DOI: 10.58168/PPSPM2025_74-81
УДК 004.056.5

РАЗРАБОТКА АРХИТЕКТУРЫ ЗАЩИЩЕННОГО ВЕБ-ПРИЛОЖЕНИЯ ИНТЕРНЕТ-МАГАЗИНА

Н.С. Здоровцов, С.А. Евдокимова

ФГБОУ ВО «Воронежский государственный университет инженерных технологий»

В статье показана необходимость использования многоуровневой архитектуры защиты веб-приложения интернет-магазина, которая возникает из-за различных видов угроз безопасности. В работе предложена схема архитектуры разрабатываемого приложения, описаны технологические механизмы, обеспечивающие его безопасность.

Ключевые слова: информационная безопасность, веб-приложение, OWASP, SQL-инъекции, XSS-атаки, FastAPI.

ARCHITECTURE DEVELOPMENT OF A SECURE WEB APPLICATION OF AN ONLINE STORE

N.S. Zdorovtsov, S.A. Evdokimova

Voronezh State University of Engineering Technologies

The article shows the need to use a multi-level architecture for protecting the web application of an online store, which arises due to various types of security threats. The paper proposes an architecture scheme for the application being developed, describes the technological mechanisms that ensure its security.

Keywords: information security, web application, OWASP, SQL injection, XSS attacks, FastAPI.

Современные веб-приложения сталкиваются с растущим числом угроз, связанных как с устаревшими архитектурными решениями, так и с новыми векторами атак. Обеспечение информационной безопасности веб-приложений

невозможно без понимания актуальных угроз и соответствующих нормативных требований.

Одним из наиболее авторитетных источников, на которые ориентируются разработчики и специалисты по кибербезопасности, является OWASP Top 10 – документ, формирующий представление о наиболее критичных уязвимостях, которые на регулярной основе фиксируются в веб-среде [1].

В последних редакциях OWASP Top 10 особое внимание уделяется проблемам, связанным с управлением идентификацией и доступом, ошибками в конфигурации безопасности, а также уязвимостям, возникающим из-за небезопасного проектирования систем. Акценты смещены с «технических багов» на недостатки архитектурных решений, что подчеркивает рост интереса к проактивной защите – к разработке приложений, устойчивых к угрозам изначально, а не после обнаружения первых эксплойтов. Например, категорией A04:2021 в OWASP выделяются «Небезопасные конструкции», где подчёркивается важность внедрения принципов защищённой разработки на этапе проектирования системы.

Не менее важным аспектом, дополняющим картину рисков, является нормативно-правовая база, к которой относятся ГОСТ Р 59407 и международный стандарт ISO/IEC 27001 [2, 3]. Большое внимание в ГОСТ Р 57580 уделяется защите персональных данных, что особенно актуально для российского законодательства.

Архитектура проектируемого приложения интернет-магазина представляет собой многослойное решение, в котором уязвимости распределены между клиентским интерфейсом, серверной логикой и системой хранения данных. Эффективная защита не может строиться вокруг одного компонента, она требует синхронной работы всех уровней – от браузера пользователя до структуры SQL-запросов [4-6].

Клиентская часть выполняет критически важную роль в цепочке безопасности, так как уязвимости на этом уровне напрямую влияют на успех атак, ориентированных на пользователя – от XSS до фишинговых подмен. Одной из ключевых проблем является отсутствие обработки пользовательского ввода в компонентах формы, особенно в модулях обратной связи, адресной информации и встроенных редакторах отзыва. Если вводимые данные обрабатываются без фильтрации, то злоумышленник может внедрить исполняемый JavaScript-код или SQL-инъекции [7-9].

На уровне клиентской логики широко распространена практика хранения токенов аутентификации в LocalStorage. Однако в условиях интернет-магазина, где аккаунт пользователя может содержать чувствительную информацию о заказах, платежах и адресах доставки, такое решение создаёт прямой вектор атаки через XSS. Более надёжный подход – это хранение токена в HttpOnly-cookie, недоступной из скриптов, с привязкой к параметрам браузерной сессии. При этом важно не только выбрать способ хранения, но и настроить поведение при окончании сессии авторизации. Например, автоматическая очистка кэша, принудительный выход и контроль активности сессии по IP-адресу и User-Agent.

Серверная логика, обеспечивающая авторизацию, валидацию и взаимодействие с БД, является ядром защищённого интернет-магазина. Ошибки в этой зоне часто связаны с тем, что контроль доступа реализуется формально. Например, по наличию JWT, без дополнительной проверки владельца ресурса. Такая модель допускает горизонтальные атаки: пользователь может получить доступ к заказам другого клиента, если изменит ID в маршруте. Особенно уязвимы API-эндпоинты, где структура запроса выглядит предсказуемо. Более надёжным подходом будет связывание каждого действия с идентификатором текущего пользователя на уровне бизнес-логики, независимо от предоставленных параметров. Важным дополнением становится журналирование, так как регистрация всех попыток обращения к критическим данным позволит не только обнаружить попытку атаки, но и воспроизвести её сценарий при последующем анализе.

Хеширование пользовательских паролей является обязательным элементом при работе с регистрацией и аутентификацией, при этом качество операции определяется выбранным алгоритмом. Использование bcrypt с адаптивной сложностью позволяет замедлить перебор при компрометации базы. Важно также предусмотреть возможность миграции хешей.

С точки зрения валидации, серверная часть интернет-магазина должна отказаться от доверия к входящим данным даже при наличии клиентской проверки. Электронная почта, телефон, адрес и особенно поля с произвольным вводом (комментарии к заказу, обратная связь) подлежат фильтрации, нормализации и проверке на соответствие шаблонам. Невнимание к деталям, например, пропуск символов новой строки в текстовом поле, позволит злоумышленнику внедрить служебные конструкции, которые изменят поведение парсера. Особенно критичны поля, содержащие JSON или HTML-

фрагменты. Их передача в непреобразованном виде может нарушить структуру ответа и привести к утечке данных.

Слой базы данных несёт основную нагрузку при обработке заказов, авторизации, хранения личных данных и истории транзакций. Использование ORM (Object-Relational Mapping, объектно-реляционное отображение) существенно снижает риск SQL-инъекций, но не устраняет его полностью. Если система допускает прямое выполнение запросов на основе пользовательского ввода, без предварительного фильтра, то даже один неправильно построенный фрагмент может привести к утечке или модификации данных. Подготовленные выражения (prepared statements) решают проблему на уровне синтаксиса, но только при условии, что переменные подставляются в строго типизированные позиции. Типовой ошибкой является подстановка имени таблицы или поля из GET-параметра, где экранирование не применяется. С точки зрения хранения, данные пользователя должны быть изолированы, архитектура БД должна предусматривать ограничения по области видимости, а система прав доступа – разграничение между административными и пользовательскими операциями.

Таким образом, при построении защищённого приложения интернет-магазина безопасность должна быть реализована не в виде отдельных механизмов, а в виде сети взаимозависимых решений, охватывающих клиентскую часть, серверную часть и СУБД. Надёжность системы будет обеспечена тем, что каждый слой проверяет, что предыдущий сделал свою работу корректно.

При проектировании веб-приложения интернет-магазина разработана схема многоуровневой защиты, представленная на рисунке 1.

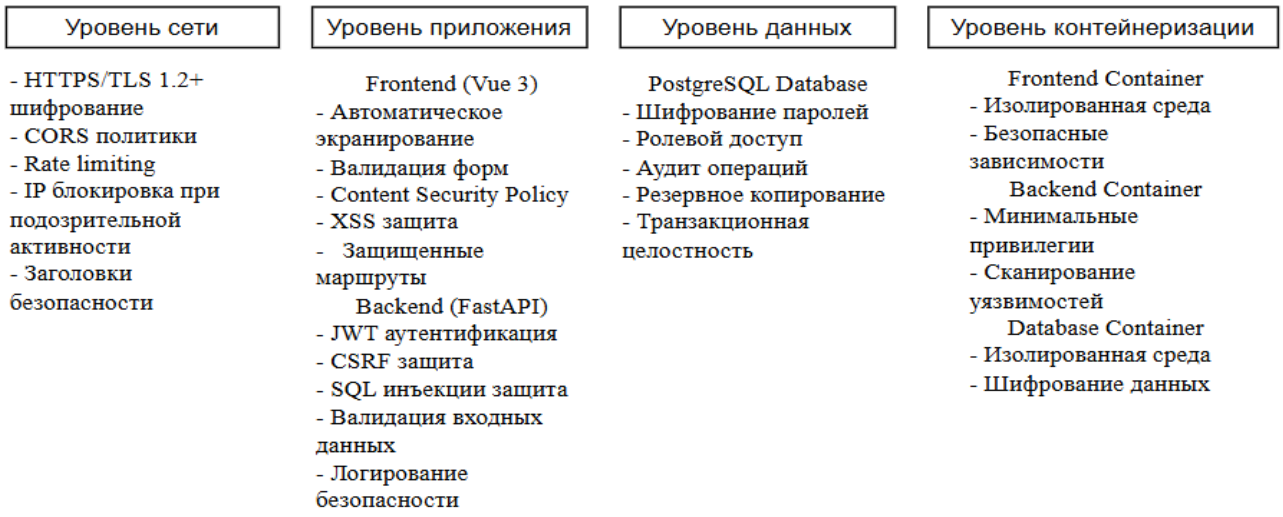

Рисунок 1 – Схема многоуровневой системы защиты веб-приложения

Архитектура безопасности разрабатываемого приложения включает несколько взаимодополняющих уровней защиты. На уровне сетевого взаимодействия реализуется защита от DDoS-атак и несанкционированного доступа через настройку файрволов и ограничение скорости запросов. Транспортный уровень обеспечивает шифрование данных с использованием TLS-протоколов, предотвращая перехват конфиденциальной информации в процессе передачи.

На уровне приложения реализуется комплексная система аутентификации и авторизации, включающая многофакторную аутентификацию и детализированное управление правами доступа. Механизмы защиты от инъекционных атак и межсайтового скриптинга интегрированы непосредственно в бизнес-логику приложения, обеспечивая валидацию и санитизацию всех входных данных.

Вторая линия защиты расположена непосредственно в веб-приложении и включает механизмы валидации входных данных, защиту от инъекций (например, использование параметризированных запросов) и управление сессиями (в том числе защиту от угонов с помощью Secure и HttpOnly флагов для cookie). Этот уровень предотвращает проникновение на уровне бизнес-логики и пользовательского взаимодействия.

Третья линия – это защита операционной системы и базы данных, где реализуется изоляция процессов, контроль прав доступа и шифрование данных на уровне хранения.

Принцип минимальных привилегий реализован во всей архитектуре. Каждому компоненту выделяются только необходимые права для выполнения своих функций, что снижает потенциальный ущерб в случае взлома. Система ролевого доступа (RBAC) обеспечивает детальное управление правами пользователей, что позволяет гибко настраивать доступ и операции в зависимости от роли и обязанностей сотрудника.

Архитектура безопасности определяет общие принципы защиты, но их практическая реализация требует тщательного выбора технологического стека. Каждая технология вносит свой вклад в общую безопасность системы – от встроенных механизмов защиты фреймворков до возможностей изоляции контейнеров [4]. FastAPI, например, предоставляет автоматическую валидацию данных через Pydantic, что исключает целый класс уязвимостей, связанных с

некорректной обработкой входных параметров. Vue.js по умолчанию экранирует данные в шаблонах, предотвращая XSS-атаки без дополнительных усилий разработчика. Docker обеспечивает изоляцию процессов на уровне операционной системы, создавая дополнительный барьер для потенциальных злоумышленников. Выбор технологий должен основываться не только на их функциональных возможностях, но и на характеристиках безопасности, способности интегрироваться с другими компонентами системы и долгосрочной поддержке сообществом [7-9]. Выбор технологического стека для защищенного веб-приложения требует тщательного анализа функциональных возможностей и характеристик безопасности каждой технологии. Современный рынок предлагает множество решений, но не все из них обеспечивают необходимый уровень защиты и соответствуют принципам безопасной разработки.

Выбор FastAPI в качестве основного фреймворка для разработки API является хорошим выбором для проектов, где безопасность и производительность являются приоритетами [7]. В отличие от традиционных решений, как Django или Flask, FastAPI предлагает встроенные механизмы защиты на архитектурном уровне, минимизируя человеческий фактор. Например, автоматическая валидация данных через Pydantic исключает риски SQL-инъекций и XSS, которые в Flask требуют ручного контроля, а в Django хоть и обрабатываются ORM, но не столь гибки для сложных API-сценариев.

Кроме того, поддержка асинхронности делает FastAPI устойчивым к атакам типа DDoS, поскольку фреймворк эффективно справляется с высокой нагрузкой без блокировки запросов — в отличие от синхронных Django и Flask, где подобные угрозы требуют дополнительных инструментов. Интеграция с OpenAPI и Swagger UI также способствует безопасности: автоматическая документация позволяет сразу выявлять потенциально опасные эндпоинты, а встроенная поддержка OAuth2 и JWT избавляет от необходимости использовать ненадежные сторонние библиотеки.

Таким образом, разработанная многоуровневая архитектура веб-приложения демонстрирует подход к защите, в котором каждый уровень включает необходимые механизмы безопасности.

Список литературы

1. OWASP Foundation. OWASP Top 10:2021. – URL: https://owasp.org/www-project-top-ten/(дата обращения: 12.10.2025).

2. ГОСТ Р 59407-2021. Информационные технологии. Методы и средства обеспечения безопасности. Базовая архитектура защиты персональных данных. – М.: Стандартинформ, 2021. – 32 с.

3. ГОСТ ISO/IEC 29100-2021. Информационные технологии. Методы и средства обеспечения безопасности. Основы защиты персональных данных. – М.: Стандартинформ, 2021. – 28 с.

4. Галатенко, В.А. Основы информационной безопасности. – М.: Интернет-университет информационных технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. – 266 с.

5. Выбор критерия оптимальности при принятии управленческих решений в сложных технических системах / А.В. Скрыпников [и др.] // Моделирование систем и процессов. – 2024. – Т. 17, № 1. – С. 120-128.

6. Мочалов, В.П. Алгоритм динамического распределения и балансировки нагрузки в распределенных облачных вычислениях / В.П. Мочалов, Н.Ю. Братченко, Д.В. Гостева // Моделирование систем и процессов. – 2024. – Т. 17, № 1. – С. 92-102.

7. Stelea, G.A. When Cybersecurity Meets Accessibility: A Holistic Development Architecture for Inclusive Cyber-Secure Web Applications and Websites / G.A. Stelea, L. Sangeorzan N. Enache-David // Future Internet. – 2025. – Vol. 17. – S. 67.

8. Тун, Ю. Разработка алгоритма повышения эффективности протокола маршрутизации C-LEACH / Ю. Тун, Т.П. Новикова, С.А. Евдокимова // Моделирование систем и процессов. – 2022. – Т. 15, № 2. – С. 93-99.

9. Prasetyawan, D. Pengembangan Sistem Seleksi Proposal Penelitian Berbasis Web Service Menggunakan REST API / D. Prasetyawan, P. D. Rahmanto // JTIM. – 2024. – Vol. 6, no. 3. – Pp. 283-295.

References

1. The OWASP Foundation. OWASP Top 10:2021. – URL: https://owasp.org/www-project-top-ten/(date of publication: 12.10.2025).

2. GOST R 59407-2021. Information technology. Methods and means of ensuring security. The basic architecture of personal data protection. – Moscow: Standartinform, 2021. – 32 p.

3. GOST ISO/IEC 29100-2021. Information technology. Methods and means of ensuring security. Fundamentals of personal data protection. – Moscow: Standartinform, 2021. – 28 p.

4. Galatenko, V.A. Fundamentals of information Security. – Moscow: Internet University of Information Technologies (INTUIT), IP Media, 2024. – 266 p.

5. Choosing the optimality criterion when making managerial decisions in complex technical systems / A.V. Skrypnikov [et al.] // Modeling of systems and processes. – 2024. – Vol. 17, No. 1. – Pp. 120-128.

6. Mochalov, V.P. Algorithm of dynamic load distribution and balancing in distributed cloud computing / V.P. Mochalov, N.Y. Bratchenko, D.V. Gosteva // Modeling of systems and processes. – 2024. – Vol. 17, No. 1. – Pp. 92-102.

7. Stele, G.A. When cybersecurity is combined with accessibility: a holistic development architecture for inclusive cybersecurity web applications and websites / G.A. Stele, L. Sangeorzan, N. Enash-David // The Internet of the Future. – 2025. – Volume 17. – S. 67.

8. Tun, Yu. Development of an algorithm to improve the efficiency of the C-lich routing protocol / Yu. Tun, T.P. Novikova, S.A. Evdokimova // Modeling of systems and processes. – 2022. – Vol. 15, No. 2. – Pp. 93-99.

9. Prasetyawan, D. Pengembangan, Proposal of the Penelitian Berbasis Menggunakan REST API web service selection system / D. Prasetyawan, P. D. Rahmanto // JTIM. - 2024. – Vol. 6, No. 3. – Pp. 283-295.