Набор инструментов для поиска гонок по данным в операционных системах

Герлиц Евгений Анатольевич
ИСП РАН, 1009004, г. Москва, ул. Александра Солженицына, д. 25
gerlits@ispras.ru

Гонка по данным в операционной системе (ОС) — это ситуация, в которой разные потоки выполнения или прерывания обращаются к разделяемым данным без синхронизации, причём одно из обращений по записи. По разным источникам [1,2] подавляющее большинство ошибок параллельного программирования — это гонки по данным.

Следствием гонки по данным может быть ошибка, приводящая к уязвимости в ОС [3,4]. Например, ошибка переполнения буфера, эксплуатацией которой может быть исполнение произвольного кода.

Изолированные среды выполнения процессов в ОС нацелены на предотвращение распространения последствий ошибок в этих процессах на ядро ОС и другие процессы. Однако гонка по данным непосредственно в ядре ОС делает всю вычислительную систему уязвимой.

В ИСП РАН проводятся исследования методов поиска гонок по данным, по результатам которых создаются новые методы, а некоторые открытые методы адаптируются к системному программному обеспечению, в частности, к операционным системам.

У операционных систем есть особенности, которые существенно усложняют применение к ним инструментов, изначально разработанных для поиска гонок по данным в прикладных программах. Это большой объём кода, нестандартные способы синхронизации, код на языке Ассемблера. Из-за этих особенностей современный динамический детектор гонок по данным TSan [5] при применении к операционным системам выдаёт ложные предупреждения. Большое их количество не даёт обнаружить истинные гонки по данным, так как гонку по данным зачастую трудно подтвердить без отладки ядра ОС. Однако оказалось, что эта проблема до некоторой степени преодолима в случае относительно небольших встраиваемых операционных систем. В ИСП РАН инструмент TSan был адаптирован к ARINC-653 совместимой операционной системе реального времени (ОСРВ). Удалось сократить количество ложных предупреждений до приемлемого уровня, поддержать прерывания и уменьшить потребление памяти.

Основным динамическим детектором гонок по данным в ОС Linux на данный момент является инструмент KCSAN [6]. Инструмент KCSAN хорошо масштабируется на длительные выполнения операционной системы, но может упускать гонки по данным из-за того, что он случайным образом выбирает события обращения к памяти для проверки. Чтобы увеличить количество обнаруживаемых гонок по данным в операционных системах, в ИСП РАН был разработан динамический детектор RaceHunter [7], который систематически проверяет операции обращения к разделяемой памяти на гонки по данным.

Инструмент KCSAN усовершенствовал метод поиска гонок по данным DataCollider [8], который применялся для поиска гонок по данным в ОС Windows. Усовершенствование состояло в программной реализации точек останова и точек наблюдения, при помощи которых устанавливаются задержки для провоцирования гонок по данным. В ИСП РАН была выполнена адаптация оригинального метода DataCollider к ОС Linux. Адаптированный метод получил название RaceHound [9].

Статические методы поиска гонок по данным анализируют код ядра ОС целиком, поэтому обладают большей полнотой, чем методы динамического мониторинга. В ИСП РАН разработан новый статический метод поиска гонок по данным в ОС [10], для которого в некоторых предположениях доказано, что он не упускает гонки по данным. В реальных проектах по верификации этот метод выявил десятки гонок по данным.

Однако статические методы поиска гонок по данным зачастую выдают много ложных предупреждений, потому что анализируют код без его выполнения. Применение динамического детектора RaceHunter или адаптированного Tsan для отладки гонок по

данным [11] позволяет существенно облегчить и ускорить отделение ложных гонок по данным от истинных.

список литературы

1. Lu S. et al. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics //Proceedings of the 13th international conference on Architectural support for programming languages and operating systems. – 2008. – С. 329-339.
2. Мутилин В. С., Новиков Е. М., Хорошилов А. В. Анализ типовых ошибок в драйверах операционной системы Linux //Труды Института системного программирования РАН. – 2012. – Т. 22. – С. 349-374.
3. Farah T. et al. Study of race condition: A privilege escalation vulnerability //WMSCI 2017-21st World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings. – 2017. – С. 100-105.
4. Jeong D. R. et al. Razzer: Finding kernel race bugs through fuzzing //2019 IEEE Symposium on Security and Privacy (SP). – IEEE, 2019. – С. 754-768.
5. Serebryany K., Iskhodzhanov T. ThreadSanitizer: data race detection in practice //Proceedings of the workshop on binary instrumentation and applications. – 2009. – С. 62-71.
6. Marco Elver, et al. Concurrency bugs should fear the big bad data-race detector, 2020. https://lwn.net/Articles/816850
7. Gerlits E. A. RaceHunter Dynamic Data Race Detector //Programming and Computer Software. – 2024. – Т. 50. – №. 6. – С. 467-481.
8. Erickson J. et al. Effective {Data-Race} detection for the kernel //9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10). – 2010.
9. Komarov N. On the Implementation of Data-Breakpoints Based Race Detection for Linux Kernel Modules //Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering. – ИСП РАН, 2013. – №. 7.
10. Andrianov P., Mutilin V. Scalable thread-modular approach for data race detection //International Workshop on Frontiers in Software Engineering Education. – Cham : Springer International Publishing, 2019. – С. 371-385.
11. Герлиц Е. А., Мутилин В. С. Фреймворк автоматизации тестирования на гонки по данным //Труды Института системного программирования РАН. – 2025. – Т. 37. – №. 1. – С. 107-120.