ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 21-04-12
j.inst.Korean.electr.electron.eng.Vol.25,No.4,672~678,December 2021
https://doi.org/10.7471/ikeee.2021.25.4.672

악성 이메일 공격의 사전 탐지 및 차단을 통한 이메일 보안에 관한 연구

A Study on Email Security through Proactive Detection and Prevention of Malware Email Attacks

유 지 현*★

Ji-Hyun Yoo*★

Abstract

New malware continues to increase and become advanced by every year. Although various studies are going on executable files to diagnose malicious codes, it is difficult to detect attacks that internalize malicious code threats in emails by exploiting non-executable document files, malicious URLs, and malicious macros and JS in documents. In this paper, we introduce a method of analyzing malicious code for email security through proactive detection and blocking of malicious email attacks, and propose a method for determining whether a non-executable document file is malicious based on AI. Among various algorithms, an efficient machine learning modeling is choosed, and an ML workflow system to diagnose malicious code using Kubeflow is proposed.

요 약

시간이 지날수록 새로운 맬웨어는 계속 증가하고, 점점 고도화되고 있다. 악성 코드를 진단하기 위해 실행파일에 관한 연구는 다양하게 진행되고 있으나, 비실행 문서파일과 악성 URL, 문서 내 악성 매크로 및 JS 등을 악용하여 이메일에 악성 코드 위협을 내재화한 공격은 탐지하기 어려운 것이 현실이다. 본 논문에서는 악성 이메일 공격의 사전 탐지 및 차단을 통한 이메일 보안을 위해 악성 코드를 분석하는 방법을 소개하고, AI 기반으로 비실행 문서파일의 악성 여부를 판단하는 방법을 제시한다. 다양한 알고리즘 중에 효율적인 학습 모델링 방법을 채택하고 Kubeflow를 활용하여 악성 코드를 진단하는 ML 워크플로 시스템을 제안하고자 한다.

Key words : Malware Detection, E-mail security, Reverse-Engineering, RandomForest, XGBoost, MLOps

Ⅰ. 서론

인터넷과 소프트웨어의 눈부신 발전으로 생활에 편리함과 이익을 가져왔으나 인간에 해가 되는 소프트웨어도 함께 생산되어 인터넷을 통해 빠르게 번져가고 있다. 맬웨어는 시스템에 해로울 수 있는 모든 악성 코드 또는 프로그램을 설명하는 일반적인 용어로 정보 시스템을 훼손하려는 수많은 유형

* Dept. of Software Convergence, Jangan University
Corresponding author
E-mail : jihyun_yoo@jangan.ac.kr Tel : +82-31-299-3024
※ Acknowledgment
This work was supported by Jangan University Research Grant in 2021.
Manuscript received Dec. 14, 2021; revised Dec. 18, 2021; accepted Dec. 20, 2021.

의 악성 코드가 있다. 시간이 지날수록 새로운 악성 코드의 숫자는 증가하는 추세이다.

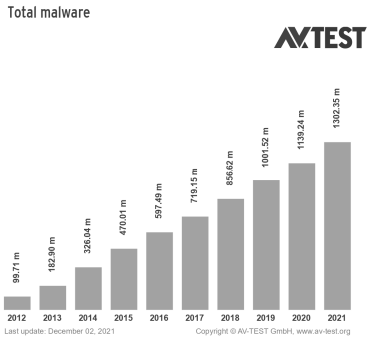

Fig. 1. Malware Statistics Last 10 Years [1].
그림 1. 10년간 악성코드 통계

공격자들의 보안 솔루션을 우회하기 위한 기법이 지속적으로 개선되고 있고, 비실행형 파일 중심으로 공격 기법이 고도화되고 있다. 따라서 이메일 보안의 중요성이 커지는 반면, 첨부된 비실행 문서 파일과 악성 URL, 문서 내 액티브 콘텐츠인 악성 매크로 및 JS 등을 악용하여 이메일에 악성 코드 위협을 내재화한 공격은 탐지하기 어려운 것이 현실이다.

본 논문에서는 악성 이메일 공격의 사전 탐지 및 차단을 통한 이메일 보안을 위해 악성 코드를 분석하는 방법을 소개하고, 효율적인 모델링 기법을 채택하여 ML 워크플로를 활용하여 악성 코드를 진단하는 시스템을 제안하고자 한다.

II. 관련 연구

1. 악성 코드 분석 방법

악성 코드를 분석하는 방법에는 다양한 작업이 포함되며 일부 작업은 간단하기도 하고, 다른 작업은 복잡하다. 관련 분석 기술의 특성에 따라 몇 단계로 그룹화할 수 있고 피라미드를 형성하며 점점 더 어려워진다.

가. 자동화 분석

의심스러운 파일의 특성을 평가하는 가장 쉬운 방법은 자동화된 도구를 사용하여 파일을 검사하는 것이다. 수없이 쏟아지는 악성 코드를 전문 분석가들이 하나씩 분석하기는 어렵고, 악성 코드 분석 도구들은 일반적으로 악성 프로그램이 사용하

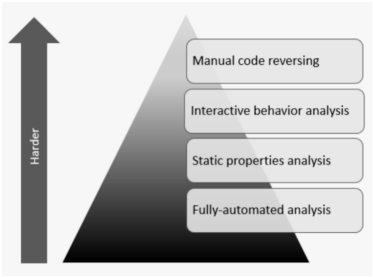

Fig. 2. 4 Ways of Malware Analysis [2].
그림 2. 악성 코드 4가지 분석 방법

는 레지스트리 키, 뮤텍스 값, 파일 활동, 네트워크 트래픽 등과 같은 세부 정보가 포함된 분석을 제공한다.

자동화된 도구는 분석가가 수동 방식으로 표본을 검사할 때 얻을 수 있는 것만큼 많은 통찰력을 제공하지 않지만 방대한 양의 악성 코드를 신속하게 처리하여 사전 예방에 기여하므로 분석가가 주의가 필요한 사례에 집중할 수 있다.

나. 정적 분석

의심스러운 파일을 정적 속성을 검사하여 진행할 수 있다. 분석가들의 세부적인 노력이 추가로 필요하고, 정적 속성에는 파일에 포함된 문자열, 헤더 세부 정보, 해시, 포함된 리소스, 패킹 여부, 생성 날짜와 같은 메타데이터 등이 포함된다.

이 정보들은 실행파일 간의 비교 데이터베이스를 구성하는데도 간단하게 활용될 수 있다.

다. 동적 분석

자동화된 도구를 사용하고 파일의 정적 분석이 이루어져도 실행 후의 모든 악의적인 행위를 정확하게 추적하기는 어렵다. 따라서 격리된 가상 시스템에서 악성 코드의 실행하여 분석하는 것이 필요하다. 이 악성 코드 분석 단계는 분석가가 표본을 수동적으로 관찰하기보다는 맬웨어와 상호 작용할 때 특히 유용하고, 다양인 방식으로 맬웨어와 상호 작용하는 것은 완전 자동화된 도구를 실행하는 것보다 더 많은 기술이 필요하다.

라. 수동 역공학 분석

동적 분석을 통해 악성 코드의 실행 후 행위에 대해 예측할 수 있지만 놓치는 요소가 있고, 표본을 구성하는 코드의 역공학 분석을 통해 좀 더 상세한 분석이 가능하다.

역공학 분석에는 디컴파일러와 디스어셈블러 및 디버거의 사용이 포함된다. 디스어셈블링된 코드에는 어떤 API가 호출되며 해당 API에 어떤 인자가 사용되는지 확인할 수 있고, 코드를 하나씩 실행해 가며 흐름을 추적할 수 있다.

2. ML 워크플로우

이러한 다양한 유형의 악의적인 프로그램으로부터의 위험을 피하기 위해서는 악성 코드를 사전에 탐지하고 예방하는 것이 중요하다. 맬웨어 분류는 널리 사용되는 작업으로, 기계 학습 모델을 통해 매우 효율적으로 수행할 수 있다.

ML 시스템을 개발 및 배포할 때 ML 워크플로는 일반적으로 여러 단계로 구성되고, ML 시스템 개발은 반복적인 프로세스이다. ML 워크플로의 다양한 단계의 출력을 평가하고 필요한 경우 모델과 매개변수에 변경 사항을 적용하여 모델이 필요한 결과를 계속 생성하도록 한다.

다음 그림은 워크플로 단계를 순서대로 보여주고 있다[3].

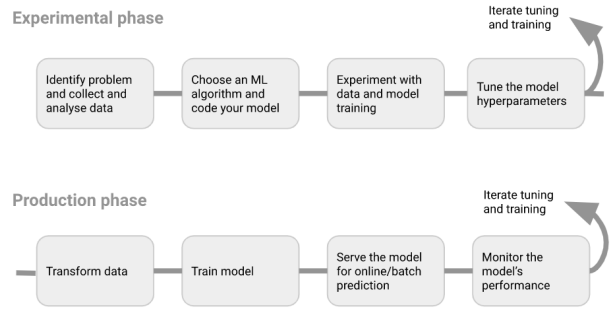

Fig. 3. ML Workflow.
그림 3. ML 워크플로

실험 단계에서는 초기 가정을 기반으로 모델을 개발하고 모델을 반복적으로 테스트 및 업데이트하여 원하는 결과를 생성한다. ML 시스템이 해결해야할 문제를 결정하고, ML 모델을 학습시키는데 필요한 데이터를 수집하고 분석한다. ML 프레임워크와 알고리즘을 선택하고 모델의 초기 버전을 코딩한다. 데이터로 실험하고 모델을 훈련시킨다. 가장 효율적인 처리와 정확한 결과를 위해 모델 하이퍼파라미터를 조정한다.

프로덕션 단계에서 데이터를 학습 시스템에 필요한 형식으로 변환한다. ML 모델을 학습시킨다. 온라인 예측 또는 배치 모드에서 실행하기 위해 모델을 실행한다. 모델의 성능을 모니터링하고 수정된 모델 또는 재학습을 위한 프로세스에 결과를 입력한다.

III. 제안 모델

1. 데이터 수집

악성 코드 대부분을 차지하는 고도화된 비실행형 파일을 분석하기 위해 정적 분석과 역공학 분석을 통해 추출한 데이터에 AI 기술을 적용하였다.

Table 1. Collected Document Data.
표 1. 수집한 문서 데이터

document	portion	label	portion
doc	26.23%	normal label	15.25%
		malicious label	4.83%
		no label	6.14%
pdf	18.2%	normal label	9.42%
		malicious label	3.76%
		no label	5.02%
xls	12.25%	normal label	6.2%
		malicious label	2.36%
		no label	3.69%
ppt	11.82%	normal label	9.59%
		malicious label	0.74%
		no label	1.49%
xlsx	9.28%	normal label	3.1%
		malicious label	2.5%
		no label	3.68%
docx	8.27%	normal label	1.88%
		malicious label	2.74%
		no label	3.63%
rtf	7.75	normal label	1.87%
		malicious label	1.74%
		no label	3.18%
hwp	5.75%	normal label	3.86%
		malicious label	0.9%
		no label	0.99%

학습 대상 데이터를 위해 바이러스토털(Virus-Total)에서 5곳 이상의 엔진에서 탐지한 악성 파일

과 정상으로 판별한 샘플, 그리고 기타 악성코드 공유 플랫폼을 이용하여 악성 파일들을 수집하였다. 바이러스토털(VirusTotal) 사이트는 전 세계의 다양한 바이러스 백신의 엔진을 이용하여 무료로 파일이나 웹사이트(URL) 등을 검사할 수 있는 서비스를 제공한다. 활용되는 바이러스 백신의 엔진은 61개의 업체에서 제공하고 있다[4].

2. 자동 학습 인프라 구축

정적 분석을 통한 특징들과 역공학 분석을 통한 특징들을 전처리하여 자동 학습 인프라를 구축하였다.

문서를 정적 분석을 통해 통합 특징 벡터를 추출한다. 문서 편집자 정보, MAC 타임, 버전, 페이지 등의 메타데이터를 학습 특징으로 사용하고, 난독화 키워드, 악성행위 가능 키워드와 같은 악성 매크로의 주요 특징을 추출하여 사용한다.

자동화된 역공학 분석으로 코드상 주요 지점에서 메모리 정보와 레지스터 정보를 추출하여 특징으로 사용한다. 객체 생성 정보와 메모리에서 추출한 스크립트 등의 역공학 분석으로 악성 문서의 동작 특징 벡터를 추출한다.

- 문서 파일을 열었을 때 주의 깊게 살펴봐야 하는 로직 내에 설정한 브레이크포인트를 통해 스레드 정보와 스택 메모라 정보를 수집하고, 수집한 정보에서 유의미한 특징을 선별하여 학습 데이터로 활용한다.
- 관찰 대상으로 선별한 사용자 정의 함수를 호출하는 경우 해당 함수의 호출자와 스레드 정보 등을 수집하고, 수집한 정보에서 유의미한 특징을 선별하여 학습 데이터로 활용한다.
- 관찰 대상으로 선별한 로직을 실행하는 경우 레지스터 정보를 수집하고, 수집한 정보에서 유의미한 특징을 선별하여 학습 데이터로 활용한다.

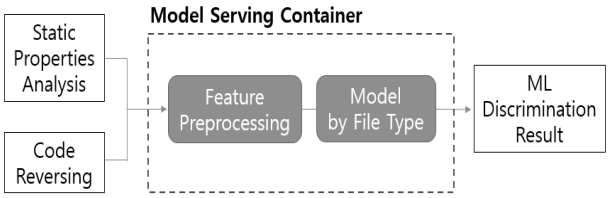

Fig. 4. Malware Diagnosis System Configuration.
그림 4. 악성 코드 진단 시스템 구성

소프트웨어가 과거에는 단순히 컴퓨터에서 실행되고, 수동 업데이트였으나, 현재는 자동 업데이트 시스템으로 발전하였고, 업데이트되더라도 지속적인 서비스가 가능해야 한다. CI/CD는 지속적 통합(Continuous Integration), 지속적 배포(Continuous Deployment)를 말하며, 이러한 과정을 통합하여 자동화한 기술인 DevOps가 매우 일반화되어있다.

MLOps는 ML 모델의 지속적인 배포 및 자동화를 위한 파이프라인으로 CT(Continuous Training) 즉, 지속적 학습이 추가된 것이다. 예를 들어서, 학습 데이터가 추가된다던가, 분류해야 하는 레이블이 늘어나는 등 다양한 학습에 관한 시나리오들이 발생하고 이러한 학습은 DevOps에서는 생각하지 못했던 구조이다.

ML(Machine Learning) 워크플로를 실행하기 위해 Kubeflow를 사용하였고, Kubeflow 파이프라인(Pipelines)으로 도커(Docker) 컨테이너를 기반으로 하는 다단계 ML 워크플로를 구축하고 배포 및 관리하였다.

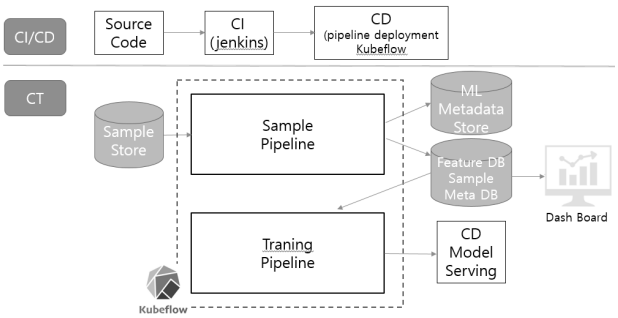

Fig. 5. MLOps Configuration.
그림 5. MLOps 구성

그림 6은 샘플 파이프라인과 학습 파이프라인이고 KubeFlow로 처리하였다.

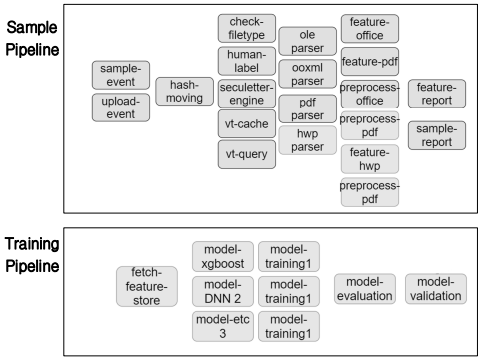

Fig. 6. Sample Pipeline and Training Pipeline.
그림 6. 샘플 파이프라인과 학습 파이프라인

랜덤 포레스트는 여러 개의 의사결정나무(Decision-Tree)를 배깅(bagging) 앙상블한 모델이다. 총 특성이 100개가 있다면 100개 모두를 사용했을 경우 과적합이 나기 때문에, 10에서 20개 정도를 선택해서 만든 의사결정나무(DecisionTree) 여러 개를 만들어 포레스트를 만드는 것이다. XG부스트 또한 여러 개의 의사결정나무를 앙상블한 알고리즘이다. 하지만 XG부스트는 랜덤 포레스트와 달리 부스팅(boosting) 앙상블로 구현된 모델이다.

Table 2. ML Modeling Result by File Type.
표 2. 파일 종류에 따른 ML 모델링 결과

Classifier	Pram.	OLE	OOXML	PDF	HWP
Random Forest Classifier	Learning Time	8.413771	1.112736	3.087538	0.158343
	Test Prediction Time	0.166979	0.066472	0.058549	0.015941
	test_accuracy	0.9565141460006986	0.9012547735951991	0.9203125	0.88
ExtraTrees Classifier	Learning Time	7.407459	1.276858	1.219326	0.117696
	Test Prediction Time	0.223291	0.074557	0.05847	0.020864
	test_accuracy	0.9549423681453021	0.8957992362247681	0.9203125	0.88
Bagging Classifier	Learning Time	22.405476	0.718645	5.902207	0.027933
	Test Prediction Time	0.150614	0.024783	0.018631	0.027933
	test_accuracy	0.9540691582256374	0.9039825422804146	0.9067708333333333	0.84
Bagging Classifier	Learning Time	40.142014	1.100456	10.4864	0.041553
	Test Prediction Time	0.141502	0.021679	0.017351	0.006916
	test_accuracy	0.9254278728606357	0.8865248226950354	0.8807291666666667	0.88
AdaBoost Classifier	Learning Time	31.74704	4.615624	1.04809	0.006612
	Test Prediction Time	0.375619	0.074587	0.004634	0.004634
	test_accuracy	0.9495284666433811	0.9018003273322422	0.871875	0.871875
Gradient Boosting Classifier	Learning Time	27.307375	2.544539	12.388306	0.15103
	Test Prediction Time	0.027468	0.00982	0.009385	0.003299
	test_accuracy	0.9556409360810338	0.900709219858156	0.9088541666666666	0.82
XGB Classifier	Learning Time	54.966618	10.110811	13.308241	0.297692
	Test Prediction Time	0.091675	0.076305	0.004697	0.0016
	test_accuracy	0.9556409360810338	0.9116202945599018	0.9125	0.90

비실행 파일들을 다양한 ML 모델링한 결과 아래 표 2에서와 같이 평균적으로 XG부스트(XGBoost)와 랜덤 포레스트(RandomForest)의 정확도가 비교적 높게 나타났다. 서비스 관점에서 XG부스트가 더 짧은 시간에 배포가 유리하다고 판단하여 XG부스트 모델을 채택하였다.

Ⅳ. 모델링 평가

오차 행렬(Confusion Matrix)는 대상 클래스가 얼마나 정확하게 예측되었고, 얼마나 잘못 예측되었는가를 판단하여 예측 모델의 성능을 평가할 수 있다.

True label이 0이면 정상 라벨, 1이면 악성 라벨, Predicted label이 0이면 정상 판단, 1이면 잘못된 판단이다.

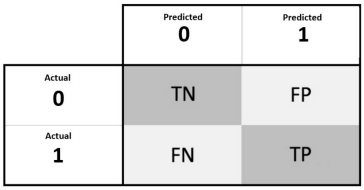

Fig. 7. Confusion Matrix.
그림 7. 오차 행렬

TN(True Negative)는 실제 정상 코드를 정상 코드로 정확히 예측, TP(True Positive)는 실제 악성 코드를 악성 코드로 정확히 예측, FP(False Positive)는 실제 정상 코드를 악성 코드로 예측(과탐), FN False Negative)은 실제 악성 코드를 정상 코드로 예측(미탐)을 의미한다.

정확도(Accuracy : ACC)

$$ACC = (TP+TN/TP+FP+TN+FN) \quad (1)$$

정밀도(Precision)

$$Precision = TP/(TP+FP) \quad (2)$$

재현율(Recall), 민감도(TPR : True Positive Rate)

$$Recall = TP/(TP+FN) \quad (3)$$

과탐(FPR : False Positive Rate)

$$FPR = FP/(FP+TN) \quad (4)$$

정밀도는 결과가 원하는 정보와 관련이 있을 확률이고, 재현율은 원하는 정보가 나올 확률이라고

볼 수 있다. 따라서, 정밀도와 재현율은 트레이드오프 관계로 정밀도가 늘어나면 재현율은 줄어들고, 재현율이 늘어나면 정밀도가 줄어든다.

다음 그림 8은 비실행 파일인 OLE, OOXML, PDF, HWP의 문서 파일에 대한 오차 행렬이다.

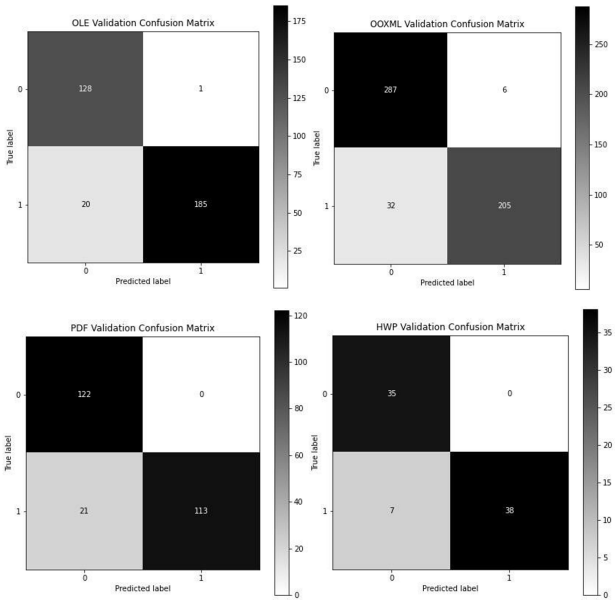

Fig. 8. Confusion Matrix of OLE, OOXML, PDF, HWP.
그림 8. 문서 파일별 오차 행렬

머신러닝에서 제안하는 모델의 성능평가는 단순히 정확도(Accuracy)만을 가지고 평가하기 어렵다. 따라서 분류기의 성능을 측정하기 위해서는 오차 행렬을 이용하여 분류기의 예측값과 실제 값 사이에 관계를 세분화하여 성능 지표를 정의한다.

Table 3. Evaluation of Confusion Matrix.
표 3. 오차 행렬 평가

Document Type (Target/Total)	Before AI	After AI Evaluation
OLE(334/600) (doc, ppt, xls) Malware : 300 Benign : 300	ACC : 0.85 FPR : 0.0033	ACC : 0.9098 Precision : 0.9946 Recall : 0.9024
OOXML(530/600) (docx, pptx, xlsx) Malware : 300 Benign : 300	ACC : 0.6133 FPR : 0.0066	ACC : 0.9283 Precision : 0.9715 Recall : 0.8649
PDF(256/300) Malware : 150 Benign : 150	ACC : 0.1533 FPR : 0.0	ACC : 0.9179 Precision : 1.0 Recall : 0.8432
HWP(80/100) Malware : 50 Benign : 50	ACC : 0.94 FPR : 0.0	ACC : 0.9125 Precision : 1.0 Recall : 0.8444

제안하는 모델의 검증을 위해 오차 행렬을 분석하여 정확도, 정밀도와 재현율의 평가를 표 3에 정리하였다. 수집한 검증 데이터셋에서 비정상 파일을 제외하여 평가대상 데이터셋의 개수에 변동 있다.

모델의 효율성을 민감도, 특이도를 이용하여 그래프로써 나타낸 ROC 커브(Curve)이다. 이 ROC 커브와 x축이 이루고 있는 면적의 넓이를 AUC (Area Under Curve)라고 하는데, AUC의 값이 1에 가까울수록 효율적인 모델이라고 할 수 있다.

x축은 특이도(False Positive Rate)로, 틀린 것을 맞았다고 잘못 예측한 수치를 나타낸다.

$$FPR = FP / (FP + TN)) \quad (1)$$

y축은 민감도(True Positive Rate)로, 맞은 것을 맞았다고 잘 예측한 수치를 나타내고 재현도(Recall)를 의미한다.

$$TPR = TP / (TP + FN) \text{ [== Recall]} \quad (2)$$

다음 그림 9는 비실행 파일인 OLE, OOXML, PDF, HWP의 ROC 커브이다.

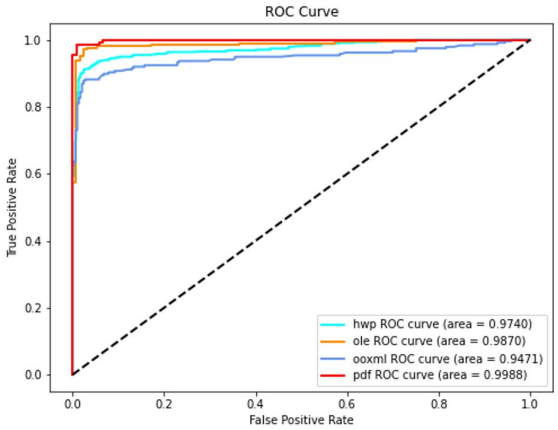

Fig. 9. ROC Curve of OLE, OOXML, PDF, HWP.
그림 9. 문서 파일들의 ROC 커브

Ⅴ. 결론

대부분의 악성 코드 분석은 실행파일 위주로 연구가 되고, 문서파일의 실행 시 동적 특징을 사용하는 것에 관한 연구는 미흡한 현실이다. 고도화되고 진화되어 가고 있는 개인정보 유출, 기반 시설 사내망 공격 등의 사이버 위협에 대한 사전 탐지와 대응이 절실하다. 기존의 연구들과 비교하였을 때

제안하는 모델의 장점은 대상이 비실행 파일이고 정적 분석과 역공학 분석을 통한 데이터를 ML 워크플로우에 적용한 것이다. 대한 국내는 해외에 비해 아직 AI 기반 Antivirus, Malware 시장이 초기 단계로서 악성 코드 분석가가 파일의 악성 여부를 판단하는 방법과 유사한 모델을 선정하여 진행한 본 연구에 의미를 두려고 한다.

References

[1] "Malware Statistics & Trends Report," https://www.av-test.org/en/statistics/malware/

[2] Lenny Zeltser, "Information Security in Businesses," https://zeltser.com/mastering-4-stages-of-malware-analysis/

[3] "Kubeflow," https://www.kubeflow.org/

[4] "Virus Total," https://www.virustotal.com/gui/home/upload

[5] J. Kinable and O. Kostakis, "Malware classication based on call graph clustering," *Journal in Computer Virology*, vol.7, no.4, pp.233-245, 2011. DOI: 10.1007/s11416-011-0151-y

[6] M. Islam, R. Tian, L. Batten, and S. Versteeg, "Classication of malware based on string and function feature selection," *Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second*, pp.9-17, 2010. DOI: 10.1109/CTC.2010.11

[7] R. Tian, L. Batten, and S. Versteeg, "Function length as a tool for malware classification," *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference*, pp.69-76, 2008. DOI: 10.1109/MALWARE.2008.4690860

[8] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," *Recent Advances in Intrusion Detection (C. Kruegel, R. Lippmann, and A. Clark, eds.), vol. 4637 of Lecture Notes in Computer Science*, pp.178-197, 2007. DOI: 10.1007/978-3-540-74320-0_10

[9] M. Zolkipli and A. Jantan, "An approach for malware behavior identification and classification," *Computer Research and Development (ICCRD), 2011 3rd International Conference*, vol.1, pp.191-194, 2011. DOI: 10.1109/ICCRD.2011.5764001

[10] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol.36, no.2, pp.646-656, 2013. DOI: 10.1016/j.jnca.2012.10.004

[11] R. Pascanu, J. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference*, pp.1916-1920, 2015. DOI: 10.1109/ICASSP.2015.7178304

[12] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol.45, no.4, pp.427-437, 2009. DOI: 10.1016/j.ipm.2009.03.002

BIOGRAPHY

Ji-Hyun Yoo (Member)

1995 : BS degree in Computer Science and Engineering, Hanyang University.
2000 : MS degree in Computer Science and Engineering, Hanyang University.
2012 : PhD degree in IT Service Management, Soongsil University.
2014 ~ : Professor in the Department of Software Convergence, Jangan University