УДК 04.056

МЕТОДЫ ЗАЩИТЫ ВЕБ-ПРИЛОЖЕНИЙ ОТ SQL-ИНЪЕКЦИЙ НА ЭТАПАХ РАЗРАБОТКИ И ЭКСПЛУАТАЦИИ С ПРИМЕНЕНИЕМ WEB APPLICATION FIREWALL

Филиппов П.И., студент гр. ИБс-211, V курс,
Научный руководитель: Прокопенко Е.В., к.т.н., заведующая кафедры ИБ
Кузбасский государственный технический университет
имени Т.Ф. Горбачева, г. Кемерово

Аннотация

В статье рассматривается проблема SQL-инъекций как одной из наиболее опасных угроз для веб-приложений. Анализируются современные методы защиты, используемые на этапе разработки, такие как параметризованные запросы, Object-Relational Mapping фреймворки, экранирование, валидация, на этапе эксплуатации – Web Application Firewall. Проводится сравнение эффективности данных методов, определяются их преимущества и недостатки. Обосновывается необходимость комплексного подхода, сочетающего меры на этапе разработки, а также средства активного мониторинга и блокирования атак. Сформулированы практические рекомендации для разработчиков по противодействию угрозе.

Введение

В условиях глобального внедрения цифровых технологий и решений для экономической и социальной сферы веб-приложения стали частью деятельности многих коммерческих предприятий и государственных учреждений. Однако публичная доступность и сложная внутренняя логика делает их привлекательным объектом для кибератак, что определяет нахождение веб-приложений в зоне повышенного риска.

SQL-инъекции остаются одной из самых распространенных и опасных угроз. Согласно актуальным данным OWASP Top 10 2025, инъекции стабильно входят в число наиболее критических уязвимостей [1].

Целью настоящей статьи является анализ методов защиты от SQL-инъекций на различных этапах жизненного цикла веб-приложения от разработки до эксплуатации и обоснование комплексного подхода для предотвращения реализации угрозы.

1. Механизм реализации SQL-инъекций

Для определения характера угрозы и принципов построения защиты необходимо рассмотреть механизм реализации SQL-инъекций. Как отмечается в работе Шелягова С.А. и Кудрявченко И.В., SQL-инъекция представляет собой разновидность атаки для несанкционированного доступа к базе данных, при

которой злоумышленник производит вставку вредоносного кода в запрос, передаваемый на сервер СУБД для дальнейшего синтаксического анализа и выполнения [2].

Уязвимость возникает, когда веб-приложение динамически формирует SQL-запросы путем конкатенации строк, используя необработанные пользовательские данные. В штатном режиме работы предполагается, что в поля ввода будут переданы только соответствующие значения. Однако отсутствие проверки и фильтрации входных данных позволяет злоумышленнику модифицировать структуру исходного SQL-запроса.

Классическим примером является атака, направленная на обход механизма аутентификации. Если приложение строит запрос вида SELECT * FROM users WHERE login = '[логин]' AND password = '[пароль]', злоумышленник может ввести в поле логина специально сформированное значение, содержащее различные операторы и символы комментария. В результате логика запроса меняется, и условие проверки пароля игнорируется, что позволяет получить несанкционированный доступ к системе.

2. Методы защиты на этапе разработки

2.1 Параметризованные запросы

Параметризованные запросы признаны фундаментальным методом предотвращения SQL-инъекций, поскольку они позволяют отделять данные от кода запроса, что исключает возможность интерпретации пользовательского ввода как части исполняемой SQL-команды [3].

Процесс работы параметризованного запроса состоит из двух этапов.

Компиляция. Приложение отправляет в СУБД шаблон запроса, в котором вместо данных используются специальные параметры. СУБД компилирует этот шаблон, анализирует, оптимизирует и транслирует его, но не выполняет. Результат компиляции сохраняется для последующего использования.

Выполнение. Приложение передает в СУБД конкретные значения параметров. База данных подставляет эти значения в уже скомпилированный шаблон и выполняет запрос. На этом этапе переданные данные интерпретируются исключительно как значения, и не могут быть интерпретированы как часть исполняемого SQL-кода.

Современные международные исследования подтверждают, что фильтрация запросов с помощью параметров является безопасным и надежным методом защиты, который используется, в том числе, в гибридных техниках обнаружения атак [4].

Однако данное решение не универсально. Ограничения связны с динамическими запросами, параметризация не позволяет подставлять имена таблиц, колок или операторов, что требует каждый раз снова производить этап компиляции для динамической сортировки или выбора таблицы. Данные запросы не защищены на уровне хранимых процедур.

2.2 Использование ORM-фреймворков

Объектно-реляционное отображение представляет собой технологию программирования, которая связывает базы данных с концепциями объектно-ориентированных языков, создавая «виртуальную объектную базу данных». ORM-фреймворки абстрагируют работу с базой данных, позволяя разработчикам взаимодействовать с ней через объекты, а не через написание SQL-запросов.

Преимущество ORM-фреймворков заключается в том, что они, как правило, сами используют параметризованные запросы. Как отмечает Усванова Д.Р. в своем исследовании методов защиты веб-приложений от SQL-инъекций, применение объектно-реляционного отображения является одним из трех ключевых подходов к защите наряду с использованием подготовленных выражений и валидацией входных данных [5]. Фреймворки автоматически преобразуют операции с объектами в параметризованные SQL-запросы.

SQL-инъекции могут встроится в SQL-запрос при использовании ORM-Фреймворков в случаях, когда разработчик сознательно обходит безопасные API, при конкатенации пользовательских данных в строки запросов или при использовании прямого SQL-запроса. ORM-фреймворки эффективно защищают от инъекций при соблюдении стандартных практик.

2.3 Экранирование спецсимволов

Экранирование спецсимволов представляет собой метод защиты, при котором символы, имеющие специальное значение в SQL, преобразуются в безопасную форму путем добавления экранирующего символа. Например, одинарная кавычка ' превращается в последовательность \', что позволяет интерпретировать ее как часть передаваемых данных, а не как управляющий символ.

Основной недостаток экранирования заключается в том, что оно требует строгого учета контекста применения и используемых кодировок. Ошибки в реализации могут привести к тому, что вредоносный ввод останется неэкранированным и будет интерпретирован в исполняемый код. Экранирование не защищает от всех типов SQL-инъекций и должно применяться только как дополнительный, но не основной механизм защиты.

2.4 Валидация входных данных

Валидация входных данных представляет собой процесс проверки соответствия пользовательского ввода ожидаемому формату перед его дальнейшей обработкой.

Наиболее надежным подходом к валидации является использование «белых списков», когда определяется множество допустимых значений и отвергает все, что не соответствует заданным критериям. Такой подход особенно эффективен для полей с четко определенным форматом: числовых идентификаторов, дат, кодов, телефонных номеров или адресов электронной почты.

Альтернативным подходом является использование «черных списков», когда система пытается обнаружить и заблокировать известные вредоносные паттерны.

В исследовании Самандарова Б.С. с соавторами, посвященном математическому моделированию уязвимостей информационных систем, показано, что валидация входных данных в сочетании с другими мерами защиты значительно снижает вероятность успешной SQL-инъекции [6].

Валидация входных данных с использованием белых списков является эффективным методом защиты, но должна рассматриваться как дополнительный уровень безопасности в комплексе с другими средствами. Поскольку она не нейтрализует вредоносный код, а проверяет его формат, что при неверной реализации может позволить злоумышленнику воспользоваться легитимными, но опасными в данном контексте символами.

3. Роль Web Application Firewall в защите от SQL-инъекций

Для защиты от SQL-инъекций Web Application Firewall использует два основных метода обнаружения: сигнатурный анализ и анализ аномалий. Сигнатурный анализ выявляет атаки путем сопоставления входящих запросов с базой известных вредоносных шаблонов, тогда как анализ аномалий обнаруживает отклонения от нормотипичного поведения, что позволяет идентифицировать как известные, так и новые угрозы [7].

Web Application Firewall выступает критически важным элементом комплексной защиты веб-приложений, обеспечивая внешний барьер, который компенсирует возможные недостатки безопасности на уровне кода и позволяет оперативно реагировать на вновь выявленные угрозы без остановки работы приложения, путем создания новых правил, блокирующих запросы к недавно выявленным, уязвимым частям приложения.

4. Сравнительный анализ и необходимость комплексного подхода

Представим сравнительную характеристику рассмотренных методов в виде таблицы.

Метод защиты	Уровень применения	Эффективность	Преимущества	Недостатки
Параметризованные запросы	Код приложения	Высокая	Устраняют саму причину уязвимости; гарантированно разделяют код и данные; повышают производительность за счет кэширования планов запросов.	Бессильны при подстановке имен таблиц, колонок или операторов, где структура запроса должна меняться.

Метод защиты	Уровень применения	Эффективность	Преимущества	Недостатки
ORM-фреймворки	Код приложения	Высокая	Автоматизируют параметризацию на уровне объектов, скрывая от разработчика написание безопасного кода.	Могут генерировать неоптимальные SQL-запросы, что снижает производительность; риск при обходе встроенных безопасных API.
Валидация входных данных	Код приложения	Средняя	Эффективна для полей с четким форматом; реализует принцип «белых списков», позволяя контролировать имена объектов БД, которые нельзя параметризовать.	Не защищает от инъекций в значениях данных; может быть обойдена при сложных многобайтовых кодировках, если используется как единственная защита.
Экранирование спецсимволов	Код приложения	Средняя	Может использоваться как запасной вариант для систем с унаследованной архитектурой, где невозможно внедрение параметризации.	Требует строгого учета контекста и кодировок; высок риск ошибки реализации, что делает метод ненадежным.
Web Application Firewall	Инфраструктура	Высокая	Обеспечивает защиту на уровне сети, перекрывая недостатки кода без немедленного внесения изменений в приложение.	Не устраняет причину уязвимости в коде; требует постоянной настройки и обновления сигнатур для обнаружения новых атак.

Таблица 1 – Сравнение методов защиты от SQL-инъекций.

Как видно из таблицы, ни один метод не является полноценным средством защиты, поскольку каждый имеет зону ответственности и зону уязвимости. Параметризация эффективно блокирует инъекции в данные, но бессильна при динамической структуре запроса, где необходима валидация по «белым спискам». Web Application Firewall компенсирует возможные ошибки кода на

инфраструктурном уровне, но не устраняет их причину. Максимальная безопасность достигается только при комбинации этих подходов, построенной по принципу эшелонированной защиты, где валидация контролирует структуру, параметризация – данные, а Web Application Firewall перехватывает атаки, сумевшие преодолеть первые рубежи. В случаях, где необходимо обезопасить систему с архаичной архитектурой, в которой реализация параметризации невозможна, необходимо использовать экранирование. При грамотной комбинации таких решений можно добиться как достаточного уровня защиты, так и сохранения производительности системы.

Заключение

SQL-инъекции остаются серьезной угрозой, однако их предотвращение вполне достижимо при реализации комплексной стратегии защиты. Проведенный анализ показывает, что наиболее эффективным подходом является сочетание превентивных мер на этапе разработки и активных средств защиты на этапе эксплуатации. Применение такого комплексного подхода позволяет минимизировать риски и обеспечить надежную защиту веб-приложений.

Список литературы:

1. A05:2025 Injection / OWASP Top 10:2025. – URL: https://owasp.org/Top10/2025/A05_2025-Injection/ (дата обращения: 03.03.2026). – Текст : электронный.

2. Шелягов, С.А. Анализ применения механизма SQL-инъекции для получения несанкционированного доступа к информации на основе усечение данных в SQL / С.А. Шелягов, И.В. Кудрявченко. – Символ науки. – 2015. – № 9. – С. 123–126. – URL: https://cyberleninka.ru/article/n/analiz-primeneniya-mehanizma-sql-inektsii-dlya-polucheniya-nesanktsionirovannogo-dostupa-k-informatsii-na-osnove-usechenie-dannyh-v-sql (дата обращения: 03.03.2026). – Текст : электронный.

3. Пантюхин, М.А. SQL Injection: основы, угрозы и современные подходы к защите / М.А. Пантюхин. – Научный аспект. – № 4-2024. – URL: https://na-journal.ru/4-2024-informacionnye-tekhnologii/11642-sql-injection-osnovy-ugrozy-i-sovremennye-podhody-k-zashchite (дата обращения: 03.03.2026). – Текст : электронный.

4. Arif, S.A.B. The Theoretical Foundations and Literature Analysis a Hybrid Detection Technique Against Malicious SQL Attacks on Web Applications / S.A.B. Arif, S. Wani. – Journal of Information Systems Engineering and Management. – 2025. – Vol. 10, no. 35s. – P. 1093–1100. – DOI: https://doi.org/10.52783/jisem.v10i35s.6218. – URL: https://jisem-journal.com/index.php/journal/article/view/6218/2878 (дата обращения: 03.03.2026). – Текст : электронный.

5. Усванова, Д.Р. Исследование методов защиты ВЕБ-приложений от атак типа SQL-ИНЪЕКЦИЯ / Д.Р. Усванова. – Международный журнал информационных технологий и энергоэффективности. – 2024. – Т. 9, № 12(50). – С. 5–7. –

URL: http://openaccessscience.ru/index.php/ijcse/article/view/763/793 (дата обращения: 03.03.2026). – Текст : электронный.

6. Самандаров, Б.С. Анализ и моделирование уязвимостей безопасности информационных систем / Б.С. Самандаров, Г.А. Гулмирзаева, Д.Р. Жолдасбаев. – Информатика. Экономика. Управление. – 2025. – Т. 4, № 1. – С. 2019–2026. – DOI: https://doi.org/10.47813/2782-5280-2025-4-1-2019-2026. – URL: https://cyberleninka.ru/article/n/analiz-i-modelirovanie-uyazvimostey-bezopasnosti-informatsionnyh-sistem (дата обращения: 03.03.2026). – Текст : электронный.

7. Durmuşkaya, M.E. Web application firewall based on machine learning models / M.E. Durmuşkaya, S. Bayraklı. – PeerJ Computer Science. – 2025. – Vol. 11:e2975. – DOI: https://doi.org/10.7717/peerj-cs.2975. – URL: https://peerj.com/articles/cs-2975/ (дата обращения: 03.03.2026). – Текст : электронный.