

А. О. Ревазова¹✉

Метод интеллектуально-адаптивной оценки и фильтрации уязвимостей в системах статического анализа безопасности программного обеспечения

¹Национальный исследовательский ядерный университет МИФИ,
г. Москва, Российская Федерация
e-mail: aanrevazova@gmail.com

Аннотация. Данная статья посвящена разработке метода интеллектуально-адаптивной оценки и фильтрации уязвимостей, выявляемых средствами статического анализа безопасности программного обеспечения. Актуальность работы обусловлена высокой долей ложноположительных срабатываний и избыточным объемом результатов анализа, затрудняющих выявление и устранение действительно критических дефектов. Разработанный подход основан на применении методов машинного обучения и контекстного анализа кода для оценки обнаруженных уязвимостей с учетом их вероятности эксплуатации, критичности и особенностей программной среды. Метод включает этапы извлечения набора признаков, интеллектуальной классификации уязвимостей и адаптивной фильтрации результатов. Предлагаемое решение обеспечивает снижение количества ложноположительных срабатываний, повышение точности ранжирования уязвимостей и оптимизацию процесса принятия решений при обеспечении безопасности ПО. Результаты экспериментальной оценки демонстрируют эффективность метода по сравнению с традиционными подходами к статическому анализу исходного кода.

Ключевые слова: статический анализ, уязвимости программного обеспечения, ложноположительные срабатывания, интеллектуально-адаптивная фильтрация, машинное обучение, контекстный анализ кода

A. O. Revazova¹✉

A method for intelligent adaptive vulnerability assessment and filtering in static software security analysis systems

¹National Research Nuclear University MEPhI, Moscow, Russian Federation
e-mail: aanrevazova@gmail.com

Abstract. The article is devoted to the development of a method for intelligent adaptive assessment and filtering of vulnerabilities identified by static software security analysis. The relevance of the work is due to the high proportion of false positives and the excessive volume of analysis results, which make it difficult to identify and eliminate truly critical defects. The developed approach is based on the use of machine learning methods and contextual code analysis to assess the discovered vulnerabilities, taking into account their likelihood of exploitation, criticality and features of the software environment. The method includes the stages of feature set extraction, intelligent vulnerability classification, and adaptive filtering of results. The proposed solution reduces the number of false positives, increases the accuracy of vulnerability ranking, and optimizes the decision-making process while ensuring software security. The results of the experimental evaluation demonstrate the effectiveness of the method in comparison with traditional approaches to static source code analysis.

Keywords: static analysis, software vulnerabilities, false positives, intelligent adaptive filtering, machine learning, contextual code analysis

Введение

Современные программные системы характеризуются высокой сложностью и масштабируемостью, что приводит к увеличению количества уязвимостей, выявляемых в них. Одним из ключевых инструментов обеспечения безопасности исходного программного кода является статический анализ кода (SAST), позволяющий выявлять уязвимости без выполнения программы.

Однако одной из главных проблем SAST является большое количество ложноположительных результатов (*false positives*), что существенно снижает эффективность применения этих инструментов на практике. Разработчики вынуждены тратить значительное время на анализ нерелевантных срабатываний, что замедляет процесс разработки и снижает доверие к инструментам анализа.

В связи с этим возникает необходимость разработки методов интеллектуальной фильтрации и приоритизации уязвимостей, позволяющих адаптироваться к контексту проекта и повышать точность анализа.

Статический анализ представляет собой метод исследования исходного кода без его исполнения с целью выявления ошибок и уязвимостей и способен выявлять значительную часть известных уязвимостей, однако его эффективность сильно зависит от типа уязвимости и используемого инструмента [1]. При этом точность анализа варьируется, а доля ложноположительных срабатываний может достигать критических значений.

Современные SAST-инструменты генерируют большое количество предупреждений, значительная часть которых оказывается нерелевантной [2]. Это создает дополнительную нагрузку на разработчиков и требует внедрения механизмов фильтрации.

Кроме того, исследования показывают, что комбинирование SAST с ручным тестированием повышает качество обнаружения уязвимостей, но увеличивает затраты времени и человеческих ресурсов [3].

Методы и материалы

Ложноположительные результаты являются одной из ключевых проблем статического анализа. Выделяется несколько подходов к обработке ложноположительных срабатываний, включая ручную фильтрацию, использование правил и применение методов машинного обучения [4]. Однако большинство существующих решений либо требуют значительных усилий со стороны разработчиков, либо они плохо масштабируются.

Таким образом, возникает необходимость создания адаптивных методов, способных учитывать специфику проекта и автоматически корректировать результаты анализа.

Современные исследования демонстрируют активное внедрение методов машинного обучения в процессы анализа безопасности. В рамках DevSecOps машинное обучение позволяет автоматизировать анализ и повышать точность обнаружения уязвимостей [5].

Работы в области AI для DevSecOps показывают, что модели могут использоваться для:

- 1) классификации уязвимостей;
- 2) ранжирования предупреждений;
- 3) выявления паттернов уязвимого кода;
- 4) снижения ложноположительных результатов.

Существующие исследования в области улучшения статического анализа исходного кода, в том числе с использованием методов машинного обучения [6–8] показывают значительное улучшение в сравнении с классическими средствами.

Предлагаемый метод интеллектуально-адаптивной фильтрации срабатываний основан на сочетании статического анализа и машинного обучения для автоматической фильтрации и приоритизации уязвимостей с добавлением искусственного интеллекта для формирования расширенного признакового набора. Предполагается, что благодаря комбинации классических средств статического анализа исходного кода с интеллектуально-адаптивными методами получится существенно снизить долю ложноположительных срабатываний анализаторов исходного кода. В свою очередь применение искусственного интеллекта для формирования признакового пространства может улучшить качество набора данных и его объективность.

Первым этапом является сбор данных. Он включает в себя результаты SAST-инструментов, в том числе предупреждения, типы уязвимостей и контекст кода.

На основе данных от инструментов статического анализа формируется признаковое пространство. Производится извлечение характеристик с последующим использованием в моделях.

Выбирается модель, на основе которой будет проводиться эксперимент и производится ее обучение.

Необходимо адаптировать модель машинного обучения, а также производить постоянное обновление модели на основе обратной связи от разработчиков.

Финальным этапом является ранжирование предупреждений и их приоритизация уязвимостей по степени риска.

По результатам работы средства на основе метода интеллектуально-адаптивного выявления ложноположительных срабатываний проводится сравнение с применением многокритериальной методики оценки результатов исследования.

Результаты

Ключевой особенностью метода является его адаптивность. В отличие от статических правил, данная модель учитывает особенности конкретного проекта, обучается на реальных данных и динамически обновляется.

Данная особенность позволяет существенно повысить точность анализа по сравнению с традиционными подходами.

Благодаря тому, что используется размеченный набор данных, включающий себя как истинно положительные, так и ложноположительные и ложноотрица-

тельные срабатывания, возможно корректно оценить работу предлагаемого метода.

Используемая модель машинного обучения на основе признакового пространства анализирует характеристики найденных предупреждений, после чего определяет, является ли данная уязвимость истинной или она относится к ложноположительным.

Модель использует исторические данные об уязвимостях, которые были ранее подтверждены или отвергнуты экспертом в ручном режиме и формирует набор признаков с качественными характеристиками. Благодаря такому подходу, отсутствует строгая привязка к конкретному языку программирования, архитектуре или особенностям конкретного проекта. Тестирование проводилось на языках программирования Python, C#, Java и др.

Эксперимент проводился с использованием виртуальной сетевой лаборатории, выстроенной в рамках методологии DevSecOps [9–11]. Обобщенная схема виртуального испытательного стенда представлена на рис. 2.

Предусматривается интеграция метода в конвейер DevSecOps для непрерывного мониторинга и анализа уязвимостей при добавлении нового кода:

- 1) на этапе CI/CD;
- 2) при автоматической проверке кода;
- 3) в системах контроля качества.

Для оценки эффективности метода используются следующие стандартные метрики машинного обучения: точность (precision) и полнота (recall).

Данные метрики позволяют в достаточной степени оценить эффективность разработанного метода в количественных характеристиках.

Эксперимент проводился на синтетических наборах данных, исходных кодах, размещенных в открытых репозиториях и фрагментах кода, сгенерированных нейронными сетями. Исследования показывают, что языковые модели способны генерировать текст близкий к оригинальному на котором можно проводить тестирование [11–13].

Благодаря использованию синтетических данных и использованию языковых моделей для генерации исходного кода удалось добиться формирования набора данных порядка 10^5 строк. Данный набор данных включал в себя классические Web уязвимости, уязвимости, связанные с сторонними компонентами, а также проблемы, связанные с качеством исходного кода.

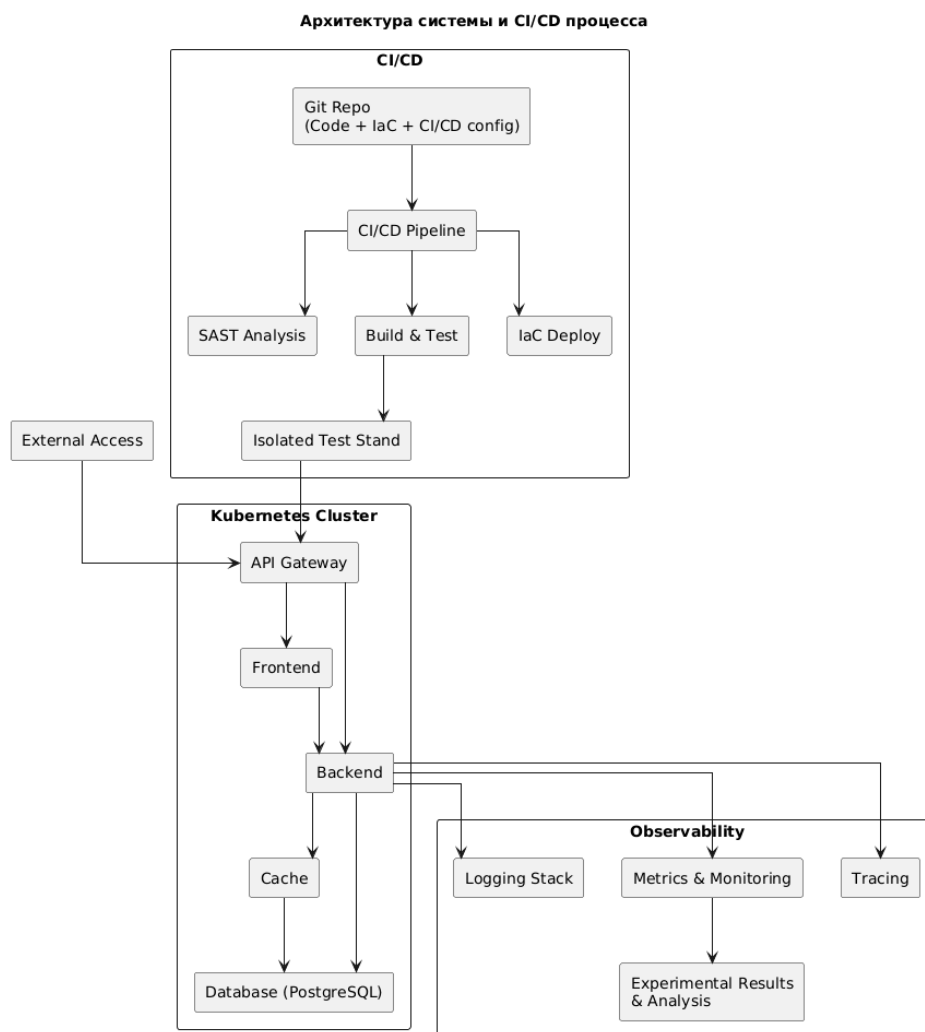


Рис. 2. Обобщенная схема виртуального испытательного стенда

В результате исследований на наборе, полученным от SAST инструментов, удалось снизить уровень ложноположительных срабатываний на 59,3% по сравнению с классическим статическим анализатором исходного кода.

Несмотря на то, что среднее количество пропущенных уязвимостей достигало 14,7% от общего объема набора данных, при условии снижения общего уровня ложно положительных срабатываний и отсутствии среди пропущенных уязвимостей с уровнем «критичный», можно утверждать о применимости данной модели.

Заключение

В работе предложен метод интеллектуально-адаптивной оценки и фильтрации уязвимостей в системах статического анализа безопасности программного обеспечения. Показано, что предлагаемый метод позволяет повысить уровень выявленных ложноположительных срабатываний в системах статического анализа безопасности программного обеспечения на 59%.

Практическая значимость метода заключается в возможности его применения в существующие системы статического анализа кода, внедренных в процессы DevSecOps.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Goseva-Popstojanova K., Perhinschi A. On the capability of static code analysis to detect security vulnerabilities // *Information and Software Technology*. – 2015. – Vol. 68. – P. 18-33.
2. Aloraini B., Nagappan M., German D. M., Hayashi S., Higo Y. An empirical study of security warnings from static application security testing tools // *Journal of Systems and Software*. – 2019. – Vol. 158.
3. Ali E. A., Ammar F. M. SAST Tools and Manual Testing to Improve the Methodology of Vulnerability Detection in Web Applications // *The International Journal of Engineering & Information Technology (IJEIT)*. – 2024. – Vol. 12 №1. P.79-83
4. Alenezi M., et al. Software security static analysis false alerts handling approaches // *International Journal of Advanced Computer Science and Applications*. – 2021. – Vol. 12, № 11.
5. Fu M., Pasuksmit J., Tantithamthavorn C. AI for DevSecOps: A Landscape and Future Opportunities // *ACM Transactions on Software Engineering and Methodology*. – 2025. – Vol. 34. №4.
6. Hüther L., Sohr K., Berger B., Rothe H. Machine Learning for SAST: A Lightweight and Adaptable Approach // *Lecture Notes in Computer Science*. – 2024. – P. 89–108.
7. Zhu J., Li K., Chen S., Fan L. A Comprehensive Study on Static Application Security Testing (SAST) Tools for Android // *Journal of LaTeX class files*. – 2024. – Vol. 14. №8.
8. Chen J., Xiang H., Li L., Zhang Y. Utilizing Precise and Complete Code Context to Guide LLM in Automatic False Positive Mitigation // *arXiv preprint arXiv:2411.03079*. – 2024.
9. Басыня Е. А. Распределенная система сбора, обработки и анализа событий информационной безопасности сетевой инфраструктуры предприятия / Безопасность информационных технологий. 2018. Т. 25. № 4. С. 43-52.
10. Басыня Е. А. Обеспечение достоверности результатов научно-практических изысканий с применением программной инженерии = Ensuring the reliability of scientific research results using software engineering / Е. А. Басыня, Е. А. Малышев. – Текст : непосредственный // Защита информации. Инсайд = Zasita informacii. Inside. – 2023. – № 4 (112). – С. 14–21.
11. Armstrong N., Hartley T. Artificially Insecure: Examining GitHub Copilot’s AI-based Vulnerability Prevention System // *Proceedings of the 2025 Silicon Valley Cybersecurity Conference (SVCC)*. – 2025.
12. Al Madi N. How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot // *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. – 2022.
13. Tony C., Mutas M., Díaz Ferreyra N. E., Scandariato R. LLMSecEval: A Dataset of Natural Language Prompts for Security Evaluations // *Proceedings of the 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. – 2023.

© А. О. Ревазова, 2026