ТЕХНИКАЛЫҚ ҒЫЛЫМДАР
ТЕХНИЧЕСКИЕ НАУКИ
TECHNICAL SCIENCE

CENTRAL ASIAN SCIENTIFIC JOURNAL
ISSN 3135-3061
2026. №4 (32), Том 2. С. 480–486
DOI: https://doi.org/10.5281/zenodo.24348073
Сайт журнала: https://cajournal.kz

УДК 004.056:004.8

КОНТЕКСТНО-ЗАВИСИМАЯ ОЦЕНКА РИСКОВ СТОРОННИХ ПРОГРАММНЫХ КОМПОНЕНТОВ НА ОСНОВЕ ДОСТУПНОСТИ ГРАФИКОВ ВЫЗОВОВ И КОЛЛЕКТИВНОГО МАШИННОГО ОБУЧЕНИЯ

Р.А. Тулеутай, С.О. Махмутов, Н.П. Ким, Р. Сабыр
Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан

Тулеутай Рустам Аяшулы — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Махмутов Санжар Оралбекулы — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Ким Никита Павлович — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Сабыр Ринат — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

© Р.А. Тулеутай, С.О. Махмутов, Н.П. Ким, Р. Сабыр, 2026

Аннотация: Современные программные продукты в основном состоят из сторонних и открытых компонентов, и каждая уязвимость в транзитивной зависимости может стать риском для конечного приложения. Инструменты анализа композиции программного обеспечения обнаруживают известные уязвимости, но создают длинные списки оповещений, многие из которых касаются кода, который приложение так и не выполняет. В результате разработчики страдают от усталости от оповещений и могут упускать из виду опасные недостатки. Цель данного исследования — разработать интеллектуальную систему, которая оценивает риски программных компонентов в контексте конкретного приложения. Литература показывает, что структурные, вероятностные и репутационные факторы риска исследовались отдельно, но они не объединены в одну обучаемую модель. Предлагаемый подход строит граф зависимостей на основе списока материалов программного обеспечения, дополняет его графом вызовов для определения, доступна ли уязвимая функция из кода приложения, а также объединяет достижимость, вероятность эксплуатации, серьёзность и репутацию репозитория в ансамбль с повышением градиента. Ансамбль даёт составную оценку риска, и вклад каждого фактора объясняется пользователю. В статье описывается архитектура, модель признаков, алгоритм оценки и экспериментальный протокол. Основная гипотеза заключается в том, что контекстная оценка снизит количество ложноположительных оповещений как минимум на 50%, при этом воспоминание о уязвимостях сохраняется на уровне 0,9 и выше. Представленные результаты являются ожидаемыми результатами исследовательского проектирования, а эмпирическая проверка планируется следующим этапом. Работа имеет практическую ценность для безопасных конвейеров разработки программного обеспечения.

Ключевые слова: безопасность цепочки поставок программного обеспечения, анализ композиции программного обеспечения, приоритетизация уязвимостей, доступность по графу вызовов, ансамблевое машинное обучение, комплексный рейтинг риска, список материалов программного обеспечения

Введение

Современные приложения собираются из большого количества сторонних библиотек, связанных через менеджеры пакетов. Каждая библиотека приносит свои транзитивные зависимости, поэтому безопасность продукта зависит от безопасности всей его цепочки поставок. Атаки на цепочки поставок с открытым исходным кодом разнообразны и систематизированы в ряде таксономий [1]. Два случая иллюстрируют масштаб последствий: уязвимость Log4Shell (CVE-2021-44228) в библиотеке Apache Log4j затронула очень большое количество приложений, а бэкдор, обнаруженный в пакете xz-utils (CVE-2024-3094), показал, что угроза может возникнуть не из-за случайного дефекта, а из преднамеренного скомпромета проекта с открытым исходным кодом. Защищённые фреймворки разработки, например NIST Secure Software Development Framework [2], требуют от организаций проверки и мониторинга сторонних компонентов, которые они повторно используют.

Инструменты анализа состава программного обеспечения сравнивают список компонентов с публичными базами данных уязвимостей и сообщают о каждом совпадении. Однако формальное присутствие уязвимой библиотеки не означает, что приложение подвержено её воздействию. На практике проект может получать очень большое количество оповещений, большинство из которых не имеют значения, и инженеры постепенно перестают на них реагировать. Это явление известно как усталость от оповещений. Точное приоритизирование уязвимостей, действительно опасных для конкретного приложения, является практической необходимостью современной безопасной разработки.

Связанные работы

Пакетные экосистемы как сети зависимостей. Исследования экосистемы NPM показали, что пакеты образуют плотно связанную сеть, в которой небольшое число пакетов и сопровождающих может влиять на значительную часть экосистемы [3]. Дальнейший анализ показал, что уязвимости распространяются через транзитивные зависимости и могут долго оставаться неисправными [4]. Эти работы описывают структурную уязвимость на уровне пакетов, но не показывают, действительно ли уязвимая функция вызывается конкретным приложением.

Степень серьёзности и вероятность эксплуатации. Система оценки уязвимостей Common Vulnerability Scoring описывает техническую серьёзность уязвимости [5], но не вероятность её эксплуатации. Система оценки по прогнозированию уязвимостей оценивает эту вероятность с помощью модели, основанной на данных [6]. Однако оценка присваивается идентификатору уязвимости и, следовательно, одинакова для всех приложений, использующих затронутую библиотеку.

Анализ, ориентированный на код. Понта и соавторы предложили анализировать известные уязвимости на уровне кода и проверять, как уязвимый код используется приложением [7]. Пащенко и др. показали, что для разработчиков важна только часть уязвимых зависимостей, сообщаемых инструментами на основе метаданных, [8]. Ни один из подходов не сочетается с вероятностной оценкой эксплуатации или с репутацией репозитория.

Требования к репутации и процессу. OpenSSF Scorecard измеряет практики безопасности репозиториев с открытым исходным кодом [9], а стандарт проверки компонентов программного обеспечения OWASP определяет требования к проверке использования сторонних компонентов [10]. Эти источники предоставляют полезные сигналы и требования к процессу, но не определяют количественный алгоритм для оценки риска компонента.

Формулировка проблемы и исследовательский пробел

Таблица 1 подводит итоги того, что было достигнуто по каждому фактору риска и что остаётся открытым. Каждый фактор был изучен, но отдельно от остальных.

Таблица 1 — Сводка исследовательских пробелов

Фактор риска	Источники	Что покрывается	Оставшийся разрыв
Степень тяжести	[5]	Техническое влияние уязвимости	Не учитывается вероятность эксплуатации или контекст применения
Вероятность эксплуатации	[6]	Вероятность эксплуатации по идентификатору уязвимости	Игнорирует, используется ли уязвимый код приложением
Структурное воздействие	[3], [4]	Распространение через транзитивные зависимости	Подключение на уровне пакета без доступности на уровне функций
Доступность кода	[7], [8]	Использование уязвимого кода; доля соответствующих зависимостей	Без сочетания с вероятностью эксплойта и репутацией
Репутация репозитория	[9]	Практики безопасности репозиториев	Рассматривается отдельно от характеристик уязвимости
Требования к процессу	[2], [10]	Требования к контролю сторонних компонентов	Отсутствие количественной модели риска

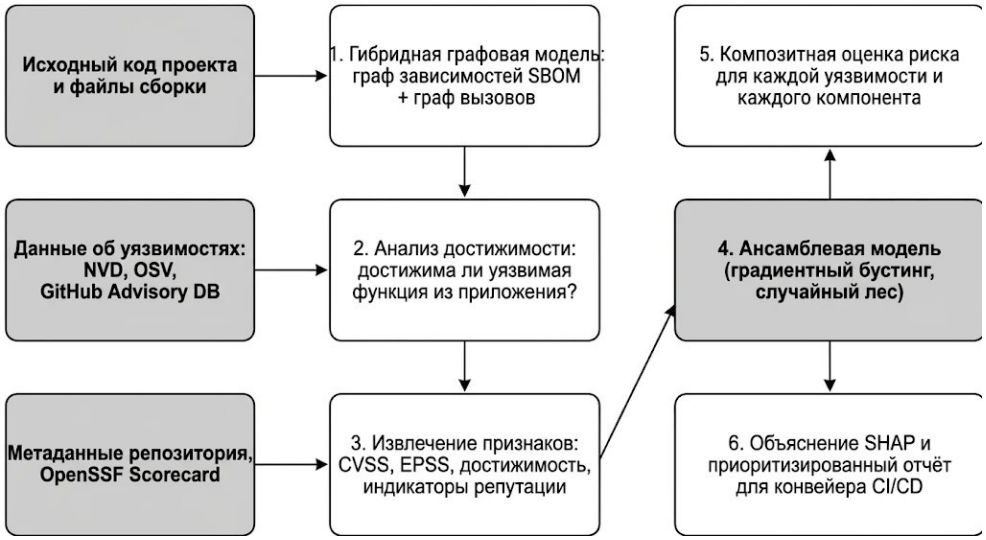
Таким образом, научная проблема заключается в противоречии между практической необходимостью приоритизировать уязвимости, действительно опасные для конкретного приложения, и отсутствием модели, интегрирующей структурный, вероятностный и репутационный контекст в единую оценку риска. Объект исследования — процесс управления безопасностью сторонних компонентов в жизненном цикле разработки программного обеспечения. Предметом исследования являются модели, методы и алгоритмы контекстно-осознанной оценки рисков на основе доступности call-graph и ансамблевого машинного обучения.

Гипотеза исследования сформулирована следующим образом: если риск компонента рассчитывается с помощью ансамблевой модели, агрегирующей достижимость уязвимой функции, вероятность эксплуатации и репутацию репозитория, а не только балл серьёзности, то доля ложноположительных оповещений снизится как минимум на 50%, в то время как отзыв уязвимостей останется на уровне 0,9 и выше. Вклад данной статьи заключается в разработке гибридной модели графа, модели признаков с алгоритмом оценки, интерпретируемого вывода для пользователя и экспериментального протокола для проверки гипотезы.

Материалы и методы

Архитектура системы

Архитектура предлагаемой системы показана на рисунке 1. Система получает исходный код и файлы сборки проекта, строит граф зависимостей и граф вызовов, проверяет доступность уязвимых функций, извлекает признаки, применяет ансамблевую модель и готовит приоритетный отчет для непрерывной интеграции и конвейера доставки.



Изображение 1 — Архитектура предлагаемой системы оценки рисков с учётом контекста

Гибридная модель графа

Проект представлен в виде направленного графа G = (V, E). Узлы — это компоненты, идентифицируемые по имени и версии, а ребра — это отношения зависимостей, взятые из программного перечета. Для каждого проекта граф вызовов приложения и его зависимостей строится методом статического анализа. Уязвимость связывается с функциями, указанными в её консультативной записи в базе данных OSV или GitHub Advisory Database. Уязвимость считается доступной, если хотя бы один путь вызова ведёт от входной точки приложения к уязвимой функции. Помимо этого двоичного флага, сохраняются минимальная длина пути вызова и количество входных точек, достигающих функции. Если консультативная запись не называет затронутую функцию, уязвимость помещается в отдельную категорию неизвестной доступности и не отбрасывается, чтобы избежать ложных отрицательных результатов.

Модель признаков и алгоритм оценки

Каждая уязвимость в проекте описывается вектором признаков, объединяющим пять групп признаков (Таблица 2).

Таблица 2 — Группы признаков модели риска

Группа	Особенности	Источник
Степень тяжести	Базовый балл тяжести	Технические характеристики [5]
Вероятность эксплуатации	Вероятность эксплуатации и процентиль	Модель [6]
Структура	Глубина зависимости; количество зависимых компонентов	Граф зависимостей
Доступность	флаг доступности; минимальная длина пути вызова; количество точек доступа	График вызовов
Репутация	Агрегированные оценки репозитория и индивидуальные проверки безопасности; время с момента последнего релиза	Scorecard [9]; метаданные репозитория

Уязвимость признаётся положительной, если она подтверждена как использованная согласно публичному каталогу известных эксплуатируемых уязвимостей или если эксперт рассматривает выборку предупреждений. На этих данных обучается ансамбль деревьев решений с усилением градиента [12], а случайный лес служит дополнительной базовой моделью. Вероятность того, что уязвимость релевантна для приложения, рассчитывается следующим образом

$$p_v = f(x_v) \quad (1)$$

а риск компонента агрегируется из рисков его уязвимостей как

$$R(c) = 1 - \prod_{v \in V(c)} (1 - p_v) \quad (2)$$

где pv — предсказанная вероятность того, что уязвимость v релевантна; f — обученный ансамбль; xv — вектор признаков уязвимости v; V(c) — множество уязвимостей компонента c; R(c) — составный балл риска компонента c. Уравнение (2) предполагает, что уязвимости компонента независимы, что является сознательным упрощением. Для объяснения каждого балла вычисляются значения SHapley Additive exPlanations [11], чтобы в отчёте отображался вклад каждой группы признаков.

Экспериментальная конструкция

Гипотеза будет протестирована на выборке проектов с открытым исходным кодом из экосистем Maven, PyPI и npm. Данные об уязвимостях будут взяты из Национальной базы данных уязвимостей, базы данных OSV и GitHub Advisory Database. Для сравнения будут использованы три базовые линии: оповещение по порогу серьёзности, оповещение по порогу вероятности эксплуатации и фильтр только по доступности. Качество модели будет измеряться по точности, воспоминанию, баллу F1 и площади под кривой ROC, а ложноположительное снижение рассчитывается относительно лучшей базовой линии. Перекрёстная валидация будет группирована по проектам, чтобы уязвимости одного проекта никогда не появлялись ни в обучающем, так и в тестовых наборах. Различия между методами будут проверены с помощью парного статистического теста по проектам, а случайная выборка оповещений будет проверена экспертами. Время анализа будет измеряться на проектах среднего размера.

Результаты и обсуждение

Ожидаемые результаты

Исследование находится на стадии проектирования, и экспериментальные данные пока не собраны. В этом разделе представлены ожидаемые результаты и критерии, которые будут использоваться для их проверки (Таблица 3). Целевые значения вытекают из гипотезы и практических требований к конвейерам непрерывной интеграции.

Таблица 3 — Ожидаемые результаты и критерии проверки

Ожидаемый результат	Метрика верификации	Целевое значение
Снижение ложноположительных предупреждений	Относительное снижение ложноположительных результатов по сравнению с лучшим исходным уровнем	Не менее 50%
Сохранено обнаружение использованных уязвимостей	Отзыв	Как минимум 0,9
Допустимое время анализа	Время для проекта среднего размера	До 10 минут
Интерпретируемость результата	Доля оповещений с объяснением на уровне факторов	100%
Маркированный набор данных	Записи об уязвимости, проектах, доступности и фактах эксплуатации	Опубликовано открыто

Помимо модели и прототипа, ожидается, что исследование создаст маркированный набор данных, связывающий уязвимости с проектами, состоянием доступности и фактом эксплуатации. Такой набор данных в открытом виде недоступен в источниках, проанализированных в этой статье.

Обсуждение

Гипотеза подтверждается результатами предыдущих исследований: анализы, ориентированные на код, показывают, что многие оповещения, сообщаемые только из метаданных, не касаются используемого кода [7, 8], в то время как модели эксплойт-предсказания дают независимую оценку привлекательной уязвимости для злоумышленников [6]. Объединение этих сигналов с репутацией репозитория [9] в единой обучаемой модели — это шаг, который не предпринимает рецензируемая литература. В отличие от закрытых коммерческих решений, предлагаемый подход предназначен быть воспроизводимым и объяснимым, поскольку вклад каждого фактора показывается пользователю.

Следует учитывать несколько ограничений. Статические графики вызовов могут пропускать пути вызовов в языках, использующих отражение или динамическую

диспетчеризацию, что приведёт к ложным отрицательным результатам; по этой причине уязвимости неизвестной доступности сохраняются и не фильтруются. Публичные каталоги эксплуатируемых уязвимостей неполны, поэтому метки содержат шум. Вероятность эксплуатации меняется со временем, и модель требует периодической переобучения. Результаты могут различаться в зависимости от экосистемы, и предположение независимости в уравнении (2) может не применяться к связанным уязвимостям.

С практической точки зрения система предназначена для интеграции в существующие конвейеры и помощи организациям в выполнении требований к контролю компонентов третьих сторон, установленных в безопасных фреймворках разработки [2]. Следующий этап работы — реализация прототипа и эксперимента, описанных в разделе 2.4.

Заключение

В статье предлагается проектирование интеллектуальной системы для контекстно-осознанной оценки рисков сторонних компонентов программного обеспечения. Анализ литературы показал, что структурные, вероятностные и репутационные факторы риска изучались отдельно и не объединены в одной модели. Предложенный подход объединяет граф зависимостей, достижимость по call-graph и модель ансамблевого машинного обучения в составную оценку риска с объяснением каждого фактора. В планируемом эксперименте будет проверена гипотеза о снижении числа ложноположительных оповещений при повторении не менее 0,9 не менее чем на 50%. Результаты эксперимента определят, можно ли рекомендовать подход для использования в безопасных конвейерах разработки.

Финансирование и признания

Исследование проводилось в рамках студенческого исследовательского проекта в Университете имени Сатбаева без внешнего финансирования. Авторы благодарят А.С. Абдуллаеву за методологическое руководство.

Ссылки:

1. Ладиса П., Плейт Х., Мартинес М., Бараис О. СоК: Таксономия атак на цепочки поставок открытого программного обеспечения. В: Симпозиум IEEE 2023 по безопасности и конфиденциальности (SP). Сан-Франциско: IEEE, 2023, стр. 1509–1526.

2. Суппая М., Скарфон К., Додсон Д. Secure Software Development Framework (SSDF) Версия 1.1: Рекомендации по снижению рисков уязвимостей программного обеспечения. Специальная публикация NIST 800-218. Гейтерсбург: Национальный институт стандартов и технологий, 2022.

3. Циммерманн М., Стаику К.-А., Тенни С., Прадель М. Малый мир с высокими рисками: исследование угроз безопасности в экосистеме NPM. В: 28-й симпозиум по безопасности USENIX. Санта-Клара: Ассоциация USENIX, 2019, стр. 995–1010.

4. Декан А., Менс Т., Константину Э. О влиянии уязвимостей безопасности в сети зависимостей пакета npm. В: Труды 15-й Международной конференции по хранилищам горного программного обеспечения (MSR). Гётеборг: ACM, 2018, стр. 181–191.

5. FIRST.Org. Система оценки уязвимостей общего уровня v3.1: Технический документ, 2019. Доступно по адресу: https://www.first.org/cvss/v3.1/specification-document (дата обращения: 14 октября 2026).

6. Джейкобс Дж., Романоски С., Эдвардс Б., Аджерид И., Ройтман М. Система прогнозирования эксплойтов (EPSS). Цифровые угрозы: исследования и практика, 2021, том 2, No 3, ст. 20.

7. Понта С.Е., Плейт Х., Сабетта А. За пределами метаданных: код-центрированный и ориентированный на использование анализ известных уязвимостей в программном обеспечении с открытым исходным кодом. В: Международная конференция IEEE 2018 по обслуживанию и развитию программного обеспечения (ICSME). Мадрид: IEEE, 2018, стр. 449–460.

8. Пащенко И., Плейт Х., Понта С.Е., Сабетта А., Массаччи Ф. Уязвимые зависимости открытого исходного кода: подсчёт важных. В: Материалы 12-го Международного

симпозиума ACM/IEEE по эмпирической инженерии и измерениям программного обеспечения (ESEM). Оулу: ACM, 2018, стр. 1–10.

9. Фонд безопасности с открытым исходным кодом. Таблица оценок OpenSSF. Доступно по адресу: https://github.com/ossf/scorecard (дата обращения: 14 октября 2026 года).

10. Фонд OWASP. Стандарт верификации компонентов программного обеспечения (SCVS). Доступно по адресу: https://owasp.org/www-project-software-component-verification-standard/ (дата обращения: 14 октября 2026).

11. Лундберг С.М., Ли С.-И. Единый подход к интерпретации прогнозов моделей. В: Достижения в системах нейронной обработки информации 30. Лонг-Бич, 2017, стр. 4765–4774.

12. Чен Т., Гестрин К. XGBoost: масштабируемая система бустера деревьев. В: Материалы 22-й международной конференции ACM SIGKDD по открытию знаний и анализу данных. Сан-Франциско: ACM, 2016, стр. 785–794.

© Central Asian Scientific Journal, 2026