악성 네트워크 탐지를 위한 AI 모델 및 XAI 활용

2025. 04. 08

KAIST CSRC Cyber Security Research Center

CYBER SECURITY RESERCH CENTER

목차

Day 2

✓데이터 전처리 & 모델링 기초

시간	주제	내용 개요
10:00 ~ 10:50	설명 가능한 AI (XAI) 개요	- XAI의 필요성 (모델 해석 가능성 문제 해결) - XAI 주요 기법 소개 (LIME, SHAP, Grad-CAM 등)
11:00 ~ 11:50	XAI 기법 적용 실습 ①: LIME & SHAP	- LIME을 활용한 악성 트래픽 탐지 결과 해석 - SHAP을 이용한 중요 피처 분석
12:00 ~ 13:00	점심시간	
13:00 ~ 13:50	XAI 기법 적용 실습 ②: Integrated Gradients & DeepLIFT	- 모델의 예측 근거 분석 - 패킷 수준에서 모델의 의사결정 과정 시각화
14:00 ~ 14:50	하이브리드 탐지 모델 설계(1)	- MLP + BERT 결합 모델 - 네트워크 흐름 기반 특징 추가 (Flow + Payload)
15:00 ~ 15:50	하이브리드 탐지 모델 설계(2)	- 실제 데이터셋을 활용한 실험 - 모델의 취약점 및 보완 방안 논의
16:00 ~ 16:50	토론 및 Q&A	- 모델의 실무 적용 방안 - 한계점 및 향후 연구 방향

0. data labeling


```
label_map = {
    # Banking Trojan (Group 1: Emotet Family)
    'emotet': 'Banking_Emotet_Family',
    'trickbot': 'Banking_Emotet_Family',
    'qakbot': 'Banking_Emotet_Family',
    'qbot': 'Banking_Emotet_Family',
    'ursnif': 'Banking_Emotet_Family',
    'icedid': 'Banking_Emotet_Family',

    # Banking Trojan (Group 2: Zeus Family)
    'zeus': 'Banking_Zeus_Family',
    'panda': 'Banking_Zeus_Family',
    'dridex': 'Banking_Zeus_Family',
    'gootkit': 'Banking_Zeus_Family',
    'bebloh': 'Banking_Zeus_Family',
    'valak': 'Banking_Zeus_Family',
    'buerloader': 'Banking_Zeus_Family',
    'danabot': 'Banking_Zeus_Family',
    'piakbot': 'Banking_Zeus_Family',
    'zloader': 'Banking_Zeus_Family',

    # Banking Trojan (Group 3: Other)
    'bazarloader': 'Banking_Other',
    'bazaloader': 'Banking_Other',
    'astaroth': 'Banking_Other',
    'guildma': 'Banking_Other',
    'svcready': 'Banking_Other',
    'matanbuchus': 'Banking_Other',
    'latrodectus': 'Banking_Other',
    'chthonic': 'Banking_Other',
    'nymaim': 'Banking_Other',
    'vidar': 'Banking_Other',
    'hancitor': 'Banking_Other',
    'darkgate': 'Banking_Other',
    'ta578': 'Banking_Other',
    'ssload': 'Banking_Other',

    # Infostealer
    'lokibot': 'Infostealer',
    'formbook': 'Infostealer',
    'agenttesla': 'Infostealer',
    'raccoon': 'Infostealer',
    'cred-stealer': 'Infostealer',
    'ftp': 'Infostealer',
    'socgholish': 'Infostealer',
    'azorult': 'Infostealer',
    'metastealer': 'Infostealer',
    'snake-keylogger': 'Infostealer',
    'lumma-stealer': 'Infostealer',
    'meduza-stealer': 'Infostealer',
    'rhadamanthys': 'Infostealer',
    'fakebat': 'Infostealer',
    'dev-0569': 'Infostealer',
    'agent-tesla': 'Infostealer',

    # RAT
    'remcos': 'RAT',
    'nanocor': 'RAT',
    'netwire': 'RAT',
    'bitrat': 'RAT',
    'revenge': 'RAT',
    'rat': 'RAT',
    'flawedammyy': 'RAT',
    'flawed-ammyy': 'RAT',
    'medusahttp': 'RAT',
    'flawedgrace': 'RAT',

    # Ransomware
    'gandcrab': 'Ransomware',
    'shade': 'Ransomware',

    # Exploit Kit
    'exploit': 'Exploit_Kit',
    'ek': 'Exploit_Kit',
    'neutrino': 'Exploit_Kit',
    'rig': 'Exploit_Kit',
    'eitest': 'Exploit_Kit',
    'afraidgate': 'Exploit_Kit',
    'log4j': 'Exploit_Kit',
    'cve-2024-4577': 'Exploit_Kit',
    # Cobalt Strike
    'cobalt': 'Cobalt_Strike',
    'systembc': 'Cobalt_Strike',
    'backconnect': 'Cobalt_Strike',
    'keyhole': 'Cobalt_Strike',
    'sliver': 'Cobalt_Strike',
    # Malspam
    'mirrorblast': 'Malspam',
    'google-ad': 'Malspam',
}
```

0. data labeling


```
✓ # 라벨 통합 매핑 다시 적용
✓ grouping_map = {
✓     'Benign': 'Benign',
✓     'Banking_Emotet_Family': 'Banking_Emotet_Family',
✓     'Banking_Zeus_Family': 'Banking_Zeus_Family',
✓     'Banking_Other': 'Banking_Other',
✓     'Infostealer': 'Infostealer',
✓     'RAT': 'RAT',
✓     'Exploit_Kit': 'Exploit_Kit',
✓     'Cobalt_Strike': 'Cobalt_Strike',
✓     'Malspam': 'Malspam_Phishing',
✓     'Phishing': 'Malspam_Phishing',
✓     'Dropper_Loader': 'Malspam_Phishing',
✓     'Execution_Traffic': 'Malspam_Phishing',
✓     'Generic_Malware_Traffic': 'Other_Generic',
✓     'Post_Infection_Traffic': 'Other_Generic',
✓     'Unknown_Malware': 'Other_Generic',
✓     'Rootkit': 'Other_Generic',
✓     'Ransomware': 'Ransomware',
✓     'Reconnaissance': 'Recon_C2',
✓     'C2_Traffic': 'Recon_C2'
✓ }
```

```
✓ # 라벨 ID 매핑
✓ label_to_id = {
✓     'Benign': 0,
✓     'Banking_Emotet_Family': 1,
✓     'Banking_Zeus_Family': 2,
✓     'Banking_Other': 3,
✓     'Infostealer': 4,
✓     'RAT': 5,
✓     'Exploit_Kit': 6,
✓     'Malspam_Phishing': 7,
✓     'Ransomware': 8,
✓     'Recon_C2': 9,
✓     'Other_Generic': 10,
✓     'Cobalt_Strike': 11
✓ }
```

공격 유형 기반 라벨링: 악성코드 계열 분류 체계

✓정확한 탐지를 위해 악성 네트워크 트래픽을 기능 및 행위 기반으로 라벨링 진행

✓총 7개 주요 범주로 구성되며, 각 범주는 실제 악성코드 계열 또는 공격 수법을 기준으로 세분화

- Banking Trojan 계열은 Emotet, Zeus 등의 대표 악성코드를 기준으로 그룹화
- InfoStealer, RAT, Ransomware, Exploit Kit 등은 행위 기반 악성 분류
- 일부 공격도구(Cobalt Strike, Metasploit 등) 및 패턴 기반 라벨도 포함

✓이러한 정교한 라벨링은 모델이 단순한 탐지를 넘어 공격 유형을 식별 하도록 학습

- Emotet/Zeus 계열은 각각의 파생형이 많아서 Group 1 / Group 2로 따로 분류
- 'Banking_Other' 같은 라벨은 명확히 Banking 기능을 가지지만 메이저 계열에 속하지 않는 것
- InfoStealer와 RAT은 기능 기반 분류이고, 실제 사용된 악성코드 이름 기준으로 수집
- 마지막에는 'Exploit Kit', 'Cobalt Strike', 'Malspam' 등 공격 도구/전파 방식 기반의 분류도 추가
- 이렇게 라벨링함으로써, 단순한 악성/정상 구분이 아닌 공격 목적/수단을 식별하는 모델로 확장 가능

공격 유형 기반 라벨링: 악성코드 계열 분류 체계

✓다양한 악성코드 계열 및 공격 유형을 바탕으로 총 12개 라벨 클래스를 정의

✓각 라벨은 기능적 특성에 따라 그룹핑

- 'Benign'은 0번으로 고정되어 정상 트래픽을 명확히 구분
- 이후 라벨은 공격 행위/유형 기준으로 세분화 (예: Phishing, RAT, Recon, Exploit 등)
- 클래스 ID는 전체 탐지 모델 및 XAI 해석 시 해석 가능성을 높이기 위한 구조로 설계됨

범주	ID 범위	예시 라벨
정상	0	`'Benign'`
금융 악성코드	1–3	`'Banking_Emotet_Family'`, `'Banking_Zeus_Family'`
정보 탈취 / 제어	4–5	`'Infostealer'`, `'RAT'`
공격 도구 / 유포 방식	6–7	`'Exploit_Kit'`, `'Malspam_Phishing'`
파괴형 공격	8	`'Ransomware'`
기타 기능	9–11	`'Recon_C2'`, `'Cobalt_Strike'`

설명 가능한 AI (XAI)

1. 설명 가능한 AI (XAI) 개요

✓XAI란?

- XAI (Explainable AI): 모델의 의사결정 과정을 사람이 이해할 수 있는 방식으로 설명해주는 기술
- 블랙박스 모델의 해석 가능성 문제 해결을 목표로 함

✓필요성

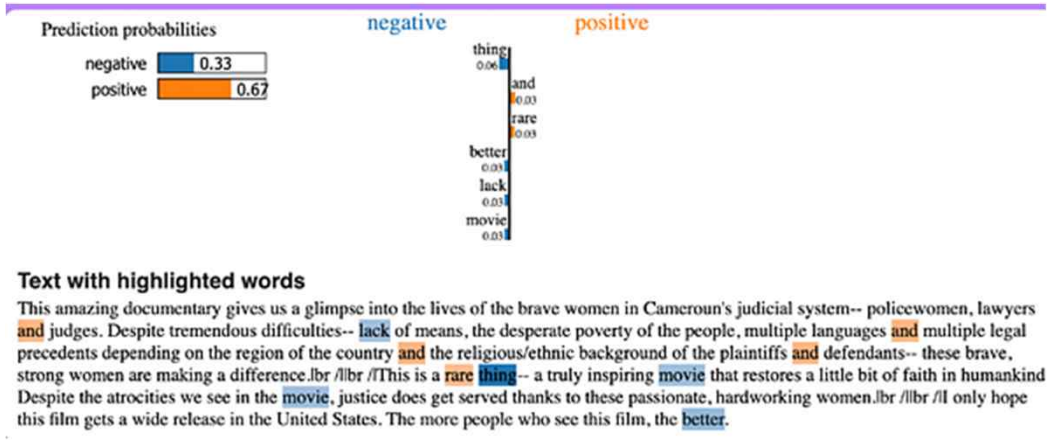
- 보안 탐지 분야에서는 모델의 탐지 근거가 중요
- 규제/감사 대응 필요 (e.g., 금융, 의료, 국방)
- 잘못된 예측 결과에 대해 설명과 개선 방향 제시 가능

기법	설명	특징
LIME	로컬 선형 모델로 예측 해석	단일 샘플 기반, 직관적
SHAP	게임 이론 기반 중요도 분석	전역 + 지역 해석 모두 가능
Integrated Gradients	경사값 누적 기반 기여도 측정	딥러닝 적합
DeepLIFT	베이스 라인 대비 뉴런 기여도 추적	복잡한 구조도 해석

2. LIME과 SHAP 기법

✓LIME (Local Interpretable Model-Agnostic Explanations)

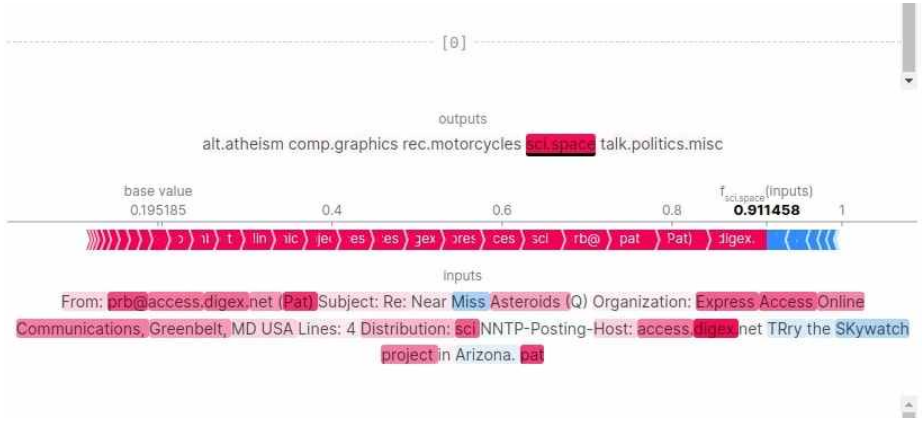
- 개념
  - 입력 샘플 주변에 유사한 가짜 데이터 생성 → 모델의 예측값 관찰
  - 로컬 선형 모델을 학습하여 해당 예측의 근거가 되는 피처 중요도 추정
- 핵심 특징
  - 모델 불문 (Model-Agnostic)로컬
  - 해석 중심 → 개별 샘플 예측 설명에 강함
- 사용 예시
  - "이 패킷을 악성으로 판단한 이유는?"
  - 텍스트 기반 악성 로그 탐지에서 어떤 단어가 중요한지 시각화



2. LIME과 SHAP 기법

✓SHAP (SHapley Additive exPlanations)

- 개념
  - 게임 이론의 Shapley Value를 기반으로 각 피처가 예측에 기여한 정도를 계산
  - 모델 전체에 대한 전역적 해석과 샘플별 지역 해석 모두 가능
- 핵심 특징
  - 정량적이고 이론적으로 안정적인 설명
  - 의사결정 트리, XGBoost 등에서 최적화된 계산 지원순서에 관계 없이 피처 기여도 측정 가능
- 사용 예시
  - 보안 이벤트 로그에서 "가장 영향력 있는 피처" 분석
  - 피처별로 악성/정상 여부에 얼마나 영향을 주는지 시각화 가능



2. LIME과 SHAP 기법

✓LIME vs SHAP 비교

항목	LIME	SHAP
접근 방식	로컬 모델 근사	Shapley Value 기반 수학적 계산
해석 범위	샘플 단위(로컬)	샘플 + 전체(로컬 + 전역)
계산 복잡도	상대적으로 낮음	상대적으로 높음(Tree SHAP 제외)
직관성	직관적인 시각화	수치적이고 정밀한 해석

✓준비 사항사전

- 학습된 모델이 필요 (e.g., 악성 탐지 BERT or XGBoost)
- 피처 중요도를 확인할 수 있는 탐지 결과 데이터셋 확보
- Python 라이브러리: lime, shap, scikit-learn, pandas, matplotlib

3. IG와 DeepLIFT

✓Integrated Gradients (IG)

- 개념
  - 모델의 입력과 기준선(baseline) 사이를 선형 보간하면서 기울기(gradient)를 누적
  - 특정 입력 피처가 예측에 얼마나 기여했는지 정량화함
- 작동 방식
  - 입력 x, 기준선 x' 사이를 $\alpha \in [0,1]$ 로 보간
  - 누적 기울기: IGi (x)=(xi−xi′)×∫α=01 ∂i ∂F(x′+α(x−x′))dα
- 기준선 예시
  - 텍스트: [PAD] 또는 빈 입력
- 장점
  - 기하학적이고 직관적인 방식
  - 입력 차원별로 기여도를 계산 가능 (시각화 적합)
- 적용 예시
  - BERT 입력에서 어떤 토큰이 예측에 영향을 줬는가
  - 이 네트워크 패킷의 어떤 필드가 악성 탐지에 중요했나?

3. IG와 DeepLIFT

✓DeepLIFT (Deep Learning Important FeaTures)

- 요약
  - 기준 입력 대비 현재 입력이 출력에 얼마나 기여했는지를 추적
  - 순전파 시 모든 뉴런의 "기여값"을 계산하여 각 입력 피처로 할당
- 작동
  - 기준선 x'과 실제 입력 x 간의 출력 차이를 각 뉴런의 활성화 차이를 통해 역추적
- 기본 아이디어
  - Gradient를 사용하는 대신 출력 변화량 (Δy)을 기준으로 입력의 변화량(Δx)과의 관계를 통해 기여도를 계산함
- 장점
  - Gradient가 0일 때도 의미 있는 결과 제공
  - ReLU 같은 비선형 함수에서 Gradient 사라지는 문제 극복

3. IG와 DeepLIFT

✓IG vs DeepLIFT 비교

항목	Integrated Gradients	DeepLIFT
방식	누적 기울기 기반	활성화 변화량 기반 기여 추적
기준선 필요	Yes	Yes
Gradient 필요	Yes	No(비교 기반 추론)
Gradient vanishing 보완	No	Yes
사용 에	NLP, Vision, Tabualr등	Vision, NLP 내부 Layer 해성 등

✓준비 사항사전

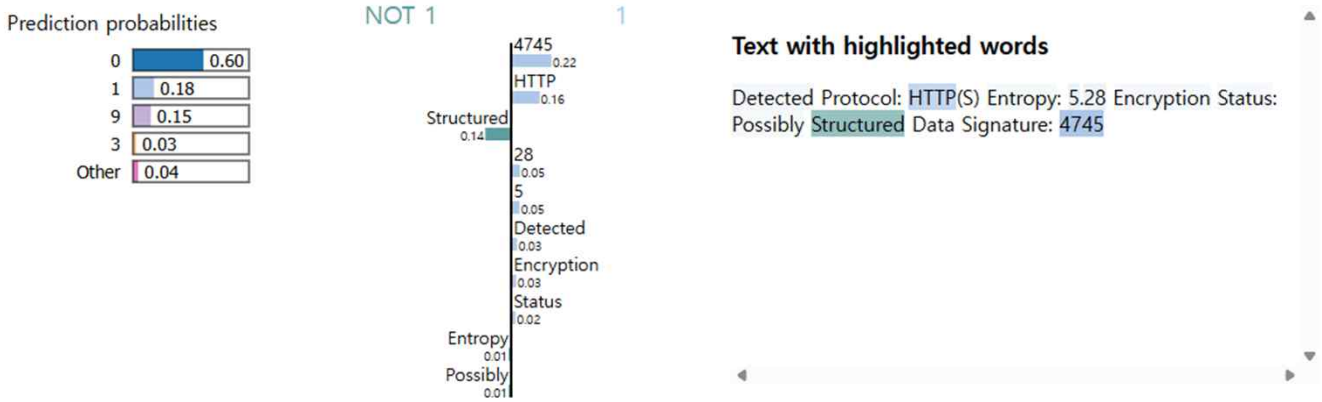
- 학습된 모델이 필요 (e.g., 악성 탐지 BERT or XGBoost)
- 피처 중요도를 확인할 수 있는 탐지 결과 데이터셋 확보
- Python 라이브러리: captum

XAI 기법 적용 실습 ①: LIME & SHAP

1. LIME

✓실행: python lime_practice.py --eval_model_type NLP --MAXLEN 100

✓출력: lime_explanation.html



좌측: Prediction probabilities (예측 확률)

- 클래스 0이 0.60 (60%)로 가장 높은 확률
- 나머지 클래스는 신뢰도 낮음 (1, 9, 3은 각각 18%, 15%, 3%)

가운데: 단어별 기여도 시각화 (Feature importance bar plot)

- 오른쪽으로 길게 뻗은 막대일수록 → 그 단어가 해당 클래스 예측에 긍정적으로 많이 기여
- 반대로 왼쪽으로 가는 막대 → 반대로 예측을 방해하는 요인이라는 뜻

우측: Text with highlighted words

- 실제 입력 문장에서, 모델이 중요하다고 판단한 단어에 색상 하이라이팅
- 색이 진할수록 → 예측에 영향이 큰 단어

XAI 기법 적용 실습 ①: LIME & SHAP

1. LIME(Tabular)

✓실행: python lime_tabular_practice.py

✓출력: lime_tabular_explanation.html

좌측: Prediction probabilities (예측 확률)　중앙: 각 피처가 특정 클래스에 얼마나 영향을 줬는지　우측: 해당 샘플의 실제 입력 값

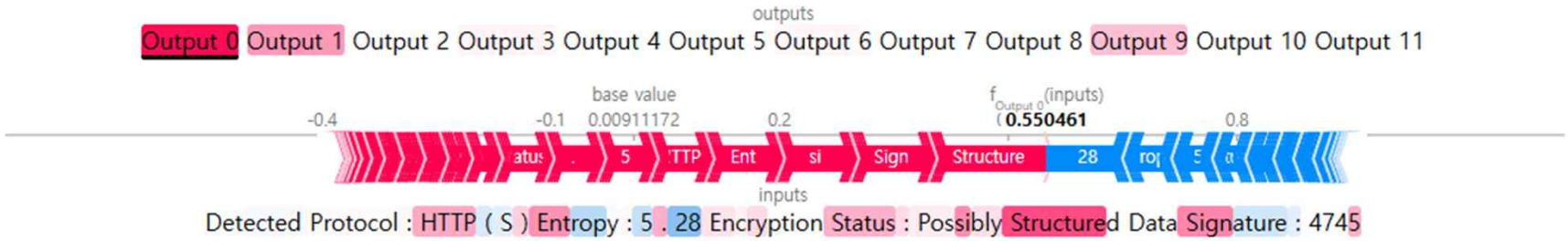
- 클래스 6으로 판별

XAI 기법 적용 실습 ①: LIME & SHAP

2. SHAP(지역)

✓실행: python shap_practice.py --eval_model_type NLP --MAXLEN 100

✓출력: shap_explanation.html



전체 구조

- 상단: 모델이 예측할 수 있는 클래스들

SHAP 값 해석

- 그래프의 중심선: 모델의 기본 예측값(base value), 즉 입력이 없을 때의 평균 예측
- 오른쪽(파란색): 예측 값을 증가시키는 단어들
- 왼쪽(분홍색): 예측 값을 감소시키는 단어들

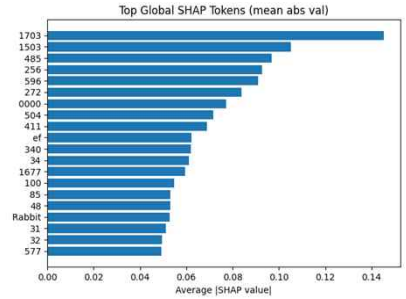
단어	영향 방향	설명
4745	증가	이 값은 패킷의 특정 시그니처로, 모델이 악성일 가능성을 높게 본 요소
Structure	증가	구조화된 데이터나 패턴 존재는 악성으로 분류될 가능성을 증가시킴
Status , Sign	감소	명확한 인증이나 정상적인 메시지일 경우, 악성으로 볼 가능성이 낮아짐
HTTP , Entropy	감소	정형화된 프로토콜 + 낮은 엔트로피 등은 악성이 아닐 수도 있음

XAI 기법 적용 실습 ①: LIME & SHAP

2. SHAP(전역)

✓실행: python shap_practice.py --eval_model_type NLP --MAXLEN 100

✓출력: shap_text_global_manual.png, shap_summary_style_plot.png



목적

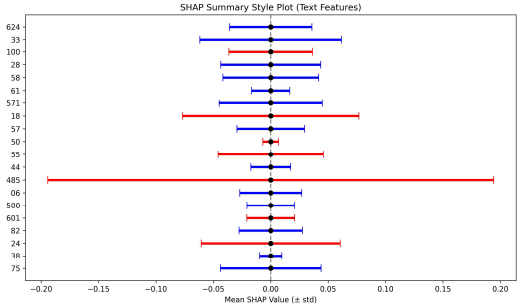
- 모델이 전반적으로 어떤 토큰에 가장 민감하게 반응하는지를 파악
- SHAP의 절대값 평균값을 기반으로 정렬 → 영향력 순위

해석

x축: Average |SHAP value| → 기여도의 강도

y축: 토큰 → 입력 데이터에서 등장한 단어 또는 숫자

막대 길이: 해당 토큰이 모델 예측에 끼친 영향력의 평균 절댓값



목적

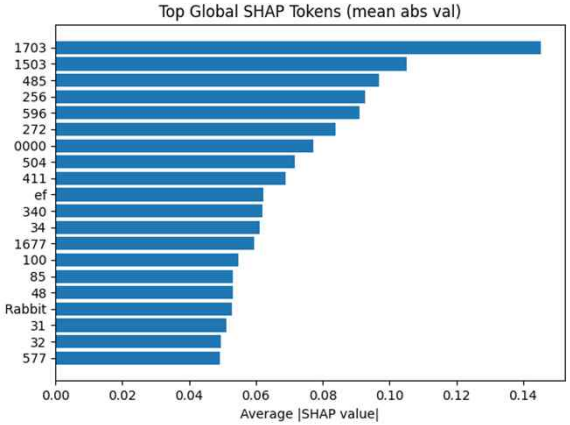
- 토큰이 예측에 긍정적인 영향을 주는지(↑) 또는 부정적인 영향을 주는지(↓) 를 보여줌
- SHAP의 평균값과 분산(std) 모두 표현

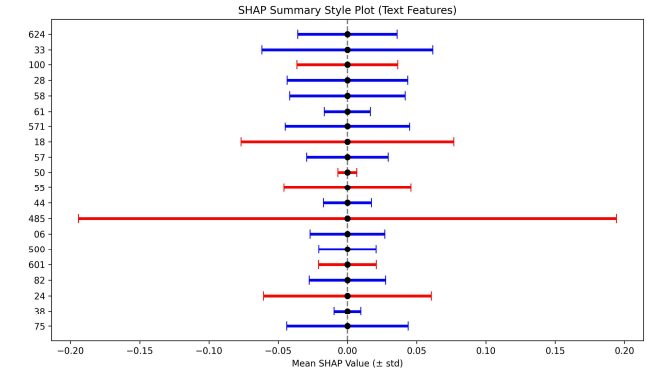
해석

- x축: Mean SHAP Value (± std) → 실제로 모델 예측에 어떤 방향으로 영향을 주는지
  - 오른쪽(+): 해당 클래스로 예측되도록 만듦 (e.g. 악성 판단 강화
  - 왼쪽(−): 예측을 반대로 끌어내림 (e.g. 악성 판단 약화)
- y축: 토큰 인덱스 (실제로는 토큰 이름으로 바꾸는 게 일반적)
- 점: 평균 SHAP 값
- 에러바: 샘플 간 기여도의 표준편차 (불확실성)

XAI 기법 적용 실습 ①: LIME & SHAP

2. SHAP(전역)





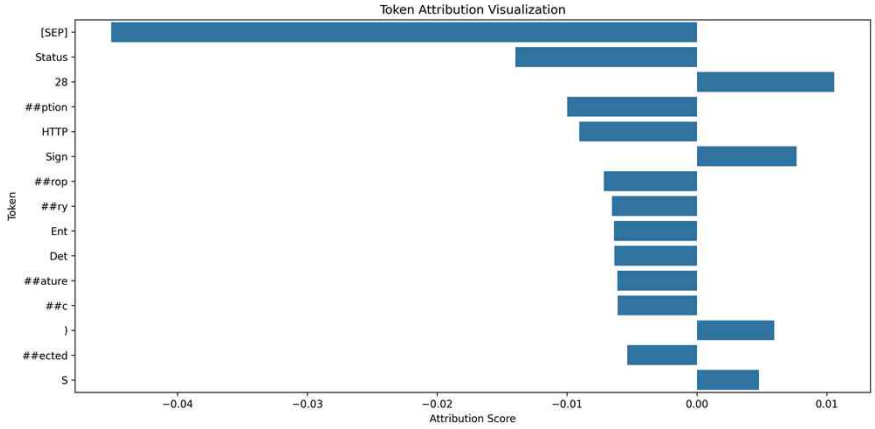
항목	Bar Plot	Summary Style Plot
기준	평균 절대 SHAP 값	평균 SHAP 값 ± std
방향성	❌ 없음 (절댓값만)	✅ 있음 (+/-)
분산 정보	❌ 없음	✅ 에러바로 표현
해석 목적	어떤 토큰이 제일 중요했는가	어떤 방향으로 예측에 영향을 미쳤는가

XAI 기법 적용 실습 ①: LIME & SHAP

3. IG

✓실행: python captum_practice.py --eval_model_type NLP --MAXLEN 100 --captum_method ig

✓출력: ig_attribution_plot.png



BERT 분류 모델이 입력 문장을 분류할 때,각 토큰이 예측에 기여한 정도를 보여줌

Y축 (Token): BERT가 처리한 토큰들 (WordPiece 단위)

- ##c, ##ected, Sign → WordPiece 분할로 인한 형태

X축 (Attribution Score)

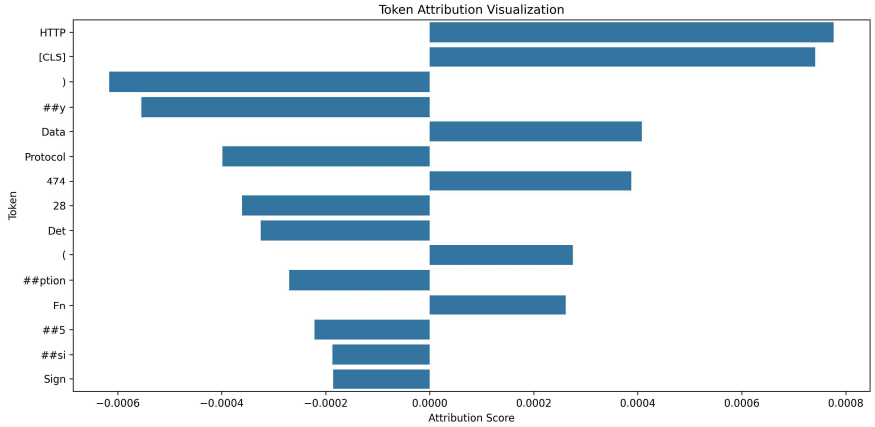
- 해당 토큰이 예측 결과에 기여한 정도양수
- 예측 클래스(예: 악성)에 긍정적인 기여
- 음수: 예측 클래스에 부정적인 기여

XAI 기법 적용 실습 ①: LIME & SHAP

4. deepLIFT

✓실행: python captum_practice.py --eval_model_type NLP --MAXLEN 100 --captum_method deepLIFT

✓출력: deepLIFT_attribution_plot.png



✓ 입력 문장에서 각 토큰이 모델의 최종 예측에 얼마나 영향을 주었는지 분석 → 즉, 어떤 단어가 "이게 악성/정상이다" 라고 모델을 설득하는 데 영향을 끼쳤는지

Y축 (Token): BERT가 처리한 토큰들 (WordPiece 단위)

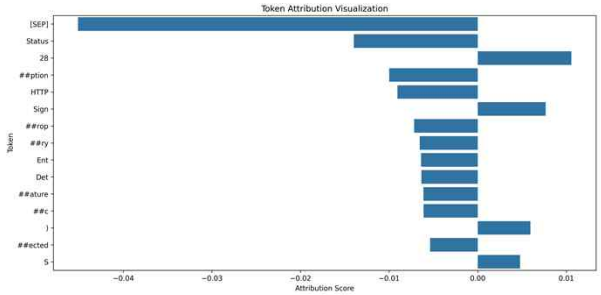
X축 (Attribution Score)

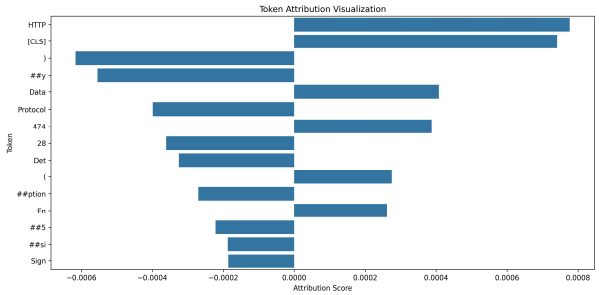
- 해당 토큰이 예측 결과에 기여한 정도양수
- 예측 클래스(예: 악성)에 긍정적인 기여
- 음수: 예측 클래스에 부정적인 기여

XAI 기법 적용 실습 ①: LIME & SHAP

5. IG vs deepLIFT

- ✓ **Integrated Gradients vs DeepLIFT**
- ✓ **IG (Integrated Gradients): 입력 → 기준값 사이 경로에서의 기울기를 누적하여 기여도 산출**
- ✓ **DeepLIFT: 기준값 대비 뉴런의 변화량을 추적하여 기여도 산출**
- ✓ **결과 비교**
  - IG는 스페셜 토큰과 같이 문장 구조적 요소에도 민감하게 반응
  - DeepLIFT는 실제 예측 결정에 핵심적인 피처를 중심으로 기여도를 계산함
  - 두 기법을 병행 사용하면 더 풍부한 해석이 가능
  - IG: 모델이 어디에 주목하는가 확인
  - DeepLIFT: 실제 분류 근거로 작용한 요소 확인





하이브리드 탐지 모델 설계(1)

1. Hybrid model의 필요성

✓네트워크 보안은 현대 디지털 환경에서 중요한 요소

✓악성 네트워크 활동은 조직과 개인에게 심각한 위협을 가하며, 정보 유출, 시스템 파괴, 금전적 손실 등 다양한 문제를 야기

✓증가하는 사이버 위협에 대비하기 위해 효과적인 탐지 및 방어 시스템이 필요

✓기존 탐지 방법의 한계

- Flow 기반 탐지의 한계
  - 네트워크 트래픽 흐름 정보를 활용하여 비정상적인 패턴을 탐지하는 Flow 기반 방식은 상대적으로 리소스 소모가 적고 빠르게 탐지할 수 있는 장점이 있지만, 트래픽의 내용을 직접 분석하지 않기 때문에 최신 공격 유형을 정확히 탐지하기에 어려움
  - 암호화된 트래픽이나 패턴이 명확하지 않은 공격을 탐지하는 데 취약하다는 단점이 존재
- Payload 기반 탐지의 한계:
  - 트래픽의 페이로드(내부 데이터)를 분석하는 Payload 기반 탐지 방식은 패킷 내용의 특정 패턴을 통해 악성 여부를 정교하게 판단 가능
  - 그러나 리소스 소모가 크고 처리 속도가 느리며 암호화된 데이터의 경우 탐지에 어려움이 있음
  - 이러한 방식은 고도의 리소스를 요구하고, 트래픽의 증가에 따라 확장성이 떨어짐

하이브리드 탐지 모델 설계(1)

1. Hybrid model의 필요성

✓하이브리드 모델의 필요성

- Flow 기반과 Payload 기반 탐지의 결합
  - Flow 기반 탐지와 Payload 기반 탐지를 결합한 하이브리드 모델은 각 접근법의 장점을 결합하여 더 높은 정확도와 효율성을 제공
  - Flow 기반 탐지의 빠른 처리와 리소스 절약 장점, Payload 기반 탐지의 정밀한 탐지 능력을 동시에 활용 가능 예상
  - 하이브리드 모델은 다양한 공격 패턴을 포괄적으로 탐지할 수 있어 기존의 단일 탐지 방식보다 악성 네트워크에 대한 탐지 성능이 향상
- 다양한 악성 네트워크 위협에 대한 대응력 향상
  - AI와 NLP 모델을 이용한 하이브리드 접근은 복잡한 네트워크 환경에서도 악성 활동을 더 효과적으로 탐지할 수 있도록 해줍니다

하이브리드 탐지 모델 설계(1)

1. Hybrid model의 필요성

항목	Flow 기반 탐지	Payload 기반 탐지
분석 대상	패킷의 메타데이터 (IP 주소, 포트, 프로토콜 등)	패킷의 내용(페이로드)
주요 탐지 방식	트래픽 흐름과 패턴을 분석하여 비정상적인 연결이나 트래픽 증가 탐지	데이터 내용에서 악성 코드, 명령어, 스크립트 등 의심 요소 탐지
장점	- 가벼운 리소스 소모 - 실시간 탐지에 적합 - 암호화된 트래픽에서도 메타데이터 분석 가능	- 정밀한 탐지 가능 - 다양한 위협 요소 식별 - 비정상적인 데이터 탐지에 유리
단점	- 세부적인 데이터 분석이 불가능 - 패턴이 불명확한 고급 공격 탐지에 한계	- 암호화된 트래픽 분석 어려움 - 높은 자원 소모로 인한 성능 저하 가능 - 대규모 네트워크에서 실시간 탐지 어려움
사용 시나리오	- 실시간으로 트래픽 이상 징후를 감지해야 하는 경우 - 대량의 네트워크 트래픽을 효율적으로 모니터링할 때	- 패킷 내용 기반의 정교한 악성 코드 탐지가 필요한 경우 - 특정 패턴이나 명령어를 탐지해야 할 때
보완 필요성	Payload 기반 탐지와 결합하여 세부 데이터 분석 보완	Flow 기반 탐지와 결합하여 암호화된 트래픽 탐지 보완

하이브리드 탐지 모델 설계(1)

1. Hybrid model의 필요성

✓하이브리드 탐지란?

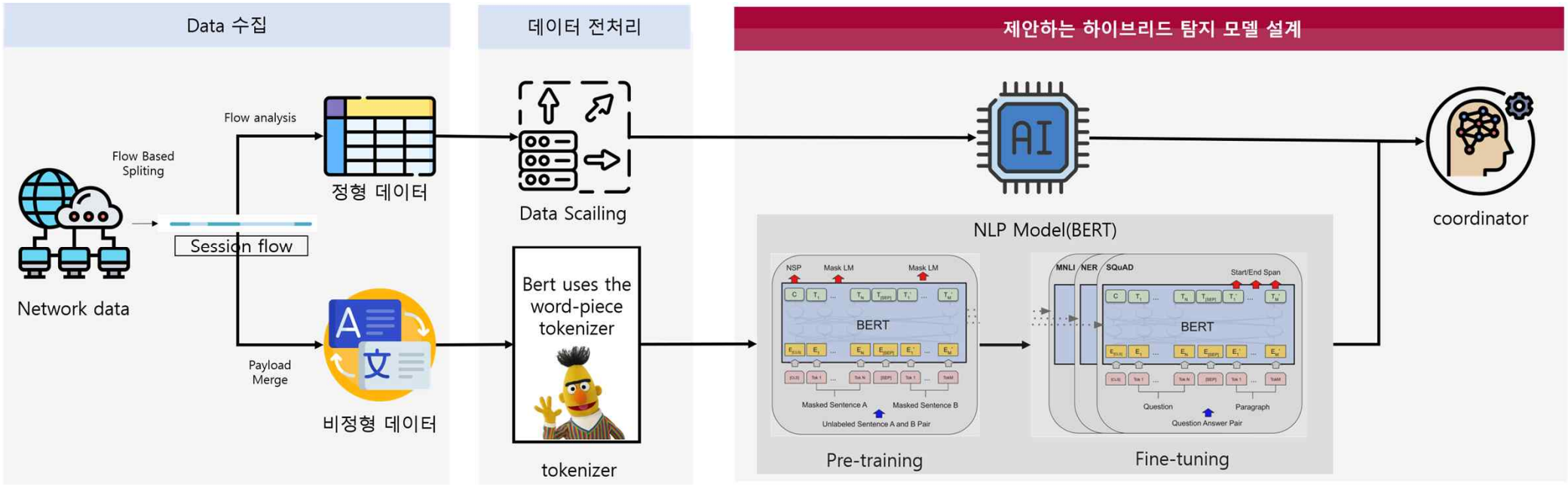
- 하이브리드 탐지는 Flow 기반 탐지와 Payload 기반 탐지의 장점을 결합하여, 악성 네트워크 활동을 더 효과적으로 탐지하는 방식
- 각 방식의 단점을 보완하여 암호화된 트래픽, 세부 데이터 분석의 한계 등을 극복하고 더 높은 탐지 정확도를 제공

✓하이브리드 탐지 모델의 구조

- Flow 기반 탐지 모듈: 트래픽의 메타데이터 학습
- Payload 기반 탐지 모듈: 패킷의 내용과 패턴을 분석 학습
- Coordinator 모듈: 두 탐지 모듈에서 얻은 결과를 바탕으로 신뢰도 기반의 최종 판단을 수행하고, 하이브리드 모델을 생성하여 탐지 결과를 종합

하이브리드 탐지 모델 설계(1)

2. Hybrid model



하이브리드 탐지 모델 설계(1)

2. Hybrid model

✓데이터 수집 (왼쪽 영역)

- 저장된 네트워크 패킷 로그로부터 데이터 수집
- Flow-based Splitting을 통해 세션 단위로 분할
- 분할된 세션은 두 가지로 나뉨
  - 정형 데이터: flow-level 통계 (ex. duration, packet size 등)
  - 비정형 데이터: payload 자체 (텍스트처럼 간주하여 처리)

✓데이터 전처리 (가운데 영역)

- 정형 데이터는 Scaling 및 feature 정제를 수행
- 비정형 데이터는 BERT tokenizer를 통해 WordPiece 기반으로 토큰화→ 이는 BERT 기반 NLP 처리에 적합한 형태로 변환하는 과정

✓모델 설계 (오른쪽 영역)

- 두 데이터를 통합하는 하이브리드 탐지 모델 구조 설계
- 비정형 데이터는 BERT 모델을 통해 fine-tuning
- 최종적으로 AI 모델이 coordinator 역할을 하여 통합 판단 결과 도출

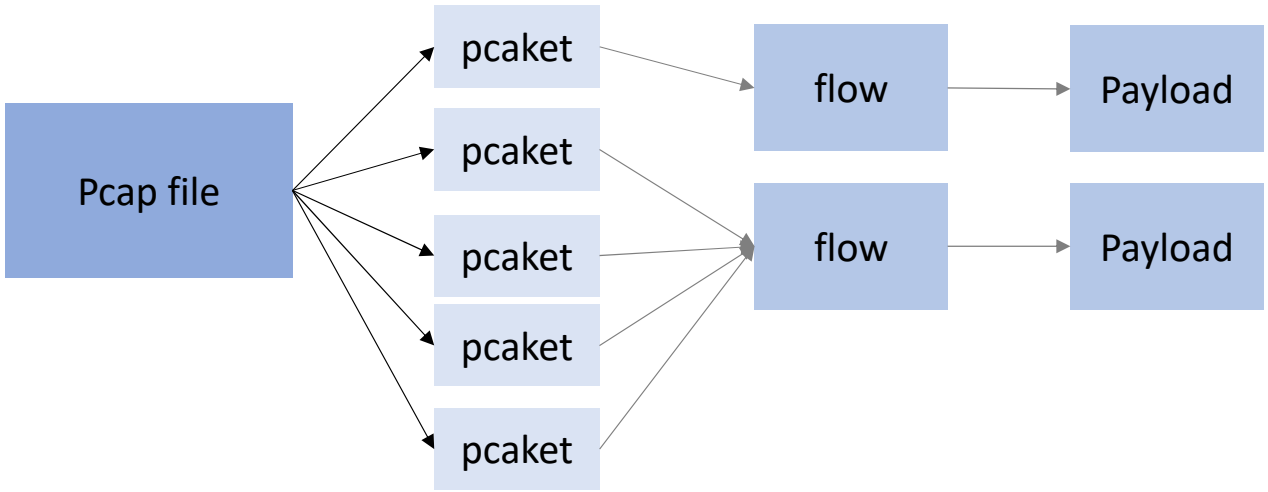
하이브리드 탐지 모델 설계(1)

3. 데이터 전처리

✓Pcap 파일로부터 packet 데이터 추적

✓Packet to flow 단위 변경

✓Payload 데이터를 packet 단위가 아닌 flow 단위로 결합



하이브리드 탐지 모델 설계(1)

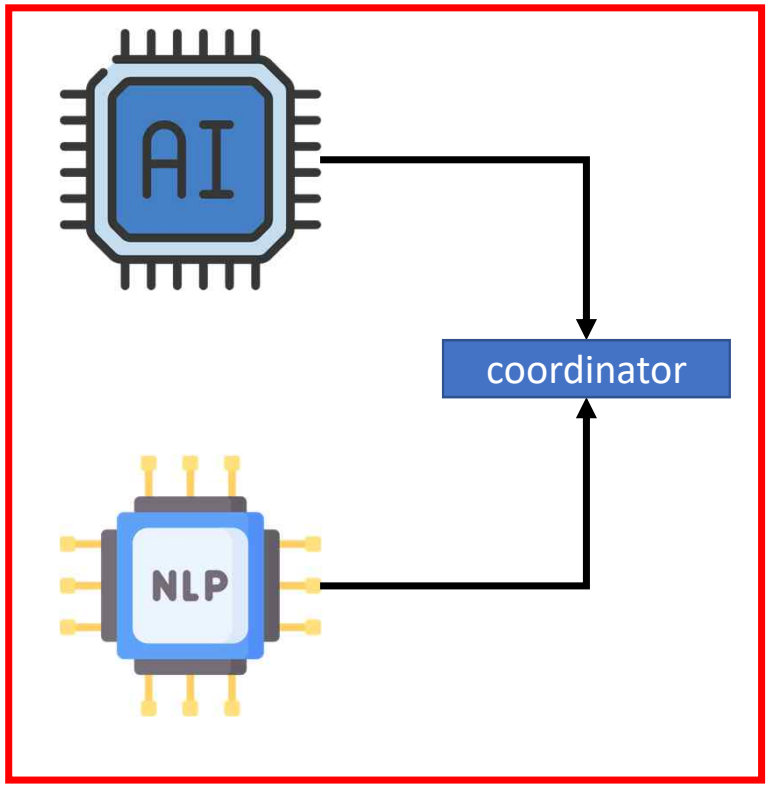
4. coordinator

✓모든 네트워크 흐름이 Payload를 포함하는 것은 아니기 때문에,

✓모델은 우선적으로 정형 데이터 기반의 AI 모델 결과를 활용합니다.

✓하지만 Payload가 존재할 경우, NLP 기반 분석 모델의 결과와 비교하여

✓최종적으로 coordinator가 판단을 내리는 하이브리드 탐지 전략을 채택합니다.

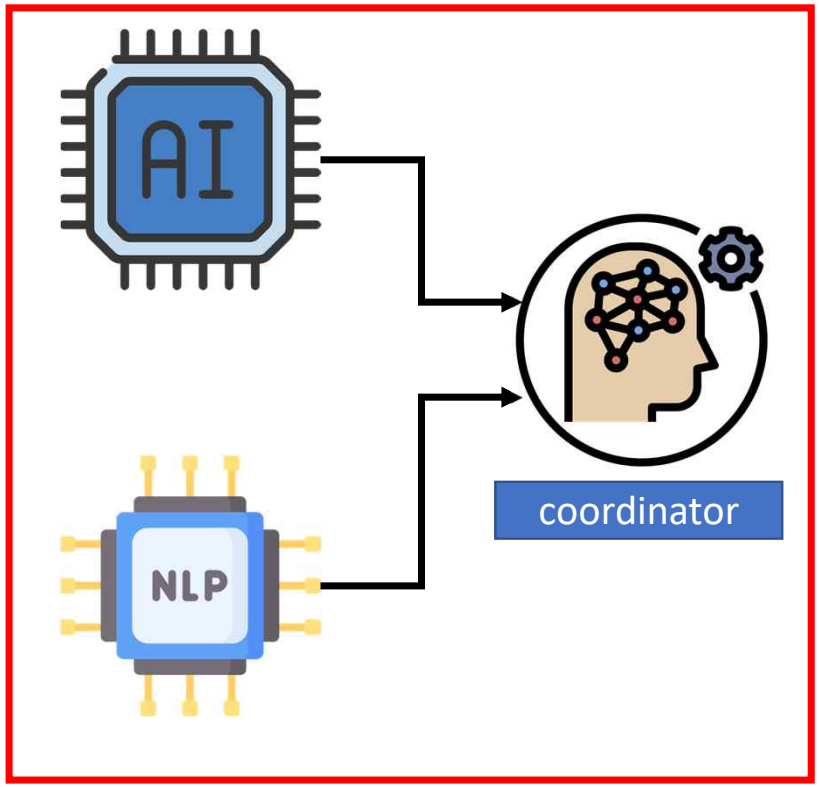


하이브리드 탐지 모델 설계(2)

1. 모델 구조

✓각 모델에서 AI를 활용한 feature extraction(or dimensionality reduction) 진행 후 새로운 모델을 생성하는 functional modeling

✓정형 데이터 모델에서 MLP와 같은 DL 기법 사용



하이브리드 탐지 모델 설계(2)

2. 데이터 구조

✓데이터 구조

- MLP Input (Tabular Data)79의 Feature에서 시작
- Dense Layer를 통해 256 차원으로 확대 후 Dense Layer로 128 차원으로 축소
- BERT Output (NLP Data)텍스트 데이터를 통해 768 차원 벡터 생성

✓데이터 결합

- MLP와 BERT의 Feature 결합MLP의 128 차원 출력과 BERT의 768 차원 벡터를 Concatenate.
- 결합된 896 차원의 데이터로 학습 진행.

```
# Tabular 데이터를 위한 MLP
self.tabular_mlp = torch.nn.Sequential(
    torch.nn.Linear(tabular_input_dim, 256),
    torch.nn.ReLU(),
    torch.nn.Dropout(0.3),
    torch.nn.Linear(256, 128),
)
# BERT와 Tabular의 출력을 결합
combined_dim = self.bert.config.hidden_size + 128  # BERT hidden_size + MLP output size
self.fc = torch.nn.Sequential(
    torch.nn.Linear(combined_dim, 128),
    torch.nn.ReLU(),
    torch.nn.Dropout(0.3),
    torch.nn.Linear(128, output_numbers)
)
```

하이브리드 탐지 모델 설계(2)

3. 학습 단계

✓학습 단계

- 결합된 Feature를 기반으로 Fully Connected Layer 설계.
- 다양한 악성 네트워크 탐지 작업에 최적화된 Loss Function 사용.
- 효과적인 학습을 위해 Batch Normalization 및 Dropout 적용

✓기대 효과

- Flow 기반 데이터와 Payload 기반 데이터의 융합으로 탐지 성능 향상
- 대규모 네트워크 트래픽에서도 높은 확장성과 정밀도 제공
- BERT의 NLP 이해력과 MLP의 Tabular 데이터 처리 능력을 활용한 포괄적 탐지 모델

하이브리드 탐지 모델 설계(2)

4. 모델의 한계 및 보완 전략

✓ 모델 취약점 (Weaknesses)

- Payload 미존재 시 탐지 한계
  - NLP 모델은 비활성화되어 AI(Multi-layer Perceptron) 단독 판단에 의존
- 두 모델 간 학습 데이터 불균형
  - 비정형 데이터가 부족하거나 편향되면 BERT 출력의 신뢰도가 낮아짐
- 연결된 벡터(896-d)의 feature scaling 불균형
  - MLP(128-d)와 BERT(768-d) 차원 분포가 달라 학습 편향 우려
- Payload 악성 회피(ex: 암호화) 시 NLP 모델 무력화 가능성

하이브리드 탐지 모델 설계(2)

4. 모델의 한계 및 보완 전략

✓보완 방안 (Improvement Points)

- Payload 존재 유무에 따른 동적 가중치 조정→ attention 또는 gating mechanism으로 BERT/MLP 결과 반영 비율 조절
- 비정형 데이터 부족 대응을 위한 Pre-training 데이터 다양화→ 악성 + 정상 payload 문장 기반으로 도메인 사전학습
- BERT feature normalization / projection layer 도입→ BERT 출력에 BatchNorm, Dense(256) 적용 → 균형 맞춘 후 concat
- Encrypted payload 탐지 우회에 대한→ Flow anomaly 기반의 backup 판단 모델 추가 고려

하이브리드 탐지 모델 설계(2)

5. 요약 및 향후 방향 (Conclusion & Future Work)

✓ 정형 데이터(Flow feature) 기반 MLP와 비정형 데이터(Payload) 기반 BERT를 하이브리드 악성 네트워크 탐지 모델을 설계

✓MLP는 실시간성과 확장성, BERT는 정밀한 의미 분석을 담당하여 payload 존재 여부에 따라 동적으로 활용 가능한 구조를 구축

취약점	보완 방향
Payload가 없을 경우 탐지 성능 제한	MLP 중심 기본 탐지 + confidence 조정
BERT 차원 수 불균형(768 vs 128)	Dense Layer로 정규화 후 feature fusion
Payload 암호화로 인한 탐지 우회	Flow 기반 이상 탐지 모델과 결합 및 Decoding 가능한 영역 Decoding
NLP 모델 입력 부족	Pre-training corpus 확정 또는 Synthetic data 생성