УДК 004.056

ОШИБКИ В ЛОГИКЕ ПАРАМЕТРОВ В ВЕБ-ПРИЛОЖЕНИЯХ: ВЫЯВЛЕНИЕ И ПРЕДОТВРАЩЕНИЕ

Горбачев В.Д.[1], Янов Д.А.[1], Платонов А.А.[1]
Научный руководитель – кандидат технических наук, доцент Платонов А.А.[1]
[1]Военно-космическая академия имени А.Ф.Можайского
vka@mil.ru

Введение

В условиях развития веб-приложений и интерфейсов программирования приложений (API) в 2026 году ошибки в бизнес-логике остаются опасной категорией уязвимостей. Согласно открытому проекту безопасности веб-приложений (OWASP Business Logic Abuse Top 10 за 2025 год) и отчётам HackerOne, их количество растёт, составляя долю критических находок в программах поиска уязвимостей (bug bounty) [1]. Автоматизированные инструменты и межсетевые экраны веб-приложений (WAF) бессильны, так как ошибки выглядят как нормальное поведение [2]. Логические ошибки в параметрах возникают из-за недочётов в обработке ввода и позволяют злоумышленнику получить преимущества в виде бесплатного доступа, финансовых хищений или обхода ограничений [3]. Эти ошибки являются следствием «непреднамеренного поведения» (unintended behavior), которое проявляется через стандартный интерфейс, и сильно отличается от инъекций структурированного языка запросов (SQL) или межсайтового скриптинга (XSS). Причинами появления таких ошибок являются: слабый дизайн, фокус на счастливом пути (happy path), отсутствие паритета фронтенда и бэкенда, недостаточное тестирование [5]. В работе предлагается подход на основе тестирования методом белого ящика (white-box) и безопасной разработки для выявления таких уязвимостей.

Основная часть

Цель исследования – рассмотреть проблему ошибок в логике параметров как одну из наиболее актуальных угроз веб-безопасности в 2026 году и показать методы их выявления и предотвращения.

Эффективность противодействия ошибкам в логике параметров повышается при анализе бизнес-логики через проверку кода, локальное тестирование и создание доказательства концепции (PoC – proof of concept) [3]. В отличие от часто применяемых методов нахождения ошибок, таких как сканеры OWASP ZAP или Burp Suite, которые полагаются на предопределенные шаблоны уязвимостей и часто пропускают ошибки бизнес-логики из-за отсутствия понимания контекста и логики конкретного приложения (например, они могут не выявить уязвимость, где ввод отрицательного количества товаров в корзине приводит к возврату средств вместо оплаты, или несоответствие валидации между фронтендом и бэкендом, позволяющее обойти проверки через модификацию запросов), предлагаемый подход опирается на семантический анализ всего запроса, включая его параметры. Это обеспечивает глубокое изучение и выявление трех основных типов ошибок в логике параметров — несоответствия логики проверки (validation logic disparity), неожиданного ввода (unexpected input) и проблем нулевой безопасности (null safety) [4].

1. *Несоответствие логики проверки* (validation logic disparity) — различия в валидации на фронтенде и бэкенде или отсутствие повторной проверки на сервере. Такие ошибки возникают, когда фронтенд и бэкенд применяют разные правила валидации или сервер доверяет данным от клиента без повторной проверки.

Выявление ошибки: для этого необходимо искать функции, которые принимают пользовательский ввод, используют динамическую проверку на основе серверных данных и не выполняют полную повторную валидацию на бэкенде.

Пример эксплуатации: в одном интернет-магазине товар «Скоро в продаже» (невыпущенный iPhone) можно было добавить в корзину путём изменения productId в запросе. Фронтенд проверял наличие только для отображения кнопки, а бэкенд не делал повторную проверку при оформлении заказа. В результате товар удалось купить до официального релиза.

2. *Неожиданный ввод* (unexpected input) — приложение не обрабатывает данные нестандартного типа (отрицательные значения, строки вместо чисел). Логика функций часто строится вокруг ожидаемого типа данных. При слабой типизации (JavaScript, PHP) неожиданные значения приводят к непредсказуемому поведению из-за неявного приведения типов.

Выявление ошибки: для этого нужно искать функции с прямым пользовательским вводом, свободными переменными (var, let) и слабыми сравнениями.

Примеры эксплуатации:

− отрицательное количество товаров в корзине снижает итоговую сумму до нуля или отрицательного значения;

− отрицательные чаевые в сервисах доставки приводят к возврату денег пользователю;

− отрицательный перевод в банковских приложениях позволяет снимать средства со счёта получателя.

3. *Нулевая безопасность* (null safety) — некорректная обработка null значений и необязательных параметров. Ошибки возникают, когда необязательные параметры принимают значения null или отсутствуют, а функция не обрабатывает эту ситуацию корректно.

Выявление ошибки: для этого необходимо анализировать схемы валидации (например, с использованием библиотеки Yup) на наличие необязательных полей и проверять, как конечные точки веб-сервиса обрабатывают значения null.

Для предотвращения ошибок в логике параметров необходимо [6]:

− выполнять полную повторную валидацию на сервере;

− использовать строгую типизацию и явные проверки типов;

− настраивать схемы валидации с обязательными полями там, где это необходимо;

− проводить ручной анализ кода (code review) и тестирование «что, если…» сценариев;

− применять моделирование угроз (threat modeling) и тестирование на основе свойств (property-based testing).

Предлагаемый подход позволяет формировать рекомендации по безопасной разработке с учетом реальных примеров эксплуатации.

Выводы

Исследование подтвердило, что ошибки в логике параметров остаются серьёзной угрозой в 2026 году, поскольку автоматизация не способна полностью заменить экспертный анализ бизнес-логики [1]. Предложенный подход — проверка кода, локальное тестирование и создание доказательства концепции (PoC) — позволяет эффективно выявлять и устранять такие уязвимости [4].

Вклад работы заключается в анализе типов ошибок с примерами и рекомендациями по предотвращению, что значительно снижает риски финансовых потерь и несанкционированного доступа [5]. Такой подход обеспечивает безопасность веб-приложений и исключает субъективность в выявлении уязвимостей.

Литература

1. OWASP Business Logic Abuse Top 10. – 2025. – URL: https://owasp.org/www-project-top-10-for-business-logic-abuse/ (дата обращения: 19.02.2026).
2. HackerOne: 8th Annual Hacker-Powered Security Report. – 2025. – URL: https://www.hackerone.com/reports (дата обращения: 21.02.2026).
3. PortSwigger Web Security Academy: Business Logic Vulnerabilities. – URL: https://portswigger.net/web-security/logic-flaws (дата обращения: 20.02.2026).
4. PortSwigger Web Security Academy: Examples of business logic vulnerabilities. – URL: https://portswigger.net/web-security/logic-flaws/examples (дата обращения: 24.02.2026).
5. HackerOne Blog: How a Business Logic Vulnerability Led to Unlimited Discount Redemption. – URL: https://www.hackerone.com/blog/how-business-logic-vulnerability-led-unlimited-discount-redemption (дата обращения: 18.02.2026).
6. Bright Security: Business Logic Vulnerabilities: Busting the Automation Myth. – URL: https://brightsec.com/blog/business-logic-vulnerabilities-busting-the-automation-myth/ (дата обращения: 22.02.2026).