Воронцов Глеб Игоревич, бакалавр
Уфимский университет науки и технологий
Vorontsov Gleb Igorevich
Ufa University of Science and Technology

КОМПЛЕКСНЫЙ ПОДХОД К ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ
A COMPREHENSIVE APPROACH TO SECURING MODERN WEB APPLICATIONS

Аннотация. Систематизированы основные меры защиты современных веб-приложений. Рассмотрены контроль доступа, управление сессиями, предотвращение инъекций, защита браузерного контура, безопасность зависимостей и эксплуатационный мониторинг. Предложена многоуровневая модель, связывающая архитектурные решения, разработку, проверку и сопровождение приложения.

Abstract. Key security measures for modern web applications are systematized. Access control, session management, injection prevention, browser-side protection, dependency security, and operational monitoring are considered. A layered model connecting architecture, development, verification, and application maintenance is proposed.

Ключевые слова: Веб-приложение, информационная безопасность, контроль доступа, XSS, аутентификация, безопасная разработка.

Keywords: Web application, information security, access control, XSS, authentication, secure development.

Введение

Современное веб-приложение включает браузерный интерфейс, программный API, серверную логику, хранилища данных и множество сторонних компонентов. Каждая граница между этими элементами становится поверхностью атаки. Уязвимость может возникнуть не только из-за ошибки в коде, но и вследствие неверной конфигурации, избыточных полномочий, компрометации зависимости либо отсутствия реакции на подозрительные события. OWASP Top 10:2025 относит к ключевым рискам нарушения контроля доступа, ошибки конфигурации, сбои цепочки поставок, инъекции и недостатки проектирования [1].

Цель статьи – сформировать целостный подход к защите веб-приложения на протяжении его жизненного цикла. Перечень популярных атак используется как основа модели угроз, но не заменяет проверяемых требований. Для практической реализации применим OWASP ASVS 5.0.0, который задает уровни и критерии проверки технических средств защиты [2]. Безопасность рассматривается как сочетание доверенной архитектуры, корректной обработки данных, защищенной аутентификации, контроля компонентов и наблюдаемой эксплуатации.

Модель доверия, аутентификация и контроль доступа

Базовый принцип состоит в том, что клиентский интерфейс не является доверенной стороной. Скрытая кнопка, проверка маршрута в браузере или значение роли в локальном состоянии не могут служить механизмом авторизации. Сервер обязан проверять полномочия при каждом обращении к объекту и каждой операции, применяя запрет по умолчанию. Особое внимание требуется к идентификаторам записей: пользователь не должен получать чужой объект простой заменой параметра запроса. Требования контроля доступа должны быть централизованы и покрыты негативными тестами [1, 2].

После аутентификации необходимо защищать жизненный цикл сессии. Идентификатор создается криптографически стойким способом, изменяется после входа или повышения привилегий и прекращает действие при выходе и превышении установленного времени. Для cookie применяются атрибуты Secure, HttpOnly и подходящее значение SameSite; область Domain и Path ограничивается минимально необходимой. Долговременные секреты и привилегированные токены не должны быть доступны произвольному клиентскому сценарию. ASVS выделяет управление сессиями в самостоятельную группу проверок [2].

Федеративная авторизация также требует актуальной модели угроз. RFC 9700 рекомендует использовать поток authorization code вместо implicit grant, точно проверять URI перенаправления и применять PKCE для защиты кода [6]. Область действия токена должна быть минимальной, а аудитория – однозначно определенной. Для публичных клиентов повторное использование refresh token обнаруживается посредством его ротации либо криптографической привязки к отправителю. Использование готовой библиотеки снижает число ошибок только при проверенной конфигурации и своевременном обновлении.

Защита обработки данных и браузерного контура

Все внешние данные рассматриваются как недоверенные: параметры URL, поля JSON, заголовки, файлы и сообщения интеграций. Синтаксическая и семантическая валидация ограничивает допустимую форму, но сама по себе не предотвращает SQL-инъекцию или XSS. Запросы к хранилищу должны быть параметризованы, а вывод – кодироваться с учетом контекста HTML, атрибута, URL, CSS или JavaScript. OWASP подчеркивает, что универсальный перехватчик кодирования не способен корректно защитить все контексты и может вызвать двойное преобразование [3].

Для предотвращения XSS предпочтительны безопасные DOM-операции, автоматически экранируемые шаблоны и специализированная очистка допустимого HTML. Возможность вставки разметки или динамического кода должна быть локализована и проверена. Content Security Policy ограничивает источники загружаемых и исполняемых ресурсов, запрещает нежелательные встроенные сценарии и формирует отчеты о нарушениях [4]. CSP является дополнительным уровнем защиты, а не заменой кодирования и безопасных приемников; политику целесообразно сначала наблюдать через режим Report-Only, затем переводить в режим применения.

CSRF использует автоматическую отправку браузером аутентификационных cookie при запросе с постороннего ресурса. Для операций изменения состояния применяются непредсказуемые CSRF-токены, проверка источника запроса и ограничения SameSite [5]. Токен можно передавать в скрытом поле либо специальном заголовке, но он не должен попадать в URL и журналы. Настройка CORS не является универсальной защитой от CSRF, а наличие XSS способно обойти многие браузерные меры, поэтому механизмы должны использоваться совместно.

Конфигурация, компоненты и эксплуатационный контроль

Безопасная программа может быть скомпрометирована небезопасным окружением. В производственной конфигурации отключаются диагностические страницы, стандартные учетные данные и ненужные сетевые точки; секреты хранятся вне репозитория и регулярно меняются. Шифрование транспортного соединения применяется на всем пути до доверенного серверного компонента, а заголовки безопасности задаются централизованно. Исключения возвращают клиенту ограниченное сообщение без трассировки и внутренней структуры, но сохраняют диагностический контекст в защищенном журнале [1, 2].

Цепочка поставок требует учета прямых и транзитивных зависимостей, фиксации версий и контроля источника пакетов. OWASP SCVS предлагает рассматривать состав программного продукта, проверку компонентов и процессы снижения риска как отдельную область обеспечения доверия [7]. В сборочный конвейер включаются анализ известных уязвимостей, проверка целостности, минимизация прав учетной записи сборки и запрет публикации при критическом нарушении. Автоматическое обновление полезно только вместе с тестами совместимости и возможностью проследить происхождение артефакта.

Журналирование должно поддерживать расследование, не создавая нового канала утечки. Регистрируются успешные и отклоненные входы, изменение прав, административные действия, срабатывание проверок и аномальное обращение к ресурсам. Пароли, полные токены, ключи и избыточные персональные данные в журнал не помещаются. Корреляционный идентификатор связывает события браузерного запроса, API и фоновых операций. Сигнал становится защитной мерой только при наличии правил оповещения, ответственного исполнителя и процедуры реагирования.

В жизненном цикле разработки требования ASVS преобразуются в критерии приемки. На этапе проектирования выполняются моделирование угроз и анализ границ доверия; при реализации – экспертная проверка и статический анализ; перед выпуском – динамические и негативные тесты, контроль зависимостей и конфигурации. Проверки повторяются после изменений, а найденная уязвимость приводит не только к исправлению конкретной строки, но и к тесту, предотвращающему регрессию. Так формируется воспроизводимый процесс, а не разовая проверка перед публикацией.

Заключение

Безопасность современного веб-приложения не обеспечивается одним фильтром, фреймворком или межсетевым экраном. Контроль доступа выполняется на сервере для каждого действия; сессии и токены имеют ограниченный жизненный цикл; недоверенные данные обрабатываются параметризованно и кодируются по контексту. Браузерные механизмы CSP, SameSite и CSRF-токены создают дополнительные уровни, но не отменяют устранение первопричин. Конфигурация, зависимости и наблюдаемость входят в ту же модель защиты, что и исходный код.

Наиболее устойчивый подход связывает модель угроз с конкретными проверяемыми требованиями и сопровождает их на всех этапах разработки. OWASP Top 10 помогает определить приоритетные классы риска, ASVS задает основу контроля, а отраслевые стандарты уточняют отдельные протоколы и браузерные механизмы. Регулярная проверка, минимальные полномочия, безопасные значения по умолчанию и готовность к реагированию уменьшают как вероятность успешной атаки, так и масштаб ее последствий.

Список литературы:

1. OWASP Foundation. OWASP Top 10:2025. URL: https://owasp.org/Top10/2025/ (дата обращения: 11.08.2026).

2. OWASP Foundation. Application Security Verification Standard 5.0.0. 2025. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата обращения: 11.08.2026).

3. OWASP Foundation. Cross Site Scripting Prevention Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (дата обращения: 11.08.2026).

4. W3C. Content Security Policy Level 3. Working Draft, 05.05.2026. URL: https://www.w3.org/TR/CSP3/ (дата обращения: 11.08.2026).

5. OWASP Foundation. Cross-Site Request Forgery Prevention Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html (дата обращения: 11.08.2026).

6. Lodderstedt T., Bradley J., Labunets A., Fett D. Best Current Practice for OAuth 2.0 Security. RFC 9700. IETF, 2025. DOI: 10.17487/RFC9700.

7. OWASP Foundation. Software Component Verification Standard. URL: https://owasp.org/www-project-software-component-verification-standard/ (дата обращения: 11.08.2026)