정보보호와 시스템보안

AI 악성탐지 프로젝트 보고서

학부	컴퓨터공학부
조 이름	박박
학번	20153182 20153183
이름	박형준 박호준

1. 과제에서 사용한 Model, Tool, Feature Selection

- Model

모델은 Random forest와 LightGBM 모델을 Ensemble해서 과제를 수행했다.

```
def load_model(**kwargs):
    if kwargs["model"] == "rf":
        return RandomForestClassifier(random_state=kwargs["random_state"], n_jobs=4)
    elif kwargs["model"] == "dt":
        return DecisionTreeClassifier(random_state=kwargs["random_state"])
    elif kwargs["model"] == "lgb":
        return LGBMClassifier(random_state=kwargs["random_state"])
    elif kwargs["model"] == "svm":
        return SVC(random_state=kwargs["random_state"])
    elif kwargs["model"] == "lr":
        return LogisticRegression(random_state=kwargs["random_state"], n_jobs=-1)
    elif kwargs["model"] == "knn":
        return KNeighborsClassifier(n_jobs=-1)
    elif kwargs["model"] == "adaboost":
        return AdaBoostClassifier(random_state=kwargs["random_state"])
    elif kwargs["model"] == "mlp":
        return MLPClassifier(random_state=kwargs["random_state"])
    else:
        print("Unsupported Algorithm")
        return None
```

```
# 학습
models = []
for model in ["rf", "lgb"]:
    clf = train(X, y, model)
    models.append(clf)
```

랜덤 포레스트는 다수의 결정 Tree들을 학습하는 Ensemble 방법이다. 랜덤 포레스트는 검출, 분류, 그리고 회귀 등 다양한 문제에 활용되고 있다고 한다.[i]

Light GBM은 Gradient Boosting 프레임워크로 Tree 기반 학습 알고리즘이다. 기존 Tree 기반 알고리즘과 달리 Light GBM은 Tree가 수직적으로 확장된다. 즉, Light GBM은 Leaf-wise 인 반면 다른 알고리즘은 Level-wise 이다. 확장하기 위해 Max delta loss를 가진 Leaf를 선택하게 되어, 동일한 Leaf를 확장할 때, Leaf-wise 알고리즘은 Level-wise 알고리즘보다 더 많은 Loss, 손실을 줄일 수 있다고 한다.[ii]

조사해보니, 현재 악성코드 탐지 머신러닝 모델 중, Random forest, LightGBM 두 모델이 다른 모델에 비해 월등히 높은 성능과 정확성을 내는 모델임을 알 수 있었다. 실제로 우리 조가 선택한 모델 외에, 튜토리얼 코드에서 제공하는 다른 모델들을 모두 실행시켜본 결과 정확도가 90~93%가 나왔다. 이는 우리가 선택한 두 모델의 정확도(95.84%)보다 낮음을 알 수 있었다.

```
ensemble_result(X_, y_, models, sha_)
```

```
앙상블 정확도 0.9584
```

이렇게 Random forest와 LightGBM을 Ensemble했고, 학습데이터에서 다음과 같은 정확도를 낼 수 있었다.

- Data set, Feature selection

다음과 같은 방식으로 Data set의 Feature를 선택해서 모델 학습 및 검증을 수행했다.

Ember

표준화된 Histogram, Strings 영역 정보, Optional header 정보, 각 Section 정보, Imports 정보, Data directories 정보, .bak라는 이름을 가진 Section의 개수를 Feature로 활용했다.

String 영역 정보 : 수업자료를 참조해서 정상 파일과 악성 파일에서 String 정보에 네트워크 관련 문자열 암시

Optional Header : 수업자료를 참조해서 정상 파일과 악성 파일에서 Optional Header의 IMAGE_OPTIONAL_HEADER에서 Address Of Entry Point, Section Alignment 등 차이가 발생한다는 것을 참고해서 Feature로 선정했다.

Imports : 수업시간 자료에서 악성 파일의 경우 import table이 크다는 것을 참조해서 Feature로 선정했다.

Data directories : 수업시간 자료에서 Data directories 필드 16개 추출한 것을 참조하여 15개 필드를 참조하여 Feature로 선정했다. (정확도를 높이기 위해 REVERSED 제외했음)

.bak 이름 Section 개수: 수업 자료에서 .bak 라는 섹션이 비정상적으로 많이 등장할 경우 악성코드일 확률이 높다고 한 것을 참고해서 Feature로 선정했다.

PEstudio

Entropy 값, Control-flow-guard 유무, Section과 Library, Import 부분에서 Blacklist의 개수, Virustotal 정보를 Feature로 활용했다. PEstudio에서 누락된 파일이 있어 존재하지 않는 경우는 -1 값을 가지는 벡터로 구성했다.

Entropy : Entropy 수치가 높으면 난독화 되어 있을 가능성이 높고, 악성코드일 확률이 있다. 이에 이를 Feature로 선정했다.

Control_flow_guard : 메모리 주소 접근에 대한 안전성을 확인하는 루틴을 추가하는 보호 기법으로, 이것의 활성 여부를 Feature로 선정했다.

Blacklist : Section과 Library, Import 부분에서 Blacklist에 등재되어있는 각각의 개수를 Feature로 선정했다.

Virustotal : Malware에 대한 많은 정보를 가지고 있는 VirusTotal 서비스의 MD5값 검색 결과를 Feature t로 선정했다.

PEMiner

튜토리얼 코드와 동일하게 PE 파일의 Static Feature 188개를 모두 사용했다.

기타

모델에 학습데이터.csv를 학습시키고 검증데이터로 진행하여 95.84의 정확도를 기록했다.

제출하는 코드엔 학습데이터.csv와 검증데이터.csv를 함께 학습해서 테스트데이터를 검사했다.

2. 짧은소견

먼저, AI 기반 악성 탐지 프로젝트라는 좋은 경험을 했다는 것 자체가 보람찬 일이었다.
프로젝트를 하면서 Feature selection 과 Model 에 대한 조사를 해보는 등, 보안에 대한 최신 동향을 몸소 배울 수 있었다고 생각한다.
가장 어려운 것은 Model 과 Feature Selection 을 바꿔보는 과정이었다. 정확도가 100%에 근접해야 하는 정보보안 분야에서, 많은 실험을 거쳐도 악성코드 분류 정확도 1%도 올리기 힘든 모습을 보며 정보보안이 결코 쉽지 않은 분야라는 것을 느낄 수 있었다.
또한 프로젝트를 하면서 Feature 가 많다고 항상 좋은 것이 아니라는 것을 알 수 있었다. 악성코드의 특징들을 확인하기 위해 여러 JSON 파일을 계속 보면서 정적 분석이 쉽지 않음을 느끼며 분석의 중요성도 깨닫게 되었다.
이렇게 수업과 프로젝트를 직접 해보기 전엔 보안에 대한 지식이 없는 상태였지만, 수업을 통해 보안 지식을 쌓음과 더불어 필요성을 느낄 수 있었다.

참고문헌

국민대학교 정보보호와 시스템 수업자료

Ember data set : https://arxiv.org/abs/1804.04637

---

[i] https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8

[ii] https://nurilee.com/lightgbm-definition-parameter-tuning/