Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.4, Aug. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.4.559

XAI기반 악성코드 그룹분류 결과 해석 연구*

김 도 연,[1†] 정 아 연,[2] 이 태 진[3‡]
[1,2,3]호서대학교 (대학원생, 학생, 교수)

Analysis of Malware Group Classification with eXplainable Artificial Intelligence*

Do-yeon Kim,[1†] Ah-yeon Jeong,[2] Tae-jin Lee[3‡]
[1,2,3]Hoseo University (Graduate student, Undergraduate student, Professor)

요 약

컴퓨터의 보급 증가와 더불어 일반 사용자들에 대한 공격자들의 악성코드 배포 횟수 또한 증가하였다. 악성코드를 탐지하기 위한 연구는 현재까지도 진행되고 있으며 최근에는 AI를 이용한 악성코드 탐지 및 분석 연구가 중점적으로 이뤄지고 있다. 하지만 AI 알고리즘은 어떠한 이유로 악성코드를 탐지하고 분류하는지 설명할 수 없다는 단점이 존재한다. 이런 AI의 한계를 극복하고 실용성을 갖도록 하기 위해 XAI 기법이 등장하였다. XAI를 사용하면 AI의 최종 결과에 대해 판단 근거를 제시할 수 있다. 본 논문에서는 XGBoost와 Random Forest를 이용하여 악성코드 그룹분류를 진행하였으며, SHAP을 통해 결과를 해석하였다. 두 분류모델 모두 약 99%의 높은 분류 정확도를 보였으며, XAI를 통해 도출된 상위 API feature와 악성코드 주요 API를 비교해보았을 때 일정 수준 이상의 해석 및 이해가 가능하였다. 향후, 이를 바탕으로 직접적인 AI 신뢰성 향상 연구를 진행할 예정이다.

ABSTRACT

Along with the increase prevalence of computers, the number of malware distributions by attackers to ordinary users has also increased. Research to detect malware continues to this day, and in recent years, research on malware detection and analysis using AI is focused. However, the AI algorithm has a disadvantage that it cannot explain why it detects and classifies malware. XAI techniques have emerged to overcome these limitations of AI and make it practical. With XAI, it is possible to provide a basis for judgment on the final outcome of the AI. In this paper, we conducted malware group classification using XGBoost and Random Forest, and interpreted the results through SHAP. Both classification models showed a high classification accuracy of about 99%, and when comparing the top 20 API features derived through XAI with the main APIs of malware, it was possible to interpret and understand more than a certain level. In the future, based on this, a direct AI reliability improvement study will be conducted.

Keywords: Malware, XGBoost, Random Forest, XAI, SHAP

Received(03. 17. 2021), Modified(06. 07. 2021), Accepted(06. 07. 2021)

* 본 연구는 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2018-0-00276, 딥러닝 기반 악성코드 패턴룰셋 생성 자동화 원천기술 개발).

† 주저자, ehdus3342@gmail.com

‡ 교신저자, kinjecs0@gmail.com(Corresponding author)

I. 서 론

컴퓨터와 인터넷의 보급과 더불어 기술의 발전이 이뤄짐에 따라 이를 이용한 악성행위 또한 계속해서 증가하고 있다. 악성코드를 이용한 악성행위로는 소유자의 동의 없이 컴퓨터 시스템에 침투하여 손상시키는 증상이 대표적으로 있으며, 이를 방지 및 탐지하기 위해 악성코드에 대한 연구는 오래전부터 이어지고 있다[1]. 최근에는 머신러닝을 이용한 악성코드 탐지 및 분석 연구가 활발히 이뤄지고 있다. 머신러닝을 이용한 악성코드 분석은 기존의 악성코드의 특징값들을 학습시킨 후 분석데이터에 대한 악성 여부를 판단한다[2]. 기존 악성코드들의 특정 패턴을 찾아내어 정교하게 분석하는 기술과 컴퓨터 데이터의 방대한 양을 감안할 때 AI를 이용한 악성코드 분석은 빠른 처리 속도와 예측 정확도에 있어서 보안 전문가에 의해 정의된 악성코드 탐지 규칙보다 더 낫다는 장점을 가지고 있다. 하지만 AI의 경우, 입력 데이터를 통해 결과를 추출하는 과정이 불투명하여 최종 결과의 근거와 도출과정의 타당성을 설명하기 어렵다는 단점이 존재한다. 또한, 최근 크고 복잡한 데이터셋에 대해 예측 성능을 보이는 여러 복잡한 모델들의 해석이 점점 어려워지고 있다. 모델 해석을 위해 모델들이 어떻게 연관되어 있는지, 다른 모델과 비교하였을 때 선호되는 모델이 무엇인지 불분명한 경우가 많다. 이를 보완하기 위해 XAI(eXplainable Artificial Intelligence) 기법이 등장하였다.

XAI는 AI가 내린 결정이나 답을 사람이 이해하는 형태로 설명하고 제시할 수 있는 기술이다[3]. AI와 XAI 기술을 접목하면 AI가 도출한 최종 결과에 대해 어떤 부분이 예측에 성공/실패했는지, 학습 데이터의 어떤 부분에 중점을 맞춰 예측값을 도출했는지 등 판단 과정을 알 수 있다. 본 논문에서는 정적분석을 통해 Windows API feature를 이용하여 XGBoost(eXtreme Gradient Boosting)와 RF(Random Forest) 기반 악성코드 그룹분류 연구를 진행하였다. 5개의 악성코드 클래스로 그룹분류를 진행하였으며, 머신러닝 모델의 블랙박스 단점을 보완하기 위해 XAI 기술 중 SHAP(SHapley Additive exPlanations)을 접목하였다. SHAP을 통해 도출된 상위 20개의 API 기능과 악성코드 클래스 별 주요 API의 기능을 비교하여 악성코드 분류에 대한 판단 근거를 해석하였다. 본 연구 결과를 통해 머신러닝의 한계를 보완하고 추후 머신러닝을 이용한 연구에서 실용성을 갖도록 하는데 있어 XAI의 필요성을 증명하는데 중점을 두고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 악성코드 분석 및 XAI 관련 연구를 제시하고 3장에서는 Windows 악성코드 그룹분류 모델과 결과 해석을 위한 SHAP 기술을 제안한다. 4장에서는 제안하는 모델의 실험 결과와 더불어 판단 근거를 제시한다. 마지막으로 5장에서 본 논문의 주요 결론을 제시한다.

II. 관련 연구

악성코드 감염은 기업의 자산, 평판 등 많은 부분에 있어서 악영향을 끼치기 때문에 손상이 일어나기 전에 탐지하는 것이 중요하다. 기존의 불법 복제된 소프트웨어의 다운로드로부터 확산하여 지역적으로 이루어졌던 과거의 악성코드는 애플리케이션이나 사용자의 심리를 이용한 피싱과 같은 다양한 방법으로 발전은 물론 다양한 변종이 등장하고 있다[4,5]. 기존의 시그니처 기반 악성코드 탐지의 한계점이 발견되었으며 이를 극복하기 위해 Shankarapani 연구에서는 API를 통한 악성코드 탐지 연구를 진행하였다. 악성코드의 특정 코드 동작을 반영하는 Win-API(Windows API) 호출을 통해 악성코드를 분류하였다. PE 파일의 파싱을 통해 API 시퀀스를 추출한 후, 일반적인 악성코드 시퀀스 등과 비교하여 정제한다[6]. API를 통한 악성코드 탐지는 동적 기반 탐지와 정적 기반 탐지로 구분된다. 동적 기반 탐지 방법은 시간이 많이 소요되며 모든 악성코드를 판단하지 못하고, 정적 기반 탐지를 통한 악성코드 탐지의 경우 시그니처 수의 증가율이 높아 오탐지율이 높다는 단점이 있다[7].

대부분의 연구에서는 정적 또는 동적 API 호출에 대한 각각의 연구를 진행했다면, Han, Weijie 연구에서는 두 API 간의 상관관계를 이용한 연구를 진행했다. 추출한 API 호출에서 불필요한 API 호출을 제거하여 Semantics 함수 모델을 기반으로 하는 하이브리드 API 호출 시퀀스로 융합했다. 각 API 호출의 가중치에 따라 상위 n개의 API 호출을 선정하여 하이브리드 API 호출 시퀀스에서 각 API가 발생하는 빈도를 사용하여 집합을 생성하였다. Table 1은 정적 API 시퀀스, 동적 API 시퀀스, 하이브리드 API 시퀀스 feature 별 악성코드 탐지 분류를 실행한 결과를 나타낸 것이다.

Table 1. Han, Weijie's Research Results of malware detection.

	Dynamic API sequence				Static API sequence				Hybrid API sequence			
	RF	Decision Tree	KNN	XG Boost	RF	Decision Tree	KNN	XG Boost	RF	Decision Tree	KNN	XG Boost
Accuracy(%)	96.14	93.33	93.33	95.44	95.79	94.39	94.04	94.74	97.89	97.54	97.54	97.54
Precision(%)	95.35	91.82	92.78	94.46	92.60	90.38	90.02	91.28	98.13	97.05	97.89	97.45
Recall(%)	94.99	91.49	90.32	94.11	95.53	94.04	93.22	93.67	96.47	96.65	95.81	96.23
F1-score(%)	95.17	91.65	91.44	94.29	93.96	92.03	91.48	92.40	97.26	96.85	96.79	96.82

RF, XGBoost의 정확도가 높게 측정되었고 의미론적 매핑을 기반으로 동적 및 정적 API 시퀀스를 융합함으로써 분류 효과를 개선할 수 있음을 보였다[8].

더불어 딥러닝을 통한 악성코드 탐지 기술이 등장하였다[9]. 딥러닝 기반 악성코드 탐지는 분석 파일의 feature를 추출하고 여러 단계의 은닉층을 거쳐 학습 후 분석 파일의 악성 여부를 판별한다. XGBoost는 의사결정 트리를 사용하는 앙상블 학습 기법으로, 각 의사결정 트리가 이전 트리를 학습하고 다음 트리에 영향을 주면서 모델의 학습 효과 개선에 영향을 준다[10]. RF는 대규모의 데이터를 분석이 필요하며 학습에 이용하기 위한 특징을 선정하거나 상호작용을 통합하여 많은 유형의 대규모 데이터에 대해 높은 예측 정확도를 제공하여 복잡한 구조를 처리가 가능한 기술이다[11]. Azmee 연구에서는 XGBoost와 RF를 포함한 주요 알고리즘을 사용하여 악성코드를 탐지하였다. PCA를 사용하여 데이터의 분산을 보존하는 축을 찾아, 저차원의 공간으로 만들어 데이터의 과적합을 방지하였다. 그리고 데이터 표준화를 통해 각 데이터를 같은 기준으로 분석할 수 있도록 했다. 또한, Feature Selection은 예측에 주는 영향을 많이 끼치는 feature를 Ranker를 사용하여 선택하였다. Fig. 1에서 상위 feature를 사용하여 9개의 분류 알고리즘을 비교한 결과 XGBoost와 RF의 순서로 약 98.6%, 약 98.1%의 가장 높은 결과 2개를 보였다[12].

Oshiro의 연구에서는 RF의 최적화된 트리 수 존재 여부를 분석하였다. 트리의 수가 증가해도 RF의 성능이 향상되지 않고, 계산을 위한 시간과 비용이 증가할 수 있다는 결과를 보이며 최적의 트리 수가 존재함을 보였다[13].

AI가 발전하면서 DNN과 같은 복잡한 모델이 등장함과 함께 AI의 결과에 대한 해석력이 떨어지는 '블랙박스'라는 문제점이 두드러졌다. 이를 보완하기 위해 XAI가 등장하여 모델의 작동에 대한 이해를 도울 수 있게 되었다. Amir Bahador Parsa 연구에서는 교통사고 감지 기능 분석을 위해 XGBoost

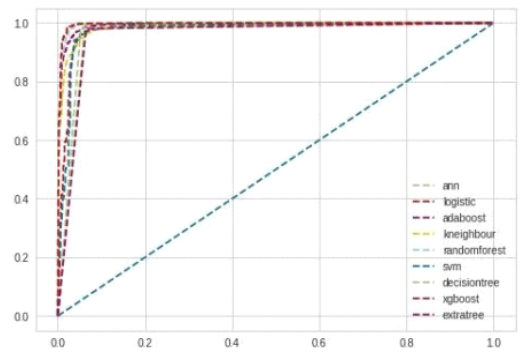


Fig. 1. Combined ROC Curve of 9 Algorithms

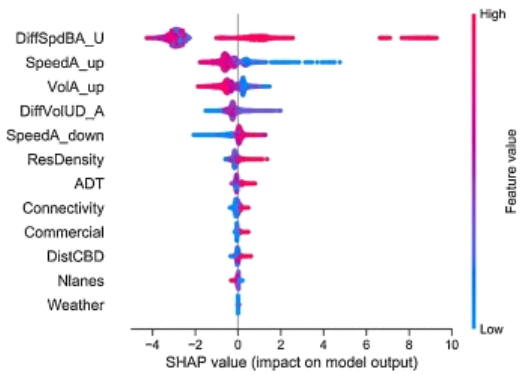


Fig. 2. The higher the item, the higher the contribution and the higher the probability of a traffic accident.

를 사용하였다. 데이터셋은 교통, 인구, 날씨 정보 등이 포함되어 있다. 불균형 데이터에 언더 샘플링 기법 대신 오버 샘플링 기법을 사용하여 모델 정확도 감소를 방지하였다. XGBoost로 학습한 모델을 SHAP을 통해 예측한 결과로 사용된 그룹의 기여도를 할당하였으며 모델에 가장 큰 영향을 끼치는 요인으로 사고 전후의 속도 차이를 뽑았다. Fig. 2에서 SHAP value가 큰 항목을 순차적으로 나열하고 교통사고가 발생할 확률도 높다는 것을 보여주면서 교통사고의 원인을 확인할 수 있다[14].

III. 제안 모델

3.1 Overview

본 논문에서는 머신러닝 기반 악성코드 그룹분류와 결과 해석을 위해 Fig. 3과 같은 시스템을 제안한다. 머신러닝 기반 Windows 악성코드 그룹분류를 위해 악성코드 별로 존재하는 정적분석 기반의 API를 추출하였다. 악성코드 별로 추출된 API 중 출현 빈도수가 높은 상위 API를 이용하여 이를 기반으로 악성코드 별 API 존재 여부 판단하였다. 해당 API가 존재할 경우 1, 없을 경우 0으로 변환하여 feature로 이용하였다. 그 후 머신러닝 기법 중 분류 예측에 높은 성능을 보이는 XGBoost와 RF를 통해 그룹분류를 진행하였다. 좀 더 세밀한 학습을 위해 세부 파라미터 조정하였으며 멀티 클래스 분류를 위해 5개의 클래스 중 하나의 클래스로 예측값을 반환해주는 OneVsRest 분류기를 이용하였다. 마지막으로 AI를 이용한 그룹분류의 최종 결과 해석을 위해 XAI 기법 중 SHAP 기법을 사용하였다. SHAP은 feature 간의 의존성을 고려하고 클래스 분류에 영향을 끼친 feature의 기여도를 시각화하여 나타낸다. 따라서 머신러닝이 어떠한 근거로 클래스 분류를 진행하였는지 시각화를 통해 확인할 수 있으며, 머신러닝의 판단에 대해 신뢰성을 증명할 수 있다.

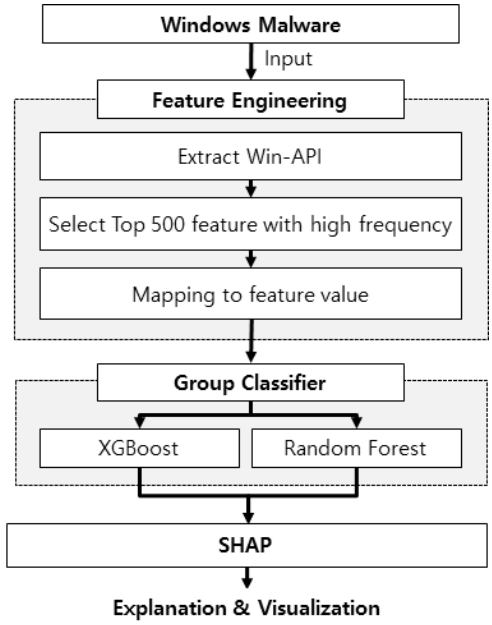


Fig. 3. Framework of Proposed Model

3.2 Feature Engineering

API는 응용프로그램에서 시스템 자원을 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 미리 정해진 메소드이다. 응용프로그램은 시스템 자원을 사용하거나 다른 응용프로그램과 상호작용을 할 때 반드시 API를 통해야 하며 프로그램 내부에서 호출되는 함수의 형태로 구현된다[15]. 또한 프로그램 실행 시 사용되는 API가 다르다. 악성코드 또한 특정 악성행위 시 사용되는 API가 다르게 나타나므로 feature로 활용할 수 있으며, 이를 이용하여 패

Table 2. Example of Win-API feature.

API / FileName	closehandle	getlasterror	getprocaddress	readfile	writefile	leavecriticalsection
008a9f053f62ddb49.vir	1	1	0	0	0	0
00af93b112cd943f83.vir	1	1	0	0	0	0
00b6f87f25a9b1d5d2.vir	1	1	1	1	1	1
00b7929b609a11c20.vir	0	0	1	0	0	1
00d884da66d795021.vir	1	1	1	1	1	1
00e24e7c21a50b4340.vir	1	1	1	1	1	1

밀리 분류 시 어떤 API에 중점적으로 판단했는지 해석이 가능하다.

본 논문에서는 Windows 악성코드 그룹분류를 위해 정적분석 기반의 Win-API를 추출하여 전처리를 진행하였다. 악성코드 별로 추출된 API 중 출현 빈도수가 높은 상위 500개를 이용하여 이를 기반으로 악성코드 별 API 존재 여부 판단하였다. 해당 API가 존재할 경우 feature 값을 1, 존재하지 않을 경우 0으로 매핑하여 총 500개의 feature를 기반으로 학습을 진행하였다.

3.3 Group Classifier

머신러닝 알고리즘은 XGBoost와 RF를 사용하였다. XGBoost와 RF는 트리 기반 앙상블 알고리즘의 대표적인 알고리즘으로 빠른 수행 속도와 분류, 회귀영역에서 높은 예측 성능을 가진다는 장점이 있다. 본 논문에서는 멀티 클래스에 대한 분류 모델을 최적화하기 위해 XGBoost의 파라미터를 조정해주었다. tree 베이스의 모델로써 gbtree 부스터를 사용하였다. 또한 모델의 과적합을 방지하기 위해 max_depth를 3으로 조정하였다. multi:softmax를 사용하여 예측 확률값을 계산하였다. RF는 여러 개의 결정 트리를 활용한 알고리즘이다. 여러 개의 결정 트리 분류기가 전체 데이터에서 bagging 방식으로 각자의 데이터를 샘플링한다. 개별적으로 학습 수행 후 최종적으로 모든 분류기가 voting을 통해 예측을 결정한다. 또한 세부 파라미터가 많아 학습 시 사람이 좀 더 세밀하게 학습을 조절할 수 있다. XGBoost와 마찬가지로 학습 시 모델 과적합을 방지하기 위해 max_depth를 3으로 설정해주었다.

Table 3. Main Hyper-parameter example of Proposed Model.

Classifier	Hyper-parameter
RandomForest Classifier	bootstrap=True max_depth=None max_features='auto' in_samples_leaf=1 min_samples_split=2 n_estimators='warn'
XGBClassifier	booster='gbtree' eval_metric='merror' importance_type='gain' max_depth=3 n_estimators=100 objective='multi:softprob'

본 논문에서는 두 모델 모두 sklearn에서 제공되는 RandomForestClassifier와 XGBClassifier 클래스를 사용하여 학습을 진행하였다. K개의 클래스가 존재할 때 각 클래스에 속하는지 아닌지를 이진 분류하여 K개의 독립된 모델을 통해 어떤 클래스에 속하는지 판별해주는 OneVsRestClassifier를 사용하여 예측 클래스를 도출하였다. Table 3은 두 제안 모델에서 사용된 주요 세부 파라미터의 예시를 나타낸 것이다.

3.4 XAI기반 해석

Shapley value는 게임 이론을 바탕으로 하나의 특성에 대한 중요도를 알기 위해 여러 특성의 조합을 통해 특성들의 유무에 따른 평균적인 변화 값을 말한다. 특성의 수가 증가할수록 연산 시간은 지수적으로 증가하고, 예측치에 대한 각 특성의 기여도를 나타낸다. Shapley value를 통해 보상의 분배를 공정하다고 평가할 수 있는 특성에는 Efficiency, Symmetry, Dummy, Additive가 있다. Efficiency는 특성의 기여도들은 반드시 x에 대한 예측치와 평균의 차이를 더한 값이어야 한다.

$$\sum_{j=1}^{p}\Phi_j = \hat{f}(x) - E_x(\hat{f}(X)) \tag{1}$$

Symmetry는 두 개의 특성값이 기여도가 모두 같은 가능한 연합에 기여한 경우, 값이 같아야 한다는 것이다. Dummy는 예측값에 영향이 없는 특성 j의 Shapley value는 0이어야 한다. Additive는 모든 연합의 특성값을 합한 값을 통해 각각의 의사결정 트리의 Shapley value를 개별적으로 계산이 가능하고 평균 낼 수 있다. 특성값의 모든 연합은 각각의 특성이 있는 것과 없는 것이 모두 평가되어야 한다. 따라서, 몬테카를로 샘플링(Monte-Carlo Sampling)을 통한 추정치를 구해야 한다.

$$\hat{\Phi}=\frac{1}{M}\sum_{m=1}^{M}(\hat{f}(\underset{+j}{x^{m}})-\hat{f}(\underset{-j}{x^{m}})) \tag{2}$$

Shapley value는 Efficiency 속성에 따라 각 예측치와 평균 예측치 간의 차이가 각 관측치의 특성값 사이에 고르게 분포한다. 또한, LIME과 달리 모

델 전체를 완전히 설명이 가능한 방법이며, 전체와 하나의 관측치 또는 전체 데이터의 일부분 또는 다른 인스턴스를 비교하여 영향을 확인할 수 있다. 한편으로는 연산량이 많고 잘못된 해석의 가능성이 존재한다[16,17].

SHAP은 feature importance와 PDP (Partial Dependence Plots)의 단점을 보완한 XAI 기법 중 하나로 예측에 대한 각 특성의 기여도를 계산하여 최종 예측값에 대한 판단 근거를 설명하는 것을 목적으로 한다[18]. 거리에 따른 가중치를 부여하는 LIME과 달리 Shapley value를 가중치로 적용한다는 차이점이 있으며 Shapley value의 계산 방법을 기반으로 하여 데이터셋의 전체적인 영역을 해석할 수 있다. SHAP의 수식은 아래와 같다[19].

$$g(z') = \Phi_0 + \sum_{j=1}^{M} \Phi_j z'_j \quad (3)$$

SHAP의 속성은 Local accuracy, Missingness, Consistency가 있다. Local accuracy가 Shapley value의 Efficiency 속성과 같아짐으로써 설명 모델이 원래의 모델과 일치함을 보인다. Missingness은 결측값이 0만큼의 영향력을 갖는다는 것을 의미하여, 연합 표기법에서 설명하고자 하는 관측치의 모든 feature value는 1로 설정하여 0의 값을 가진 feature가 결과에 영향을 미치지 않도록 한다. Consistency는 feature value의 기여도가 증가하거나 같게 모델이 변경되면 Shap value 값도 증가하거나 같게 유지된다는 특성이다[20]. SHAP의 경우 클래스 분류에 기여한 feature 별 영향도를 나타낸다는 것에서 feature importance와 비슷한 부분이 있으나, feature importance의 경우 어떠한 feature가 클래스 분류에 부정적인 영향을 끼쳐도 학습 시 고려하지 않기 때문에 결과가 왜곡될 수 있다는 단점이 있다. 반면, SHAP의 경우 feature들이 서로 의존적인 경우도 고려한다. 따라서 본 논문에서는 feature로 사용한 Win-API의 경우 악성행위를 하는 데 있어서 모두 상호 유기적으로 연관이 되어 있기 때문에 feature 간 의존성을 무시할 수 없어 SHAP 기법을 사용하여 판단 근거를 해석하였다.

SHAP은 Local interpretation과 Global interpretation 두 가지 해석 방법이 있다. Local interpretation은 하나의 데이터에 대해 feature 별 예측 기여도를 이용하여 머신러닝 결과를 해석하고, Global interpretation은 클래스에 대한 전체 데이터의 feature 별 예측 기여도를 시각화하여 사용자에게 설명 가능한 해석을 할 수 있도록 도와준다. LIME과 Shapley value를 활용한 추정 접근법인 Kernel SHAP와 트리 기반 모델을 위한 효율적인 추정 접근법인 Tree SHAP 등이 있다. Kernel SHAP은 모든 머신러닝 모델에 구애받지 않고 SHAP 값을 추정한다. feature 조합을 생성하여 random coalition을 생성한 후, 각 feature 조합들의 가중치를 계산한다. 가중치가 정해진 데이터를 이용하여 LIME에서 사용하는 로컬 대리 모델에 쓰이는 가중 선형(linear) 모델 생성 및 Shapley value를 계산한다. 모든 feature에 대한 확률값을 모두 계산하기 때문에 다른 SHAP 방법에 비해 속도가 느린 반면, 모든 feature에 대해 예측 기여도를 추정할 수 있다는 장점이 있다. 본 논문에서는 학습모델에서 사용한 OneVsRestClassifier의 특징과 모든 feature에 대해 예측 기여도를 산출하기 위해 Kernel SHAP을 이용하였다. Kernel SHAP 계산을 위하여 feature 별 예측 확률값을 사용하였다. Tree SHAP은 RF, Decision tree와 같은 트리 기반의 ML 모델 해석을 위한 알고리즘으로 Kernel SHAP보다 알고리즘이 복잡하지 않기 때문에 Kernel SHAP에 비해 분석 속도가 빠르다.

IV. 실험 결과

4.1 Dataset

악성코드 그룹분류 성능검증을 위해 1,650개의 학습데이터와 338개의 테스트 데이터로 실험을 진행하였으며, virustotal[21]에서 가져온 avclass를 기반으로 autoit, ramnit, winactivator, scar, zegost 5개 그룹의 Windows 악성코드를 이용하였

Table 4. Dataset by Family.

Family Name	Train	Test
autoit	516	171
ramnit	367	57
scar	281	17
winactivator	288	56
zegost	198	37
Total	1,650	338

다. Table 4는 악성코드 패밀리 별 데이터셋을 나타낸 것이다.

4.2 실험 결과

4.2.1 Malware family 분류

5개의 악성코드 클래스에 대해 XGBoost와 RF 기반 악성코드 그룹분류를 진행하였다. Table 5는 제안 모델의 악성코드 그룹별 분류 정확도와 최종 정확도를 나타낸 표이다. 표에서 볼 수 있듯이 두 제안 모델 모두 약 99%의 높은 정확도를 도출하였다. 악성코드 클래스별 분류 정확도도 약 98%로 높은 정확도를 도출하였으며, 정확도가 가장 낮은 scar의 경우에도 약 94%로 비교적 높은 정확도를 도출하였다. Fig. 4와 5는 XGBoost, RF 기반 악성코드 분류결과를 Confusion Matrix로 나타낸 것이다. ramnit의 경우 한 개의 데이터의 예측값이 다르다는 것을 알 수 있다.

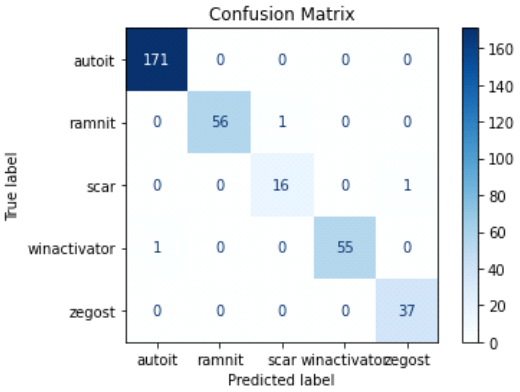


Fig. 4. Confusion Matrix of XGBoost

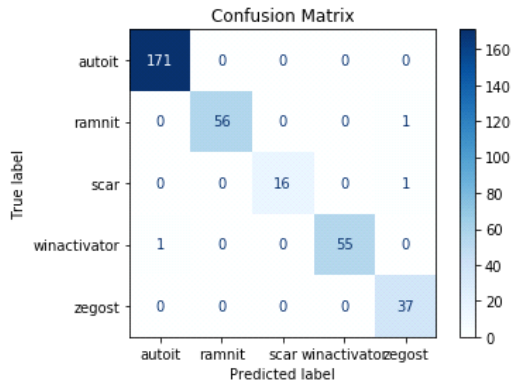


Fig. 5. Confusion Matrix of RF

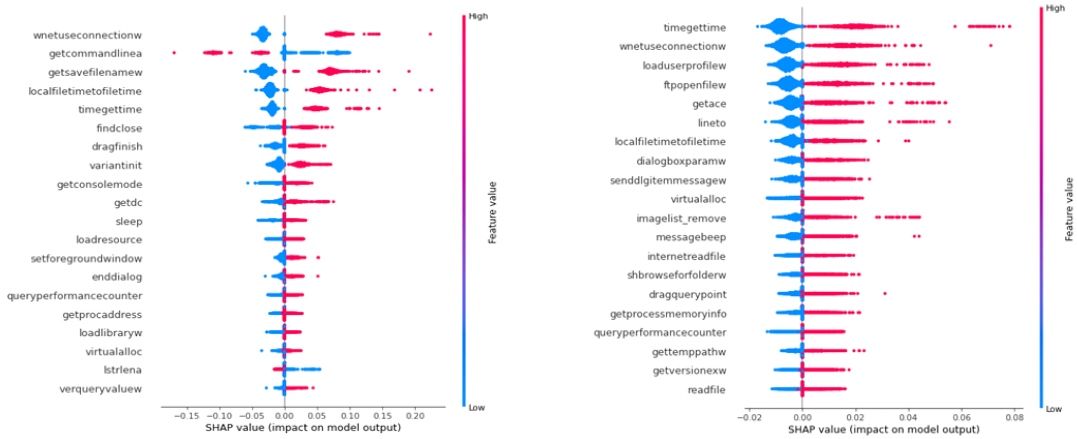
4.2.2 Malware family 분류결과에 대한 해석

AI 기반 악성코드 그룹분류 최종 결과 해석을 위해 클래스 별 주요 API 기능과 SHAP을 통해 도출된 상위 20개의 API 기능을 비교하였으며 클래스별 결과와 하나의 클래스로 분류된 샘플 악성코드 한 개의 결과에 대한 API와 주요 API를 비교한다.

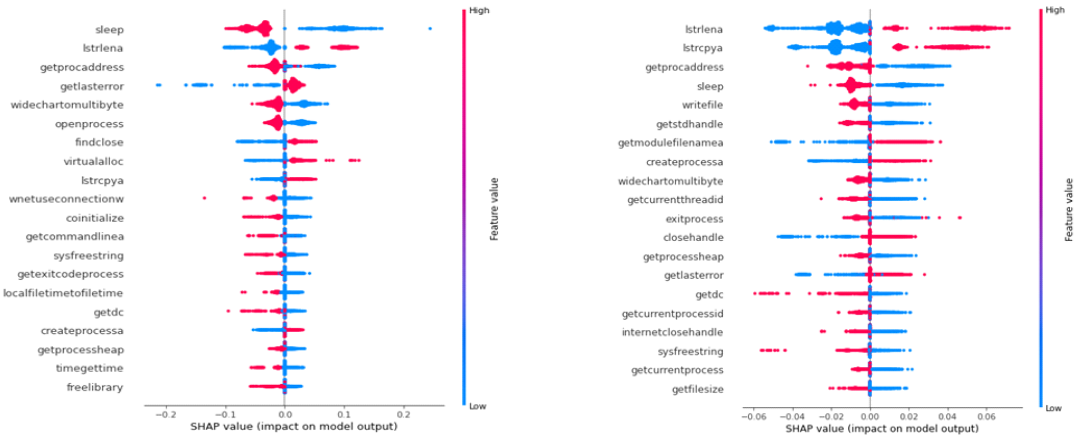
XAI 결과 해석을 위해 Global interpretation과 Local interpretation으로 나누어 설명한다. Global interpretation은 전체 데이터에서 feature 간의 관계를 바탕으로 하며 클래스 분류에 각 feature가 어떤 영향을 끼쳤는지를 나타내는 기여도를 양수 또는 음수로 시각화한다. 전체 데이터에서 판단 근거를 제시하는 Global interpretation과는 다르게 Local interpretation은 한 개의 데이터에서 feature 별 기여도를 시각화한다. Fig. 6은 Global interpretation을 위해 XGBoost-SHAP과 RF-SHAP의 결과를 클래스별로 나타내었으며, Fig. 7은 Local

Table 5. Results of Malware Classification.

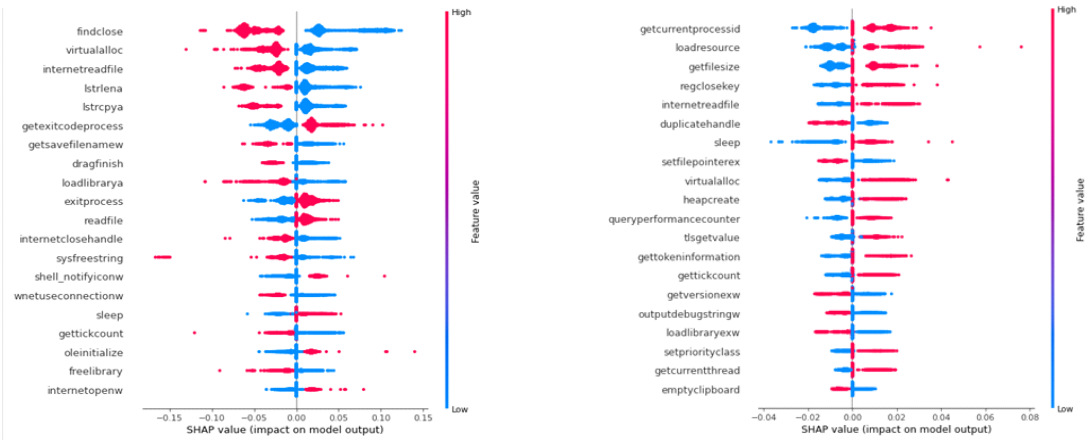
Index / Malware	XGBoost			RF		
	precision	recall	f1-score	precision	recall	f1-score
autoit	0.99	1.00	1.00	1.00	1.00	1.00
ramnit	1.00	0.98	0.99	1.00	0.98	0.99
scar	0.94	0.94	0.94	0.94	0.94	0.94
winactivator	1.00	0.98	0.99	1.00	1.00	1.00
zegost	0.97	1.00	0.99	0.97	1.00	0.99
Total Accuracy	0.9911			0.9940		



(a) autoit



(b) ramnit



(c) scar

Fig. 6. Top 20 important features of five malware group(Extracted from XGBoost-SHAP/RF-SHAP)

Table 6. Example of compared winactivator main API with Top 20 API extracted by XGBoost-SHAP /RF-SHAP.

	XGBoost-SHAP	RF-SHAP	Main API
File	sleep,Getlasterror,raisexception, localfiletimetofiletime, getsavefilenameW	getfilesize, sleep, setfilepointerex, tlsgetvalue, getcurrentthread	Getfiletype
Process	getcommandlineA, getcurrentprocessid, virtualfree,virtualalloc, getexitcodeprocess,	Getcurrentprocessid, virtualalloc, heapcreate, setpriorityclass	Getcommandlinea, Virtualfree, virtualalloc
Network	wnetuseconnectionW, findclose,	-	-
Handle	loadresource, internetreadfile, getdc,	loadresource, internetreadfile, duplicatehandle, emptyclipboard	-
etc.	gettickcount,timegettime, widechartomultibyte, messageboxA	queryperformancecounter, gettokeninformation, gettickcount, getversionexW, outputdebugstringW	Multibytetowidecha

interpretation을 위해 winactivator 클래스로 분류된 한 개의 테스트 데이터에 대한 feature 별 예측 기여도를 시각화한 것이다. 모델 학습에 중점적으로 영향을 끼친 상위 20개의 feature에 따른 클래스별 해당 feature의 영향 정도를 plot으로 나타내었다. 그래프의 y축은 feature value를 나타낸다. x축의 위치는 Shapley value에 의해 결정되며, Feature 당 Shapley value의 분포를 나타낸다. 또한, feature는 importance에 따라 정렬된다. 본 논문에서는 feature value가 0 또는 1로 구성되어 있기 때문에 feature가 존재할 경우 빨간색으로 표시되며, 반대로 존재하지 않을 경우 파란색으로 표시된다.

autoit, ramnit, scar은 악성코드 분류 중 Trojan의 하위 그룹으로 정상프로그램으로 위장하여 사용자 PC에 침입한다. ramnit은 파일 실행 시 악성파일과 다수의 DLL파일을 설치하고 인터넷 익스플로러명으로 다수의 파일에 접근하는 악성 프로세스를 실행한다. Windows 실행파일(.bat, .exe, .com 등), Microsoft Office 파일(.doc, .ppt, .pptx 등) 등을 감염시키며, 네트워크 연결 및 이동식 드라이브를 통해 확산한다. 프로세스와 관련된 행위가 많은 악성코드로 Fig. 6의 (b)에서 볼 수 있듯이 getcurrentprocess, getcurrentprocessid 등 현재 실행되고 있는 프로세스를 검색하고, 프로세스의 ID를 검색하는 프로세스와 관련된 API가 클래스 분류에 긍정적인 영향을 끼친 것을 알 수 있다. 반면, 제한 시간 간격이 경과 할 때까지 현재 스레드의 실행을 일시 중단하는 sleep은 해당 feature 값이 0일 때 Shapley value가 높다. 이는 프로세스를 중단하는 sleep feature의 경우 존재하지 않는 것이 긍정적인 영향을 끼쳤다는 것을 의미한다.

scar는 정상프로그램으로 위장하여 사용자 PC의 시스템 설정 수정 및 사용자의 키보드 및 마우스 입력을 기록, 특정 사이트 접속 등을 시도한다. 폴더 및 파일 아이콘으로 위장한 프로그램으로 다수의 복제파일 생성을 위해 readfile, loadresource, getfilesize 등 프로세스나 파일에 대한 정보를 알 수 있는 API가 feature로 존재할 시 클래스 분류에 있어서 긍정적인 영향을 끼쳤다.

winactivator은 Hacktool의 하위그룹으로, 사용자의 기밀 정보에 액세스, 팝업으로 사용자 공격 및 컴퓨터 속도 저하, 시스템 충돌을 야기시키는 악성코드이다. Windows에서 메모리를 할당하고 악성

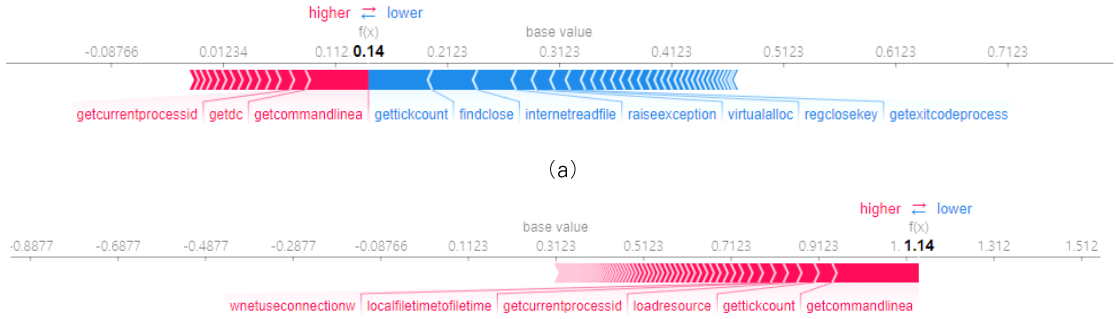


Fig. 7. Example of sample data plot about winactivator class(Local interpretation)

코드 실행 시 command-line을 이용하여 명령어를 입력하고 실행하는 특징을 가지고 있다. winactivator 클래스를 분류할 때 위에서 언급한 해당 클래스의 주요 특징들을 실행하는 API를 통해 클래스를 분류했음을 알 수 있다. cmd를 이용하여 현재 프로세스의 명령줄 문자열을 검색할 때 사용하는 getcommandlinea와 windows에서 메모리를 할당할 때 사용하는 virtualalloc가 존재할 때 Shapley value가 커짐을 알 수 있다. 따라서, 해당 API들이 존재할 시 winactivator 클래스로 분류할 가능성이 높다.

zegost는 Backdoor 종류 중 하나로 방화벽을 허용시키며 다수의 프로세스를 생성시켜 원활한 PC 이용을 방해한다. 사용자가 메일, 메신저, 게시판, 자료실 등에서 실행 파일을 다운로드해 실행하거나 웜, 트로이목마 등 다른 악성코드를 통해서 설치된다. process 관련 API가 모두 상위 feature로 선정되었으며, 대표적으로 openprocess, getcurrentprocessid, getprocaddress 등이 있다.

Fig. 7은 임의의 테스트 데이터에 대해 Local interpretation을 제시하였다. (a)와 (b) 모두 Global interpretation 결과와 유사하게 winactivator 클래스를 분류할 때 가장 많이 영향을 끼쳤던 getcommandlinea feature가 해당 데이터의 클래스 분류에도 가장 많은 영향을 끼쳤으며 이 외에도 getcurrentprocessid, loadresource 등이 클래스 분류에 영향을 끼쳤음을 알 수 있다. Table 6은 winactivator의 주요 API와 제안 모델에서 추출한 상위 feature 20개를 비교한 것이다. 프로세스 부분에서 주요 API와 XGBoost-SHAP의 API가 일치하는 것을 알 수 있다. winactivator 뿐만 아니라 다른 악성코드 클래스에서도 XGBoost-SHAP에 의해 추출된 feature가 RF-SHAP에 의해 추출된 feature보다 주요 API와 더 일치한다. 또한, RF-SHAP보다 XGBoost-SHAP에서 주요 API가 더 상위에 선정되었다.

실험 결과, 몇몇의 클래스에서 주요 API 기능과 도출된 상위 API의 기능을 비교하였을 때 일정 수준 이상이 일치함을 보였으며 어떤 API가 악성코드 분류에 있어서 영향을 끼쳤는지 해석이 가능하게 한다. 이를 통해 AI의 한계점을 극복하는데 있어 XAI의 필요성을 증명하였으며 XAI를 이용한 결과 해석은 추후 머신러닝 기반 악성코드 탐지 및 분석 연구에 있어서 함께 연구되어야 한다.

V. 결 론

오늘날의 실생활, 의료 외에도 AI는 다양한 분야에서 활용되고 있지만, AI의 결과에 대한 판단 근거를 설명하는 데 있어서 어려운 부분이 존재한다. 이러한 AI의 단점을 보완하기 위해 XAI 기법이 등장하였다. XAI는 이미지, 텍스트 뿐만 아니라 여러 AI 모델의 결과에 대한 판단 근거를 제시한다. 이는 모델이 판단한 결과에 대한 명확한 해석을 제공하고 모델에 대한 신뢰성을 향상시킬 수 있다.

본 논문에서는 XGBoost와 RF를 이용하여 5개의 악성코드 클래스를 분류하는 모델을 제안하였다. 정적분석을 통한 Win-API를 추출하여 출현 빈도수가 높은 500개의 Win-API를 feature로 사용하여 학습을 진행하였다. 두 제안 모델에서 약 99%의 높

은 분류 정확도를 도출하였으며 악성코드 클래스별 분류 정확도에서도 높은 정확도가 도출되었다. 또한 AI의 블랙박스 단점을 보완하기 위해 분류 결과를 XAI 기법 중 SHAP을 이용하여 판단 근거를 제시하였다. 결과 해석을 위해 모델 학습에 중점적으로 영향을 끼친 상위 20개의 feature에 따른 클래스별 영향도를 시각화하였다. 클래스별로 두 모델의 그래프를 비교해보았을 때 상위 20개의 feature가 유사하였다. 이를 통해 XGBoost와 RF 학습 시 중점적으로 영향을 끼친 feature가 유사한 것을 알 수 있다. 또한 5개의 악성코드 클래스의 주요행위에 이용되는 API와 AI 분류 결과에 영향을 끼친 API를 비교해보았을 때 일치하는 부분이 존재함을 확인하였다. 위 결과를 통해 AI의 한계를 극복하고 실용성을 갖도록 하기 위해 XAI의 필요성을 증명하였으며, AI 결과에 대한 일정 수준의 해석과 이해를 얻을 수 있었다.

하지만 본 논문에서 악성코드 클래스 분류 모델 학습에 사용한 데이터셋의 개수가 적어 분류 정확도의 신빙성을 확보하기 위해 추후 연구에서는 더 많은 데이터셋을 이용하여 실험을 진행할 예정이다. 또한, 클래스 분류에 영향을 준 API들이 악성코드만의 특징적인 부분이라고 단정할 수 없기 때문에, 악성코드 분류 기준의 해석 근거로 활용되는데는 무리가 있으며 주요 API 기능과 XAI를 통해 도출된 API 기능을 비교하였을 때 불일치하는 부분이 존재하였다. 사용한 Kernel SHAP의 경우 feature마다 확률값을 계산하기 때문에 속도가 느리고 추정된 값들이 실제 인스턴스에 비해서 높은 가중치를 가진다는 단점이 존재한다. 따라서 많은 데이터셋에 대해 Kernel SHAP을 적용할 경우 많은 시간이 소요된다. 이를 보완하기 위해 논문에 명시된 API들로 악성코드를 판단할 수 있는 근거가 될 수 있는지에 대한 추가 연구 및 다른 XAI 기법을 이용하여 추가 연구를 진행할 것이다. 이를 통해 더 빠른 속도와 좋은 성능을 보일 것으로 기대하며 이와 더불어 본 논문 실험 결과의 이해를 바탕으로 직접적인 AI 신뢰성 향상 방안을 추진할 예정이다.

References

〔1〕 Dragoş Gavriluţ, Mihai Cimpoeşu, Dan Anton and Liviu Ciortuz, "Malware detection using machine learning," 2009 International Multiconference on Computer Science and Information Technology, IEEE, pp. 735-741, Oct. 2009.

〔2〕 Gi-seung Baek, "Machine learning based malware analysis algorithm suitability study," KISA-WP-2017-0014, KISA, Aug. 2017.

〔3〕 Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila and Francisco Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Information Fusion vol. 58, pp. 82-115, Jun. 2020.

〔4〕 Aslan, Ömer Aslan and Refik Samet, "A comprehensive review on malware detection approaches," IEEE Vol. 8, pp. 6249-6271, Jan. 2020.

〔5〕 Si-haeng Cho, "Trends in response technology and standardization according to the evolution of malicious code," TTA Journal No.118, IT Standard & Test TTA, Jul. 2008.

〔6〕 M. Shankarapani, K. Kancherla, S. Ramammoorthy, R. Movva and S. Mukkamala, "Kernel machines for malware classification and similarity analysis," The 2010 international joint conference on neural networks(IJCNN). IEEE, pp. 1-6, Jul. 2010.

〔7〕 Hellal, Aya and Lotfi Ben Romdhane, "Minimal contrast frequent pattern mining for malware detection," Computers & Security, Vol. 62 pp. 19-32, Sep. 2016.

〔8〕 Han Weijie, Xue Jingfeng, Wang Yong, Huang Lu, Kong Zixiao and

Mao Limin, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," Computers & Security, Vol. 83, pp. 208-233, Jun. 2019.

[9] Sami, A., Yadegari, B., Rahimi, H., Peiravian, N., Hashemi, S. and Hamze, A., "Malware detection based on mining API calls," Proceedings of the 2010 ACM symposium on applied computing, pp. 1020-1025, Mar. 2010.

[10] Chen, Tianqi and Carlos Guestrin, "XGBoost: A scalable tree boosting system," Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785-794, Aug. 2016.

[11] Qi, Yanjun, "Random forest for bioinformatics," Ensemble machine learning. Springer, Boston, MA, pp. 307-323, Jan. 2012.

[12] Azmee, ABM.Adnan, Choudhury, Pranto Protim, Alam, Md.Aosaful, Dutta and Orko, "Performance analysis of machine learning classifiers for detecting PE malware," PhD Thesis, Brac University, Dec. 2019.

[13] Oshiro, T.M., Perez, P.S., and Baranauskas, J.A, "How many trees in a random forest?," MLDM 2012: Machine Learning and Data Mining in Pattern Recognition, Vol. 7376, pp. 154-168, Jul. 2012.

[14] Amir Bahador Parsa, Ali Movahedi, Homa Taghipour, Sybil Derrible, Abolfazl (Kouros) and Mohammadian, "Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis," Accident Analysis & Prevention, Vol. 136, Mar. 2020.

[15] Gupta, S., Sharma, H. and Kaur, S, "Malware characterization using windows API call sequences," International Conference on Security, Privacy, and Applied Cryptography Engineering, Springer, Cham, pp. 271-280, Dec. 2016.

[16] Maonan Wang, Kangfeng Zheng, Yanqing Yang and Xiujuan Wang, "An explainable machine learning framework for intrusion detection systems," IEEE Vol. 8, pp. 73127-73141, Apr. 2020.

[17] Alvin E. Roth, "A value for n-person games," Princeton University Press, The Pitt Building, Trumpington Street, Cambridge CB2 IRP 32 East 57th street, New York, NY 10022, USA, 2016.

[18] Lundberg, Scott, and Su-In Lee, "A unified approach to interpreting model predictions," arXiv preprint arXiv:1705.07874, Nov. 2017.

[19] Štrumbelj, Erik, and Igor Kononenko, "Explaining prediction models and individual predictions with feature contributions," Knowledge and information systems, Vol. 41, No. 3 pp. 647-665, Dec. 2014.

[20] Staniak, Mateusz, and Przemyslaw Biecek, "Explanations of model predictions with live and breakDown packages," arXiv preprint arXiv:1804.01955, Vol. 10, No. 2, pp. 395-405, Apr. 2018.

[21] virustotal, "VirusTotal." https://www.virustotal.com/

〈저자소개〉

김 도 연 (Do-yeon Kim) 학생회원
2018년 2월: 건양대학교 정보보호학과 졸업
2020년 3월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 정보보호, 기계학습, 악성코드 분석

정 아 연 (Ah-yeon Jeong) 학생회원
2018년 3월~현재: 호서대학교 컴퓨터공학부
〈관심분야〉 머신러닝, 정보보호, 포렌식

이 태 진 (Tae-jin Lee) 종신회원
2003년 1월~2017년 2월: 한국인터넷진흥원 R&D 팀장
2017년 3월~현재: 호서대학교 컴퓨터공학부 교수
〈관심분야〉 시스템 보안, 악성코드 분석, 기계학습