A Stateful Fuzzer for the TCP/IP Stack of the Real-Time Operating System Zephyr

—

Report

Valentin Huber

Institute of Applied Information Technology

Zürich University of Applied Sciences ZHAW

contact@valentinhuber.me

November 30, 2024

Contents

1 Introduction

Daniele *et al.* introduce a taxonomy of stateful fuzzing in which they define a stateful system as "a system that takes a sequence of messages as input, producing outputs along the way,and where each input may result in an internal state change" [1].

2 Related Works

2.1 Read

- *Stateful Greybox Fuzzing* [2]: "In this paper, we argue that protocols are often explicitly encoded using state variables that are assigned and compared to named constants [...] More specifically, using pattern matching, we identify state variables using enumerated types (enums). An enumerated type is a group of named constants that specifies all possible values for a variable of that type. Our instrumentation injects a call to our runtime at every program location where a state variable is assigned to a new value. Our runtime efficiently constructs the state transition tree (STT). The STT captures the sequence of values assigned to state variables across all fuzzer-generated input sequences, and as a global data structure, it is shared with the fuzzer." [2] Built on LibFuzzer
- *StateAFL: Greybox fuzzing for stateful network servers* [3]: compile-time probes observing memory allocation and I/O operations; state inference based on fuzzy hashing of long-lived memory areas.
- *Ijon: Exploring Deep State Spaces via Fuzzing* [4]: Manual annotations of code, to manually add entries to an AFL-style map (set/inc at calculated offset), include state information (variable values) in how edge coverage is calculated, and store the max value a certain variable reaches during execution for the fuzzer to then maximize.
- *SandPuppy: Deep-State Fuzzing Guided by Automatic Detection of State-Representative Variables* [5]: Ijon [4], but automatic (initial run capturing variable-value traces, analyze along with source code, add Ijon-style instrumentation, repeat during fuzzing)
- *The Use of Likely Invariants as Feedback for Fuzzers* [6]: run for 24 hours, record variable values and relationships between them, then add a feedback that rewards when the generated assertions are violated
- *Ankou: guiding grey-box fuzzing towards combinatorial difference* [7]: take combination of executed branches into consideration, reduce to manageable adaptive fitness function
- *FuzzFactory: domain-specific fuzzing with waypoints* [8]: framework to add custom feedbacks like number of basic blocks executed, amount of memory allocated, etc.
- *ParmeSan: Sanitizer-guided Greybox Fuzzing* [9]: Use sanitizers checks as fuzzing targets
- *Fuzzing with Data Dependency Information* [10]: Use execution of new data dependencies as feedback
- *StateFuzz: System Call-Based State-Aware Linux Driver Fuzzing* [11]: Find state variables (long-lived, can be updated by users, change control flow or memory access) using static analysis, use that to guide fuzzing (new coverage, new value-range, new extreme value). (Talk: Good Example of why coverage-guided alone is insufficient). Check value ranges instead of all values (static symbex!). 4-digit number of state varia les in linux kernel and Qualcomm MSM kernel (Google Pixel).
- *ProFuzzBench: a benchmark for stateful protocol fuzzing* [12]: Suite of 10 protocols and 11 open-source implementations of those to be tested. TCP is notably absent from this list. Certain protocols (like FTP) already return HTTP status codes, others are patched to do so. Dockerized. The authors note that configuration is not taken into account and multi-party (≥ 3) protocols can not be fuzzed right now. Non-determinism in the programs make feedback (like code coverage) less predictable and thus fuzzing less performant because it introduces non-differentiable duplicate entries into the corpus. Speed is another issue, where complex setup-processes, costly network operations (resp. synchronization for me), and long multipart-inputs contribute. Finally, state identification is only superficially handled.

2.2 ToDo

- *Fuzzers for Stateful Systems: Survey and Research Directions* [1]: Review paper!
- *AFLNET: A Greybox Fuzzer for Network Protocols* [13]: FTP and RTSP

In the interest of open science, the source code of this project is released under an open-source license. During development, thousands of lines of code have been introduced to upstream projects across a range of pull request. The source code of the project is publicly available at

github.com/riesentoaster/fuzzing-zephyr-network-stack.

Bibliography

[1] C. Daniele, S. B. Andarzian, and E. Poll, "Fuzzers for stateful systems: Survey and research directions," *ACM Comput. Surv.*, vol. 56, no. 9, Apr. 2024, ISSN: 0360-0300. DOI: 10.1145/3648468. [Online]. Available: https://doi.org/10.1145/3648468.

[2] J. Ba, M. Böhme, Z. Mirzamomen, and A. Roychoudhury, "Stateful greybox fuzzing," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 3255–3272, ISBN: 978-1-939133-31-1. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/ba.

[3] R. Natella, "Stateafl: Greybox fuzzing for stateful network servers," *Empirical Software Engineering*, vol. 27, no. 7, p. 191, Oct. 2022, ISSN: 1573-7616. DOI: 10.1007/s10664-022-10233-3. [Online]. Available: https://doi.org/10.1007/s10664-022-10233-3.

[4] C. Aschermann, S. Schumilo, A. Abbasi, and T. Holz, "Ijon: Exploring deep state spaces via fuzzing," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1597–1612. DOI: 10.1109/SP40000.2020.00117.

[5] V. Paliath, E. Trickel, T. Bao, R. Wang, A. Doupé, and Y. Shoshitaishvili, "Sandpuppy: Deep-state fuzzing guided by automatic detection of state-representative variables," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, F. Maggi, M. Egele, M. Payer, and M. Carminati, Eds., Cham: Springer Nature Switzerland, 2024, pp. 227–250, ISBN: 978-3-031-64171-8.

[6] A. Fioraldi, D. C. D'Elia, and D. Balzarotti, "The use of likely invariants as feedback for fuzzers," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, Aug. 2021, pp. 2829–2846, ISBN: 978-1-939133-24-3. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/fioraldi.

[7] V. J. M. Manès, S. Kim, and S. K. Cha, "Ankou: Guiding grey-box fuzzing towards combinatorial difference," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20, Seoul, South Korea: Association for Computing Machinery, 2020, pp. 1024–1036, ISBN: 9781450371216. DOI: 10.1145/3377811.3380421. [Online]. Available: https://doi.org/10.1145/3377811.3380421.

[8] R. Padhye, C. Lemieux, K. Sen, L. Simon, and H. Vijayakumar, "Fuzzfactory: Domain-specific fuzzing with waypoints," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, Oct. 2019. DOI: 10.1145/3360600. [Online]. Available: https://doi.org/10.1145/3360600.

[9] S. Österlund, K. Razavi, H. Bos, and C. Giuffrida, "ParmeSan: Sanitizer-guided greybox fuzzing," in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020, pp. 2289–2306, ISBN: 978-1-939133-17-5. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/osterlund.

[10] A. Mantovani, A. Fioraldi, and D. Balzarotti, "Fuzzing with data dependency information," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, 2022, pp. 286–302. DOI: 10.1109/EuroSP53844.2022.00026.

[11] B. Zhao, Z. Li, S. Qin, *et al.*, "StateFuzz: System Call-Based State-Aware linux driver fuzzing," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 3273–3289, ISBN: 978-1-939133-31-1. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/zhao-bodong.

[12] R. Natella and V.-T. Pham, "Profuzzbench: A benchmark for stateful protocol fuzzing," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021, Virtual, Denmark: Association for Computing Machinery, 2021, pp. 662–665, ISBN: 9781450384599. DOI: 10.1145/3460319.3469077. [Online]. Available: https://doi.org/10.1145/3460319.3469077.

[13] V.-T. Pham, M. Böhme, and A. Roychoudhury, "Aflnet: A greybox fuzzer for network protocols," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 460–465. DOI: 10.1109/ICST46399.2020.00062.