Pusan National University
Computer Science and Engineering
Technical Report 2025-09

모바일 엔드 포인트 행동 분석 기반 Policy Engine 구현

권태현
구현서
이승원
지도교수 최윤호

목 차

1. 서론

1.1. 연구 배경 및 필요성

최근 정보 보안 환경은 점차 제로 트러스트 네트워크 액세스(Zero Trust Network Access, ZTNA) 모델을 요구하고 있다. 전통적인 네트워크 보안은 사용자와 장치를 한 번 인증한 후 지속적으로 신뢰하는 방식에 머물렀으나, 내부자 위협, 계정 탈취, 모바일 단말 확산 등 새로운 공격 벡터 앞에서는 더 이상 효과적이지 않다. 이에 따라, 절대 신뢰하지 말고, 항상 검증하라(Never Trust, Always Verify)는 원칙을 실현하는 기술이 필요하다

특히 스마트폰을 포함한 모바일 디바이스는 사용자의 물리적·행동적 특성을 그대로 반영하는 로그 데이터를 생성한다. 터치 패턴, 센서 움직임, 네트워크 환경은 각각 사용자의 고유한 사용 습관을 담고 있으며, 이를 활용하면 정적 인증 수단만으로는 포착하기 어려운 이상 행위를 탐지할 수 있다. 따라서, 사용자 및 엔티티 행동 분석(UEBA, User and Entity Behavior Analytics)을 기반으로 한 동적 접근 제어 기술이 보안 패러다임의 핵심으로 부상하고 있다.

1.2 기술 동향 및 문제 인식

User and Entity Behavior Analytics (UEBA), Zero Trust Network Access (ZTNA), Behavioral Biometrics 등의 최신 보안 트렌드는 단순 인증을 넘어 지속적인 사용자 행동의 이상 탐지와 이에 기반한 실시간 정책 적용을 강조하고 있다. 이에 따라, 단말기에서 발생하는 다양한 로그(예: 터치, 센서, 네트워크)를 수집하고, 이상 여부를 판단하여 네트워크 제어를 수행하는 동적 NAC 시스템의 개발이 요구된다.

특히 모바일 디바이스에서는 기존의 패턴 기반 보안 접근이 효과적으로 작동하기 어려운 경우가 많으며, 이에 대한 대응으로 에이전트 기반 실시간 행동 로그 수집과 딥러닝/머신러닝 기반 이상 탐지, 그리고 이상 탐지 결과에 따른 정책 기반 제어 및 사용자 인터페이스 제공이 필수적으로 요구된다.

1.3 연구 목표

본 연구의 목표는 모바일 모니터링 에이전트를 활용하여 사용자 행동 로그를 수집·분석하고, 이상징후를 탐지해 실시간으로 네트워크 접근을 동적으로 제어하는 NAC 시스템을 개발하는 것이다. 구체적인 세부 목표는 다음과 같이 설정하였다.

1. 에이전트 기반 로그 수집
   - 안드로이드 단말에서 발생하는 Touch(Drag, Pressure), Sensor(Accelerometer, Gyroscope), Network(GPS, 접속 유형) 로그를 실시간으로 수집한다.
   - 로그는 JSON형식으로 정규화되어 백엔드 서버로 전달되며, 이후 머신러닝 서버와 연동된다.
2. 머신러닝 기반 이상 행위 탐지
   - Isolation Forest와 LSTM Autoencoder를 결합한 하이브리드 모델을 통해 개별 이벤트의 이상 여부와 시계열 패턴의 변칙성을 동시에 탐지한다.
   - Sensor와 Touch로그는 ML 서버에서 학습 및 추론을 수행하며, Network 로그는 GPS 좌표 기반의 규칙(rule-based) 방식으로 처리한다.
   - Hybrid 모드를 통해 단일 모델 대비 안정성과 탐지율을 향상시킨다.
3. 백엔드 및 정책 엔진 구현
   - Django REST Framework 기반 백엔드 서버는 로그 수집(Behavior App)과 이상 결과 관리를 분리하여 설계되었다.
   - FastAPI 기반 머신러닝 서버와 REST API를 연동하여, anomaly 결과를 데이터베이스에 저장하고 대시보드 및 안드로이드 앱에서 활용할 수 있도록 제공한다.
4. 대시보드 및 사용자 피드백
   - React 기반 대시보드는 실시간 리스크 게이지, 최근 로그 차트, 통계 분석(Stats)기능을 제공하여 관리자가 시스템 전반의 보안 상태를 직관적으로 파악할 수 있도록 한다.
   - 안드로이드 클라이언트는 이상 탐지 시 사용자에게 네트워크 차단, 비행기 모드 전환, 무시 등의 선택지를 제공하거나, 센서/터치 이상 탐지 시 잠금화면으로 이동 시키는 방식을 통해 즉각적인 보안 조치를 취한다.

2. 연구 배경

2.1 NAC와 ZTNA 보안 패러다임의 변화

기존의 NAC(Network Access Control)는 네트워크에 접속하는 사용자와 기기의 신원을 확인하고, 사전에 정의된 정책에 따라 접근을 허용하거나 차단하는 정적 방식이 주를 이루었다. 이러한 기존 NAC는 주로 네트워크 진입 시점에서의 일회성 검증에 의존하여, 한 번 인증이 완료되면 세션 종료까지 지속적으로 신뢰하는 방식이었다.

그러나 클라우드 서비스, 원격 근무, 모바일 단말기의 보편화로 인해 경계 기반 보안 모델은 한계에 직면하게 되었다. 특히 내부자 위협이나 세션 하이재킹과 같은 공격에서는 초기 인증 이후의 행동 변화를 탐지하기 어렵다는 문제가 드러났다.

이에 따라 Zero Trust Network Access(ZTNA)가 새로운 보안 패러다임으로 부상하였다. ZTNA는 "절대 신뢰하지 말고 항상 검증하라(Never Trust, Always Verify)"라는 원칙을 바탕으로, 사용자의 신원뿐만 아니라 행동, 맥락, 위치 등을 종합적으로 평가하여 동적으로 접근 권한을 부여한다. 기존 NAC와 달리 ZTNA는 세션 전반에 걸쳐 지속적인 신뢰 검증을 수행하며, 실시간으로 위험도를 평가하여 접근 권한을 조정한다.

2.2 사용자 및 엔티티 행동 분석(UEBA)

UEBA(User and Entity Behavior Analytics)는 사용자와 기기의 정상적인 행동 패턴을 학습하고, 그 기준에서 벗어나는 행동을 이상으로 탐지하는 기법이다. 기존 보안 방식은 알려진 공격 유형만 탐지할 수 있는 한계가 있었으나, UEBA는 정상 행동 패턴의 통계적 기준선을 설정하여, 제로데이 공격이나 내부자 위협과 같이 기존 시그니처 기반 탐지로는 발견하기 어려운 알려지지 않은 위협도 탐지할 수 있다.

특히 스마트폰 환경에서 수집 가능한 센서(가속도계, 자이로스코프), 터치(압력, 드래그), 네트워크 로그는 사용자의 무의식적 행동 특성을 반영하기 때문에, UEBA 기반 이상 탐지 모델을 학습시키기에 적합하다.

2.3 이상 탐지 기법과 한계

이상 탐지에는 다양한 기계학습·딥러닝 접근법이 활용되고 있다. 본 프로젝트에서는 Isolation Forest와 LSTM을 채택하였다.

- Isolation Forest는 트리 기반 비지도 학습 모델로, 분할 과정에서 소수의 외곽 데이터가 짧은 경로 길이로 구분되는 특성을 활용해 이상치를 탐지한다. 학습속

도가 빠르고 경량화하기 좋지만, 시계열 패턴과 같은 맥락 정보는 반영하기 어렵다.

- LSTM Auto-Encoder는 순차 데이터를 인코딩·복원하면서
  재구성 오차(Reconstruction Error)를 계산하여 이상 여부를 판단한다. 이는 행동 로그와 같은 시계열 데이터에서 사용자의 과거 행동 패턴을 기억하여 현재 행동의 정상/이상 여부를 판단할 수 있다는 장점이 있으나, 충분한 데이터가 확보되지 않으면 학습이 지연되거나 성능이 불안정해지는 문제가 있다.

이를 보완하기 위해 본 연구에서는 Isolation Forest와 LSTM을 결합한 하이브리드 구조를 설계하였다. 단일 모델보다 안정적이고 효과적인 이상 탐지를 수행할 수 있으며, 주기적인 재학습을 통해 장기 실행 환경에서도 성능을 유지할 수 있음을 확인하였다.

2.4 기존 NAC 연구 및 한계

선행 연구에서는 주로 네트워크 로그나 인증 절차를 기반으로 한 NAC 기술이 많았으나, 이는 사용자의 실제 행동까지는 반영하지 못한다는 한계가 있다. 또한 일부 연구에서는 기기의 터치나 센서 데이터를 활용한 사용자 인증 기법이 제안되었으나, 대부분 단일 인증 혹은 사후 분석 수준에 머물렀다.

따라서 **실시간 로그 수집부터 이상 탐지, 정책 제어, 시각화까지 일련의 보안 프로세스**를 구현한 사례는 드물다.

2.5 본 연구의 차별성

본 프로젝트는 다음과 같은 차별성을 가진다.

1. 에이전트 기반 실시간 로그 수집: 안드로이드 단말에서 Touch, Sensor, Network 데이터를 실시간으로 수집하여 백엔드와 연동.

2. 하이브리드 이상 탐지 모델: iForest와 LSTM을 결합해 개별 이벤트와 시계열 맥락을 동시에 반영.

3. 정책 엔진과 연동된 동적 제어: 이상 탐지 결과를 즉시 반영하여 네트워크 차단, 잠금화면 이동, 사용자 피드백 창 제공 등 보안 대응 수행

4. 대시보드 시각화: 관리자용 웹 대시보드를 통해 복잡한 anomaly 탐지 결과를 직관적으로 제공

이로써 본 연구는 모바일 환경에서의 행동 기반 보안 위협 대응 시스템을 구현하였으며,

기존 NAC 시스템의 한계를 보완하고 ZTNA·UEBA 원칙을 모바일 환경에 적용해보았다.

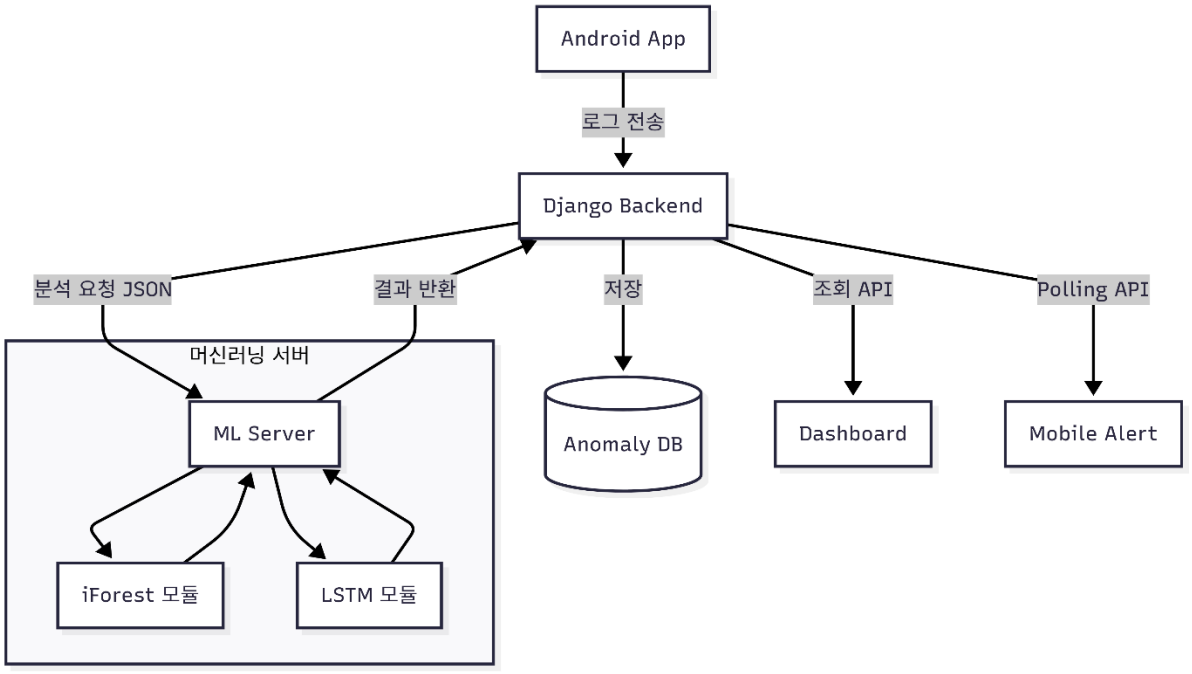
3. 연구 내용

3.1 시스템 전체 구조

본 연구는 모바일 에이전트 기반 데이터 수집부터 실시간 이상 탐지 및 대응까지 이어지는 종단간(End-to-End) 파이프라인 구조로 설계되었다 (그림 3.1 참조).

- 모바일 에이전트 : 사용자의 터치(드래그/압력), 센서(가속도계/자이로스코프), 네트워크(GPS, 접속 유형) 로그를 실시간으로 수집한다.
- 백엔드 서버 : Django 기반으로 구현되어 로그 저장 및 전처리를 수행하고, 머신러닝 서버로 데이터를 전달하며, 탐지 결과를 Anomaly DB에 저장한다.
- 머신러닝 서버 : FastAPI 기반으로 구축되어 Isolation Forest와 LSTM Auto-Encoder를 결합한 하이브리드 모델을 통해 실시간 이상 탐지를 수행한다.
- 모니터링 및 대응 : React 기반 대시보드를 통해 관리자에게 탐지 결과를 시각화하여 제공하며, 안드로이드 앱은 사용자에게 이상 상황별 보안 대응 옵션을 제시한다.

[그림 3.1] 시스템 전체 아키텍처



3.2 로그 수집 에이전트

본 연구는 실제 모바일 환경에서의 사용자 행동을 정확히 모델링하기 위해 독자적인 안드로이드 에이전트를 통해 직접 데이터를 수집하는 방식을 채택하였다. 초기 기획 단계에서 검토된 공개 데이터셋(CLUE-LDS)의 경우, 데스크톱 환경 기반으로 구성되어 모바일 디바이스의 터치 인터랙션, 센서 데이터, 위치 정보 등의 고유 특성을 충분히 반영하지 못하는 한계가 있었다. 따라서 모바일 UEBA 시스템의 현실 적합성을 높이기 위해 다음과 같은 세 가지 핵심 로그를 직접 수집하였다.

3.2.1 Touch 로그

- touch_pressure: 터치 좌표(x, y), 터치 영역 크기, 접촉 시간을 기록한다.
- touch_drag: 드래그 시작/종료 좌표, 이동 거리, 방향, 지속 시간, 이동 횟수를 수집한다.
- 터치 지속 시간과 드래그 패턴은 개인별 고유한 생체학적 특성을 반영하므로, 사용자 행동 기반 이상 탐지(UEBA)의 핵심 특징으로 활용한다.

[그림 3.2] Touch 로그 수집 예시

(a)Touch_Drag 이벤트 로그


```
[
  {
    "seq": 1026,
    "action_type": "drag",
    "ts": 1754559124429,
    "start_x": 569.0039,
    "start_y": 418.15625,
    "end_x": 594.3164,
    "end_y": 407.02344,
    "total_distance": 27.652525,
    "duration": 5,
    "drag_direction": "right",
    "move_count": 1,
    "touch_session": 1754559124424
  }
]
```


(b) Touch_Pressure 이벤트 로그


```
[
  {
    "seq": 1007,
    "action_type": "touch_pressure",
    "ts": 1754559108891,
    "x": 312.1875,
    "y": 687.5703,
    "size": 0.039215688,
    "touch_duration": 188,
    "touch_session": 1754559108703
  }
]
```

3.2.2 Sensor 로그

- **수집 데이터** : 가속도계(accelerometer)와 자이로스코프(gyroscope)의 3축(x, y, z) 데이터를 1초 단위로 수집한다.
- **행동 특성 반영** : 사용자의 걸음걸이, 기기 흔들림, 자세 변화 등 개인별 고유한 물리적 행동 패턴을 반영한다.
- **이상 탐지 목적** : 평소와 다른 움직임 패턴을 통해 디바이스 탈취나 무단 사용을 탐지한다.

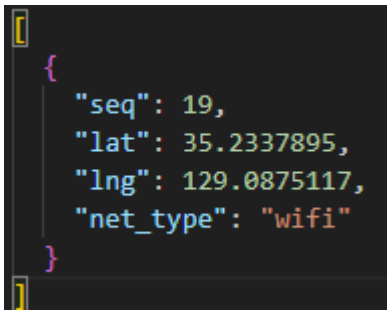
[그림 3.3] Sensor 로그 수집 예시


```
[
  {
    "seq": 634,
    "type": "gyro",
    "ts": 1754558532867,
    "x": -0.0016798794,
    "y": -0.00045814895,
    "z": -0.0009162979
  },
  {
    "seq": 634,
    "type": "accel",
    "ts": 1754558532867,
    "x": 0.7613561,
    "y": 0.79248077,
    "z": 9.799767
  }
]
```


3.2.3 Network 로그

- 접속 정보 수집 : Wi-Fi, LTE 등 네트워크 접속 유형 및 GPS 좌표를 실시간으로 기록한다.
- 지리적 이상 탐지 : 대한민국 행정구역 경계값(위도 33.1° ~ 38.6°N, 경도 124.6° ~ 129.5°E)을 벗어날 경우 GPS 스푸핑(Spoofing) 공격으로 간주하여 즉시 anomaly로 분류한다.
- 효율적 탐지 전략 : 지리적 경계라는 명확한 기준이 존재하므로 규칙 기반 접근법을 적용한다.

[그림 3.4] Network 로그 수집 예시



데이터 전처리

수집된 원시 로그는 JSON 형식으로 저장되며, x, y 좌표의 정규화, 시간 차이(time_delta) 계산, 드래그 속도 및 가속도 등의 파생 특징을 추출하여 머신러닝 모델의 입력으로 활용한다.

[Touch 로그 예시]

JSON


```
{
  "category": "touch",
  "logs": [
    {
      "action_type": "touch_pressure",
      "params": {
        "timestamp": 1754031944377,
        "event_type": "touch_pressure",
        "x": 608.9502,
        "y": 649.93555,
        "pressure": 1.0,
        "size": 5.9604645e-07,
        "view_info": {
          "view_id": 16908290,
          "view_class": "ContentFrameLayout",
          "view_width": 1080,
          "view_height": 2006,
          "screen_x": 0,
          "screen_y": 154
        },
        "sequence_index": 13
      }
    }
```

```
  ]
}
```

이 과정에서 x, y 좌표는 다양한 화면 크기에 대응하기 위해 0~1 범위로 정규화되며, 연속된 로그 간의 시간 차이(time_delta), 드래그 속도, 가속도 등의 파생 특징을 추출한다. 또한 view_info를 활용하여 터치가 발생한 UI 컴포넌트 정보를 포함시켜 사용자의 앱 사용 패턴까지 종합적으로 분석할 수 있도록 구성하였다.

3.3 백엔드 서버 구현

백엔드는 Django REST Framework 기반으로 구축되었으며, 사용자 행동 데이터 처리(Behavior 앱)와 이상 탐지 결과 관리(Anomaly 앱)로 기능을 분리하여 구현하였다.

3.3.1 Behavior 앱

Behavior 앱은 안드로이드 단말에서 발생하는 모든 사용자 행위 로그를 수집하고 처리하는 핵심 모듈이다. 수집된 데이터는 센서, 터치, 네트워크의 세 가지 모달리티로 분류되어 처리된다.

센서로그 수집

- SensorDataIngestView를 통해 가속도계(accelerometer), 자이로스코프(gyroscope) 데이터를 수집한다.
- 수집된 로그는 UserBehaviorLog 모델에 action_type, timestamp, params(x,y,z) 형태로 저장된다.
- 저장 즉시 머신러닝 서버의 /predict_hybrid 엔드포인트로 전달되어 실시간 이상 탐지를 수행한다.

터치 로그 수집

- TouchDataIngestView를 통해 터치 좌표, 영역 크기, 드래그 방향 등의 터치 이벤트를 수집한다.
- 단순 터치부터 드래그, 롱프레스까지 다양한 제스처 유형을 구분하여 저장함으로써 세밀한 사용자 행동 분석을 가능하게 한다.

네트워크 로그 수집

- NetworkDataIngestView를 통해 네트워크 연결 상태와 GPS 위치 정보를 동시에 수집한다.
- GPS 좌표가 대한민국 행정구역 경계값(위도 33.1° ~ 38.6°N, 경도 124.6° ~ 129.5°E)을 벗어날 경우 즉시 anomaly로 분류한다.
- GPS 스푸핑 시뮬레이션을 통한 검증 결과, 이상 상황이 정상적으로 탐지되어 anomaly 테이블에 기록됨을 확인하였다.

3.3.2 Anomaly 앱

Anomaly 앱은 머신러닝 서버의 탐지 결과를 체계적으로 관리하며, behavior_log와 1:N 관계를 통해 각 로그의 정상/비정상 판정 결과를 매핑한다

저장 항목

- modality: 데이터 모달리티 유형 (sensor, touch, network)
- timestamp: 로그 발생 시각
- anomaly_score: 모델별 이상 점수
- is_anomaly: 이상 여부 판정 (Boolean)
- detection_method: 탐지 모델 유형 (iforest, lstm, hybrid)

API 엔드포인트

- AnomalyResultList: 대시보드의 이상 탐지 결과 조회를 위한 API
- MobileAnomalyUpdates: 안드로이드 앱의 실시간 anomaly 상태 확인을 위한 폴링 API
  - since_id 매개변수를 통한 증분 업데이트 지원
  - is_anomaly=True 데이터만 선별 전송하여 네트워크 효율성 확보
  - 단일 엔드포인트 호출로 이상 탐지 원인과 대응 조치를 즉시 파악 가능

3.3.3 머신러닝 서버 연동

머신러닝 서버는 Django 대신 FastAPI를 기반으로 구현하였다. FastAPI는 비동기(Asynchronous) 처리를 지원하는 고성능 웹 프레임워크로, 머신러닝 추론(inference) API 서버 구축에 최적화되어 있다. Pydantic 기반의 자동 데이터 검증과 직관적인 라우팅 시스템을 통해 안정적인 REST API 설계가 가능하며, 동시 다발적인 anomaly 탐지 요청에 대해서도 실시간 응답을 보장한다.

연동 아키텍처

- **통신 프로토콜**: HTTP/REST API 기반의 동기식 통신을 채택하였다.
- **API 엔드포인트**: 단일 모델 추론을 위한 /predict와 하이브리드 모델 추론을 위한 /predict_hybrid 엔드포인트를 제공한다.
- **데이터 플로우**: Behavior 앱에서 수집된 로그는 실시간으로 머신러닝 서버로 전송되며, 탐지 결과는 Anomaly 앱을 통해 데이터베이스에 영구 저장된다.
- **성능 최적화**: FastAPI의 비동기 처리 기능을 활용하여 다중 요청의 병렬 처리와 응답 지연 최소화를 구현하였다.

3.3.4 하이브리드 모델 연동 방식

머신러닝 서버는 Isolation Forest(iForest)와 LSTM AutoEncoder를 결합한 하이브리드 아키텍처를 채택하여, 각 알고리즘의 장점을 활용한 정확하고 안정적인 이상 탐지 시스템을 구축하였다.

1. Isolation Forest (iForest)

- **학습 데이터 요구량**: 센서 로그 500개 이상, 터치 로그(드래그/압력) 각 500개이상 수집 후 초기 학습을 수행한다.
- **이상 판별 방식**: decision_function의 출력값을 기준으로 백분위수(percentile) 임계값을 적용하여 이상 여부를 판정한다.
- **적응형 재학습**: 모달리티별 설정된 주기(센서 500개 이상, 터치 500개 이상)에 따라 점진적 재학습을 수행하여 드리프트 현상에 대응한다.

2. LSTM AutoEncoder

- **시계열 모델링**: 고정 길이 시퀀스(seq_len=20)를 입력 단위로 하여 시간적 의존성을 학습한다.
- **이상 탐지 매커니즘**: 인코더-디코더 구조를 통한 시퀀스 재구성 과정에서 발생하는 reconstruction error를 이상 지표로 활용한다.
- **동적 임계값**: 재구성 오차 분포의 1~99 백분위수 범위를 벗어나는 경우 anomaly로 분류한다.
- **연속 학습**: 사용자 행동 패턴의 변화와 환경적 요인에 적응하기 위해 주기적 재학습 매커니즘을 구현하였다.

3. Hybrid 융합 전략

- **보수적 탐지 정책**: OR 논리를 적용하여 두 모델 중 하나라도 anomaly를 탐지하면 최종 이상으로 분류한다. 이는 보안 도메인에서 False Negative보다 False Positive가 상대적으로 덜 치명적이라는 특성을 반영한 설계이다.
- **통합 점수 계산**: 최종 anomaly_score는 두 모델의 정규화된 점수에 대한 가중평균으로 산출한다.
- **상호 보완성**: iForest의 통계적 이상 탐지와 LSTM의 시계열 패턴 분석이 결합되어 다양한 유형의 이상 행위에 대한 포괄적 탐지가 가능하다.

3.3.5 네트워크 모달리티 특수 처리

네트워크 모달리티는 센서 및 터치 데이터와 달리 규칙 기반(Rule-based) 접근법을 적용한다.

- 처리 방식
  - 머신러닝 서버를 거치지 않고 백엔드에서 직접 GPS 좌표 기반 이상 탐지를 수행한다.
  - 수집된 위치 정보가 대한민국 행정구역 경계값(위도 33.1° ~ 38.6°N, 경도 124.6° ~ 129.5°E)을 벗어날 경우 즉시 anomaly로 분류한다.
- 설계 근거
  - 명확한 판단 기준: GPS 스푸핑과 같은 위치 기반 공격은 지리적 경계라는 명확하고 절대적인 기준이 존재한다.
  - 실시간 대응: 복잡한 모델 추론 과정 없이 즉각적인 탐지와 차단이 가능하

여 보안 대응 시간을 최소화할 수 있다.

- 계산 효율성: 단순 좌표 비교 연산으로 처리되어 시스템 리소스 사용량이 최소화되며, 높은 처리량을 보장한다.
- 오탐지 최소화: 학습 데이터에 의존하지 않아 환경 변화나 사용자 패턴에 영향받지 않는 안정적인 탐지가 가능하다.

3.3.6 대시보드 및 안드로이드 연동

1. 대시보드

- 데이터 조회: AnomalyResultList API를 통해 실시간 이상 탐지 결과를 테이블 형태로 시각화한다.
- 통합 모니터링: 사용자별, 시간대별, 모달리티별 이상 행위 발생 현황을 통합적으로 제공하여 관리자의 신속한 상황 파악을 지원한다.
- 세부 분석: 각 anomaly 이벤트에 대한 상세 정보(탐지 시각, 행위 유형, 위험도 점수)를 제공하여 심층적인 보안 분석이 가능하다.

2. 안드로이드 클라이언트

- 폴링 기반 동기화: MobileAnomalyUpdates API를 5초 주기로 호출하여 새로운 anomaly 탐지 결과를 확인한다.
- 실시간 알림: 이상 행위 탐지 시 해당 액션 유형과 권장 보안 조치를 포함한 즉시 알림을 제공한다.
- 통신 방식 선택: WebSocket 기반 실시간 푸시와 HTTP 폴링 방식을 비교 검토한 결과, 시스템 안정성과 구현 복잡도를 고려하여 폴링 방식을 채택하였다.

3.4 머신러닝 서버 및 이상 탐지 모델

사용자 행동 데이터의 다차원적 특성을 고려하여, 통계적 이상 탐지(Isolation Forest)와 딥러닝 기반 시계열 분석(LSTM Auto-Encoder)을 융합한 하이브리드 탐지 아키텍처를 설계하였다. 이를 통해 단일 모델의 한계를 극복하고 다양한 유형의 이상 행위에 대한 안정적인 탐지 성능을 확보하였다.

3.4.1 Isolation Forest (iForest)

본 연구에서는 사용자 행동 데이터의 특성을 고려하여 **Isolation Forest** 알고리즘을 핵심 이상 탐지 방법론으로 채택하였다. Isolation Forest는 이상치가 정상 데이터에 비해 "분리

하기 쉽다"는 가정에 기반한 트리 기반 앙상블 방법으로, 레이블이 없는 환경에서도 효과적인 이상 탐지가 가능하다는 장점이 있다.

모달리티별 특화 모델

사용자 행동의 다양성을 고려하여 각 데이터 모달리티별로 독립적인 탐지 모델을 구축하였다:

- **센서 모달리티**: 가속도계 데이터(x, y, z)를 3차원 특징 벡터로 변환하여 기기의 물리적 움직임 패턴을 학습한다.
- **터치 드래그 모달리티**: 드래그 지속시간, 총 이동거리, 직진성, 속도, 방향성(상/하/좌/우)을 포함한 9차원 특징 벡터를 구성하여 동적 제스처의 복합적 특성을 모델링한다.
- **터치 압력 모달리티**: 터치 지속시간, 접촉 크기, 좌표 정보를 4차원 특징 벡터로 구성하여 개인별 터치 고유 패턴을 학습한다.

적응형 학습 매커니즘

실시간 환경에서의 효율적인 모델 운영을 위해 다음과 같은 적응형 학습 전략을 도입했다:

초기 학습 단계: 센서 모달리티는 500개, 터치 모달리티(드래그/압력)는 각각 200개의 샘플을 수집한 후 초기 모델을 학습한다. 이 과정에서 StandardScaler를 통한 특징 정규화와 함께, 오염률(contamination) 1%로 설정된 Isolation Forest 모델(n_estimators=200)을 구축한다.

임계값 설정: 학습 데이터에 대한 결정 함수(decision function) 점수 분포를 기반으로, 하위 5% 백분위수를 기본 임계값으로 설정하고 3~4% 백분위수 범위의 안전 마진을 적용하여 오탐지를 최소화한다.

재학습 매커니즘: 모델 성능 유지를 위해 모달리티별 설정된 추론 횟수(센서 500회, 터치 200회)마다 자동 재학습을 수행한다. 이를 통해 사용자 행동 패턴의 변화나 환경적 요인에 대한 모델의 적응성을 확보한다.

실시간 탐지 프로세스

구현된 시스템은 다음과 같은 단계로 실시간 이상 탐지를 수행한다:

1. **특징 추출**: 입력된 행동 로그에서 action_type을 기반으로 해당 모달리티를 식별

하고, 사전 정의된 특징 추출 함수를 통해 수치형 특징 벡터를 생성한다.

2. **정규화 및 추론**: 학습된 StandardScaler를 사용하여 특징을 정규화한 후, Isolation Forest 모델을 통해 이상 점수를 계산한다.
3. **이상 판정**: 계산된 점수가 설정된 임계값 이하인 경우 이상 행위로 분류하며, 결과는 JSON 형태로 반환된다.

3.4.2 LSTM Auto-Encoder

사용자 행동은 개별 사건의 나열이 아닌, 시간적 맥락이 중요한 시계열 데이터이다. 이러한 순차적 특성을 모델링하기 위해 순환 신경망(RNN)의 한 종류인 LSTM(Long Short-Term Memory)을 채택하였다. LSTM은 입력 게이트(input gate), 망각 게이트(forget gate), 출력 게이트(output gate)로 구성된 메모리 셀 구조를 통해 장기 의존성 문제를 해결함으로써, 시간적으로 멀리 떨어진 이벤트 간의 복잡한 관계를 학습할 수 있다.

본 연구에서는 레이블이 없는 데이터를 학습하기 위해 비지도 학습 프레임워크인 Auto-Encoder를 LSTM과 결합하였다. LSTM Auto-Encoder는 인코더와 디코더 두 부분으로 구성된다.

모델 구조

- **인코더(Encoder)**: 입력된 행동 시퀀스(seq_len=20)를 저차원의 잠재 공간 벡터로 압축한다.
- **디코더(Decoder)**: 압축된 잠재 벡터를 사용하여 원본 시퀀스를 다시 복원한다.

학습 및 탐지 매커니즘: 모델은 오직 '정상' 행동 데이터만으로 학습된다. 이로 인해 모델은 정상적인 패턴을 압축하고 복원하는 데 특화된다. 학습된 모델에 정상 범주에서 벗어나는 '비정상' 시퀀스가 입력되면, 모델은 이를 제대로 복원하지 못하고 높은 재구성 오차(Reconstruction Error)를 발생시킨다.

이상 판정 기준: 재구성 오차 분포의 1~99 백분위수(q01, q99) 범위를 벗어나는 경우 해당 시퀀스를 이상 행위로 탐지한다. 이러한 동적 임계값 설정을 통해 사용자 행동 패턴의 변화에 적응적으로 대응할 수 있다.

적응형 재학습: iForest와 동일하게 데이터가 일정량 누적되면 주기적으로 재학습을 수행하여 새로운 사용자 행동 패턴을 반영하고, 시간 경과에 따른 환경 변화에도 대응할 수 있도록 하였다.

3.5 대시보드 구현

대시보드는 React + TypeScript로 개발되었으며, anomaly API를 기반으로 관리자에게 보안 상태를 시각적으로 제공한다.

3.5.1 메인 대시보드

메인 대시보드는 관리자가 시스템의 전반적인 보안 상태를 실시간으로 모니터링할 수 있도록 설계된 핵심 인터페이스이다.

구성 요소

- **리스크 게이지:** 현재 시스템의 평균 리스크 수준을 게이지 형태로 시각화
- **리스크 점수 차트:** 최근 20개 이벤트의 리스크 점수를 시계열로 표시
- **최근 로그 테이블:** 최신 사용자 활동 로그와 판정 결과를 실시간 제공

[그림 3.5] 메인 대시보드 화면

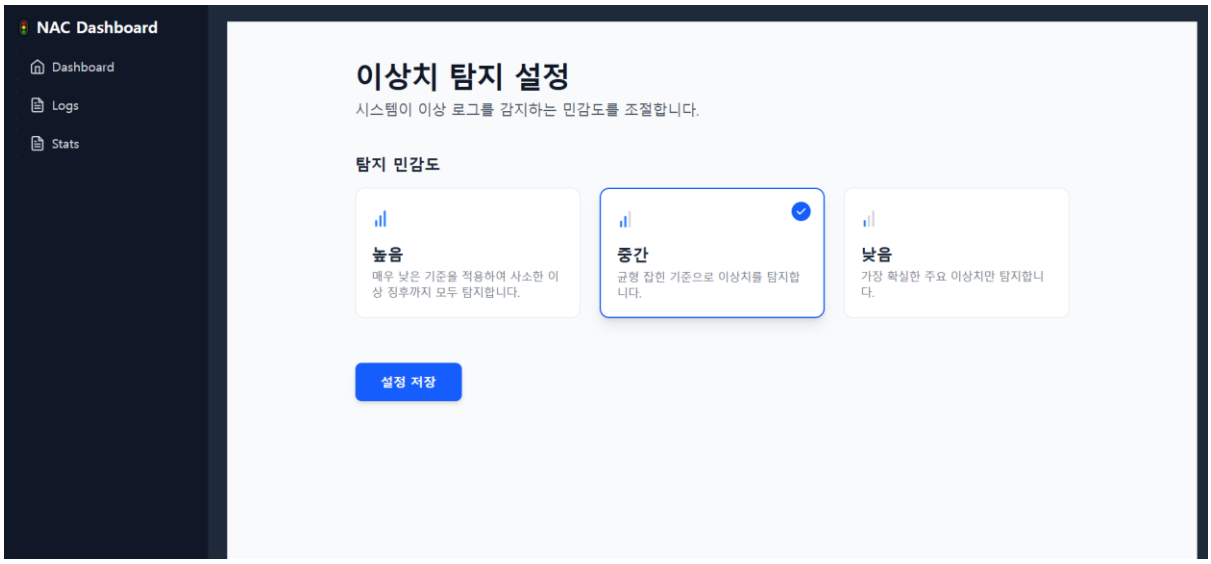
3.5.2 메인-설정페이지

관리자가 시스템 정책을 동적으로 조정할 수 있는 설정 인터페이스이다.

향후 확장 계획

- **정책 강도 조정 :** 이상 탐지 민감도 및 대응 수준 실시간 조정
- **사용자별 정책 설정** : 개별 또는 그룹별 맞춤형 보안 정책 적용

[그림 3.6] 설정 페이지 화면



3.5.3 Logs 페이지

시스템에서 수집된 모든 로그 데이터를 상세하게 조회하고 분석할 수 있는 인터페이스이다.

주요 기능:

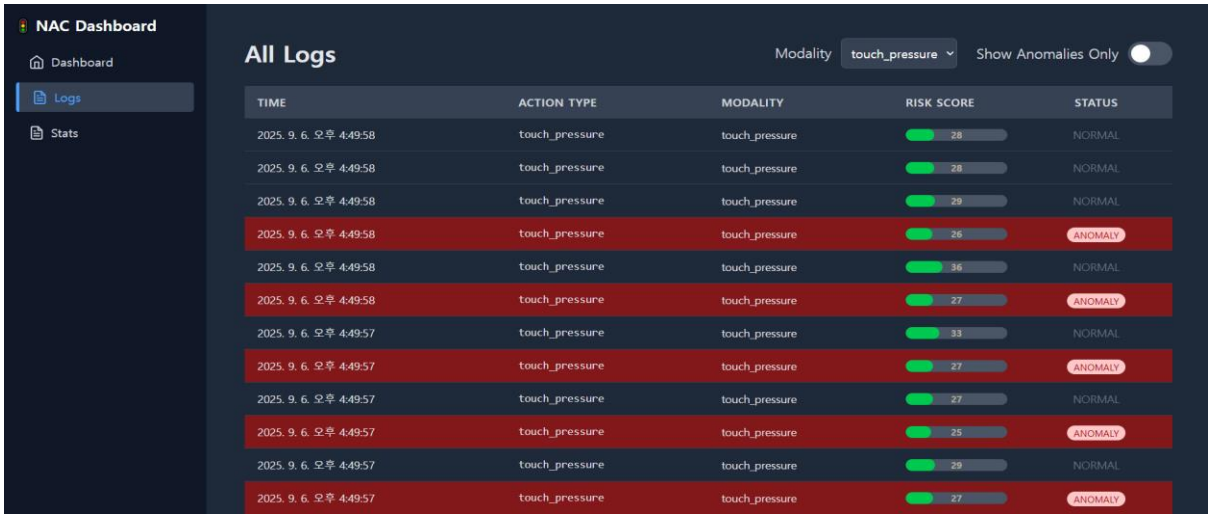
- **페이지네이션 :** 20개 단위로 로그를 표시하며, 추가 로딩을 통한 확장 조회 지원
- **필터링 :** 로그 유형별(All, network, sensor, touch_drag, touch_pressure) 선별 조회
- **이상 로그 필터 :** anomaly로 분류된 로그만 별도 조회 가능

데이터 표시

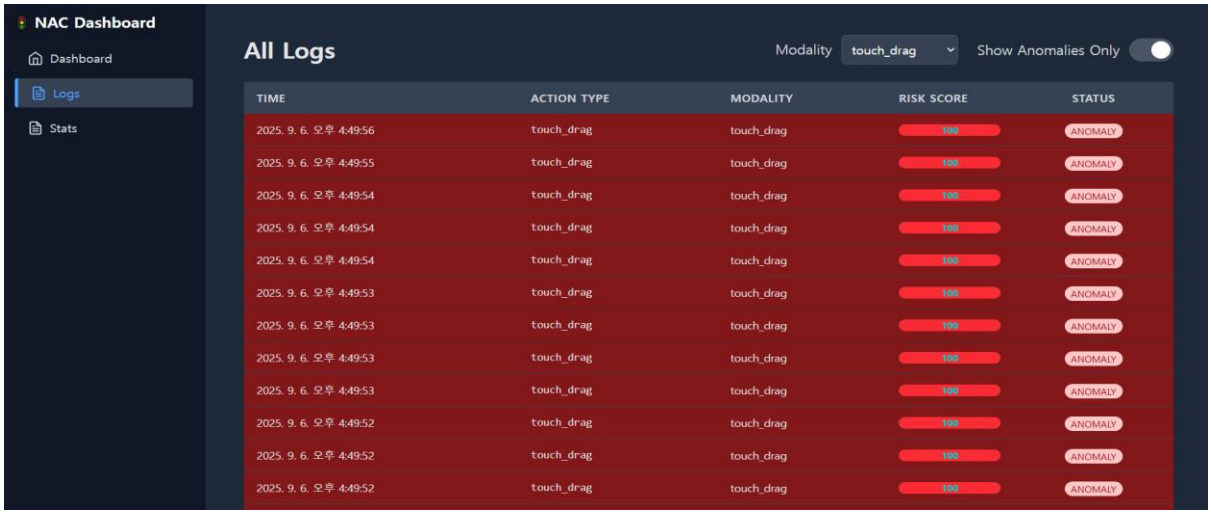
- 타임스탬프, 사용자 ID, 기기 정보, 활동 내역, 정책 결정을 테이블 형태로 구조화
- 높은 리스크 로그는 시각적 강조를 통해 즉시 식별 가능
- 시간대별, 사용자별, 기기별 검색 및 필터링 기능 제공

[그림 3.7] 20개 단위로그 및 더보기

[그림 3.8] 로그 유형별 선별 조회



[그림 3.9] 이상 로그 필터



3.5.4 Stats 페이지

로그 데이터에서 위험도가 높은 이벤트들을 우선순위별로 정리하여 관리자의 효율적인 위협 분석을 지원하는 통계 페이지이다.

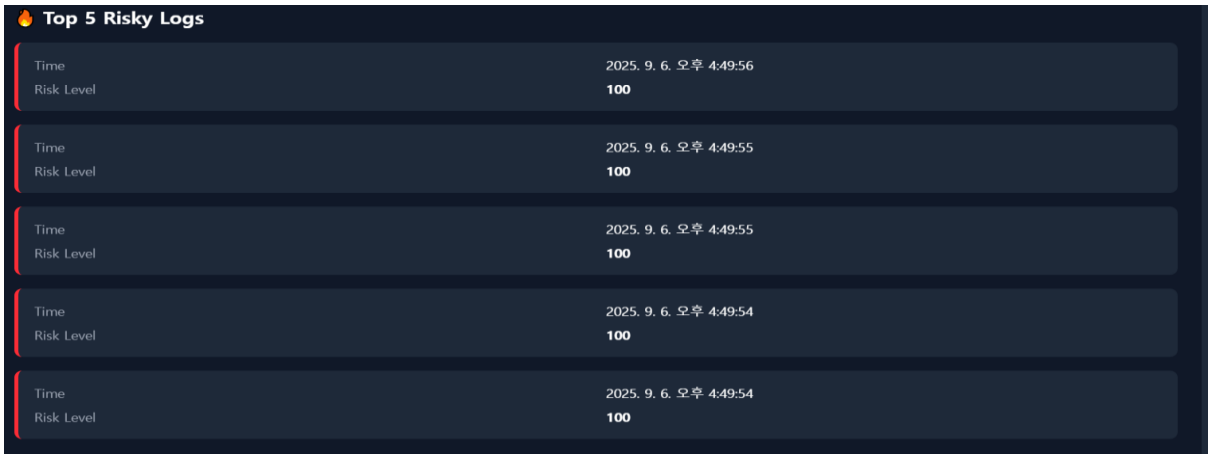
주요 기능: 최신 리스크 트렌드를 시각화하고, 높은 위험도를 가진 로그들을 우선순위별로 제공하여 관리자의 위협 분석 및 대응 우선순위 결정을 지원한다.

구성요소:

- **리스크 점수 트렌드 :** 최근 20 개 이벤트의 리스크 점수를 시계열 그래프로 표시하여 위험도 변화 추이를 분석할 수 있다.
- **최고 위험 로그 :** 전체 로그 중 가장 높은 리스크 점수를 기록한 로그의 발생 시간과 위험도를 표시한다.
- **상위 위험 로그 :** 위험도 상위 5 개 로그를 순위별로 나열하여 주요 위협 이벤트들을 일괄 확인할 수 있다.

[그림 3.10] Stats 페이지

[그림 3.11] 상위 위험 로그



3.6 안드로이드 연동

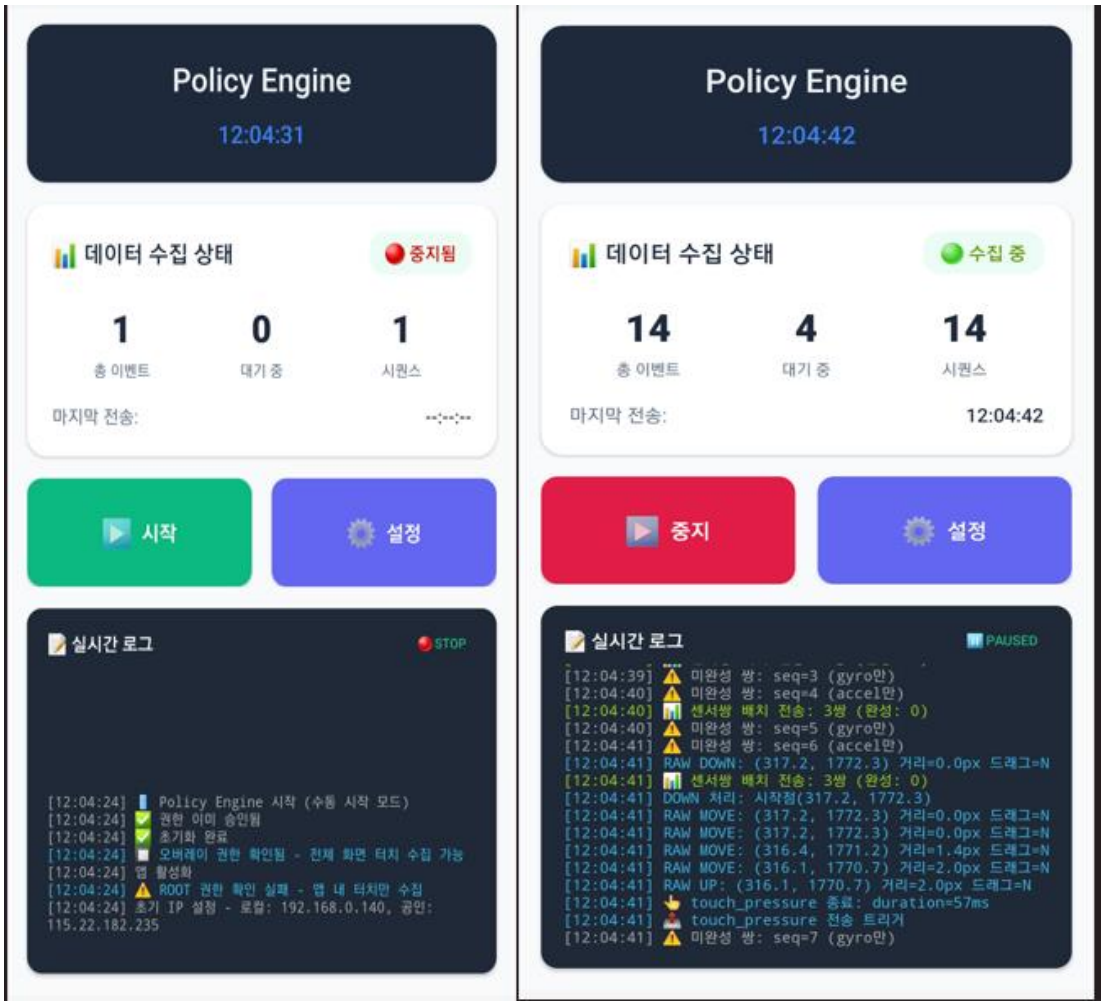
안드로이드 클라이언트는 MobileAnomalyUpdates API를 통해 5초 주기로 이상 탐지 결과를 폴링하여 실시간 보안 대응을 지원한다.

사용자 인터페이스

- 서비스 시작/정지 버튼을 통한 모니터링 제어
- 설정 메뉴를 통한 사용자 맞춤 구성
- 실시간 로그 화면을 통한 현재 상태 확인

[그림 3.12] 안드로이드 앱 사용자 인터페이스

(a) 초기 화면 – 시작 버튼 **(b) 로그 수집 중 화면**

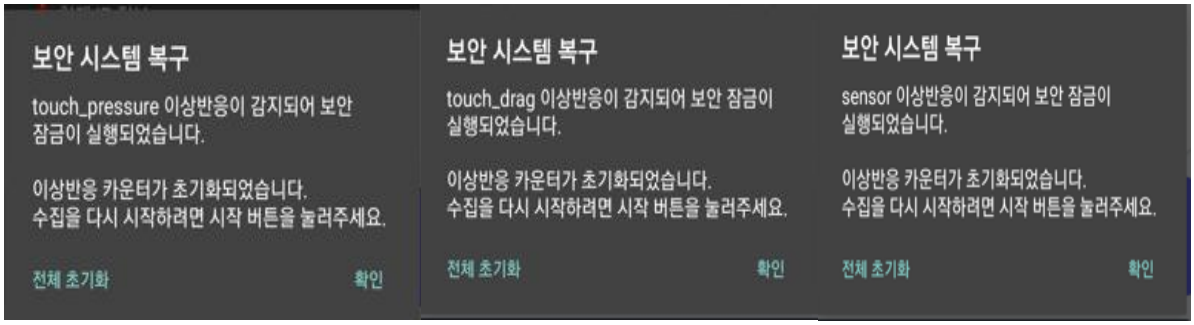


(c) 설정 화면

이상 탐지별 대응 방식

- **네트워크 이상 :** 팝업을 통해 "비행기 모드 전환", "Wi-Fi 차단", "무시" 옵션을 제공하여 사용자가 직접 대응 방식을 선택할 수 있다

- **터치/센서 이상 :** 자동으로 잠금화면으로 전환하며, 앱 재실행 시 anomaly 발생 알림을 표시한다.



시스템 성능

API 응답 시간 최적화를 통해 실시간에 근접한 사용자 경험을 제공하며, 폴링 방식의 안정성과 네트워크 효율성을 확보하였다.

3.7 동작 흐름 요약

본 시스템의 전체 동작 흐름은 다음과 같다 (그림 3.14 참조):

1. 데이터 수집: 안드로이드 에이전트가 사용자의 터치, 센서, 네트워크 로그를 실시간으로 수집하여 Django 백엔드로 전송한다.

2. 데이터 처리: Behavior 앱에서 수신된 로그를 UserBehaviorLog 모델에 저장하고, 전처리를 거쳐 머신러닝 서버로 전달한다.

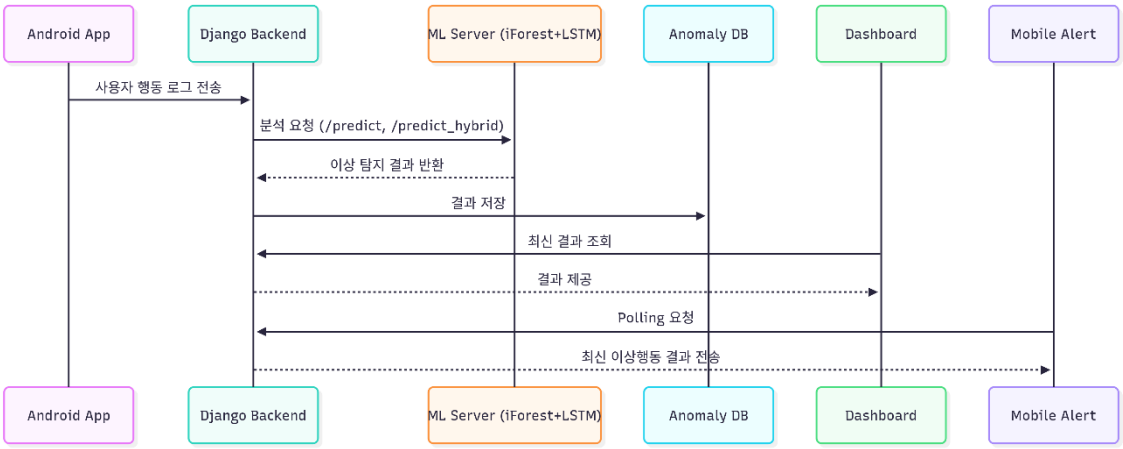
3. 이상 탐지: ML 서버에서 iForest와 LSTM AutoEncoder를 결합한 하이브리드 모델을 통해 이상 여부를 판별한다.

4. 결과 저장: 탐지 결과는 Anomaly 앱을 통해 AnomalyResult 테이블에 저장되며, 관련 API를 통해 조회 가능하다.

5. 관리자 모니터링: React 기반 대시보드에서 실시간 이상 탐지 현황과 통계를 시각화하여 관리자에게 제공한다.

6. 사용자 알림: 안드로이드 앱이 폴링 방식으로 anomaly 결과를 확인하여 사용자에게 적절한 보안 대응 옵션을 제공한다.

[그림 3.14] 시스템 전체 동작 흐름도



4. 연구결과 분석 및 성능 평가

4.1 모델 성능 평가

4.1.1 평가 방법론

LSTM Auto-Encoder 모델의 성능은 Sensor, Touch Drag, Touch Pressure 세 가지 행동 양식에 대해 수집된 실제 로그 데이터셋을 사용하여 평가하였다. 모델의 성능은 정밀도(Precision), 재현율(Recall), F1-Score, 정확도(Accuracy)를 통해 정량적으로 평가하였다.

이상 탐지 임계값은 정상 데이터의 재구성 오차 분포를 분석하여 설정하였다. 특히 본 평가에서는 EXTREMELY RELAXED THRESHOLDS 전략을 사용하였다. 이는 오차 분포의 1%(q01) 및 99%(q99) 지점을 기준으로 상하한 경계를 매우 완화하여 설정하는 방식이다 (예: lower_bound = 1% * 0.5, upper_bound = 99% * 1.1). 이는 학습 데이터에서 관찰된 것보다 평가 데이터의 정상 행동 변동성이 더 컸기 때문이다.

4.1.2 모드별 성능 분석

Sensor 데이터 성능 센서 데이터에 대한 모델 성능은 F1-Score 0.9654, 정밀도 0.9887, 재현율 0.9432를 기록하였다. 이는 모델이 이상 행위로 판단한 예측의 98.87% 이상이 정확했으며, 실제 비정상 시퀀스의 94% 이상을 성공적으로 탐지했음을 의미한다. 이 결과는 기기를 다루는 무의식적인 물리적 패턴이 매우 안정적이고 고유한 행동 식별자로 작용할 수 있음을 입증한다.

Touch 데이터 성능 (Drag 및 Pressure) 추가 로그를 통해 Touch Drag와 Touch Pressure의 정상 데이터 샘플이 각각 954개와 913개로 크게 증가하면서, 모델 성능에 의미 있는 변화가 관찰되었다.

Touch Drag 모델의 경우, 비정상 행위 탐지 F1-Score는 0.8425로 이전(0.9377)보다 감소했지만, 정상 클래스에 대한 정밀도가 0.5582에서 0.7620으로 크게 향상되었다. 이는 정상 데이터가 대폭 보강되면서 모델이 정상적인 드래그 패턴을 더 잘 학습하게 되어, 정상적인 사용자를 비정상으로 오탐하는 경우가 줄었음을 의미한다.

Touch Pressure 모델 역시 유사한 경향을 보였다. 비정상 탐지 F1-Score는 0.8871로 이전(0.9620)보다 감소했지만, 정상 클래스에 대한 정밀도는 0.8566에서 0.9472로 크게 상승했다.

[그림 4.1] 모드별 성능 평가 결과

```
==================================
    EVALUATION REPORT: SENSOR
==================================
              precision    recall  f1-score   support

      Normal     0.9367    0.9905    0.9628      8600
    Abnormal     0.9915    0.9432    0.9668     10142

    accuracy                         0.9649     18742
   macro avg     0.9641    0.9668    0.9648     18742
weighted avg     0.9663    0.9649    0.9649     18742
```

```
==================================
    EVALUATION REPORT: TOUCH_DRAG
==================================
              precision    recall  f1-score   support

      Normal     0.7865    0.8648    0.8238       954
    Abnormal     0.9053    0.8463    0.8748      1457

    accuracy                         0.8536      2411
   macro avg     0.8459    0.8555    0.8493      2411
weighted avg     0.8583    0.8536    0.8546      2411
```

```
==================================
    EVALUATION REPORT: TOUCH_PRESSURE
==================================
              precision    recall  f1-score   support

      Normal     0.9393    0.6780    0.7875       913
    Abnormal     0.7584    0.9585    0.8468       963

    accuracy                         0.8220      1876
   macro avg     0.8489    0.8182    0.8172      1876
weighted avg     0.8465    0.8220    0.8180      1876
```

4.1.3 전체 성능 요약

평가 결과, 시스템의 전반적인 성능은 평균 F1-Score 0.8961, 평균 재현율 0.9160을 기록하였다. 재현율이 90%를 상회하여 시스템이 대부분의 이상 행위를 성공적으로 탐지하는 능력을 보여주며, 특히 센서 데이터에서 가장 높은 성능(F1-Score 0.9668)을 달성하였다.

[그림 4.2] 전체 성능 요약 (DETECT-STYLE)

```
############################################################
     OVERALL PERFORMANCE SUMMARY (DETECT-STYLE)
############################################################
Model (Mode)          |   F1-Score |   Accuracy |  Precision |     Recall
------------------------------------------------------------------------
Sensor                |     0.9668 |     0.9649 |     0.9915 |     0.9432
Touch Drag            |     0.8748 |     0.8536 |     0.9053 |     0.8463
Touch Pressure        |     0.8468 |     0.8220 |     0.7584 |     0.9585
############################################################

🎯 Average F1-Score: 0.8961
🎯 Average Recall: 0.9160
```

4.2 Isolation Forest 기반 이상 탐지 성능 평가

4.2.1 알고리즘 구현 및 설정

본 연구에서는 사용자 행동 데이터의 특성을 고려하여 **Isolation Forest** 알고리즘을 이상 탐지 방법론으로 채택하였다. Isolation Forest는 이상치가 정상 데이터에 비해 "분리하기 쉽다"는 가정에 기반한 트리 기반 앙상블 방법으로, 레이블이 없는 환경에서도 효과적인 이상 탐지가 가능하다는 장점이 있다.

모델 구현에서는 다음과 같은 핵심 파라미터를 사용하였다:

- **n_estimators=200**: 200개의 결정 트리 앙상블을 통한 안정적 성능 확보
- **contamination=0.03**: 전체 데이터의 3%를 이상치로 가정하여 실제 사용 환경 반영
- **임계값 설정**: 학습 데이터 결정 함수 점수의 하위 0.5% 백분위수 + 0.2% 마진 적용

4.2.2 모달리티별 성능 분석

센서 모달리티:

- 3차원 가속도계 데이터(x, y, z)를 활용한 물리적 패턴 모델링
- 500개 샘플 기준 초기 학습으로 안정적 기준선 구축
- 무의식적 기기 조작 패턴의 고유성 활용으로 개인별 움직임 특성 반영

터치 드래그 모달리티:

- **9차원 특징 벡터 구성:** 지속시간, 이동거리, 속도, 방향성(상/하/좌/우)
- 200개 샘플로 개인별 제스처 특성 학습
- 동적 움직임의 복합적 특성 반영으로 개인 고유 패턴 모델링

터치 압력 모달리티:

- **4차원 특징 벡터:** 지속시간, 접촉 크기, 좌표 정보
- 생체학적 터치 특성 기반 개인 식별 특징 활용
- 안드로이드 플랫폼 제약 고려한 효율적 특징 추출

[그림 4.3] Isolation Forest 모달리티별 성능 결과

```
======================================================================
요약 결과
======================================================================
| Modality       | Precision  | Recall    | F1-Score    | Accuracy |
----------------------------------------------------------------------
| sensor         | 0.8676     | 0.8551    | 0.8613      | 0.9456     |
| touch_drag     | 0.8752     | 0.9484    | 0.9103      | 0.8376     |
| touch_pressure | 0.9182     | 0.8719    | 0.8944      | 0.8211     |
----------------------------------------------------------------------
평가 완료 시간: 2025-09-10 21:49:09
```

4.3 하이브리드 모드 성능

구현 확인:

- Isolation Forest와 LSTM AutoEncoder의 개별 모델이 정상적으로 학습되고 추론을 수행함을 확인하였다.
- OR 논리 기반 융합 전략이 의도한 대로 동작하여, 두 모델 중 하나라도 이상을 탐지하면 최종 anomaly로 분류되는 것을 검증하였다.

동작 특성 관찰:

- 단일 모델 운영 시: 각 모델의 고유한 탐지 특성 확인
- 하이브리드 모드 운영 시: 단일 모델에서 놓치는 패턴을 상호 보완하는 동작 확

인

- 통합 점수 계산이 두 모델의 정규화된 점수 가중평균으로 정상 산출됨을 검증

보수적 탐지 정책 검증:

- 보안 시스템 특성상 False Negative 방지를 우선시하는 OR 논리 적용
- 개별 모델의 임계값 조정을 통한 전체 시스템 민감도 제어 가능성 확인

4.4 시스템 운영 검증

4.4.1 시스템 동작 검증

구현한 백엔드와 머신러닝 서버를 실제 로그 데이터와 연동하여 검증하였다.

정상 동작 확인

- Behavior 앱에 로그가 입력되면, ML 서버로 전달 → anomaly 앱에 결과 저장까지 전 과정이 정상적으로 수행되었다.
- 대시보드와 안드로이드 앱은 anomaly API를 호출해서 즉시 이상여부를 받아올 수 있었다.

실시간 학습/추론

- Isolation Forest: 센서는 500개, 터치는 200개씩 누적되면 자동 학습을 시작하고, 이후에는 일정 간격마다 재학습이 이루어졌다.
- LSTM AutoEncoder: seq_len=20 기준으로 시퀀스를 구성하고, reconstruction error 기반으로 anomaly를 판별하였다.
- 두 모델을 결합한 Hybrid 모드에서는 단일 모델보다 이상탐지 민감도가 높아졌다.

네트워크 모달리티 검증

- GPS 스푸핑(Spoofing) 기법을 사용해 위치를 해외로 조작했을 때 anomaly가 정상적으로 탐지되었다.
- anomaly 테이블에는 modality="location_check"로 기록되어 네트워크의 이상행위가 명확히 구분되었다.

4.4.2 성능 분석

탐지 정확도

- Isolation Forest 단독: 오탐이 줄어드는 대신 민감도가 떨어졌다.
- LSTM 단독: 패턴 기반 탐지가 강하지만, 데이터 부족 시 "시퀀스 생성 실패" 같은 문제가 발생하였다.
- Hybrid: 두 모델을 결합하니 단일 모델에서 탐지하지 못하는 anomaly도 보완할 수 있었다.

응답 속도 및 안정성

- REST API 호출→응답 시간은 평균 50~100ms 정도로 측정되었다.
- 폴링(polling) 주기를 5초로 설정해도 사용자 입장에서는 "실시간에 가까운 알림" 경험을 제공할 수 있었다.
- 로그 데이터가 1500개 이상 쌓이면 오래된 데이터를 자동 삭제하도록 하여 DB 폭주를 방지하였다.

4.4.3 시스템 한계점 및 개선방향

한계점

1. LSTM 학습 데이터 부족 문제: seq_len이 20 이상 필요하기 때문에 초반에 데이터가 부족할 경우 학습이 지연되었다.

2. 네트워크 모달리티 한계: GPS 좌표 기반 탐지만으로는 VPN이나 프록시를 통한 네트워크 우회를 완전히 탐지하기 어려웠다. 이를 보완하기 위해 안드로이드 클라이언트에서 IP 주소 변경을 모니터링하여 네트워크 환경 변화 시 동일한 이상 탐지 로직을 적용하도록 구현하였다.

3. 폴링(Polling) 방식의 실시간성 한계: WebSocket을 사용하면 더 즉각적인 알림이 가능하다.

개선방향

1. 모델 고도화: GRU, Transformer 기반 이상탐지 모델 적용 및 adversarial training 기법 도입

2. 네트워크 이상탐지 강화: 현재 구현된 IP 주소 변경 탐지 기능을 확장하여 GeoIP 기반 상세 분석 및 VPN/프록시 패턴 학습 기능 추가

3. 실시간 전송 개선: WebSocket 도입을 통한 즉시 push 알림 시스템 구축

4.5 대시보드 시스템 평가

본 대시보드는 UEBA 기반 보안 시스템의 운영에 필요한 주요 기능들을 구현하였다.

4.5.1 완결된 탐지-분석-대응 워크플로우

시스템은 이상 탐지부터 분석, 대응까지의 통합된 워크플로우를 제공한다. 관리자는 메인 대시보드에서 이상 징후를 확인한 후, Logs 페이지와 Stats 페이지를 통해 상세 분석을 수행하고, 정책 관리 페이지에서 대응 조치를 설정할 수 있도록 구현하였다. 이를 통해 별도의 도구 전환 없이 단일 인터페이스에서 보안 운영 업무를 수행할 수 있다.

4.5.2 직관적 시각화를 통한 다층적 정보 전달

메인 대시보드는 실시간 보안 상태 개요를, Stats 페이지는 위험도 기준 우선순위 정보를 제공한다. 머신러닝 모델의 수치 결과를 게이지, 차트, 테이블 등의 시각적 요소로 변환하여 관리자의 직관적 이해를 지원하도록 설계하였다. 각 페이지는 서로 다른 층위의 정보를 제공하여 상황별 필요에 따른 정보 접근이 가능하다.

4.5.3 강화된 데이터 분석 역량

Logs 페이지에는 모달리티별 필터링, 이상 로그 선별 조회, 시간대별 검색 기능을 구현하였다. Stats 페이지는 최고 위험 로그 식별과 상위 5개 위험 로그 목록을 제공한다. 이러한 기능들을 통해 관리자는 대량의 로그 데이터에서 특정 조건에 해당하는 정보를 효율적으로 조회하고 분석할 수 있다.

4.6 안드로이드 클라이언트 고도화 구현

4.6.1 네트워크 보안 강화 메커니즘 구현

안드로이드 클라이언트에서는 VPN이나 프록시를 통한 네트워크 우회 시도를 탐지하기 위한 IP 주소 변경 모니터링 시스템을 구현하였다.
getCurrentLocalIP()와 getCurrentPublicIP() 함수를 통해 5초 주기로 현재 로컬 IP와 공인 IP를 확인하여 이전 값과 비교하고, 변경이 감지되면 이상 반응으로 분류하여 처리한다.

구현 특징: VPN 활성화 상태를 포함한 네트워크 환경 변화를 모니터링하며, 변경 감지 시 비행기모드 설정, WiFi 차단, 모바일 데이터 차단 등의 보안 조치 옵션을 사용자에게 제시하는 대화상자를 표시한다.

4.6.2 다중 모달 데이터 수집 시스템 구현

센서, 터치, 네트워크 로그 수집에서 성능과 효율성을 고려한 차별화된 수집 전략을 적용하였다. 센서 데이터는 50Hz로 수집한 후 1초 단위로 배치 전송하여 네트워크 부하를 감소시키고, 터치 이벤트는 즉시 전송하여 실시간 분석을 지원한다. 네트워크 상태는 1초 간격으로 수집하여 연속성을 확보한다.

품질 관리 매커니즘: 센서 데이터 페어링 상태, 터치 이벤트 분류 결과, 네트워크 상태 연속성을 실시간으로 모니터링한다. 큐 크기를 1500개로 제한하고 자동 정리 기능을 통해 메모리 사용량을 관리한다.

4.6.3 사용자 인터페이스 구현

사용자 제어 기능과 시스템 모니터링 기능을 통합한 인터페이스를 구현하였다. 실시간 로그 시작/중지 기능을 제공하며, 로그 화면에서는 모달리티별 색상 코딩(터치-파란색, 센서-초록색, 네트워크-보라색)을 적용하여 데이터 유형을 구분한다.

모니터링 기능: 설정 화면에서 모달리티별 이상 반응 카운터, 총 수집 이벤트 수, 큐 상태, 전송 시간을 확인할 수 있다. 데이터 초기화, 카운터 리셋, Device Admin 권한 확인 기능을 제공한다.

복구 매커니즘: 이상 탐지로 인한 자동 잠금 후 앱 재시작 시 복구 모드를 실행하여 이상 반응 플래그를 정리하고 정상 상태로 복원한다.

5. 결론 및 향후 연구 방향

5.1 연구 요약

본 연구는 **모바일 에이전트를 활용한 사용자 행동 분석 기반 동적 네트워크 접근 제어 (NAC) 시스템**을 개발하는 것을 목표로 하였다.
시스템은 크게 **로그 수집 에이전트, 백엔드 서버, 머신러닝 기반 이상 탐지 모델, 대시보드 및 사용자 피드백 모듈**로 구성되며, 다음과 같은 흐름을 따른다.

1. **에이전트**를 통한 Touch, Sensor, Network 로그 수집
2. **백엔드 서버**에서 로그 저장 및 전처리, ML 서버와 연동
3. **머신러닝 서버**에서 Isolation Forest와 LSTM Auto-Encoder를 활용한 하이브리드 이상 탐지 수행

4. **대시보드**를 통한 관리자용 실시간 시각화 및 로그 분석 지원
5. **안드로이드 앱**을 통한 사용자 피드백 및 즉각적인 네트워크 제어 실행

이를 통해 "수집–탐지–제어–시각화"의 완결된 보안 워크플로우를 구현하였다.

5.2 연구 성과

1. **안정적 로그 수집 및 이상 탐지**
    - Touch, Sensor, Network 로그를 실시간으로 수집하여 anomaly 여부를 판별할 수 있었다.
    - iForest와 LSTM Auto-Encoder를 결합한 하이브리드 모델을 구현하여 통계적 이상 탐지와 시계열 패턴 분석의 상호 보완적 특성을 활용하였다.
2. 실시간성 및 사용자 피드백 구현
    - API 응답 시간 50~100ms 수준의 빠른 처리가 가능하며, 안드로이드 앱은 5초 주기 폴링을 통해 사용자에게 탐지 결과를 전달한다.
    - 네트워크 이상 탐지 시 사용자 선택 옵션을, 터치·센서 이상 탐지 시 자동 잠금 기능을 구현하였다.
3. 시각화 및 관리 효율성 확보
    - React 기반 대시보드를 통해 Risk Gauge, Trend Chart, Logs Table, Stats 페이지를 제공하여 anomaly 결과를 직관적으로 시각화하였다
    - 관리자가 탐지 결과를 바탕으로 즉시 분석하고 대응할 수 있는 통합 인터페이스를 구현하였다.

5.3 한계점

1. **데이터 의존성**
    - LSTM 모델은 seq_len=20 이상의 연속 로그가 필요하여 초기 학습 단계에서 지연이 발생하였다.

- Touch Drag 로그는 변동성이 커서 정상 행동에 대한 오탐률이 상대적으로 높았다.

2. **네트워크 모달리티 단순화**
   - GPS 좌표 기반의 단순 범위 체크 방식으로 anomaly를 탐지했기 때문에 VPN, Proxy 기반 위협까지는 커버하지 못하여 안드로이드 앱에서 이를 처리하였다.
3. **실시간성 제약**
   - 폴링(polling) 방식은 안정성이 높지만 WebSocket 기반 push 방식에 비해 anomaly 탐지–피드백 간 지연이 존재한다.

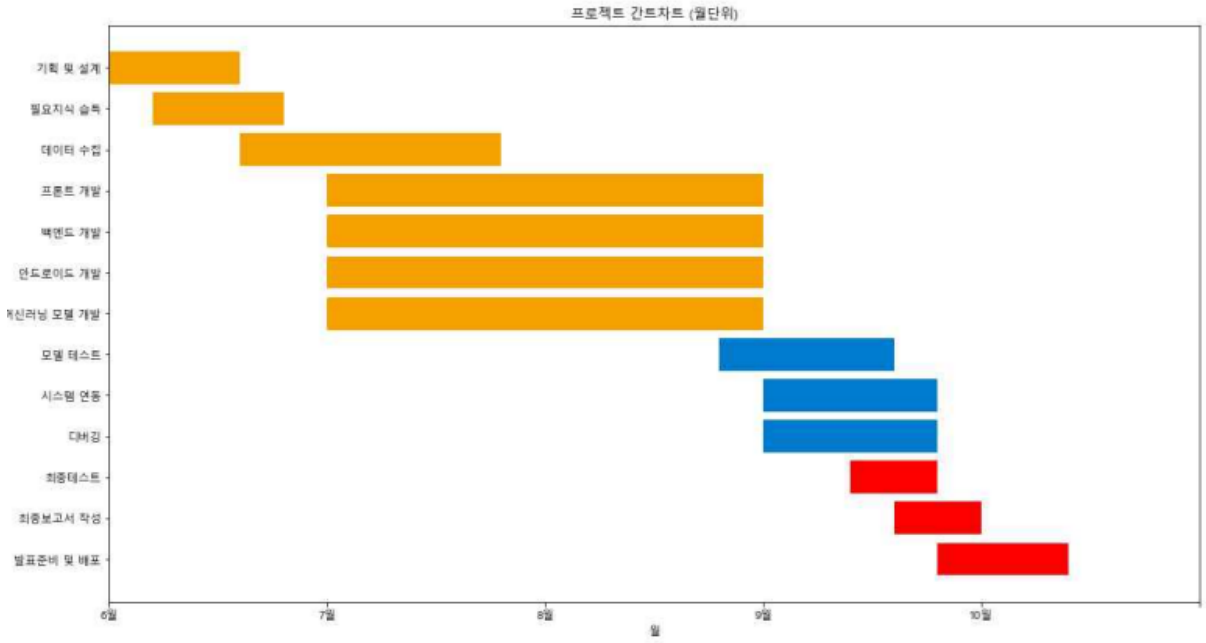
5.4 향후 연구 방향

1. **모델 고도화**
   - LSTM 외에도 GRU, Transformer 기반 이상 탐지 모델을 적용하여 더 복잡한 시계열 패턴까지 학습할 수 있도록 한다.
   - Auto-Encoder에 adversarial training, 데이터 증강 기법을 도입하여 일반화 성능을 강화한다.
2. **네트워크 탐지 강화**
   - GPS 좌표 기반 단순 로직 대신 GeoIP, 기지국 정보, VPN 탐지 기법을 결합하여 네트워크 모달리티 탐지를 정교화한다.
3. **실시간 전송 개선**
   - 현재의 폴링(polling) 방식을 WebSocket 기반 push 알림으로 전환하여 anomaly 발생 즉시 사용자에게 알림을 제공한다.
4. **시스템 확장성**
   - 모바일 외에도 노트북, IoT 기기 등 다양한 엔드포인트 로그를 통합 수집·분석하는 다중 플랫폼 보안 체계로 발전 가능하다.
   - 관리자 대시보드에 정책 제어 기능을 추가하여 탐지–분석–대응을 하나의 인터페이스에서 실시간 수행할 수 있도록 개선한다.

6. 구성원별 역할 및 개발

본 프로젝트는 3인 팀으로 진행되었으며, 각 구성원은 시스템의 주요 모듈을 분담하여 개발하였다.

성명	담당 영역	주요 구현 내용
권태현	대시보드 및 LSTM 모델	- React + TypeScript 기반 관리자 대시보드 개발 - Risk Gauge, Risk Trend Chart, Logs Table, Stats Page 구현 - 관리자용 anomaly 시각화 및 정책 관리 UI 설계 - LSTM Auto-Encoder 모델 구현 및 성능 평가 - Sensor/Touch 로그 전처리 및 특징 추출 -재구성오차(Reconstruction Error) 기반 anomaly 탐지 및 임계치 설정
이승원	백엔드 서버 및 정책 엔진	- Django REST Framework 기반 백엔드 서버 구현 - Behavior 앱: 사용자 로그 수집 및 전처리 - Anomaly 앱: 이상 탐지 결과 저장 및 조회 API 개발 - ML 서버 연동(/predict, /predict_hybrid) 및 anomaly DB 관리 - 네트워크 모달리티 anomaly 판정(GPS 기반 rule 적용) - 안정적인 로그 처리 및 예외 처리 구조 설계
구현서	안드로이드 에이전트 및 iForest 모델	- Android 앱 기반 에이전트 개발 -Touch(Drag/Pressure), Sensor(Accel/Gyro), Network 로그 수집 모듈 구현 -JSON 포맷 변환 및 백엔드 서버 전송 기능 구현 -anomaly 결과 폴링(polling) 기반 알림 처리(잠금화면 전환, 네트워크 제어 선택지 제공) -Isolation Forest 기반 이상 탐지 모델 구현 및 초기 학습/재학습 기능 개발 -VPN/프록시를 통한 네트워크 우회 시도를 탐지하기 위한 IP 주소 변경 모니터링 시스템 구현



6.1 협업 방식

- **GitHub**을 활용한 버전 관리 및 모듈 단위 협업
- **API 명세서 작성**을 통해 백엔드–ML–안드로이드–대시보드 간 연동 규격 통일
- **정기회의**를 통해 진행 상황 공유 및 오류 수정

6.2 개발 환경

- 백엔드: Python (Django, Django REST Framework)
- 머신러닝: Python (FastAPI, scikit-learn, PyTorch)
- 대시보드: React, TypeScript
- 데이터베이스: SQLite / PostgreSQL
- 모바일 에이전트: Android (Java/Kotlin)

7. 참고 문헌

1. Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. *Network Anomaly Detection Using LSTM Based Autoencoder*. 2020.

2. O. I. Provotar, Y. M. Linder, and M. M. Veres. *Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders*. 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 513-517. doi: 10.1109/ATIT49449.2019.9030505.

3. Sensor-Based Continuous Authentication of Smartphones' Users Using Behavioral Biometrics: A Contemporary Survey

4. Mobile User Authentication Using Statistical Touch Dynamics Images

5. A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing

6. A Dynamic Access Control Model Based on Attributes and Intro VAE