Состязательное тестирование моделей машинного обучения, предназначенных для обнаружения SQL-инъекций

Е.С. Егорова, О.Р. Лапонина

***Аннотация* – SQL-инъекция остаётся одной из наиболее распространённых уязвимостей веб-приложений, стабильно входящей в перечень критических угроз согласно классификации OWASP. В ответ на возрастающую сложность атак современные исследовательские и инженерные подходы всё активнее интегрируют методы машинного и глубокого обучения для автоматического обнаружения вредоносных SQL-запросов. Несмотря на высокие показатели точности таких моделей на статичных выборках, они демонстрируют уязвимость к состязательным атакам — специально модифицированным входным данным, сохраняющим синтаксическую корректность и семантическую эквивалентность исходным запросам, но способным нарушить логику классификации модели.**

В данной работе предложен метод состязательного тестирования моделей машинного обучения, предназначенных для выявления SQL-инъекций. Подход основан на последовательном применении мутационных преобразований, сохраняющих синтаксическую корректность запросов и семантическую эквивалентность исходным запросам. Алгоритм реализован как итеративный процесс, направленный на снижение уверенности модели в классификации вредоносного запроса при отсутствии доступа к внутренним параметрам модели, что делает его применимым в условиях «чёрного ящика».

Экспериментальная валидация метода проведена на трёх моделях различных архитектур: MobileBERT, LSTM и гибридной CNN-LSTM. Полученные результаты показали снижение метрик точности и полноты по классу инъекционных запросов после применения состязательных мутаций.

Предложенный метод может быть использован как инструмент объективной оценки устойчивости моделей к полиморфным SQL-инъекциям и основа для построения более надёжных систем защиты. Благодаря универсальности и воспроизводимости, он также применим для расширения обучающих выборок в рамках адверсариального обучения. В перспективе метод может быть адаптирован для других типов атак на текстовые данные (например, XSS), а также интегрирован в промышленные средства обеспечения веб-безопасности.

***Ключевые слова* – SQL-инъекции, машинное обучение, состязательные атаки, тестирование устойчивости, мутационные операторы, MobileBERT, LSTM, CNN-LSTM.**

I. Введение

SQL-инъекция (SQLI) — одна из наиболее распространённых уязвимостей веб-приложений, стабильно входящая в число главных угроз по версии OWASP [1]. Современные веб-приложения всё чаще используют методы машинного и глубокого обучения для автоматического выявления вредоносных запросов. Эти подходы демонстрируют высокую точность на статических наборах данных, но зачастую оказываются уязвимыми к состязательным атакам — модифицированным запросам, сохраняющим исходную семантику, но нарушающим логику классификации [2].

В последние годы исследователи предложили множество методов обнаружения SQL-инъекций, которые в основном классифицируются на три подхода, и каждый из них имеет свои недостатки [3].

Классические сигнатурные подходы к обнаружению SQLI страдают от высокой доли ложных срабатываний, неспособности к генерализации и необходимости ручного обновления правил. Использование моделей машинного обучения позволило автоматизировать извлечение признаков, повысить гибкость и адаптивность систем. Глубокие нейросетевые архитектуры, такие как LSTM и трансформеры, особенно эффективны при обработке последовательных данных. Однако даже такие модели подвержены ошибкам при минимальных, но грамотно спроектированных изменениях входных данных.

Исследования последних лет показывают, что адаптивные атаки с использованием мутаций, кодирования, логических перестроек или эволюционных стратегий способны обходить даже продвинутые детекторы SQL-инъекций [4–7]. Такие методы позволяют генерировать реалистичные запросы, которые сохраняют свою функциональность, но не распознаются системами защиты.

В данной работе рассматривается проблема оценки устойчивости моделей машинного обучения к состязательным атакам. Предлагается унифицированная методика состязательного тестирования, основанная на каскадной системе мутационных преобразований SQL-запросов. Метод реализован как итеративный процесс, при котором каждое преобразование направлено на снижение уверенности модели в распознавании вредоносного запроса, без нарушения его синтаксиса и логики. Алгоритм ориентирован на практическое применение — он не требует доступа к внутренним данным модели, может быть масштабирован на

Статья получена 17 мая 2025.
Е. С. Егорова - МГУ имени М.В. Ломоносова (email: elenaziy1@gmail.com)

О.Р. Лапонина - МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru)

различные архитектуры и использован на этапах валидации защитных решений. Экспериментальная проверка метода проведена на трёх моделях различной архитектуры: MobileBERT, LSTM и гибридной CNN-LSTM. Результаты показали значительное снижение точности классификации после применения к вредоносным запросам адаптивных мутаций. Таким образом, предложенная методика может быть использована как инструмент объективной оценки устойчивости, а также как средство генерации обучающих данных для адверсариального обучения — процесса, направленного на повышение надёжности моделей в условиях реальных угроз.

II. Анализ существующих решений

A. Обзор моделей машинного и глубокого обучения, предназначенных для обнаружения SQL-инъекций

В последнее десятилетие в области обнаружения SQL-инъекций активно развиваются методы, основанные на машинном и глубоком обучении. Среди них высокую эффективность показывают ансамблевые алгоритмы (Boosting, Random Forest), методы опорных векторов (SVM с различными ядрами), а также нейросетевые архитектуры, включая CNN, LSTM и их гибридные комбинации (CNN-BiLSTM, MVC-BiCNN) [8-13]. Современные модели, такие как SQLNN [12] и Elastic-Pooling CNN (EP-CNN) [14], достигают точности выше 96–99%. Однако остаются актуальными проблемы вычислительной нагрузки, чувствительности к качеству препроцессинга данных и ограниченной адаптивности к новым видам атак. Эти недостатки указывают на важность оценки устойчивости моделей к внешнему воздействию, в частности к состязательным атакам.

B. Методы генерации состязательных атак

Состязательные атаки (adversarial attacks) эксплуатируют уязвимости моделей, изменяя входные данные так, чтобы они вызывали ошибочные предсказания при сохранении первоначальной семантики [15]. Атаки типа «чёрный ящик», при которых злоумышленник не имеет доступа к внутренним параметрам модели и её архитектуре, но может взаимодействовать с ней через входные данные и анализировать выходные ответы, обладают высокой практической значимостью, так как отражают реальные условия функционирования моделей в закрытых программных и инфраструктурных системах [16].

Среди известных инструментов для тестирования устойчивости моделей к атакам типа «чёрный ящик» можно выделить систему WAF-A-MoLE [4], основанную на мутационном подходе к генерации модифицированных SQL-запросов. Алгоритм применяет фиксированный набор синтаксических преобразований, при этом сохраняется исходная семантика запросов. Эти преобразования не нарушают логическую структуру атакующих конструкций, что позволяет сохранять работоспособность инъекций. Проведённые авторами эксперименты показали, что даже ограниченный набор мутаций обеспечивает успешный обход ряда промышленных систем фильтрации, однако универсальность инструмента ограничена, так как он ориентирован преимущественно на обход статических сигнатурных WAF.

Более гибкие и адаптивные подходы реализуются с применением методов обучения с подкреплением и генеративного моделирования. Так, Guan и др. [5] разработали агентную модель на основе алгоритма Soft Actor-Critic, способную последовательно изменять SQL-запросы в рамках атаки типа чёрный ящик. Генеративные методы, представленные, например, в работах Alqhtani и др. [6] (Conditional Tabular GAN) и Lu и др. [7] (комбинация GAN и генетического алгоритма), продемонстрировали применимость к задачам обхода систем обнаружения, а также перспективность в задачах обогащения обучающих выборок за счёт синтеза разнообразных вариантов инъекций.

Указанные подходы подтверждают, что даже при отсутствии информации о структуре модели возможно целенаправленно модифицировать входные данные с целью снижения точности классификации. Это подчёркивает необходимость включения методов оценки устойчивости в общий цикл разработки моделей обнаружения SQL-инъекций, а также актуальность использования методов состязательного обучения как средства повышения надёжности.

C. Метрики качества

Для оценки эффективности моделей обнаружения SQL-инъекций в большинстве рассмотренных исследований используются стандартные метрики бинарной классификации. Ниже приводится описание каждой из используемых метрик [6]:

Основные показатели:

1. True Positive (TP): Количество правильно идентифицированных атак. Это случаи, когда модель корректно классифицирует атакующий запрос как вредоносный.
2. True Negative (TN): Количество правильно идентифицированных легитимных запросов, т.е. случаев, когда модель корректно определяет отсутствие атаки.
3. False Positive (FP): Количество ложноположительных срабатываний, т.е. случаев, когда легитимный запрос ошибочно классифицируется как вредоносный.
4. False Negative (FN): Количество ложноотрицательных срабатываний, т.е. случаев, когда вредоносный запрос не обнаруживается моделью.

Производные метрики:

1) Accuracy: Отражает общую долю правильно классифицированных запросов, как вредоносных, так и легитимных. Рассчитывается по формуле:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

2) Precision: Определяет, какая доля запросов, классифицированных как вредоносные, действительно являются таковыми. Определяется формулой:

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

3) Recall (Полнота): Доля правильно классифицированных вредоносных запросов относительно всех вредоносных запросов, присутствующих в наборе данных. Рассчитывается как:

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

4) F1-score: среднее гармоническое точности и полноты. Рассчитывается по формуле:

$$F1 = 2\frac{Precision*Recall}{Precision+Recall} \quad (4)$$

III. Метод состязательного тестирования моделей обнаружения SQL-инъекций

A. *Описание метода*

Представленный в данной работе метод предназначен для оценки устойчивости моделей машинного обучения, применяемых для обнаружения SQL-инъекций. Метод характеризуется универсальностью: он применим к моделям любой архитектуры и не требует знаний о внутреннем устройстве модели. Основу метода составляет последовательное применение семантически инвариантных мутаций SQL-запросов с контролем за изменением отклика модели. Преимущество метода заключается в строгом сохранении синтаксической корректности и логической эквивалентности исходных и мутированных запросов, что обеспечивает единообразный протокол тестирования и позволяет проводить сравнительный анализ моделей в контролируемых условиях.

Процедура тестирования, реализованная в виде программного инструмента на языке Python, организована как итеративный процесс, реализующий жадную стратегию оптимизации. На каждом шаге выбирается такая мутация, которая локально минимизирует вероятность классификации запроса как инъекционного (P_{inj}). По достижении порогового значения $P_{inj} \leq \theta$ мутация фиксируется как успешная. Алгоритм включает фазы предварительной и финальной обработки, использование которых позволяет реализовать многослойную стратегию атаки.

Логическая структура пайплайна тестирования устойчивости модели к атакам типа SQL-инъекций:

1) Инициализация.

- Загрузка предобученной модели классификации SQL-запросов.
- Загрузка валидационного набора данных, содержащего вредоносные SQL-запросы.

2) Основной цикл обработки запросов.

Последовательная обработка всех вредоносных SQL-запросов из валидационного набора данных. Для каждого запрос вычисляется вероятность классификации данного запроса как инъекции, обозначаемая как P_{inj}. Если $P_{inj} > \theta$ (пороговое значение, например, θ=0,5), запрос направляется на этап мутационного тестирования.

3) Мутационное тестирование (итерационный процесс).

К исходному запросу последовательно применяются базовые мутации:

- замена числовых констант;
- вставка SQL-комментариев;
- замена строковых литералов.

После применения каждой мутации вычисляется вероятность $P_{inj\_mutated}$ для мутированного запроса.

▪ Если $P_{inj\_mutated} \leq \theta$, мутация считается успешной, поскольку модель классифицировала вредоносный запрос как допустимый (валидный). В противном случае процесс мутации повторяется не более чем N раз (заданное максимальное количество итераций).

По завершении N итераций сохраняется мутированный вариант запроса с минимальным значением $P_{inj\_mutated}$ для последующей обработки.

4) Применение финальных мутаций.

К мутированному запросу последовательно применяются дополнительные модификации в рандомизированном порядке:

- шестнадцатеричное (hex) кодирование;
- Unicode-кодирование;
- смешение регистра символов в ключевых словах (mix_case);
- замена логических операторов (logic_op);
- URL-кодирование(url_encode).

После каждой из модификаций выполняется классификация мутированного запроса моделью. В случае, если запрос классифицирован как валидный, фиксируется успешный обход модели.

5) Логирование результатов.

Для каждого обработанного запроса сохраняются:

- исходный вредоносный SQL-запрос;
- итоговый мутированный SQL-запрос;
- тип последней примененной мутации;
- количество итераций мутаций;
- успешность/неуспешность обхода модели;
- $P_{inj\_mutated}$ - вероятность классификации мутированного запроса как инъекции.

6) Переход к обработке следующего SQL-запроса из валидационного набора данных с повторением описанных выше этапов.

7) Анализ устойчивости модели.

Вычисляются метрики качества классификации на валидационном наборе данных до проведения тестирования на устойчивость, а также после применения атакующих мутаций. Это позволяет объективно оценить снижение эффективности модели вследствие осуществлённых атак. Сопоставление результатов обеспечивает количественную характеристику степени уязвимости модели к состязательным воздействиям.

На Рисунке 1 изображена схема разработанного решения.

B. *Операторы мутаций*

Мутации реализуются в рамках статически заданного множества операторов. Для обеспечения корректности процесса состязательного тестирования моделей обнаружения SQL-инъекций каждый мутационный оператор, применяемый в рамках фреймворка, обязан удовлетворять следующим формальным требованиям:

1) Семантическая инвариантность

Мутация должна сохранять исходную семантику запроса. Это означает, что результат выполнения мутированного запроса на целевой базе данных должен быть эквивалентен результату выполнения исходного запроса.

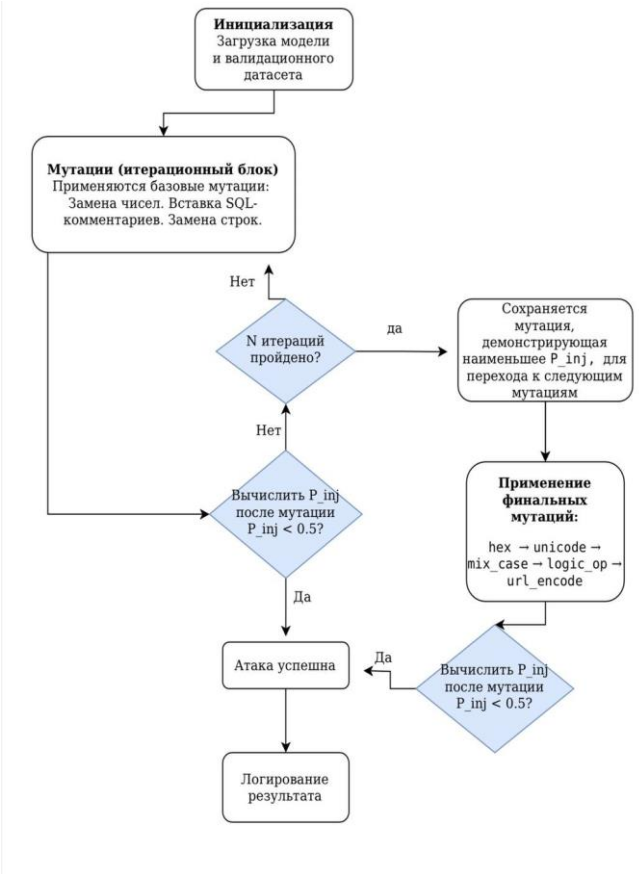

Рисунок 1. Схема разработанного решения

Пусть Q — исходный SQL-запрос, $Q' = M(Q)$ — результат применения мутационного оператора M. Тогда для любой допустимой базы данных D должно выполняться:

$$Result(Q, D) = Result(Q', D) \quad (5)$$

где $Result(\cdot)$ — функция исполнения запроса в базе данных.

2) Синтаксическая корректность

Мутация должна сохранять синтаксическую корректность запроса в рамках формальной грамматики языка SQL. Мутированный запрос должен успешно парситься стандартным SQL-интерпретатором без синтаксических ошибок.

$$Q' \in L_{SQL} \quad (6)$$

где L_{SQL} — множество всех синтаксически корректных SQL-запросов.

3) Детерминированность применения

Применение мутационного оператора к одному и тому же исходному запросу должно давать воспроизводимый результат в отсутствие внешнего стохастического влияния (идемпотентность при фиксированных параметрах применения).

Для всех Q,

$$M(Q) = M(Q) \quad (7)$$

4) Поддержка итеративного применения

Операторы должны быть совместимы с последовательным наложением: после применения одного оператора результат должен оставаться допустимым входом для последующих мутаций.

Пусть $M_1, M_2, \dots, M_k$ — последовательность мутационных операторов. Тогда для всех $i \in [1, k]$:

$$M_i(M_{i-1}(\dots M_1(Q))) \in L_{SQL} \quad (8)$$

и

$$Result(M_i(M_{i-1}(\dots M_1(Q))), D) = Result(Q, D) \quad (9)$$

Набор мутационных операторов, реализованный в предлагаемом фреймворке для состязательного тестирования моделей обнаружения SQL-инъекций, был сформирован на основе анализа распространённых методов обхода фильтрации запросов в реальных атакующих сценариях. Реализованные мутации направлены на систематическое воздействие на различные уровни обработки запроса (лексический, синтаксический, семантический), что позволяет объективно оценивать устойчивость моделей к широкому спектру реалистичных атакующих воздействий. В Таблице 1 приведены мутационные операторы, задействованные в нашем методе.

Таблица 1 Описание мутационных операторов, задействованных при тестировании

Название мутации	Описание
insert_comment	Вставка SQL-комментария (/**/) после указанного токена в запросе
mix_case_sql_keywords	Случайное смешение регистра символов в ключевых словах SQL
replace_logical_operators	Замена логических операторов (AND, OR, NOT) на альтернативные символические формы (&&, || , !)
hex_encoded	Замена символов в запросе на шестнадцатеричное представление
unicode_encoded	Замена символов в запросе на эквивалентные Unicode-представление
replace_sql_string	Замена строковых литералов на новое строковое значение
replace_numbers	Замена числовых значений в запросе на альтернативное число
url_encoded	Замена символов на их представление в URL-кодировке

C. Тестируемые модели

Для комплексной оценки устойчивости моделей машинного обучения к состязательным атакам на SQL-запросы был сформирован репрезентативный набор моделей, охватывающий три архитектурных подхода:

1) Модель на основе MobileBERT.

Архитектурные особенности: модель построена на базе компактной трансформерной архитектуры MobileBERT (google/mobilebert-uncased), оптимизированной для задач классификации последовательностей. Исходная модель была дополнительно дообучена на специализированном корпусе SQL-запросов для решения задачи бинарной классификации валидных запросов и SQL-инъекций.

Следует отметить, что комбинация трансформерной архитектуры и эффективных методов обучения делает данную модель высокоадаптивной к задачам фильтрации SQL-запросов и особенно интересной для оценки воздействия состязательных мутаций.

2) LSTM-модель

Модель базируется на архитектуре долгосрочной краткосрочной памяти (LSTM), широко применяемой для анализа текстовых последовательностей. Особенности обучения и обработки данных: входные запросы токенизируются с помощью стандартного Keras Tokenizer, что обеспечивает обобщённую обработку текста без учета специфики языка SQL. Такая стратегия упрощает обучение, но делает модель потенциально более уязвимой к мутациям, направленным на изменение лексических шаблонов. Эта модель выступает в роли базового бенчмарка, позволяя оценить, насколько рекуррентные архитектуры подвержены воздействию атак через последовательные мутации запросов.

3) CNN-LSTM модель

Архитектурные особенности: модель реализует гибридную архитектуру, сочетающую сверточные нейронные сети (CNN) и рекуррентные сети (LSTM). Такой подход позволяет одновременно учитывать локальные паттерны в тексте запроса и захватывать долгосрочные зависимости. Первичная обработка запроса осуществляется с помощью сверточных фильтров, выделяющих локальные структуры, характерные для инъекций (например, OR 1=1, UNION SELECT). Последующая обработка выполняется LSTM-слоем, что позволяет учитывать глобальный контекст в последовательности. Модель использует кастомный токенизатор, специально адаптированный под синтаксические особенности языка SQL. Это повышает её чувствительность к модификациям синтаксиса и усложняет задачу обхода модели при помощи мутаций. Данная модель рассматривается как представитель устойчивых гибридных архитектур.

D. Результаты тестирования

В рамках проведённого эксперимента были проанализированы изменения основных метрик качества моделей машинного обучения до и после применения предложенного пайплайна состязательных мутаций SQL-запросов. Анализ проводился по следующим метрикам: точность (precision), полнота (recall), F1-мера (f1-score) и accuracy.

1) Модель MobileBERT

До мутаций:

- Модель продемонстрировала почти идеальные характеристики на обоих классах: precision и recall превышали 0.99 для классов Valid и Injection.
- Accuracy классификации составила 0.99.

После мутаций:

- Наблюдается снижение точности по классу Injection: precision упал с 1.00 до 0.83, recall — с 0.99 до 0.82, F1-мера — до 0.82.
- Accuracy модели снизилась до 0.90.

Вывод:

Несмотря на относительно высокую устойчивость, модель на основе MobileBERT продемонстрировала уязвимость к атакующим мутациям. Падение метрики полноты (recall) свидетельствует о росте числа ложных отрицаний — то есть инъекционные запросы стали чаще классифицироваться как валидные.

2) Базовая LSTM-модель

До мутаций:

- Умеренные показатели качества: precision — 0.93 для класса Valid и 0.88 для класса Injection, recall — 0.95 и 0.85 соответственно.
- Accuracy составила 0.91.

После мутаций:

- Резкое ухудшение качества для класса Injection: precision упал до 0.70, recall — до 0.60, F1-мера снизилась до 0.65.
- Accuracy модели уменьшилась до 0.82.

Вывод:

LSTM-модель оказалась наименее устойчивой к мутационным атакам. Существенное падение recall на классе инъекций указывает на значительное количество нераспознанных вредоносных запросов после мутаций.

3) Гибридная CNN-LSTM модель

До мутаций:

- Precision и recall на обоих классах находятся на уровне 0.90–0.95.
- Accuracy составляет 0.91.

После мутаций:

- Снижение качества для класса Injection наблюдается, но в меньшей степени, чем у LSTM:
  - Precision упал с 0.90 до 0.80
  - Recall снизился с 0.85 до 0.72
  - F1-мера снизилась с 0.87 до 0.76
- Accuracy снизилась до 0.87.

Вывод:

Гибридная CNN-LSTM модель показала большую устойчивость к атакующим мутациям по сравнению с LSTM, но также продемонстрировала ухудшение распознавания инъекций, сопоставимое с MobileBERT.

Для наглядности в Таблице 2 приведем сравнительную таблицу метрик моделей до и после мутаций.

Результаты тестирования показали снижение качества

классификации вредоносных SQL-запросов для всех рассмотренных моделей после применения состязательных мутаций. Различия в степени деградации метрик подчеркивают влияние архитектуры на устойчивость к полиморфным модификациям.

Таблица 2 Сравнение метрик моделей до и после мутаций

Модель	Метрика	До мутаций	После мутаций
MobileBERT	Accuracy	0.99	0.90
	Recall (Injection)	0.99	0.82
	Precision (Injection)	1.00	0.83
LSTM	Accuracy	0.91	0.82
	Recall (Injection)	0.85	0.60
	Precision (Injection)	0.88	0.70
CNN-LSTM	Accuracy	0.91	0.87
	Recall (Injection)	0.85	0.72
	Precision (Injection)	0.90	0.80

Эти результаты подчёркивают критическую важность использования состязательных данных в процессе обучения и дообучения моделей для задач обнаружения SQL-инъекций. Такой подход называется адверсариальной тренировкой [17], суть которого заключается в переобучении классификатора с включением в обучающую выборку также атакующих состязательных примеров. Применение данного подхода позволяет повысить устойчивость модели к состязательным примерам, однако сопровождается снижением общей точности классификации. Вместе с тем, как показано N.Carlini и др. [18], данный метод не является окончательным решением проблемы: он лишь усложняет злоумышленнику задачу поиска эффективных состязательных примеров, не устраняя уязвимость полностью. Как бы то ни было, без регулярной адаптации к новым вариантам обходов даже современные модели показывают значительное снижение качества классификации при реализации состязательных атак.

IV. Заключение

В рамках данной работы представлен метод состязательного тестирования устойчивости моделей машинного обучения к атакам на основе SQL-инъекций, базирующийся на последовательном применении мутационных преобразований запросов, сохраняющих синтаксическую корректность и семантическую эквивалентность запросов. Предложенный подход реализует контролируемую адаптивную оптимизацию, выбирая локально оптимальную трансформацию, минимизирующую вероятность обнаружения инъекции, и позволяет накапливать последовательные мутации для моделирования сложнообнаружимых атак.

Эксперименты с MobileBERT, LSTM и CNN-LSTM моделями подтвердили, что даже современные модели с высокими начальными показателями качества классификации оказываются уязвимыми к целенаправленным мутациям входных данных.

Разработанный метод позволяет выявлять уязвимости моделей к полиморфным SQL-инъекциям, проводить оценку качества моделей в реалистичных условиях атак и объективное сравнение различных архитектурных подходов. Способность логировать полную цепочку трансформаций запросов делает предложенный инструмент тестирования пригодным генерации расширенных обучающих выборок данных, которые можно использовать для повышения устойчивости систем.

В перспективе разработанный подход может быть адаптирован к другим типам мутаций и к другим типам атак, эксплуатирующим уязвимости в текстовых данных (например, XSS-инъекции), а также интегрирован в промышленные системы анализа сетевого трафика и защиты веб-приложений.

Благодарности

Тема статьи, написанной по результатам магистерской диссертации, соответствует направлению “Кибербезопасность” на факультете ВМК МГУ [19]. Как примеры похожих работ см. публикации [20,21].

Как обычно, отмечаем работы В.П. Куприяновского и его многочисленных соавторов, положивших начало цифровой повестке в журнале [22, 23].

Библиография

[1] OWASP Top Ten // OWASP. 2020. URL: https://owasp.org/www-project-top-ten/ (Дата обращения: 01.05.2025).

[2] Wang Y., Kosinski M. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images //Journal of personality and social psychology. – 2018. – Т. 114. – №. 2. – С. 246.

[3] Kolhe A. K., Adhikari P. A SQL Injection: Internal Investigation of Injection, Detection and Prevention of SQL Injection Attacks //International Journal of Engineering Research & Technology (IJERT). – 2014. – Т. 3. – №. 1.

[4] Demetrio L. et al. Waf-a-mole: evading web application firewalls through adversarial machine learning //Proceedings of the 35th Annual ACM Symposium on Applied Computing. – 2020. – С. 1745-1752.

[5] Guan Y. et al. Ssqli: A black-box adversarial attack method for sql injection based on reinforcement learning //Future Internet. – 2023. – Т. 15. – №. 4. – С. 133.

[6] Alqhtani M., Alghazzawi D., Alarifi S. Black-Box Adversarial Attacks Against SQL Injection Detection Model //Contemporary Mathematics. – 2024. – С. 5098-5112.

[7] Boekweg K. I. Developing a SQL Injection Exploitation Tool with Natural Language Generation : дис. – Brigham Young University, 2024.

[8] Юдова Е. А., Лапонина О. Р. Сравнительный анализ подходов к обнаружению SQL-инъекций с помощью методов машинного обучения //International Journal of Open Information Technologies. – 2023. – Т. 11. – №. 6. – С. 175-181.

[9] Alghawazi M., Alghazzawi D., Alarifi S. Detection of sql injection attack using machine learning techniques: a systematic literature review //Journal of Cybersecurity and Privacy. – 2022. – Т. 2. – №. 4. – С. 764-777.
[10] Gandhi N. et al. A CNN-BiLSTM based approach for detection of SQL injection attacks //2021 International conference on computational intelligence and knowledge economy (ICCIKE). – IEEE, 2021. – С. 378-383.
[11] Kakisim A. G. A deep learning approach based on multi-view consensus for SQL injection detection //International Journal of Information Security. – 2024. – Т. 23. – №. 2. – С. 1541-1556.
[12] Zhang W. et al. Deep Neural Network-Based SQL Injection Detection Method //Security and Communication Networks. – 2022. – Т. 2022. – №. 1. – С. 4836289.
[13] Tang P. et al. Detection of SQL injection based on artificial neural network //Knowledge-Based Systems. – 2020. – Т. 190. – С. 105528.
[14] Xie X. et al. Sql injection detection for web applications based on elastic-pooling cnn //IEEE Access. – 2019. – Т. 7. – С. 151475-151481.
[15] Finlayson S. G. et al. Adversarial attacks on medical machine learning //Science. – 2019. – Т. 363. – №. 6433. – С. 1287-1289.
[16] Alatwi H. A., Aldweesh A. Adversarial black-box attacks against network intrusion detection systems: A survey //2021 IEEE World AI IoT Congress (AIIoT). – IEEE, 2021. – С. 0034-0040.
[17] Yan S. et al. A survey of adversarial attack and defense methods for malware classification in cyber security //IEEE Communications Surveys & Tutorials. – 2022. – Т. 25. – №. 1. – С. 467-496.
[18] Carlini N., Wagner D. Adversarial examples are not easily detected: Bypassing ten detection methods //Proceedings of the 10th ACM workshop on artificial intelligence and security. – 2017. – С. 3-14
[19] Сухомлин, Владимир Александрович. "Создание профиля" Кибербезопасность и искусственный интеллект" для направления подготовки ФИИТ на основе куррикулумного подхода." Современные информационные технологии и ИТ-образование 17.3 (2021): 724-734.
[20] Korniukhina, Sofia P., and Olga R. Laponina. "Research of the Capabilities of Deep Learning Algorithms to Protection Against Phishing Attacks." International Journal of Open Information Technologies 11.6 (2023): 163-174.
[21] Zubrienko, G. A., and O. R. Laponina. "Data Sampling Techniques for Anomaly Detection in Network Traffic." International Journal of Open Information Technologies 4.10 (2016): 1-8.
[22] О работах по цифровой экономике / В. П. Куприяновский, Д. Е. Намиот, С. А. Синягов, А. П. Добрынин // Современные информационные технологии и ИТ-образование. – 2016. – Т. 12, № 1. – С. 243-249. – EDN XEQRFJ.
[23] Розничная торговля в цифровой экономике / В. П. Куприяновский, С. А. Синягов, Д. Е. Намиот [и др.] // International Journal of Open Information Technologies. – 2016. – Т. 4, № 7. – С. 1-12. – EDN WCMIWN.

Adversarial testing of machine learning models designed for SQL injection detection

E. S. Egorova, O.R. Laponina

Annotation **– SQL injection remains one of the most prevalent web application vulnerabilities, consistently ranked among the most critical security threats according to the OWASP classification. In response to the increasing complexity of such attacks, contemporary research and engineering approaches have actively adopted machine learning (ML) and deep learning (DL) techniques for the automated detection of malicious SQL queries. Despite their high classification accuracy on static datasets, these models are susceptible to adversarial attacks—intentionally crafted inputs that preserve both the syntactic validity and semantic equivalence of the original queries but disrupt the model's decision logic.**

This paper presents a method for adversarial testing of machine learning models designed for SQL injection detection. The proposed approach is based on the sequential application of mutation operators that maintain the syntactic correctness and semantic equivalence of the input queries. The algorithm is implemented as an iterative process aimed at reducing the model's confidence in identifying a malicious query, while operating under black-box constraints without access to internal parameters or architecture.

The method has been experimentally validated on three model architectures: MobileBERT, LSTM, and a hybrid CNN-LSTM. The evaluation results demonstrate a notable decrease in precision and recall for the injection class after applying adversarial mutations. The proposed framework serves as an effective tool for objective assessment of model robustness against polymorphic SQL injections and provides a foundation for developing more resilient security systems. Due to its universality and reproducibility, the method is also applicable for augmenting training datasets in adversarial learning settings. Furthermore, it holds potential for adaptation to other types of text-based attacks (e.g., XSS) and integration into industrial-grade web application security solutions.

Keywords **– SQL injection, machine learning, adversarial attacks, robustness testing, mutation operators, MobileBERT, LSTM, CNN-LSTM.**

REFERENCES

[1] OWASP Top Ten // OWASP. 2020. URL: https://owasp.org/www-project-top-ten/ (Data obrashhenija: 01.05.2025).

[2] Wang Y., Kosinski M. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images //Journal of personality and social psychology. – 2018. – T. 114. – #. 2. – S. 246.

[3] Kolhe A. K., Adhikari P. A SQL Injection: Internal Investigation of Injection, Detection and Prevention of SQL Injection Attacks //International Journal of Engineering Research & Technology (IJERT). – 2014. – T. 3. – #. 1.

[4] Demetrio L. et al. Waf-a-mole: evading web application firewalls through adversarial machine learning //Proceedings of the 35th Annual ACM Symposium on Applied Computing. – 2020. – S. 1745-1752.

[5] Guan Y. et al. Ssqli: A black-box adversarial attack method for sql injection based on reinforcement learning //Future Internet. – 2023. – T. 15. – #. 4. – S. 133.

[6] Alqhtani M., Alghazzawi D., Alarifi S. Black-Box Adversarial Attacks Against SQL Injection Detection Model //Contemporary Mathematics. – 2024. – S. 5098-5112.

[7] Boekweg K. I. Developing a SQL Injection Exploitation Tool with Natural Language Generation : dis. – Brigham Young University, 2024.

[8] Judova E. A., Laponina O. R. Sravnitel'nyj analiz podhodov k obnaruzheniju SQL-in"ekcij s pomoshh'ju metodov mashinnogo obuchenija //International Journal of Open Information Technologies. – 2023. – T. 11. – #. 6. – S. 175-181.

[9] Alghawazi M., Alghazzawi D., Alarifi S. Detection of sql injection attack using machine learning techniques: a systematic literature review //Journal of Cybersecurity and Privacy. – 2022. – T. 2. – #. 4. – S. 764-777.

[10] Gandhi N. et al. A CNN-BiLSTM based approach for detection of SQL injection attacks //2021 International conference on computational intelligence and knowledge economy (ICCIKE). – IEEE, 2021. – S. 378-383.

[11] Kakisim A. G. A deep learning approach based on multi-view consensus for SQL injection detection //International Journal of Information Security. – 2024. – T. 23. – #. 2. – S. 1541-1556.

[12] Zhang W. et al. Deep Neural Network-Based SQL Injection Detection Method //Security and Communication Networks. – 2022. – T. 2022. – #. 1. – S. 4836289.

[13] Tang P. et al. Detection of SQL injection based on artificial neural network //Knowledge-Based Systems. – 2020. – T. 190. – S. 105528.

[14] Xie X. et al. Sql injection detection for web applications based on elastic-pooling cnn //IEEE Access. – 2019. – T. 7. – S. 151475-151481.

[15] Finlayson S. G. et al. Adversarial attacks on medical machine learning //Science. – 2019. – T. 363. – #. 6433. – S. 1287-1289.

[16] Alatwi H. A., Aldweesh A. Adversarial black-box attacks against network intrusion detection systems: A survey //2021 IEEE World AI IoT Congress (AIIoT). – IEEE, 2021. – S. 0034-0040.

[17] Yan S. et al. A survey of adversarial attack and defense methods for malware classification in cyber security //IEEE Communications Surveys & Tutorials. – 2022. – T. 25. – #. 1. – S. 467-496.

[18] Carlini N., Wagner D. Adversarial examples are not easily detected: Bypassing ten detection methods //Proceedings of the 10th ACM workshop on artificial intelligence and security. – 2017. – S. 3-14

[19] Suhomlin, Vladimir Aleksandrovich. "Sozdanie profilja" Kiberbezopasnost' i iskusstvennyj intellekt" dlja napravlenija podgotovki FIIT na osnove kurrikulumnogo podhoda." Sovremennye informacionnye tehnologii i IT-obrazovanie 17.3 (2021): 724-734.

[20] Korniukhina, Sofia P., and Olga R. Laponina. "Research of the Capabilities of Deep Learning Algorithms to Protection Against Phishing Attacks." International Journal of Open Information Technologies 11.6 (2023): 163-174.

[21] Zubrienko, G. A., and O. R. Laponina. "Data Sampling Techniques for Anomaly Detection in Network Traffic." International Journal of Open Information Technologies 4.10 (2016): 1-8.

[22] O rabotah po cifrovoj jekonomike / V. P. Kuprijanovskij, D. E. Namiot, S. A. Sinjagov, A. P. Dobrynin // Sovremennye informacionnye tehnologii i IT-obrazovanie. – 2016. – T. 12, # 1. – S. 243-249. – EDN XEQRFJ.

[23] Roznichnaja torgovlja v cifrovoj jekonomike / V. P. Kuprijanovskij, S. A. Sinjagov, D. E. Namiot [i dr.] // International Journal of Open Information Technologies. – 2016. – T. 4, # 7. – S. 1-12. – EDN WCMIWN.