УДК УДК 004.056

РАЗРАБОТКА ПОДХОДА К ВЕРИФИКАЦИИ РЕЗУЛЬТАТОВ СТАТИЧЕСКОГО АНАЛИЗА НА ОСНОВЕ НАПРАВЛЕННОГО ФАЗЗИНГА

Насонова М. М.[1]
Научный руководитель – Кашин С. В.[1]
[1]Университет ИТМО
senina.m.m@yandex.ru

Введение

Статический анализ – один из обязательных этапов сертификации программного обеспечения в Российской Федерации. И основной трудностью в проведении этого этапа остается высокий уровень ложных срабатываний, требующих ручной верификации [1]. Существующие подходы фаззинг-тестирования, такие как символьное исполнение, могут упростить эту верификацию, однако делают это не оптимально, рассматривая все возможные пути исполнения [2]. Направленный фаззинг позволяет исследовать только заданные пути программы, но требует предварительной подготовки кода для изоляции целевых участков [3]. Целью работы является разработка подхода, позволяющего использовать направленный фаззинг для поиска размеченных статическом анализом срабатываний в автоматическом режиме.

Основная часть

Предлагаемый подход реализован в виде LLVM модуля, обрабатывающего исследуемый код на основе данных от внешнего статического анализатора. В качестве входных данных модулю передаются исходный код и конфигурационный файл с координатами потенциально уязвимых участков (имя функции и номер строки), полученных из отчёта статического анализатора. Модуль строит граф потока управления (CFG) и анализирует достижимость целевых участков, определяя все базовые блоки, ведущие к ним. Далее модуль проводит инструментацию нерелевантных базовых блоков, заменяя их вызовом метода exit(0). Это позволяет при дальнейшем фаззинг-тестировании исследовать только целевые пути исполнения.

Таким образом, архитектура подхода позволяет выполнять все шаги анализа автоматически и интегрировать этот подход в конвейер безопасности. В самом простом случае для реализации всех шагов достаточно стандартной системы сборки, например, Makefile, которая сможет по порядку запустить: статический анализ, LLVM анализ, инструментацию исходного кода и фаззинг-тестирование.

В настоящее время проводятся экспериментальные исследования этого подхода на разных видах типовых уязвимостей (переполнение буфера, небезопасные вызовы, целочисленные переполнения), и различных структурах потоков выполнения. Выполняется сравнительный анализ двух сценариев: базового (при котором фаззинг-тестирование запускается без предварительной инструментации) и экспериментального с применением разработанного подхода. Предварительные наблюдения показывают, что текущая реализация подхода работает корректно и не искажает семантику целевых путей после инструментации. В дальнейшем планируется оценка таких метрик, как время до обнаружения уязвимости, процент тестовых случаев, достигающих цели, и точность идентификации уязвимых путей.

Ожидается, что предлагаемый подход позволит специалистам уменьшить время проверки срабатываний статического анализатора за счёт более быстрой работы фаззера, сфокусированного исключительно на целевых участках кода.

Выводы

В работе представлен подход к верификации результатов статического анализа на основе направленного фаззинга, с автоматизированной инструментацией кода для изоляции

целевых путей выполнения. Тестовая реализация подхода работает с программами на языке C, используя Clang Static Analyzer в качестве статического анализатора и AFL++ как инструмент для фаззинг-тестирования. Дальнейшие исследования будут направлены на оценку эффективности разработанного подхода и его адаптацию для работы с реальными программными проектами.

Литература

1. Воротникова Т.Ю. Надежный код: статический анализ программного кода как средство повышения надежности программного обеспечения информационных систем // Информационные технологии в УИС. 2020. №2. С. 22–27.
2. Герасимов А. Ю. и др. Комбинирование динамического символьного исполнения, статического анализа кода и фаззинга //Труды Института системного программирования РАН. – 2018. – Т. 30. – №. 6. – С. 25-38.
3. Manès, V., Han, H., Han, C., Cha, S., Egele, M., Schwartz, E., & Woo, M. "The Art, Science, and Engineering of Fuzzing: A Survey." IEEE Transactions on Software Engineering, vol. 47, 2019, pp. 2312-2331. https://doi.org/10.1109/tse.2019.2946563 (дата обращения: 17.02.2026).
4. А. Ю., Круглов Л. В., Ермаков М. К., Вартанов С. П. Подход к определению достижимости программных дефектов, обнаруженных методом статического анализа, при помощи динамического символьного исполнения // Труды ИСП РАН. 2017. №5. URL: https://cyberleninka.ru/article/n/podhod-k-opredeleniyu-dostizhimosti-programmnyh-defektov-obnaruzhennyh-metodom-staticheskogo-analiza-pri-pomoschi-dinamicheskogo (дата обращения: 17.02.2026).