실행 개체의 행위-성능적 시각화를 통한 악성코드 분류에 대한 고찰

(주)이스트시큐리티 ▌김준섭
세종대학교 ▌박기웅*

1. 서 론

보안에서 악성코드로 인한 위협은 언제나 매우 위협적이었지만 공격의 목적이나 패러다임이 바뀜에 따라 그에 따른 악성코드 분석의 패턴이나 관점의 변화가 요구되고 있다. 예를 들어, 기존에는 불특정 다수를 대상으로 악성코드를 이용한 공격이 빈번히 발생하였다. 이를 통해 대량의 악성코드를 유포하여 비트코인 채굴 등에 활용하기 위한 봇넷을 구성하는 것이 주요 공격의 목적이었다 [1, 2]. 이와 같은 공격에 대한 대응은 사용자 컴퓨터에 침투한 악성코드의 탐지 및 제거를 통해 해결될 수 있었다. 하지만 최근에는 공격의 목적이나 대상이 변화하고 있는데, 대표적인 것이 랜섬웨어의 고도화이다. 랜섬웨어도 초기에는 불특정 다수를 대상으로 공격을 시도했지만, 최근에는 특정한 대상을 목표로 해서 공격이 성공할 때까지 고도화된 다양한 방법으로 집요하게 공격을 수행한다. 더구나 공격의 목적도 데이터를 암호화해서 돈을 요구하는 것뿐만 아니라 기업의 정보를 유출하여 2차 피해도 유발시키고 있다 [3]. 이와 같은 특정한 대상을 목표로 하는 고도화된 표적 공격에 대한 대응 전략으로 기존의 정형화된 악성코드의 탐지 및 제거로는 미봉책이 될 수 있다. 공격자가 누구인지, 공격의 목적과 기존의 공격 방법이 무엇인지를 파악하여 그에 맞는 대응을 해야 한다. 그러기 위해서는 먼저 탐지된 악성코드를 분석하고 분류해서 어떤 악성코드인지를 파악해야 한다. 분류 결과 기존에 본 적이 없는 전혀 새로운 종류의 악성코드라면 전문적인 분석가를 투입해서 빠르게 분석해서 대응하는 것이 최선이겠지만, 기존에 이미 알고 있던 악성코드와 같거나 유사하다면 공격자를 특정할 수 있게 된다. 그러면 공격자가 수행했던 이전의 공격을 분석하여 공격의 목적이나 방법을 파악할 수 있게 되고, 현재 진행 중인 공격이나 이후 진행될 공격도 어느 정도 예측하여 고도화된 대응을 할 수 있게 된다. 이렇게 악성코드를 분석하고 분류하는 작업은 최근의 고도화된 표적 공격에 대응하기 위한 필수불가결한 작업인 것이다. 하지만 악성코드 제작자들 역시 이런 분석과 분류작업을 방해하기 위해서 다양한 은폐 및 회피 기술을 이용하여 악성코드를 고도화시켜왔다 [4]. 이런 고도화된 은폐 및 회피 기술을 이용하는 악성코드의 경우 전통적인 분석 기술 [5]로는 악성코드 분류에 필요한 충분한 특성을 추출하기가 어렵다. 최근 악성코드 분류를 위한 머신러닝이나 딥러닝과 같은 인공지능 기술들도 기존의 전통적인 분석 기술을 통해서 추출한 특성에 의존하기 때문에 고도화된 은폐 및 회피 기술을 이용하는 악성코드는 분류하기가 어렵다. 이런 어려움을 극복하기 위해서는 기존의 전통적인 악성코드 분석 기술을 계속 고도화시키는 연구도 필요하지만, 분류에 이용할 수 있는 새로운 특성과 이 특성을 이용한 분류 기술을 연구하는 것도 필요하다.

본 논문에서는 새로운 특성과 분류 기술 연구의 하나로 실행 프로그램의 성능적 특징 분석 및 튜닝 등에 활용되는 CPU, 메모리, I/O의 성능적 메트릭과 실행 개체의 행위적 특징을 나타내는 메트릭을 시계열 시각화시키고, 그에 따른 분류 성능 평가를 통해 본 논문에서 제안하는 실행 개체의 행위-성능적 시각화 기법의 악성코드 분류 활용 가능성을 고찰해 보고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 설명하고, 3장에서는 시스템 자원의 사용패턴을 이용한 실행 개체의 행위-성능적 시각화 기법을 제시한다. 4장에서는 데이터 추출을 위한 악성코드 샘플

* 정회원
† 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2019-0-00426, 50%) 및 한국연구재단(No.NRF-2020R1A2C4002737, 50%)의 지원을 받아 수행된 연구임.

셋 구성과 추출할 데이터, 그리고 데이터를 추출하는 방법을 설명한다. 5장에서는 추출한 데이터를 이용한 시각화 방법을 설명하고, 6장에서는 시각화한 이미지를 대상으로 딥러닝 기술을 이용하여 악성코드 분류 성능을 측정한다. 마지막으로 7장에서는 결론과 향후 연구 방향에 대해서 논의한다.

2. 관련 연구

CPU나 메모리와 같은 시스템 자원의 사용패턴을 이용해서 특정한 종류의 악성코드를 탐지하기 위한 연구 [6]에서는 drive-by download [7]를 통해서 유포되는 악성코드 실행 전과 실행 후의 시스템 전체 CPU와 메모리 사용패턴을 머신러닝 기술인 OC-SVM (One Class Support Vector Machine) [8, 9]을 이용하여 탐지하는 방법을 제안했다. 그림 1은 특정 시스템 환경에서 악성코드 실행 전과 실행 후의 CPU와 메모리 사용패턴 차이를 보여준다. 이 연구는 시스템 전체의 CPU와 메모리 사용패턴을 이용하기 때문에 같은 시스템 환경에서도 시스템과 다른 프로그램들의 상태에 영향을 받아서 매번 수집되는 데이터에 상당히 많은 차이가 발생할 수 있고, 핵심적인 특징인 데이터의 변화가 악성코드에 의해서만 발생한다는 것을 보장하지 못하는 한계점에도 불구하고 CPU와 메모리라는 시스템 자원의 사용패턴을 악성코드 탐지에 이용했다는 점에서 의미가 있다.

3. 실행 개체의 시각화를 통한 악성코드 분류 기법 개요

본 논문에서 악성코드를 분류하기 위한 목적으로 시스템 자원 사용패턴과 연관된 메트릭을 활용하려고 하는 가장 큰 이유는 악성코드도 프로그램이기 때문이다. 프로그램은 개발에 사용하는 라이브러리나 코드 작성 방법, 그리고 특정한 기능을 수행하기 위한 코드의 흐름 등이 다르다. 프로그램의 이런 특징으로 인해서 실행 시에 CPU나 메모리, 디스크와 같은 시스템 자원의 사용패턴에 차이가 발생하게 되고 이런 차이는 다른 프로그램과 구별되는 고유한 특징이 될 수 있다. 이렇게 시스템 자원의 사용패턴에 나타나는 악성코드의 고유한 특징을 시각화하고, 시각화한 이미지를 이용한 분류 성능 측정을 통해서 시스템 자원 사용패턴의 가능성을 제시한다.

시스템 자원의 사용패턴을 이용하여 악성코드를 분류하기 위한 시스템의 동작흐름은 그림 2와 같다. 악성

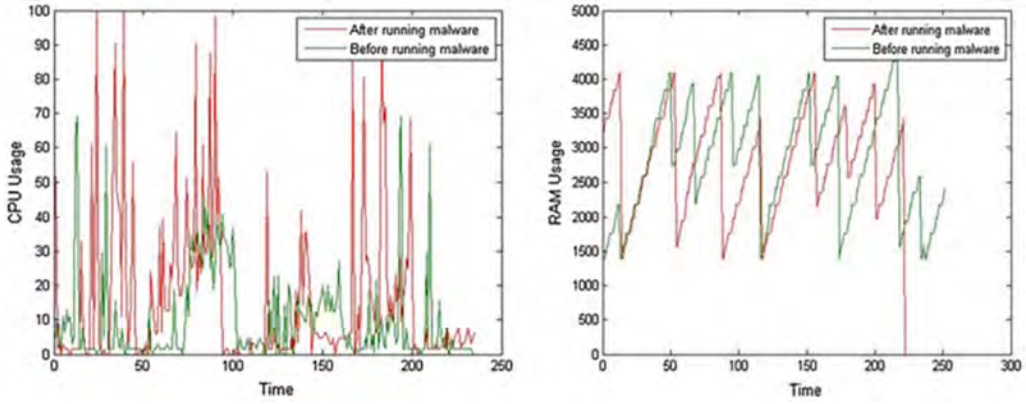

그림 1 특정 시스템 환경에서의 CPU와 RAM 사용량 차이 [4]

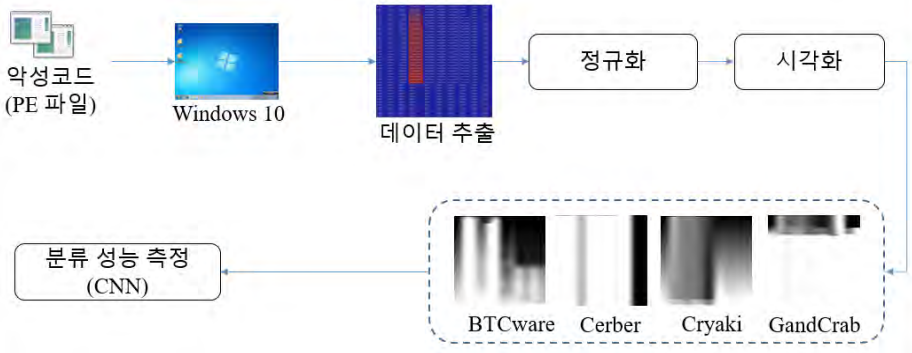

그림 2 시스템 자원 사용패턴 및 행위적 특징의 시각화를 통한 악성코드 분류 개요

코드를 실행시키고 실행된 악성코드 프로세스로부터 시스템 자원 사용량을 시계열로 추출한 다음 정규화 과정을 거쳐서 시각화한다. 이렇게 시스템 자원 사용 패턴을 시각화한 이미지를 대상으로 딥러닝 기술인 CNN [10]을 이용하여 분류 성능을 측정한다.

4. 실행 개체 선정 및 메트릭 추출

본 장에서는 데이터 추출을 위한 악성코드 샘플 셋 구성과 추출 메트릭 선정, 그리고 데이터 추출 방법을 설명한다.

4.1 악성코드 샘플 셋 구성

악성코드 샘플 셋은 2018년 ~ 2020년 사이에 VirusTotal [11]에 등록된 샘플 중에서 수집이 가능한 랜섬웨어 28종을 275개를 수집했다. 랜섬웨어 샘플로만 수집한 이유는 최근 가장 피해를 많이 주고 있는 악성코드 이면서 파일을 암호화하는 유사한 기능이 있고 다양한 종류가 있기 때문에 분류 가능성을 실험하기에 적합했기 때문이다. 악성코드 샘플 셋 구성은 표 1과 같다.

표 1 28종의 랜섬웨어 샘플

No	Class	Family	Count
1	Ransomware	Amnesia	10
2	Ransomware	BTCware	10
3	Ransomware	Cerber	10
4	Ransomware	Cryaki	10
5	Ransomware	Crypton	5
6	Ransomware	Crysis	10
7	Ransomware	Filecoder	10
8	Ransomware	GandCrab	10
9	Ransomware	GlobeImposter	10
10	Ransomware	Gryphon	10
11	Ransomware	Hermes	10
12	Ransomware	Iron	10
13	Ransomware	Kangaroo	10
14	Ransomware	LockerGoga	10
15	Ransomware	Matrix	10
16	Ransomware	Mole	10
17	Ransomware	MyRansom	10
18	Ransomware	Radamant	10
19	Ransomware	Rapid	10
20	Ransomware	RotorCrypt	10
21	Ransomware	Ryuk	10
22	Ransomware	Scarab	10
23	Ransomware	Screenlocker	10
24	Ransomware	Shade	10
25	Ransomware	Tescrypt	10
26	Ransomware	WannaCryptor	10
27	Ransomware	Xorist	10
28	Ransomware	Xrat	10

4.2 추출 데이터 선정

본 연구에서 실행 개체의 행위-성능적 시각화를 위해 수집하는 메트릭은 다음과 같다.

- CPU
  - ① CPU 클럭 수 (Cycle time)
- 메모리 사용량
  - ② 페이징 풀 (Paged pool)
  - ③ 비페이징 풀 (NonPaged pool)
  - ④ 개인 작업 집합 (Private)
  - ⑤ 작업 집합 (WorkingSet)
- 핸들
  - ⑥ 핸들 수 (Handle count)
- I/O
  - ⑦ 기타 작업 횟수 (Other count)
  - ⑧ 기타 작업의 바이트 수 (Other bytes)
  - ⑨ 읽기 작업 횟수 (Read count)
  - ⑩ 읽기 작업의 바이트 수 (Read bytes)
  - ⑪ 쓰기 작업 횟수 (Write count)
  - ⑫ 쓰기 작업의 바이트 수 (Write bytes)

CPU 클럭 수는 해당 프로세스가 동작하는 과정에서 사용한 CPU 클럭 수를 의미한다. 메모리 사용량은 윈도우 시스템이 시스템에 필요한 메모리를 관리하기 위해서 사용하는 페이징 풀과 비페이징 풀, 공유 라이브러리를 제외한 프로그램에 할당된 개인 작업 집합, 작업관리자에서 보이는 메모리 사용량인 작업 집합을 선정했다. 핸들 수는 프로세스가 할당받은 핸들(윈도우 운영체제에서 리소스를 관리하기 위해서 사용) 수를 의미한다. 그리고 I/O와 관련해서는 읽기 및 쓰기, 기타 작업 횟수와 바이트 수를 선정했다. 이렇게 선정된 12개의 메트릭은 프로그램 코드의 실행 및 시스템 자원 사용과 관련된 특징을 나타낸다.

4.3 데이터 추출 시스템 구축

본 논문에서는 실제 사용자들의 환경과 유사한 환경에서 최소한의 시스템 구성을 통해서 데이터를 수집하기 위해서 가상머신이나 Cuckoo sandbox [12]와 같은 전문적인 샌드박스를 이용하지 않고 실제 시스템을 이용했다. 데이터 추출을 위한 시스템은 표 2와 같다.

데이터 추출을 위한 프로그램들은 파이썬으로 작성하여 윈도우용 실행 파일로 만들었고, 연속적인 데이터

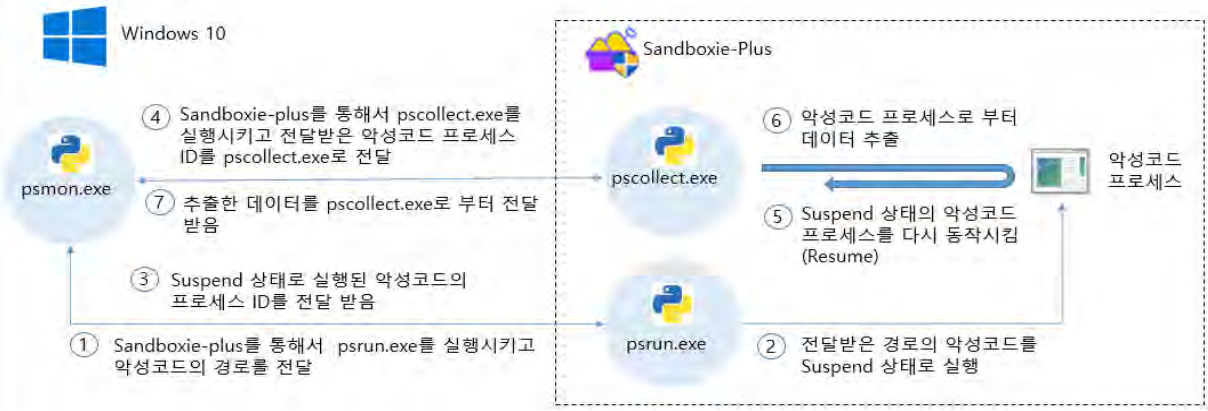

그림 3 데이터 추출을 위한 프로그램 구성과 동작 방식

표 2 데이터 추출 시스템

CPU	Intel Core i5-4690 CPU 3.50GHz
RAM	16GB
OS	Microsoft Windows 10 64-Bit (OS build 19042.685)

추출과 악성코드가 데이터 추출을 위한 프로그램에 영향을 주지 못하도록 윈도우에서 프로그램을 격리 상태로 동작하게 만들어 주는 오픈소스 프로그램인 Sandboxie-plus 0.8.8 [13]를 이용해서 추출 환경을 구성했다. 데이터 추출을 위한 프로그램 구성과 동작 방식은 그림 3과 같다. 악성코드를 실행시키는 psrun.exe와 데이터를 수집하는 pscollect.exe를 분리한 이유는 악성코드 중에서 자신을 실행시킨 상위 프로세스를 강제 종료시켜 버리는 경우가 있기 때문이었다. 그리고 시스템 자원 사용량은 실시간으로 계속 변화하는 데이터이기 때문에 모든 데이터를 다 수집할 수는 없고 0.001초 간격으로 최대 10,000번 추출했다. 데이터 수집은 파이썬 프로그램에서 관련 윈도우 API를 직접 호출했다.

5. 실행 개체 행위-성능적 시각화

이 장에서는 추출한 데이터를 이용해서 시각화하는 방법을 설명한다.

5.1 데이터 정규화

추출한 원시 데이터를 살펴보면 표 3과 같이 각 데이터 사이에 스케일의 차이가 매우 크다. 스케일의 차이가 큰 상태로는 시각화가 제대로 될 수 없기 때문에 정규화 과정을 거쳐야 한다. 데이터를 정규화하는 방법에는 여러 가지가 있지만 본 논문에서는 일반적으로 스케일의 차이를 조정하는 용도로 많이 사용하는 최소-최대(Min-Max) 정규화를 이용했다. 최소-최대(Min-Max) 정규화는 값들을 0에서 1 사이로 만들어 주는 정규화 방법이다. 그리고 정규화는 각각의 메트릭별로 진행했다.

표 3 추출한 원시 데이터의 값

메트릭	값
CPU 클럭 수	4206985569
페이징 풀	227784
비페이징 풀	17120
개인 작업 집합	3354624
작업 집합	17096704
핸들 수	237
기타 작업 횟수	1135
기타 작업의 바이트 수	16626
읽기 작업 횟수	3
읽기 작업의 바이트 수	10624
쓰기 작업 횟수	0
쓰기 작업의 바이트 수	0

5.2 행위-성능적 시각화

이 절에서는 정규화된 데이터를 이용해서 시계열 그래프를 이용한 시각화 방법과 그레이스케일 이미지를 이용한 시각화 방법을 설명한다.

5.2.1 시계열 그래프를 이용한 시각화

추출한 데이터는 12개의 메트릭을 0.001초 간격으로 10,000번 추출했기 때문에 (10000 × 12)의 2차원 배열에 저장할 수 있다. 이 2차원 배열을 파이썬의 matplotlib 라이브러리의 pyplot 모듈에서 제공하는 plot 함수를 이용하여 시계열 그래프로 시각화하면 그림 4와 같다.

Crypton 랜섬웨어는 5개의 샘플이 수집되었는데, 그림 4에서 볼 수 있듯이 시계열 그래프가 매우 유사한 것을 확인할 수 있었다. 각 샘플 사이에 유사성이 얼마나 높은지를 정적 코드를 이용해서 살펴보았다.

그림 5에서와 같이 5개 샘플의 시작 함수의 구조는 동일했고, 함수 구성에서는 유사성이 있었지만 두 가지 형태로 분류될 수 있다. ①③⑤가 유사하였고, ②④가 유사하였다. 이 부분은 그림 6의 함수 전체의 콜그래프에서도 확인할 수 있었다.

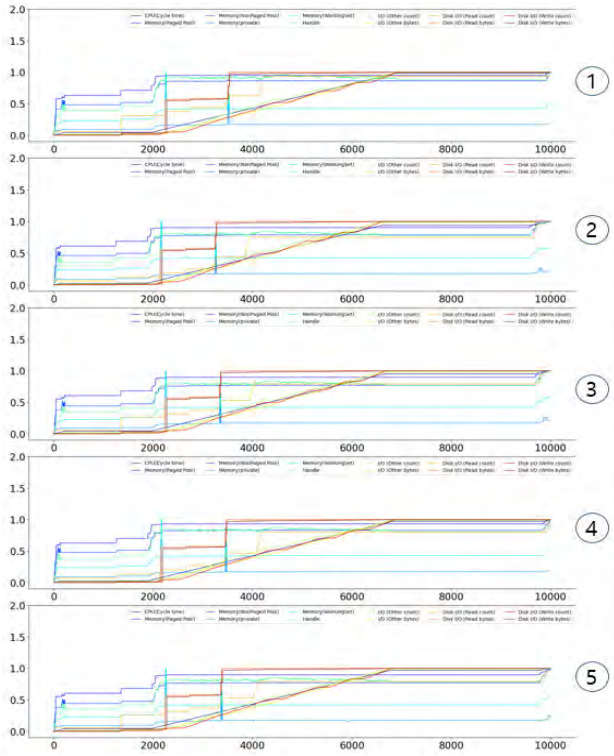

그림 4 Crypton 랜섬웨어의 시계열 그래프

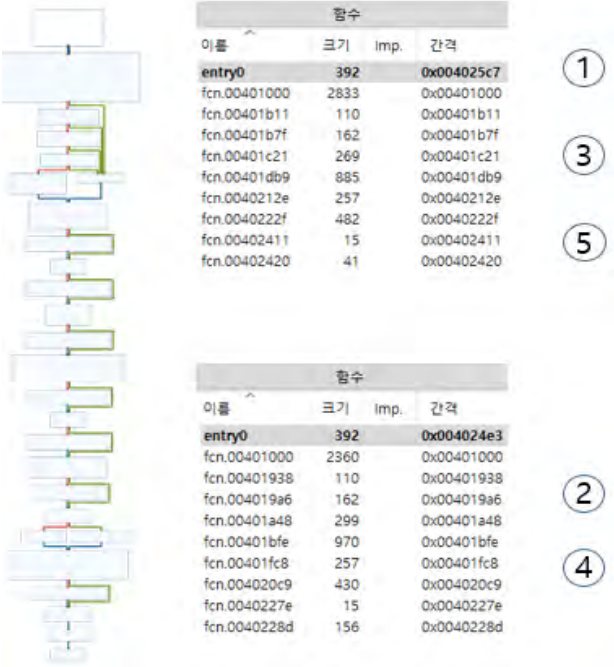

그림 5 Crypton 랜섬웨어 시작 함수의 유사성

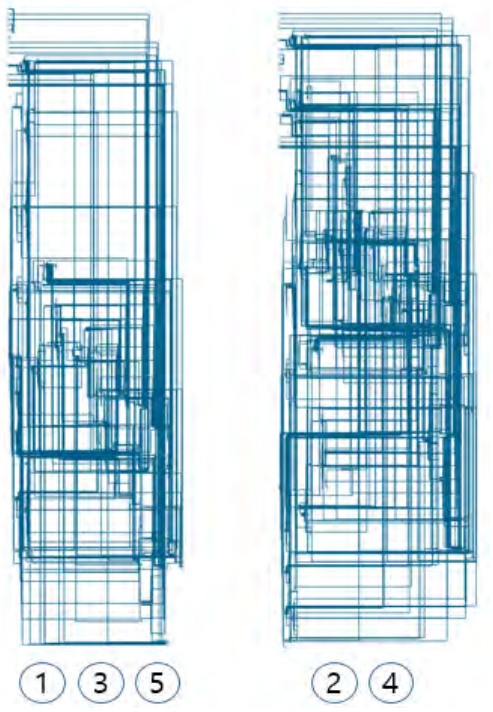

그림 6 Crypton 랜섬웨어의 함수 전체 콜그래프

위에서 살펴본 바와 같이 정적 코드의 구조와 함수 호출 구조가 같다면 시스템 자원 사용 패턴도 같거나 매우 유사하게 나타났는데, 이는 같은 코드가 실행된다면 시스템 자원 사용 패턴도 유사하다는 것을 보여준다. 물론 실행 중에 랜덤하게 기능이 동작하도록 코드가 작성되어 있다면 같은 결과가 나오지 않을 수 있는데, 이런 경우에는 반복적으로 실행시켜서 나타나는 여러 가지 유형을 모두 수집하는 것이 필요할 것이다.

5.2.2 그레이스케일 이미지를 이용한 시각화

위에서 (10000 × 12)의 2차원 배열을 시계열 그래프로 시각화하는 방법을 설명했는데, 2차원 배열은 그레이스케일 이미지로도 표현할 수 있다. 즉, (10000 × 12)의 2차원 배열은 12(넓이) × 10000(높이)의 그레이스케일 이미지로 표현할 수 있다. 본 논문에서는 파이썬의 Pillow 라이브러리의 Image 모듈에서 제공하는 fromarray 함수를 이용하여 이미지로 만든 다음 그레이스케일로 변환했다. 생성된 이미지는 세로로 매우 긴 이미지이기 때문에 시각적으로 확인하기가 어렵다. 그래서 시각적으로 확인하기 쉽도록 200 × 200 크기로 리사이징했다. 이렇게 생성된 그레이스케일 이미지는 그림 7과 같다.

시계열 그래프와 마찬가지로 샘플의 그레이스케일 이미지가 높은 유사성을 보이고 있는 것을 확인할 수 있었다. 그런데 악성코드 중에는 실행 조건이 맞지 않아서

그림 7 Crypton 랜섬웨어 그레이스케일 이미지

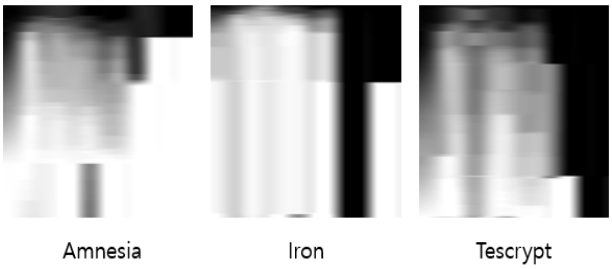

그림 8 추출한 데이터만을 이용한 시각화

바로 실행이 종료되거나 자신을 시스템의 다른 영역으로 복사한 후에 복제된 실행개체를 실행시키고 자신은 종료되는 경우 등 실행 시간이 매우 짧아서 추출할 수 있는 데이터가 적은 경우가 있는데, 이런 경우에 추출한 데이터 만을 가지고 시각화를 하게 되면 악성코드의 실행 초기 시스템 자원 사용패턴만 반영되어 그림 8과 같이 나타났다. 악성코드의 실행 초기 시스템 자원 사용패턴만으로도 다른 악성코드와 구분할 수 있는 특징들이 나타나는 것을 볼 수 있었다.

6. 분류 성능 측정

이 장에서는 시각화한 이미지를 대상으로 이미지 분류에 주로 사용되는 딥러닝 기술인 CNN [8]을 이용하여 악성코드 분류 성능을 측정한 결과를 설명한다.

6.1 악성코드 라벨 지정

CNN을 이용한 이미지 분류 성능을 측정하기 위해서는 라벨을 지정해야 하는데, 악성코드를 시각화했을 때 같은 종류의 악성코드 내에서도 그림 9와 같이 세부적으로 더 구분이 필요한 형태가 나타났다.

이렇게 세부적인 형태들을 묶어서 라벨을 지정하는 작업은 그레이스케일 이미지를 대상으로 수작업으로 진행했다. 라벨별로 최소 2개 이상의 샘플이 있는 경우에만 라벨을 지정하여, 239개 샘플에 45개의 라벨을 지정했다.

6.2 CNN 모델 구성

분류 성능 측정에 이용할 CNN 모델은 그림 10과 같이 복잡하지 않은 단순한 모델을 이용했고, 입력 이미지는 200 × 200 크기의 그레이스케일 이미지를 이용했다.

그림 9 시각화한 악성코드의 다양한 형태

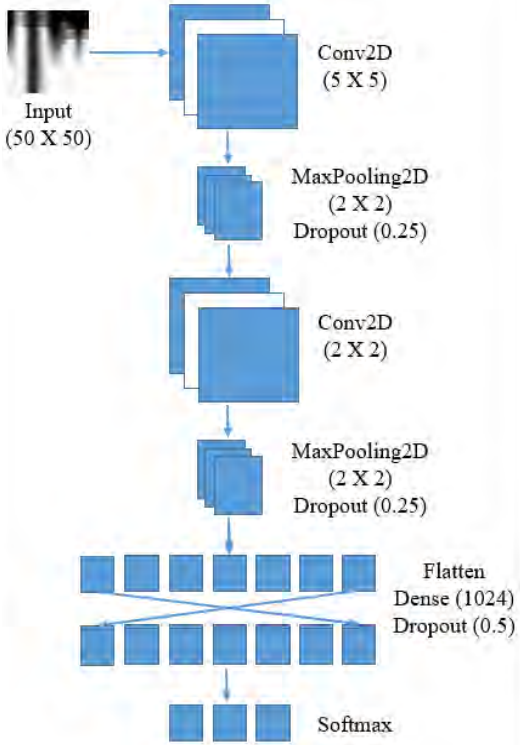

그림 10 분류 성능 측정에 이용한 CNN 모델

6.3 분류 성능 측정

본 실험에서는 새로 라벨을 지정한 그레이스케일 악성코드 이미지를 CNN 모델을 이용해서 분류 성능을 측정했다. 테스트 셋의 비율은 20%, 배치 크기는 64로 설정했고, 총 10번을 실행시켜서 측정했다. 정확도 기준으로 가장 좋은 결과에서는 정확도가 1.000000이고 손실은 0.018148였다. 가장 나쁜 결과에서는 정확도가 0.916667이고 손실은 0.229976이었다. 성능 측정 결과는 그림 11, 그림 12와 같다.

10번의 실행한 전체 결과는 표 4와 같다.

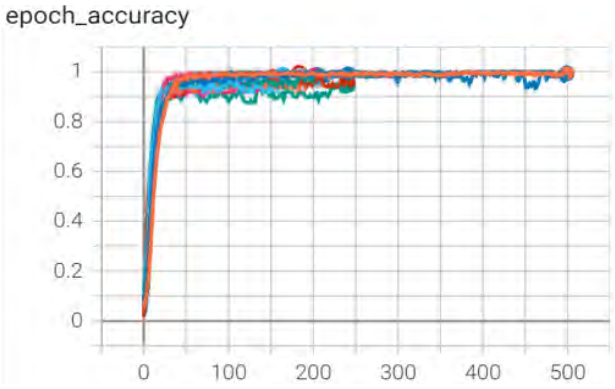

그림 11 정확도 그래프

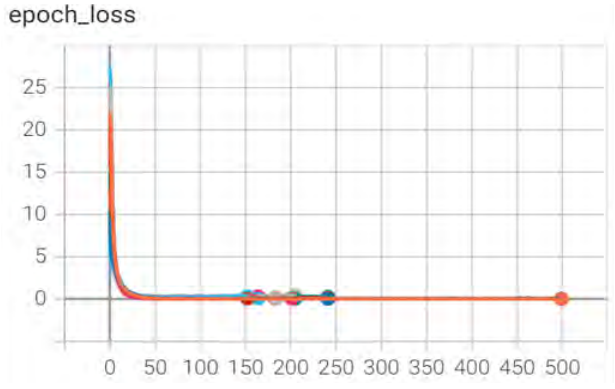

그림 12 손실 그래프

표 4 10번 실행한 전체 성능 측정 결과

횟수	정확도 (Accuracy)	손실 (Loss)	에포크 (Epoch)
1	1.000000	0.018148	500
2	0.958333	0.203253	153
3	0.979167	0.066445	202
4	0.937500	0.102053	185
5	0.958333	0.127434	242
6	0.937500	0.208053	165
7	0.937500	0.245659	206
8	1.000000	0.025026	500
9	0.958333	0.102971	184
10	0.916667	0.229976	242

7. 결론 및 고찰

악성코드를 이용한 고도화된 표적 공격에 능동적으로 대응하기 위해서는 탐지된 악성코드를 분석하고 분류해서 어떤 악성코드인지를 파악하고, 공격자가 누구인지 알아내는 것이 필요하다. 이 과정에서 가장 중요한 것이 어떤 악성코드인지를 파악하기 위한 악성코드 분류 기술이다. 지금까지 많은 분류 기술이 연구되어왔지만, 악성코드 역시 분류를 방해하기 위한 다양한 기술을 발전시켜왔기 때문에 고도화되고 있는 악성코드를 분류하기 위해서는 새로운 분류 기술과 분류에 이용할 수 있는 특성의 연구가 필요하다.

본 논문에서는 새로운 관점에서의 특성과 분류 기술 연구의 하나로 실행 프로그램의 성능적 특징 분석 및 튜닝 등에 활용되는 CPU나 메모리, I/O와 같은 시스템 자원의 사용패턴을 이용한 악성코드 분류 가능성을 제시하고자 했다. 실행 개체인 악성코드 샘플 셋을 구성하고 데이터 추출을 위한 메트릭 선정과 데이터 추출 방법, 그리고 추출된 데이터를 이용한 시각화 방법을 제시했다. 마지막으로 이렇게 시각화한 이미지를 대상으로 딥러닝 기술인 CNN을 통해서 분류 성능을 측정하였고, 상당히 좋은 결과를 얻었다. 시스템 자원의 사용패턴이 일종의 동적 분석을 통한 특성 추출이기 때문에 동적 분석이 가지는 한계점이 있고, 실험에 이용한 악성코드 샘플 셋의 크기가 작다는 점을 고려하더라도 악성코드 분류에 이용할 수 있는 충분한 가능성을 보여주는 유의미한 결과로 볼 수 있을 것이다. 또한 시스템 자원의 사용패턴은 시각화를 통해서 CNN과 같은 딥러닝 기술을 이용한 악성코드 분류에도 충분히 이용할 수 있다는 점도 보여주었다.

향후 계획으로는 시스템 자원 사용패턴을 이용한 악성코드 분류와 관련된 다양한 연구를 진행하고, 좀 더 다양한 악성코드 샘플을 대상으로 한 검증 작업을 수행할 계획이다.

참고문헌

[1] Soltani, Somayeh, et al. “A survey on real world botnets and detection mechanisms.” International Journal of Information and Network Security 3.2 (2014): 116.

[2] Sigler, Karl. “Crypto-jacking: how cyber-criminals are exploiting the crypto-currency boom.” Computer Fraud & Security 2018.9 (2018): 12-14.

[3] Magno, L., Erika, M., & Ryan, M. (2021, February 3). “The State of Ransomware: 2020’s Catch-22.” Trendmicro. https://www.trendmicro.com/vinfo/us/

security/news/cybercrime-and-digital-threats/the-state-of-ransomware-2020-s-catch-22
[4] Singh, Jagsir, and Jaswinder Singh. “Challenge of malware analysis: malware obfuscation techniques.” International Journal of Information Security Science 7.3 (2018): 100-110.
[5] Sihwail, Rami, Khairuddin Omar, and Khairul Akram Zainol Ariffin. “A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis.” International Journal on Advanced Science, Engineering and Information Technology 8.4-2 (2018): 1662.
[6] Poornachandran, Prabaharan, et al. “Drive-by-download malware detection in hosts by analyzing system resource utilization using one class support vector machines.” Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Springer, Singapore, 2017.
[7] Drive-by Download. (2021, June 29). Wikipedia. https://en.wikipedia.org/wiki/Drive-by_download
[8] Support-Vector Machine. (2021, July 12). Wikipedia. https://en.wikipedia.org/wiki/Support-vector_ machine
[9] One-Class Classification. (2021, May 7). Wikipedia. https://en.wikipedia.org/wiki/One-class_classification
[10] Convolutional Neural Network. (2021, July 72). Wikipedia. https://en.wikipedia.org/wiki/Convolutional_neural_network
[11] Virustotal, https://www.virustotal.com/
[12] Cuckoo Sandbox, https://cuckoosandbox.org/
[13] Sandboxie-Plus, https://github.com/sandboxie-plus/Sandboxie

약 력

김 준 섭

1997 충북대학교 컴퓨터공학과 졸업(학사)
2020~현재 세종대학교 정보보호학과 석사과정
2008~현재 ㈜이스트시큐리티 부사장
관심분야 : 시스템 보안, 엔드포인트 보안, 클라우드 보안, 악성코드 분석
Email : junseob9010@gmail.com

박 기 웅

2005 연세대학교 Computer Science 학사
2007 KAIST Electrical Engineering 석사
2012 KAIST Electrical Engineering 박사
2008~2009 Microsoft Research, Network Research Group, Graduate Research Fellow
2012 국가보안기술연구소 연구원
2012~2016 대전대학교 정보보안학과 교수
2016~현재 세종대학교 정보보호학과 교수
관심분야: 시스템보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식
Email: woongbak@sejong.ac.kr