УДК 004.415.538

РАЗРАБОТКА ИНСТРУМЕНТА ДИФФЕРЕНЦИАЛЬНОГО МУТАЦИОННОГО ФАЗЗИНГА ФАЙЛОВЫХ СИСТЕМ DIFFUZZER

Ковалевский В.М. (ИТМО), **Кечин В.В.** (ИТМО)
Научный руководитель – кандидат технических наук, доцент Ицыксон В.М. (ИТМО)

Введение. Файловая система является критически важным компонентом компьютерных систем. Сбои, происходящие в работе файловых систем, могут приводить к катастрофическим последствиям, например, к потере или нарушению целостности данных. При этом обычного тестирования может быть недостаточно для выявления ошибок файловых систем из-за огромного числа возможных состояний и большого множества последовательностей операций. Кроме того, файловые системы обычно реализуются на языках низкого уровня, в которых, в частности, распространены ошибки при работе с памятью. По этим причинам существует необходимость создания автоматических средств анализа качества файловых систем.

В области анализа файловых систем существует много работ: модель поведения SibylFS [1], генератор тестов Dogfood [2], фаззер Hydra [3], инструмент проверки модели Metis [4], инструмент проверки согласованности данных при сбое CrashMonkey [5] и другие. При этом часть из существующих инструментов уже невозможно запустить на современных системах, а другие имеют существенные ограничения в обнаружении ошибок файловых систем.

Одним из самых эффективных методов динамического анализа программ является фаззинг-тестирование (или просто фаззинг). Основная идея фаззинга заключается в том, чтобы генерировать и передавать программе псевдослучайный набор входных данных, который может привести к серьёзным программным ошибкам. В силу особенностей организации и функционирования файловых систем, фаззинг является наиболее перспективным методом анализа качества и обнаружения ошибок в файловых системах.

В данной работе представлена реализация нового инструмента DIFFuzzer [6], созданного на основе методов дифференциального и мутационного фаззинга, который решает проблемы существующих инструментов и использует идеи, лежащие в их основе, для оптимального выявления ошибок файловых систем.

Основная часть. Современные файловые системы реализуют большое число операций, часть из которых описана в стандарте POSIX, который поддерживается большинством unix-подобных систем: Linux, BSD, MacOS и др. Основная сложность тестирования файловых систем заключается в том, что нужно учитывать семантику этих вызовов. По этой причине фаззер системных вызовов Syzkaller [7], который реализует фаззинг на основе грамматики, значительно хуже справляется с обнаружением ошибок в файловых системах, как было показано в Hydra [3].

Методы фаззинга можно разделить на три вида: черный ящик, белый ящик и серый ящик в зависимости от того, сколько информации они требуют от целевой программы во время выполнения. Фаззинг серого ящика является более эффективным, чем фаззинг черного ящика, и проще в реализации, чем фаззинг белого ящика. Фаззер AFL [8] является одним из лучших фаззеров серого ящика. Эффективность AFL достигается за счет мутации входных данных на основе информации о покрытии кода. К сожалению, AFL нельзя напрямую применить к файловым системам, ввиду особенностей генерации входных данных и фокусировании его только на инструментировании прикладного ПО.

Другой метод фаззинга — дифференциальный фаззинг, позволяет обнаруживать семантические ошибки, сравнивая поведение двух и более систем на одинаковых входных данных. Его легко применить для файловых систем, так как обычно они имеют одинаковый интерфейс вызовов. Подобный подход уже был применен в Metis [4] и показал свою эффективность.

В новом инструменте будут использованы все идеи, представленные выше: использование семантики вызовов при генерации входных данных, мутация данных на основе

информации о покрытии кода и дифференциальный фаззинг. Комбинация этих подходов позволит обеспечить эффективный поиск ошибок в файловых системах.

Выводы. Разработанный инструмент автоматического тестирования файловых систем на основе методов управляемого мутационного фаззинга может быть использован для проверки качества файловых систем, реализующих интерфейс POSIX.

Список использованных источников:

1. Tom Ridge, David Sheets, Thomas Tuerk, Andrea Giugliano, Anil Madhavapeddy, and Peter Sewell. 2015. SibylFS: formal specification and oracle-based testing for POSIX and real-world file systems. In Proceedings of the 25th Symposium on Operating Systems Principles (SOSP '15). Association for Computing Machinery, New York, NY, USA, 38–53. — URL: https://doi.org/10.1145/2815400.2815411 (дата обращения 17.01.2025).
2. Dongjie Chen, Yanyan Jiang, Chang Xu, Xiaoxing Ma, and Jian Lu. 2020. Testing file system implementations on layered models. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE '20). Association for Computing Machinery, New York, NY, USA, 1483–1495. — URL: https://doi.org/10.1145/3377811.3380350 (дата обращения 17.01.2025).
3. Seulbae Kim, Meng Xu, Sanidhya Kashyap, Jungyeon Yoon, Wen Xu, and Taesoo Kim. 2019. Finding semantic bugs in file systems with an extensible fuzzing framework. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP '19). Association for Computing Machinery, New York, NY, USA, 147–161. — URL: https://doi.org/10.1145/3341301.3359662 (дата обращения 17.01.2025).
4. Yifei Liu, Manish Adkar, Gerard Holzmann, Geoff Kuenning, Pei Liu, Scott A. Smolka, Wei Su, Erez Zadok. 2024. Metis: File System Model Checking via Versatile Input and State Exploration. 22nd USENIX Conference on File and Storage Technologies (FAST 24). — URL: https://www.usenix.org/conference/fast24/presentation/liu-yifei (дата обращения 17.01.2025).
5. Jayashree Mohan, Ashlie Martinez, Soujanya Ponnapalli, Pandian Raju, and Vijay Chidambaram. 2019. CrashMonkey and ACE: Systematically Testing File-System Crash Consistency. ACM Trans. Storage 15, 2, Article 14 (May 2019), 34 pages. — URL: https://doi.org/10.1145/3320275 (дата обращения 17.01.2025).
6. DIFFuzzer: differential filesystem fuzzer [Электронный ресурс]. — URL: https://github.com/Slava0135/DIFFuzzer (дата обращения 17.01.2025).
7. Syzkaller: kernel fuzzer [Электронный ресурс]. — URL: https://github.com/google/syzkaller (дата обращения 17.01.2025).
8. AFL: security-oriented fuzzer [Электронный ресурс]. — URL: https://github.com/google/AFL (дата обращения 17.01.2025).