv + 354 + 28 pp. LCCN Q325.5; QA75.5-76.95. URL http://link.springer.com/book/10.1007/978-1-4842-5121-8.

Baumgartner:2019:BDB

[BHM19] L. Baumgärtner, J. Höchst, and T. Meuser. B-DTN7: Browser-based disruption-tolerant networking via Bundle Protocol 7. In *2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–8. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Baranowski:2018:VRP

[BHR18] Marek Baranowski, Shaobo He, and Zvonimir Rakamari. Verifying Rust programs with SMACK. In *Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7–10, 2018, Proceedings*, pages 528–535. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2018. URL http://link.springer.com/chapter/10.1007/978-3-030-01090-4_32.

Blandy:2015:RPL

[Bla15] Jim Blandy. *The Rust Programming Language: Fast, Safe, and Beautiful*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2015. ISBN 1-4919-2544-2. ???? pp. LCCN ???? Video file (1h10m).

Blandy:2017:PR

[BO17] Jim Blandy and Jason Orendorff. *Programming Rust*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2017. ISBN 1-4919-2728-3 (paperback), 1-4919-2727-5, 1-4919-2723-2 (e-book), 1-4919-2725-9 (e-book). xx + 598 pp. LCCN QA76.73.R88 B53 2017. URL http://proquest.safaribooksonline.com/9781491927274.

Bos:2023:RAL

[Bos23] Mara Bos. *Rust Atomics and Locks: Low-Level Concurrency in Practice*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2023. ISBN 1-0981-1944-4. xvii + 230 pp. LCCN QA76.73.R87 B67 2023.

Blandy:2021:PRF

[BOT21] Jim Blandy, Jason Orendorff, and Leonora F. S. Tindall. *Programming Rust: Fast, Safe Systems*

Development. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, second edition, 2021. ISBN 1-4920-5259-0. xix + 713 pp. LCCN QA76.73.R87 B58 2021.

Chifflier:2017:WPL

[CC17] P. Chifflier and G. Couprie. Writing parsers like it is 2017. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 80–92. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2017.

Cui:2023:SDM

[CCXZ23] Mohan Cui, Chengjun Chen, Hui Xu, and Yangfan Zhou. SafeDrop: Detecting memory deallocation bugs of Rust programs via static data-flow analysis. *ACM Transactions on Software Engineering and Methodology*, 32(4): 82:1–82:??, July 2023. CODEN ATSMER. ISSN 1049-331X (print), 1557-7392 (electronic). URL `https://dl.acm.org/doi/10.1145/3542948`.

Cimler:2016:CRC

[CDP16] Richard Cimler, Ondřej Doležal, and Pavel Pscheidl. Comparison of RUST and C# as a tool for creation of a large agent-based simulation for population prediction of patients with Alzheimer's disease in EU. In *Computational Collective Intelligence: 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28–30, 2016. Proceedings, Part II*, pages 252–261. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2016. URL `http://link.springer.com/chapter/10.1007/978-3-319-45246-3_24`.

Chanda:2018:NPR

[Cha18] Abhishek Chanda. *Network Programming Rust: Build Fast and Resilient Network Servers and Clients by Leveraging Rust's Memory-safety and Concurrency Features*. Packt Publishing, Birmingham, UK, 2018. ISBN 1-78862-171-9 (e-book), 1-78862-489-0 (paperback). iii + 265 pp. LCCN QA76.73.R88. URL `http://proquest.safaribooksonline.com/?fpi=9781788624893`.

Couprie:2015:NBO

[Cou15] G. Couprie. Nom, a byte oriented, streaming, zero copy, parser combinators library in Rust. In *2015 IEEE Security and Privacy Workshops*, pages 142–148. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2015.

Chakraborty:2019:EAG

[CSI19] P. Chakraborty, R. Shahriyar, and A. Iqbal. Empirical analysis of the growth and challenges of new programming languages. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 191–196. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Cogo:2023:AAB

[CXH23] Filipe Roseiro Cogo, Xin Xia, and Ahmed E. Hassan. Assessing the alignment between the information needs of developers and the documentation of programming languages: a case study on Rust. *ACM Transactions on Software Engineering and Methodology*, 32(2):43:1–43:??, April 2023. CODEN ATSMER. ISSN 1049-331X (print), 1557-7392 (electronic). URL `https://dl.acm.org/doi/10.1145/3546945`.

Dang:2020:RMR

[DJKD20] Hoang-Hai Dang, Jacques-Henri Jourdan, Jan-Oliver Kaiser, and Derek Dreyer. RustBelt meets relaxed memory. *Proceedings of the ACM on Programming Languages (PACMPL)*, 4(POPL): 34:1–34:29, January 2020. URL `https://dl.acm.org/doi/abs/10.1145/3371102`.

Denisov:2018:MIM

[DP18] A. Denisov and S. Pankevich. Mull it over: Mutation testing based on LLVM. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 25–31. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Dreyer:202x:EPR

[Dre2x] Derek Dreyer. ERC project “RustBelt”. Web site, 202x. URL `https://plv.mpi-sws.org/rustbelt/`.

Dewey:2015:FRT

[DRH15] K. Dewey, J. Roesch, and B. Hardekopf. Fuzzing the Rust typechecker using CLP (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 482–493. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2015.

Emre:2023:ALT

[EBP$^+$23] Mehmet Emre, Peter Boyland, Aesha Parekh, Ryan Schroeder, Kyle Dewey, and Ben Hardekopf. Aliasing limits on translating C to safe Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 7(OOPSLA1):94:1–94:??, April 2023. CODEN ???? ISSN 2475-1421 (electronic). URL `https://dl.acm.org/doi/10.1145/3586046`.

Evans:2020:RUS

[ECS20] Ana Nora Evans, Bradford Campbell, and Mary Lou Soffa. Is Rust used safely by software developers? In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 246–257. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Emmerich:2019:CWN

[EEB$^+$19] P. Emmerich, S. Ellmann, F. Bonk, A. Egger, E. G. Sánchez-

Torija, T. Günzel, S. di Luzio, A. Obada, M. Stadlmeier, S. Voit, and G. Carle. The case for writing network drivers in high-level programming languages. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 1–13. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

EguiaMoraza:2018:RHP

[Egu18] Iban Eguia Moraza. *Rust High Performance: Learn to Skyrocket the Performance of Your Rust Applications*. Packt Publishing, Birmingham, UK, 2018. ISBN 1-78847-823-1. 265 pp. LCCN QA76.7.

Emre:2021:TCS

[ESDH21] Mehmet Emre, Ryan Schroeder, Kyle Dewey, and Ben Hardekopf. Translating C to safer Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 5 (OOPSLA):121:1–121:29, October 2021. CODEN ???? ISSN 2475-1421 (electronic). URL `https://dl.acm.org/doi/10.1145/3485498`.

Evans:2013:URU

[Eva13] David Evans. Using Rust for an undergraduate OS course. Web site., 2013. URL `http://rust-class.org/0/pages/using-rust-for-an-undergraduate-os-course.html`.

Fluet:2022:ERT

[Flu22] M. Fluet. Experience report: Two semesters teaching Rust. In ????, editor, *Proceedings of the Rust-Edu Workshop*, pages 48–58. ????, ????, August 2022. URL `https://rust-edu.org/workshop/proceedings.pdf`.

Finkbeiner:2020:VRM

[FOPS20] Bernd Finkbeiner, Stefan Oswald, Noemi Passing, and Maximilian Schwenger. Verified Rust monitors for Lola specifications. In *Runtime Verification: 20th International Conference, RV 2020, Los Angeles, CA, USA, October 6–9, 2020, Proceedings*, pages 431–450. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-60508-7_24`.

Gomez:2018:RPE

[GB18] Guillaume Gomez and Antoni Boucher. *Rust Programming by Example: Enter the World of Rust by Building Engaging, Concurrent, Reactive, and Robust Applications*. Packt Publishing, Birmingham, UK, 2018. ISBN 1-78839-063-6, 1-78847-030-3 (e-book). viii + 437 pp. LCCN QA76.73.R88. URL `http://proquestcombo.safaribooksonline.com/9781788390637`.

Georgiou:2018:WYP

[GKLS18] S. Georgiou, M. Kechagia, P. Louridas, and D. Spinellis. What are your programming language's energy-delay implications? In *2018 IEEE/ ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 303–313. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018. URL `https://ieeexplore.ieee.org/document/8595213`.

Hardin:2022:HSC

[Har22] David Hardin. Hardware/ software co-assurance for the Rust programming language applied to zero trust architecture development. *ACM SIGADA Ada Letters*, 42(2): 55–61, December 2022. CODEN AALEE5. ISSN 1094-3641 (print), 1557-9476 (electronic). URL `https://dl.acm.org/doi/10.1145/3591335.3591340`.

Han:2020:SMF

[HKC+20] J. Han, S. Kim, D. Cho, B. Choi, J. Ha, and D. Han. A secure middlebox framework for enabling visibility over multiple encryption protocols. *IEEE/ACM Transactions on Networking*, 28 (6):2727–2740, 2020. CODEN IEANEP. ISSN 1063-6692 (print), 1558-2566 (electronic).

Ho:2022:ARV

[HP22] Son Ho and Jonathan Protzenko. Aeneas: Rust verification by functional translation. *Proceedings of the ACM on Programming Languages (PACMPL)*, 6(ICFP): 116:1–116:??, August 2022. CODEN ???? ISSN 2475-1421 (electronic). URL `https://dl.acm.org/doi/10.1145/3547647`.

Holk:2013:GPR

[HPC+13] E. Holk, M. Pathirage, A. Chauhan, A. Lumsdaine, and N. D. Matsakis. GPU programming in Rust: Implementing high-level abstractions in a systems-level language. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 315–324. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2013.

Jung:2020:SBA

[JDKD20] Ralf Jung, Hoang-Hai Dang, Jeehoon Kang, and Derek Dreyer. Stacked borrows: an aliasing model for Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 4(POPL): 41:1–41:32, January 2020. URL `https://dl.acm.org/doi/abs/10.1145/3371109`.

Jung:2018:RSF

[JJKD18] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. RustBelt: securing the foundations of the Rust programming language. *Proceedings of the ACM on Programming Languages (PACMPL)*, 2(POPL): 66:1–66:??, January 2018. CODEN ???? ISSN 2475-1421.

Jung:2021:SSP

[JJKD21] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. Safe systems programming in Rust. *Communications of the ACM*, 64 (4):144–152, April 2021. CODEN CACMA2. ISSN 0001-0782 (print), 1557-7317 (electronic). URL https://dl.acm.org/doi/10.1145/3418295.

Johnson:2018:HFP

[Joh18] Andrew Johnson. *Hands-on Functional Programming in Rust: Build Modular and Reactive Applications with Functional Programming Techniques in Rust 2018*. Packt Publishing, Birmingham, UK, 2018. ISBN 1-78883-935-8 (paperback). v + 278 pp. LCCN QA76.62.

Jung:2020:UER

[Jun20] Ralf Jung. *Understanding and Evolving the Rust Programming Language*. Doktors der Ingenieurwissenschaften, Fakultät für Mathematik und Informatik der Universität des Saarlandes, Saarbrücken, Germany, August 2020. x + 387 pp. URL https://research.ralfj.de/phd/thesis-print.pdf.

Kikas:2017:SEP

[KGDP17] R. Kikas, G. Gousios, M. Dumas, and D. Pfahl. Structure and evolution of package dependency networks. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 102–112. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2017.

Klabnik:2017:RPL

[KN17] Steve Klabnik and Carol Nichols. *The Rust Programming Language*. No Starch Press, San Francisco, CA, USA, 2017. ISBN 1-59327-828-4 (paperback), 1-59327-851-9 (e-pub). xxvii + 519 pp. LCCN QA76.73.R87 K53 2018.

Klabnik:2019:RPL

[KN19] Steve Klabnik and Carol Nichols. *The Rust programming language*. No Starch Press, San Francisco, CA, USA, second edition, 2019. ISBN 1-0981-2253-4, 1-71850-044-0 (paperback). xxix + 526 pp. LCCN QA76.73.R87. URL http://proquest.safaribooksonline.com/?fpi=9781098122539; https://nostarch.com/download/samples/RustProgrammingLanguage2018_Sample_ToC.pdf; https://nostarch.com/Rust2018.

Kolodin:2019:HMR

[Kol19] Denis Kolodin. *Hands-on Microservices with Rust: Build, Test, and Deploy Scalable and Reactive Microservices with Rust 2018*. Packt Publishing, Birmingham, UK, 2019. ISBN 1-78934-198-1, 1-78934-275-9. 511 (est.) pp. LCCN QA76.73.R87. URL http://proquest.safaribooksonline.com/?fpi=9781789342758.

Kruppe:2019:ELL

[KOSK19] R. Kruppe, J. Oppermann, L. Sommer, and A. Koch. Extending LLVM for lightweight SPMD vectorization: Using SIMD and vector instructions easily from any language. In *2019 IEEE/ ACM International Symposium on Code Generation and Optimization (CGO)*, pages 278–279. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Kyriakou:2018:ICC

[KTK18] Kyriakos-Ioannis D. Kyriakou, Nikolaos D. Tselikas, and Georgia M. Kapitsaki. Improving C/C++ open source software discoverability by utilizing Rust and `Node.js` ecosystems. In *Open Source Systems: Enterprise Software and Solutions: 14th IFIP WG 2.13 International Conference, OSS 2018, Athens, Greece, June 8–10, 2018, Proceedings*, pages 181–192. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2018. URL `http://link.springer.com/chapter/10.1007/978-3-319-92375-8_15`.

Kyriakou:2019:ECC

[KTK19] Kyriakos-Ioannis D. Kyriakou, Nikolaos D. Tselikas, and Georgia M. Kapitsaki. Enhancing C/C++ based OSS development and discoverability with CBRJS: a `Rust/Node.js/WebAssembly` framework for repackaging legacy codebases. *The Journal of Systems and Software*, 157(??): Article 110395, November 2019. CODEN JSSODM. ISSN 0164-1212 (print), 1873-1228 (electronic). URL `http://www.sciencedirect.com/science/article/pii/S0164121219301700`.

Lindner:2018:NPV

[LAL18] M. Lindner, J. Aparicius, and P. Lindgren. No panic! Verification of Rust programs by symbolic execution. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 108–114. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Lindner:2018:HLB

[LAT+18] M. Lindner, J. Aparicio, H. Tjader, P. Lindgren, and J. Eriksson. Hardware-in-the-loop based WCET analysis with KLEE. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 345–352. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Li:2023:ESY

[LCB23] Hao Li, Filipe R. Cogo, and Cor-Paul Bezemer. An empirical study of yanked releases in the Rust package registry. *IEEE Transactions on Software Engineering*, 49(1):437–449, January 2023. CODEN IESEDJ. ISSN 0098-5589 (print), 1939-3520 (electronic).

Lindner:2019:VSF

[LFEL19] M. Lindner, N. Fitinghoff, J. Eriksson, and P. Lindgren. Verification of safety functions implemented in Rust — a symbolic execution based approach. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 432–439. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Lunnikivi:2020:TPR

[LH20] Henri Lunnikivi and Kai JylkkäTimo Hämäläinen. Transpiling Python to Rust for optimized performance. In *Embedded Computer Systems: Architectures, Modeling, and Simulation: 20th International Conference, SAMOS 2020, Samos, Greece, July 5–9, 2020, Proceedings*, pages 127–138. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-60939-9_9`.

Lattuada:2023:VVR

[LHC$^+$23] Andrea Lattuada, Travis Hance, Chanhee Cho, Matthias Brun, Isitha Subasinghe, Yi Zhou, Jon Howell, Bryan Parno, and Chris Hawblitzel. Verus: Verifying Rust programs using linear ghost types. *Proceedings of the ACM on Programming Languages (PACMPL)*, 7(OOPSLA1):85:1–85:??, April 2023. CODEN ???? ISSN 2475-1421 (electronic). URL `https://dl.acm.org/doi/10.1145/3586037`.

Lee:2018:RHL

[LHJ$^+$18] Juneyoung Lee, Chung-Kil Hur, Ralf Jung, Zhengyang Liu, John Regehr, and Nuno P. Lopes. Reconciling high-level optimizations and low-level code in LLVM. *Proceedings of the ACM on Programming Languages (PACMPL)*, 2 (OOPSLA):125:1–125:28, October 2018. URL `https://dl.acm.org/doi/abs/10.1145/3276495`.

Lankes:2020:RSR

[LKBP20] Stefan Lankes, Jonathan Klimt, Jens Breitbart, and Simon Pickartz. RustyHermit: A scalable, Rust-based virtual execution environment. In *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers*, pages 331–342. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-59851-8_22`.

Lagaillardie:2020:IMS

[LNY20] Nicolas Lagaillardie, Rumyana Neykova, and Nobuko Yoshida. Implementing multiparty session types in Rust. In *Coordination Models and Languages: 2nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15–19, 2020, Pro-*

ceedings, pages 127–136. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-50029-0_8`.

Lundberg:2018:SKP

[Lun18] Johannes Lundberg. Safe kernel programming with Rust. Master's thesis, KTH, Software and Computer systems, SCS, Stockholm, Sweden, July 2018. 56 pp. URL `http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1238890&dswid=-1806`.

Lyu:2020:PCR

[Lyu20a] Shing Lyu. Physical computing in Rust. In *Practical Rust Projects: Building Game, Physical Computing, and Machine Learning Applications*, pages 155–185. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-1-4842-5599-5_5`.

Lyu:2020:PRP

[Lyu20b] Shing Lyu. *Practical Rust Projects*. Apress, Berkeley, CA, USA, 2020. ISBN 1-4842-5598-4, 1-4842-5599-2 (e-book). xiii + 257 + 56 + 42 pp. URL `http://link.springer.com/book/10.1007/978-1-4842-5599-5`.

Lyu:2020:WWR

[Lyu20c] Shing Lyu. Welcome to the world of Rust. In *Practical Rust Projects: Building Game, Physical Computing, and Machine Learning Applications*, pages 1–8. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-1-4842-5599-5_1`.

Lyu:2020:WEC

[Lyu20d] Shing Lyu. What else can you do with Rust? In *Practical Rust Projects: Building Game, Physical Computing, and Machine Learning Applications*, pages 237–250. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-1-4842-5599-5_7`.

Lyu:2021:PRW

[Lyu21] Shing Lyu. *Practical Rust Web Projects: Building Cloud and Web-Based Applications*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2021. ISBN 1-4842-6588-2, 1-4842-6589-0, 1-4842-6590-4. xi + 256 + 30 pp. LCCN QA76.73.R87. URL `http://link.springer.com/book/10.1007/978-1-4842-6589-5`.

Liu:2020:SUR

[LZH20] P. Liu, G. Zhao, and J. Huang. Securing unsafe Rust programs with XRust. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 234–245. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Matzinger:2019:HDS

[Mat19a] Claus Matzinger. *Hands-on Data Structures and Algorithms with Rust: Learn Programming Techniques to Build Effective, Maintainable, and Readable Code in Rust 2018*. Packt Publishing, Birmingham, UK, 2019. ISBN 1-78899-149-4. vii + 298 pp. LCCN Q76.73.R87.

Matzinger:2019:RPC

[Mat19b] Claus Matzinger. *Rust Programming Cookbook: Explore the Latest Features of Rust 2018 for Building Fast and Secure Apps*. Packt Publishing, Birmingham, UK, 2019. ISBN 1-78953-066-0, 1-78953-174-8 (PDF e-book). 434 pp. LCCN QA76.73.R87.

McNamara:2021:RAS

[McN21] Timothy Samuel McNamara. *Rust in Action: Systems Programming Concepts and Techniques*. Manning Publications, Greenwich, CT, USA, 2021. ISBN 1-61729-455-1 (paperback). xxiii + 430 pp. LCCN QA76.73.R87 M36 2021.

Milanesi:2018:BRN

[Mil18] Carlo Milanesi. *Beginning Rust: From Novice to Professional*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2018. ISBN 1-4842-3467-7, 1-4842-3468-5 (e-book). xvii + 376 + 3 pp. URL `http://link.springer.com/book/10.1007/978-1-4842-3468-6`.

Matsakis:2014:RL

[MK14] Nicholas D. Matsakis and Felix S. Klock II. The Rust language. *ACM SIGADA Ada Letters*, 34 (3):103–104, December 2014. CODEN AALEE5. ISSN 1094-3641 (print), 1557-9476 (electronic).

Mindermann:2018:HUR

[MKW18] K. Mindermann, P. Keck, and S. Wagner. How usable are Rust cryptography APIs? In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 143–154. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Modrzyk:2019:WR

[Mod19] Nicolas Modrzyk. Week 4: Rust. In *Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API*, pages 57–84. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2019. URL `http://link.springer.com/chapter/10.1007/978-1-4842-4197-4_4`.

Matsushita:2020:RCB

[MTK20] Yusuke Matsushita, Takeshi Tsukada, and Naoki Kobayashi. RustHorn: CHC-based verification for Rust programs. In *Programming Languages and Systems: 29th European Symposium on Programming, ESOP 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*

2020, Dublin, Ireland, April 25–30, 2020, Proceedings, pages 484–514. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-44914-8_18`.

Matsushita:2021:RCB

[MTK21] Yusuke Matsushita, Takeshi Tsukada, and Naoki Kobayashi. RustHorn: CHC-based verification for Rust programs. *ACM Transactions on Programming Languages and Systems*, 43(4): 15:1–15:54, December 2021. CODEN ATPSDT. ISSN 0164-0925 (print), 1558-4593 (electronic). URL `https://dl.acm.org/doi/10.1145/3462205`.

Miller:2022:RCF

[MZH22] Barton P. Miller, Mengxiao Zhang, and Elisa R. Heymann. The relevance of classic fuzz testing: Have we solved this one? *IEEE Transactions on Software Engineering*, 48(6):2028–2039, June 2022. CODEN IESEDJ. ISSN 0098-5589 (print), 1939-3520 (electronic). URL `https://arxiv.org/abs/2008.06537; https://ieeexplore.ieee.org/document/9309406`.

Ning:2020:SMD

[NQ20] Pengxiang Ning and Boqin Qin. Stuck-me-not: A deadlock detector on blockchain software in Rust. *Procedia Computer Science*, 177:599–604, 2020. ISSN 1877-0509. URL `https://www.sciencedirect.com/science/article/pii/S1877050920323565`. The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020) / The 10th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2020) / Affiliated Workshops.

Nusairat:2020:RI

[Nus20] Joseph Faisal Nusairat. *Rust for the IoT: Building Internet of Things Apps with Rust and Raspberry Pi*. Apress, Berkeley, CA, USA, 2020. ISBN 1-4842-5859-2, 1-4842-5860-6 (e-book). LCCN QA76.73.R87 N87 2020. URL `http://link.springer.com/book/10.1007/978-1-4842-5860-6`.

Oda:2017:DIS

[OOI+17] T. Oda, R. Obukata, M. Ikeda, L. Barolli, and M. Takizawa. Design and implementation of a simulation system based on deep Q-network for mobile actor node control in wireless sensor and actor networks. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 195–200. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2017.

Pinho:2019:TRC

[PCO19] A. Pinho, L. Couto, and J. Oliveira. Towards Rust for critical systems. In *2019 IEEE International Symposium on Software*

Reliability Engineering Workshops (ISSREW), pages 19–24. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Pearce:2021:LFR

[Pea21] David J. Pearce. A lightweight formalism for reference lifetimes and borrowing in Rust. *ACM Transactions on Programming Languages and Systems*, 43(1): 3:1–3:73, April 2021. CODEN ATPSDT. ISSN 0164-0925 (print), 1558-4593 (electronic). URL `https://dl.acm.org/doi/10.1145/3443420`.

Pipek:2019:RCI

[PP19] T. Pipek and M. Pirker. Revisiting the challenges of input parsing for robust and secure software. In *2019 International Conference on Software Security and Assurance (ICSSA)*, pages 41–45. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2019.

Popescu:2021:SSA

[PXA$^+$21] Natalie Popescu, Ziyang Xu, Sotiris Apostolakis, David I. August, and Amit Levy. Safer at any speed: automatic context-aware safety enhancement for Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 5 (OOPSLA):103:1–103:23, October 2021. CODEN ???? ISSN 2475-1421 (electronic). URL `https://dl.acm.org/doi/10.1145/3485480`.

Rivera:2021:PMS

[Riv21] Elijah E. Rivera. Preserving memory safety in Safe Rust during interactions with unsafe languages. Master of Engineering in Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2021. 66 pp. URL `https://dspace.mit.edu/bitstream/handle/1721.1/139052/Rivera-eerivera-meng-eecs-2021-thesis.pdf`.

Romano:2020:WVT

[RW20] A. Romano and W. Wang. WasmView: Visual testing for WebAssembly applications. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 13–16. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020. URL `https://ieeexplore.ieee.org/document/9270402`.

Sharma:2019:MRL

[SK19] Rahul Sharma and Vesa Kaihlavirta. *Mastering Rust: Learn About Memory Safety, Type System, Concurrency, and the New Features of Rust 2018 Edition.* Packt Publishing, Birmingham, UK, second edition, 2019. ISBN 1-78934-118-3, 1-78934-657-6 (paperback). 543 (est.) pp. LCCN QA76.73.R87. URL `http://proquest.safaribooksonline.com/?fpi=9781789346572`.

Sharma:2019:CRP

[SKM19] Rahul Sharma, Vesa Kaihlavirta, and Claus Matzinger. *The Complete Rust Programming Reference Guide: Design, Develop, and Deploy Effective Software Systems Using the Advanced Constructs of Rust*. Packt Publishing, Birmingham, UK, 2019. ISBN 1-83882-810-9. xii + 679 pp. LCCN ????

Sydow:2020:TPG

[SNGH20] S. Sydow, M. Nabelsee, S. Glesner, and P. Herber. Towards profile-guided optimization for safe and efficient parallel stream processing in Rust. In *2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 289–296. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Sydow:2018:SUF

[SNPH18] S. Sydow, M. Nabelsee, H. Parzyjegla, and P. Herber. A safe and user-friendly graphical programming model for parallel stream processing. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 239–243. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Shetty:2019:CCC

[SST19] Nishanth Shetty, Nikhil Saldanha, and M. N. Thippeswamy. CRUST: A C/C++ to Rust transpiler using a nano-parser methodology to avoid C/C++ safety issues in legacy code. In *Emerging Research in Computing, Information, Communication and Applications: ERCICA 2018, Volume 1*, pages 241–250. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2019. URL `http://link.springer.com/chapter/10.1007/978-981-13-5953-8_21`.

Terber:2016:DSC

[Ter16] Matthias Terber. Domänenorientierte Softwarearchitektur mit Céu und Rust am Beispiel eines Heizungsgateways zur Fernüberwachung und Fernparametrisierung. (German) [Domain-oriented software architecture with Céu and Rust using the example of a heating gateway for remote monitoring and remote parameterization]. In *Internet der Dinge*, pages 117–126. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2016. URL `http://link.springer.com/chapter/10.1007/978-3-662-53443-4_13`.

Tsai:2018:RMH

[TGS18] P. Tsai, Y. L. Gan, and D. Sanchez. Rethinking the memory hierarchy for modern languages. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 203–216. IEEE Computer Society Press, 1109

Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Takano:2020:ACC

[TOK20a] K. Takano, T. Oda, and M. Kohata. Approach of a coding conventions for warning and suggestion in transpiler for Rust convert to RTL. In *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, pages 789–790. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Takano:2020:DDC

[TOK20b] Keisuke Takano, Tetsuya Oda, and Masaki Kohata. Design of a DSL for converting Rust programming language into RTL. In *Advances in Internet, Data and Web Technologies: The 8th International Conference on Emerging Internet, Data and Web Technologies (EIDWT-2020)*, pages 342–350. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2020. URL `http://link.springer.com/chapter/10.1007/978-3-030-39746-3_36`.

Tomb:2020:STC

[TPD20] A. Tomb, S. Pernsteiner, and M. Dodds. Symbolic testing for C and Rust. In *2020 IEEE Secure Development (SecDev)*, page 33. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Toman:2015:CBV

[TPT15] J. Toman, S. Pernsteiner, and E. Torlak. Crust: a bounded verifier for Rust (N). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 75–80. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2015.

Troutwine:2018:HCR

[Tro18] Brian L. Troutwine. *Hands-on Concurrency with Rust: Confidently Build Memory-safe, Parallel, and Efficient Software in Rust.* Packt Publishing, Birmingham, UK, 2018. ISBN 1-78839-997-8 (paperback), 1-78847-835-5. v + 449 pp. LCCN QA76.76.A65. URL `http://proquest.safaribooksonline.com/?fpi=9781788399975`.

Uzlu:2017:URP

[Ua17] T. Uzlu and E. aykol. On utilizing Rust programming language for Internet of Things. In *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 93–96. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2017.

vanOorschot:2023:MEMb

[vO23] Paul C. van Oorschot. Memory errors and memory safety: a look at Java and Rust. *IEEE Security & Privacy*, 21(3):62–68, May/June 2023. ISSN 1540-7993 (print), 1558-4046 (electronic).

Wolff:2021:MSV

[WBM+21] Fabian Wolff, Aurel Bílý, Christoph Matheja, Peter Müller, and Alexander J. Summers. Modular specification and verification of closures in Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 5(OOPSLA): 145:1–145:29, October 2021. CODEN ???? ISSN 2475-1421 (electronic). URL https://dl.acm.org/doi/10.1145/3485522.

Wang:2018:KFE

[WSZ+18] F. Wang, F. Song, M. Zhang, X. Zhu, and J. Zhang. KRust: a formal executable semantics of Rust. In *2018 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 44–51. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2018.

Wen:2020:WBE

[WW20] E. Wen and G. Weber. Wasmachine: Bring the edge up to speed with a WebAssembly OS. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 353–360. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2020.

Xu:2022:MSC

[XCS+22] Hui Xu, Zhuangbin Chen, Mingshen Sun, Yangfan Zhou, and Michael R. Lyu. Memory-safety challenge considered solved? An in-depth study with all Rust CVEs. *ACM Transactions on Software Engineering and Methodology*, 31(1):3:1–3:25, January 2022. CODEN ATSMER. ISSN 1049-331X (print), 1557-7392 (electronic). URL https://dl.acm.org/doi/10.1145/3466642.

Youens-Clark:2022:CLR

[YC22] Ken Youens-Clark. *Command-line Rust: a Project-Based Primer for Writing Rust CLIs*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2022. ISBN 1-0981-0943-0 (paperback). xviii + 377 pp. LCCN QA76.73.R87 Y68 2022.

Yanovski:2021:PGS

[YDJD21] Joshua Yanovski, Hoang-Hai Dang, Ralf Jung, and Derek Dreyer. `GhostCell`: separating permissions from data in Rust. *Proceedings of the ACM on Programming Languages (PACMPL)*, 5(ICFP):92:1–92:30, August 2021. CODEN ???? ISSN 2475-1421 (electronic). URL https://dl.acm.org/doi/10.1145/3473597.

Zheng:2024:CLS

[ZWZ+24] Xiaoye Zheng, Zhiyuan Wan, Yun Zhang, Rui Chang, and David Lo. A closer look at the security risks in the Rust ecosystem. *ACM Transactions on Software Engineering and Methodology*, 33(2): 34:1–34:??, February 2024. CO-

DEN ATSMER. ISSN 1049-331X (print), 1557-7392 (electronic). URL `https://dl.acm.org/doi/10.1145/3624738`.