난독화에 강한 AI 기반 웹쉘 탐지

중간보고서

부산대학교 정보컴퓨터공학부

지도교수 : 최윤호

팀　명 : 문지기

팀　원 : 201914119 문정윤

202155507 구지원

202155614 차기은

< 목차 >

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1. 1. 요구조건

- 난독화가 적용된 코드를 비난독화
- 비난독화된 코드로부터 opcode 와 AST 생성
- opcode, AST 시퀀스를 이용하여 opcode, AST 의 빈도 확인
- 정규화를 수행하고 단어 간의 유사도를 나타내는 행렬, 즉 단어 그래프를 생성
- 그래프를 입력으로 받아 각 노드의 중요도를 평가

1. 2. 제약 사항 분석에 대한 수정사항

(1) 웹쉘 파일 보안 문제

과제에서 다뤄야하는 웹쉘이라는 파일은 보안관련 위험 파일이라 안티 바이러스프로그램에 의해 사전 차단 될 수 있음. 따라서 도커에 우분투를 설치하여 가상환경에서 실행함.

(2) 비난독화 도구 선택

비난독화를 해주는 사이트의 API 를 이용하여 비난독화를 수행하려고 했으나 문제로 실패하여 다른 방법을 모색함. 이후 PHPDeobfuscator 를 사용한 비난독화에 성공하여 이를 이용하여 해결하기로 함.

(3) opcode 추출 명령어

`phpdbg -qrr` 명령어를 사용하여 opcode 를 추출하려했으나 실패하여 PECL 을 이용한 VLD 로 추출을 시도하였음. 이 또한 실패하여 방법을 모색하던 중 `phpdbg -p` 명령어로 opcode 추출에 성공함.

2. 설계 상세화 및 변경 내역

2.1 비난독화

2.1.1 비난독화(Deobfuscation)란?

비난독화는 난독화된 코드를 원래의 가독성 있는 형태로 되돌리는 과정이다. 난독화는 원본 코드를 의도적으로 복잡하게 만들어 분석이나 이해를 어렵게 하는 방법으로, 주로 소프트웨어의 소스 코드를 보호하거나 악성 코드 분석을 방해하기 위해 사용된다. 비난독화는 이러한 난독화된 코드를 해독하여 원래의 의미를 파악할 수 있도록 해준다.

2.1.2 PHP 비난독화의 중요성

PHP 와 같은 웹 스크립팅 언어는 서버 측에서 실행되기 때문에 코드의 보안과 무결성이 중요하다. 하지만 난독화 기술은 악성 행위자가 코드를 숨기기 위해 사용할 수 있으며, 이 경우 코드의 실제

의도를 파악하기 어려워진다. 비난독화는 악성 코드 분석, 보안 취약점 식별, 코드 유지보수 등에서 중요한 역할을 한다.

2.2. PHPDeobfuscator 도구 소개

2.2.1 PHPDeobfuscator 란?

PHPDeobfuscator 는 난독화된 PHP 코드를 비난독화하여 원래의. 코드 형태로 복원하는 오픈 소스 도구다. 이 도구는 자동으로 난독화된 PHP 코드를 분석하고, 가독성 있는 형태로 되돌리는 데 사용된다.

2.2.2 주요 기능

- **코드 디코딩:** 난독화된 PHP 코드를 원래의 가독성 있는 코드로 복원한다.
- **자동 분석:** 코드 내 난독화된 패턴을 자동으로 식별하고, 이를 기반으로 디코딩을 진행한다.
- **오픈 소스:** GitHub 에서 무료로 제공되며, 누구나 사용할 수 있다.

2.3. PHPDeobfuscator 사용 방법

2.3.1 CLI 를 통한 사용 방법

PHPDeobfuscator 는 CLI(명령줄 인터페이스)를 통해 사용될 수 있다. 아래는 주요 사용법이다.

- **기본 사용법:**

  php index.php -f obfuscated.php

  - f: 필수 인자로, 비난독화할 난독화된 PHP 파일을 지정하는 데 사용된다.

- **옵션:**

  - t: 출력 노드 트리를 디버깅 목적으로 덤프하는 옵션이다.
  - o: 각 표현식 옆에 원본 코드에 대한 주석을 추가하여 출력하는 옵션이다.

디코딩된 결과는 표준 출력(STDOUT)으로 출력된다.

2.4 AST

2.4.1. AST 란?

추상 구문 트리(Abstract Syntax Tree, AST)는 소스 코드의 계층적 구문 구조를 표현한 트리이다. 소스 코드의 문법적 구조를 추상화하여 표현하며, 각 노드는 연산자, 변수, 함수 호출 등 다양한 코드 요소를 나타낸다. 컴파일러나 정적 분석 도구에서 널리 사용되는 AST 는 코드 최적화, 구문 오류 검출, 코드 리팩토링 등에 중요한 역할을 한다.

2.4.2. AST 생성 방법

PHP 의 AST 생성에는 **PHP-Parser** 라이브러리를 사용했다. 이. 라이브러리는 PHP 코드를 파싱하여 AST 를 생성하고, 이를 구조화된 데이터로 제공한다. PHP-Parser 를 통해 생성된 AST 는 다양한 형식(JSON, XML 등)으로 저장하거나, 코드 분석 및 변환 작업에 활용할 수 있다.

AST 를 생성하는 일반적인 과정은 다음과 같다:

1. **PHP-Parser 초기화**: PHP-Parser 라이브러리를 사용해 파서를 설정한다.
2. **코드 파싱**: 분석할 PHP 소스를 파싱하여 AST 를 생성한다.
3. **AST 저장**: 생성된 AST 는 원하는 형식으로 변환 후 파일로 저장하거나, 코드 분석 작업에 바로 활용할 수 있다.

2.5 Opcode

2.5.1. Opcode 란?

Opcode(운영 코드)는 PHP 의 가상 머신에서 실행되는 저수준 명령어로, 프로그램의 소스 코드가 파싱되고 추상 구문 트리(Abstract Syntax Tree, AST)가 생성된 후 컴파일 과정에서 생성된다. PHP 프로그램의 실행 과정에서, AST 는 의미 분석과 최적화 단계를 거쳐 opcode 로 변환되며, 이후 가상 머신(zend 엔진)에 의해 실제로 실행된다.

Opcode 는 소스 코드의 최종 실행 단계에서 중요한 역할을 한다. 즉, AST 는 코드의 구조적 분석에 사용되지만, opcode 는 코드의 실제 동작을 결정하는 명령어들로 구성됩니다. 이로 인해, opcode 는

프로그램이 어떻게 동작하는지를 분석하는 데 핵심적인 정보를 제공한다.

2.5.2. PHPDBG 를 통한 Opcode 추출

Opcode 를 추출하기 위해 PHP 에서는 PHPDBG 라는 도구를 사용한다. PHPDBG 는 PHP 의 공식 디버깅 도구로, 코드 실행 제어, 변수 추적, 중단점 설정 등의 기능을 제공하며, 특히 opcode 를 추출하는 데 유용하다.

2.6 Opcode 와 AST 의 차이점

- **AST**: 코드의 구조적 분석과 관련이 있으며, 코드의 구문적 구성 요소와 논리적 흐름을 나타낸다.
- **Opcode**: 코드의 실제 실행 단계에서 가상 머신에 의해 처리되는 명령어로, 코드의 동작을 직접적으로 제어한다.

본 연구에서는 이러한 AST 와 opcode 를 활용하여 코드의 특징을 추출하고 분석한다. AST 는 코드의 논리 구조를 이해하는 데 유용하고, opcode 는 코드가 실제로 어떻게 동작하는지를 분석하는 데 도움을 준다.

2.7 OPCODE 와 AST 활용 방안

특징 추출 (Feature Extraction)

특징 추출 과정은 머신러닝 모델의 성능을 높이기 위해 원시 데이터로부터 유용한 특징들을 선택하고, 조합하며, 생성하는

과정이다. 본 연구에서는 opcode 와 AST 를 이용하여 웹쉘 탐지에 필요한 유의미한 특징을 추출한다.

2.7.1 Opcode 기반 특징 추출

Opcode 는 PHP 코드가 실행되는 과정 에서 생성 되는 명령어 집합으로, PHP 의 가상 머신에서 실제로 실행된다. 본 연구에서는 다음과 같은 과정을 통해 opcode 기반의 특징을 추출한다:

- **Opcode 시퀀스 생성:** PHP 코드에서 실행된 opcode 를 추출하여 시퀀스를 만든다. 예를 들어, PHP 코드 <?php eval($_POST['a']); ?>는 연속적으로 FETCH_R, FETCH_DIM_R, INCLUDE_OR_EVAL, RETURN 과 같은 opcode 로 변환된다.
- **TF-IDF 알고리즘 적용:** 생성된 opcode 시퀀스에 대해 TF-IDF (Term Frequency-Inverse Document Frequency) 알고리즘을 사용하여 텍스트 벡터를 생성한다. TF-IDF 는 특정 opcode 가 문서에서 얼마나 중요한지를 측정하는 통계적 방법이다.
    - **Term Frequency (TF):** opcode 가 문서 내에서 얼마나 자주 나타나는지를 측정한다.
    - **Inverse Document Frequency (IDF):** opcode 가 여러 문서에 걸쳐 얼마나 드물게 나타나는지를 측정한다.
- **웹쉘 맞춤 TextRank 알고리즘 적용:** TF-IDF 값을 기반으로 TextRank 알고리즘을 사용하여 중요한 opcode 를 추출한다. TextRank 는 단어 간의 문맥적 연관성을 고려하여 중요도를 평가한다.

2.7.2 AST 기반 특징 추출

AST 는 PHP 소스 코드의 구문 구조를 트리 형태로 표현한 것이다.

AST 기반의 특징 추출 과정은 다음과 같다:

- **AST 노드 필터링:** AST 를 분석하여 필요 없는 노드를 제거하고, 중요한 노드들만을 추출한다.
- **특징 벡터 생성:** 필터링된 AST 노드를 기반으로 웹쉘 탐지에 유용한 피쳐를 생성한다.

3. 갱신된 과제 추가계획

3.1. 웹페이지 구성 요소

특징 추출을 기반으로 웹 페이지에 띄울 내용을 구성할 때, 사용자에게 의미 있는 정보와 성능 결과를 시각적으로 전달하는 것이 중요합니다. 아래는 웹 페이지에 포함될 주요 요소들입니다.

1. 웹쉘 탐지 시스템 소개

- **간략한 설명**: 웹쉘 탐지 시스템의 목적과 특징 추출의 중요성을 설명하는 간단한 텍스트.
- **특징 추출의 역할**: 특징 추출이 웹쉘 탐지에서 어떤 역할을 하는지, 그리고 그 중요성을 설명하는 부분.
- **시스템 아키텍처 다이어그램**: 전체 웹쉘 탐지 시스템의 흐름을 보여주는 다이어그램(특징 추출, 모델 학습, 탐지 등).

2. 특징 추출 과정 시각화

- **특징 추출 과정의 단계별 설명**:

- 각 단계(코드 복잡도 분석, 문자열 패턴 추출, 난독화 탐지 등)를 텍스트와 함께 간단한 아이콘이나 다이어그램으로 설명.

- **데이터 처리 흐름**: 원본 데이터에서 특징이 어떻게 추출되고, 이를 통해 모델이 학습되는 과정을 보여주는 인터랙티브한 플로우 차트.

3. 성능 평가 지표

- **성능 지표 요약 테이블**: Accuracy, Precision, Recall, F1-Score, ROC-AUC 등의 주요 성능 지표를 테이블 형태로 제공.
- **모델별 성능 비교 그래프**:
  - RF, SVM, XGBoost 의 성능을 비교하는 막대 그래프 또는 선 그래프.
  - 각 성능 지표를 시각적으로 쉽게 비교할 수 있도록 색상으로 강조.
- **ROC 커브 시각화**: RF, SVM, XGBoost 의 ROC 커브를 한 화면에 보여주는 그래프. AUC 값도 함께 표시.

4. 기존 탐지 방법론과의 비교

- **기존 방법론과 제안 방법론의 성능 비교 테이블**: Pan et al.과 Cui et al.의 방법론과 본 논문에서 제안한 방법론의 성능을 한눈에 비교할 수 있는 표.
- **비교 그래프**: 각 방법론의 성능을 시각적으로 비교하는 막대 그래프, 특히 제안 방법론이 어떻게 더 나은 성능을 보이는지 강조.

- **난독화 편향 검증 결과**: 난독화된 데이터셋에서의 성능 변화를 보여주는 그래프. 기존 방법론이 성능이 저하되는 반면, 제안된 방법론은 성능이 유지되는 모습을 강조.

5. 결과 요약 및 결론

- **요약 섹션**: 특징 추출 과정의 중요성과 제안된 방법론의 우수성을 요약하는 텍스트.
- **다음 단계**: 추가 연구 방향이나 시스템의 향후 발전 가능성에 대한 짧은 논의.

6. FAQ 및 지원

- **자주 묻는 질문(FAQ)**: 시스템 사용 중 발생할 수 있는 일반적인 질문에 대한 답변을 제공.
- **지원 섹션**: 추가 질문이나 기술 지원이 필요한 경우 연락할 수 있는 방법.

7. 부가 정보 및 다운로드

- **논문 및 자료 다운로드**: 관련 연구 논문이나 실험 데이터셋을 다운로드할 수 있는 링크.
- **관련 리소스**: 웹쉘 탐지와 관련된 추가 자료나 외부 링크 제공.

8. 사용자 피드백

- **피드백 섹션**: 사용자가 시스템을 사용한 후 피드백을 제공할 수 있는 양식이나 설문지.

이런 구성 요소들을 통해 웹 페이지 방문자는 웹쉘 탐지 시스템의 작동 원리와 성능을 명확하게 이해하고, 시스템을 실험해 볼 수 있는 기회를 가지며, 추가 자료를 탐색할 수 있습니다.

3.2. 인공지능 - 특징추출

웹쉘 탐지 시스템에서 인공지능 알고리즘을 효과적으로 활용하기 위해서는 적절한 특징 추출 과정이 필수적입니다. 웹쉘은 정상적인 웹 서버 파일과 매우 유사한 구조를 가지기 때문에, 단순한 규칙 기반 탐지 방법으로는 탐지하기 어렵습니다. 따라서, 웹쉘 파일과 정상 파일을 구분할 수 있는 중요한 특징을 추출하는 것이 탐지 성능의 핵심입니다. 특징 추출은 높은 정확도와 낮은 오탐율을 달성하기 위한 필수 단계입니다.

1. 성능 평가 지표

특징 추출 후, 본 논문에서는 제안된 방법의 유효성을 검증하기 위해 다양한 성능 평가 지표를 사용합니다. 주요 성능 지표는 다음과 같습니다:

- **정확도(Accuracy)**: 전체 샘플 중 올바르게 분류된 샘플의 비율을 나타내며, 모델의 전반적인 성능을 평가합니다.
- **정밀도(Precision)**: 양성으로 분류된 샘플 중 실제 양성인 샘플의 비율을 측정합니다. 이는 오탐율이 낮아야 하는 상황에서 특히 중요합니다.
- **재현율(Recall)**: 실제 양성 샘플 중 양성으로 정확하게 분류된 샘플의 비율을 나타내며, 민감도(Sensitivity)로도 불립니다.
- **F1-점수(F1-Score)**: 정밀도와 재현율 간의 조화 평균으로, 두 지표 간의 균형을 평가합니다.

- **ROC-AUC**: ROC 커브 아래 영역을 측정하여 모델의 분류 성능을 시각적으로 표현합니다. AUC 값이 1 에 가까울수록 성능이 우수하며, 0.5 에 가까울수록 성능이 낮습니다.

이러한 지표를 바탕으로 다양한 머신러닝 알고리즘(RF, SVM, XGBoost)을 적용해 제안된 방법론의 성능을 평가합니다.

2. 머신러닝 알고리즘

- **랜덤 포레스트(Random Forest, RF)**: 여러 개의 의사 결정 트리를 조합하여 높은 정확도를 제공하지만, 많은 트리를 사용할 경우 메모리 사용량이 증가하고 모델 해석이 어려울 수 있습니다.
- **서포트 벡터 머신(Support Vector Machine, SVM)**: 데이터를 고차원 공간으로 매핑하여 복잡한 패턴을 잘 분류할 수 있습니다. 마진 최대화를 통해 과적합에 강한 모델을 만들 수 있지만, 데이터셋이 클 경우 학습과 예측 시간이 오래 걸릴 수 있습니다.
- **엑스트림 그레디언트 부스팅(Extreme Gradient Boosting, XGBoost)**: 앙상블 학습과 Gradient Boosting 알고리즘을 활용하여 높은 예측 성능을 제공합니다. 하지만 하이퍼파라미터 설정 과정이 복잡하며, 잘못된 값을 설정할 경우 학습이 제대로 이루어지지 않을 수 있습니다.

특히, 기존 방법론과 비교했을 때 난독화 편향 문제에서도 제안된 방법이 높은 정확도를 유지하며, 난독화 편향 문제에 강인한 성능을 보일 것으로 기대됩니다.

3.3. 갱신된 과제 수행 계획

구분	작업 일정						
	8 월	9 월				10 월	
	5	1	2	3	4	1	2
웹쉘 특화 textrank 구현							
AI 부분 구현							
결과 분석 (전체)							
웹 페이지 개발							
최종 보고서 및 발표 준비							

4. 구성원별 진척도

문정윤	**복호화 및 전처리** - 난독화된 파일 비난독화 - AST 추출 **웹 페이지 개발** - 깃허브 레포지터리 생성 - React 초기 프로젝트 생성

구지원	**복호화 및 전처리** - Opcode 추출 - AST 추출 **결과 분석** - 특징 추출 초기 코드 구현
차기은	**복호화 및 전처리** - 난독화된 파일 비난독화 - Opcode 추출 **결과 분석** - 특징 추출 초기 코드 구현

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5.1 비난독화

PHPDeobfuscator 를 이용해서 webshell 코드를 비난독화 진행.

5.1.1 비난독화 진행 코드

웹쉘 php 파일은 총 2917 개로 구성되어 있다.

PHPDeobfuscator 를 활용해서 전체 파일을 한번에 비난독화 할 수있는 php 파일을 만든 후 비독화를 진행했다.

```php
<?php
$inputDir = '/root/gradProj/dataset/webshell_file';

$outputDir = '/root/gradProj/result/webshell_deobfuscated';

$phpDeobfuscatorPath = '/root/PHPDeobfuscator';

if (!file_exists($outputDir)) {
    mkdir($outputDir, 0777, true);
}

$files = scandir($inputDir);

foreach ($files as $file) {
    if ($file == '.' || $file == '..' || is_dir($inputDir . '/' . $file)) {
        continue;
    }

    $inputFilePath = $inputDir . '/' . $file;

    $outputFilePath = $outputDir . '/' . $file;

    $command = "php $phpDeobfuscatorPath/index.php -f $inputFilePath > $outputFilePath";
    exec($command, $output, $return_var);

    if ($return_var !== 0) {
        echo "error while de... file  $file\n";
        echo "Command: $command\n";
        echo "Output:\n";
        print_r($output);
    } else {
        echo "file  $file de... complete: $outputFilePath\n";
    }
}

echo "de~ complete...~L\n";
?>
```

5.1.2 비난독화 결과 코드

- 웹쉘 파일

```php
<?php
eval(str_rot13('riny($_CBFG[pzq]);'));
?>
```

이 코드는 PHP 에서 난독화된 형태로 작성된 웹쉘이다. str_rot13(). 함수는 입력 문자를 ROT13 방식으로 변환하여 암호화한다. 이 경우 str_rot13() 함수 안에 있는 문자열을 복호화하면 eval($_POST[cmd])이 된다.

- 비난독화 한 웹쉘 파일

```php
<?php
eval($_POST[cmd]);
```

5.2 AST

PHP-Parser 라이브러리를 통해 웹쉘과 정상 PHP 파일에서 AST 를 추출했다. 추출된 AST 는 JSON 으로 저장해서 데이터 가공을 용이하도록 했다.

5.2.1 AST 추출 진행 코드

PHP-Parser 를 통해 AST 를 생성하는 과정은 다음과 같은 코드로 설명할 수 있다. 다음은 PHP-Parser 를 이용해서 특정 디렉토리에 있는 PHP 파일들로부터 AST 를 추출하고, 이를 별도의 디렉토리에 저장하는 스크립트 이다.:

```php
<?php
require 'vendor/autoload.php'; // Composer의 autoload 파일 로드

use PhpParser\Error;
use PhpParser\ParserFactory;

// PHP 파일이 있는 디렉토리 경로
$phpFilesDir = '/Users/jiwon/Downloads/dataset/normal_file';

// AST를 저장할 디렉토리 경로
$outputDir = '//Users/jiwon/Downloads/dataset/astVersion2Json';
if (!is_dir($outputDir)) {
    mkdir($outputDir, 0777, true);
}

// PHP-Parser 인스턴스 생성
$parser = (new ParserFactory)->create(ParserFactory::PREFER_PHP7);

// PHP 파일들을 순회하면서 AST 추출 및 저장
foreach (glob("$phpFilesDir/*.php") as $phpFile) {
    $code = file_get_contents($phpFile);
    $filename = basename($phpFile, '.php'); // 확장자를 제외한 파일명 추출

    try {
        $ast = $parser->parse($code);
        // AST를 JSON 형식으로 저장
        file_put_contents("$outputDir/{$filename}_ast.json", json_encode($ast, J
SON_PRETTY_PRINT));
        echo "AST extracted for $phpFile and saved to $outputDir/{$filename}_as
t.json\n";
    } catch (Error $e) {
        echo "Parse error in $phpFile: {$e->getMessage()}\n";
    }
}
```

위 코드에서는 PHP-Parser 를 사용하여 입력된 PHP 코드의 AST 를 생성하고, 이를 JSON 형식으로 저장한다.

5.2.2 AST 추출 결과


```
"stmts": [
                    {
                        "nodeType": "Stmt_Property",
                        "flags": 1,
                        "props": [
                            {
                                "nodeType": "Stmt_PropertyProperty",
                                "name": {
                                    "nodeType": "VarLikeIdentifier",
                                    "name": "id",
                                    "attributes": {
                                        "startLine": 43,
                                        "endLine": 43
                                    }
                                },
                                "default": null,
                                "attributes": {
                                    "startLine": 43,
                                    "endLine": 43
                                }
                            }
                        ],
                        "type": null,
                        "attrGroups": [],
                        "attributes": {
                            "startLine": 43,
                            "comments": [
                                {
                                    "nodeType": "Comment_Doc",
                                    "text": "\/**\n * @var string ID of the action\n
                                    "line": 40,
                                    "filePos": 1273,
                                    "tokenPos": 25,
                                    "endLine": 42,
                                    "endFilePos": 1319,
                                    "endTokenPos": 25
                                }
                            ],
                            "endLine": 43
                        }
                    },
```


위의 코드는 추출된 AST 의 일부로, Stmt_Property 노드는 프로퍼티 선언을 나타내고 있다.

5.3 Opcode

Phpdbg 라이브러리를 통해 웹쉘과 정상 PHP 파일에서 Opcode 를 추출했다. 추출된 Opcode 는 txt 파일로 저장된다.

5.3.1 Opcode 추출 진행 코드

Phpdbg 를 통해 Opcode 를 제출하는 과정은 다음과 같은 코드로 설명할 수 있다. 다음은 PHPDBG 를 이용해 특정 디렉토리에 있는 PHP 파일들로부터 opcode 를 추출하고, 이를 별도의 디렉토리에 저장하는 스크립트이다:

```
#!/bin/bash

# PHP 파일이 있는 디렉토리 경로
PHP_FILES_DIR="/Users/jiwon/Downloads/dataset/deobfu"

# Opcode를 저장할 디렉토리 경로
OUTPUT_DIR="/Users/jiwon/Downloads/dataset/webshell_opcode_deobfu"
mkdir -p "$OUTPUT_DIR"

# 모든 출력을 저장할 파일
LOG_FILE="$OUTPUT_DIR/extraction_log.txt"

# PHP 파일을 순회하면서 Opcode 추출
for php_file in "$PHP_FILES_DIR"/*.php; do
    # 파일 이름만 추출 (확장자 제외)
    filename=$(basename -- "$php_file")
    filename="${filename%.*}"

    # 출력 파일 경로
    output_file="$OUTPUT_DIR/${filename}_opcode.txt"

    # Opcode 추출 및 저장
    phpdbg -p "$php_file" > "$output_file" 2>> "$LOG_FILE"

    echo "Opcode extracted for $php_file and saved to $output_file" >> "$LOG_FIL
E"
done
```

이 스크립트는 지정된 디렉토리 내의 모든 PHP 파일을 대상으로 phpdbg 를 사용하여 opcode 를 추출한다. 추출된 opcode 는 파일별로 텍스트 파일로 저장되며, 모든 작업 로그는 extraction_log.txt 파일에 기록된다.

5.3.2 Opcode 추출 결과

```
function name: (null)
L1-396 {main}() /Users/jiwon/Downloads/dataset/normal_file/translations.php - 0x10149f70
0 + 18 ops
 L10   #0     INCLUDE_OR_EVAL<16>     "/Users/jiwon/Down"+
 L11   #1     INCLUDE_OR_EVAL<16>     "/Users/jiwon/Down"+
 L13   #2     INIT_FCALL<2>           112                 "class_exists"
 L13   #3     SEND_VAL                "Translations"      1
 L13   #4     SEND_VAL                false               2
 L13   #5     DO_ICALL                                                    @2
 L13   #6     BOOL_NOT                @2                                  ~3
 L13   #7     JMPZ                    ~3                  J10
 L15   #8     DECLARE_CLASS           "translations"                      @4
 L184  #9     DECLARE_INHERITED_CLASS "gettext_translati"+ "Translations"  @5
 L314  #10    INIT_FCALL<2>           112                 "class_exists"
 L314  #11    SEND_VAL                "NOOP_Translations" 1
 L314  #12    SEND_VAL                false               2
 L314  #13    DO_ICALL                                                    @6
 L314  #14    BOOL_NOT                @6                                  ~7
 L314  #15    JMPZ                    ~7                  J17
 L319  #16    DECLARE_CLASS           "noop_translations"                 @8
 L396  #17    RETURN<-1>              1
[0;32m[Script ended normally][0m
```

Opcode 분석

위 예시는 PHP 코드를 기반으로 생성된 opcode 를 보여준다. 각 줄은 명령어(opcode), 인수 및 해당 코드의 위치를 나타낸다. 몇 가지 중요한 명령어를 설명하면:

- INCLUDE_OR_EVAL: 파일 포함 또는 코드 평가 작업을 수행한다.
- INIT_FCALL: 함수 호출을 초기화한다.
- SEND_VAL: 함수 호출 시 인수를 전달한다.

- DECLARE_CLASS: 새로운 클래스를 선언한다.
- JMPZ: 조건이 참이 아닐 경우, 특정 위치로 점프한다.

이러한 opcode 는 PHP 코드의 실행 흐름을 매우 구체적으로 보여주며, 코드의 실행 단계를 추적하고 최적화하는 데 유용하다.