Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.4, Aug. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.4.537

SVM 기반 유전 알고리즘을 이용한 컴파일러 분석 프레임워크 : 특징 및 모델 선택 민감성*

황 철 훈,[1†] 신 건 윤,[1] 김 동 욱,[1] 한 명 묵[2‡]
[1,2]가천대학교(대학원생, 교수)

Compiler Analysis Framework Using SVM-Based Genetic Algorithm : Feature and Model Selection Sensitivity*

Cheol-Hun Hwang,[1†] Gun-Yoon Shin,[1] Dong-Wook Kim,[1] Myung-Mook Han[2‡]
[1,2]Gachon University(Graduate student, Professor)

요 약

악성코드 기술 발전으로 변이, 난독화 등의 탐지 회피 방법이 고도화되고 있다. 이에 악성코드 탐지 기술에 있어 알려지지 않은 악성코드 탐지 기술이 중요하며, 배포된 악성코드를 통해 저자를 식별하여 알려지지 않은 악성코드를 탐지하는 악성코드 저자 식별 방법이 연구되고 있다. 본 논문에서는 바이너리 기반 저자 식별 방법에 대해 중요 정보인 컴파일러 정보를 추출하고자 하였으며, 연구 간에 특징 선택, 확률 및 비확률 모델, 최적화가 분류 효율성에 미치는 민감성(Sensitive)을 확인하고자 하였다. 실험에서 정보 이득을 통한 특징 선택 방법과 비확률 모델인 서포트 벡터 머신이 높은 효율성을 보였다. 최적화 연구 간에 제안하는 프레임워크를 통한 특징 선택 및 모델 최적화를 통해 높은 분류 정확도를 얻었으며, 최대 48%의 특징 감소 및 51배가량의 빠른 실행 속도라는 결과를 보였다. 본 연구를 통해 특징 선택 및 모델 최적화 방법이 분류 효율성에 미치는 민감성에 대해 확인할 수 있었다.

ABSTRACT

Advances in detection techniques, such as mutation and obfuscation, are being advanced with the development of malware technology. In the malware detection technology, unknown malware detection technology is important, and a method for Malware Authorship Attribution that detects an unknown malicious code by identifying the author through distributed malware is being studied. In this paper, we try to extract the compiler information affecting the binary-based author identification method and to investigate the sensitivity of feature selection, probability and non-probability models, and optimization to classification efficiency between studies. In the experiment, the feature selection method through information gain and the support vector machine, which is a non-probability model, showed high efficiency. Among the optimization studies, high classification accuracy was obtained through feature selection and model optimization through the proposed framework, and resulted in 48% feature reduction and 53 faster execution speed. Through this study, we can confirm the sensitivity of feature selection, model, and optimization methods to classification efficiency.

Keywords: Authorship Attribution, linear-chain Conditional Random Field, Genetic Algorithm, Support Vector Machine

Received(02. 03. 2020), Modified(1st: 04. 20. 2020, 2nd: 07. 10. 2020), Accepted(07. 10. 2020)
* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2018R1D1A1B07050864).
† 주저자, qewqsa@naver.com
‡ 교신저자, mmhan@gachon.ac.kr(Corresponding author)

I. 서 론

5G 네트워크, 스마트 홈, 스마트 카 등의 여러 분야의 기술들이 발전함으로써 이용자들에게 생활의 편의성을 제공한다. 그러나 이러한 발전은 악성코드의 기술 발전에 영향을 주어 악의적인 목적의 위협 수준이 증가하게 된다. 또한 보안 회피 능력을 지니도록 발전 되어 악성코드 탐지 기술 연구는 알려지지 않은 악성코드에 대하여 탐지 가능한 기술로 고도화 되고 있다[1, 2].

악성코드 저자 식별 방법은 알려진 악성코드로부터 제작한 저자에 대한 특징 패턴을 파악하고 이를 통해 알려지지 않은 악성코드로부터 저자를 식별하여 자동적으로 분류하기 위한 방법이다[3, 4, 5]. 이를 위한 저자 식별 방법은 소스코드 기반 저자 식별 방법과 바이너리 기반 저자 식별 방법으로 구분된다. 다만 소스코드 기반 저자 식별 방법은 악성코드 배포 시 소스코드를 얻기 힘든 한계점으로 인해 바이너리 기반 저자 식별 방법이 현실적이며, 이는 저자가 작성한 결과물뿐만 아니라 컴파일 된 결과물을 분석하는 것으로 프로그램의 특수한 패턴을 파악하여 탐지 및 분류하는 방법과 동일한 양상을 갖는다[6].

바이너리 기반 분석은 변수, 함수, 문자열 정보 등 다양한 특징을 활용할 수 있는 소스코드 기반 분석과 달리 특징 정의의 한계점을 갖는다. 이로 인해 다수의 클래스 분류 시 특징의 중복으로 인하여 정확성이 낮아지는 문제점을 갖는다. 이를 해결하기 위해 바이너리로부터 다양한 정보를 활용할 수 있도록 연구가 진행되고 있다. 대표적으로 바이너리로부터 컴파일러, System Call, Opcode 등의 정보를 추출 또는 변환하여 특징 정의 폭을 넓히는 것이다.

본 연구는 바이너리로부터 특징을 정의하여 컴파일러를 분류하는 선행 연구를 기반으로 한다. 선행 연구는 연속된 바이너리를 하나로 묶는 Idiom 템플릿을 정의하여 패턴을 통해 컴파일러를 분류하는 연구를 진행하였다. 정의된 특징은 linear-chain CRF(Conditional Random Field) 모델 학습을 통해 컴파일러 분류를 진행하였다. 발표된 연구의 실험은 시간이 지남에 따라 크게 두 가지의 변화를 갖는다. 첫 번째는 특징 선택 방법의 변화이며, 두 번째는 확률 모델인 linear-chain CRF 모델에서 비확률 모델인 Linear SVM(Support Vector Machine)으로 변화이다. 전자는 기존의 특징 선택 방법이 학습으로 인한 정확도 변화 차이를 통해 이루어짐으로써 학습시간이 오래 걸리기 때문에 변경하였음을 언급하였지만, 후자는 학습 모델의 다양성으로 인해 변경하였음을 확인할 수 있었다.

이에 본 연구에서는 바이너리로부터 컴파일러 분류 실험을 통해 특징 선택과 모델 선택이 정확도에 미치는 민감성(Sensitive)에 대해 실험을 진행하였다[14]. 모델 선택은 크게 확률 모델과 비확률 모델의 차이를 중점으로 하였다. 또한 최적화로 인한 분류 효율성을 확인하기 위해 유전 알고리즘을 이용한 프레임워크를 제시하여 실험을 진행하였다.

본 글은 2장에서 해당 연구에 기반이 되는 선행 연구에 대해 설명하며, 3장에서는 특징 선택 및 모델 최적화를 위한 프레임워크를 소개한다. 4장에서는 실험 및 결과를 확인하고, 마지막으로 5장에서 결론으로 마무리 한다.

II. 관련 연구

2.1 바이너리 기반의 컴파일러 분류

바이너리를 이용하여 분류, 탐지, 식별한다는 것은 가능하지만 도전적인 일이다. 이를 위해 바이너리로부터 해당 클래스가 동일하거나 명확히 다름을 보여야한다[3]. 바이너리를 이용한 여러 선행 연구에서 n-gram, idiom, 제어 흐름 그래프(control flow graph), 함수 흐름 그래프(function flow graph) 등의 여러 특징 정의 방법을 이용하여 바이너리를 통한 식별이 가능함을 보였다[7].

선행 연구에서는 idiom이라는 연속된 바이너리로부터 미리 정의된 템플릿을 이용하여 컴파일러를 분류하기 위한 실험이 진행되었다[8, 9, 10]. 해당 연구는 바이너리로부터 특징 패턴을 정의하여 분류가 가능함을 보이기 위한 목표를 가지며, 연속 확률 그래프인 linear-chain CRF (Conditional Random Field)모델을 이용하여 컴파일러 분류 실험을 하였다. GCC(GNU C Compiler), ICC(Intel C Compiler), MSVS(Microsoft Visual Studio) 컴파일러를 대상으로 실험이 진행되었으며, 바이너리를 통해 컴파일러 분류가 가능함을 보였다. 선행 연구에서 보인 특징 정의 방법은 Fig.1.과 같다. 바이너리의 각 바이트를 x_i라고 하며, 해당 바이너리의 컴파일러를 y_i라고 정의한다. 이후 일정 크기의 x_i의 연속을 idiom u_i로 정의한다. idiom u_i는

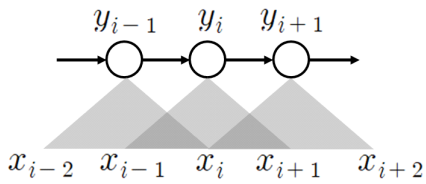

Fig. 1. linear-chain Conditional Random Filed. each x_i in the binary is paired with a label node y_i indicating its source compiler.

식 1과 같이 정의한다.

$$u_i = (push\,ebp|*|mov\,esp,ebp) \quad (1)$$

식 1은 x_i를 어셈블리 코드로 변환한 결과의 연속인 (push ebp | push edi | mov esp, ebp)를 idiom 특징으로 변환한 결과이다. 변환 시 처음과 마지막을 제외하고 나머지는 와일드카드 '*'로 변환한다. idiom u_i와 컴파일러인 y_i를 통해 모델을 학습한다. linear-chain CRF 모델은 연속적인 특징 u_i에 대하여 목표 컴파일러로 분류될 확률을 학습하고 산출한다. 해당 확률 모델은 입력된 모든 값에 대하여 확률을 구하기 때문에 입력이 많아질수록 확률 그래프가 비대해지는 문제가 발생한다. 2008년 연구에서는 회귀 모델을 통한 특징 선별 방법을 이용하였으며, 2010년 연구에서는 상호의존정보(mutual information)방법을 이용하였다. 2011년 연구에서는 기존 확률 모델에서 비확률 모델인 Linear SVM 모델로 변경하여 진행하였으며, 컴파일러 분류 결과 GCC 93.2%, ICC 96.9%, MSVS 87%의 결과를 보였다.

2.2 유전알고리즘을 이용한 모델 최적화

네트워크 침입 탐지 시스템에 이용된 분류 모델을 최적화시키기 위한 방법으로 유전 알고리즘을 이용한 연구가 진행되었다[11]. 네트워크 침입 탐지에 이용된 분류 모델은 SVM 모델을 이용하였으며, 높은 정확도를 보인 SVM 모델에 대한 최적화를 통해 더 높은 효율성을 갖게 하는 것을 연구 목표로 하였다. 모델 최적화 시 '어떤 커널을 이용할 것인가?', '커널을 합성할 것인가?', '어떤 파라미터 값을 넣어야 되는가?'에 대한 부분을 해결하기 위하여 Fig.2.와 같

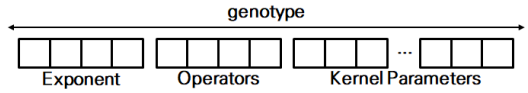

Fig. 2. genotype defined. Exponent indicates SVM kernel, Operators indicates whether there is kernel synthesis, and Kernel Parameters indicates parameter values for each kernel.

은 염색체 방식을 제안하였다. Exponent는 SVM 커널을 나타내며, Operators는 커널 합성 유무를, Kernel Parameters는 각 커널에 대한 파라미터 값을 나타낸다. 유전 알고리즘을 통해 무작위로 해당 값이 변경되며, $fitness$ 값은 모델의 분류 정확도로 이루어진다. 해당 실험에서는 20세대를 기준으로 진행하였으며, Neural 커널, Inverse Multi Qua-dric 커널, Radial 커널을 대상으로 하였다. 결과적으로 Neural 커널을 이용한 SVM 모델이 높은 정확도를 보였으며, 유전 알고리즘을 이용한 최적화 SVM 모델은 그렇지 않은 모델보다 3~5% 정도의 향상 결과를 보여주었다.

III. 유전 알고리즘을 이용한 제안 프레임워크

제안하는 프레임워크는 분류 모델에 따른 최적의 특징 및 모델 튜닝을 통해 분류 효율성을 높이는 것에 목적을 갖는다. 분류 모델은 SVM 모델을 선택하였다. SVM 모델은 데이터를 분석하고 패턴을 인식하는 감독 학습 모델이며, 모델의 유연성과 계산 효율성으로 인해 여러 도메인의 분류 문제에 적용되어 좋은 성과를 거두었다. 보안 분야에서는 안드로이드 악성 앱 분류를 위해 유전 알고리즘을 적용하여 분류 모델인 SVM 모델에 대한 최적의 특징을 찾기 위한 방법으로 적용되었다. 결과적으로 적은 양의 특징으로 높은 정확도를 보였다[12, 13]. 제안하는 프레임워크에서는 이러한 선행 연구 결과를 바탕으로 SVM 모델을 분류 모델로 선정하였으며, 유전 알고리즘을 통해 최적의 특징 선택뿐만 아니라 SVM 모델의 커널, 파라미터 등을 통해 최적의 모델 튜닝이 이루어질 수 있도록 제안하였다. 유전 알고리즘은 모델 및 특징 선택을 최적화할 수 있도록 염색체를 Fig.3.과 같이 정의하였다. Exponent는 SVM 모델에 사용될 커널을 의미한다. 선행 연구의 Expon-ent와 동일하게 커널 합성을 위해 중복 선택이 가능하다. Kernel Parameter는 각 커널에 대한 파

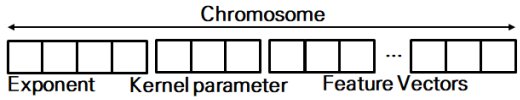

Fig. 3. Chromosome defined for feature selection and model building. Exponent indicates SVM kernel, Kernel Parameters indicates parameter values for each kernel, and Feature vector indicates the presence or absence of feature selection.

라미터의 값을 나타내며, 일정 크기의 무작위 실수 값을 갖는다. Feature Vectors는 모델 학습에 이용될 특징의 선택 유무를 나타낸다.

제안하는 프레임워크의 전체 흐름은 Fig.4.와 같다. 입력 데이터인 바이너리는 디어셈블 과정을 통해 어셈블리 코드로 변환한다. 변환 과정에서 IDA pro를 이용하였으며, 변환된 어셈블리 코드는 선행 연구와 동일한 Idiom 템플릿을 이용하여 특징을 정의한다. Idiom 템플릿에 정의된 특징들은 염색체의 Feature Vectors에 해당한다. 정의된 특징 벡터 정보는 초기 염색체(Initial Chromosome) 단계로 전달하여 0세대의 염색체를 정의되며, 유전 알고리즘의 세대 학습이 진행된다. 세대 학습 단계에서 Feature Vectors, Exponents, Kernel param-eter에 따라 특징 선택 및 모듈 튜닝이 이루어진다. Feature Vectors에 따라 데이터 전처리를 통해 훈련 및 테스트 세트로 분리되며, 훈련 세트를 통해 SVM을 학습하고 테스트 세트를 이용하여 분류 결과를 측정한다. 분류 정확도는 f-score를 통해 산출한다. $fitness$ 값은 식 2와 같이 정의하였다.

$$fitness = w_{fs}FS + (1 - w_{len}FL) \quad (2)$$

FS는 분류 정확도이며, FL은 선택된 특징의 비율을 나타낸다. w_{fs}, w_{len}는 각 분류 정확도와 선택된 특징 수의 가중치이다. 별도의 $fitness$를 정의한 이유는 모델 최적화 및 특징 선택에 대한 최적화 결과를 반영하기 위함이며, 동일한 정확도의 경우 특징의 수가 적은 경우를 더 높게 평가하였다. 선행 연구[5, 6, 8, 9, 10]에서 보인 각 컴파일별 정의된 특징 수와 분류 정확도를 식 2에 적용하여 산출된 값의 평균보다 fitness 값이 높게 나타나며, 200세대 간에 더 이상 변동이 없을 때 학습 세대 종료하도록 Break Point를 설정하였다. 학습 종료 시 최적화된 염색체 결과가 도출된다. 최적화된 염색체를 통해 SVM 모델의 최적화 커널 및 커널 파라미터가 정의되며, Feature Vectors를 통해 분류에 최적화된 특징이 도출된다. 학습 진행 시 Genetic Opera-tions 과정을 통해 다음 세대의 염색체를 정의하였으며, One-point crossover, 토너먼트 선택 방식을 이용하였다. 변이 확률은 0.05, 자손은 100으로

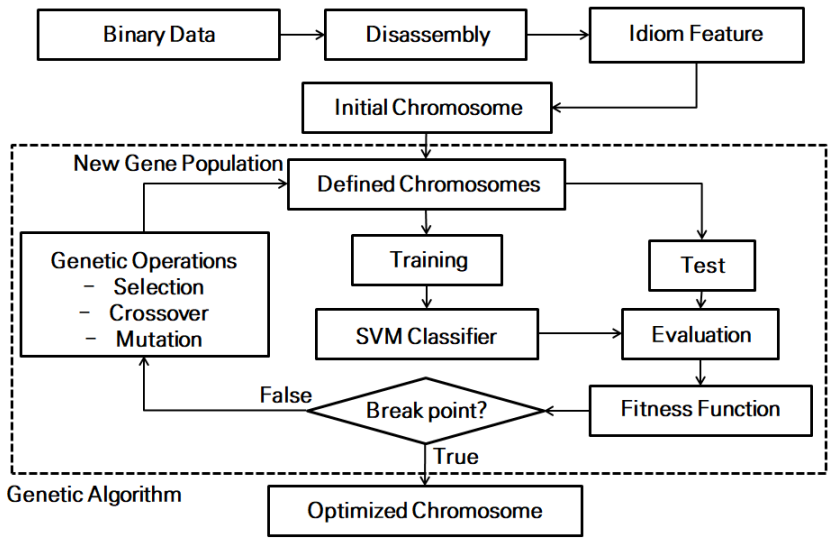

Fig. 4. proposed non-probability model. Genetic algorithms are used to build optimization models for SVM and select features.

고정하였다.

IV. 실험 및 결과

4.1 데이터 세트

본 연구의 사용된 데이터는 선행 연구에서 배포한 GCC, ICC, MSVS, Xcode 총 4개의 컴파일러를 통해 컴파일 된 실행파일이다[5]. 해당 데이터는 수업 및 Codejam을 통하여 수집한 데이터로 실험에서는 각 컴파일러 당 150개의 실행파일을 이용하여 총 600개의 파일을 대상으로 진행하였다.

4.2 특징 선택 및 모델 선택에 따른 차이

바이너리를 이용한 컴파일러 분류 실험에서 기존 회귀 방식을 이용한 특징 선택 방법에서 상호의존정보(mutual information)로 변경하였으며, 이에 정확도가 변하는 것을 확인할 수 있었다. 이에 특징 선택 방법의 차이가 어느 정도의 차이를 가져올 수 있는지를 확인하기 위해 상호의존정보, 정보 이득(information gain), 빈도수(item frequency)를 달리하여 실험을 진행하였다. 결과는 Table 1.과 같다. idiom은 해당 컴파일러로부터 도출된 Idiom 템플릿 특징의 수를 나타낸다. Accuracy는 f-sc-ore를 통한 정확도 산출 결과이며, Runtime은 실행 시간이다. 특징 선택 과정을 거치지 않는 경우, 충분한 학습 데이터가 없다면 낮은 분류 정확도를 보이며, 충분한 데이터가 있더라도 실행 시간이 길다는 문제점을 갖는다. 정보 이득을 이용한 경우 더 높은 효율을 보이며, 이는 클래스에 따라 특징 간의 상호의존정보를 이용하는 것보다 엔트로피를 이용한 방법이 효율적일 수 있음을 알 수 있었다.

Table 1. Comparison of classification result according to feature selection.
(- : no selection / if : item frequency / ig : information gain / mi : mutual information)

		idiom	Accuracy	Runtime
GCC	-	2,253,229	95.0	61179336
	if		94.03	3159741
	ig		95.23	5348468
	mi		93.72	10196556
ICC	-	179,694	18.33	57758196
	if		92.45	3126210
	ig		99.46	4588761
	mi		99.46	9626366
MSVS	-	112,003	13.75	59059836
	if		88.56	3259846
	ig		96.74	4347295
	mi		84.89	9843306
Xcode	-	93,639	13.75	59837496
	if		91.93	3074920
	ig		98.96	4490277
	mi		90.06	9972916

또한 확률 모델인 linear-chain CRF 모델에서 비확률 모델인 Linear SVM 모델로 변경 시 나타나는 차이를 확인하기 위해 동일한 데이터 및 특징 선택 방법을 사용하여 실험을 진행하였다. 결과는 Table 2.와 같으며, 각 컴파일러에 대하여 linear-chain CRF 모델은 97.96의 정확도를 보이며, SVM 모델은 98.1%의 정확도로 0.2%의 미미한 차이를 보이고 있다. 하지만 실행 속도는 54.7배가량의 차이를 보였으며, 이는 입력된 특징의 순서에 대한 분류 확률을 계산하는 산출 시간에 따른 차이임을 알 수 있었다.

Table 2. Difference between probability model and non-probability model.

	Accuracy	Runtime
linear-chain CRF	97.96	4066755
Linear SVM	98.1	74262

4.3 모델 및 특징 최적화에 따른 차이

4.2 단락에서 제안하는 프레임워크를 통하여 모델 및 특징 선택으로 인해 컴파일러 분류 정확도 및 실행 시간에 얼마 영향을 미치는가(Sensitive)에 대한 실험을 진행하였다. 이번 실험은 제안하는 프레임워크를 통해 분류 모델의 특징 및 튜닝 최적화로 인하여 선행 연구보다 얼마나 좋은 결과를 보이는지 확인하기 위한 실험을 진행하였다. 비교를 위해 선택한 선행 연구는 linear-chain CRF[9], Linear SVM[10], Bincomp[5], Oba2[6]이다. 각 선행 연구는 바이너리 기반 컴파일러 분류에 대한 대표적인 선행 연구이며, 제안한 프레임워크의 기반이 되는 연구이기에 선택하였다. 제안하는 프레임워크의 분류 모델인 SVM 모델을 튜닝하기 위해 커널은 선형(linear), 다항(polynomial), 시그모이드(sigmoi-d), 가우시안 기저 함수(radial basis function)를 대상으로 진행하였다.

최종 실험 결과는 Table 3.을 통해 확인할 수 있

Table 3. Classification result comparison between probability model and non-probability model. (model 1 : linear-chain CRF / model 2 : Linear SVM / model 3 : Bincomp / model 4 : Oba2 / model 5 : proposed framework)

		Idiom	Accuracy	Runtime
model 1	GCC	20	96.60	5035872
	ICC	41	99.52	5073504
	MSVS	32	96.78	5105050
	Xcode	31	98.96	5056588
model 2	GCC	20	97.80	73424
	ICC	37	99.52	71345
	MSVS	32	97.82	70146
	Xcode	31	98.96	70684
model 3	GCC	20	98.20	124872
	ICC	36	99.70	126587
	MSVS	30	97.97	133654
	Xcode	31	99.60	114873
model 4	GCC	20	98.20	106931
	ICC	36	99.70	108976
	MSVS	29	98.60	101621
	Xcode	28	99.60	105987
model 5	GCC	16	98.20	86443
	ICC	20	99.70	82103
	MSVS	22	98.60	81459
	Xcode	22	99.60	87049

다. Idiom은 각 컴파일러 분류를 위한 특징 선택 수를 나타내며, Accuracy는 f-score을 통한 정확도를, Runtime은 실행 시간을 나타낸다. 실험 결과에서 가우시안 기저 함수 단일 커널을 이용하였을 때 가장 최적의 결과를 보였다. 이는 특징 간의 복잡한 결정경계선에 대하여 합성 커널이 아닌 가우시안 기저 함수 커널을 이용하여도 충분히 분류할 수 있다는 결과로 해석하였다. 전체적으로 제안한 방법이 낮은 양의 특징을 선택하였으며, ICC 컴파일러 분류에서는 최대 48.78%의 특징 감소율을 보였다. 또한 5개의 비교 모델 중 가장 높은 분류 정확도를 보였으며, 실행 시간은 최대 51배가량의 차이를 보였다. 이는 최적화 튜닝 및 특징 선택으로 인한 결과로 해석하였다. linear-chain CRF은 가장 낮은 정확도와 느린 실행속도를 보였다. 또한 특징 선택 시 별도의 특징 선택 방법을 결정해야 하며, 가장 많은 양의 특징이 선택되었다. Linear SVM은 가장 빠른 실행 시간을 보였으나, ICC 컴파일러 이외에는 별도의 특징 감소가 보이지 않으며, 정확도 역시 최대 0.78% 낮게 나타났다. Bincomp, Oba2는 분류 정확도가 제안하는 프레임워크와 비슷하게 나타났으나, 컴파일러에 따라 최대 16개의 특징 선택 차이를 보이며 실행 시간 역시 제안하는 방법보다 느림을 확인할 수 있었다.

V. 결 론

본 논문에서는 바이너리를 기반으로 컴파일러 분류 실험을 통해 특징 선택 및 모델 선택이 분류 정확도에 미치는 민감성에 대해 연구를 진행하였다. 선행연구를 기반으로 바이너리에서 특징 정의 및 컴파일러 분류 실험을 하였으며, 실험 간에 특징 선택 및 모델, 최적화에 따른 민감성을 확인하고자 하였다. 최적화 실험을 위해 유전 알고리즘을 이용한 특징 선택 및 모델 최적화 프레임워크를 제안하였으며, 분류 정확도, 실행 시간, 특징 선택에 더 좋은 성능을 보였다. 본 연구는 바이너리 분석을 통한 분류, 탐지, 식별에 대하여 특징 선택 및 모델 선택 최적화에 대하여 고민의 필요성을 재고할 수 있으며, 제안하는 프레임워크를 통해 특징 선택 및 모델 최적화의 가능성을 확인할 수 있었다.

차후 연구에서는 제안하는 유전 알고리즘에 대한 최적화 방법에 있어 속성 파리미터 확대, 유사 방법 및 최적화에 대한 차이에 대하여 실험을 확장해 나갈 예정이다.

References

[1] Y. Ye, T. Li, D. Adjeroh, and S.S. Iyengar, "A survey on malware detection using data mining techniques," ACM Computing Surveys (CSUR), vol. 50, no. 2, pp. 1 - 41, Jun. 2017.

[2] Su-jeong Kim, Ji-hee Ha, Soo-hyun Oh, and Tae-jin Lee, "A Study on Malware Identification System Using Static Analysis Based Machine Learning Technique," Journal of the Korea Institute of Information Security & Cryptology, vol. 29, no. 4, pp. 775 - 784, Aug. 2019.

[3] E.H. Spafford, and S.A. Weeber, "Software forensics: Can we track code to its authors?," Computers & Security, vol. 12, no. 6, pp. 585 - 595, Feb. 1993.

[4] E. Stamatatos, "A survey of modern authorship attribution methods," Journal of the American Society for information Science and Technology, vol. 60, no. 3, pp. 538-556, Mar. 2009.

[5] A. Rahimian, P. Shirani, S. Airbaee, and L. Wang, "Bincomp: A stratified approach to compiler provenance attribution," Digital Investigation, vol. 14, pp. 146-155, Aug. 2015.

[6] S. Alrabaee, N. Saleem, S. Preda, L. Wang, and M. Debbabi, "OBA2: An Onion approach to Binary code Authorship Attributio," Digital Investigation, vol. 11, pp. 94-103, Mar. 2014.

[7] S. Alrabaee, P. Shirani, M. Debbabi, and L. Wang, "On the Feasibility of Malware Authorship Attribution," *International Symposium on Foundations and Practice of Security. Springer*, pp. 256-272, Jan. 2017.

[8] N.E. Rosenblum, X. Zhu, and B.P. Miller, "Learning to Analyze Binary Computer Code," Proceedings of the the Twenty-Third AAAI Conference on Artificial Intelligence, pp. 798-804, Jul. 2008.

[9] N.E. Rosenblum, B.P. Miller, and X. Zhu, "Extracting compiler provenance from program binaries," Proceedings of the 9th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering. ACM, pp. 21-28, Jun. 2010.

[10] N.E. Rosenblum, X. Zhu, and B.P. Miller, "Who wrote this code? identifying the authors of program binaries," European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, pp. 172-189, 2011.

[11] Dong-Seong Kim, Ha-Nam Nguyen, and Jong-Sou Park, "Genetic algorithm to improve SVM based network intrusion detection system," In 19th International Conference on Advanced Information Networking and Applications, vol. 1, pp. 155-158, 2005.

[12] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," Computers & Electrical Engineering, vol. 61, pp. 266-274, Oct. 2017.

[13] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," In Ndss, vol. 14, pp. 23-26, 2014.

[14] M. N. Yusoff, and A. Jantan, "Optimizing Decision Tree in Malware Classification System by using Genetic Algorithm Naïve Bayes : Sensitive to the correlated attributes," computing, vol. 10, no. 14, pp. 694-713, 2011.

〈저자소개〉

황 철 훈 (Cheol-Hun Hwang) 학생회원
2019년 가천대학교 컴퓨터공학과(공학사)
2019~현재 가천대학교 일반대학원 소프트웨어학과 석사과정
〈관심분야〉 정보보호, 머신러닝, 인공지능, 모바일

신 건 윤 (Gun-Yoon Shin) 학생회원
2017년 가천대학교 인터랙티브 미디어 융합학과(공학사)
2018년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2018~현재 가천대학교 컴퓨터공학과 박사과정
〈관심분야〉 기계 학습, 악성코드 분석, 공격자 식별, 저자 분석, 인공지능

김 동 욱 (Dong-Wook Kim) 학생회원
2015년 가천대학교 컴퓨터공학과(공학사)
2017년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2017~현재 가천대학교 컴퓨터공학과 박사과정
〈관심분야〉 Data Mining, 인공지능, Data fusion, Anomaly Detection

한 명 묵 (Myung-Mook Han) 종신회원
1980년 연세대학교 공과대학(공학사)
1987년 뉴욕공과대학교 대학원 컴퓨터공학과(공학석사)
1997년 오사카시립대학교 대학원 정보공학부(이학박사)
1998~현재 가천대학교 소프트웨어학과 교수
〈관심분야〉 정보보호, 인공지능