ТЕХНИКАЛЫҚ ҒЫЛЫМДАР
ТЕХНИЧЕСКИЕ НАУКИ
TECHNICAL SCIENCE

CENTRAL ASIAN SCIENTIFIC JOURNAL
ISSN 3135-3061
2026. №4 (32), Том 3. С. 33–39
DOI: https://doi.org/10.5281/zenodo.24535593
Сайт журнала: https://cajournal.kz

УДК 004.056:004.8

AUTOSEC-WEB: АРХИТЕКТУРА ПЛАТФОРМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ С УНИФИЦИРОВАННОЙ НОРМАЛИЗАЦИЕЙ РЕЗУЛЬТАТОВ И ПРИОРИТИЗАЦИЕЙ РИСКОВ

Н.Д. Бекентаев, Ш.Б. Маткаримов, К.Р. Талипов, Ә.Е. Құрал
Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан

Бекентаев Нурбек Дулатулы — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Маткаримов Шерзот Баходирович — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Талипов Кямран Рашитович — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

Құрал Әмірхан Ергенұлы — студент, кафедра кибербезопасности, Казахский национальный исследовательский технический университет имени К. И. Сатпаева, Алматы, Казахстан.

© Н.Д. Бекентаев, Ш.Б. Маткаримов, К.Р. Талипов, Ә.Е. Құрал, 2026

Аннотация: Веб-приложения остаются основной мишенью атак, а существующие автоматические сканеры фрагментированы и выдают разнородные отчёты с большим числом ложных срабатываний. Цель работы — предложить архитектуру платформы автоматизированного тестирования безопасности веб-приложений AutoSec-Web. Использован подход науки о проектировании (design science): требования выведены из рекомендаций Open Web Application Security Project (OWASP) и Национального института стандартов и технологий США (NIST), а также из обзора литературы. Предложены модульная архитектура с оркестрацией сканирующих движков с открытым кодом, единая схема данных с привязкой к перечню типовых слабостей (CWE) и категориям OWASP, дедупликация находок, формула приоритизации риска и механизм контроля качества (quality gate) в конвейерах непрерывной интеграции и поставки (CI/CD). Определён протокол оценки на заведомо уязвимых приложениях и OWASP Benchmark; экспериментальные измерения на данном этапе не приводятся.

Ключевые слова: безопасность веб-приложений, автоматизированное тестирование безопасности, DAST, сканирование уязвимостей, DevSecOps, OWASP Top 10, приоритизация рисков

1. Введение

Веб-приложения сосредоточивают бизнес-логику, персональные данные и аутентификацию в одном компоненте, доступном из сети Интернет, что делает их постоянной целью атак. Проект Open Web Application Security Project (OWASP) Top 10, широко используемый документ по рискам безопасности веб-приложений, продолжает относить к наиболее критичным внедрение кода, нарушение контроля доступа, небезопасную конфигурацию и родственные классы [1]. Поэтому оценка защищённости —

рутинная задача команд разработки, а международные рекомендации, такие как SP 800-115 Национального института стандартов и технологий США (NIST), описывают планирование, проведение и анализ технических тестов безопасности, включая сканирование уязвимостей и тестирование на проникновение [2].

На практике эффективность тестирования ограничивают две проблемы. Во-первых, ручное тестирование на проникновение дорого и обычно выполняется в конце цикла выпуска, тогда как современные команды выкладывают изменения несколько раз в день. Во-вторых, автоматических инструментов много, но каждый покрывает свою часть задачи (обход приложения, активная имитация атак, пассивная проверка заголовков и cookie, анализ сторонних компонентов), выдаёт отчёт в собственном формате и шкале серьёзности и даёт ложные срабатывания, которые приходится разбирать вручную. В итоге автоматическое тестирование запускают от случая к случаю, его результаты игнорируют, и найденные проблемы не доходят до разработчиков в пригодном для работы виде.

1.1. Возможности и ограничения автоматических сканеров

Ранние систематические исследования показали, что сканеры «чёрного ящика» ограничены не столько набором атакующих нагрузок, сколько способностью исследовать приложение. Doupé и соавт. проанализировали несколько сканеров на специально созданном уязвимом приложении и пришли к выводу, что недостатки обхода (неполная обработка форм и многошаговых сценариев) мешали обнаружению уязвимостей, которые сканеры в принципе способны эксплуатировать [3]. Bau и соавт. оценили коммерческие сканеры и сообщили о неравномерном покрытии по классам уязвимостей, при более слабых результатах для уязвимостей, требующих сохранённого или вторичного взаимодействия [4].

Более новые работы сравнивают движки с открытым кодом. Potti и соавт. сопоставили две версии инструмента динамического тестирования безопасности приложений (dynamic application security testing, DAST) Zed Attack Proxy (ZAP) от OWASP на OWASP Benchmark по доле истинных срабатываний, доле ложных срабатываний, точности и индексу Юдена и показали, что качество обнаружения зависит от категории уязвимости и может ухудшаться между версиями [5]. Этот результат подтверждает необходимость непрерывной автоматической переоценки используемых движков вместо однократного выбора инструмента.

1.2. Тестирование безопасности в конвейерах DevSecOps

Систематический обзор технологий и инструментов DevSecOps охватил 87 исследований, из которых проанализировано 42; выделено тринадцать устойчивых категорий инструментов безопасности, а искусственный интеллект оказался наиболее часто упоминаемой технологией [6]. Обзор поддерживает тезис о том, что проверки безопасности следует встраивать в каждый этап конвейера непрерывной интеграции и поставки (continuous integration/continuous delivery, CI/CD), а не сосредоточивать в конце, и одновременно показывает широту ландшафта инструментов, которые командам приходится интегрировать.

1.3. Машинное обучение для обнаружения уязвимостей

Машинное обучение всё чаще применяют к отдельным классам уязвимостей. Например, Alhamyani и Alshammari обучили несколько классификаторов на открытом наборе из 138 569 образцов для обнаружения межсайтового скриптинга (cross-site scripting, XSS) и сообщили о точности 99,78 % для модели Random Forest [7]. Такие результаты получены на одном наборе данных без кросс-валидации по другим наборам, а классификаторы подобного рода выявляют вредоносные нагрузки, но не подтверждают эксплуатируемые уязвимости в работающем приложении. Поэтому мы рассматриваем их как дополнительные компоненты, которые можно интегрировать позднее, а не как замену активному тестированию.

1.4. Нерешённая проблема, цель и задачи исследования

В литературе остаётся практический пробел: движки изучены по отдельности, но мало рекомендаций о том, как объединить их в одном воспроизводимом процессе, дающем дедуплицированные, приоритизированные результаты, пригодные для конвейеров. Исследовательские вопросы: RQ1 — каким функциональным и нефункциональным

требованиям должна удовлетворять платформа тестирования безопасности веб-приложений? RQ2 — какая архитектура удовлетворяет этим требованиям и остаётся расширяемой для новых движков? RQ3 — как объективно измерить качество такой платформы?

Цель работы — разработать архитектуру платформы AutoSec-Web, объединяющей отдельные сканирующие движки в удобный, повторяемый процесс тестирования безопасности веб-приложений. **Задачи:** (i) сформулировать требования к платформе; (ii) спроектировать архитектуру, которая оркестрирует несколько движков в рамках одного плана сканирования; (iii) определить единую схему результатов с привязкой к перечню типовых слабостей (Common Weakness Enumeration, CWE) и категориям OWASP, а также дедупликацию и ранжирование находок по прозрачной оценке риска; (iv) предоставить результат конвейерам CI/CD через контроль качества (quality gate); (v) определить протокол оценки с измеримыми гипотезами.

Вклад работы: набор требований к такой платформе; архитектура и модель данных; модель приоритизации; протокол оценки с измеримыми гипотезами. В статье представлены проект и планируемая оценка; экспериментальные результаты намеренно не заявляются. Далее статья организована следующим образом. В разделе 2 описаны методология и предлагаемая платформа. В разделе 3 приведены результаты проектирования, ожидаемые результаты оценки, а также обсуждение ограничений и угроз валидности. В разделе 4 сформулированы выводы.

2. Материалы и методы

2.1. Дизайн исследования

Исследование следует парадигме науки о проектировании (design science): выявляется проблема, выводятся требования, проектируется артефакт (архитектура платформы) и определяется протокол оценки. Требования выведены из трёх источников: семейств контролей стандарта OWASP Application Security Verification Standard (ASVS) [8], этапов планирования и анализа NIST SP 800-115 [2] и ограничений, описанных в исследованиях из раздела 1.

2.2. Требования

В таблице 1 приведены основные требования. Функциональные (F) описывают, что делает платформа; нефункциональные (N) ограничивают, как она это делает.

Таблица 1 – Требования к платформе

Код	Требование	Обоснование
F1	Обход с аутентификацией: формы, сессии, одностраничные приложения	Покрытие ограничено обходом [3]
F2	Активные и пассивные проверки на внедрение, XSS, контроль доступа и неверную конфигурацию	Классы из [1]
F3	Анализ сторонних компонентов (SCA)	Уязвимые компоненты — распространённый класс риска [1]
F4	Нормализация всех находок в единую схему с привязкой к CWE/OWASP	Разнородные отчёты [6]
F5	Дедупликация и приоритизация по риску	Трудоёмкость разбора, ложные срабатывания [5]
F6	Интеграция с CI/CD и настраиваемый quality gate	Практика DevSecOps [6]
N1	Белый список разрешённых целей, ограничение частоты запросов, журнал аудита	Правовая и эксплуатационная безопасность [2]
N2	Подключаемые движки; воспроизводимые сканирования (версионируемые конфигурации)	Поведение движков меняется между версиями [5]

Примечание. SCA — анализ состава программного обеспечения (software composition analysis); XSS — межсайтовый скриптинг (cross-site scripting); CWE — перечень типовых слабостей (Common Weakness Enumeration); OWASP — Open Web Application Security

Project; CI/CD — непрерывная интеграция и поставка (continuous integration/continuous delivery); DevSecOps — практика встраивания проверок безопасности в процесс разработки и эксплуатации.

2.3. Предлагаемая архитектура

Платформа состоит из шести слоёв (рисунок 1). Пользователи и системы CI/CD отправляют запросы на сканирование через веб-интерфейс, программный интерфейс REST API (Representational State Transfer Application Programming Interface) или интерфейс командной строки (command-line interface, CLI). Шлюз API аутентифицирует вызывающего и проверяет, что цель входит в его разрешённую область. Оркестратор превращает запрос в план сканирования — ориентированное множество задач — и помещает задачи в очередь. Изолированные исполнители выполняют четыре вида задач: обход, активное динамическое тестирование, пассивную проверку и анализ компонентов. Активное тестирование делегируется зрелому движку с открытым кодом, например ZAP [9], через его API, что избавляет от повторной реализации атакующих нагрузок и позволяет заменить движок.

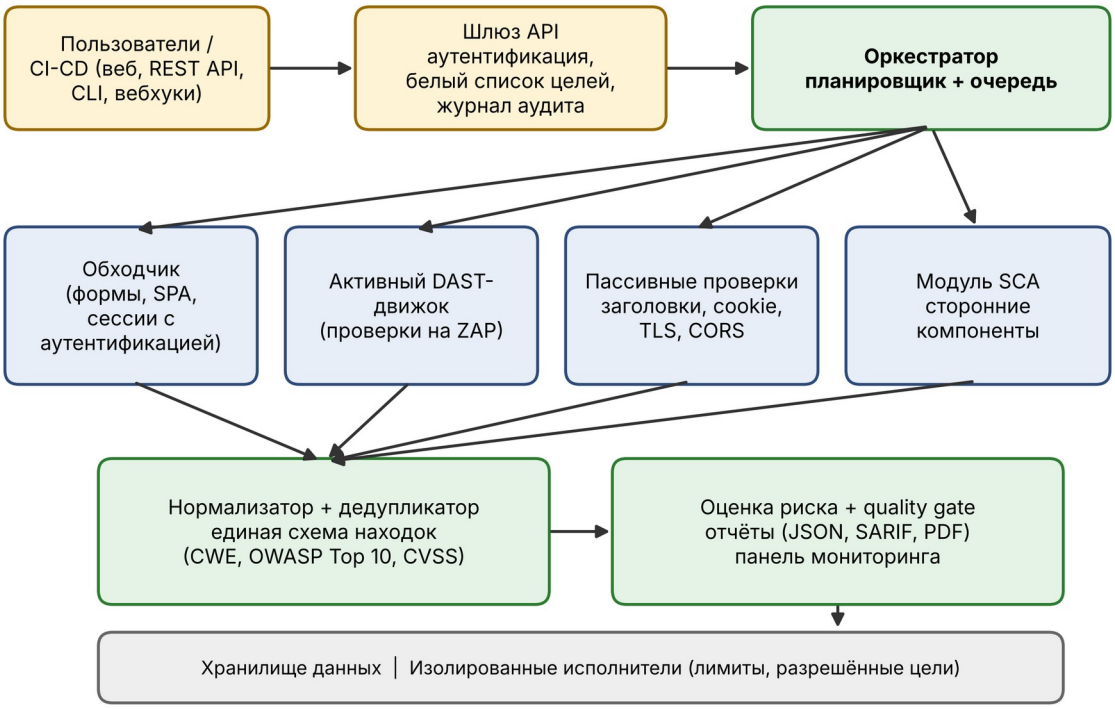

Рисунок 1 - Слоистая архитектура платформы AutoSec-Web.

Примечание. DAST — динамическое тестирование безопасности приложений (dynamic application security testing); SCA — анализ состава программного обеспечения; CWE — перечень типовых слабостей; CVSS — общая система оценки уязвимостей (Common Vulnerability Scoring System); CI/CD — непрерывная интеграция и поставка; REST API — программный интерфейс на основе архитектурного стиля REST; CLI — интерфейс командной строки; SPA — одностраничное приложение (single-page application); TLS — протокол защиты транспортного уровня (Transport Layer Security); CORS — совместное использование ресурсов между источниками (cross-origin resource sharing); JSON — формат обмена данными (JavaScript Object Notation); SARIF — формат обмена результатами анализа (Static Analysis Results Interchange Format); PDF — переносимый формат документов (Portable Document Format); ZAP — Zed Attack Proxy; OWASP — Open Web Application Security Project.

2.4. Единая схема находок

Каждая задача возвращает необработанные результаты, которые нормализатор преобразует в общую запись со следующими полями: движок и его версия, унифицированный адрес ресурса (URL) цели и метод протокола передачи гипертекста

(HTTP), уязвимый параметр, доказательство (фрагмент запроса и ответа), идентификатор CWE [10], категория OWASP, базовая оценка CVSS [11], уверенность движка и воспроизводимый идентификатор сканирования. Находки разных движков с одинаковыми CWE, конечной точкой и параметром объединяются в одну запись со списком сообщивших движков.

2.5. Модель приоритизации

Одной серьёзности недостаточно для разбора: высокая серьёзность при низкой уверенности отнимает время разработчика. Находки ранжируются прозрачной оценкой, которая объединяет базовую оценку CVSS, уверенность в находке и экспозицию затронутой конечной точки:

$$R = 10\left(w_s \cdot \frac{S}{10} + w_c \cdot C + w_e \cdot E\right), w_s + w_c + w_e = 1 \quad (1)$$

где S — базовая оценка CVSS (0–10); $C \in [0, 1]$ — уверенность, оцениваемая как доля независимых движков, сообщивших об одной и той же находке, с весом по измеренной точности движка; $E \in [0, 1]$ — экспозиция (1 для публичной конечной точки без аутентификации, меньше — для точки за аутентификацией); w_s, w_c, w_e — настраиваемые веса. Веса по умолчанию (0,5; 0,3; 0,2) — начальные значения, подлежащие калибровке при оценке, а не эмпирически установленные константы. Quality gate прерывает этап конвейера, если хотя бы одна находка имеет R выше заданного порога.

2.6. Протокол оценки

Платформа будет оцениваться на целях двух видов. Первая — OWASP Juice Shop, заведомо небезопасное приложение на Node.js, Express и Angular, охватывающее широкий круг рисков OWASP Top 10 [12]. Вторая — OWASP Benchmark, исполняемый набор тестов, в котором каждый случай является либо реальной уязвимостью, либо ложным срабатыванием для одного CWE, а инструменты оцениваются по долям истинных и ложных срабатываний [13]. Вычисляются следующие метрики:

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN}, Precision = \frac{TP}{TP + FP} \quad (2)$$

$$F_1 = \frac{2 \cdot Precision \cdot TPR}{Precision + TPR}, J = TPR - FPR \quad (3)$$

где TP, FP, TN и FN — числа истинно положительных, ложно положительных, истинно отрицательных и ложно отрицательных результатов; TPR (true positive rate) — доля истинных срабатываний; FPR (false positive rate) — доля ложных срабатываний; Precision — точность; F_1 — F-мера; J — индекс Юдена, используемый в оценочной карте OWASP Benchmark [13]. Дополнительно фиксируются число находок до и после дедупликации, время сканирования и накладные расходы конвейера. Все эксперименты будут проводиться только на локально развёрнутых заведомо уязвимых целях. Гипотезы и метрики планируемой оценки приведены в таблице 2.

Таблица 2 – Гипотезы и метрики планируемой оценки

Гипотеза	Метрика	База сравнения
H1: объединение движков повышает покрытие	TPR по категориям CWE	Один движок на основе ZAP
H2: нормализация и дедупликация снижают нагрузку на разбор	Число находок до/после	Простое объединение отчётов
H3: оценка с учётом уверенности снижает долю ложных срабатываний в первых k находках	Precision@k	Сортировка только по CVSS
H4: накладные расходы конвейера приемлемы	Время сканирования, задержка очереди	Конвейер без этого этапа

Примечание. TPR — доля истинных срабатываний; Precision@k — точность среди первых k находок в ранжированном списке; CVSS — общая система оценки уязвимостей; CWE — перечень типовых слабостей; ZAP — Zed Attack Proxy.

3. Результаты и обсуждение

На текущем этапе результатами являются артефакты проектирования и прогнозы, которые проверит планируемая оценка. Измеренные показатели производительности не приводятся и не должны выводиться из этого раздела.

3.1. Результаты проектирования

Проект отвечает на RQ1 и RQ2 следующим образом. Восемь требований таблицы 1 отображены на шесть слоёв архитектуры (рисунок 1), и каждое требование реализуется хотя бы одним компонентом: обход и проверки (F1–F3) — слоем исполнителей, нормализация и приоритизация (F4, F5) — сервисами нормализации и оценки, интеграция с CI/CD (F6) — API и quality gate, безопасность и расширяемость (N1, N2) — шлюзом, изолированными исполнителями и интерфейсом адаптера движка. Интерфейс адаптера требует от движка трёх операций — запустить задачу, опросить состояние и выгрузить результаты, — поэтому новый движок подключается написанием одного адаптера и одной таблицы соответствия. Ответ на RQ3 даёт протокол оценки (раздел 2.6): метрики (2)–(3) и четыре гипотезы таблицы 2.

3.2. Ожидаемые результаты

На основе литературы ожидается следующее. Поскольку качество обнаружения различается по категориям и версиям [5], объединение нескольких движков должно повысить TPR в категориях, где отдельный движок слаб (H1), возможно ценой большего числа сырых ложных срабатываний, которое призваны компенсировать дедупликация и взвешивание по уверенности (H2, H3). Влияние обхода на покрытие, описанное в [3], позволяет предположить, что обход с аутентификацией станет главным фактором различий на Juice Shop. Накладные расходы конвейера, как ожидается, определяются временем активного сканирования и поэтому регулируются профилями сканирования (короткий профиль для каждого коммита, полный — для ночных сборок) (H4). Это гипотезы, а не выводы.

3.3. Обсуждение и ограничения

Предлагаемая платформа не вводит новых методов атак; её ценность — в интеграции, нормализации и приоритизации, то есть в тех точках, где рассмотренные исследования показывают практические трудности [3], [5], [6]. Архитектура, в которой движки являются заменяемыми адаптерами, согласуется с наблюдением, что качество движков меняется между версиями [5], а повторный запуск протокола оценки в регрессионной задаче — естественный способ обнаруживать такие изменения.

Следует отметить ряд ограничений. Во-первых, цели оценки — заведомо уязвимые приложения и синтетический эталонный набор; результаты могут не переноситься на промышленные системы со сложной бизнес-логикой, которую автоматические сканеры не могут полностью проверить. Во-вторых, веса приоритизации — начальные значения и могут переобучиться на выбранных целях, если настраивать их на них же; поэтому нужны раздельные цели для калибровки и проверки. В-третьих, оценка уверенности по согласию движков предполагает независимость их ошибок, что не гарантировано, так как разные движки могут использовать схожие эвристики. В-четвёртых, детекторы на основе машинного обучения, такие как в [7], не входят в текущий проект, а их интеграция потребует проверки на других наборах данных. Наконец, активное тестирование несёт эксплуатационный риск; белый список целей, ограничение частоты запросов и журнал аудита (требование N1) обязательны, а платформу можно использовать только на системах, которые оператору разрешено тестировать [2].

4. Выводы

Представлена архитектура платформы автоматизированного тестирования безопасности веб-приложений AutoSec-Web, которая объединяет несколько сканирующих

движков, нормализует и дедуплицирует их результаты, ранжирует их прозрачной оценкой риска и обеспечивает quality gate в конвейерах CI/CD. В работе определены требования, архитектура, модель данных, формула приоритизации и протокол оценки с четырьмя проверяемыми гипотезами. Практическая ценность предложенного проекта — в основе для реализации прототипа и последующей эмпирической проверки. Дальнейшая работа: реализация прототипа, оценка на Juice Shop и OWASP Benchmark, калибровка весов на отдельных данных, расширение на тестирование API и оценка интеграции детекторов на основе машинного обучения.

Список литературы:

1. OWASP Top 10: The Ten Most Critical Web Application Security Risks [Электронный ресурс] / OWASP Foundation. – Режим доступа: https://owasp.org/www-project-top-ten/ (дата обращения: 16.10.2026).

2. Scarfone K., Souppaya M., Cody A., Orebaugh A. Technical Guide to Information Security Testing and Assessment: NIST Special Publication 800-115. – Gaithersburg, MD, USA: National Institute of Standards and Technology, 2008.

3. Doupé A., Cova M., Vigna G. Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners // Proceedings of the 7th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA). – Bonn, Germany, 8–9 July 2010.

4. Bau J., Bursztein E., Gupta D., Mitchell J. State of the Art: Automated Black-Box Web Application Vulnerability Testing // Proceedings of the 2010 IEEE Symposium on Security and Privacy. – Oakland, CA, USA, 16–19 May 2010. – P. 332–345.

5. Potti U.-S., Chen H.-T., Sun H.-M. Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an Example // Proceedings of the Universal Academic Cluster International April Conference. – Tokyo and Kyoto, Japan, 1–7 April 2024. – Режим доступа: https://arxiv.org/abs/2501.05907 (дата обращения: 16.10.2026).

6. Pimentel F.C.S., Fonseca L.C.C., Cortes O.A.C. Technologies and Tools for DevSecOps: A Systematic Literature Review // iSys – Journal of Information Systems. – 2025. – Vol. 18. – P. 18:1–18:24. – DOI: 10.5753/isys.2025.6207.

7. Alhamyani R., Alshammari M. Machine Learning-Driven Detection of Cross-Site Scripting Attacks // Information. – 2024. – Vol. 15. – Art. 420. – DOI: 10.3390/info15070420.

8. OWASP Application Security Verification Standard (ASVS) [Электронный ресурс] / OWASP Foundation. – Режим доступа: https://owasp.org/www-project-application-security-verification-standard/ (дата обращения: 16.10.2026).

9. ZAP (Zed Attack Proxy) [Электронный ресурс]. – Режим доступа: https://www.zaproxy.org/ (дата обращения: 16.10.2026).

10. Common Weakness Enumeration (CWE) [Электронный ресурс] / The MITRE Corporation. – Режим доступа: https://cwe.mitre.org/ (дата обращения: 16.10.2026).

11. Common Vulnerability Scoring System v3.1: Specification Document [Электронный ресурс] / FIRST.Org, Inc. – Режим доступа: https://www.first.org/cvss/v3.1/specification-document (дата обращения: 16.10.2026).

12. OWASP Juice Shop [Электронный ресурс] / OWASP Foundation. – Режим доступа: https://owasp.org/www-project-juice-shop/ (дата обращения: 16.10.2026).

13. OWASP Benchmark [Электронный ресурс] / OWASP Foundation. – Режим доступа: https://owasp.org/www-project-benchmark/ (дата обращения: 16.10.2026).

© Central Asian Scientific Journal, 2026