딥 러닝을 이용한 시드기반 악성 트래픽 연속적 그룹핑

백의준, 박지태, Huru Hasanova, 김명섭
고려대학교
{pb1069, pjj5846, hhuru, tmskim}@korea.ac.kr

Using Deep-Learning, Seed Based Sequential Grouping of Malicious Traffic

Ui-Jun Baek, Jee-Tae Park, Huru Hasanova, Myung-Sup Kim
Korea Univ.

요 약

오늘날의 네트워크가 급속하게 성장하고 네트워크 기능이 다양한 응용 및 서비스 개발에서 중요한 요소로 자리잡음과 더불어 네트워크 환경을 위협하는 다양한 악성 트래픽이 발생하고 있다. 이러한 악성 트래픽은 네트워크 환경에 막대한 피해를 입히므로 악성 트래픽 탐지 연구는 네트워크 관리 분야에서 필수 불가결하다. 이에 본 논문에서는 딥-러닝을 이용한 시드기반 연속적 그룹핑을 통해 악성 트래픽을 탐지하는 방법을 제안한다. 또한 다양한 악성 트래픽 실험 결과를 통하여 기존의 악성 트래픽 탐지 모델과 비교하고 평가한다.

Ⅰ. 서 론

오늘날의 네트워크가 급속하게 성장하고 네트워크 기능이 다양한 응용 및 서비스 개발에서 중요한 요소로 자리잡고 있다. 동시에 네트워크 환경을 위협하는 다양한 악성 트래픽이 발생하고 있다. 이러한 악성 트래픽은 네트워크 환경에 막대한 피해를 입히므로 악성 트래픽 탐지 연구는 필수 불가결하다. 기존에 악성 트래픽 탐지에 대한 연구가 진행되었으나 정확한 탐지에 있어 한계점을 지닌다. [1] 에서는 악성 트래픽을 Flow 형태로 변환하고 이 중 한 플로우에서 5-tuple(Source IP, Destination IP, Source Port, Destination Port, Protocol)과 Flow의 발생 시간인 Start time을 추출하고 이를 Seed 라고 정의한다. 이 Seed의 정보와 주어진 트래픽 데이터 셋의 다른 Flow 간 정보를 통해 유사성 및 연결성을 계산하고 계산된 값이 일정 임계값을 넘으면 그룹핑하고 이를 연속적으로 반복하여 악성 트래픽을 탐지한다. 하지만 가중치의 조절이 정교하지 못하여 악성 트래픽 Flow가 가진 다양한 특성을 모두 반영하기 힘들고 가중치를 조절하는 과정이 Brute-Force하여 큰 시간-복잡도를 가진다. [2]에서는 딥-러닝을 이용한 연결성 계산하는 방법을 제안하였고 기존의 탐지 모델과의 실험 결과 비교를 통해 Seed기반 악성 트래픽 탐지에 딥-러닝 방법을 적용하는 것이 적절함을 증명하였다. 이에 본 논문에서는 보다 높은 정확도 및 악성 트래픽 탐지 Coverage 개선을 위하여 딥-러닝을 이용한 시드 기반 연속적 그룹핑을 통해 악성 트래픽을 탐지하는 방법을 제안한다.

본 논문은 1장 서론에 이어, 2장 본론에서 수집한 데이터의 구조와 데이터 추출 및 전처리 과정, 탐지 모델의 구조, 탐지 모델을 학습하고 테스트하는 과정에 대해 서술한다. 마지막 4장에서 결론 및 향후 연구에 대해 서술하고 본 논문을 마친다.

이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원(No.2015R1D1A3A01018057) 및 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원(No. 2017-0-00513, Security Analytics 기반의 이기종 보안솔루션 위협 분석 및 대응 기술 개발)을 받아 수행된 연구임.

Ⅱ. 본론

본 장에서는 수집한 트래픽으로부터 데이터를 추출하고 학습을 위해 데이터의 구조를 변경하는 과정, 딥-러닝 기반 연결성 계산 모델의 구조, 이 모델을 학습하는 과정 그리고 학습된 모델을 바탕으로 연속적 그룹핑 하는 과정에 대해 서술한다.

실험 데이터는 악성 트래픽과 정상 트래픽으로 구성되며 악성 트래픽은 [3]로부터 수집하고 정상 트래픽은 Wireshark을 이용하여 다양한 응용 및 서비스로부터 수집한다. 수집한 트래픽을 Flow의 형태로 변환하고 IP(Source,Destination), Port(Source, Destination), Protocol, Start Time of Flow 정보를 추출한다. 추출한 데이터는 정상 및 악성 여부에 따라 0또는 1로 라벨링되며 구조는 그림 1과 같다.

Flow Data Structure

Start time	Source IP [0:4]	Destination IP [0:4]	Src_Port	Dst_Port	Protocol	Label
Example of Data						
0.1278 (sec)	163 152 219 202	168 126 63 1	3656	80	6	1

그림 1. 플로우 데이터 구조

추출한 정상 및 악성 트래픽의 Flow 데이터를 통해 Seed Flow를 포함한 Source Flow와 Target Flow를 구성하며 이는 수식 1과 같다.

$$(1)\quad \begin{aligned} SourceFlow &= \{x | x \in Malicious\ Traffc\} \\ TargetFlow &= \{x, y | x \in Malicious\ Traffic, y \in Normal\ Traffic\} \end{aligned}$$

Source Flow와 Target Flow의 연결성 계산을 위해 두 Flow를 병합하고 시드 기반 연결성 계산의 개념을 구현한다. 또한 Source Flow와 Target Flow를 통해 병합된 Flow를 생성할 때 Source Flow와 Target Flow의 라벨에 따라 새롭게 라벨링하며 병합된 데이터의 구조와 라벨링 과정은 그림 2와 같다. 병합된 Flow가 [악성, 정상] 형태이면 0으로 라벨링하며 이는 악성 트래픽 Flow로부터 정상 트래픽 Flow는 검출되지 않도

록 학습시킨다는 의미를 가진다. 또한 병합된 Flow가 [악성, 악성] 형태라면 1로 라벨링하며 이는 악성 트래픽 Flow인 Source Flow로부터 악성 트래픽 Flow인 Target Flow를 검출하도록 학습시킨다는 의미를 가진다.

Example of Labeling

Source Flow (Malicious)	1	Target Flow (Normal)	0	0
Source Flow (Malicious)	1	Target Flow (Malicious)	1	1

그림 2. 병합 Flow 데이터의 구조 및 라벨링 예시

학습은 Multi-Layer Logistic Regression [4]을 사용하였으며 Input Layer, 3개의 Hidden Layer, Output Layer로 구성되고 활성 함수로는 Sigmoid 함수, Cost 최소화 알고리즘으로는 경사하강법을 사용한다. Learning Rate은 0.1로 설정하였으며 학습 횟수는 모든 트레이스에 대해 동일하게 10000번 진행하였다. 모델의 간략한 구조는 그림 3과 같다.

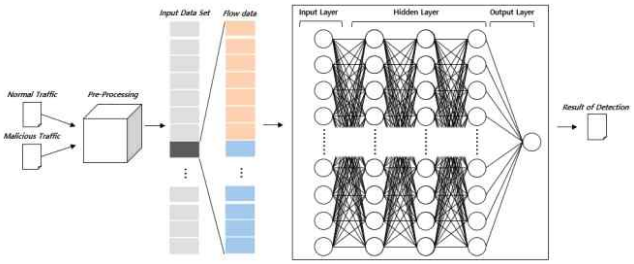

그림3. 학습 모델의 구조 및 학습 과정

학습 데이터 셋을 정해진 횟수만큼 반복적으로 학습하고 악성 트래픽 탐지에 적합하게 가중치를 조절한다. 모델의 학습 과정은 그림 4와 같다.

모델을 학습을 하는 과정을 마친 후 테스트 데이터 셋과 학습된 모델을 통해 악성 트래픽을 연속적으로 그룹핑하여 탐지한다. 첫 번째 그룹핑 단계에서는 Seed(Source Flow)와 병합된 Flow 데이터(Target Flows)를 모델에 입력하고 예측된 라벨 값에 따라 Attack Group과 Normal Group으로 나눈다. 이후 Attack Group과 Normal Group을 Source Flows와 Target Flows로 정의하고 병합하여 두 번째 입력 데이터 셋을 만들고 동일한 모델에 입력으로 넣는다. 이때 Attack Group의 다수의 Source Flow들이 Normal Group의 Target Flow들에 대해 다른 예측 값을 나타낼 수 있으므로 과반 수 이상의 Source Flows가 동일한 Target Flow에 대하여 Attack 이라 예측하였을 때 이 Target Flow를 Attack Group에 포함시킨다. 반대로 과반 수 이하의 Source Flows가 동일한 Target Flow에 대하여 Attack이라고 예측하였을 땐 이 Target Flow는 Attack Group에 포함시키지 않는다. 이 과정을 두 Group 간 Flow 이동이 없을 때까지 연속적으로 반복한다. 이 과정은 그림 4에 간략하게 나타나있다.

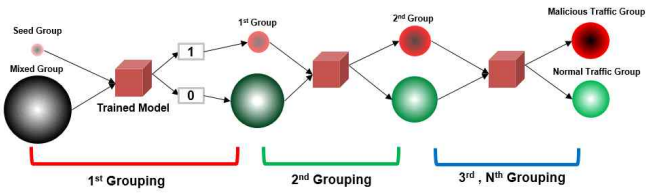

그림 4. 학습된 모델을 통한 시드 기반 연속적 그룹핑 과정

두 Group간 이동이 없을 때 Attack Group의 예측된 라벨 값과 실제 라벨 값을 대조하여 탐지 정확도를 계산한다. 탐지 정확도는 Recall, Precision으로 표현하며 계산은 수식 2와 같다.

$$(2)\quad Recall = \frac{detected\ TRUE}{total\ number\ of\ existing\ TRUE}$$

$$Precision = \frac{TRUE\ detections}{whole\ detections\ of\ an\ algorithm}$$

Ⅲ. 실험결과

본 장에서는 앞서 생성한 데이터 셋과 학습된 모델을 통해 기존 모델과의 탐지 정확도 비교 결과를 표로 보이고 분석한다. 실험 결과는 표1의 기존 모델 및 딥-러닝 모델의 학습 과정의 정확도로 구성되고 표2의 기존 모델 및 딥-러닝 모델의 테스트 과정의 정확도로 구성된다. Recall과 Precision은 각 트레이스 별 모든 시드에 대한 탐지 결과의 평균으로 계산하였다.

	기존 모델		딥-러닝	
	Recall	Precision	Recall	Precision
T1	0.3535	1.00	1.00	1.00
T2	0.2193	1.00	1.00	1.00
T3	0.4866	1.00	1.00	1.00
T4	0.538	1.00	1.00	1.00

표 1. 기존 및 딥-러닝 모델 간 학습 정확도 비교

	기존 모델		딥-러닝	
	Recall	Precision	Recall	Precision
T1	0.1202	1.00	0.9728	1.00
T2	0.1097	1.00	0.8732	1.00
T3	0.3406	1.00	1.0000	1.00
T4	0.2152	1.00	0.8333	1.00

표 2. 기존 및 딥-러닝 모델 간 테스트 정확도 비교

실험한 결과, 딥-러닝을 이용한 모델이 기존 모델보다 학습 과정에서 높은 Recall을 나타내는 것을 확인하였고 각 트레이스의 학습한 모델로 임의의 데이터 셋에 대해 테스트한 결과에서도 기존모델에 비해 평균 4.6배 높은 성능을 보이는 것을 확인하였다.

Ⅳ. 결론

본 논문은 딥-러닝을 이용하여 시드 기반 악성 트래픽을 연속적으로 그룹핑 하는 방법을 제안하고 실험하였다. 실험 결과를 통해 기존 모델의 탐지 방법보다 탐지 Coverage에서 좋은 결과를 나타내는 것을 확인하였으며 이를 통해 딥-러닝을 이용한 연속적 그룹핑 모델이 기존 모델보다 악성트래픽 탐지에 적합함을 증명하였다. 그러나 실제로 악성트래픽의 공격 타입은 다양하며 같은 공격타입의 악성트래픽이어도 다른 공격 과정을 담고 있는 경우도 있어 실험에 사용한 단순한 모델로는 정확한 악성트래픽 탐지가 힘들고 딥-러닝의 특성상 탐지 과정을 자세히 설명하기 힘들다는 한계점을 지닌다. 이에 향후 연구로 다양한 악성트래픽 탐지에 적합하도록 모델을 개선하여 탐지 성능을 고도화하고 탐지 과정의 정확한 설명을 위한 학습 및 테스트 과정의 모델 내 가중치 값 분석을 할 예정이다.

참 고 문 헌

[1] 박지태, 이성호, 구영훈, Huru Hasanova, “플로우의 연관성 모델과 시드 기반의 연속적인 그룹핑을 이용한 악성 트래픽 탐지”, 한국통신학회 학술대회논문집, pp. 608-609, 2017년 11월

[2] 백의준, 박지태, Huru Hasanova, 김명섭, "딥 러닝을 이용한 시드 기반 악성 트래픽 탐지", 2018년 통신망운용관리 학술대회 (KNOM 2018), pp.25-26, 2018년 5월

[3] “Malware-traffic-analysis.net.”. Malware-Traffic-Analysis.net. last modified Mar 28. 2018, accessed Mar 15. 2018, https://www.malware-traffic-analysis.net/index.html

[4] “GitHub.”. hunkim/DeepLearningZeroToAll. Last modified May 13. 2017, accessed Jan 11. 2018 https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-09-2-xor-nn.py