난독화에 강한 AI 기반 웹쉘 탐지

부산대학교 정보컴퓨터공학부

지도교수 : 최윤호

팀 명 : 문지기
팀 원 : 201914119 문정윤
202155507 구지원
202155614 차기은

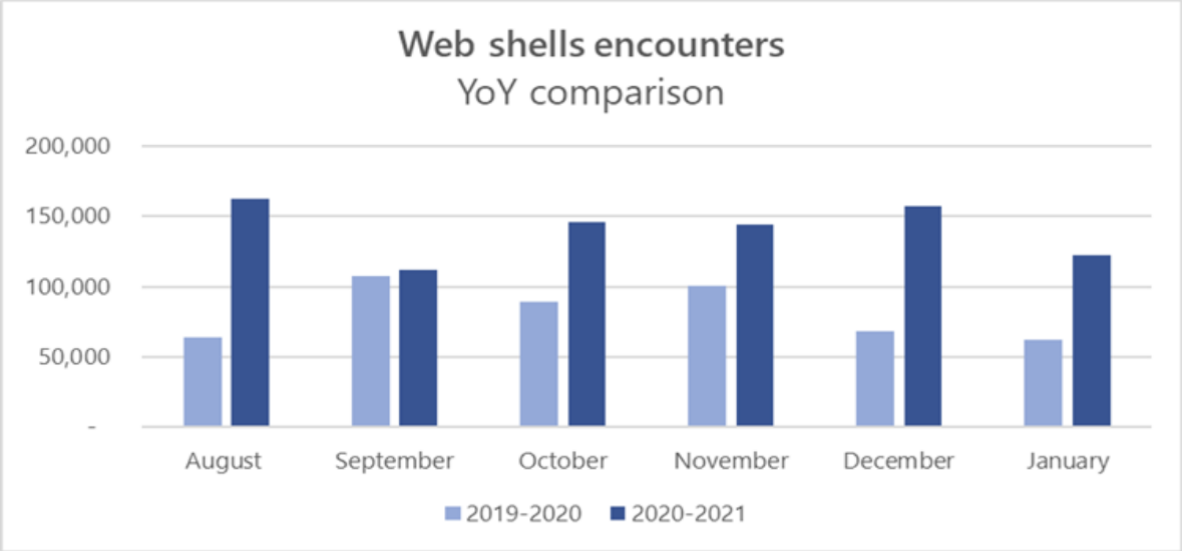
< 목차 >

1. 과제 배경 및 목표

1.1. 과제 배경

웹쉘은 웹 서버를 침투하여 추가적인 공격을 시작할 수 있는 악성 스크립트이다. 이러한 악성 스크립트 웹쉘을 이용한 공격은 지속적으로 증가하고 있다.

마이크로소프트의 Microsoft 365 Defender 에서 발표한 'Web shell attacks continue to rise'에 따르면, 2022 년과 2021 년에 발견된 웹쉘은 14 만 건으로 2019 년부터 2 년간 발생된 웹쉘 평균 개수의 7 만 7 천 건에 비해 2 배 증가한 수치이다. 이러한 추세는 웹쉘이 지속적으로 증가함을 보여준다. 또한, 그 추세를 나타내는 아래의 그래프는 웹쉘이 단순히 지속적으로 증가한다는 사실과 더불어 가속화되고 있다는 것을 보여준다.



국내의 경우, LG 유플러스에서 2009 년과 2018 년에 업로드 된 악성코드(웹쉘)가 2023 년 1 월까지도 삭제되지 않은 채로 남아 있었고, 대규모 개인정보 유출 사고로 인해 과징금 68 억원과 과태료 2700 만원을 부과하였다. 이와 같은 사례로 웹쉘의 심각성을 알 수 있다.

이러한 웹쉘의 증가하는 보편성은 공격자에게 얼마나 단순하고 효과적인지에 기인할 수 있다. 또한 기존 웹쉘 정적 탐지에서 많이 사용된 통계적인 기법은 웹쉘 탐지 성능이 감소하는 난독화 편향 문제가 발생한다. 그러므로 난독화 편향 문제가 발생하지 않으면서, 웹쉘을 탐지하는 방법을 제안하려 한다.

1.2. 과제 목표 및 기대효과

본 과제는 딥 러닝 및 머신 러닝 기법을 활용한 난독화 웹쉘 탐지 모델 개발을 목표로 한다. 먼저 난독화 및 비난독화 도구를 기반으로 난독화 웹쉘 전처리 모듈을 구현하는 등 웹쉘 분석 환경을 구축한 다음, 난독화 웹쉘 탐지를 위한 textrank 기반 알고리즘을 구현한다. 마지막으로 시각화 도구를 이용하여 웹쉘 탐지 결과를 시각적으로 쉽게 파악할 수 있도록 한다.

이러한 과제는 난독화 웹쉘을 탐지함은 물론, 결과를 한눈에 파악하기 쉽게 하여 누구나 쉽게 웹쉘 탐지의 과정과 결과에 쉽게 접근할 수 있도록 한다.

1. **난독화 편향 문제 해결**: 웹쉘과 일반 파일의 opcode 와 AST 시퀀스를 이용해 이들 간의 관계 및 연관성을 분석하는 웹쉘 맞춤 TextRank 알고리즘을 활용하여, 전체적인 문서의 관계 및 연관성을 파악하고 특징을 추출하는 방법을 제안합니다. 이를 통해 난독화 편향 문제가 발생하지 않는 웹쉘 탐지 방법을 도입합니다. 데이터 셋에 난독화된 일반 파일이 포함되어 있을 때, 웹쉘 탐지 과정에서 난독화 편향 문제가 나타나지 않도록 특징을 추출하고 탐지하는 방법을 제공합니다. 이를 통해 오탐을 줄이고 탐지 정확성을 향상시킵니다.
2. **성능 평가**: 제안하는 방법에 다양한 머신러닝 알고리즘을 적용하여 정확도(accuracy), 정밀도(precision)에 대한 성능 평가를 진행합니다.
3. **시각화**: 시각화 도구를 이용하여 웹쉘 탐지 결과를 시각적으로 쉽게 파악할 수 있도록 한다. 이러한 과제는 난독화 웹쉘을 탐지함은 물론, 결과를 한눈에 파악하기 쉽게 하여 누구나 쉽게 웹쉘 탐지의 과정과 결과에 쉽게 접근할 수 있도록 한다.

2. 배경 지식

2.1. 웹쉘(Webshell)

Webshell 은 일반적으로 서버 관리를 돕기 위해 관리자가 사용하는 소프트웨어의 일종이다. 그러나 일부 경우, 공격자들은 악의적인 목적을 달성하기 위해 악성 webshell 을 사용하여 서버를 제어하기도 한다. 공격자에게 webshell 은 일종의 백도어(backdoor)로, 주로 ASP,

PHP, JSP 와 같은 스크립팅 언어로 작성된다. 이러한 웹 스크립트는 동적 상호작용 사이트를 생성할 수 있어, 공격자는 웹 페이지를 통해 웹 서버를 제어할 수 있다. 이러한 행위는 접근 로그(access log)에 기록된다 .

2.1.1. 웹쉘의 원리

Webshell 은 특정 페이지에 HTTP 요청을 보내고, 공격자가 보낸 명령을 실행하는 함수들을 사용하여 동작한다. 이러한 명령의 결과는 HTTP 요청에 대한 응답으로 공격자에게 전송된다 .

2.1.2. Webshell 의 분류[1]

웹쉘은 스크립트 언어의 기능과 크기에 따라 대략 세 가지 범주로 나눌 수 있다:

i. Big Trojan : 크기가 크고 명령 실행, 데이터베이스 작업 및 기타 악의적인 의도를 위한 종합적인 기능을 갖추고 있다. 또한, 대형 트로이 목마는 사용자 친화적인 그래픽 인터페이스를 갖추고 있다.

ii. One Word Trojan : 한 줄의 코드로 이루어진 트로이 목마이다. 그 짧은 길이 때문에 일반 파일이나 그림에 종종 삽입된다. 초기 공격 도구(예: Chinese Chopper)와 연결되면 대형 트로이 목마와 같은 다양한 기능을 수행할 수 있다.

iii. Small Trojan : 크기가 작고 숨기기 쉽지만 일반적으로 업로드 기능만 갖추고 있다. 대부분의 웹사이트가 파일 업로드 시 크기 제한을 두기 때문에, 공격자들은 일반적으로 소형 트로이 목마를 통해 먼저 업로드 권한을 획득한 후, 주요 기능을 수행하기 위해 대형 트로이 목마를 웹사이트에 업로드한다.

2.2. 텍스트랭크(TextRank)

TextRank 알고리즘은 문서 내의 문장 또는 단어들 간의 중요도를 계산하는 그래프 기반 텍스트 요약 및 키워드 추출 알고리즘이다. 이 알고리즘은 구글의 검색 엔진 기반 알고리즘인 PageRank 알고리즘에서 파생되었으며, PageRank 알고리즘의 개념을 텍스트에 적용한 것이다[3]. PageRank 알고리즘이 그래프 랭킹 알고리즘으로

페이지의 링크 정보를 기반으로 페이지의 중요도를 판단하는 반면[4][5], TextRank 알고리즘은 단어의 연관성을 통해 중요도를 판단한다[3].

TextRank 알고리즘의 전체 과정은 다음과 같다.

1. 토큰화 및 태깅: 텍스트를 토큰화하고 각 토큰에 품사 태그를 부여한다.
2. Syntactic Filtering: 특정 품사만 남겨두고 나머지 단어들은 제거한다.
3. 그래프 생성: 단어 간의 동시 출현 빈도를 계산하여 그래프를 생성한다.
4. 랭킹 계산: 생성된 그래프를 기반으로 단어 또는 문장의 중요도를 계산한다.
5. 키워드 추출 및 문서 요약: 중요도를 바탕으로 키워드를 추출하거나 문장을 요약한다[3].

TextRank 알고리즘은 단어 간의 의미적 유사성을 고려하여 문맥을 반영한 텍스트 요약을 제공하며, 문서 요약, 키워드 추출, 정보 검색 등 다양한 텍스트 마이닝 작업에 활용된다[3].

2.3. 지도학습 알고리즘

지도학습(Supervised Learning)은 기계 학습의 한 유형으로, 알고리즘이 주어진 입력 데이터와 해당하는 정답(레이블)을 사용하여 학습하는 방법이다. 지도학습의 주요 목표는 새로운 데이터에 대해 정확한 예측을 수행할 수 있는 모델을 구축하는 것이다. 이는 웹쉘 난독화 탐지와 같은 보안 문제를 해결하는 데에도 효과적으로 적용될 수 있다.[2]

2.3.1. 지도학습의 원리

지도학습에서, 알고리즘은 입력 데이터(features)와 이와 관련된 출력 데이터(labels)를 포함하는 학습 데이터셋을 사용한다. 웹쉘 탐지의 경우, 입력 데이터는 웹 요청 및 서버 로그 등의 특징을 포함하며, 출력 데이터는 정상 및 악성 웹쉘로 레이블링된 데이터이다. 알고리즘은 이 데이터셋을 분석하여 입력과 출력 간의 관계를 학습한다. 학습이 완료되면, 새로운 웹 요청이 주어졌을 때 이를 기반으로 악성 여부를 예측할 수 있다.

2.3.2. 지도학습 알고리즘의 종류

웹쉘 난독화 탐지에 효과적인 몇 가지 주요 지도학습 알고리즘은 다음과 같다:

i. 서포트 벡터 머신(Support Vector Machine, SVM): 데이터를 분류하기 위한 초평면을 찾는 방법으로, 고차원 공간에서도 효과적으로 작동한다. SVM 은 웹쉘 탐지에서 높은 정확도를 제공할 수 있다 .

ii. 랜덤 포레스트(Random Forest, RF): 여러 결정 트리를 앙상블하여 예측 성능을 향상시키는 방법으로, 과적합을 방지하고 일반화 성능을 높인다. RF 는 다양한 특징을 고려할 수 있어 웹쉘 탐지에 유리하다 .

iii. XGBoost(Extreme Gradient Boosting): 여러 결정 트리를 순차적으로 학습하고 앙상블하여 예측 성능을 향상시키는 강력한 그래디언트 부스팅 알고리즘으로, 매우 높은 예측 정확도를 제공한다. XGBoost 는 웹쉘의 복잡한 패턴을 효과적으로 학습할 수 있다 .

iv. 결정 트리(Decision Tree): 데이터의 특징을 기반으로 분할하여 예측을 수행하는 트리 구조의 모델이다. 이해와 해석이 용이하지만, 단독으로 사용 시 과적합될 가능성이 있다.

v. 로지스틱 회귀(Logistic Regression): 이진 분류 문제를 해결하기 위한 방법으로, 웹쉘 탐지에서 정상과 악성의 이진 분류를 수행할 수 있다.

vi. 신경망(Neural Networks): 인간의 두뇌 구조를 모방한 모델로, 복잡한 패턴 인식과 예측 작업에 유리하다. 다층 퍼셉트론(Multi-layer Perceptron, MLP)과 같은 구조가 웹쉘 탐지에 활용될 수 있다.

2.3.3. 지도학습의 응용 분야

지도학습 알고리즘은 웹쉘 난독화 탐지를 포함한 다양한 보안 분야에서 활용될 수 있다. 예를 들어, 스팸 이메일 필터링, 악성 코드 탐지, 이상 행위 탐지 등이 있다. 이러한 알고리즘은 대량의 레이블 데이터가 존재할 때 매우 효과적이며, 데이터에서 유의미한 패턴을 찾아내어 정확한 예측을 가능하게 한다.

지도학습 알고리즘은 웹쉘 난독화 탐지의 중요한 부분을 차지하며, 웹 보안을 강화하기 위해 널리 사용되고 있다. 이 알고리즘을 잘 이해하고 적절히 활용하는 것이 성공적인 웹쉘 탐지와 보안 모델 개발의 핵심이다.

2.4. PHP(Hypertext Preprocessor)

PHP 는 웹 개발을 위해 설계된 서버측 스크립트 언어이다. 실행 흐름은 다음과 같다. PHP 엔진이 소스 코드를 읽고 파싱한 뒤, AST 를 생성한다. 파싱된 AST 는 실행할 수 있는 opcode 로 컴파일된다. 생성된 opcode 는 PHP 가상 머신(Virtual Machine)에서 실행되고 웹 서버 응답을 생성한다.

2.4.1. Opcode(Operation Code)

Opcode 는 PHP 스크립트가 서버에서 실행될 때, PHP 엔진이 이해하고 실행하기 위해 변환하는 중단 단계의 코드를 나타낸다.

2.4.2. AST(Abstract Syntax Tree)

PHP 코드로부터 추출 가능한 AST 는 PHP 코드의 추상 구문 트리를 나타낸다. AST 는 파서(parser)에 의해 생성되며 PHP 코드를 토큰(Token)으로 분해하고 이를 트리 형태로 구성한다. 각 노드는 코드의 특정 구성 요소에 해당하며, 이러한 노드 간의 관계는 코드의 구조를 반영한다.

3. 프로젝트 소개 및 설계

3.1. 프로젝트 소개

기존 웹쉘 탐지 알고리즘은 난독화 되어있지 않은 파일들의 웹쉘 여부는 올바르게 판단하나 난독화된 파일들은 일반 파일도 웹쉘 파일로 분류해버리는 단점을 지니고 있다. 또한 이러한 단점은 일반 파일을 악성 파일로 분류하여 처리해버려 파일을 사용할 수 없게 된다는 문제점을 지니고 있다.

따라서, 이 프로젝트에서는 TextRank 를 기반으로하는 알고리즘을 설계하여 이러한 문제점을 해결하여 비난독화된 파일들의 웹쉘 여부 판단은 물론, 난독화된 파일을 분류할 때에도 일반 파일과 웹쉘 파일 여부를 정확하게 판별하고자 한다.

3.2. 시스템 동작 방식

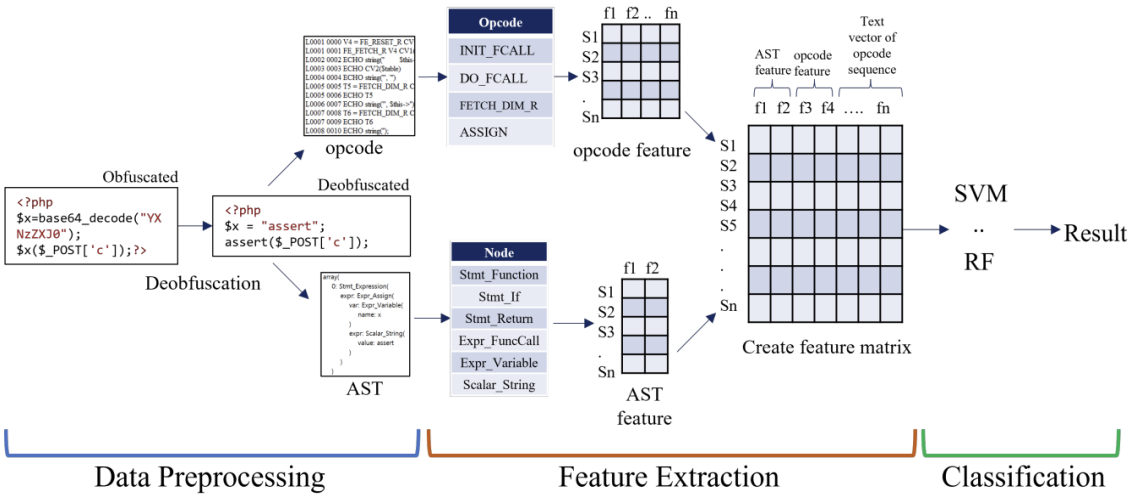


Figure 2: Detailed Proposed Webshell Detection Method Using the Webshell-Tailored TextRank Algorithm

난독화된 소스코드를 비난독화하고 소스 코드로부터 opcode, AST 시퀀스를 생성하여 웹쉘 맞춤 TextRank 알고리즘을 적용한 후 특성 행렬을 생성하고 RF, SVM, XGBoost 머신러닝 알고리즘을 활용하여 성능 평가를 수행한다.

1. 전처리 및 특징 추출
    a. 난독화가 적용된 코드를 비난독화한다.

b. 비난독화된 코드로부터 opcode 와 AST 를 생성한 뒤, opcode, AST 시퀀스를 이용해 opcode, AST 의 빈도를 확인한다. 그래프의 크기 증가를 방지하기 위해 threshold 를 지정하여 임계값 이하의 opcode, AST 는 삭제한다.

c. 단어 그래프를 생성한 뒤, 주어진 opcode, AST 시퀀스를 토큰화하고 단어 빈도를 계산한다. 그리고 정규화를 수행하고 단어 간의 유사도를 나타내는 행렬, 즉 단어 그래프를 생성한다. 또한 가중치를 추가 부여하는 부분으로 opcode, AST 의 빈도를 확인하는 단계에서 빈도수가 높은 opcode 와 AST 를 리스트로 만들어 상위 항목에 대해서는 단어 그래프를 생성할 때 추가 가중치를 부여한다.

d. calculation 단계는 그래프를 입력으로 받아 각 노드의 중요도를 평가한다. 모든 노드에 대해 연결된 링크의 합계를 구한 뒤 0 이 아닌 경우에만 동작한다. 결과값은 딕셔너리 형태로 반환한다.

2. 분류

본 과제에서는 제안하는 특징 추출 알고리즘이 웹쉘 탐지에 효과적인지에 대해 초점을 맞춰 다양한 머신러닝 알고리즘(RF, SVM, XGBoost)을 사용해 성능 평가를 수행한다. 특징 추출 단계에서 생성된 특징 행렬을 다양한 머신러닝 알고리즘의 입력으로 사용하여 웹쉘을 탐지하는 단계이다.

3.3. 웹 페이지 구현

TextRank 를 기반으로 하여 개발한 알고리즘을 이용하여 탐지한 웹쉘 탐지 결과 데이터를 보기 좋게 웹 페이지 상에 그래프로 나타낸다. 이러한 웹 페이지는 React + tailwind 를 사용하여 구현할 예정이다.

4. 개발 일정 계획 및 담당 업무

4.1. 개발 일정

구분	작업 일정																	
	6 월				7 월				8 월				9 월				10	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
기초 지식 공부																		
복호화 및 전처리																		
웹쉘 특화 textrank 구현																		
AI 부분 구현																		
결과 분석																		
웹 페이지 개발																		
최종 보고서 및 발표 준비																		

4.2. 담당 업무

문정윤	- 기초 지식 공부 - 데이터 전처리 - 알고리즘 구현 - 웹 프론트 개발 - 발표 및 시연 준비
차기은	- 기초 지식 공부 - 머신러닝 알고리즘 구현 - 보고서 작성 - 웹 프론트 개발
구지원	- 기초 지식 공부 - 데이터 전처리 및 특징 추출 - 웹쉘 특화 textrank 구현 - 웹 백엔드 개발

5. 참고 문헌

[1] Zhu, T., Weng, Z., Fu, L., & Ruan, L. (2020). A Web Shell Detection Method Based on Multiview Feature Fusion. *Applied Sciences, 10*(18), 6274.https://www.mdpi.com/2076-3417/10/18/6274

[2] Yixin Wu, Yuqiang Sun, Cheng Huang (2020). Session-Based Webshell Detection Using Machine Learning in Web Logs. https://www.mdpi.com/2076-3417/10/18/6274

[3] Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 404-411.

[4] Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30(1-7), 107-117.

[5] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab.