УДК 004.052.42

Исследование применения формальных моделей для тестирования технологии eBPF в ядре Linux

Д. А. Максимов[1,2], Д. В. Буздало́в[2], А. В. Хорошилов[1,2]

[1]Московский физико-технический институт (национальный исследовательский университет)
[2]Институт системного программирования им. В. П. Иванникова Российской академии наук

Подсистема eBPF (extended Berkeley Package Filter) операционной системы Linux [1] – уникальная технология, позволяющая вставлять и запускать пользовательский код в модулях ядра (фильтры пакетов, системных вызовов и пр.). Ключевой особенностью от других технологий является наличие верификатора eBPF программ, который производит статический анализ поданной на вход программы и решает, должна ли она быть допущена к запуску.

Будучи разработанной из практических соображений, эта подсистема не имеет официальной формализации. Помимо этого, текущий верификатор часто выдаёт ложно положительные результаты, что заставляет разработчиков вставлять лишние проверки в свои программы, тем самым понижая их эффективность. Также не могут исключаться уязвимости, возникновение которых наблюдается достаточно часто ([10], [11], [12]).

На текущий момент, имеются некоторые теоретические ([2], [3], [4]) и практические ([5], [6]) результаты в анализе и верификации eBPF. В тестировании используют как Example-Based-Testing, так и Random-Testing (также известный как Fuzzing) подходы. Первый способ позволяет проверять поведение системы в конкретных ситуациях, в то время как второй способ позволяет автоматически генерировать случайные тесты и, соответственно, покрывать широкий класс входных входных данных. Property-Based-Testing (PBT) представляет собой комбинацию этих подходов. Его идея состоит в том, что у тестируемой системы присутствует формальное описание (спецификация), позволяющая описать эту систему на специализированном языке, который будет способен генерировать случайные тесты, удовлетворяющие конкретным свойствам этой модели.

В настоящей работе произведено исследование применимости PBT подхода для тестирования eBPF. На основе формальной модели eBPFPL [2] была построена модель его подмножества, описанная на языке Idris2 [5]. С использованием библиотеки DepTyCheck [6] построены генераторы eBPF программ, удовлетворяющих нижеописанным свойствам. Проведено тестирование верификатора eBPF на соблюдение этих свойств.

Построенная модель eBPF соответствует подмножеству eBPFPL, в котором разрешены только бинарные операции add, sub, mul над скалярными значениями, допустима любая mov-инструкция, частично поддержаны операции jmp в виде примитива ветвления (if-then-else выражения) и намеренного прыжка за пределы программы. Проверялись следующие свойства модели: «Любая корректная программа должна быть принята верификатором» и «Любая программа, у которой все ALU-операции корректны и существует хотя бы 1 исполнение с некорректным завершением, должна быть отвергнута верификатором».

В результате тестирования на стенде с процессором AMD Ryzen 5 1600 и ядром Linux версии 6.5.0-28-generic ошибок в работе верификатора не найдено. За 6 часов было сгенерировано 510 тестов, из них 102 относятся к первому свойству, остальные 408 – ко второму соответственно. Проведён численный анализ полученных результатов (рис. 1, рис. 2).

Рис. 1, Анализ 1 свойства

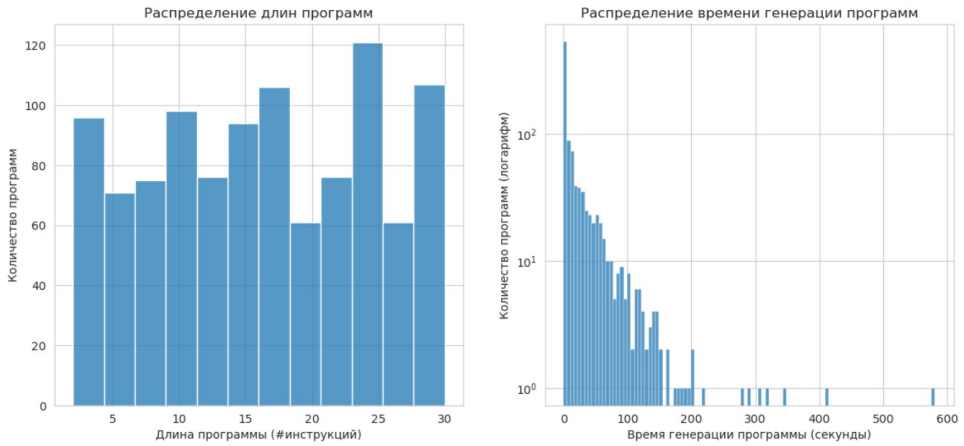
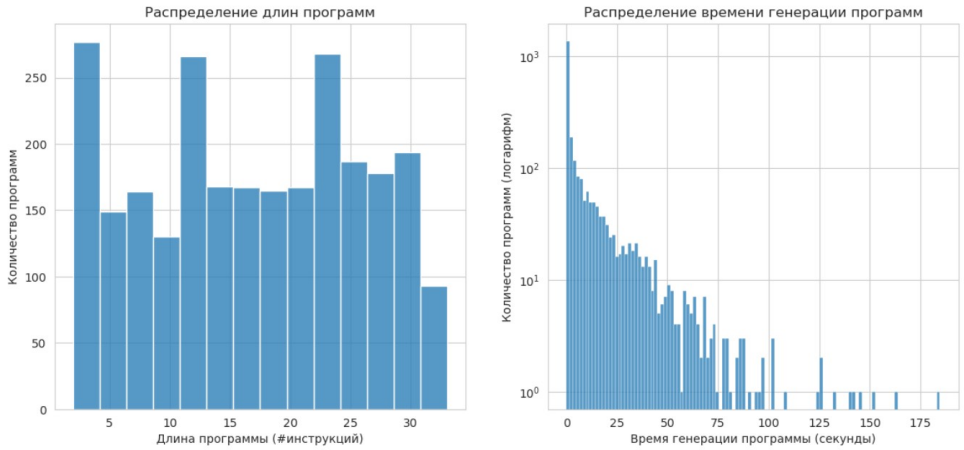


Рис. 2, Анализ 2 свойства



Литература

[1] Schulist J., Borkmann D., Starovoitov A., Linux Socket Filtering aka Berkeley Packet Filter (BPF) [Электронный ресурс] // URL: https://www.kernel.org/doc/Documentation/networking/filter.txt (Дата обращения: 11.05.2024)
[2] Gershuni E., Amit N., Gurfinkel A., Narodytska N., Navas J., Rinetzky N., Ryzhyk L., Sagiv M. Simple and Precise Static Analysis of Untrusted Linux Kernel Extensions // Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'19), 2019, Phoenix, AZ, USA. ACM, New York, NY, USA
[3] Vishwanathan H., Shachnai M., Narayana S., Nagarakatte S. Verifying the Verifier: eBPF Range Analysis Verification // Computer Aided Verification (CAV 2023), 2023, Lecture Notes in Computer Science, vol 13966, Springer, Cham.
[4] Nelson L., Geffen J., Torlak E., Wang Xi Specification and verification in the field: applying formal methods to BPF just-in-time compilers in the Linux kernel // Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI'20), 2020, art. 3, pp. 41-61
[5] Vyukov D., Konovalov A. Syzkaller: an unsupervised coverage-guided kernel fuzzer [Электронный ресурс] // URL: https://github.com/google/syzkaller (Дата обращения: 11.05.2024)
[6] Sun Hao, Xu Yiru, Liu J., Shen Y., Guan Nan, Jiang Yu Finding Correctness Bugs in eBPF Verifier with Structured and Sanitized Program // Proceedings of the 19th European Conference on Computer Systems (EuroSys'24), 2024, pp. 689-703
[7] Brady E. Idris, a general-purpose dependently typed programming language: Design and implementation. // Journal of Functional Programming, 2013, V. 23, pp. 552-593
[8] DepTyCheck [Электронный ресурс] // URL: https://github.com/buzden/deptycheck (Дата обращения: 11.05.2024)
[9] Vyukov D., Konovalov A. Syzkaller: an unsupervised coverage-guided kernel fuzzer [Электронный ресурс] // URL: https://github.com/google/syzkaller (Дата обращения: 11.05.2024)
[10] CVE-2021-34556, CVE-2021-35477 [Электронный ресурс] // URL: https://www.openwall.com/lists/oss-security/2021/08/01/3 (Дата обращения: 11.05.2024)
[11] CVE-2021-3489 [Электронный ресурс] // URL: https://www.openwall.com/lists/oss-security/2021/05/11/10 (Дата обращения: 11.05.2024)
[12] CVE-2021-3490 [Электронный ресурс] // URL: https://www.openwall.com/lists/oss-security/2021/05/11/11 (Дата обращения: 11.05.2024)