Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.3, Jun. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.3.501

파워쉘 기반 악성코드에 대한 역난독화 처리와 딥러닝 기반 탐지 방법*

정 호 진,[1†] 유 효 곤,[1] 조 규 환,[1] 이 상 근[2‡]
[1,2]고려대학교 (학생, 교수)

Deobfuscation Processing and Deep Learning-Based Detection Method for PowerShell-Based Malware*

Ho-jin Jung,[1†] Hyo-gon Ryu,[1] Kyu-whan Jo,[1] Sangkyun Lee[2‡]
[1,2]Korea University (Undergraduate student, Professor)

요 약

2021년에는 코로나의 여파로 랜섬웨어를 활용한 공격이 유행했으며 그 수는 매년 급증하고 있다. 그 중 파워쉘은 랜섬웨어에 주요 기술로 사용되고 있어 파워쉘 기반 악성코드 탐지 기법의 필요성은 증가하고 있으나 기존의 탐지 기법은 난독화가 적용된 스크립트를 탐지하지 못하거나 역난독화에 시간이 오래 소요되는 한계가 존재한다. 이에 본 논문에서는 간단하고 빠른 역난독화 처리과정, Word2Vec과 CNN(Convolutional Neural Network)으로 구성되어 스크립트의 의미를 학습하고 특징을 추출해 악성 여부를 판단할 수 있는 딥러닝 기반의 분류 모델을 제안한다. 2021 사이버보안 AI/빅데이터 활용 경진대회의 AI 기반 파워쉘 악성 스크립트 탐지 트랙에서 제공된 1400개의 악성코드와 8600개의 정상 스크립트를 이용하여 제안한 모델을 테스트한 결과 기존보다 5.04배 빠른 역난독화 실행 시간, 100%의 역난독화 성공률, 0.01의 FPR(False Positve Rate), 0.965의 TPR(True Positive Rate)로 악성코드를 빠르고 효과적으로 탐지함을 보인다.

ABSTRACT

In 2021, ransomware attacks became popular, and the number is rapidly increasing every year. Since PowerShell is used as the primary ransomware technique, the need for PowerShell-based malware detection is ever increasing. However, the existing detection techniques have limits in that they cannot detect obfuscated scripts or require a long processing time for deobfuscation. This paper proposes a simple and fast deobfuscation method and a deep learning-based classification model that can detect PowerShell-based malware. Our technique is composed of Word2Vec and a convolutional neural network to learn the meaning of a script extracting important features. We tested the proposed model using 1400 malicious codes and 8600 normal scripts provided by the AI-based PowerShell malicious script detection track of the 2021 Cybersecurity AI/Big Data Utilization Contest. Our method achieved 5.04 times faster deobfuscation than the existing methods with a perfect success rate and high detection performance with FPR of 0.01 and TPR of 0.965.

Keywords: Powershell, deobfuscation, deep learning

Received(03. 14. 2022), Modified(04. 18. 2022), Accepted(04. 19. 2022)

* 본 연구는 국방과학연구소의 연구비 지원(UD210010ED)으로 수행되었습니다.

† 주저자, froggagul@korea.ac.kr

‡ 교신저자, sangkyun@korea.ac.kr(Corresponding author)

I. 서 론

2021년에는 코로나-19 대유행에 따른 랜섬웨어(ransomware)를 활용한 위장형 공격과 원격근무 시스템을 노린 이메일, RDP(Remote Desktop Protocol), 소프트웨어 취약점에 대한 공격이 유행하였다[7]. 그중 마이크로소프트에서 개발된 객체 지향적 동적 타입 스크립트 언어인 파워쉘(powershell)은 스크립트 언어의 특성상 파일리스(fileless) 공격에 활용될 수 있어 랜섬웨어에 주요 기술로 사용되고 있다. 파워쉘 기반 악성코드의 중요성은 각 보안 업체의 악성코드 동향 보고서에서도 강조된 바 있는데[8], 특히 2020년에는 파워쉘을 사용한 악성 공격이 4분기에 689%, 올해 1902% 증가하였다[9]. 증가한 파워쉘 기반 악성코드 공격에는 'Donoff' 계열의 악성코드 등 파워쉘을 사용한 새로운 형태의 악성코드들도 다수 포함돼있으며, 공격자들은 악성코드 탐지를 회피하는 기법들을 개발하고 있다[23].

악성코드를 분석하는 방법은 크게 정적/동적 프로그램 분석이 있다. 동적 프로그램 분석의 경우, 프로그램을 직접 실행하여 해당 프로그램의 행위를 분석하므로 난독화의 영향을 적게 받지만, 프로그램을 실행시켜야 하므로 시간이 많이 소요되는 분석방법이다. 반면 정적 프로그램 분석의 경우, 프로그램을 실행시키지 않고 소스코드를 분석하는 방법이다. 그러나 파워쉘 기반 악성코드에 코드를 해석하기 어렵게 파일을 변환시키는 난독화가 적용되어있는 경우 코드의 의미를 파악할 수 없다. 이에 따라 기존의 악성코드 탐지 규칙만으로 신종 악성코드를 탐지하기에는 한계가 있고, 소스코드가 해석 가능한 수준으로 복구될 수 있는 역난독화 처리가 요구된다.

Rubin 등[6]은 난독화 된 파워쉘 기반 악성코드에 역난독화를 적용하기 위해 AMSI(Antimalware Scan Interface)[10]를 사용하는 동적 기반의 역난독화 방법을 제안했다. 그러나, AMSI의 실행 시간이 다른 역난독화 방법에 비해 추론 시간 및 정확도가 얼마나 되는지는 분석하지 않았다. 또한, Handler 등[4]은 난독화 된 파워쉘 코드의 특징을 추출하기 위해 텍스트를 글자 단위로 분석하는 전처리 방법을 제안하지만, 별도의 역난독화 처리를 적용하지 않는다.

이에 본 논문은 두 가지 기여점을 지닌다. 첫째, 스크립트 블록에 존재하는 'iex' 혹은 'invoke-expression'의 인자로 들어가는 부분을 구문분석하고 'echo' 명령어를 통해 간단하고 효율적인 역난독화 처리방법을 제안하고 해당 역난독화 처리방법의 역난독화 성공률과 속도를 평가한다. 둘째, Word2Vec과 CNN으로 스크립트의 의미를 학습해 악성 여부를 판단할 수 있는 딥러닝 기반의 분류모델을 제시하고 1600개의 악성코드와 8400개의 정상코드로 이루어진 데이터 세트를 잘 구분하는지 TPR, FPR, AUC(Area Under Curve), 스루풋(throughput)을 기준으로 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 파워쉘 기반의 악성코드를 탐지하기 위한 머신러닝(machine learning) 및 딥러닝(deep learning) 방법론에 관련된 연구를 제시하고, 3장에서 본 논문에서 제안하는 파워쉘 기반 악성코드 탐지 모델의 구성을 기술한다. 4장에서 제안한 모델의 성능을 탐지하기 위한 정량적이고 정성적인 실험들과 5장에서 연구의 결론과 향후 연구 방향을 제시한다.

II. 관련 연구

2.1 난독화 된 파워쉘에 대한 역난독화 처리기법

AMSI[10]는 Microsoft에서 개발되어 Windows 10에 내장된 악성코드 관련 인터페이스로, 파워쉘과 관련된 역난독화 기능을 제공한다. 그 외에 파워쉘 역난독화 처리를 제공하는 여러 오픈 소스(open source)들이 존재하고 개발되고 있다[13].

Liu 등[11]은 파워쉘 코드에 적용된 난독화 기법을 4가지로 정의해 각 기법마다 역난독화 처리를 제안하고 해당 기법의 역난독화 성공률과 추론 시간을 측정했다. 그러나, 이러한 방법은 정의된 난독화 기법 외의 기법에 대한 역난독화 처리를 제공하지 못하고 여러 가지 역난독화 처리 기법을 어떤 순서로 적용해야 하는지에 대한 방법을 제시하고 있지 않다.

AbdelKhalek 등[12]은 파워쉘과 유사한 스크립트 언어인 자바스크립트(Javascript)에 적용할 수 있는 함수 단위의 난독화 기법에 대한 역난독화 처리기법을 제안하고 해당 기법의 역난독화 성공률을 측정했다.

2.2 딥러닝 모델

2.2.1 CNN(Convolutional Neural Networks)

CNN은 이미지 분류[1][19] 및 자연어 처리[20][21] 등 여러 분야에서 자주 사용되는 신경망(neural network)의 한 종류이다. CNN은 Fig. 1.과 같이 여러 개의 컨볼루션 층(convolution layer)과 풀링 층(pooling layer)으로 구성되어있어[1] 데이터의 한 구역에 밀집된 공간의 정보를 공간의 위치와 무관하게 학습할 수 있다[2].

컨볼루션 층은 "FH * FW * C"의 차원을 가지는 필터와 "H * W * C"의 차원을 가지는 입력을 사용해 연산을 수행한다. 필터는 정의된 간격(stride)을 움직이며 입력과 겹치는 부분에서 연산을 수행하는데 이를 통해 출력의 차원과 값이 정해진다. 입력값이 필터를 거치고, 출력은 다시 활성화 함수의 일종인 ReLU(Rectified Linear Unit)[14]의 입력값으로 들어가게 된다.

풀링 층에서는 차원상에서 인접한 요소끼리 평균, 혹은 최댓값을 취해 하나의 값만을 남기는 연산을 진행한다. 이를 통해 데이터 내에서 필요한 정보를 보존하면서 차원을 감소시킬 수 있다.

컨볼루션 층의 필터의 크기, 간격, 채널 수, 풀링 층의 필터의 크기, 연산 방식 등은 하이퍼 파라미터(hyper parameter)로 초기 모델 설계 시 데이터의 크기 등을 고려하여 상황에 맞게 설계할 수 있다.

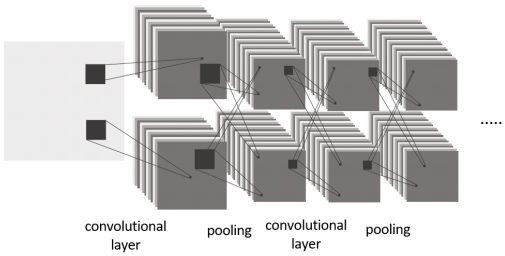

Fig. 1. Structure of CNN

2.2.2 Word2Vec

Word2Vec은 텍스트 데이터를 받아 단어의 의미를 학습하는 자연어처리 분야의 신경망 기반 모델이다. 문서 내 단어의 의미는 그 단어의 주변에 있는 단어들로부터 결정된다는 가정을 지닌다.

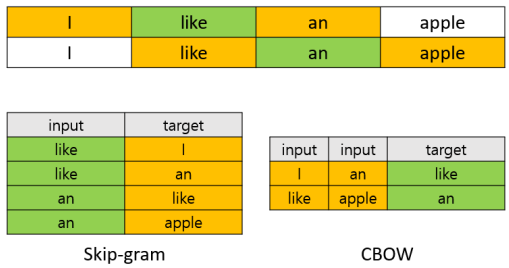

Fig. 2. Structure of Word2Vec

Word2Vec의 학습 방식은 CBOW(continuous bag of words), Skip-gram의 두 가지 방식이 존재한다. CBOW는 Fig.2.와 같이 문장에서 주변 단어로부터 중심에 있는 단어를 예측하고, Skip-gram은 반대로 중심 단어로부터 주변 단어를 예측한다[3].

이 기술을 이용하여 단어를 특정 차원의 벡터(vector)로 대응시킬 수 있으며, 특히 문서 내에서 비슷한 의미를 지니는 단어는 벡터 공간상에서 가깝게 위치하는 특징을 지니게 된다.

CBOW에서 주변 단어로부터 중앙의 단어를 예측하는 확률 P는 수식 (1)과 같이 정의된다. 여기서, u_c는 중앙의 단어를 나타내는 벡터이고, $W_o = \{w_{o_1},...,w_{o_{2m}}\}$는 주변의 단어를 나타내는 벡터 w_o의 집합이다. $2m$은 중앙의 단어를 예측할 때 참조할 단어의 개수이다.

$$P(w_c|W_o) = \frac{\exp\left(\frac{1}{2m}u_c^T(v_{o_1}+\cdots+v_{o_{2m}})\right)}{\sum_{i\in\nu}\exp\left(\frac{1}{2m}u_c^T(v_{o_1}+\cdots+v_{o_{2m}})\right)} \quad (1)$$

2.3 파워쉘 기반 악성코드에 대한 딥러닝 기반 탐지

파워쉘 기반 악성코드에 대한 딥러닝 기반 탐지에 관한 초기 연구는 난독화 된 코드에서 특징을 추출하려 시도했다. Handler 등[4]은 난독화 된 파워쉘 코드의 특징을 추출하기 위해 텍스트를 글자 단위로 분석하는 전처리 방법과 데이터에 적용할 수 있는 모델 중 CNN이 제일 좋은 성능을 냄을 보였다. Rusak 등[5]은 파워쉘을 AST(Abstract Syntax Tree) 형태로 변환시켜 각 node를 학습시키는 방식을 제안했다. 최근에는 코드에 역난독화 처리를 한 뒤 전처리가 완료된 코드에 분류기를 적용하려는 시

도가 이어지고 있다. Rubin 등[6]은 난독화 된 파워쉘 기반 악성코드에 역난독화를 적용하기 위해 AMSI[10]를 사용하는 동적 기반의 역난독화 방법을 제안했다.

III. 제안 방법

3.1 개요

본 논문에서 제안하는 딥러닝 기반 악성코드 탐지 방법은 Fig.3.과 같이 크게 3가지 부분인 역난독화, 전처리, 분류기로 구성되어있다.

역난독화 단계에서는 난독화 되어있는 파워쉘 스크립트에 역난독화를 적용하여 의미를 지니는 원래의 코드로 복구하는 기법을, 전처리에서는 분류기의 성능 향상을 위해 텍스트 데이터에 적용할 수 있는 전처리 방법을, 분류기에서는 전처리된 데이터에서 의미를 학습하고 특징을 추출해 악성 여부를 판단할 수 있는 분류기 모델을 제시한다.

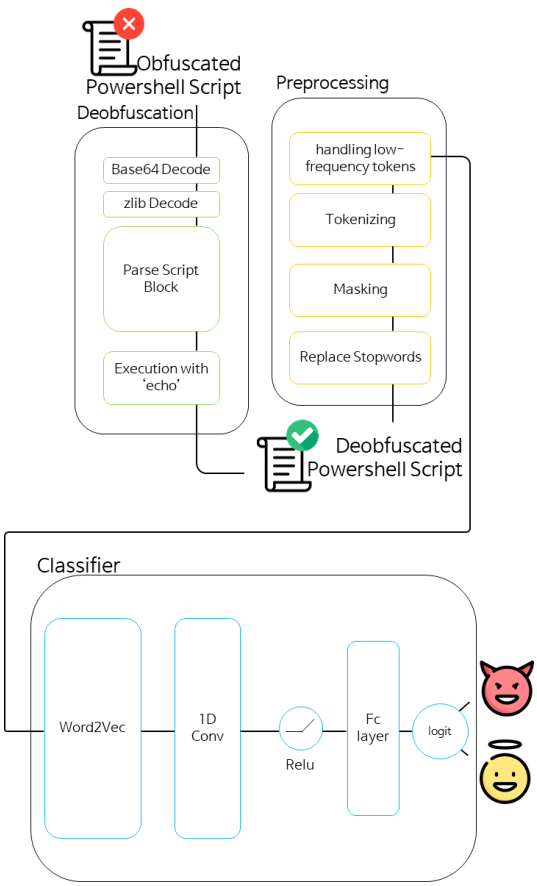

Fig. 3. Overall Structure of Malware Detection Method

3.2 역난독화

많은 난독화 파워쉘 스크립트들은 역난독화 과정을 거친 후, 'iex' 또는 'invoke-expression'을 통해서 실행하는 형식으로 실행된다. 이에 착안하여, 난독화 파워쉘 스크립트를 스크립트 블록(파워쉘 인터프리터가 스크립트를 실행하는 가장 작은 단위)별로 구문 분석하였고, echo 명령어와 함께 이를 실행하여 역난독화를 처리하였다.

이때, 난독화 방법 중 Base64와 zlib를 통하여 난독화를 진행한 것은 디코딩이 간단히 가능하기에, 따로 처리하여 파싱하는 데 걸리는 시간 줄였다.

파워쉘 스크립트를 파싱할 때는 Microsoft 공식 문서[18]의 파싱 규칙을 따라 script block을 파싱하였고, 이후 이를 다음과 같이 실행하였다.

이를 수행하게 되면 각 스크립트 블록의 역난독화 스크립트를 확인할 수 있는 데, 그 결과가 'iex' 또는 'invoke-expression'의 경우를 제외하면 역난독화 스크립트를 확인할 수 있다.

위 과정은 난독화가 적용되어있는 파워쉘 스크립트들에 대해서만 진행한다. 난독화가 여러 번 적용되어있을 수도 있기에, 파워쉘 스크립트가 난독화되어 있는지 판별을 하여, 난독화 파워쉘 스크립트가 더 난독화가 적용되지 않았을 때까지 반복한다.

Table 1. Python Style Pseudocode of Deobfuscation

```
def deobfuse(script):
  # get parsed obfused command
  # which is input of Invoke-Expression
  parsed_script = parser(script)

  # base64 check
  if base64_encoded:
    return base64_decode(parsed_script)

  # get deobfused script
  p = sp.Popen(f"powershell echo
{parsed_script}".split(' '), stdout=sp.PIPE,
stderr=sp.PIPE)
  out, err = p.communicate()
  rc = p.returncode

  return out
```

기존의 다른 역난독화 방법은 hooking을 통해서 실행되기 직전에 스크립트를 가져오거나, 실제로 파워쉘 스크립트를 실행시킨 후, 로그를 가져오는 방법을 사용하였다. 하지만 hooking을 통하는 방법은 파워쉘을 실행하는 과정에서 overhead가 발생해 역난독화에 걸리는 시간을 증가시켰고, 로그를 가져오는 방법은 실제 악성 스크립트를 모두 실행해야 했기에 너무 긴 시간이 걸렸다. 제안하는 방법은 overhead도 발생하지 않으며, 악성 스크립트를 실행하지도 않기에, 빠른 역난독화가 가능하다.

3.3 전처리

데이터에 자주 드러나지 않는 희귀한 글자 및 단어들을 모두 분석하면 오버피팅(over-fitting) 문제가 일어날 가능성이 증가하고 데이터의 차원이 필요 이상으로 증가하기 때문에 자주 사용되는 문자와 토큰(token)들만을 사용한다.

자주 사용되는 글자들은 A-Za-z*$- 이다. 영어 대소문자를 제외하고 $와 -를 포함시키는 이유는 이들이 변수 및 함수를 선언할 때 사용되는 문자들이기 때문이다.

코드 내에서 선언된 모든 숫자 부분은 문자 *로 마스킹 처리하였다. 다른 문자는 모두 공백으로 치환하였다.

그 뒤 공백을 기준으로 데이터를 분리하는 토큰화 작업을 수행했다. 이때, 스크립트에서 추출된 초기 n개의 토큰만 사용하며 k개 미만의 스크립트에서 나타나는 토큰은 이 과정 중 생략한다. 전처리된 토큰의 길이가 n보다 짧은 경우에는 제로 패딩을 적용한다. 초기 n개의 토큰만을 사용하는 이유는 이후 4.2. 데이터 분석에서 정성적으로 설명한다.

3.4 분류기

분류기는 Fig.3.과 같이 Word2Vec 기반의 임베딩 층과 CNN 기반 분류기으로 구성된다.

3.4.1 Word2Vec 기반의 임베딩 층

임베딩 층에서는 전 처리된 토큰이 각각 특정 차원의 벡터로 대응된다.

CNN 계층과 임베딩 층을 합쳐서 훈련하기 전, 각 토큰의 의미를 미리 학습시키기 위해 Word2Vec

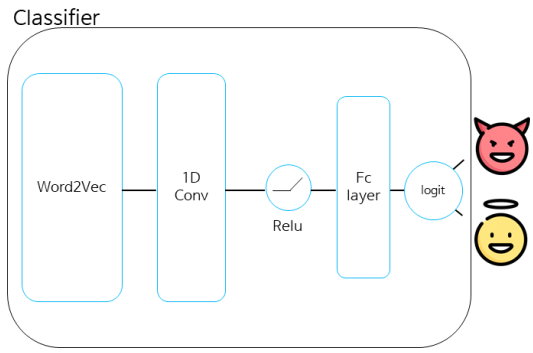

Fig. 4. Overall Structure of Classifier

으로 사전 학습(pretrain) 된[17] 임베딩을 사용한다. 훈련 방식은 CBOW[3] 방식을 사용하며, 훈련 과정에서 네거티브 샘플링(negative sampling)[16] 방식을 사용한다.

3.4.2 CNN 기반의 분류기 층

CNN 층은 1차원 컨볼루션 층 - 최댓값 풀링 층으로 구성된다. 컨볼루션 층에는 활성화 함수가 포함되며 본 논문에서는 ReLU[14]를 사용했다.

1차원 컨볼루션을 포함하는 CNN 층을 사용하는 이유는 데이터의 특성과 관련돼있다. 악성코드의 경우, 악성코드를 실행하는 부분이 텍스트 중 특정 부분에 모여있다. CNN은 특정 정보가 데이터의 특정 부분에 모여있다고 가정하는 귀납 편향(inductive bias)[2]을 지니기 때문에 해당 문제에 적합한 구조라고 판단했다.

CNN을 거친 후 나온 출력값은 다시 fully connected 층의 입력값으로 들어간다. 최종적으로 출력된 1차원 값은 특정 로짓(logit) 값을 기준으로 악성 여부를 판단한다. 기준 로짓 값은 별도의 교정(calibration) 과정을 통해 결정한다.

IV. 실험 및 평가

4.1 데이터셋

실험을 위한 데이터셋으로는 한국인터넷진흥원에서 주관하고 과학기술정보통신부가 주최하는 2021 사이버보안 AI/빅데이터 활용 경진대회의 트랙 A, AI 기반 파워쉘 악성 스크립트 탐지에서 제공된 예선 데이터셋을 사용했다[22]. 해당 데이터셋은

8600개의 정상 스크립트, 1400개의 악성 스크립트로 구성되어있다.

4.2 실험 환경 구축

본 논문에서 역난독화 및 모델 훈련을 위해 사용한 시스템의 환경은 Table 2.와 같다.

역난독화 도구는 제안한 알고리즘을 python3 에서 구현했다. 그 뒤 주어진 가상환경에서 데이터에 대한 역난독화 처리를 진행했다.

분류기의 경우 'pytorch', 'numpy', 'pandas' 라이브러리로 구현한 후 그래픽카드 RTX2080Ti에서 훈련했다.

Table 2. Experiment Environment Specification

experient tool	version
python	3.10
pytorch	1.7.1
gensim	4.0.1
VMware Workstation Player 16	16.1.1
pandas	0.25.3
numpy	1.19.5
jupyter notebook	6.0.2

4.3 데이터 분석

본 논문에서 제안한 모델의 경우, 입력의 길이가 일정해야 한다는 제약하고 있다. 그러나, 역난독화가 적용된 스크립트의 길이는 매우 다양하다.

따라서, 본 논문의 해당 부분에서는 스크립트 길이를 n으로 제안하였을 때 얻는 이점에 대한 정성적인 설명을 제시한다. 역난독화가 해제된 스크립트에 대해 TF-IDF (Term Frequency-Inverse Document Frequency) 알고리즘을 적용해 주어진 스크립트를 임베딩한 뒤, UMAP (Uniform Manifold Approximation and Projection for Dimension Reduction) 알고리즘을 적용해 차원을 축소시켜 2차원상에서 표현했다.

Fig.5.는 그 결과를 보여주고 있다. 왼쪽은 2048개의 토큰만을 사용했을 때, 오른쪽은 전체의 토큰을 사용했을 때를 보여준다. 첫 2048개의 토큰만을 사용하는 것이 더 분명한 군집을 형성하고 있음을 관찰할 수 있다. 이는 곧 분류기가 해당 데이터 분포를 더 쉽게 구분할 수 있음을 의미한다. 따라서, 본 논문에서는 앞선 분석 결과를 토대로 첫 n개의 토큰만을 사용하는 것이 모델의 성능에 오히려 긍정적인 영향을 줄 것으로 판단하여 이를 실험 디자인에 반영했다.

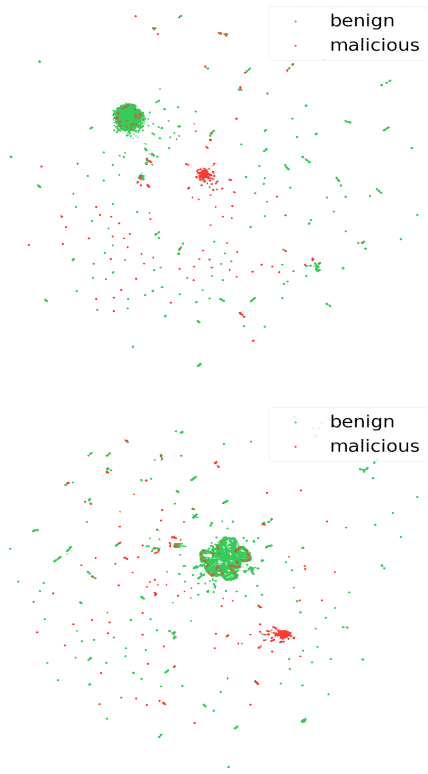

Fig. 5. Comparison of using first 2048 tokens (top) and using all tokens (bottom). Using first 2048 tokens forms better cluster.

4.4 실험 결과

4.4.1 역난독화

난독화 스크립트는 Table 3.과 같이 스크립트의 내용을 알기 어렵게 되어있다. 이를 분류기를 통해서 악성/비악성 분류를 하기 위해서는 역난독화 과정이 필요하다.

역난독화 과정에서 파워쉘 스크립트의 스크립트 블록별로 구문 분석을 진행하게 된다. Table 3.의 난독화된 스크립트의 경우 그 아래의 형태로 스크립트 블록이 구문 분석되게 된다. block[1]에 해당하

는 스크립트를 파워쉘로 실행해보게 되면, “invoke-expression” 문자열을 확인할 수 있다. 또한, block[0]에 해당하는 스크립트를 echo 명령어와 함께 실행하게 되면, 난독화가 해제된 파워쉘 스크립트를 확인할 수 있다.

주어진 데이터 10000개의 스크립트에 대해서 기존의 역난독화 방법과 본 논문에서 제안한 역난독화 방법의 성능 차이를 비교한다. 같은 환경에서 스크립트마다 역난독화 처리에 든 시간과 성공률을 기준으로 비교한다. Table 4.는 역난독화 방법별로 소요된 시간 및 역난독화 성공 여부를 표시한다. PS Decode의 경우 모든 데이터에 역난독화를 제공하는 데 실패했으며, AMSI의 경우 본 논문에서 제안한 알고리즘

Table 3. Obfuscated PowerShell Script(top) and Seperated Script Blocks(bottom)

Obfuscated PowerShell Script	(new-ObjECT SYstEm.IO.cOmprEsSIon.dEf L a t e s T R e a M ([sYsTeM.Io.MemOrysTrEAm][cONVERt]::FrOMbAsE64striNG('7Vt7c9u4Ef8/M/ … Sw16+WL+fw==′),[iO.cOmPressiOn.cOmPRESSiONmoDE]::DEcoMPREsS) | % { n e w - O b j E C T sysTEm.iO.STreamreAdEr($_ ,[TExT.enCOdIng]::AsCII) }).READTOeND( )|iNVOKE-ExpRessioN
Separated Script Blocks	file0: block[0] (new-ObjECT SYstEm.IO.cOmprEsSIon.dEf L a t e s T R e a M ([sYsTeM.Io.MemOrysTrEAm][cONVERt]::FrOMbAsE64striNG('7Vt … nSU10Omau+CRNPASw16+WL+fw==′),[iO.cOmPressiOn.cOmPRESSiONmoDE]::DEcoMPREsS) | % { n e w - O b j E C T sysTEm.iO.STreamreAdEr($_ ,[TExT.enCOdIng]::AsCII) }).READTOeND( ) block[1] iNVOKE-ExpRessioN

Table 4. Success Rate and Execution Time by Deobfuscation Method

deobfuscation method	execution time (s/script)	success rate (%)
Ours	0.648	100%
AMSI	3.267	85.2%
PSDecode	Failed	0%

에 비해 더 많은 시간이 소요됐다.

AMSI를 사용할 경우, 역난독화가 실패하는 경우도 존재하였다. AMSI는 파워쉘 스크립트를 실행하면서 검사를 진행하기에, 파워쉘 스크립트의 실행이 완료되기까지의 시간이 소요되었다. 이는 실행 도중 사용자가 입력값을 주어야 하거나, 서버와 통신을 하는 스크립트가 존재하게 되면, 역난독화가 중단되는 문제점이 발생한다. 또한, 이러한 문제점은 command line logging 방법에서도 동일하게 나타났다. 반면에서 본 논문에서 제안하는 알고리즘은 위와 같은 문제가 발생하지 않고, 모든 데이터셋에 대해서 역난독화를 AMSI와 비교했을 때 실행 시간은 약 5.04배 빠르며 성공률은 14.8% 높다.

4.4.2 분류기 학습 및 검증

분류기에 대한 학습은 RTX 2080Ti에서 이루어졌으며, 최적화 알고리즘으로는 SGD(Stochastic Gradient Descent) 계열의 ADAM[15]을 사용했으며, 목적 함수로는 이진 교차 엔트로피 함수를 사용했다. 이진 교차 엔트로피 함수는 식 (2)와 같다. 미니 배치 크기는 128이다.

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_n \log x_n + (1-y_n)\log(1-x_n)\right) \qquad (2)$$

오버피팅(overfitting)을 방지하기 위해 학습 데이터 세트의 1/3을 검증 데이터 세트로 사용한다. 검증된 데이터 세트에서 손실 그래프를 그린 뒤, 학습 중 손실 그래프가 최저점에서 올라가는 경우 미리 학습을 중지시킨다.

4.4.3 분류기 성능 평가

분류기 성능 평가 방식은 Rubin 등[6]이 제시한 방식과의 비교를 위해 같은 실험 방식을 따른다. 데

이터 세트 분포에 크게 치우치지 않고 분류기의 성능을 적은 편향으로 예측하기 위해 학습, 그리고 테스트용 데이터 세트를 분리하였다. 각 데이터 세트는 7:3으로 랜덤하게 분배되었으며, 학습용 데이터 세트에서는 다시 3-fold 교차 검증을 통해 최적의 하이퍼파라미터(hyperparameter)를 계산했다. 설정한 하이퍼파라미터는 Word2Vec의 임베딩 차원, CNN의 필터 수, 입력 시퀀스 길이 등이 있다. 3-fold 교차 검증에 사용된 3개의 모델은 다시 테스트용 데이터 세트에서 성능 평가를 진행하여 각 값을 평균 냈다.

분류기는 TPR FPR, AUC, 스루풋을 평가에 활용한다. 이러한 평가 기준을 잡은 이유는 분류기를 훈련하는 데이터 세트의 경우 악성 스크립트의 수가 정상에 비해 적기 때문이다.

FPR은 수식 (3)과 같이 실제 정상 중 악성이라 예측한 대상의 비율을 나타내는 지표이고, TPR은 수식 (4)와 같이 실제 악성 중 악성이라 예측한 대상의 비율을 나타내는 지표다. AUC는 이진 분류기의 분류 기준에 따라 달라지는 FPR과 TPR을 기준으로 그래프를 그렸을 때, 그 그래프의 아래 영역의 넓이를 의미한다. 스루풋은 한 단위 시간에 처리되는 데이터 단위의 수를 나타낸다. 본 논문에서는 1초에 처리할 수 있는 데이터 단위의 수를 측정한다.

실제 배포 환경에서 FPR의 값이 조금만 높아도 악성이라고 판정한 정상 프로그램들이 많아지기 때문에 탐지 시스템에 치명적이다. 따라서 본 논문에서는 FPR의 값이 1% 이하일 때의 TPR 최대값과 AUC를 평가 지표로 사용한다.

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

$$TPR = \frac{TP}{FN + TP} \tag{4}$$

Table 5.는 기존 연구의 모델과 비교한 결과를 나타낸다. 같은 데이터에 대해서 Handler 등[4]이 제시한 방식과 비교했을 때, 해당 논문에서 가장 좋은 성능을 보이는 문자 단위의 4-계층 CNN 모델을 기준으로 비교했을 때 테스트 데이터 세트에서 측정한 TPR 값과 AUC 값은 본 논문이 더 우수한 값을 가진다. AUC 값이 본 논문이 큰 것을 Fig.6.을 통해서 정성적으로도 파악할 수 있다. 스루풋은 Handler 등[4]이 약간 더 나은 편이나 크게 다르지 않았다.

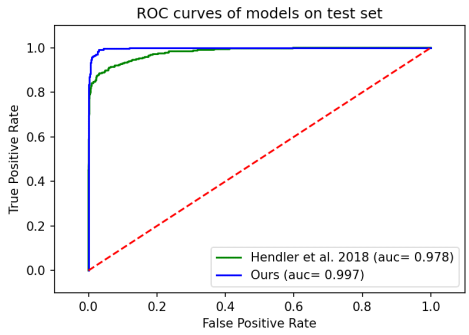

Fig. 6. ROC curves of models on test set

Table 5. Performance comparison for detection. Standard deviations are less then 0.012 on TPR, 0.0011 on AUC, and 0.0004 for FPR

model	AUC	TPR	FPR	throughput
Ours	0.997	0.965	0.008	24591.39
Handler et al. 2018 [4]	0.978	0.821	0.001	30718.35

위의 결과를 통해 본 논문에서 제안한 모델이 더 나은 일반화 능력을 지니고 있음을 알 수 있다.

V. 결 론

이에 본 논문에서는 간단하고 효율적인 역난독화 방법, 스크립트의 의미를 학습해 악성 여부를 판단할 수 있는 딥러닝 기반의 분류 모델, 그리고 데이터 분석을 통해 모델에 설명을 부여하여 기존의 악성 파워쉘 스크립트 탐지 방법론보다 악성코드를 효과적으로 탐지할 수 있는 모델을 제안했다.

역난독화 기법은 파워쉘에서 스크립트 블록을 구문분석하고, 'iex' 혹은 'invoke-expression'의 인자로 들어가는 부분을 구문분석하고 'echo' 명령어를 통해서 역난독화 스크립트를 출력한다. 딥러닝 기반의 분류 모델은 Word2Vec과 CNN을 기반으로 스크립트에 내재한 의미를 학습하여 악성 여부를 판단한다. 해당 역난독화 기법은 기존의 역난독화 기법에 비해 실행이 약 5배 빠르며 역난독화 성공률 또한 약 15% 높았다.

제안한 모델을 주어진 악성과 정상 파워쉘 스크립트로 구성된 데이터 세트에서 훈련 및 검증, 테스트

하였을 때, 테스트 데이터 세트에서 기존의 모델보다 뛰어난 성능을 보이는 데 성공했다.

본 논문은 향후 기업과 기관의 파워쉘 기반 악성코드에 대한 대응책에 도움이 될 것이며, 악성코드 분류에 있어서 딥러닝 기법을 활용하는 연구에 있어 지침이 될 것이라 기대한다.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, pp. 1106-1114, Dec. 2012

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, and Yujia Li, Razvan Pascanu, "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261v3, Oct. 2018

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1301.3781v3, Sep. 2013

[4] D. Hendler, S. Kels, and A. Rubin, "Detecting malicious powershell commands using deep neural networks," Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 187-197, Jun. 2018

[5] G. Rusak, A. Al-Dujaili, and U.M. O'Reilly, "AST-Based Deep Learning for Detecting Malicious PowerShell," Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 2276-2278, Oct. 2018

[6] D. Hendler, S. Kels, and A. Rubin, "AMSI-Based Detection of Malicious PowerShell Code Using Contextual Embeddings," Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 679 - 693, Oct. 2020

[7] Coveware, "Ransomware attackers down shift to 'Mid-Game' hunting in Q3 2021" https://www.coveware.com/blog/2021/10/20/ransomware-attacks-continue-as-pressure-mounts, Apr 01. 2022

[8] KrCERT, "2021 Ransomware Special Report," https://www.krcert.or.kr/filedownload.do?attach_file_seq=3278&attach_file_id=EpF3278.pdf, Apr 01. 2022

[9] McAfee, "McAfee Labs COVID-19 Threats Report" https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf, Apr 01. 2022

[10] Microsoft, "Antimalware scan interface" https://docs.microsoft.com/en-us/windows/desktop/AMSI/antimalware-scan-interface-portal, Apr 01. 2022

[11] C. Liu, B. Xia, M. Yu and Y. Liu, "PSDEM: A Feasible De-Obfuscation Method for Malicious PowerShell Detection," 2018 IEEE Symposium on Computers and Communications, pp. 825-831, Jun. 2018

[12] M. AbdelKhalek, and A. Shosha, "JSDES: An Automated De-Obfuscation System for Malicious JavaScript," Proceedings of the 12th International Conference on Availability, Reliability and Security, pp. 1-13, Aug. 2017

[13] GitHub, "PSDecode" https://github.com/R3MRUM/PSDecode, Apr 01. 2022

[14] V. Nair and G.E. Hinton, "Rectified li

near units improve restricted boltzmann machines," Proceedings of the 27th international conference on machine learning, pp. 807-814, Jun. 2010

[15] D.P. Kingma and J.L. Ba, "Adam: a method for stochastic optimization," arXiv preprint arXiv:1412.6980v9, Jan. 2017

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean, "Distributed Representations of Words and Phrases and their Compositionality", Advances in Neural Information Processing Systems, pp. 3111-3119, Dec. 2013

[17] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio, "Why Does Unsupervised Pre-training Help Deep Learning?," Journal of Machine Learning Research, pp. 625-660, Mar. 2010

[18] Microsoft, "about_Parsing," https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_parsing?view=powershell-7.2, Apr 01. 2022

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," IEEE Conference on Computer Vision and Pattern Recognition, pp. 1 - 9, Jun. 2015

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," Journal of Machine Learning Research, vol. 12, no. 76, pp. 2493 - 2537, Jun. 2011

[21] Y. Kim. "Convolutional neural networks for sentence classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1746 - 1751, Oct. 2014

[22] KISA, "Cyber security AI/bigdata challenge 2021" https://aibigdatasec.kr/main, Apr 01. 2022

[23] McAfee, "McAfee Labs Threats Report" https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-apr-2021.pdf, Apr 01. 2022

〈저 자 소 개〉

정 호 진 (Ho-jin Jung) 학생회원
2019년 3월~현재: 고려대학교 사이버국방학과 학사
〈관심분야〉 정보보호, 사회 관계망 분석, 설명 가능 인공지능, 연합 학습

유 효 곤 (Hyo-gon Ryu) 학생회원
2019년 3월~현재: 고려대학교 사이버국방학과 학사
〈관심분야〉 정보보호, 자연어처리, 데이터 증강

조 규 환 (Kyu-whan Jo) 학생회원
2018년 3월~2022년 2월: 고려대학교 사이버국방학과 학사
〈관심분야〉 정보보호, 신경망 구조 탐색, 모델 경량화

이 상 근 (Sangkyun Lee) 정회원
〈관심분야〉 인공지능보안, 인공지능 모델 복제 공격과 방어 기술, 설명 가능한 인공지능, 인공지능 모델 압축