https://doi.org/10.7840/kics.2026.51.1.35

패턴 매칭과 머신 러닝을 활용한 2단계 SQL 삽입 공격탐지 방법

어 명 규*, 전 상 훈°

Two-Stage SQL Injection Detection Method Using Pattern Matching and Machine Learning

Myeong-Gyu Eo*, Sanghoon Jeon°

요 약

SQL 삽입 공격은 웹 애플리케이션 보안에 심각한 위협을 주는 대표적인 사이버 공격 기법이다. 기존 탐지 방식은 탐지 속도가 빠르면 정확도가 낮고, 정확도가 높으면 탐지 속도가 느린 구조적 한계를 가진다. 이를 개선하기 위해 1단계 패턴 매칭과 2단계 머신러닝을 결합한 2단계 탐지(Two-stage Detection, TSD) 프레임워크를 제안한다. TSD는 1단계에서 패턴 매칭으로 알려진 공격을 신속히 걸러내고, 2단계에서 머신러닝 모델로 탐지되지 않은 공격을 정밀하게 분석한다. Kaggle SQL Injection Dataset을 활용한 실험 결과, TSD는 Random Forest, Support Vector Machine, Logistic Regression XGBoost 등 모든 모델에서 단일 머신러닝 대비 재현율이 일관되게 증가했고, 탐지 시간도 단축됨을 확인하였다. 본 연구는 재현율 향상과 탐지 시간 단축을 동시에 달성한 실시간 SQL 삽입 공격 탐지의 현실적 대안을 제시한다. 향후에는 온라인 패턴 갱신과 다양한 공격 대응을 위한 데이터셋 확장 연구를 통해 실용성을 더욱 강화할 예정이다.

키워드 : SQL 삽입 공격, 패턴 매칭, 머신 러닝, 침입 탐지, 2단계 SQL공격 탐지

Key Words : SQL Injection, Pattern Matching, Machine Learning, Intrusion Detection, Two-stage SQL Injection Detection

ABSTRACT

SQL injection is a major security threat in web applications. Existing detection methods are limited by a structural trade-off: fast detection comes at the cost of lower accuracy, while higher accuracy results in slower detection. To address this, we propose a two-stage detection (TSD) framework that combines pattern matching in the first stage with machine learning in the second stage. In the TSD framework, known attacks are rapidly filtered through pattern matching, and undetected queries are analyzed in detail using a machine learning model. Experiments using the Kaggle SQL Injection Dataset showed that TSD consistently increased recall across all models (Random Forest, Support Vector Machine, Logistic Regression XGBoost) compared to standalone machine learning, while also reducing detection time. This paper presents a practical solution for real-time SQL injection detection that simultaneously improves recall and reduces detection latency. Future work will focus on enhancing the practicality through online pattern updates and the expansion of datasets to address diverse attack scenarios.

※ This study was supported by the research grant of The University of Suwon in 2025.
◆ First Author : The University of Suwon Department of Information Security, amg8521@naver.com, 학생회원
° Corresponding Author : The University of Suwon Department of Information Security, shjeon@suwon.ac.kr, 정회원
논문번호 : 202503-065-A-RN, Received March 24, 2025; Revised july 24, 2025; Accepted August 12, 2025

Ⅰ. 서 론

디지털 기술의 발전은 전자상거래, 클라우드 컴퓨팅, 금융 서비스 등 다양한 웹 애플리케이션 환경에서 혁신적인 변화를 가져왔다. 이로 인해 데이터 활용성과 접근성이 증가했지만, 동시에 심각한 사이버 위협의 증가라는 문제를 초래했다[1]. SQL 삽입 공격은 웹 애플리케이션에서 가장 빈번하게 발생하는 보안 위협 중 하나로, 데이터베이스를 조작하여 개인정보 유출, 서비스 장애, 데이터 파괴와 같은 심각한 문제를 초래한다[2,3].

SQL 삽입 공격을 효과적으로 탐지하기 위해 다양한 접근 방식이 연구되었으며, 그중 패턴 매칭 기반 탐지 기법은 가장 널리 사용된다. 패턴 매칭은 알려진 SQL 삽입 공격 패턴을 탐지하는 데 효과적이며, 탐지 속도가 빠르고 구조가 단순하다는 장점을 가진다[4][5]. 그러나 고정된 규칙에 의존하기 때문에 기존 패턴에서 조금이라도 변형된 공격이나 알려지지 않은 새로운 공격 유형을 탐지하는 데 한계를 가진다[6,7].

반면, 머신러닝 기반 탐지 기법은 대규모 데이터를 기반으로 정상 및 비정상 패턴을 학습하여 다양한 형태의 공격을 식별할 수 있다. 그러나 정확한 탐지를 위해 다량의 학습 데이터와 전처리 과정이 필요하며, 복잡한 모델 구조로 인해 계산 비용이 높고, 실시간 탐지 환경에는 부적합한 경우가 많다[8,9]. 특히 실시간 탐지 환경에서 높은 정확성을 유지하며 빠르게 탐지 결과를 제공하기에는 한계가 존재한다[10].

본 논문에서는 이러한 문제점을 해결하기 위해 패턴 매칭 기반 탐지의 빠른 탐지 속도와 머신러닝 기반 탐지의 높은 정확성을 결합한 2단계 탐지(Two-stage Detection, TSD) 프레임워크를 제안한다. 제안된 TSD 프레임워크는 1단계에서 패턴 매칭을 이용하여 알려진 공격을 신속히 필터링하고, 2단계에서는 머신러닝 모델을 사용하여 탐지되지 않은 변형된 공격을 정밀하게 분석하는 구조로 구성된다.

본 논문의 구조는 다음과 같다. II장에서는 SQL Injection 공격 탐지를 위한 기존 패턴 매칭 및 머신러닝 기법에 대해 논의하고, 기존 기술들과의 차이점을 확인한다. III장에서는 제안된 2단계 탐지 프레임워크의 설계 및 구현 방안을 설명하고, 데이터 구성 및 전처리 과정에 대해 다룬다. IV장에서는 실험 결과를 통해 2단계 탐지 프레임워크의 성능을 평가한다. 마지막으로, V장에서는 본 연구의 결론과 향후 연구 방향을 논의한다.

Ⅱ. 관련 연구

2.1. 패턴 매칭 기반 SQL 삽입 공격 탐지

패턴 매칭 기반 탐지 기법은 사전에 정의된 공격 패턴과 입력된 SQL 질의문을 비교하여, 기존에 알려진 공격을 탐지하는 방식이다. 이 기법은 정규 표현식이나 해시 기반 탐지 기법을 활용하여 공격을 식별하며, 구조가 단순하고 탐지 속도가 빠르다는 장점이 있다[1]. 예를 들어, SQL 삽입 공격에서 자주 사용되는 특정 키워드나 문법적 특징을 정규 표현식으로 정의하고, 이를 기반으로 입력된 SQL 명령어를 검사함으로써 공격을 탐지할 수 있다[2].

패턴 매칭 기반 기법의 가장 큰 장점은 구현의 단순성과 빠른 탐지 속도이다. 사전 정의된 패턴에만 의존하기 때문에 알고리즘 자체가 복잡하지 않으며, 고속으로 대량의 데이터를 처리할 수 있다[3]. 특히, SQL 삽입 공격과 같은 정형화된 기존 공격 유형을 효과적으로 탐지하는 데 적합하다. 공격자가 자주 사용하는 특수문자, 연산자 및 SQL 키워드를 패턴화함으로써 탐지 성능을 높일 수 있다[4].

하지만 패턴 매칭 기법은 새로운 유형의 공격이나 변형된 공격을 탐지하는 데 구조적인 한계를 가진다. 고정된 규칙에 의존하기 때문에 공격자가 패턴을 변형하거나 우회할 경우 탐지가 어려울 수 있다[5]. 또한, 웹 환경이 점점 복잡해짐에 따라 대규모 데이터 및 복잡한 SQL 구문에 대한 탐지 정확도가 저하될 가능성이 있다[6]. 이러한 한계는 웹 해킹이 진화하는 환경에서 기존 탐지 기법의 취약점으로 작용할 수 있다.

2.2 머신 러닝 기반 SQL 삽입 공격 탐지

머신러닝 기반 탐지 기법은 정상 및 비정상 SQL 패턴을 학습하여, 기존 탐지 기법이 놓치는 공격도 효과적으로 탐지할 수 있도록 설계되었다. 지도 학습뿐만 아니라 비지도 학습도 적용 가능하며, 특히 SVM, Random Forest, 오토인코더와 같은 다양한 모델이 활용된다[7].

머신러닝 기반 기법의 주요 장점은 신규 및 변형 공격에 대한 일반화 능력에 있다. 데이터로부터 직접 비정상 패턴을 학습하므로, 사전에 정의되지 않은 새로운 공격이나 변형된 공격에도 높은 탐지 성능을 보인다[8]. 또한, 패턴 매칭 기법이 놓치는 변형된 공격이나 복잡한 데이터 구조에서도 높은 탐지율을 기대할 수 있다[9]. 예를 들어, 머신러닝 알고리즘은 SQL 질의문 내의 비정상적 구조나 요청 패턴을 분석해 높은 정확도로 공격을 식별한다[10].

그러나 머신러닝 기반 기법은 학습 데이터의 품질과 양에 크게 의존하며, 모델 학습 및 추론 과정에서의 높은 계산 비용이 현실적인 제약 요인으로 작용한다. 실제 웹 환경에서는 이러한 비용 문제와 실시간성 한계로 인해 효율적으로 적용되기 어렵다는 지적이 많다.

이와 같은 한계로 인해, 실제 환경에서 높은 탐지 정확성과 효율적인 처리 속도를 모두 달성하기 위해서는, 두 기법의 장점을 결합하는 하이브리드 탐지 전략이 요구된다. 본 논문에서는 이러한 한계를 극복하기 위해, 두 기법을 결합해 상호 보완하도록 설계된 TSD (Two-Stage Detection) 프레임워크를 제안한다.

Ⅲ. 제안 방법

본 논문에서는 패턴 매칭과 머신러닝을 결합한 2단계 탐지 프레임워크를 제안한다. 먼저, 패턴 매칭 기법을 적용하여 기존에 알려진 공격을 신속하게 탐지한 후, 머신 러닝 모델을 활용하여 패턴 매칭 단계에서 탐지되지 않은 미탐지 입력에 대해 추가 분석을 수행하는 구조를 설계하였다. 이를 통해 기존 탐지 방식이 가지는 한계를 보완하고, 탐지 정확도와 처리 효율성 간의 균형을 향상시키는 전략을 제시한다.

3.1 패턴 매칭

패턴 매칭(Pattern Matching)은 입력된 데이터가 사전에 정의된 특정 패턴과 일치하는지를 확인하여 공격 여부를 판단하는 기법이다. 이 방식은 웹 애플리케이션 방화벽(WAF)과 침입 탐지 시스템(IDS)에서 사용되며, SQL 삽입 공격, XSS, Command 삽입 공격 등의 공격 탐지에 활용된다. 패턴 매칭은 사전 정의된 시그니처와 입력 쿼리를 비교하여 특정 시그니처가 포함되어 있는지를 검사하는 방식으로 동작한다.

예를 들어, SQL Injection 탐지를 위해 “--”, “DROP TABLE”, “’ OR ’1’=’1”, “UNION SELECT” 등의 문자열을 탐지 대상으로 설정하여 사용한다.

그림 1은 패턴 매칭 기법을 활용한 SQL 삽입 공격 탐지 흐름을 나타낸다. 사용자가 입력한 SQL 쿼리는 시스템이 수집한 후, 사전에 정의된 공격 패턴과 비교된다. “--”, “l”DROP TABLE”, “ ’ OR ’1’=’1”, “UNION SELECT” 등의 공격 시그니처가 포함된 경우, 공격으로 판단되어 차단되거나 경고가 발생한다. 반면 정상적인 입력은 별도의 조치 없이 처리된다. 최종적으로 탐지 결과가 반환되며, 공격이 감지된 경우 1, 그렇지 않으면 0을 반환한다. 패턴 매칭 기반 탐지기의 성능은 전체 데이터 중 테스트 데이터(20%)를 활용하여 평가하였다.

3.2 머신 러닝

머신 러닝은 대량의 데이터를 분석하여 잠재된 패턴을 학습하고, 이를 바탕으로 새로운 입력을 분류하거나 예측하는 기술이다. 그림 2는 머신 러닝 기법을 이용한 SQL 삽입 공격 탐지의 전체 흐름을 보여준다. 머신 러닝은 데이터 기반의 복잡한 패턴을 식별할 수 있어, 고정된 규칙에 의존하는 패턴 매칭 기법보다 일반적으로 더 높은 분류 성능을 기대할 수 있다.

본 연구에서는 탐지 성능과 실시간 적용 가능성을 고려하여 Random Forest(RF), Support Vector Machine (SVM), Logistic Regression, XGBoost 네 가지 알고리즘을 선정하였다. 이들 모델은 학습 구조, 계산 복잡도, 해석 가능성 등의 측면에서 서로 다른 특성을 가지며, 다양한 조건에서의 탐지 성능을 비교하는 데 유의미하다.

Random Forest는 다수의 결정 트리를 앙상블하여 예측 정확도를 높이는 모델로, 이상치에 강하고 과적합을 방지하는 데 효과적이다[11]. 무작위 특성 선택과 배깅을 통해 안정적이고 일관된 결과를 도출할 수 있다.

SVM은 고차원 공간에서 최적의 결정 경계를 찾는

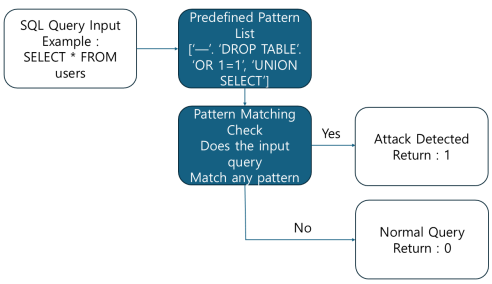

그림 1. 패턴 매칭 기반 SQL 삽입 공격 탐지 흐름도
Fig. 1. Flow Diagram of Pattern Matching-Based SQL Injection Detection

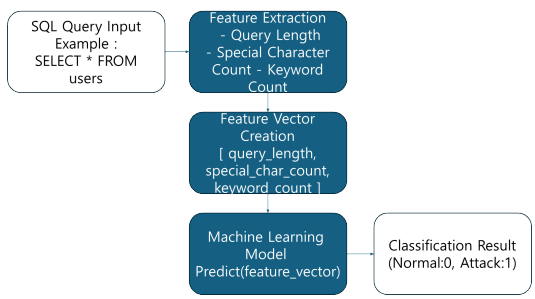

그림 2. 머신러닝 기반 SQL 삽입 공격 탐지 흐름도
Fig. 2. Flow Diagram of Machine Learning-Based SQL Injection Detection

선형 분류기로, 특히 소규모 또는 고차원 데이터에서 뛰어난 성능을 보인다[12]. Logistic Regression은 구조가 단순하고 예측 결과의 해석이 용이한 확률 기반 선형 분류 모델이며, 본 연구에서는 선형 모델 중 가장 기본적인 형태로서 비교 대상으로 포함하였다[13]. XGBoost는 Gradient Boosting 기반의 고성능 앙상블 모델로, 학습 속도와 예측 정확도 측면에서 우수하며 결측값 처리나 불균형 데이터 대응에도 강점을 지닌다[14].

모델 성능 비교를 위한 실험은 전체 SQL 데이터셋을 8:2 비율로 무작위 분할하여 훈련 데이터와 테스트 데이터로 구성하였다. 훈련 데이터에 대해 5-fold 교차 검증을 기반으로 Grid Search를 수행하여 각 모델의 최적 하이퍼파라미터를 도출하고, 이를 바탕으로 최종 모델을 학습한 뒤 테스트 데이터로 성능을 평가하였다.

각 모델에 대해 탐색한 하이퍼파라미터의 범위는 다음과 같다. Random Forest는 n_estimators(50, 100, 150), max_depth(6, 10, 16), min_samples_split(2, 4), min_samples_leaf(1, 3)를 조합하였으며, SVM은 linear 커널을 고정한 상태에서 패널티 파라미터 C를 0.1, 0.5, 1.0, 3.0, 5.0으로 설정하였다. Logistic Regression은 C(0.1 – 5.0)와 solver(lbfgs, liblinear)를 조합하여 실험하였고, XGBoost는 n_estimators(50, 100), max_depth(3, 5, 8), learning_rate(0.01, 0.05, 0.1)를 대상으로 탐색하였다.

3.3 2단계 프레임워크 구조

본 논문에서는 SQL 삽입 공격에 대한 탐지 성능을 향상시키기 위해, 패턴 매칭과 머신 러닝 기법을 결합한 2단계 탐지 프레임워크를 설계하였다. 기존의 시그니처 기반 탐지 방식은 빠른 처리 속도를 제공하지만, 사전에 정의되지 않은 공격에 대해서는 탐지율이 낮다는 한계가 있다. 특히 탐지 누락(False Negative)이 발생할 경우, 보안상 치명적인 결과로 이어질 수 있다. 이러한 한계를 극복하기 위해, 본 연구에서는 탐지 속도를 유지하면서도 재현율(Recall)을 효과적으로 향상시킬 수 있는 구조를 도입하였다.

제안된 프레임워크는 입력된 SQL 쿼리에 대해 1단계에서 패턴 매칭 기법을 적용하여, 이미 알려진 공격 시그니처를 신속하게 탐지한다. 이 단계에서 공격으로 판별된 쿼리는 즉시 차단되며, 이후 단계로 전달되지 않는다. 반면, 1단계에서 탐지되지 않은 쿼리는 2단계 머신 러닝 기반 탐지기로 전달되어 추가적인 분석이 수행된다. 이 구조를 통해 1단계에서 누락된 공격을 2단계에서 정밀하게 판별함으로써, 전체 시스템의 재현율을 실질적으로 향상시킬 수 있다.

TSD 프레임워크의 성능을 평가하기 위해, 기존의 PM 및 ML 기법과 동일하게 전체 데이터를 8:2 비율로 훈련 데이터과 테스트 데이터로 분할 하였다. 분할된 훈련 데이터는 TSD 프레임워크 내 2차 분류기(머신 러닝 탐지기)의 학습에 사용되며, 이때 각 모델의 최적 하이퍼파라미터는 5-fold 교차 검증을 통해 도출하였다. 도출된 최적 하이퍼파라미터를 적용해 2차 분류기를 학습시킨 후, 이를 TSD 프레임워크의 내부 모듈로 구성하고 테스트 데이터를 통해 전체 탐지 성능을 평가하였다.

그림 3은 제안된 TSD 프레임워크의 전체 구조를 나타낸다. 전처리된 입력 데이터는 1단계 시그니처 기반 탐지기와 2단계 머신 러닝 탐지기에 순차적으로 적용되며, 두 단계는 상호보완적인 방식으로 작동한다. 1단계에서는 비교적 단순한 패턴 매칭을 통해 연산 자원을 절약하고, 2단계에서는 복잡한 특징 기반 분석을 수행하여 탐지 정확도를 향상시키는 구조이다.

알고리즘 1은 제안된 2단계 탐지 절차를 구체적으로 기술한 것이다. 입력된 SQL 쿼리는 먼저 시그니처 데이터베이스와 비교되어 공격 여부를 판단한다. 시그니처와 일치할 경우, 탐지를 종료하고 '1'을 반환하며, 알려진 공격을 빠르게 차단할 수 있다. 반면 일치하지 않는 쿼리는 머신 러닝 탐지 단계로 전달되어, 쿼리 길이, 특수문자 개수, SQL 키워드 수 등의 특징을 기반으로 공격 여부를 재판단한다.

이러한 2단계 구조는 1단계에서 걸러지지 않은 입력에 대해서만 머신 러닝 기반의 정밀 분석을 수행함으로써, 탐지 속도를 유지하면서도 탐지 누락을 최소화할 수 있도록 설계되었다.

재현율이 중요한 침입 탐지 시스템(IDS), 웹 방화벽 등에서는 공격을 놓치지 않는 것이 핵심이며, 본 구조는

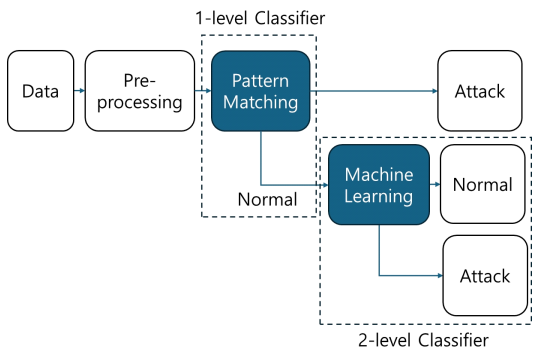

그림 3. 제안하는 2단계 SQL 삽입 공격 탐지 프레임워크 흐름도
Fig. 3. Flow Diagram of the Proposed Two-Stage SQL Injection Detection Framework

알고리즘 1: 이단계 탐지 프레임워크
Algorithm 1: Two-Stage Detection Framework

```
Input: SQL Query query, Trained Model model
Output: Classification Label (0: Normal, 1: Attack)

attack_detected ← pattern_matching(query, pattern_list)
if attack_detected == 1 then
    return 1
end if    // Pattern Matching

// Feature Extraction
query_length ← len(query)
special_char_count ← 0
for each character c in query do
    if c is in { '--', ';', '"', "'", '/*', '*/' } then
        special_char_count← special_char_count+1
    end if
end for

keyword_count ← 0
for each keyword in { 'SELECT', 'DROP', 'UNION',
'INSERT', 'DELETE', 'UPDATE' } do
    if keyword is found in query then
        keyword_count ← keyword_count + 1
    end if
end for

feature_vector ← [query_length, special_char_count,
keyword_count]

// Machine Learning Prediction
prediction ← model.predict(feature_vector)

return prediction
```

이를 효과적으로 보완하는 방식이다. 제안된 TSD 프레임워크는 빠른 응답성과 높은 탐지 정확도를 동시에 달성할 수 있는 현실적인 보안 대안으로 활용될 수 있다.

3.4 데이터 세트

본 논문에서는 Kaggle에서 제공하는 SQL Injection 데이터 세트를 사용하였다(Kaggle SQL Injection Dataset). 해당 데이터 세트는 정상 SQL 쿼리와 다양한 형태의 SQL 삽입 공격 쿼리를 포함하고 있으며, 탐지 기법의 성능 평가를 위한 레이블 정보를 제공한다.

3.4.1 데이터 세트 구성

본 논문에서 사용된 데이터 세트는 Kaggle의 Sajid576 SQL Injection Dataset을 기반으로 구성되었다. 총 31,469개의 SQL 쿼리로 이루어져 있으며, 이 중 19,537개는 정상 쿼리, 11,932개는 공격 쿼리로 분류된다. 각 샘플은 Label 속성을 통해 이진 분류 라벨을 제공하며, 정상(0) 또는 공격(1)으로 구분된다. 본 데이터 세트는 총 31,469개의 SQL 쿼리로 구성되어 있으며, 이 중 19,537개는 정상 SQL 쿼리, 11,932개는 SQL 삽입 공격 쿼리로 분류된다. 데이터는 지도 학습을 위한 레이블을 포함하고 있어 머신러닝 모델이 정상쿼리와 비정상 쿼리를 효과적으로 학습하고 평가하는 데 활용될 수 있다. 본 연구에서는 이 데이터 세트를 기반으로 패턴 매칭 기법과 머신 러닝 기법을 비교하고, 이를 통합한 2단계 탐지 프레임워크의 탐지 성능을 평가하였다.

3.4.2 주요 탐지 특성 및 역할 분석

본 논문에서 제안한 SQL 삽입 공격 탐지 모델은 세 가지 주요 특징(feature)에 기반하여 정상 쿼리와 비정상 쿼리를 분류한다.

이 세 가지 특징은 쿼리 길이(query_length), 특수문자 개수(special_char_count), 그리고 SQL 명령어 개수(keyword_count)이며, 각 속성은 표 1에 정의되어 있다.

먼저, 쿼리 길이는 쿼리 전체 문자열의 길이를 나타내는 값으로, 정상 쿼리는 대체로 구조가 단순하고 짧은 반면, 비정상 쿼리는 다양한 우회 방식과 복잡한 논리 구조를 포함하므로 쿼리 길이가 더 길게 나타나는 경향이 있다. 이와 같은 특성은 SQL 삽입 공격 탐지에서 중요한 분류 기준이 된다.

둘째, 특수문자 개수는 SQL 쿼리 내에 등장하는 주요 특수문자(--, ;, ", ', /*, */)의 출현 빈도를 의미한다. SQL 삽입 공격에서는 쿼리의 논리 흐름을 변경하거나 추가 명령 실행을 유도하기 위해 다양한 특수문자가 빈번하게 사용된다. 따라서, 쿼리 내 특수문자의 개수가 많을수록 비정상 쿼리일 가능성이 높다고 볼 수 있다.

셋째, SQL 명령어 개수는 SELECT, DROP, UNION, INSERT, DELETE, UPDATE 등 위험성이 높은 SQL 명령어의 출현 빈도를 의미한다. 공격자는 여러 SQL 명령어를 반복적으로 활용하여 데이터 탈취, 변조, 삭제 등의 다양한 악의적 목적을 달성할 수 있다. 이 때문에, 해당 명령어의 출현 빈도 역시 비정상 쿼리 탐지에서 효과적인 기준이 된다.

이들 세 가지 feature는 Algorithm 1에서 쿼리별로 추출되며, 탐지 모델의 입력값으로 사용된다. 표 1은 각 특징의 정의와 이들이 SQL 삽입 공격 탐지에서 수행하는 역할을 요약하여 보여준다.

표 1. SQL 삽입 공격 탐지에 사용된 주요 특징과 설명
Table 1. Major Features and Descriptions for SQL Injection Detection

Attribute Name	Description
Query Length	Length of the SQL query string
Special Char Count	Number of --, ;, ", ', /*, */ in the query
SQL Keyword Count	Occurrences of SELECT, DROP, UNION, etc.

Ⅳ. 성능 평가

본 장에서는 패턴 매칭, 머신러닝, 그리고 제안한 2단계 탐지 프레임워크의 탐지 성능을 비교 및 분석한다.

모든 탐지 기법은 동일한 데이터셋에서 8:2로 분할된 학습 데이터와 테스트 데이터를 기반으로 성능을 비교하였으며, 머신러닝 기반 탐지기와 2단계 탐지기의 경우 5-fold 교차 검증을 통해 하이퍼파라미터를 최적화한 후 훈련 모델을 생성하였다. 생성된 모델은 테스트 데이터에 적용되어 최종 탐지 성능을 측정하고 세가지 SQL삽입 공격 탐지 방식의 성능을 비교 및 평가한다.

4.1 환경 설정

본 논문의 실험은 Jupyter Notebook 환경에서 Python 3.9 기반으로 수행되었으며, 머신러닝 모델 구현과 분석을 위해 Scikit-learn, XGBoost, Pandas 등의 주요 라이브러리를 활용하였다. 이는 머신러닝 기반 탐지 모델의 구현, 학습, 평가를 안정적으로 수행할 수 있는 기본적인 분석 환경을 제공한다.

4.2 성능 평가 지표

본 논문에서는 2단계 탐지 프레임워크의 성능을 평가하기 위해 다음과 같은 주요 평가지표를 사용하였다.

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

수식 (1)은 정확도(Accuracy)에 대한 지표로, 전체 샘플 중 올바르게 분류된 비율을 나타낸다.

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

수식 (2)는 정밀도(Precision)에 대한 지표로, 탐지된 공격 중 실제로 공격인 비율을 측정한다.

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

수식 (3)은 재현율(Recall)에 대한 지표로, 실제 공격을 탐지한 비율을 나타내며, 탐지 민감도를 측정한다.

$$F1\text{-}score = 2 \times \frac{Percision + Recall}{\mathrm{Precision} \times Recall} \quad (4)$$

수식 (4)는 F1 점수(F1-score)에 대한 지표로, 정밀도와 재현율 간의 균형을 측정하여 모델의 전반적인 성능을 평가한다.

$$Detection\ Time = \frac{1}{n}\sum_{i=1}^{n} t_i \quad (5)$$

수식 (5)는 모델이 벡터화된 입력에 대해 탐지 판단을 수행하는 데 소요된 평균 시간(ms)으로, 추론 속도 비교를 위한 지표이다. 단, 쿼리에서 특징을 추출하는 전처리 과정은 제외되며, 모델의 순수 예측 시간만을 측정하였다.

4.3 탐지 기법별 성능 지표 비교 및 분석

본 절에서는 세 가지 탐지 방식인 PM, ML, 그리고 제안하는 TSD 프레임워크의 성능을 정확도, 정밀도, 재현율, F1-score의 네 가지 분류 지표와 탐지 시간(Detection Time)을 기준으로 정량적으로 비교 및 분석한다. 이때, 탐지 시간 측정은 테스트 데이터셋 전체에 대해 각 탐지 방법별로 100회 반복 실행하여 평균과 표준편차(Mean ± SD)로 산정하였다. 측정 과정에는 Python의 time.perf_counter() 함수를 사용해 실행 구간을 정밀 계측하였으며, 각 탐지 방식(PM, ML, TSD)에 맞게 동일한 환경에서 측정을 수행하였다.

표 2는 Accuracy, Precision, Recall, F1-score, and detection time 등 탐지 성능의 전반적인 정량 수치를 요약하고 있으며, 각 탐지 기법의 성능 우열을 정리하는 기초 자료로 사용된다. PM 기법은 탐지 시간 3.64ms로 세 번째로 빠른 속도를 기록하였으나, 정확도 81.06%, 정밀도 69.65%, 재현율 86.03%, F1-score 76.98%로 탐지 정확성 측면에서는 낮은 값을 보였다. 이는 고정 패턴 기반 탐지의 구조적 한계로 인해 변형된 공격 탐지에 취약함을 시사한다. ML 기법은 특히 Random Forest 기반에서 정확도 95.10%, 정밀도 94.54%, 재현율 92.00%, F1-score 93.25%로 가장 뛰어난 성능을 보였고, XGBoost 모델도 유사한 수준의 결과를 기록하였다. 다만, SVM의 경우 탐지 시간이 583.01ms에 이르는 등 실시간 적용에는 부담이 존재한다. TSD 기법은 PM의 속도와 ML의 정확성을 결합하여, 탐지 속도와 성능 간의 균형을 실현하였다. 예를 들어, TSD_RF는 탐지 시간 21.73ms, 재현율 94.11%로 균형 잡힌 결과를 도출하였다. 한편, TSD는 1단계 PM 결과에 의존하기 때문에, ML에 비해 Accuracy와 F1-score가 낮게 났다. 이는 1단계에서 잘못 분류된 쿼리가 2단계 분석에서 제외되기 때문이다. 본 연구는 PM 개선 없이 탐지 구조 설계에 중점을 두었으며, 향후 PM 성능을 높이면 TSD

표 2. 테스트 데이터의 SQL 삽입 공격 탐지 성능 비교 결과
Table 2. Comparison of SQL Injection Detection Performance on the Test Dataset

No.	Detection Method	Detection Performance				Detection Time (ms)
		Accuracy	Precision	Recall	F1-score	
1	PM	81.06%	69.65%	**86.03%**	76.98%	**3.64 ± 0.30**
2	ML-RF	95.10%	94.54%	**92.00%**	93.25%	**35.71 ± 0.67**
3	TSD-RF	83.46%	70.67%	**94.11%**	80.72%	**21.73 ± 0.74**
4	ML-SVM	76.94%	82.29%	**47.58%**	60.30%	**583.01 ± 7.37**
5	TSD-SVM	81.21%	68.23%	**91.61%**	78.21%	**322.23 ± 4.12**
6	ML-LR	76.16%	76.14%	**51.32%**	61.31%	**0.18 ± 0.02**
7	TSD-LR	81.21%	68.18%	**91.78%**	78.24%	**3.89 ± 0.26**
8	ML-XGBoost	95.00%	94.40%	**91.87%**	93.12%	**1.73 ± 0.43**
9	TSD-XGBoost	83.44%	70.69%	**93.98%**	80.69%	**6.89 ± 0.46**

PM:Pattern Matching, ML-:Machine Learning-based, TSD-:Two-stage SQL Injection Detection-based,
Detection Time measured with 100 iterations, test set size = 6184

전체 성능도 향상될 여지가 있다.

그림 4는 이러한 표 2의 수치 중에서도, 본 논문에서 핵심적으로 강조하는 두 지표인 재현율과 탐지 시간의 관계를 시각적으로 보여준다.

재현율은 실제 공격을 놓치지 않고 탐지하는 능력을 나타내며, 보안 시스템에서 가장 중요한 지표로 간주된다. 반면, 탐지 시간은 실시간 응답성과 직결되며, 지연 없는 처리 성능을 평가하는 핵심 기준이다.

Random Forest, SVM과 같은 복잡한 ML 모델은 높은 탐지 성능을 보이지만 탐지 시간이 길게 소요되며, 반대로 모델 구조가 단순한 Logistic Regression이나 멀티스레딩이 가능한 XGBoost는 상대적으로 짧은 탐지 시간을 나타내는 것을 확인할 수 있다.

TSD 구조를 적용한 경우, 복잡한 모델(RF, SVM)은 탐지 시간이 유의미하게 단축된 반면, LR과 XGBoost는 오히려 시간이 증가하는 경향을 보였다. 이는 PM 처리 시간이 ML 예측 시간보다 작을 경우 TSD가 상대적으로 느려질 수 있음을 의미하며, TSD 구조는 연산 비용이 큰 ML 모델일수록 탐지 시간 절감 효과가 크다는 점을 보여준다. 특히, TSD-RF는 재현율 94.11%, 탐지 시간 21.73ms를 기록하며 가장 높은 탐지 성능과 실시간성의 균형을 달성한 모델로 확인되었다. 이는 실시간 웹 보안 환경에서도 충분히 적용 가능한 수준의 처리 속도임을 확인할 수 있다.

그림 5는 각 탐지 모델의 Confusion Matrix를 통해 오탐(False Positive, FP)과 누락(False Negative, FN)

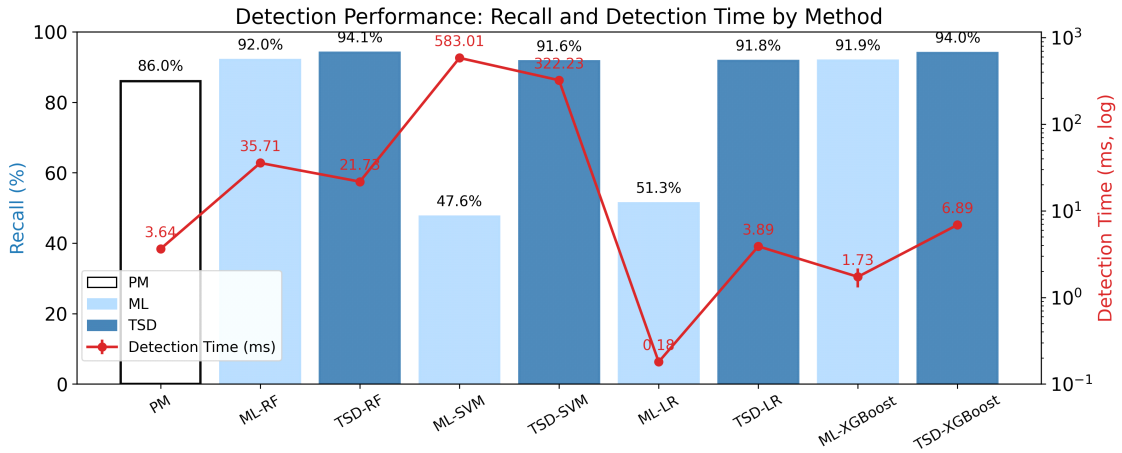

그림 4. 탐지 기법별 재현율과 탐지 시간 비교
Fig. 4. Comparison of Recall and Detection Time by Detection Method

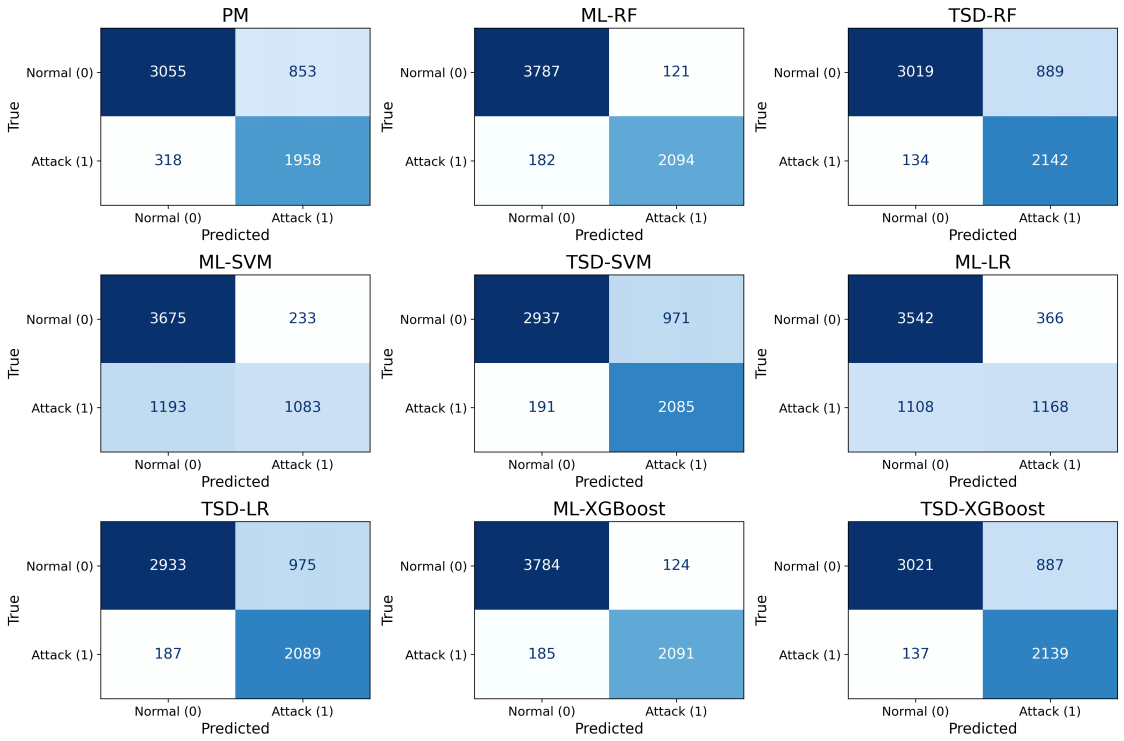

그림 5. 탐지 기법별 혼동 행렬 비교
Fig. 5. Comparison of Confusion Matrices by Detection Method

의 분포를 시각적으로 비교한 결과이다.

PM 기법은 FP(853)가 상대적으로 적고 FN(318)은 높은 편으로, 정형화된 공격에는 효과적이나 변형된 공격 탐지에는 취약한 구조임을 보여준다. ML 기법은 전반적으로 FN을 크게 줄였으며(예: ML-RF의 FN은 182), 특히 XGBoost와 RF는 FN과 FP 모두에서 안정적인 성능을 보였다. 그러나 일부 모델(예: ML-SVM)은 FP(233)와 FN(1,193)이 모두 높은 편으로 분류 성능에 한계가 있음을 보였다.

정리하면, PM은 속도 면에서 강점을 가지나 탐지 정확성에는 한계가 존재한다. ML 기법은 높은 탐지 정확성을 보이지만 실시간 처리에는 부담이 존재한다. 제안하는 TSD 기법은 PM과 ML의 장점을 결합하여 재현율과 탐지 시간이라는 상충 지표 간 균형을 효과적으로 달성하며, 실용적인 대안으로서의 가능성을 실험을 통해 확인하였다.

V. 결 론

SQL 삽입 공격은 웹 애플리케이션 보안에 있어 여전히 주요한 위협으로, 탐지 속도와 정확도 간의 trade-off는 기존 탐지 기법의 구조적 한계로 지적되어 왔다. 본 논문에서는 이러한 한계를 극복하기 위해, 패턴 매칭 기반의 빠른 선별 기능과 머신러닝 기반의 정밀 탐지 기능을 결합한 2단계 탐지 프레임워크(TSD, Two-Stage Detection)를 제안하였다.

제안된 TSD 프레임워크는 1단계에서 알려진 공격 패턴을 신속히 필터링하고, 2단계에서 ML을 통해 변형된 공격을 탐지하는 구조로 설계되었다. 이러한 구조적 설계는 단일 탐지 기법의 약점을 상호 보완함으로써, 실시간성 유지와 정밀 탐지간의 균형을 효과적으로 달성할 수 있도록 한다.

실험 결과, 제안된 TSD 기법은 단일 ML 모델 대비 FN과 FP를 모두 효과적으로 억제하였으며, 특히 재현율 측면에서 우수한 성능을 보였다. 탐지 속도 측면에서는, 머신러닝으로 처리하는 기존 구조에 비해 1단계에서 알려진 공격을 선별적으로 필터링함으로써 연산 부담을 줄였으며, 결과적으로 실시간 탐지에 적합한 성능을 확보하였다.

본 논문의 의의는 탐지 정확성과 연산 효율성을 동시에 확보한 실용적 구조에 있으며, 정밀 탐지와 실시간 대응이 모두 가능한 프레임워크를 제시했다는 점에 있

다. 다만, 정확도나 정밀도는 일부 ML 모델에 비해 다소 낮았는데, 이는 TSD 구조상 1단계 패턴 매칭 결과가 전체 탐지 흐름에 직접적인 영향을 미치기 때문이다.

향후에는 더 정교한 시그니처 기반 탐지(PM) 기법을 적용하고, 고성능 딥러닝 모델과의 결합을 통해 TSD 프레임워크의 전반적인 탐지 성능을 향상 시키는 방향으로 연구를 확장할 계획이다.

또한, 실제 웹 환경에서 발생할 수 있는 다양한 SQL 삽입 공격 시나리오(예: 인증 우회, 데이터 유출, Blind SQLi, UNION 기반 공격 등)를 체계적으로 수집 및 분석하고, 이를 바탕으로 데이터셋 구성 방식을 세분화하며, 각 시나리오별 탐지 대상 및 성능을 구체적으로 정의할 계획이다. 이를 통해 탐지 기법의 실효성을 높이고, 논문의 실무적 활용성과 독자 이해도를 한층 강화할 예정이다.

References

[1] S.-H. Min, H.-J. Yu, M.-J. Lim, and J.-M. Kim, “Detecting SQL injection logs leveraging ELK stack,” in *Proc. KICS Int. Conf. Commun.*, pp. 337-340, Daejeon, Korea, Oct. 2022. (https://doi.org/10.6109/CFKO.2022.26.2.337)

[2] Y. Park and D. Lee, “Development of SQL injection vulnerability advance diagnosis system,” in *Proc. KIIT Summer Conf.*, pp. 471-472, Korea, Jun. 2023.

[3] I.-Y. Lee, J.-I. Cho, K.-H. Cho, and J.-S. Moon, “A method for SQL injection attack detection using the removal of SQL query attribute values,” *J. Inf. Secur. Appli.*, vol. 18, no. 5, pp. 135-148, Oct. 2008. (https://doi.org/10.13089/JKIISC.2008.18.5.135)

[4] S.-Y. Shin, S.-W. Lee, and H.-C. Lee, “Case of security coding guide,” in *Proc. Korean Computer and Inf. Sci. Summer Conf.*, pp. 77-78, Kunsan National University, Korea, Jul. 2015.

[5] M.-S. Ha, J.-I. Namgung, and S.-H. Park, “Counter measures by using execution plan analysis against SQL injection attacks,” *J. IEIE*, vol. 53, no. 2, pp. 76-85, Feb. 2016. (https://doi.org/10.5573/ieie.2016.53.2.076)

[6] Y. Kim and J. Lee, “Development of a malicious URL machine learning detection model reflecting the main feature of URLs,” *J. KIICE*, vol. 26, no. 12, pp. 1786-1793, Dec. 2022. (https://doi.org/10.6109/jkiice.2022.26.12.1786)

[7] Y. Seo, M. Kim, S. Park, and S. Lee, “Web application attack detection scheme using convolutional neural networks,” *J. KIISE*, vol. 45, no. 7, pp. 744-754, Jul. 2018. (https://doi.org/10.5626/JOK.2018.45.7.744)

[8] J.-H. Im, J. Kim, G.-W. Kim, and J.-S. Yu, “Classification of true and false positives in IDPS detection data using deep learning,” *J. KIISC*, vol. 29, no. 3, pp. 22-28, Jun. 2019.

[9] H.-G. Rouw and G.-Y. Kim, “A study on detection method of web attack using machine learning,” *J. KICS*, vol. 45, no. 9, pp. 1642-1655, Sep. 2020. (https://doi.org/10.7840/kics.2020.45.9.1642)

[10] W. Jin and H. Oh, “A BERT-based deep learning approach for vulnerability detection,” *J. Commun. and Netw.*, vol. 32, no. 6, pp. 1139-1140, Dec. 2022. (https://doi.org/10.13089/JKIISC.2022.32.6.1139)

[11] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5-32, Jan. 2001. (https://doi.org/10.1023/A:1010933404324)

[12] H. Kothakapu, D. Nallamasa, and A. Mothikar, “SQL injection attack detection using logistic regression and TF IDF vectorization,” *SSRN Electronic J.*, pp. 1-8, Feb. 2025. (https://doi.org/10.2139/ssrn.5068429)

[13] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed., Hoboken, NJ, USA: John Wiley & Sons, 2013.

[14] T. Chen and C. Guestrin, “XGBoost: A salable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 785-794, San Francisco, CA, USA, Aug. 2016. (https://doi.org/10.1145/2939672.2939785)

어 명 규 (Myeong-Gyu Eo)

2021년 3월~현재 : 수원대학교 정보보호학과 학사과정

<관심분야> 정보보호

전 상 훈 (Sanghoon Jeon)

2012년 2월 : 경북대학교 IT대학 심화 전자공학 공학사

2014년 2월 : 대구경북과학기술원 정보통신융합공학전공 공학석사

2020년 8월 : 대구경북과학기술원 정보통신융합전공 공학박사

2020년 3월~2020년 8월 : 한양대학교 산학협력단 선임연구원

2020년 9월~2022년 9월 : 한양대학교 의과대학 응급의학과 포닥연구원

2022년 10월~2023년 9월 : 한양대학교 의과대학 응급의학과 연구조교수

2023년 10월~현재 : 수원대학교 지능형SW융합대학 정보보호학과 조교수

<관심분야>웨어러블컴퓨팅, 의료인공지능, CPS보안

[ORCID:0000-0001-5636-7555]