모바일 애플리케이션 정적 분석 자동화를 이용한 오펜시브 관점의 웹 서비스 보안성 향상

송상준 [1]
[1] 고려대학교 SW · AI 융합대학원 소프트웨어보안학과 석사과정

s0ngsari@korea.ac.kr

Improving Web Service Security from an Offensive Perspective Using Automated Static Analysis of Mobile Applications

Sang-jun Song[1]
[1]Korea University Graduate School of SW · AI Convergence

요 약

본 논문에서는 오펜시브 보안 관점에서 ANTLR4 를 이용한 모바일 앱 정적 분석 자동화를 통해 API 서버에서 제공하는 엔드포인트와 파라미터 정보를 알아낸다. 이를 통해 API 서버에 잔존하는 레거시 엔드포인트를 식별할 뿐만 아니라 API 서버를 대상으로 퍼징 및 블랙 박스 테스팅과 같은 취약점 점검의 초석을 마련한다.

1. 서론

현대 소프트웨어 개발 환경에서 Application Programming Interface (API)[1]의 역할은 중요하게 작용한다. API 는 애플리케이션 간 효율적인 통신 및 데이터 교환이 가능하며, 웹 및 모바일 애플리케이션 간 개발 환경에 있어 재사용성, 표준화와 같은 편의를 제공한다. 서비스 제공자는 다양한 REST API[2]서버 및 엔드포인트를 통해 이용자의 개인정보를 비롯한 서비스에서 다루는 정보를 송수신하거나 저장한다. 이처럼 API 중요성이 증가함에 따라, 이를 겨냥한 보안 위협 또한 급증하고 있다. API 서버를 공격하여 이용자의 개인정보 및 기업의 민감한 자산 탈취까지 이어질 수 있기 때문에 API 서버 보안이 중요한 과제로 대두되고 있다. 이에 따라 공격자 관점에서 실제 서비스를 대상으로 취약점을 발견하는 오펜시브 보안은 보안 향상에 있어 중요하게 작용한다.

일반적으로 API 서버에 접속했을 때 서버에서 제공하는 엔드포인트와 파라미터를 알 수 있는 방법이 없다. 따라서 공격자는 API 서버를 이용하는 프론트엔드 또는 모바일 앱을 분석하여 관련 정보를 파악한다. 프론트엔드 분석은 기업이 다양한 기능을 별도의 서비스로 제공하는 경우 API 서버에서 제공하는 모든 엔드포인트를 알아내는 것은 서비스를 제공하는 서버가 다양하게 분포되어 있을수록 모든 정보를 알아내기 힘들다는 한계점이 존재한다. 그러나 모바일 앱은 앱 이용자가 서비스의 모든 기능을 이용할 수 있도록 모든 API 에 대한 정보를 명시하고 HTTP 클라이언트를 통해 통신한다. 따라서, 이를 이용해 명시된 모든 API 엔드포인트와 파라미터를 분석할 수 있다.

정적 분석 자동화를 통해 앱에 구현된 API 정보를 알아내는 것은 각 라이브러리마다 구현 패턴이 다르기 때문에 완벽하게 분석하는 것은 어렵다. 그러나 ANother Tool for Language Recognition (ANTLR)[3]를 이용해 자바 문법을 분석하고, 분석한 결과를 모델링하여 라이브러리에 따라 분석기를 개발하는 방법으로 API 엔드포인트 및 파라미터를 알아낼 수 있다.

본 연구에서는 퍼징 및 블랙박스 테스팅과 같은 취약점 점검의 기반을 마련하기 위한 목적으로, ANTLR4 를 이용해 모바일 앱 분석을 자동화하여 API 정보를 확인하는 방법에 대한 연구를 소개한다.

2. Android HTTP Client 라이브러리와 분석 자동화

안드로이드에서 제공하는 HTTP Client 라이브러리는 크게 Retrofit[4]과 Okhttp[5]가 존재한다. 앱 개발자는

HTTP 통신이 필요한 경우 해당 라이브러리들을 이용하여 편리하게 통신 코드를 작성할 수 있다. 따라서, 최근 웹과 모바일 서비스를 모두 제공하는 서비스가 많아짐에 따라 해당 라이브러리에 대한 의존도가 높아지고 있다.

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

(그림 1) Retrofit 라이브러리 사용 예시

(그림 1)은 Retrofit 을 이용하여 REST API 통신을 위해 인터페이스를 구현한 코드이다. 통신을 원하는 엔드포인트와 파라미터, 그리고 반환하는 객체까지 간편하게 정의할 수 있다. 개발 관점에서는 다양한 서비스를 제공하기 위해 방대한 API 가 존재하는 경우 정확하고 빠르게 앱을 개발할 수 있다. 반면, 오펜시브 보안 관점에서 방대한 클라이언트 코드를 사람이 직접 분석하는 것은 취약점 점검 사전 작업에 있어 많은 리소스를 요구하는 등의 어려움이 존재한다. 따라서 분석 자동화에 대한 중요성은 점차 부각되고 있다.

분석 자동화에 있어서는 크게 정적 분석과 동적 분석으로 나뉘어진다. 정적 분석은 코드를 실행하지 않고 비 런타임 관점 분석 기법이며, 동적 분석은 코드를 실행하여 분석하는 런타임 관점의 분석 기법이다. 동적 분석은 실행되는 코드를 분석하는데에 있어서 코드 실행 흐름에 따라 변경되는 메모리와 레지스터를 추적하여 정적 분석에 비해 상대적으로 더 높은 신뢰도를 가진 결과를 보여준다. 그러나, 실행되는 코드에 한해서만 분석이 가능하기 때문에 실행되지 않은 코드에 대한 분석은 불가능하다.[6] 뿐만 아니라, 모바일 보안 솔루션의 보편화에 따라 많은 앱에 루팅 탐지를 비롯한 다양한 보호 기법을 반영하고 있기 때문에 동적 분석 자동화는 불가능하다. 따라서, ANTLR 를 이용한 정적 분석을 통해 코드를 모델링하고, 라이브러리에 따라 분석기를 개발하여 API 파싱을 자동화하는 방법에 대해 연구를 진행했다.

3. ANTLR4 를 이용한 정적 분석 자동화 방법

ANTLR4 는 파서 생성기 중 하나로, 구문 분석에 용이한 오픈소스이다. 이는 (그림 2)와 같이 렉서 (Lexer) 및 파서 (Parser)를 이용하여 지정한 문법 규칙에 따라 문법을 분석하고, 파스 트리를 제공한다.

```
grammar Expr;
prog:   (expr NEWLINE)* ;
expr:   expr ('*'|'/') expr
    |   expr ('+'|'-') expr
    |   INT
    |   '(' expr ')'
    ;
NEWLINE : [\r\n]+ ;
INT     : [0-9]+ ;
```

```
$ antlr4-parse Expr.g4 prog -gui
10+20*30
^D
$ antlr4 Expr.g4  # gen code
$ ls ExprParser.java
ExprParser.java
```

(그림 2) ANTLR4 파스 트리 생성 과정

이와 같이 생성된 파스 트리를 이용하여 코드를 원하는 방식으로 코드를 모델링하고, 모델링한 결과를 분석하는 분석기를 작성하면 HTTP 클라이언트에 대한 코드 분석 자동화가 가능하다.

정적 분석 자동화에 앞서, 안드로이드 APK 를 대상으로 JADX[7] 도구를 활용해 디컴파일을 시도하고, 결과를 파일 시스템에 저장해야 한다. 그리고, 디컴파일된 모든 코드를 분석하기 위해 (그림 3)과 같이 파서를 등록하고, 파서를 통해 분석한 코드를 목적에 맞게 저장하기 위해 (그림 4)와 같은 데이터 모델을 생성한다.

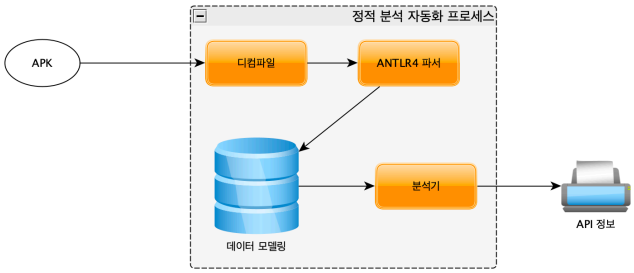
```
def parse(self, path) -> dict:
    code = open(path, 'r').read()
    input_stream = InputStream(code)
    lexer = JavaLexer(input_stream)
    token = CommonTokenStream(lexer)
    parser = JavaParser(token)
    walker = ParseTreeWalker()
    tree = parser.compilationUnit()
    listener = JavaInterface(parser)

    walker.walk(listener, tree)
    listener.models["file"] = path
    return listener.models
```

(그림 3) 파서를 통해 코드를 분석하기 위한 함수

```
class Pair:
    key: str
    value: Text
    def __init__(self, key, value):
        self.key = key
        self.value = value

class Annotation:
    name: str
    pair: list[Pair]
    value: list[Text]

class Param:
    annotation: list[Annotation]
    type: str
    name: str

class Function:
    annotation: list[Annotation]
    name: str
    returns: str
    param: list[Param]

class Result:
    package: str
    imports: list
    function: list[Function]
```

(그림 4) 분석한 코드를 저장하기 위한 데이터 모델

위와 같이 코드 파싱을 위한 작업을 마치면, 파싱을 위한 인터페이스를 작성해야 한다. 파서를 통해 분석을 완벽하게 하려면 클래스 및 인스턴스, 어노테이션, 변수 등을 모두 분석해야 한다. Retrofit 을 이용한 예시 코드를 살펴본 바와 같이 위와 같은 정보를 모델링하면 HTTP 메소드와 엔드포인트, 그리고 전달되는 파라미터와 반환 객체까지 모두 자동으로 분석이 가능하다. 이렇게 분석한 데이터를 기반으로 분석기를 작성하여 API 를 조합하여 최종적인 결과를 만들어낼 수 있다. (그림 4)는 입력값인 APK 를 시작으로, 정적 분석 자동화 프로세스를 거쳐 최종 결과까지 획득하는 프로세스이다.



(그림 4) 정적 분석 자동화 프로세스

4. 결론

본 논문에서는 오펜시브 보안 관점에서 ANTLR4 를 이용한 모바일 앱 정적 분석 자동화를 통해 API 서버에서 제공하는 엔드포인트와 파라미터 정보를 알아내는 연구를 진행하고, 방법을 제시하였다.

실제 상용 애플리케이션을 대상으로 실행한 결과, 수십개에 해당하는 API 엔드포인트와 메소드, 파라미터, 그리고 반환 객체까지 모두 자동으로 분석하는데에 성공하였다.

최근 들어, 제공하는 서비스가 많아짐에 따라 코드가 방대해지고 있으므로, 정적 분석 자동화 정확도를 높일 수 있도록 지속적인 연구가 필요하다.

참고문헌

[1] Amazon, What is an API? From https://aws.amazon.com/what-is/api/
[2] IBM, What is a REST API? from https://www.ibm.com/topics/rest-apis
[3] ANTLR, from https://www.antlr.org/
[4] Square, from https://square.github.io/retrofit/
[5] Square, from https://square.github.io/okhttp/
[6] Sciencedirect, from https://www.sciencedirect.com/topics/computer-science/dynamic-analysis
[7] Skylot, from https://github.com/skylot/jadx