Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.6, Dec. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.6.1193

악성코드 분석에서의 AI 결과해석에 대한 평가방안 연구*

김 진 강,[1†] 황 찬 웅,[2] 이 태 진[3‡]
[1,2,3]호서대학교 (학생, 대학원생, 교수)

A Study on Evaluation Methods for Interpreting AI Results in Malware Analysis*

Jin-gang Kim,[1†] Chan-woong Hwang,[2] Tae-jin Lee[3‡]
[1,2,3]Hoseo University (Student, Graduate student, Professor)

요 약

정보보안에서 AI 기술은 알려지지 않은 악성코드를 탐지하기 위해 사용한다. AI 기술은 높은 정확도를 보장하지만, 오탐을 필연적으로 수반하므로 AI가 예측한 결과를 해석하기 위해 XAI 도입을 고려하고 있다. 그러나, XAI는 단순한 해석결과만 제공할 뿐 그 해석을 평가하거나 검증하는 XAI 평가 연구는 부족하다. XAI 평가는 어떤 기술이 더 정확한지 안전성 확보를 위해 필수적이다. 본 논문에서는 악성코드 분야에서 AI 예측에 크게 기여한 feature로 AI 결과를 해석하고, 이러한 AI 결과해석에 대한 평가방안을 제시한다. 약 94%의 정확도를 보이는 tree 기반의 AI 모델에 두 가지 XAI 기술을 사용하여 결과해석을 진행하고, 기술 정확도 및 희소성을 분석하여 AI 결과해석을 평가한다. 실험 결과 AI 결과해석이 적절하게 산출되었음을 확인하였다. 향후, XAI 평가로 인해 XAI 도입 및 활용은 점차 증가하고, AI 신뢰성 및 투명성이 크게 향상될 것으로 예상한다.

ABSTRACT

In information security, AI technology is used to detect unknown malware. Although AI technology guarantees high accuracy, it inevitably entails false positives, so we are considering introducing XAI to interpret the results predicted by AI. However, XAI evaluation studies that evaluate or verify the interpretation only provide simple interpretation results are lacking. XAI evaluation is essential to ensure safety which technique is more accurate. In this paper, we interpret AI results as features that have significantly contributed to AI prediction in the field of malware, and present an evaluation method for the interpretation of AI results. Interpretation of results is performed using two XAI techniques on a tree-based AI model with an accuracy of about 94%, and interpretation of AI results is evaluated by analyzing descriptive accuracy and sparsity. As a result of the experiment, it was confirmed that the AI result interpretation was properly calculated. In the future, it is expected that the adoption and utilization of XAI will gradually increase due to XAI evaluation, and the reliability and transparency of AI will be greatly improved.

Keywords: Malware, XAI, AI, Explanation, XAI Evaluation

I. 서 론

IT 시대가 도래함에 따라 4차 산업혁명에 이어 5차 산업혁명으로 넘어가려는 지금 다양한 AI 기술들이 탄생하고 연구되고 있다. 머신러닝, 딥러닝 기술들이 여러 방면으로 사용되고 있으나 인간은 이러

Received(10. 18. 2021), Modified(12. 03. 2021), Accepted(12. 03. 2021)
* 본 연구는 2021년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행되었습니다(2021-0455)
† 주저자, krch9707@naver.com
‡ 교신저자, kinjecs0@gmail.com(Corresponding author)

한 AI 기술들이 어떻게 적용되고 잘 적용되고 있는지 이해하기 힘들다. 그렇기에 XAI(Explainable AI) 기술들이 적용되고 있으며 AI가 결과를 분류, 예측하는 이유를 각 특징(feature)으로 설명하는 기술이 탄생하였다. 그러나 XAI 기술들 또한 다양한 방식이 존재한다. 다양한 XAI 기술들이 연구되고 있으며 기술마다 다른 초점의 설명을 진행하기에 어떤 기술을 사용해야 좋은 해석 결과가 나오는지에 대한 평가 방법론이 연구되고 있다. 이와 별개로 기술의 발전에 따라 악성 행위도 다양한 종류, 다양한 방식으로 침투하고 있다. 악성코드 방지 및 탐지에 관한 연구는 오래전부터 이어지고 있다[1]. AI를 이용한 악성코드 탐지 및 분석 연구 또한 함께 다양하게 이루어지고 있으며, 악성코드의 특징을 학습시켜 분석 데이터의 악성 여부를 판단한다[2-5]. 이 외에도 바이너리 분석[6-8]을 이용한 탐지 및 취약점 발견[9] 등에 AI가 적용된다. 수많은 악성코드 데이터가 생겨남에 따라 AI를 이용한 악성코드 분석은 빠른 처리 속도와 정확한 예측을 할 수 있게 한다. 그렇기에 악성코드를 AI 모델로 분석하는 경우, 해석 시 사용하는 XAI 모델들을 평가 방법론을 통해 비교하려고 한다. 우리는 악성 파일들의 API(Application Programming Interface) feature를 추출하여 XGBoost, RandomForest를 통해 분류하고 각 feature들의 정확도가 의미가 있는지를 Black-box 기법인 LIME(Local Interpretable Model-agnostic Explanations), SHAP(SHapley Additive exPlanation)을 사용하여 해석하려고 한다. 해당 해석 도구들을 평가 방법론을 통해 비교하여 악성코드 AI 모델 분류 후 해석 시 어떠한 XAI 해석 도구가 더 정확하고 희귀성 있는 결과를 도출하는지에 중점을 두고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 XAI 해석 및 평가 방법론 관련 연구를 제시하고 3장에서는 악성코드 분류 해석을 위한 평가 방법론을 제안한다. 4장에서는 제안하는 모델의 실험 결과와 평가 방법론 판단 근거를 제시한다. 마지막으로 5장에서는 본 논문의 주요 결론을 제시한다.

II. 관련 연구

악성코드 탐지에 관한 연구로는 DLL 정보를 Data로 이용한 NaiveBayes을 사용하는 방법[10]과 우리와 같이 API를 추출하여 CNN(Convolutional Neural Network), RNN(Recurrent Neural Network) 등과 같은 딥러닝 모델을 적용하는 탐지방식이 존재한다[11]. 딥러닝을 이용한 악성코드 탐지는 악성코드를 실행하여 API를 추출하거나 정적분석을 통해 opcode 같은 어셈블리 코드를 사용하는 방법, 악성코드를 이미지로 간주하여 feature를 추출하는 방법 등이 있다. 이후 해당 feature를 사용하여 딥러닝 모델을 적용하며 학습 후 악성 여부 판단 및 악성코드 그룹 분류 등을 진행한다[12]. 우리는 다양한 AI 모델 중에서 XGBoost와 RandomForest가 API 악성코드 탐지 시 높은 정확도를 보여 선택하게 되었다.

해당 딥러닝 및 머신러닝 모델을 이용한 악성코드 탐지에 대한 해석을 제공하는 XAI 기술 연구는 다음과 같다. 크게 Black-box 기법과 White-box 기법으로 나눌 수 있으며, Black-box 모델은 해당 알고리즘의 내부 논리, 작동원리가 숨겨져 있는 시스템을 의미하며, White-box 모델은 이와 반대로 모델에 대해 파라미터, 알고리즘 구조 등이 알려진 시스템을 의미한다. 대표적인 White-box 모델로는 LRP(Layer-wise Relevance Propagation), Gradients, Integrated Gradients 등이 존재하며 내부 구조가 알려져 있다 보니 설명을 결정하는데 사용된다. White-box는 Black-box보다 설명이 더 잘되는 것으로 알려져 있으나 특정 네트워크 레이아웃을 위해 설계되었기에 몇 개의 딥러닝 모델에서만 사용할 수 있다. Black-box 모델로는 LIME, SHAP, LEMNA 등이 존재한다. 내부 구조가 알려지지 않아 White-box 모델보다는 설명하는데 필요한 정보들을 놓칠 수 있으나 대부분의 AI 모델에서 사용 가능하여 범용성이 높아 자주 사용된

Table 1. Interpretability Indicator

Interpretability Indicator(weight)
▪ Intrinsically interpretable model(3) ▪ Feature Summary statistic(2) ▪ Feature Summary visualization(2) ▪ model internals(2) ▪ model-specific(2) ▪ global(2) ▪ intrinsic(2) ▪ model-agnostic(2) ▪ post-hoc(2) ▪ data points(1) ▪ local(1)

다. 우리는 범용성이 높고 비슷한 모델들을 비교하기 위해 LIME과 SHAP을 선택했다.

다른 XAI 평가 방법론으로는 해석 가능성 점수를 사용하는 방법이 존재한다[13]. 11가지의 해석 가능성 지표를 생성하여 XAI 해석 모델이 해당 해석 가능성 지표의 특징을 가지고 있다면 가중치를 부여한다. 해당 XAI 기술이 존재하는 모든 지표의 누적 점수를 통해 가중치 종합 점수를 추출하여 비교한다. 비슷한 유형의 XAI 기술들이 여러 개 존재하는 경우 종합 점수의 평균값을 구하여 해석 가능성 점수를 구한다.

해석 가능성 지표의 가중치로는 첫 번째로 가장 높은 3점을 가진 본질적 해석 가능 모델이 있으며 특별한 해석 방법 없이도 전역적이고 안정적인 해석 결과를 제공한다. 두 번째 feature 요약 통계와 세 번째 feature 요약 시각화가 있다. 결과해석을 할 수 있으나 feature 요약은 local로 이어지기에 고유 모델 해석만큼 높지 않다. 네 번째 model-internal, 다섯 번째 model-specific, 여섯 번째 global 해석, 7번째 고유해석 이 4가지 요소는 특정 모델에서만 참조 가능하며, 모델의 내부 구조에 더 집중하여 추가 탐색을 위한 feature의 통찰력을 제공하기에 2의 가중치를 준다. 여덟 번째 model-agnostic, 아홉 번째 post-hoc은 입력 feature와 outcomes 사이의 관계에만 관심이 있어 특정 모델을 제거하는 요소로 2의 가중치를 준다. 열 번째 데이터 포인트, 열한 번째 local 해석은 예측을 해석하기 위해 추가로 사용하거나 global 해석을 얻기 어려움이 있어 1의 가중치를 부여한다.

Table 1.의 해석 가능성 지표들을 통해 XAI 모델들에 존재하는 경우 가중치를 부여하며 다음과 같은 공식을 사용하여 해석 가능성 점수를 얻는다.

$$W(j) = \sum_{i=1}^{n} w(i) \tag{1}$$

$$I(k) = \frac{1}{m} \sum_{i=1}^{m} W(j) \tag{2}$$

$w(i)$는 i의 해석 가능성 지표 가중치를 의미하며 $W(j)$는 j작업에 존재하는 모든 지표의 누적 점수로 종합 점수라 부른다. m은 $W(j)$의 작업 횟수, $I(k)$는 k 유형의 작업 모델 해석 가능성 점수로 비슷한 유형을 분류하여 포함되는 XAI 연구들의 종합 점수의 평균값을 구하는 것이다.

이러한 평가 방법론은 잠재적인 해석 가능한 방법을 찾고 향후 작업에서 평가를 정량화하는데 통찰력을 제공한다. 우리는 위의 평가 방법론과 다른 좀 더 자세히 성능적 수치로 XAI 해석 도구를 비교하는 평가 방법론을 사용한다. 실제 XAI 해석 도구가 제대로 해석하는지 정확성을 통해 검증하고, feature 전체 해석이 아닌 핵심 feature에 대해서만 해석하였는가에 대한 희소성을 수치로 계산하는 것에 중점을 둔 평가 방법론이다.

III. 악성코드 분류 해석 평가 방법론

본 장에서는 악성코드를 분류하는 XAI 해석 방법들과 평가 방법론을 소개한다. 제안하는 악성코드 AI 분류 해석 평가를 위한 시스템은 아래의 Fig. 1.과 같다. 악성코드를 분류하는 방식으로는 정적분석 기반의 API를 추출하였고, 상위 K개의 빈도수에 따른 API를 통해 악성코드마다 해당 API가 존재하면 1, 존재하지 않으면 0으로 feature를 생성하였다. API는 운영체제나 프로그래밍 언어가 정해둔 메소드로 실제 프로그램을 사용하거나 다른 응용프로그램에서 사용하는 경우 상호작용을 통해 프로그램 내부에서 호출되는 함수로 구현된다[14].

그렇기에 악성코드 분류 시 의미 있는 결과를 도출할 feature로 사용하였다. 상위 500개의 빈도수가 높은 feature를 사용하여 AI 분류를 진행하였고 이후 XAI 평가를 진행한다.

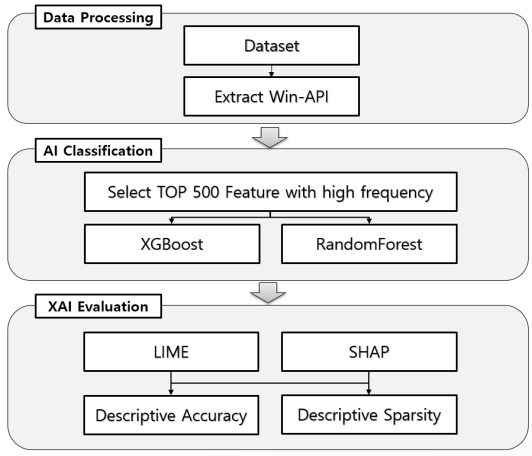

Fig. 1. Framework of Proposed Model

3.1 악성코드 탐지 모델

악성코드를 분류하기 위한 머신러닝 모델로는 XGBoost와 RandomForest를 사용하였다. 두 AI 모델은 트리 기반 앙상블의 대표적인 알고리즘으로 수행 속도가 빠르며, 분류 및 회귀 영역에서 높은 성능을 보인다는 장점이 있다.

3.2 XAI 기반 모델 해석 기법

사용할 XAI 해석 기법으로는 SHAP과 LIME을 사용한다. shapley value는 여러 특징이 서로 영향을 미치는 상황에서 서로가 어떤 의사결정이나 행동을 하는가에 따른 평균적인 변화를 통해 얻어낸 값을 의미한다. 모델 전체를 전부 설명할 수 있는 유일한 방법이나 그만큼 시간도 오래 걸리고 잘못된 설명을 할 수도 있다.

SHAP은 특징의 중요도를 기반으로 shapley value 을 사용하여 설명하는 모델로 SHAP value를 설명의 이유로 사용한다. SHAP value를 추출하기 위한 shapely value 추출 공식은 다음과 같다[15-16].

$$\Phi_i = \sum_{S \subseteq F \setminus j} \frac{|S|!(|F|-|S|-1)!}{|F|!}.(v(S \cup j) - v(S)) \quad (3)$$

F는 모든 feature set, S는 특징 j를 제외한 모든 집합, $v(x)$는 x의 하위 집합의 기여도로 특징 j가 있는 모델과 없는 모델의 결과 차이의 평균을 측정값으로 추출한다. 특징 j의 유무에 따라 기여도를 측정한다. 특징의 수가 증가할수록 측정 시간이 증가하여 SHAP은 shapley vale을 통한 x의 근사치를 사용한다[17].

Fig. 2.는 SHAP의 시각화이며 clustering feature의 결과를 지원, 이를 통해 데이터들이 모델의 결과에 어떤 영향을 미치는지 알 수 있다. x축은 SHAP value의 인스턴스, 색상은 feature value 값이 클수록 붉은색, 작을수록 푸른색으로 표현된다.

LIME은 인스턴스의 local 부분의 결정 경계를 설명하는 해석 가능 모델로 해석 가능성을 AI 사용자에게 제공하는 연구이다. 대표적인 Black-box AI model 해석기로 모델 예측의 이유를 이해하는 것이 신뢰도 상승에 가장 중요하다. 예측 결과를 이

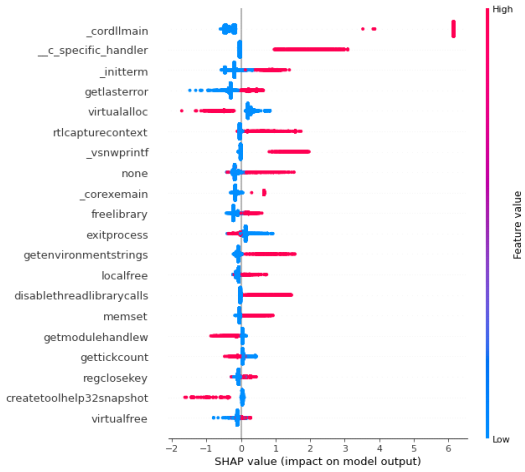

Fig. 2. SHAP summary value

해하면 모델의 신뢰성을 판단하는 데 사용할 수 있다. LIME은 local 예측을 기반으로 설명할 모델을 검증하는 것이 목표이다. 다음은 LIME의 계산 방식이다[18].

$$\epsilon(x) = \text{argmin}(L(f, g, \pi_x) + ohm(g)) \quad (4)$$

g는 모델을 사용한 설명을 나타내며, G는 설명할 수 있는 모델을 나타낸다. (g)는 g를 설명하기 위한 복잡성을 나타내고, f는 현재 설명할 모델을 나타내고 $f(x)$는 x가 입력으로 사용될 때 특정 클래스에 속할 확률을 나타낸다. $\pi_x(z)$는 인스턴스 z에서 x까지 거리를 나타내며 $L(f, g, \pi_x)$는 f를 설명할 때 g가 얼마나 신뢰할 수 없는지 나타낸다. 해석 가능성이 충분히 나타내기 위해서는 L을 최소화해야 하고 (g)가 해석 가능할 만큼 작아야 한다. 목표는 f없이 $L(f, g, \pi_x)$값이 최소화되어야 한다. 마지막으로 예측에 대한 설명인 $\epsilon(x)$를 구한다.

LIME과 SHAP의 차이점으로는 LIME은 벡터 간의 거리를 사용하지만 SHAP은 shapley value을 이용하는 것이다.

3.3 XAI 평가 방법론

3.3.1 기술 정확도(descriptive accuracy) 평가

AI 알고리즘의 모델을 설명하는 경우 정확도, precision, recall 등 다양한 설명 방법이 존재하지

만 XAI 알고리즘의 경우 각 feature 별 기여도 점수를 이용한 순위, 통계 등을 제공한다. 하지만 실제 해당 점수가 정확한지 직접적인 영향을 확인할 방법이 없기에 우리는 간접적으로 AI의 정확도를 이용하는 기술 정확도를 확인한다.

기술 정확도를 측정하기 위해서 XAI 알고리즘이 추출한 기여도 점수가 높은 top feature K개를 도출한다. 이후 가장 높은 feature를 테스트 데이터 전체에 대하여 0으로 수정, AI 모델을 돌리기 전 영향을 주지 않게 하여 AI의 정확도를 측정한다. 이후 K개만큼 반복 및 해석을 진행한다. 만약 XAI의 해석이 정확했다면 해당 feature가 AI 모델의 기여도가 높았기에 정확도가 낮아질 것이며 별 의미 없던 feature였던 경우 정확도가 변화하지 않을 것이다. 그렇다면 XAI의 해석이 정확하지 않다는 것을 의미한다. 우리는 높은 기여도 점수의 feature K개를 0으로 바꾸는 동안의 정확도를 측정하여 그래프를 통해 두 XAI 설명을 비교하여 더 깊은 굴곡의 그래프가 나오는 XAI의 기술 정확도가 더 높다고 판단하였다. 아래는 정확도 평가의 계산 방식이다.

$$DA_k(x,f_N) = f_N(x|x_1=0,...,x_k=0)_c. \quad (5)$$

파일 x가 주어지면 관련성이 높은 k개의 특성을 파일에서 0으로 변환하고 x_1부터 x_k의 샘플을 생성한다. f_N을 사용하여 새로운 예측을 통해 feature k가 없는 정확도 c를 추출한다.

3.3.2 희소성(sparsity) 평가

XAI 해석을 진행할 때 어느 정도 자동화가 이루어지더라도 마지막에는 사람이 직접 진행해야 한다. 이렇게 수동 해석을 진행하여야 하는 경우 AI가 판단한 악성코드 중 기여도 점수가 높은 feature가 한 자리에서 두 자리 정도면 가능하겠지만 세 자리를 넘어가게 되면 담당자의 시간을 많이 소모하게 된다. 그렇기에 담당자가 해석해야 하는 XAI 해석 모델이 대부분의 feature가 기여도 점수가 높다고 하지 않고, 소수의 기여도 점수가 높은 feature를 중요하게 판단하는 경우 해석이 피곤하지 않다. 만약 XAI 해석 모델이 세 자리가 넘어가는 feature 수에 전부 기여도 점수가 높게 표현한다면 담당자는 해석이 다소 피곤해진다. 각 AI 해석 기술이 정확히 판단하더라도 1.가 높은 feature가 많이 나오는 경우가 존재하며, 소수로 나타나는 경우도 존재하기에 정확도만으로 희소성을 판단할 수 없다. 정확도와 희소성이 모두 높게 나오는 XAI 해석 모델이 가장 해석 가능성이 크며 완벽한 통찰력을 제공한다. 그러나 희소성은 예외의 경우가 존재한다. feature의 수가 적을 때, 해당 feature가 실제로 전부 영향을 주는 경우 희소성이 높다고 판단되는 XAI는 적은 수의 feature 중 1개만 높은 기여도 점수가 나타나고 다른 feature에는 낮은 기여도 점수를 주는 XAI가 선택될 것이다. 이러면 희소성이 낮은 XAI가 적은 수의 feature 전부에 유사한 기여도 점수를 주는 경우가 더 좋은 해석이기에 정반대의 해석이 된다. 그러나 우리의 실험에서는 feature 500개를 사용하였기에 희소성 결과에서 일반적인 결과가 나왔다.

희소성을 측정하기 위해서 우리는 '정규화된 히스토그램'을 사용하였다. 1개의 테스트 파일에 대해 추출된 feature의 기여도 점수를 전부 더한 뒤, 각 feature 별 기여도 점수에 나누어 준다. feature 별 기여도 점수는 0에서 1 사이의 실수가 되며, 나누어진 각 Feature 별 기여도 점수를 더한 값은 총 기여도 점수 / 총기여도 점수가 되어 1이 된다. 이후 첫 번째 기여도 점수부터 누적한 값을 추출한다. 희소성 평가의 마지막 값은 모두 더한 값처럼 누적된 값이기에 1이 된다. 그래프 표현 시 0부터 1 사이로 표현되며 x축에 평행한 상수함수처럼 균등해진다.

희소성을 포괄적 수치가 아닌 샘플 데이터로 확인하려면 A 설명과 B 설명 중 기여도 점수가 가장 높은 K개의 feature를 비교한다. 임의의 수치를 지정하여 해당 수치 이상인 feature의 개수를 확인하면 어느 설명이 희소성이 더 높은지 알 수 있다. 예를 들어 임의의 수치 0.2 이상인 해석 설명의 개수를 지정하면 A 설명은 14개의 feature만 추출되고 B 설명은 2,048개의 이상의 feature가 추출되어 A 설명의 희소성이 더 높다고 판단한다[19]. 아래는 희소성을 수치로 표현하기 위한 MAZ(Mass Around Zero) 계산 방식이다.

$$MAZ(r) = \int_{-r}^{r} h(x)dx \text{ for } r \in [0,1]. \quad (6)$$

기여도 점수 r 은 [-1, 1] 사이의 값으로 스케일링이 되어 있으며, 이들을 정규화된 히스토그램 h를 진행하여 $MAZ(r)$ 값을 계산한다.

IV. 실험 결과

4.1 Dataset

악성코드 데이터는 '2019 데이터 챌린지'에서 사용한 40,000개를 사용하였다. Train은 악성 파일이 18,000개, 정상 파일은 12,000개이고, Test의 악성 파일은 5,000개, 정상 파일은 5,000개이다. Win-API를 추출한 결과 11,299개 학습 데이터와 4,447개의 테스트 데이터가 생성되었고 API가 추출되지 않은 데이터를 제외하고 추출된 데이터를 이용하여 실험을 진행하였다. 실제 학습 데이터의 악성 파일 개수는 1,407개이며 정상 파일 개수는 9,982개이다. 테스트 데이터의 악성 파일 개수는 394개이며 정상 파일 개수는 4053개이다.

4.2 악성코드 탐지 결과

악성코드를 탐지하기 위해 AI 모델 중 XGBoost를 이용하여 분류를 진행하였다. 세부 설정으로 XGBClassifier를 사용하였으며 base_score는 0.5, booster로는 gbtree를 사용하였고, 정확도는 93%의 결과가 나왔다. 이외의 악성코드를 탐지하기 위한 AI 모델로 RandomForest를 이용하여 탐지를 진행하였다. 세부설정으로는 max_depth를 12, 결정트리 개수인 n_estimators를 100, 노드 분할과 리프노드의 최소 샘플 데이터 수를 8로 설정하였다. 정확도는 94%가 나왔으며 XGBoost 보다 약간 더 높게 측정되었다. 자세한 탐지 결과는 Table. 2.와 같다.

Table 2. AI model Detection Result

	XGBoost	RF
Accuracy	0.93	0.94
Precision	0.84	0.97
Recall	0.75	0.70
F1-score	0.78	0.77

4.3 XAI 분석 결과

SHAP, LIME 두 XAI 해석 모델을 비교하기 위해 각 파일당 K개의 top feature와 기여도 점수를 추출하였다. K는 임의로 30으로 지정하였으며, K는 임의 조절 가능하다.

4.3.1 SHAP 악성코드 분류 해석

Dataset 의 각 파일에 대한 local 해석 및 global 해석을 진행하여 SHAP value를 도출하였다. SHAP은 LIME과 다르게 기여도 점수가 0부터 1 사이의 실수가 아닌 1 이상의 기여도 점수를 갖는 feature가 존재한다. 파일마다 존재하는 feature를 해석에 사용하나 global 해석을 통해 한번 해석을 진행해두면 파일별 결과 추출 시 이미 추출된 SHAP value에 의해 개별 해석 시 시간이 오래 걸리지 않는다. SHAP value는 입력 특징 중요도를 나타낼 수 있으며 다음 Fig. 3.와 같이 시각화 표현이 가능하다. Fig. 3.는 SHAP이 판단하기에 가장 영향도가 높은 순으로 나열되어있으며 가장 영향도가 높은 feature는 '_cordllmain'이었다. 두 번째로 영향도가 높게 나온 feature는 '__c_specific_handler' 이며 2번째 이후부터는 영향도에서 큰 차이를 보여준다. SHAP 에서는 이러한 글로벌 해석 외에도 Fig. 4.와 같이 local 해석도 가능하다.

Fig. 4.는 각 정상, 악성인 경우의 해석을 시각화한 그래프로 붉은색일수록 정상으로 판단한 Feature,

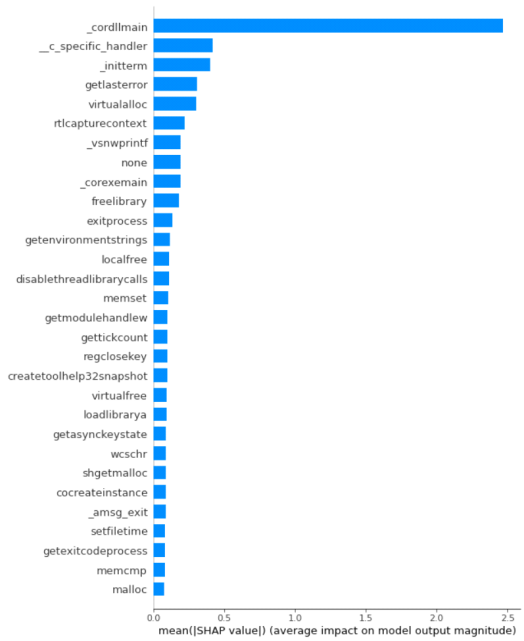

Fig. 3. SHAP summary value result

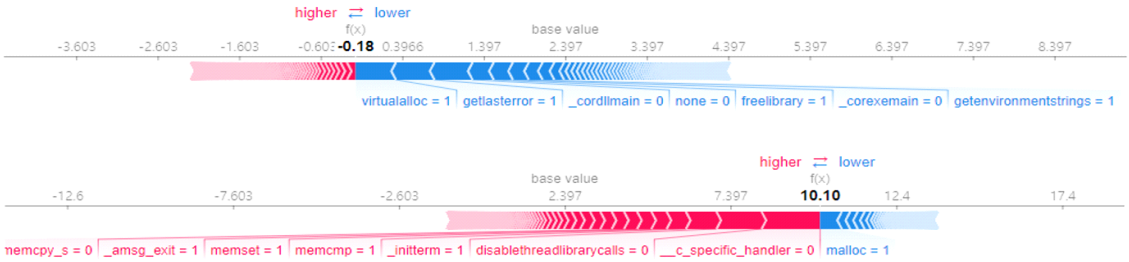

Fig. 4. SHAP local explain result

푸른색은 악성으로 영향을 주는 top feature이다.

4.3.2 LIME 악성코드 분류 해석

LIME은 파일별 local 해석을 진행하였다. 파일별 중요하다고 판단되는 feature와 해당하는 기여도 점수를 제공한다. 기여도 점수는 서로 다른 파일에 같은 feature가 존재하더라도 파일마다 미치는 영향이 다르기에 기여도 점수가 다르게 추출된다. SHAP과 다르게 매번 파일마다 local 해석을 진행하기에 총 해석에 걸린 시간이 SHAP 보다 오래 걸렸다. LIME의 기여도 점수는 Fig. 5.와 같이 시각화 표현이 가능하다. Fig. 5.을 통해 악성인지 정상인지 판단하는 확률을 백분율로 나타내며, 판단의 영향도가 큰 순서대로 feature와 기여도 점수를 나타낸다. 0과 1로 이루어진 feature의 값도 그래프를 통해 함께 제공된다. Fig. 5.는 정상 파일인 LIME 결과 예시로 주황색일수록 정상 파일로 판단하는 근거, 푸른색일수록 악성으로 판단하는 근거이다.

Fig. 5.의 top 기여도 점수는 긍정적인 feature ‘createtoolhelp32snapshot’가 0.16이며, 부정적인 feature ‘_cordllmain’는 -0.13으로 도출됐다.

Table 3. Malware Classification Analysis Result

XAI	File Name	TOP Features									
		1	2	3	4	5	6	7	8	9	10
LIME	05eed3a0653d92c1f392f72cf9c183c0.vir	virtualalloc	loadimagea	getlasterror	isbadreadptr	rtlcaptureContext	regcreatekeyexa	regdeletevaluea	lstrcmpia	getfocus	charnextw
		0.14033206	0.08662133	0.06203204	0.06094413	0.06038442	0.04939903	0.04726033	0.03813560	0.03649356	0.03618152
SHAP		rtlcaptureContext	setenvironmentvariablew	getexitcodeprocess	getlasterror	gettickcount	loadlibraryа	createtoolhelp32snapshot	virtualalloc	virtualquery	initializecriticalsection
		1.56900823	0.53513926	0.53163671	0.44932147	0.41123673	0.38389384	0.37698259	0.37419244	0.37015694	0.36404299
LIME	fead1015781a05f25433338cdab185b9.vir	createtoolhelp32snapshot	virtualalloc	loadimagea	_corexemain	regcreatekeyexa	setfiletime	getcapture	charnextw	getfocus	beginpaint
		0.15589353	0.13096112	0.05963602	0.04778243	0.04367385	0.04046551	0.03902521	0.03822076	0.03709224	0.03700068
SHAP		_corexemain	_initterm	_cordllmain	getlasterror	virtualalloc	none	localfree	getenvironmentstrings	virtualfree	regcloseкey
		0.66639888	0.46267500	0.43488463	0.35454702	0.27221432	0.22674688	0.14378030	0.13517427	0.11562214	0.09831316

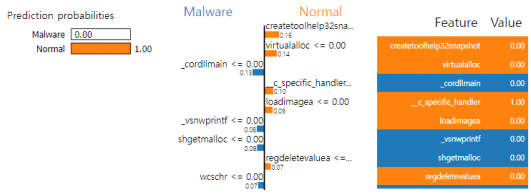

Fig. 5. LIME local explain result

SHAP과 LIME에서 추출한 TOP 30 Feature에서 중복되는 feature는 존재하였으나 완전히 동일한 경우는 찾기 힘들었고, SHAP에서 TOP-1으로 판단한 feature가 LIME에서는 TOP-3, TOP-4 사이에 나오는 경우가 다수 보였다. 두 해석 결과는 동일한 파일 비교결과 예시로 Table. 3. 와 같다.

4.4 XAI 분석결과 해석

4.4.1 기술 정확도(Descriptive Accuracy) 평가 결과

기술 정확도를 측정하기 위해 우리는 LIME과 SHAP을 비교하였다. Test Data 4,447개에 대해 추출된 SHAP value와 LIME의 기여도 점수를 사용하였다. SHAP value는 전체에 대하여 가장 영향을 주는 TOP 20개의 feature를 추출하였고, LIME은 각 Test 파일 4,447개 별 TOP-3 까지의 Feature 중 Count를 통해 상위 20개의 Feature를 추출하였다. 우리는 LIME, SHAP 등 XAI 기여도 점수에 대한 공정한 평가를 진행하기 위하여 AI 모델의 accuracy를 이용하였다. 원본 accuracy인 0.94를 기본으로 하여 추출한 상위 20개의 feature 중 TOP-1인 feature를 전체 Test 4,447개 전체에 영향을 주지 않게끔 0으로 만들어 AI model의 정확도를 측정하였다. 원본 accuracy와 새로 추출한 accuracy의 차이가 크면 변경된 feature가 해석에 큰 도움이 되었다고 판단되어 평가 정확도가 높다고 볼 수 있다. 그러나 원본 accuracy와 새로 추출한 accuracy의 차이가 작으면 기여도 점수의 총합이 크게 떨어지지 않은 것이기에 중요하지 않은 feature를 추출한 것으로 판단하

Table 4. Descriptive Accuracy Result

Descriptive Accuracy	LIME		SHAP	
	Accuracy	Feature	Accuracy	Feature
Original	0.93950978187	-	0.93950978187	-
TOP-1	0.94085900607	virtualalloc	0.57252080053	_cordllmain
TOP-2	0.57387002473	_cordllmain	0.57027209354	__c_specific_handler
TOP-3	0.57409489543	loadimagea	0.57004722284	_initterm
TOP-4	0.57184618844	__c_specific_handler	0.56757364515	getlasterror
TOP-5	0.57049696424	getlasterror	0.56982235214	virtualalloc
TOP-6	0.57072183494	regcreatekeyexa	0.56712390375	rtlcapturecontext
TOP-7	0.56802338655	shgetmalloc	0.56352597256	_vsnwprintf
TOP-8	0.56802338655	regdeletevaluea	0.55835394648	none
TOP-9	0.56914774004	isbadreadptr	0.53114459185	_corexemain
TOP-10	0.56914774004	getcapture	0.53091972116	freelibrary
TOP-11	0.56914774004	charnextw	0.53181920395	exitprocess
TOP-12	0.56577467955	_vsnwprintf	0.53159433325	getenvironmentstrings
TOP-13	0.56554980885	getenvironmentstrings	0.52799640206	localfree
TOP-14	0.56442545536	wcschr	0.52709691927	disablethreadlibrarycalls
TOP-15	0.56150213627	rtlcapturecontext	0.52484821227	memset
TOP-16	0.56060265347	lstrcmpia	0.52732178997	getmodulehandlew
TOP-17	0.56037778277	sprintf	0.52889588486	gettickcount
TOP-18	0.56037778277	lookupprivilegevaluew	0.52754666067	regclosekey
TOP-19	0.55947829997	postmessagea	0.52754666067	createtoolhelp32snapshot
TOP-20	0.56015291207	getwindowtexta	0.52574769507	virtualfree

여 평가 정확도가 낮다고 판단한다. 20개의 feature에 대해 전부 feature가 영향을 주지 않게 끔 반복하였고 기술 정확도 결과는 Fig. 6.과 같다. Table. 4.는 원본 accuracy와 20번을 반복하며 추출된 기술 정확도(Descriptive Accuracy)이다.

Feature가 영향이 없을 때 약간 상승하는 경우가 존재하지만, 전체적으로 accuracy 값이 낮아지는 결과를 보여준다.

Fig. 6.를 통해 LIME, SHAP 모두 유사하게 하강하는 그래프를 이루고 있음을 확인 가능하며, 정확도가 떨어지는 결과를 알 수 있다. Table 4.를 통해 자세한 원인을 분석해 보면 SHAP에서 가장 영향을 많이 주는 feature는 '_cordllmain'로 가장 큰 폭으로 정확도의 하강을 보여준다. LIME에서도 'cordllmain' feature가 2번째로 나오지만 가장 accuracy를 많이 떨어트리는 결과를 보여준다. LIME에서 1번째로 결정된 feature는 'virtualalloc'으로 정확도 하락에 큰 영향을 주지 못하였다. 'virtualalloc'은 SHAP 에서도 5번째 순서로 도출된 feature로 오히려 LIME, SHAP 둘 다 정확도를 조금 상승시키는 결과를 보여준다. 이후로 가장 많은 accuracy를 떨어트린 feature는 SHAP의 9번째로 해석된 '_corexemain'이 있으며 LIME에서는 TOP 20에 포함되지 않아 그래프에서 차이를 보여주고 있다. Accuracy를 떨어트리는 feature가 상위권에 나올수록 더 정확한 해석을 진행한 것으로 평가 가능하며 더 정확한 수치를 확인하기 위하여 AUC(Area Under the Curve)를 이용하였다. AUC 수치로 비교한 결과 LIME은 0.63, SHAP은 0.59로 더 점수가 낮은 SHAP의 정확도가 높다고 판단한다.

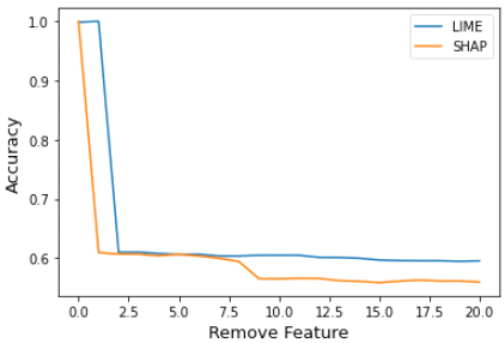

Fig. 6. Descriptive Accuracy Result

4.4.2 희소성(Sparsity) 평가 결과

희소성의 경우 모든 상황에서 평가의 객관성이 있지는 않다. 예를 들어 적은 수의 feature가 있는 Dataset 에서 전부가 사용되는 중요한 feature일 때 적은 수의 feature 전부 다 기여도 점수가 높게 나오는 경우가 옳은 해석이며, 소수의 feature만 기여도 점수가 높은 경우 다른 feature는 기여도 점수가 낮아 오히려 옳지 않은 해석으로 판단한다. 이러한 예외도 존재하지만 위 기술 정확도 결과와 같이, 한눈에 파악하기 힘든 몇백 개 이상의 feature를 사용하는 경우에서는 100개 이상의 feature가 전부 높은 기여도 점수를 가지고 있는 것보다 1~5개 사이의 중요 feature만이 높은 기여도 점수를 가지고 있는 것을 옳은 해석으로 판단한다. 이러한 희소성 평가는 feature의 수, 데이터의 상황에 따라 옳은 해석인지의 내용이 달라지는 점을 유의하며 적용하여야 한다. 우리 실험에서는 Win-API로 추출된 feature 중 상위 500개를 사용하였기에 희소성이 높은 경우 좋은 해석으로 평가하기에 적절한 Dataset 을 사용하였다.

악성코드 탐지 해석의 희소성을 평가하는 MAZ 값을 구하기 위해 우리는 기여도 점수에 스케일링을 적용했다. LIME의 기여도 점수와 SHAP value 중 top 40 feature를 이용하였으며, 서로 범위가 다르기에 비교를 하기 위한 작업으로 -1부터 1 사이의 값으로 MaxAbsScaler를 사용하여 스케일링을 진행하였다. 이후 히스토그램 정규화를 진행하였고, 기여도 점수를 전부 더한 값 h를 각각의 기여도 점수에 나누어준 값이 추출되어 누적 히스토그램을 통해 0부터 시작하여 마지막에는 1이 되는 결과를 확인하였다. LIME과 SHAP을 같은 plot에 보여

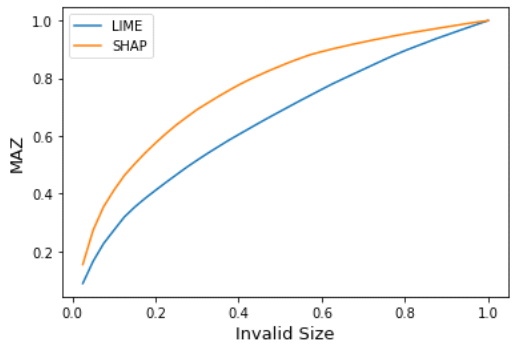

Fig. 7. Descriptive Sparsity(MAZ) Result

비교를 진행하였고 다음 Fig. 7.과 같다.

해당 Fig. 7.은 상위 40개의 feature의 샘플 데이터에 대한 희소성 결과이며 SHAP에서 더 가파른 기울기의 곡선의 그래프를 보여주고 있다. 이는 LIME보다 SHAP에서 대부분의 feature가 거의 관련성을 할당하지 않는다고 판단한다. 4,447개의 테스트 데이터 중 몇 개를 제외한 대부분이 기여도 점수가 낮다는 의미이다. 히스토그램 정규화 이후 누적 히스토그램을 통해 기여도 점수의 상승량을 곡선 그래프로 표현한다. 이는 파일별 기여도 점수의 차이가 클수록 기울기가 큰 곡선의 그래프를 이루며, 기여도 점수의 분포도가 평균과 유사할수록 평탄한 기울기의 그래프를 이룬다. 우리는 여러 번의 샘플로 테스트를 시도해 보았으나 매번 SHAP이 더 좋은 결과가 나오는 유사한 그래프를 얻을 수 있었다. 이를 통해 SHAP의 희소성이 LIME의 희소성보다 더 높아 더 좋은 해석임을 알 수 있다.

해당 실험을 통해 feature가 10이었을 경우에는 SHAP, LIME의 결과에서 별 차이가 보이지 않았지만, feature가 40이었을 경우에서 SHAP, LIME에서 차이가 벌어지는 것을 확인하였다. feature 500개 전부를 사용하면 기여도 점수가 낮은 값의 비율이 늘어나는 것이기에 그래프의 기울기가 더 심하게 기운 결과가 나타날 것이다.

V. 결 론

우리는 다양한 분야에서 AI의 결과가 사용되고 있기에 근거로 설명할 수 있는 XAI 기술들이 정확한 결과인지 검증하고 유용한지 판단할 수 있도록 평가 방법론을 제시한다. 이는 모델이 판단한 결과에 대한 명확한 해석과 신뢰성을 제공할 수 있으며, XAI의 해석의 정확도와 이 기술이 효율적인지 판단하는 희소성을 제공한다.

본 논문에서는 Win-API를 추출하여 출현 빈도수가 높은 500개의 Win-API를 feature로 사용하여 XGBoost, RandomForest 두 제안모델로 학습하였고 94%의 정확도를 얻었다. 이후 XAI 기술인 SHAP과 LIME을 사용하여 AI의 단점을 보완하기 위한 판단 근거로 기여도 점수를 얻었으며, 평가 방법론으로 비교하기 위해 상위 30개의 기여도 점수가 높은 feature를 도출하였다. LIME, SHAP 에서 추출된 30개의 feature 순서가 같은 사례는 없었지만 유사한 feature가 다수 검출되었고 대부분 순위가 다른 경우였다. 평가 방법론으로는 기술 정확도(descriptive accuracy) 평가와 희소성(sparsity) 평가를 사용하였으며 두 XAI 평가 결과 사이의 연관 관계가 있음을 확인하였다. LIME과 SHAP의 기술 정확도 결과를 비교하였을 때 순서는 다르지만 같은 feature에서 정확도가 크게 떨어지는 점이 같다는 것을 확인하였다. 해당 중요 feature의 순위가 얼마나 앞에 있는지에 따라 XAI의 해석 정확도가 높은지 판단된다. 희소성 결과에서 SHAP은 LIME보다 적은 수의 중요 feature를 추출하고 대부분의 feature에서는 기여도 점수를 주지 않는다는 결과가 나왔다. 이는 중요 feature의 기여도 점수가 상위권에서 더 많이 추출된다는 해석이 된다. 기술 정확도에서 SHAP이 LIME보다 실제 accuracy가 크게 떨어지는 중요 feature의 순위가 더 빠르게 탐지되기에 급격한 경사의 결과가 추출될 수 있었다. 기술 정확도에서 희소성의 특징을 찾을 수는 없지만, 희소성이 높을수록 기술 정확도에서 accuracy가 떨어지는 순서에 영향을 줄 수 있음을 확인 가능하였다.

본 논문에서는 기술 정확도 평가와 희소성 평가를 통해 XAI 결과가 적절하게 산출되었음을 확인하였다. 현재는 XAI 평가기술이 초기 단계이지만, 추후 정립되면 XAI 결과 활용성이 높아지며, 이는 AI 신뢰성 및 투명성이 크게 향상될 것으로 예상한다.

References

[1] D. Gavriluţ, M. Cimpoeşu, D. Anton and L. 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. 2017.

[3] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial examples for malware detection,” European Symposium on Research in Computer

Security, LNCS 10493, pp. 62 - 79, Aug. 2017.
[4] W. Huang, J.W. Stokes "A multi-task neural network for dynamic malware classification," In Proc. of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS 9721, pp. 399 - 418, June. 2016.
[5] N. McLaughlin, J.M. del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, and G.J. Ahn, "Deep android malware detection," Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 301 - 308, March. 2017.
[6] Z.L. Chua, S. Shen, P. Saxena, and Z. Liang, "Neural nets can learn function type signatures from binaries," Proceedings of the 26th USENIX Security Symposium, pp. 99 - 116, Aug. 2017.
[7] E.C.R. Shin, D. Song, and R. Moazzezi, "Recognizing functions in binaries with neural networks," Proceedings of the 24th USENIX Security Symposium, pp. 611 - 626, Aug. 2015.
[8] X. Xu, C. Liu, Q. Feng, H. Yin, L. Song, and D. Song, "Neural network-based graph embedding for cross-platform binary code similarity detection," ACM Conference on Computer and Communications Security (CCS 17), pp. 363 - 376, Aug. 2017.
[9] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong, "Vuldeepecker: a deep learning-based system for vulnerability detection," Network and Distributed System Security Symposium (NDSS), Jan. 2018.
[10] M.G. Schultz, E. Eskin, F. Zadok, and E.J. Stolfo, "Data mining methods for detection of new malicious executables," Proceedings 2001 IEEE Symposium on Security and Privacy, pp. 38-49, May 2000.
[11] C.D. Manning, P. Raghavan, and H. Schutze, "An introduction to information retrieval," Cambridge University Press, April. 2009.
[12] Sun-oh Choi, Young-soo Kim, jong-hyun Kim, and Ik-kyun Kim, "Research trends in malware detection using deep learning," Journal of The KIISC, 27(3), pp. 20-26, June. 2017.
[13] Y. Lin, X. Chang, "Towards interpreting ml-based automated malware detection models:a survey," arXiv Computer Science Cryptography and Security arXiv:2101.06232, Jan. 2021.
[14] S. Gupta, H. Sharma, and S. Kaur, "Malware characterization using windows api call sequences," International Conference on Security, Journal of Cyber Security and Mobility vol.7, pp. 363-378, Oct. 2018.
[15] S. Lundberg, Su-In Lee, "A unified approach to interpreting model predictions," Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4765 - 4774, May. 2017.
[16] L. S. Shapley, "A value for n-person games," Published by Princeton University Press, 1953.
[17] Hong-bi Kim, Yong-soo Lee, Eun-gyu Lee and Tae-jin Lee, "Cost-effective valuable data detection based on the reliability of artificial intelligence," in IEEE Access, vol. 9, pp. 108959-108974, July. 2021.
[18] M.T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?':

explaining the predictions of any classifier," Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp.97-101, Jun. 2016.

[19] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Evaluating explanation methods for deep learning in security," 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 158-174, Sep. 2020.

〈저자소개〉

김 진 강 (Jin-gang Kim) 학생회원
2016년 3월~현재: 호서대학교 정보보호학과 학사과정
〈관심분야〉 기계학습, 악성코드 분석, 포렌식

황 찬 웅 (Chan-woong Hwang) 학생회원
2020년 2월: 호서대학교 정보보호학과 졸업
2020년 3월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 네트워크 보안, 악성코드 분석, 기계학습

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