

2020 춘계학술 발표대회

[논문집]

2020년 5월 22일(금) ~ 23일(토)

장소: 오션스위츠 제주호텔



주최 사단법인 한국인터넷정보학회 주관 제주대학교 SW중심대학사업단

후원 (주)카카오, (주)LGCNS, (주)아이티센, (주)콤텍정보통신, (주)한컴위드, 도서출판 길벗, (주)넷케이티아이, LG히다찌(주)

B-4 ☆ 건설의 객관적 평가 방법 및 시스템 개발	29
문초이, 유현중, 이지원, 김은빈, 김수복, 계슬아, 위예진(순천향대), 백유상(고려대), 이언석(순천향대)	
B-5 2차원 이미지(깊이 카메라)를 통한 실시간 환경에서 물체의 부피를 구하는 알고리즘 연구	31
배성현(성균관대)	
B-6 자동차 번호판 검출을 위한 인조 번호판 생성 시스템	33
김상준, 이유진, 박구만(서울과기대)	
B-7 요구 공학을 이용한 DAQ system의 비기능적 요구사항 추출	35
김경식, 이석원(아주대)	

C. 인터넷보안 (좌장 : 강지원 교수 (세종대))

C-1 안드로이드 디바이스 자원 사용량 분석을 이용한 크립토재킹 탐지 기술	41
이희승, 곽진(아주대)	
C-2 비정형 시나리오 기반 원자력시설의 사이버보안 훈련 평가 체계 강화 방법론 제안	43
김상우, 류진호(한국원자력통제기술원)	
C-3 CLAPP 기반 안드로이드 악성코드 분류 기법	45
정해선, 김득훈, 곽진(아주대)	
C-4 Just Walk Out 서비스에서의 안전한 소비자 그룹화 방법	47
이민경, 곽진(아주대)	
C-5 웹사이트 광고 배너 분석을 이용한 웹툰 침해 의심 사이트 탐지 기술	49
김의진, 최슬기, 곽진(아주대)	
C-6 지멘스 S7 PLC 취약점을 이용한 공격 방법 연구	51
김준원, 김장훈, 이민규, 서정택(순천향대)	
C-7 PLC 제어로직 변조를 통한 DoS 공격 및 대응방안 연구	53
김장훈, 김준원, 정다운, 서정택(순천향대)	

CLAPP 기반 안드로이드 악성코드 분류 기법

정해선*, 김득훈**, 박진***

**아주대학교 컴퓨터공학과 정보보호융합및보안연구실

***아주대학교 사이버보안학과

e-mail : *hsjeong.isaa@gmail.com, **dhkim.isaa@gmail.com, ***security@ajou.ac.kr

Android Malware Classification Method based on CLAPP

Hae-Seon Jeong*, Deuk-Hun Kim**, Jin Kwak***

**ISAA Lab., Department of Computer Engineering, Ajou University

***Department of Cyber Security, Ajou University

요 약

안드로이드 악성코드의 행위 기반 분류 시 수행 가능한 전체 행위를 분석하기 위해 실행 코드에 대하여 정적 분석을 수행한다. 정적 분석 기법 중에서도 CFG(Control Flow Graph)를 사용한 정적 분석 기법은 API 간의 호출 관계, 코드 분기 등의 실행 흐름을 분석할 수 있기 때문에 높은 빈도로 사용되고 있다. 이전에 실행된 코드가 다시 실행될 경우 루프가 생성되는데, 루프 반복 패턴 및 횟수는 코드 실행 흐름을 반영하는 주된 요소이다. 하지만 기존의 CFG 기반 안드로이드 악성코드 분류 기법들은 루프를 고려하지 않은 채 악성코드 행위 분석을 수행하고 있으며, 이에 따라 루프 반복 횟수 및 패턴이 악성 행위를 의미하는 경우 행위 분석의 정확도가 감소할 수 있다. 따라서 본 논문에서는 안드로이드 환경의 루프를 정의하고 루프 유형 등의 루프 정보를 분석하는 프레임워크인 CLAPP를 사용하여, 루프 정보를 고려한 안드로이드 악성코드 분류 기법을 제시한다.

1. 서 론

안드로이드 악성코드는 기존의 악성코드를 재활용하거나, 새로운 악성 기능을 추가하며 계속해서 발전하고 있다. 신·변종 악성코드에 대응할 때 악성코드 탐지에 그치지 않고 악성코드 군을 분류하는 것은 신·변종 악성코드의 행위 및 위험 수준을 이해하고 악성코드의 공격에 대응할 때 도움이 될 수 있다. 이때 안드로이드 악성코드가 수행하는 행위 정보를 사용하여 분류할 수 있는데, 동적 분석 기반 행위 정보를 사용할 경우 실행된 행위에 대해서만 분석하는 한계가 있다. 하지만 정적 분석 기반 행위 정보를 사용할 경우 악성코드가 수행할 수 있는 가능한 모든 행위에 대해 분석할 수 있다.

안드로이드 악성코드의 정적 분석 기반 분류 기법에는 크게 Opcode 기반 분류 기법과 CFG(Control Flow Graph) 기반 분류 기법이 존재한다. CFG 내에는 API 간의 호출 관계, 코드 분기 등의 실행 흐름 정보가 포함되어 있기 때문에 악성코드의 행위 기반 분류 시 CFG 기반 분류 기법은 Opcode 기반 분류 기법보다 높은 정확도를 보인다. 이때 코드가 이전에 실행된 코드로 분기될 경우 루프가 생성되는데, 루프 및 루프의 반복 횟수는 코드 실행 흐름을 반영하는 주된 요소이다. 하지만 기존의 CFG 기반 안드로이드 악성코드 분류 기법들은 루프를 고려하지 않고 악성코드 행위 분석을 수행하고 있으며, 이에 따라 루프 정보가 악성 행위를 의미하는 경우 행위 분석의 정확도가 감소할 수 있다.

따라서 본 논문에서는 2장 관련 연구에서 루프 정보를 분석·활용하기 위해 CLAPP(Characterizing Loops in Android Application) 루프 분석 프레임워크를 분석하고, 3장에서는 CLAPP 루프 정보를 활용한 악성코드 분류 기법을 제안하고, 4장에서 결론을 맺는다.

2. 관련 연구

2.1 CLAPP 루프 분석 프레임워크

CLAPP(Characterizing Loops in Android Application)는 안드로이드 애플리케이션 디컴파일 시 생성되는 기계어 실행 코드인 Smali 코드에서 CFG를 추출하고, 추출한 CFG 내 루프 및 루프의 유형을 정의한다[1].

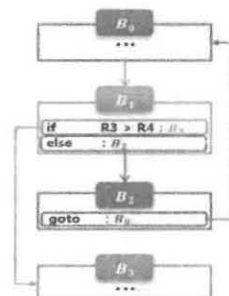
□ CFG 정의

CFG(Control Flow Graph)는 코드 분기가 일어나지 않는 명령어의 가장 작은 단위인 기본 블록으로 구성되어 있다. 기본 블록이 다른 블록으로 분기되면 블록 간에 선으로 연결된다.

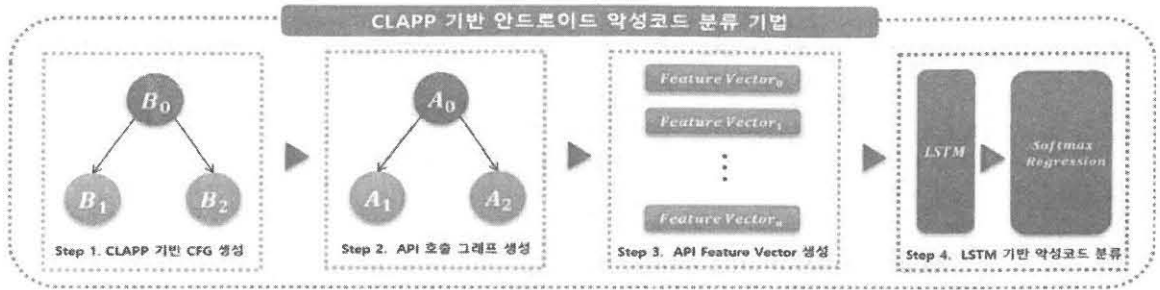
□ 루프 생성 및 종료

- 루프 생성 : 현재 실행되는 명령어가 이전에 실행된 기본 블록으로 분기되면 루프가 생성된다. 루프는 2개 이상의 기본 블록으로 구성된다. 그림 1과 같이 기본 블록 B_1 가 기본 블록 B_0 으로 분기되는 경우 B_0 , B_1 , B_2 로 구성된 루프가 생성된다.

- 루프 종료 : 루프 밖의 기본 블록으로 분기되는 경우, 루프가 종료된다. 그림 1과 같이 루프 내 기본 블록 B_1 이 루프 밖의 기본 블록 B_3 으로 분기되는 경우 루프는 종료된다.



(그림 1) 루프 생성 및 종료



(그림 2) CLAPP 기반 안드로이드 악성코드 분류 기법

□ 루프 유형

루프가 정의되면 루프 종료 지점과 루프 반복 횟수 관련 레지스터값 변화유형을 통해 루프 유형을 정의한다.

- 루프 종료 지점에 따른 유형 : 루프 종료 지점이 존재하지 않는 경우 루프 유형은 '무한(Infinte)'으로, 루프 종료 지점이 1개 이상 존재하는 경우, '유한(Finite)'으로 정의한다.

- 레지스터값 변화유형 : '유한(Finite)' 루프의 경우 루프 반복 횟수를 결정하는 레지스터 값의 변화유형에 따라 루프 유형을 추가로 정의할 수 있다. 루프 반복 횟수의 유형으로는 '고정(Fixed)', '증가(Increasing)', '감소(Decreasing)', '제한적(Bounded)'과 비트워크 입력으로 결정되어 알 수 없는 경우인 '알 수 없음(Unknown)'으로 총 5개 유형으로 정의한다.

3. CLAPP 기반 안드로이드 악성코드 분류 기법

본 장에서는 그림 2와 같은 CLAPP 기반 루프 정보를 활용한 안드로이드 악성코드 분류 기법을 제시한다. 악성코드 군을 분류하기 위해 CLAPP의 루프 정보를 활용하여 Feature Vector(128 bytes)를 생성하고, 이를 입력으로 하는 분류 기법을 제시한다. 분류 기법의 세부 과정은 다음과 같다.

Step 1. CLAPP 기반 CFG 생성

사용자는 애플리케이션의 UI 인터페이스인 Activity를 통해 애플리케이션을 조작 및 사용할 수 있다. 이때 Activity 사용 시 onCreate() API가 가장 먼저 호출되기 때문에 본 논문에서는 onCreate() API 호출 지점을 CFG 생성 지점으로 정의한다. onCreate() API가 속한 기본 블록 B_0 의 인덱스는 0으로, 실행 순서에 따라 기본 블록의 인덱스가 증가한다. 또한, CFG 생성 시 CLAPP 기반으로 해당 기본 블록이 루프에 포함 여부를 저장하고, 기본 블록이 루프에 포함되는 경우 루프 시작 블록의 인덱스, 루프 유형이 함께 저장된다.

Step 2. API 호출 그래프 생성

악성코드가 특정 행위를 수행하기 위해 API를 호출하기 때문에 API 호출 정보는 악성코드의 행위를 반영하는 중요 정보이다. 이에 따라 앞선 과정에서 생성된 CFG에서 API 호출이 수행되는 기본 블록만을 연결하여 API 호출 그래프를 생성한다. API 호출 그래프는 API 블록 $A_i(i=0,1,\dots)$ 로 구성되어 있으며, 각각의 API 블록에는 API 블록이 루프에 포함되어있는지 여부가 저장된다. 해당 API 블록이 루프에 포함되는 경우, API 블록이 속해있는 루프의 시작점에 해당하는 API 블록의 인덱스, 루프 유형이 함께 저장된다.

Step 3. API Feature Vector 생성

API 블록에 저장된 정보를 기반으로 고정된 길이의 Feature Vector를 생성한다. API Feature Vector의 구성요소는 고정된 길이의 문자열을 갖게 하는 Feature Hashing을 통해 생성된다.

(표 1) API Feature Vector의 세부 구조

API Feature 구성요소	세부사항
API명 (16 bytes)	- API명에 Feature Hashing을 수행한 값
루프 시작 지점의 API명 (16 bytes)	- 루프가 시작되는 블록의 API명에 Feature Hashing을 수행한 값 (루프가 아닐 경우 0의 값을 가짐)
루프 유형 (32 bytes)	- 루프 종료 지점에 따른 유형 (16 bytes) : 'Infinte', 'Finite' 중 해당하는 유형의 단어에 Feature Hashing을 수행한 값 - 레지스터값 변화유형 (16 bytes) : 'Fixed', 'Increasing', 'Decreasing', 'Bounded', 'Unknown' 중 해당하는 유형의 단어에 Feature Hashing을 수행한 값 (무한 루프일 경우 0의 값을 가짐)
루프에 포함되는 API명 (64 bytes)	- 루프에 포함되는 모든 API 명을 이어 붙인 문자열에 Feature Hashing을 수행한 값 (루프가 아닐 경우 0의 값을 가짐)

Step 4. LSTM 기반 악성코드 분류

LSTM(Long Short-Term Memory)은 장기 메모리 학습에 특화된 딥러닝 모델로, 가변적이며 길이가 긴 악성코드의 API 호출 순서를 학습하기에 적합한 딥러닝 모델이다[2]. 이에 따라 본 논문에서는 API Feature Vector 값을 LSTM의 입력으로 사용하여, LSTM을 통해 악성코드의 API 호출에 대하여 학습하고, 학습된 결과에 대해 다중 클래스 분류 기법인 Softmax Regression을 사용하여 악성코드 군들에 대한 분류를 수행하는 기법을 제안한다.

4. 결론

본 논문에서는 CFG 내 루프 정보를 포함하여 악성코드의 행위 정보로부터 고정된 길이의 Feature Vector를 형성하고, 이를 입력으로 받는 안드로이드 악성코드 분류 기법을 제안하였다. 해당 분류 기법은 루프 내에 악성코드의 행위 정보가 포함된 경우에도 악성코드를 분류할 수 있으며, 분류의 각 세부 단계를 분류 목적에 맞게 수정할 경우 다양한 분류 모델에 적용될 수 있다.

향후에는 본 논문이 제안하는 분류 기법의 성능을 측정하여 제안 기법의 성능을 검증하고, 결과를 분석할 예정이다.

참고 문헌

- [1] Yanick Fratanonio, Aravind Machiry, Antonio Bianchi, Christopher Kruegel, Giovanni Vigna, "CLAPP: Characterizing Loops in Android Applications", ESEC/FSE, pp.687-697, 2015.
- [2] Eduardo de O. Andrade, José Viterbo, Flavia Cristina Bernardini, Joris Guérin, Joris Guérin, "A Model Based on LSTM Neural Networks to Identify Five Different Types of Malware", Procedia Computer Science, Vol.150, pp.182-191, 2019.