정 보 보 호 학 회 지
제27권 제3호, 2017. 6

기계 학습을 활용한 변종 악성코드 식별 연구 동향 분석 (Analysis of Research Trend on Machine Learning Based Malware Mutant Identification)

유 정 빈*, 신 민 식, 권 태 경**

요 약

기하급수적으로 증가하고 있는 변종 악성코드에 대응하기 위한 식별 연구가 다양화 되고 있다. 최근 연구에서는 기존 악성코드 분석 기술 (정적/동적)의 개별 사용 한계를 파악하고, 각 방식을 혼합한 하이브리드 분석으로 전환하는 추세이다. 나아가 변종 식별이 어려운 악성코드를 더욱 정확하게 식별하기 위해 기계 학습을 적용하기에 이르렀다. 이에 따라, 본 논문에서는 변종 악성코드 식별을 위해 각 연구에서 활용한 기계 학습 기술과 사용한 악성코드 특징을 중심으로 변종 악성코드 식별 연구를 분류 및 분석한다.

Ⅰ. 서 론

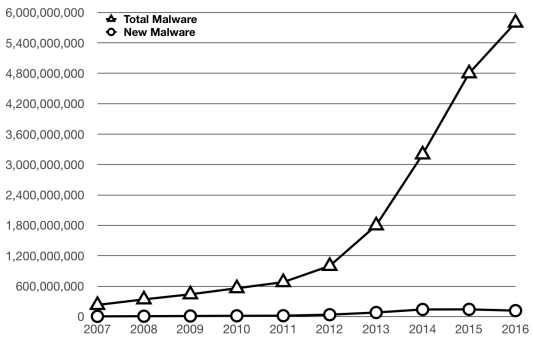
자동화된 악성코드 생성도구 (Automatic malware creation toolkit)가 인터넷을 통해 유포됨에 따라 악성코드 출현 개수가 기하급수적으로 증가하고 있다. 2017년 AV-Test 악성코드 동향 보고서에 따르면 DDoS (Distributed Denial of Service), 스팸 발송, APT (Advanced Persistent Threat) 공격 등에 사용된 악성코드는 연간 기준 약 60억 개에 달한다 [1]. 그러나 그림 1과 같이 전체 악성코드 가운데 신종 악성코드는 6억 개 미만으로 대부분의 악성코드가 기존 악성코드의 변종 (Mutant)임을 알 수 있다.

변종 악성코드는 패턴매칭 (Pattern matching)을 기반으로 한 악성코드 탐지 방식을 회피하기 위해 기존 악성코드를 변형한 악성코드이다. 변종 악성코드는 변형 엔진 (Mutation engine) 복잡도에 따라 다형성 (Polymorphic) 악성코드, 변성 (Metamorphic) 악성코드로 분류된다. 다형성 악성코드는 단순히 외형을 변형하기 위해 암호화 또는 데이터 확장을 기존 악성코드에 적용한다. 반면, 변성 악성코드는 다형성 악성코드보다 진보된 형태로 유포될 때마다 기존 악성코드가 사용하는 명령어 코드 재정렬, 레지스터 변경, 난독화 (Obfuscation), 의미 없는 명령어 및 분기문 등을 추가해 코드가 다시 작성된다.

기하급수적으로 증가하는 악성코드 가운데 99% 이상이 변종 악성코드라는 점에 초점을 맞추어 변종 악성코드 식별 연구가 활발하게 진행되고 있다. 그러나 악성코드 식별 기술이 진화함에 따라 악성코드 제작자의 제작 기술 또한 더욱 교묘해 지고 있다. 예를 들어, 무파일 (Fileless) 악성코드는 감염된 컴퓨터의 메모리에만 존재하는 악성코드이다. 기존 악성코드 식별 방식으로는 이러한 새로운 변종 악성코드를 식별하기 힘들다. 이에 따라, 악성코드 변종 식별 연구는 여전히 학계에서 중요한 연구 주제이다.

최근 연구에서는 변종 식별이 어려운 악성코드를 더욱 정확하게 식별하기 위해 다양한 기계 학습을 적용하

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음" (IITP-2017-2012-0-00646)
* 주저자, symnoisy@yonsei.ac.kr
** 교신저자, taekyoung@yonsei.ac.kr (Corresponding author)



〔그림 1〕 전체/신종 악성코드 증가 추이

는 추세이다. 본 논문에서는 변종 악성코드 식별을 위해 각 연구에서 활용한 기계 학습 기술과 사용한 악성코드 특징을 중심으로 변종 악성코드 식별 연구를 분류 및 분석한다.

Ⅱ. 기반 기술 소개

기계 학습을 활용한 악성코드 식별 연구의 동향 분석을 수행하기 전 문헌 연구에 요구되는 기반 기술을 소개한다.

2.1. 기계 학습 알고리즘

기계 학습은 프로그래밍 되지 않은 데이터를 컴퓨터가 학습할 수 있도록 돕는 컴퓨터 분야이다. 특히, 명시적인 프로그래밍으로 좋은 성능을 기대하기 어려운 부분에 주로 사용되며, 최근에는 알려지지 않은 변종 악성코드를 더욱 정확하게 식별하기 위해 악성코드 식별에도 접목하고 있는 추세이다. 학습 데이터 의존도에 따라 지도 학습 (Supervised Learning), 비지도 학습 (Unsupervised Learning)으로 분류된다.

2.1.1. 지도 학습

사전에 식별된 훈련 데이터 셋을 통해 최적 함수를 도출하는 방법이다. 분류 (Classification) 알고리즘, 회귀 (Regression) 알고리즘이 이에 해당하며, 변종 악성코드 식별 분야에서는 신뢰할 수 있는 레이블링 정보를 포함한 악성코드를 토대로 알고리즘을 학습시키고, 새로 유입되는 악성코드를 식별하는 연구에서 주로 사용된다.

2.1.2. 비지도 학습

분류되지 않은 데이터 셋을 사용하는 머신러닝 알고리즘이다. 클러스터링 (Clustering) 알고리즘이 이에 해당하며, 거리 기반 알고리즘 (예: Euclidean, Cosine)으로 알려지지 않은 데이터 간 특징을 계산해 분리된 데이터 구조로 나누는 것을 목표로 한다. 지도 학습 알고리즘을 적용한 연구와는 달리 신뢰할 수 있는 레이블링 정보가 없더라도 변종 악성코드를 식별할 수 있다.

2.2. 악성코드 분석 기술

악성코드 분석 기술에 사용하는 특징에 따라 정적 분석, 동적 분석, 하이브리드 분석으로 분류된다. 각 분석 방식의 특징은 아래와 같으며, 최근에는 정적/동적 개별 분석의 한계를 완화하기 위해 하이브리드 (Hybrid) 분석을 활용하는 추세이다.

2.2.1. 정적 분석

악성코드를 실행하지 않고 사용하는 PE (Portable Executable) 정보, 동적 라이브러리 정보 (DLL; Dynamic Linking Library), Opcode 시퀀스, 바이트 시퀀스, 어셈블리 명령어 (x86), API 함수 이름, ASCII 포맷 16진수 코드, 제어 흐름 그래프 (CFG; Control Flow Graph), 디스어셈블 된 바이너리, 시스템 명령어 추적, 정적 바이너리 정보, 문자열 정보, 섹션 정보, Entropy, 파일크기 등을 활용해 악성코드의 구조를 분석한다 [2,3,4,5,6]. 정적 특징을 사용한 정적 분석은 악성코드를 빠르게 식별할 수 있다는 장점이 있지만 난독화 (Obfuscation), 패킹 (Packing) 등이 적용된 변종 악성코드를 정확하게 식별할 수 없다는 한계가 있다.

2.2.2. 동적 분석

통제가 가능한 가상 환경 (예: 가상 머신, 시뮬레이터, 샌드박스 등)에서 악성코드를 직접 실행시켜 사용하는 API 함수, 네트워크 활동, 프로세스, 윈도우 서비스, 파일 시스템 변화, 레지스트리 키, Mutex, 실행되는 명령어, 파일 읽기 정보 등을 활용해 악성코드의 행위를 분석한다 [7,8,9,10,11]. 동적 특징을 사용한 동적 분석은 다형성 악성코드와 변성 악성코드에 어느 정도 유연하다

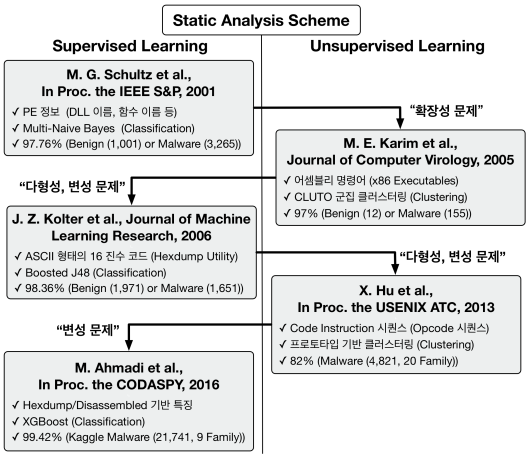
는 장점을 갖고 있지만 시간 비용이 높다는 한계가 있다.

2.2.3. 하이브리드 분석

정적/동적 분석을 개별적으로 사용해 악성코드를 식별할 때, 각 분석 방식이 갖는 한계를 상호보완하기 위해 두 분석 방식을 혼합해 사용하는 방식이다 [12,13,14,15]. 잘 구성된 하이브리드 분석 도구는 각 분석 방식이 갖는 한계를 최소화할 수 있다는 장점이 있지만, 그렇지 못한 하이브리드 분석 도구는 각 분석 방식이 갖는 한계를 모두 가질 수 있다.

Ⅲ. 기계 학습과 정적 분석을 이용한 악성코드 식별 기술

본 문단에서는 그림 2와 같이 지도/비지도 학습을 기반으로 정적 분석을 활용한 변종 악성코드 식별 연구를 살펴본다.



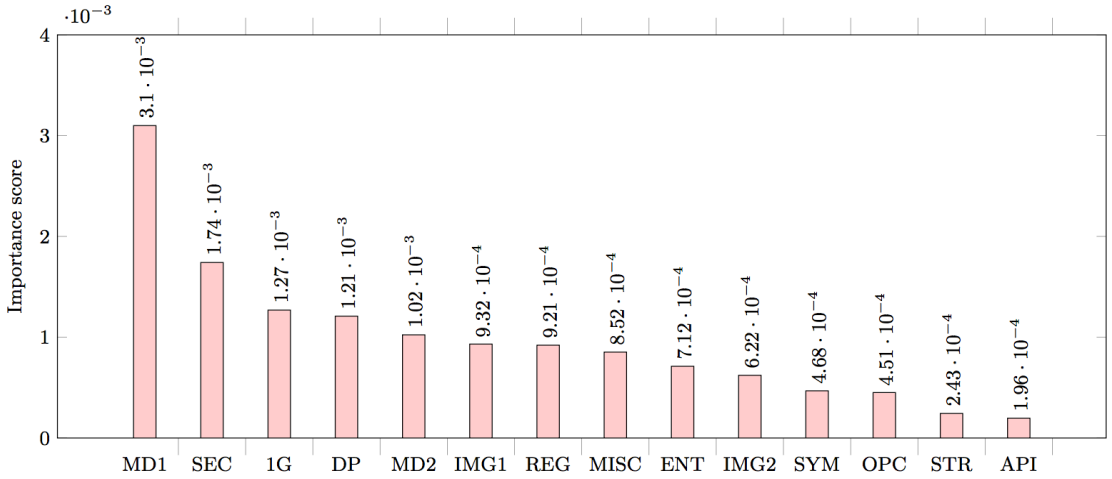
〔그림 2〕 정적 특징 기반 악성코드 식별 연구

3.1. 지도 학습을 활용한 연구

Scultz 등은 1,001개의 정상 프로그램과 3,265개의 악성코드를 대상으로 추출한 PE 정보 (예: 파일 크기, DLL 이름, API 이름), 문자열 정보, Hexdump 도구를 이용해 추출한 특징을 토대로 악성코드 식별 연구를 진행했다[2]. 전통적인 시그니쳐 식별 방식, RIPPER, Naive Bayes, Multi Naive Bayes 분류 알고리즘을 각기 프레임워크에 적용해 성능을 측정한 결과, Multi-Naive Bayes 분류 알고리즘을 적용했을 때97.76%의 정확도로 가장 성능이 좋았다. 전통적인 시그니쳐 방식 (49.28%)보다 높은 정확도 (97.76%)로 악성코드를 식별할 수 있다는 장점이 있지만, 확장성이 좋지 못하다는 한계가 있다.

Kolter 등은 Hexdump 도구를 이용해 ASCII 형태의 16진수 코드를 추출하여 정상/악성코드 식별 연구를 진행했다 [3]. 476 개의 정상프로그램과 561 개의 악성코드를 토대로 가장 많이 사용되는 500 개의 4-gram 정보를 선정했으며, 이를 포함한 1,971 개의 정상 프로그램과 1,651 개의 악성코드가 프레임워크 성능측정에 사용했다. 프레임워크 성능을 극대화하기 위해 총 7개의 분류 (classification) 알고리즘 가운데 가장 좋은 성능을 보이는 알고리즘을 선정했다 (Boosted J48, SVM, IBk (k=5), Boosted SVM, Boosted Naive Bayes, J48, Naive Bayes). 10-fold cross validation을 활용해 각 분류 알고리즘을 적용한 프레임워크 성능을 측정한 결과, Boosted J48이 98.36%의 정확도로 가장 높은 성능을 보였다. 제안하는 방식을 토대로 실제 업계에서 사용할 수 있는 프로토타입을 만들었다는 장점이 있지만, 학습된 정적 특징 (가장 많이 사용되는 500 개의 4-gram)에 의존한다는 한계가 있다.

Ahmadi 등은 IDA Pro를 이용해 추출한 disassembled 기반 (MD1은 file 크기, 제 1 바이트 시퀀스 주소, 1G은 바이너리 파일의 16 진수 값을 의미한다), hex dump 기반 악성코드 특징 (SEC은 섹션정보, DP는 Data definition Proportion 정보를 의미한다)을 활용해 악성코드 식별연구를 진행했다 [4]. 특히 Microsoft Kaggle 변종 악성코드 식별 대회에서 제공한 21,741 (9 family)개의 악성코드를 대상으로 추출한 특징 가운데 악성코드를 효율적으로 식별할 수 있는 조합을 선정하기 위해 forward stepwise selection 알고리즘을 적용했다. Forward stepwise selection 알고리즘은 각 특징을 순차적으로 대입해가면서 최적의 특징 조합을 선정하는 알고리즘이다. 결과적으로 그림 3과 같이 각 악성코드 특징의 중요도를 산정했다. XGBoost 분류 알고리즘을 적용해 악성코드를 식별한 결과, 99.42%의 정확도를 보였다. 정적 특징을 활용해 악성코드를 식별했지만 다형성 악성코드에 어느 정도 유연하다는 장점이 있지만 Kaggle 대회에서 제공한 악성코



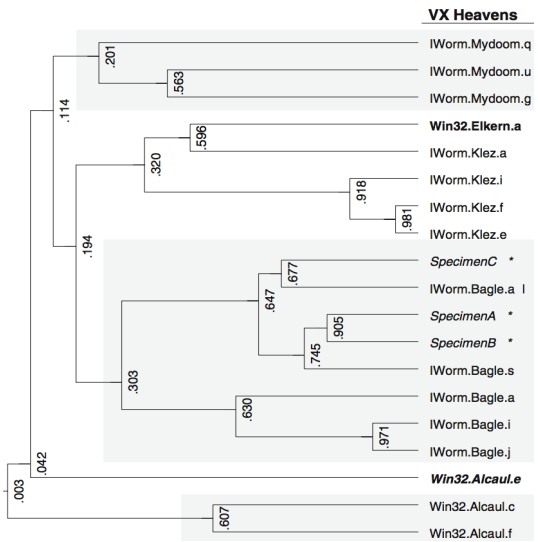
〔그림 3〕 Forward stepwise selection 알고리즘으로 악성코드 정적 특징의 중요도를 산정한 그래프

드만을 대상으로 성능을 측정했다는 한계가 있다.

3.2. 비지도 학습을 활용한 연구

Karim 등은 악성코드에서 추출한 어셈블리 명령어(x86)를 활용해 정상/악성코드 식별 연구를 진행했다[5]. 12 개의 정상프로그램과 155개의 악성코드를 토대로 10-gram과 10-perm을 적용해 각 악성코드에서 사용되는 어셈블리 명령어의 빈도수 테이블을 생성했다. 생성된 빈도수 테이블을 토대로 CLUTO 군집 클러스터링을 적용하여 그림 4와 같이 dendogram을 생성해 식별한 결과, 97%의 정확도를 보였다. 재정렬 엔진 (예: 명령어, 블록 서브루틴 등)이 적용된 변성 악성코드에 유연하고 새로운 악성코드가 유입되었을 때 계통을 파악하는데 효율적이라는 장점이 있지만, UPX 패킹이 적용된 악성코드나 패킹이 적용되지 않은 악성코드를 대상으로 성능을 측정했기 때문에 다형성 악성코드에 유연하지 못하다.

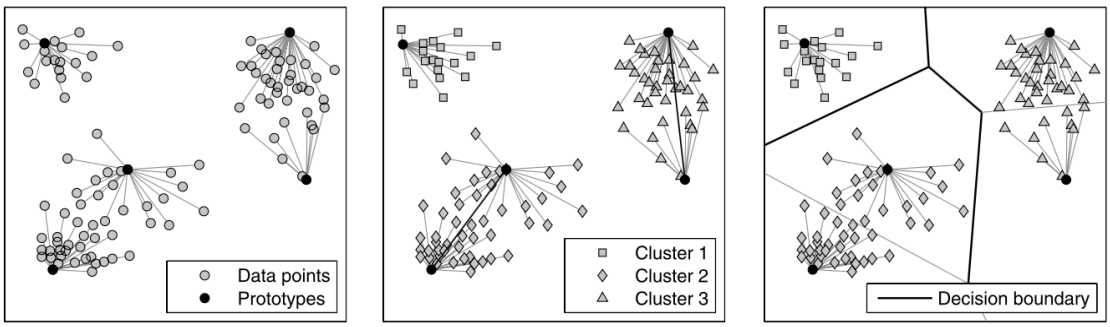
Hu 등은 악성코드에서 추출한 opcode 시퀀스를 활용해 악성코드 식별 연구를 진행했다 [6]. 4,821 (20 family)개의 악성코드를 토대로 4-gram 정보를 추출했으며, '차원의 저주 (Curse of dimensionality)' 문제를 해결하기 위해 Feature Hashing을 적용했다. Feature Hashing을 적용한 악성코드의 4-gram 정보를 토대로 프로토타입 기반 클러스터링 (Prototype-based clustering) 알고리즘을 적용하여 악성코드를 식별한 결과, 82%의 정확도를 보였다. 프레임워크의 시간 비용 문제를 해소하기 위해 Feature Hashing과 프로토타입 기반 클러스터링 알고리즘 적용을 시도했다는 장점이 있지만 바이너리/명령어 레벨 난독화에 취약하다는 한계가 있다.



〔그림 4〕 CLUTO 클러스터링을 활용해 dendogram 생성 및 변종 악성코드를 식별한 그림

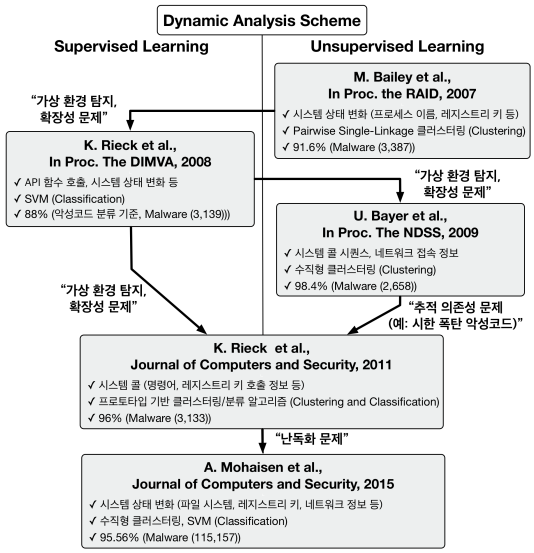
Ⅳ. 기계 학습과 동적 분석을 이용한 악성코드 식별 기술

본 문단에서는 그림 6과 같이 지도/비지도 학습을 기반으로 동적 분석을 활용한 변종 악성코드 식별 연구를 살펴본다.



(a) 악성코드 내 프로토타입 선정 (b) 프로토타입 기반 클러스터링 적용 (c) 프로토타입 기반 분류알고리즘 적용

〔그림 5〕 프로토타입 기반 클러스터링, 분류 알고리즘을 적용해 변종 악성코드를 식별하는 모식도



〔그림 6〕 동적 특징 기반 악성코드 식별 연구

4.1. 지도 학습을 활용한 연구

Rieck 등은 CWSandbox를 활용해 3,139 개의 악성코드로부터 추출한 API 함수 호출, 시스템 상태 변화 (예: 파일 시스템, 레지스트리 키), 프로세스 정보, 네트워크 활동, 윈도우 서비스 정보 등을 활용해 악성코드 식별 연구를 진행했다 [7]. 추출한 악성코드의 특징들을 key, value 쌍의 빈도수 테이블로 표현해 SVM 분류 알고리즘을 적용한 결과, 88%의 정확도를 보였다. 난독화가 적용된 악성코드에 유연하다는 장점이 있지만 확장성 문제와 가상화 환경을 탐지하는 변종 악성코드에 유연하지 못하다는 한계가 있다.

4.2. 비지도 학습을 활용한 연구

Bailey 등은 Windows XP 가상화 환경에서 3,387개의 악성코드로부터 추출한 시스템 상태 변화 (프로세스 이름, 레지스트리 키) 정보를 활용해 악성코드 식별 연구를 진행했다 [8]. 추출한 악성코드 특징의 빈도수 테이블을 생성하고 pairwise single-linkage 클러스터링 알고리즘을 적용한 결과 91.6%의 정확도를 보였다. 난독화가 적용된 악성코드에 유연하다는 장점이 있지만 확장성 문제 (악성코드 한 개를 식별하는데 5분의 시간 소요)와 가상화 환경을 탐지하는 변종 악성코드에 유연하지 못하다는 한계가 있다.

Bayer 등은 ANUBIS 샌드박스를 활용해 2,658개의 악성코드로부터 추출한 시스템 콜 시퀀스, 네트워크 접속 정보를 활용해 악성코드 식별 연구를 진행했다 [9]. 악성코드로부터 추출한 시스템 콜 시퀀스를 대상으로 taint tracking을 수행해 시스템 함수가 사용하는 인자, 반환 값들의 후속호출 정보를 추적해 프로필 정보를 생성하고 수직형 클러스터링 알고리즘을 적용한 결과, 98.4%의 정확도를 보였다. Taint tracking을 통해 악성코드의 행위를 정확하게 식별할 수 있다는 장점이 있지만, 추적 의존성 (Trace dependence) 문제로 특정 시간이 지정된 시한폭탄 (Time-bomb) 악성코드를 정확히 식별할 수 없다는 한계가 있다.

4.3. 지도/비지도 학습을 활용한 연구

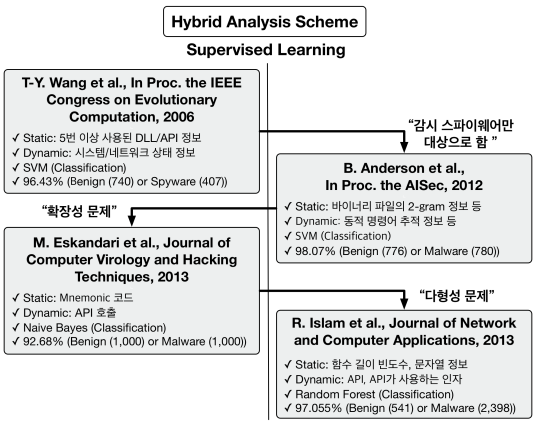
Rieck 등은 CWSandbox를 활용해 3,133개의 악성코드로부터 추출한 시스템 콜 정보 (예: 명령어 정보,

레지스트리 키 호출 정보 등)를 활용해 악성코드 식별 연구를 진행했다 [10]. 추출한 악성코드 특징을 중요도에 따라 3 개의 계층으로 분류했다. 분류한 특징을 토대로 그림 5와 같이 변종 악성코드 그룹을 가장 잘 분류할 수 있는 대표 악성코드를 프로토타입 기반 클러스터링을 통해 선정하고 (a), 클러스터링을 수행했다 (b). 그리고 분류알고리즘 (c)을 적용한 결과, 96%의 정확도로 악성코드를 식별했다. 프로토타입 기반 기계 학습 알고리즘을 적용해 기존 동적 분석 방식 (7일 기준 5 GB)에 비해 메모리 소모가 낮고 다형성 악성코드에 유연하다는 장점이 있지만 난독화가 적용된 악성코드를 정확하게 식별할 수 없다는 한계가 있다.

Mohaisen 등은 연구를 통해 제작한 악성코드 동적 행위 분석도구 (AMAL)을 활용해 악성코드 식별 연구를 진행했다 [11]. 115,157 개의 악성코드로 부터 추출 시스템 상태 변화 (파일 시스템 정보, 레지스트리 키, 네트워크 정보 등)를 추출해 정규화하고 수직형 (Hierarchical) 클러스터링과 SVM 분류 알고리즘을 적용한 결과, 95.56%의 정확도를 보였다. 자체 제작한 동적 분석 도구를 활용해 악성코드를 식별했다는 장점이 있지만 가상 환경을 탐지하는 변종 악성코드에 유연하지 못하다는 한계가 있다.

Ⅴ. 기계 학습과 하이브리드 분석을 이용한 악성코드 식별 기술

본 문단에서는 그림 7과 같이 지도/비지도 학습을 기반으로 하이브리드 분석을 활용한 변종 악성코드 식별 연구를 살펴본다.



〔그림 7〕 하이브리드 특징 기반 악성코드 식별 연구

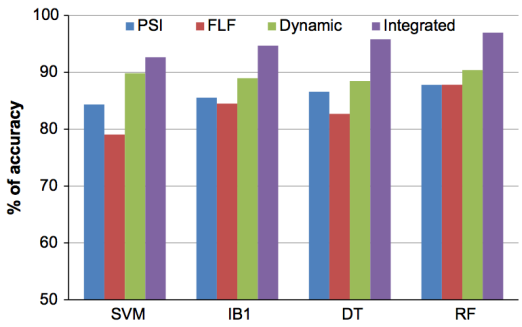
5.1. 지도 학습을 활용한 연구

Wang 등은 740개의 정상 프로그램과 407개의 악성코드에서 추출한 5번 이상 사용된 import table 정적 정보와 Windows XP에서 동적으로 추출한 시스템 상태, 네트워크 상태정보를 활용해 정상/악성코드 식별 연구를 진행했다 [12]. Information Gain (IG) 알고리즘을 적용해 최적의 특징 조합을 선정했으며 (67 개의 정적 특징, 14 개의 동적 특징), SVM 분류 알고리즘을 적용했다. k-fold cross validation (k = 2, 5, 10, 100)을 활용해 성능을 측정한 결과 96.43% 정확도를 보였다. 다른 AV 벤더 (예: Kaspersky, Norton, Pccillin, Nod32)보다 높은 정확도를 보였다는 장점이 있지만, 감시 스파이웨어 (Surveillance spyware)만을 대상으로 했기 때문에 다른 유형의 악성코드에 유연하지 못하다는 한계가 있다.

Anderson 등은 776개 정상 프로그램과 780개 악성코드에서 추출한 바이너리 파일의 2-gram 정보와 IDA Pro를 활용해 추출한 Disassembled binary, Entropy 정적 특징과 Xen Virtual Machine을 활용해 추출한 동적 명령어 추적 정보를 토대로 악성코드 식별 연구를 진행했다 [13]. '차원의 저주' 문제를 해결하기 위해 경험적 결과를 활용하여 악성코드 특징들을 카테고리로 분류했다 (정적 특징 86 개, 동적 특징 94 개). 분류한 악성코드 특징들을 토대로 SVM 분류 알고리즘을 적용하여 10-fold cross validation으로 성능을 측정한 결과, 98.07%의 정확도를 보였다. 새로운 악성코드 특징을 쉽게 추가할 수 있다는 점에서 확장성이 높다는 장점이 있지만, 악성코드 특징을 모두 추출하는데 약 5분 (303.52 초)이 걸린다는 한계가 있다.

Eskandari 등은 1,000개의 정상 프로그램과 1,000개의 악성코드에서 추출한 mnemonic 코드 정적 특징과 API 시퀀스 동적 특징을 활용해 악성코드 식별 연구를 진행했다 [14]. Mnemonic 코드를 활용해 제어 흐름 그래프를 생성하고 CALL/RET 명령어를 JMP 명령어로 변경, 제어 흐름 그래프 내에서 발생하는 API 호출을 기준으로 동적 특징을 결합하여 추상화했다. 추상화한 정상/악성코드 특징을 토대로 Naive Bayes 분류 알고리즘을 적용하여 10-fold cross validation으로 성능을 측정한 결과, 92.68% 정확도를 보였다.

제어 흐름 그래프를 기반으로 동적 특징을 추출하기



〔그림 8〕 다양한 분류알고리즘에 문자열 정보 (PSI), 함수 길이 빈도수 (FLF), 정적, 동적, 하이브리드 특징을 고려해 각 분류기의 정확도를 측정한 도표

때문에 다른 동적 분석 도구에 비해 빠르다는 장점이 있지만 패킹이 되지 않은 네트워크 웜 (Network worm)만을 대상으로 했기 때문에 다형성 악성코드에 유연하지 못하다는 한계가 있다.

Islam 등은 IDA Pro를 이용해 추출한 정적 특징 (함수 길이 빈도수, 문자열 정보)와 샌드박스를 이용해 추출한 동적 특징 (API, API가 사용하는 인자)를 활용해 정상/악성코드 식별 연구를 진행했다 [15]. 541개 정상 프로그램과 2,398개 악성코드의 경험적 결과를 토대로 함수 길이 (50으로 설정), 문자열 정보 (1,742,490 개) 이진 테이블, API 빈도 테이블 (72,259 개)을 설정했다. 선정한 악성코드 특징을 토대로 그림 8과 같이 SVM, IB1 (Instance based along with Boosting technique), Decision Tree, Random Forest 분류 알고리즘을 적용해 프레임워크에 가장 적합한 알고리즘을 선정했다. 10-fold cross validation을 활용해 각 알고리즘의 성능을 측정한 결과 Random Forest가 97.055%로 가장 성능이 좋았다. 또한. 동적 특징만을 사용했을 때 (90.398%)보다 하이브리드 특징을 사용했을 때 높은 정확도 (97.055%)를 보일 수 있다는 가능성을 제시했지만 확장성이 좋지 못하다는 한계가 있다.

Ⅵ. 결 론

본 논문에서는 다양한 기계 학습을 활용한 변종 악성코드 식별 연구를 분류하여 살펴보았다. 초기에는 정적/동적 개별 분석을 기반으로 변종 악성코드 식별 연구가 발전해 왔으며, 최근에는 각 분석 방식의 한계를 극복하기 위해 하이브리드 분석 방식이 제안되고 있다 [12, 13, 14, 15]. 최근 연구에서는 변종 악성코드를 더욱 정확하게 식별하기 위해 기계 학습을 적용하고 있다. 변종 악성코드 식별 연구에 지도 학습을 적용한 경우, 초기 학습 데이터에 의존하기 때문에 신종 악성코드가 유입되면 정확하게 식별하기 힘들다는 문제가 있다. 반면, 비지도 학습을 적용한 경우, 거리 기반 알고리즘 (예: Euclidean, Cosine)으로 각 변종 악성코드간 관계를 식별하기 때문에 상대적으로 시간 비용이 높다는 문제가 있다. 이에 따라, 최근에는 각 기계 학습 알고리즘의 한계를 극복하기 위해 지도/비지도 학습을 혼합하여 사용하거나, 사전에 각 그룹의 프로토타입을 선정하여 시간 비용을 줄이는 연구가 제안되고 있다 [6,10].

참 고 문 헌

[1] https://www.av-test.org/en

[2] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In Proc. IEEE Symposium on Security and Privacy (S&P), pages 38 – 49, 2001.

[3] J. Z. Kolter, and M. A. Maloof. Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research, pages 2721 – 2744, 2006.

[4] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto. Novel feature extraction, selection and fusion for effective malware family classification. In Proc. Data and Application Security and Privacy (CODASPY), pages 183 – 194, 2016.

[5] M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida. Malware phylogeny generation using permutations of code. Journal of Computer Virology, 1(1-2):13 – 23, 2005.

[6] X. Hu, K. G. Shin, S. Bhatkar, and K. Griffin. MutantX-S: scalable malware clustering based on static features. In Proc. USENIX Conference on Annual Technical Conference (ATC), pages 187 – 198, 2013.

[7] K. Rieck, T. Holz, C. Willems, P. Düssel, and P.

Laskov. Learning and classification of malware behavior. In Proc. Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), pages 108 – 125, 2008.

[8] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In Proc. International Workshop on Recent Advances in Intrusion Detection (RAID), pages 178 – 197, 2007.

[9] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In Proc. Network and Distributed System Security Symposium (NDSS), volume 9, pages 8 – 11, 2009.

[10] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic Analysis of Malware Behavior Using Machine Learning. Journal of Computer and Security, 19(4):639 – 668, 2011.

[11] A. Mohaisen, O. Alrawi, and M. Mohaisen. Amal: High-fidelity, behavior-based automated malware analysis and classification. Journal of Computers and Security, 2015.

[12] T. Y. Wang, S. J. Horng, M. Y. Su, C. H. Wu, P. C. Wang, and W. Z. Su. A surveillance spyware detection system based on data mining methods. In Proc. IEEE Congress on Evolutionary Computation, pages 3236 – 3241, 2006.

[13] B. Anderson, C. Storlie, and T. Lane. Improving malware classification: bridging the static/dynamic gap. In Proc. Artificial Intelligence and Security (AISec), pages 3 – 14, 2012.

[14] M. Eskandari, Z. Khorshidpour, and S. Hashemi. Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection. Journal of Computer Virology and Hacking Techniques, 9(2):77 – 93, 2013.

[15] 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, 36(2): 646 – 656, 2013.

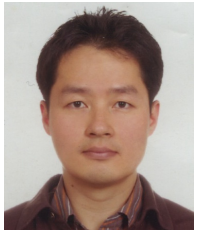
〈저자소개〉

유 정 빈 (JungBeen Yu)
학생회원

2016년 2월 : 대구카톨릭대학교 정보보호학 졸업
2016년 3월~현재 : 연세대학교 정보보호연구실 석사과정
관심분야 : 시스템 보안, 악성코드 탐지

신 민 식 (MinSik Shin)
학생회원

2016년 2월 : 연세대학교 컴퓨터공학과 졸업
2016년 3월~현재 : 연세대학교 정보보호연구실 석사과정
관심분야 : 시스템 보안, 네트워크 보안, 스마트폰 보안 등



권 태 경 (Taekyoung Kwon)
종신회원

1992년 2월 : 연세대학교 컴퓨터과학과 학사
1995년 2월 : 연세대학교 컴퓨터과학과 석사
1999년 8월 : 연세대학교 컴퓨터과학과 박사
1999년~2000년 : U.C. Berkely Post-Doc
2001년~2013년 8월 : 세종대학교 컴퓨터공학과 교수
2007년~2008년 : Univ. Maryland at College Park 교환교수
2013년 9월~현재 : 연세대학교 정보대학원 교수
관심분야 : 암호 프로토콜, 유저블 시큐리티, 사물인터넷 보안, 소프트웨어 보안 등