Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.1, Feb. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.1.187

API 특성 정보기반 악성 애플리케이션 식별 기법*

조 태 주,[1†] 김 현 기,[1] 이 정 환,[1] 정 문 규,[2] 이 정 현[1‡]
[1]숭실대학교, [2]삼성전자

A Scheme for Identifying Malicious Applications Based on API Characteristics*

Taejoo Cho,[1†] Hyunki Kim,[1] Junghwan Lee,[1] Moongyu Jung,[2] Jeong Hyun Yi[1‡]
[1]Soongsil University, [2]Samsung Electronics

요 약

안드로이드 애플리케이션은 악성코드를 삽입한 후 재서명하여 배포하는 리패키징 공격에 취약하다. 이러한 공격을 통해 사용자의 사생활 정보나 개인정보 유출 등의 피해가 자주 발생하고 있는 실정이다. 모든 안드로이드 애플리케이션은 사용자가 직접 작성한 메소드와 API로 구성된다. 이중 플랫폼의 리소스에 접근하며 실제 애플리케이션의 기능적인 특징을 나타내는 것은 API이고, 사용자가 작성한 메소드 역시 API를 이용하며 기능적 특징을 나타낸다. 본 논문에서는 악성 애플리케이션이 주로 활용하는 민감한 API들을 분석 대상으로 하여 악성애플리케이션이 어떤 행위를 하고, 어떤 API 를 사용하는지 사전에 식별할 수 있는 분석 기법을 제안한다. 사용하는 API를 토대로 API의 특성정보를 기반으로 나이브 베이즈 분류 기법을 적용하여 비슷한 기능을 하는 API에 대해 기계 학습하도록 한다. 이렇게 학습된 결과를 토대로 악성 애플리케이션이 주로 사용하는 API를 분류하고, 애플리케이션의 악성 위험 정도에 대한 정량적 판단 기준을 제시한다. 따라서, 제안 기법은 모바일 애플리케이션의 취약점 정도를 정량적으로 제시해 줌으로써 모바일 애플리케이션 개발자들이 앱 보안성을 사전에 파악하는데 많은 기여를 할 수 있을 것으로 기대된다.

ABSTRACT

Android applications are inherently vulnerable to a repackaging attack such that malicious codes are easily inserted into an application and then resigned by the attacker. These days, it occurs often that such private or individual information is leaked. In principle, all Android applications are composed of user defined methods and APIs. As well as accessing to resources on platform, APIs play a role as a practical functional feature, and user defined methods play a role as a feature by using APIs. In this paper we propose a scheme to analyze sensitive APIs mostly used in malicious applications in terms of how malicious applications operate and which API they use. Based on the characteristics of target APIs, we accumulate the knowledge on such APIs using a machine learning scheme based on Naive Bayes algorithm. Resulting from the learned results, we are able to provide fine-grained numeric score on the degree of vulnerabilities of mobile applications. In doing so, we expect the proposed scheme will help mobile application developers identify the security level of applications in advance.

Keywords: Android Malware, Android Repackaging Attack, API Classification, Naive Bayes Classification

Received(11. 02. 2015), Accepted(11. 10. 2015)
* 본 연구는 삼성전자 소프트웨어센터의 연구비 지원으로 수행하였습니다.

† 주저자, taejoo90@gmail.com
‡ 교신저자, jhyi@ssu.ac.kr(corresponding author)

I. 서 론

현대사회에서 모바일 디바이스의 사용률은 계속해서 증가 하고 있으며, 안드로이드 기반의 플랫폼은 모바일 디바이스 시장에서 가장 큰 점유율을 가지고 있다. 이런 환경 속에서 악성 애플리케이션 제작자들은 수많은 공격을 시도 하고 있으며, 이는 원본의 애플리케이션에 악성 코드를 삽입하고 리패키징[6] 후 자신의 키로 사인 후 배포하는 등의 행위가 어렵지 않기[1][2][3] 때문이다. 이러한 공격을 통하여 안드로이드 플랫폼 사용자는 자신도 모르는 사이에 사생활 정보나 개인정보 유출 등의 피해를 입을 수 있다. 이와 같은 피해를 방지하기 위하여 애플리케이션 개발자들은 난독화 등의 기법으로 애플리케이션을 보호하려고 하지만, 난독화[4][5] 적용 후에도 악성행위를 시도하는 것이 가능하며, 완벽한 보호를 수행할 수 없는 것이 사실이다.

애플리케이션은 사용자가 기능을 수행하기 위해 직접 작성한 메소드와 운영체제나 프로그램 언어가 제공하는 기능을 제어할 수 있도록 만든 인터페이스인 API로 구성된다. 이중 플랫폼의 리소스에 접근하며 실제 애플리케이션의 기능적인 특징을 나타내는것은 API이고, 사용자가 작성한 메소드 역시 API를 이용하며 기능적 특징을 나타낸다.

본 논문에서는 API를 악성 애플리케이션 관점에서 분석한다. 악성애플리케이션이 어떤 행위를 하고, 어떤 API 를 사용하는지 분석하고 사용하는 API를 토대로 API의 특성정보를 이용하여, 머신러닝 알고리즘을 통해 비슷한 기능을 하는 API를 학습하고 분류하고자 한다.

해당 과정을 통해 악성 애플리케이션이 주로 사용하는 API를 분류 할 수 있으며, 이를 이용하여 애플리케이션 분석의 입장에서 애플리케이션의 악성의 위험에 있는지에 대해서 판단하는데 중요한 정보를 제공 할 수 있다. 또한 이를 활용하면 이러한 부류의 API 검증을 통해 안전한 플랫폼 설계 또한 기대할 수 있고, 애플리케이션 배포과정에서도 해당부류의 API 디스크립션 정보를 활용하여 개발자가 제공하는 설명 이외의 정보 또한 제공 할 수 있다.

본 논문의 구성은 이렇다. 2장에서는 관련 연구에 대해 다루고, 3장에서는 모바일 악성 애플리케이션 분석을 통해 기능적 특징과 어떤 행위를 하며 또 어떤 API를 사용하는지 분석한다. 4장에서는 나이브 베이즈 분류법을 이용하여 악성 API를 분류한다. 5장에서는 구현 및 실험을 통하여 실제 구현한 API 분류기를 통해 도출된 결과로 실험 결과를 보이며, 6장에서는 결론을 맺는다.

II. 관련 연구

2.1 나이브 베이즈 분류

나이브 베이즈 분류[9][10]는 나이브 베이즈 정리를 근거로 각 특성들의 확률을 파라미터로 하여 부류를 결정 하는 기법이다. 나이브 베이즈 분류에서는 모든 특성 값은 서로 독립임을 가정하며, 파라미터의 추정은 최대 우도 방법(maximum likelihood estimation)을 사용한다.

나이브 베이즈 분류의 공식은 다음과 같다.

$$p(c|x_1,x_2,x_3,...,x_n) = \frac{1}{Z}\prod_{i=1}^{n} p(x_i|c)p(c) \quad (1)$$

$$posterior = p(c|x_1,x_2,x_3,...,x_n) \quad (2)$$
$$evidence = Z$$
$$likelihood = \prod_{i=1}^{n} p(x_i|c)$$
$$prior = p(c)$$

$$Class = \arg max\ p(x_1,x_2,x_3,...,x_n|c)p(c) \quad (3)$$
$$= \arg max \prod_{i=1}^{n} p(x_i|c)p(c)$$

(1)의 공식은 나이브 베이즈 정리에 의해 특정 n개의 특성(x)이 있을 때 어떤 부류(c)일 확률을 공식으로 나타낸 것이다.

(2)에서는 사후 확률(posterior), 관찰 값(evidence), 우도(likelihood), 사전 확률(prior)을 나타내고 있으며 (1)의 공식의 각각이 무엇을 의미하는지 나타내고 있다.

(3)의 공식은 부류의 결과 선택을 나타낸 공식이며, 나이브 베이즈 분류기는 최대 우도 방법을 사용한다고 하였으므로 우도(likelihood) 값이 최대가 되는 부류를 결과 값으로 선택한다.

2.2 모바일 악성 코드의 종류

악성 코드는 행위에 따라 바이러스, 트로이목마,

스파이웨어, 랜섬웨어, 애드웨어가 있다[17].

2.2.1 바이러스/웜

바이러스는 스스로를 복제하여 악의적인 행동을 하는 악성 코드를 의미한다. 스스로를 복제하고 파일들을 감염시키지만 다른 컴퓨터로 전파되지는 않는다. 바이러스와 기능적으로 유사하지만 다른 컴퓨터로 전파 될 경우는 웜이라고 한다.

2.2.2 랜섬웨어

랜섬웨어는 ransom(인질)와 ware(제품)를 합친 말로, 동작을 방해하며 중요 데이터들에 접근할 수 없게 암호화를 걸고 해당 데이터를 인질로 삼아 돈을 요구하는 악성코드이다.

2.2.3 애드웨어

애드웨어는 프로그램 실행 중 광고를 보여주고 이를 이용하여 돈을 버는 프로그램을 의미하지만 악성코드로서의 애드웨어는 사용자의 몰래 설치되어 광고창을 띄우는 프로그램을 의미한다. 대다수의 애드웨어는 돈을 목적으로 만들어지기 때문에 유저의 불편은 고려하지 않고 광고를 노출한다.

2.2.4 트로이 목마

트로이 목마는 트로이 전쟁의 트로이 목마에서 유래되었으며, 악성행위를 숨기고 정상적인 프로그램으로 위장하는 악성 코드의 한 종류이다. 트로이 목마는 핸드폰을 원격으로 제어 할 수 있도록 백도어를 생성하여 악성행위를 시도할 수 있다. 이를 통해 사용자의 단말기를 원격제어 하거나 개인정보 유출시키는 등의 문제를 일으킬 수 있다.

2.2.5 스파이웨어

스파이웨어는 이름 그대로 악의적인 스파이처럼 행동하는 악성 코드를 의미한다. 스파이웨어는 사용자의 동의 없이 몰래 설치되며 개인정보를 수집 하고 제작자에게 전송하며, 사용자의 행동을 감시한다.

III. 모바일 악성 애플리케이션 분석

3.1 모바일 악성 애플리케이션 수집 및 분석 방법

본 논문에서는 모바일 악성 애플리케이션 API 분류를 위해 악성애플리케이션의 행위 및 주로 사용되어 지고 있는 API를 분석하였다. 분석한 애플리케이션은 총 194개로 해당 정보는 Contagio[13]와 VirusShare[14]의 데이터를 기반으로 구성된다. 데이터 수집 후 악성 애플리케이션을 분석하여, APKtool[15]과 같은 도구를 통해 정적 관점에서 어떤 API를 사용하는지 파악하였다. 또한 동적 관점에서 악성코드의 실제 동작을 분석하고, Anubis[16]와 같은 샌드박스 기반 방법을 이용하여 애플리케이션의 행위 분석을 시도하였다.

3.2 모바일 악성 애플리케이션 분석 및 분류

수집된 악성 애플리케이션은 랜섬웨어, 애드웨어, 트로이목마, 스파이웨어로 구성 된다. Table 1.은 분석결과를 이용하여 각각의 부류에서 사용된 주요 API를 정리한 결과이다.

Table 1. Classification of malicious application set and used API

Class	Used API
ransomware	acquire() release() newWakeLock()
adware	openConnection() getDeviceId() getLastKnownLocation()
trojan	getSimSerialNumber() getActiveNetworkInfo() getLine1Number()
spyware	requestLocationUpdates() sendTextMessage() getDeviceId()

3.2.1 랜섬웨어 분석

랜섬웨어는 스마트폰 사용자의 동작을 방해하고 중요 파일을 암호화하는 등의 행위를 하여 데이터 복호화 조건으로 금전등을 요구하는 악성 애플리케이션을 의미한다. 본 논문에서 수집한 애플리케이션 데

이터를 기반으로 분석 한 결과 acquire(), release(), newWakeLock()와 같은 API 가 주로 사용되었다. 각각의 API 는 모두 android.os.PowerManager 클래스에 속하며, 랜섬웨어는 이와 같은 함수를 사용하여 애플리케이션의 화면이 종료되는 것을 방해하고 금전을 요구한다.

3.2.2 애드웨어 분석

애드웨어는 사용자에게 광고와 같은 특정 홈페이지를 노출함으로써 동작을 방해하고, 개인정보를 수집하고 유출하는 등의 역할을 한다. 본 논문에서 수집한 데이터를 기반으로 분석한 결과 openConnection(), getDeviceID(), getLastKnownLocation() 과 같은 API를 사용하고 있다. openConnection()은 getjava.net.URL 클래스에 속해있는데, 해당 API를 이용하여 서버에 정보를 획득한 다음 특정 사이트 또는 게임을 광고한다. getLastKnownLocation() API는 android.location.LocationManager 클래스에 속해있으며, 해당 함수를 사용하여 위치 정보를 파악하고 위치에 맞는 광고를 노출한다. 또한, android.telephony.TelephonyManager 클래스의 getDeviceId()를 사용하여 사용자의 데이터를 수집하여 유출한다.

3.2.3 트로이목마 분석

트로이 목마는 정상적인 애플리케이션처럼 위장하고 사용자에게 자신의 행위를 노출하지 않으며 악성행위를 시도한다. 대부분의 경우 트로이 목마는 핸드폰을 원격으로 제어 할 수 있도록 백도어를 생성하여 악성행위를 시도한다. 해당 부류에서는 android.telephony.TelephonyManager 클래스의 API를 주로 사용하며, 해당 클래스에 속하는 getSimSerialNumber(), getLine1Number()와 같은 함수는 주로 개인정보 유출에 사용된다. 또한 android.net.ConnectivityManager 클래스에 속하는 getActiveNetworkInfo() 함수는 데이터를 전송하기 위해 현재 네트워크 상태를 검사한다.

3.2.4 스파이웨어 분석

스파이웨어는 개인정보를 수집하여 sms나 네트워크를 통해 공격자에게 전송한다. 해당 부류에서는 주로 requestLocationUpdates(), sendTextMessage()의 API를 사용하고 있다. android.location.LocationManager 클래스의 requestLocationUpdates()를 사용하여 사용자의 위치를 파악한다. android.telephony.TelephonyManager 클래스의 getDeviceId()를 사용하여 개인정보를 유출하고 android.telephony.gsm.SmsManager 클래스의 sendTextmessage()를 이용하여 데이터를 sms를 통하여 전송한다.

IV. 나이브 베이즈 분류법을 이용한 악성 API의 분류

4.1 정적 코드 추출법

나이브 베이즈 분류법을 이용한 악성 API의 분류를 위해 애플리케이션의 DEX (Dalvik Executable)[7]로부터 코드를 추출한다. 애플리케이션의 실제적인 코드를 가지고 있는 것은 DEX 파일이며 DEX 파일은 Table 2.과 같은 필드 값으로 구성된다.

이중 data 영역에서 실제적인 코드를 추출하는 과정은 다음 Fig. 1. 와 같다.

Table 2. Composition of DEX field

Field name	Composition
header	The header
string_ids	String identifiers list. All strings used by a file.
type_ids	Type identifiers list. Identifiers for classes, arrays, and primitive types.
proto_ids	Method prototype identifiers list, Prototype Information.
field_ids	Field identifiers list. Field Information.
method_ids	Method Identifiers List, Method Information.
class_def	Class definitions list. Class Information.
data	Data area containing all the support data for the tables listed above.
link_data	Statically linked files

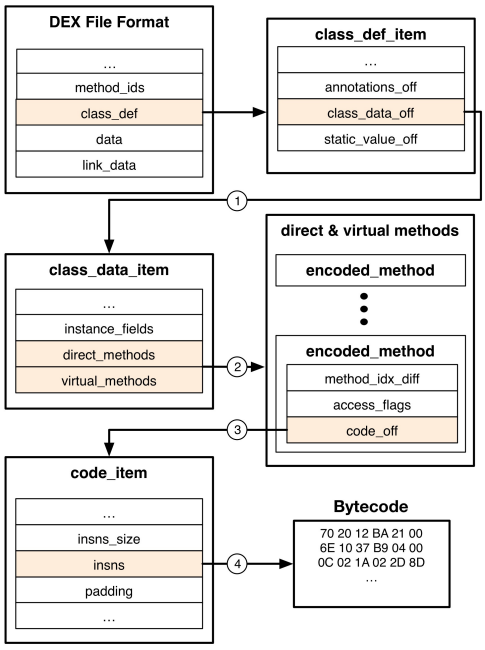


Fig. 1. Process of extracting bytecodes in DEX file format

1) class_def_item → class_data_item

class_def_item 은 Dex 파일 구조에서 class_def 필드에 존재하고 있고 클래스 정보를 가지고 있다. class_def_item 내의 class_data_off 필드는 class_data_item의 위치를 가리키고 있다.

2) class_data_item → encoded_method

class_data_item은 필드와 메소드의 정보를 가진다. class_data_item 내부의 메소드 정보를 찾기 위하여 메소드 필드를 이용해야하는데 메소드 필드는 encoded_method로 구성되어 있다.

3) encoded_method → code_item

encoded_method 내부에는 code_off 필드가 존재한다. 해당 필드를 이용하여 code_item의 위치를 찾을 수 있다.

4) code_item

메소드의 명령어 정보가 존재하는 곳으로 insns 필드에는 실질적인 바이트코드[12]의 배열을 가지고 있다. 해당 필드의 정보를 활용하면 정적 방법으로 DEX 파일 구조 내의 코드 및 호출되는 API 값을 추출할 수 있다.

4.2 악성 애플리케이션 API 분류법

악성애플리케이션의 API 분류를 위해 나이브 베이즈 분류법을 이용한다. Table 3.은 나이브 베이즈 분류법을 이용하기 위하여 각 부류일 때, 즉 sensitive API 혹은 normal API 일 때의 조건부 확률을 구한 것이다. 해당 조건부 확률은 악성 애플리케이션의 특성 값으로부터 추출된 확률이며, senstivie라는 의미는 악성 애플리케이션이 주로 사용하는 API와 같은 부류이며, 민감한 API 될 수 있음을 의미한다.

조건부확률을 구하는 특성(x) 값은 API 특성정보(API가 속한 패키지, 클래스, API의 이름, 파라미터의 형태, 리턴 값, API의 설명 등)값을 이용한다. 그 후 해당 조건부 확률을 이용하여 API 들을 분류한다.

API 분류는 API 특성정보 로딩, 특성정보 전처리, 벡터 구성, 검증의 과정을 거친다. 각 과정에 대한 설명은 다음과 같다.

Table 3. Conditional probability for each word

word	$p(x_i \mid SensitiveAPI)$	$p(x_i \mid NormalAPI)$
get	0.8	0.1
returns	0.7	0.2
phone	0.7	0.2
number	0.8	0.1
telephony	0.7	0.2
string	0.3	0.5
line	0.2	0.6
android	0.2	0.6
manager	0.3	0.7
…	…	…

4.2.1 API 특성정보 로딩

API 부류를 결정할 API를 추출하기 위해 API의 특성정보를 로딩 한다. 예제로 getLine1Number API를 선정한다. getLine1Number API의 특성정보는 다음 Table 4. 와 같이 나타낼 수 있다.

Table 4.와 같이 안드로이드 레퍼런스 사이트[8]에서 추출한 getLine1Number의 특성정보는 API name, 해당 API 가 속한 패키지, 해당 API가 속

Table 4. Properties of getLine1Number

API name	getLine1Number
package	android.Telephony
class	TelephonyManager
description	Returns the phone number string for line 1
...	

한 클래스, 설명으로 나뉜다. getLine1Number API가 속한 패키지는 android.Telephony 패키지이며, 속한 클래스는 TelephonyManager이다. 해당 API의 설명은 "Returns the phone number string for line 1" 이라는 값을 가지고 있는데 내용과 같이 실제 해당 API는 전화번호부의 첫 번째 전화번호 , 즉 자신의 번호를 획득하는 역할을 한다.

4.2.2 특성정보 전처리

기법 적용에 앞서 전처리 과정이 필요한데, API 이름인 getLine1Number와 같이 복수의 단어로 구성된 특성일 경우, 각각의 단어로 나누어 준다. 여기서는 getLine1Number 가 get, line, number 로 나뉠 수 있으며 숫자인 1은 제외시킨다. API가 속한 패키지와 클래스도 마찬가지로 각각의 단어로 나누어주는 과정이 필요하다. API 설명의 경우 이 예제에서는 "Returns the phone number string for line 1" 이라는 값을 가지고 있는데, 관사나 숫자와 같은 부분을 제외 하고 단어 각각을 나누어준다.

4.2.3 벡터 구성

전처리 과정을 마치고 특성 값의 중복이 없도록 벡터를 구성한다. getLine1Number API 특성정보를 통해 구성된 벡터는 아래와 같이 표현 할 수 있다.

$$\{x_1, x_2, x_3, \cdots, x_9\}$$
$$= \{get,\ line,\ number,\ android,\ telephony,\ manager,\ returns,\ phone,\ string\}$$

4.2.4 검증

위에서 구성된 벡터를 기반으로 나이브 베이즈 분류를 수행한다. 나이브 베이즈 분류를 수행하기 위해 학습 대상의 벡터를 선정하며, 학습대상 벡터를 구성하는 각각의 특성에 대한 조건부 확률을 구한다. Table 3. 는 사전에 계산된 API 단어 별 조건부 확률의 예시를 나타낸 것이다.

getLine1Number API 분류를 시작한다. 나이브 베이즈 공식에 의하여 민감한 API 부류, 일반 API 부류에 속할 확률을 구한다.

※ $p(sensitiveAPI) = p(NormalAPI) = 0.5$로 가정

1) getLine1Number API 가 Sensitive API에 속할 확률

$$p(SenstivieAPI\ |X)$$
$$= \prod p(x_i|SensitiveAPI)p(SensitiveAPI)$$
$$= 0.8*0.2*0.8*0.2*0.7*0.3*0.7*0.7*0.3*0.5$$
$$= 0.000395136$$

2) getLine1Number API 가 Normal API에 속할 확률

$$p(NormalAPI\ |X)$$
$$= \prod p(x_i|NormalAPI)p(NormalAPI)$$
$$= 0.1*0.6*0.1*0.6*0.2*0.7*0.2*0.2*0.5*0.5$$
$$= 0.00000504$$

나이브 베이즈 분류법은 최대 우도 방법을 사용하고, 민감한 API에 속할 확률 (0.000395136)이 일반 API에 속할 확률 (0.00000504) 보다 크므로 Sensitive API의 부류로 선택한다. 즉 getLine1Number API는 Sensitive API 부류로 분류 된다.

4.3 악성 애플리케이션 기반 API 분류기 설계

앞서 설명한 정적 코드 추출법과, 악성 API 분류법을 적용하여 악성 애플리케이션 기반 API 분류기를 설계한다.

Fig. 2.와 같이 API 분류기는 APK 전처리기 (APK pre-processor), 데이터 구조 생성기(data structure generator), 코드 출력기 (code printer), API 분류기(API classifier), API 학습기 (API learner), API 검증기(API verifier)로 구성되는데, 각 모듈에 대한 설명은 다음과 같다.

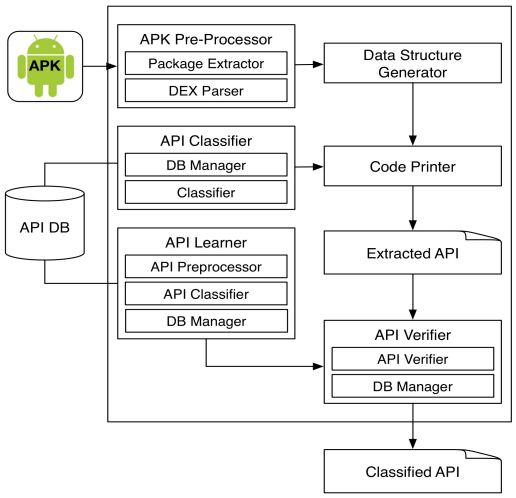


Fig. 2. Module structure of API Classifier API

4.3.1 API 전처리기

API 전처리기는 패키지 추출기(package extractor), 덱스 파서(DEX parser)로 구성된다. 패키지 추출기는 추출대상의 애플리케이션 파일을 입력받는다. 입력 후 애플리케이션을 unzip 하고, unzip 한 디랙토리 내의 DEX 파일을 추출한다. 추출된 DEX 파일을 이용하여 덱스 파서는 덱스 파일을 파싱한다. 파싱된 DEX 파일은 트리형태로 구성되며 트리는 패키지, 클래스, 메소드의 각 정보로 구성되어 있다.

4.3.2 데이터 구조 생성기

데이터 구조 생성기는 앞서 전처리 모듈로부터 생성된 트리를 파싱하여 데이터구조를 생성한다. 데이터 구조는 리스트의 형태로 되어있으며 추출된 특정 API의 패키지정보, 클래스정보, 메소드 정보, 파라미터, 리턴 값 등의 API의 특성을 나타낼 수 있는 정보로 구성되어 있다.

4.3.3 API 분류기

API 분류기는 API의 특성 정보를 저장하고 있는 API 데이터베이스(API DB)를 기반으로 입력받은 메소드가 API 인지를 판단한다. 데이터베이스 관리자(DB manager)는 API 데이터베이스를 조회 하는 등의 관리 역할을 하며, 분류자(classifier)는 데이터베이스 관리자로부터 조회된 API 정보와 분류를 원하는 API정보를 매칭하는 역할을 한다.

4.3.4 코드 출력기

데이터 구조 생성모듈로부터 생성된 데이터 구조 리스트 중 프레임워크 API 분류기로부터 분류한 코드를 출력하는 역할을 한다.

4.3.5 API 학습기

API 학습기는 나이브 베이즈 분류법을 통하여 프레임워크 API를 분류한다. API 학습기는 API 전처리기(API preprocessor), API 분류기(API classifier), 데이터베이스 관리자(DB manager)로 구성된다. API 전처리기는 API 특성 정보 중 필요 없는 정보나 의미 없는 정보를 제외시키며 학습하기 좋은 형태로 가공하여 벡터로 생성한다. 벡터의 형태로 생성된 API 정보는 API 분류자가 나이브 베이즈 분류를 이용하여 sensitive API 부류와 normal API 부류로 분류한다. 데이터베이스 관리자는 앞서 분류된 정보를 기반으로 API 데이터베이스 내에 데이터를 생성한다.

4.3.6 API 검증기

API 검증기는 민감한 API 데이터베이스(sensitive API DB)를 이용하여 입력받은 API가 sensitive API 인지 normal API 인지 구분한다. API 검증기는 API 검증기와 데이터베이스 관리자로 나뉘는데, API 검증기는 API를 입력받으며 부류의 결과를 전달하는 역할을 한다. 데이터베이스 관리자는 API 검증기가 민감한 API 데이터베이스로부터 데이터를 조회할 때 중간자의 역할을 한다.

V. 구현 및 실험

5.1 제안 기법 구현

본 논문에서 제안하는 악성 애플리케이션 기반 API 분류방법을 검증하기 위하여 제안 기법을 구현한다. 개발환경으로 Windows 7 운영체제를 사용하였으며, 개발언어로는 java, python을 사용하였다.

5.1.1 정적 코드 추출기법 구현

APK 상에서 정적으로 추출된 API 정보를 활용하기 위해 API 분류기 중 정적 코드 추출기법을 구현한다. 해당 기법을 구현하기 위해, 안드로이드 애플리케이션 패키지 파일을 unzip 하고, AsmDex[11] 라이브러리를 사용하여 DEX 파일을 파싱한다. 파싱 후 파싱된 코드 중 invoke 되는 코드, 즉 메소드 호출 관련 명령어만을 추출한다. Fig. 3.의 parseDexFile 함수는 실제 DEX 파일을 로딩하고 로딩 된 DEX 파일 상에서 invoke 명령어를 추출하는 부분이다.

DexParser.java

```
public void parseDexFile() {
    try{
      FileWriter fw = new FileWriter(new File("output.txt"));
      String caller;
       FileNodeDex fn = new FileNodeDex(null, unZipDir+"\\classes.dex");
       for(ClassNode cn : fn.root.classes){
          caller = cn.name;
          for(MethodNode mn : cn.methods){
            AbstractInsnNode ain = mn.instructions.getFirst();
            while(ain!=null){
              If(ain instanceof MethodInsnNode){
                 fw.append("caller:" + caller + "\n");
                 fw.append("owner:"+ ((MethodInsnNode) ain).owner + "\n");
                 fw.append("name:"+ ((MethodInsnNode) ain).name + "\n");
                 fw.append("desc:"+ ((MethodInsnNode) ain).desc + "\n\n");
              }
              ain = ain.getNext();
            }
          }
    }
    fw.close();
   }catch (Exception e) {
    e.printStackTrace();
 }
}
```

Fig. 3. Implementation of static extraction method

5.1.2 API 분류 기법 구현

악성 애플리케이션의 API 분류를 위해 나이브 베이즈 분류법을 이용한 API 분류기법을 구현한다. 해당 기법을 구현하기 위해, python 의 nltk 라이브러리 내부에 구현되어있는 naivebayes 모듈을 사용한다. naivebayes 모듈을 이용하여 4.2의 악성 애플리케이션 API 분류법을 구현한다. Fig. 4.의 sensitivit.py는 악성 애플리케이션의 API 특성정보를 학습대상으로 하고 결과로 비슷한 부류의 API를 분류한다.

sensitivity.py

```
...

malware_api = [malware API vector1,malware API vector2, ...]
normal_api = [normal API vector1,normal API vector2, ...]

for (words, sentiment) in malware_api + normal_api:
    api.append((words_filtered, sentiment))

word_features = getWordsFeatures(getWordsInAPI(api))
training_set = nltk.classify.apply_features(extractFeatures, api)
classifier = nltk.NaiveBayesClassifier.train(training_set)

f = open('testResult.csv','w')
target = [target API vector1, target API vector2,...]
for ins in target
    f.write(apiName+","+ classifier.classify(sen.extractFeatures(ins.split())))

...
```

Fig. 4. Implementation of API classification method using naive bayes classifier

5.2 실험 결과

악성 애플리케이션 상 추출된 API를 이용하여 분류 후 도출된 결과는 Table 5. 와 같다.

Table 5.에서 볼 수 있듯이 도출된 결과는 위치 관련, 계정관련, SMS관련, 알림관련, 네트워크 관련 API로 구성됨을 알 수 있다. 이와 같은 API는 위에서 분석한 랜섬웨어, 애드웨어, 트로이목마, 스파이웨어 등에 이용될 수 있는 API 이며, 악성 행위를 할 수 있는 소지가 있다.

또한 분류된 API를 기반으로 통계를 낸 결과, 전체 API 부류 중에 약 14.1%를 차지하는 것을 알 수 있었다.

Table 5. Predicted API list and description

API Name	Description
getAltitude	Computes the Altitude in meters from the atmospheric pressure and the pressure at sea level.
getAccounts	Lists all accounts of any type registered on the device.
getAccountsByType	Lists all accounts of a particular type.
getMessagesFromIntent	Read the PDUs out of an SMS_RECEIVED_ACTION or a DATA_SMS_RECEIVED_ACTION intent.
getId	The id supplied to notify(int Notification).
getAllNetworkInfo	Returns connection status information about all network types supported by the device.
getDefaultSensor	Return a Sensor with the given type and wakeUp properties.

VI. 결 론

본 논문에서는 악성애플리케이션이 어떤 행위를

하고, 어떤 API 를 사용하는지 분석하고 어떤 API 를 사용하는지 분석을 시도하였다. 이를 토대로 API 의 특성정보를 이용하여 전처리 과정, 벡터구성, 검증의 단계를 통한 나이브 베이즈 분류법을 통해 비슷한 기능을 하는 API를 학습하고 분류하였다.

해당 과정을 통해 악성 애플리케이션이 주로 사용하는 API와 같은 부류의 API를 유추하여 분류 할 수 있었고, 실제로 해당 부류의 API는 전체에서 약 14.1%의 비중을 차지하고 있었다.

이를 이용하여 애플리케이션 분석의 입장에서 애플리케이션의 악성의 위험에 있는지에 대해서 판단하는데 중요한 정보를 제공 할 수 있다. 또한 벤더 입장에서 이를 활용하면 이러한 부류의 API 검증을 통해 안전한 플랫폼 설계 또한 기대할 수 있고, 플레이 스토어와 같은 애플리케이션 배포자 입장에서도 배포과정에서 해당부류의 API 디스크립션 정보를 활용하면 개발자가 제공하는 설명 이외의 정보 또한 제공 할 수 있다.

향후에는 본 논문에서 제안한 분류기법의 정확성을 더욱더 높이며, 분류한 API를 토대로 새로운 프레임워크 혹은 안전한 배포과정의 설계 등을 제공할 수 있을 것으로 기대된다.

References

[1] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A Study of Android Application Security," Proceedings of the 20th USENIX conference on Security, p.21-21, Aug. 2011.

[2] J. H. Jung, J. Y. Kim, H. C. Lee, and J. H. Yi, "Repackaging Attack on Android Banking Applications and Its Countermeasures," Journal of Wireless Personal Communications, vol.73, pp. 1421-1437, June 2013.

[3] T. J. Cho, G. B. Na, D. G. Lee, and J. H. Yi "Account Forgery and Privilege Escalation Attacks on Android Home Cloud Devices," Advanced Science Letters, vol. 21, pp. 381-386, Mar. 2015.

[4] C. Collberg and J. Nagra. "Surreptitious Software: Obfuscation, Watermarking, and Tamper Proofing for Software Protection," Addison Wesley Professional, 2009.

[5] C .Collberg, C.Thomborson, and D.Low, "A Taxonomy of Obfuscating Transformations," Technical report 148, Department of computer science, the University of Auckland, Auckland, New Zealand, 1997.

[6] F. Zhang, H. Huang, S, Zhu, D. Wu, and P. Liu, "ViewDroid: towards obfuscation-resilient mobile application repackaging detection," Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks, pp.25-36, July 2014.

[7] DEX File Format, http://source.android.com/devices/tech/dalvik/dex-format.html

[8] Android API Reference, http://developer.android.com/reference/

[9] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," AAAI-98 Workshop on Learning for Text Categorization, Vol. 752, pp. 41-48, 1998.

[10] D. Pavlov, R. Balasubramanyan, S. Kapur, and J. Parikh, "Document preprocessing for naive Bayes classification and clustering with mixture of multinomials," Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp.829-834, Aug. 2004.

[11] ASMDEX, http://asm.ow2.org/asm-dex-index.html

[12] Bytecode, http://source.android.com/devices/tech/dalvik/dalvik-bytecode.html

[13] Contagio, http://contagiominidump.blogspot.kr/

[14] VirusShare, http://virusshare.com/

[15] Apktool, http://ibotpeaches.github.io/Apktool/

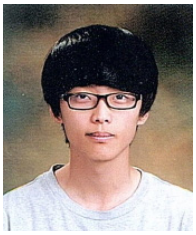
[16] Anubis, https://anubis.iseclab.org/

[17] Virus Bulletin, http://www.virusbtn.com/resources/glossary/malware.xml

〈저자소개〉

조 태 주 (Taejoo Cho) 학생회원
2014년 2월: 숭실대학교 컴퓨터학부 졸업
2014년 3월~현재: 숭실대학교 컴퓨터학과 석사과정
〈관심분야〉 모바일 시스템 보안, 모바일 서비스 보안

김 현 기 (Hyunki Kim) 학생회원
2015년 2월: 숭실대학교 컴퓨터학부 졸업
2015년 3월~현재: 숭실대학교 융합소프트웨어학과 석사과정
〈관심분야〉 모바일 시스템 보안, 모바일 서비스 보안



이 정 환 (Junghwan Lee) 학생회원
2012년 3월~현재: 숭실대학교 컴퓨터학부 학사과정
〈관심분야〉 모바일 시스템 보안

정 문 규 (Moongyu Jung) 정회원
2008년~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호

이 정 현 (Jeong Hyun Yi) 종신회원
1993년 2월: 숭실대학교 전자계산학과 학사
1995년 2월: 숭실대학교 컴퓨터학과 석사
2005년 8월: University of California at Irvine, Computer Science 박사
1995년 2월~2001년 7월: 한국전자통신연구원(ETRI) 연구원
2000년 4월~2001년 3월: 미국 표준기술연구소(NIST) 객원연구원
2005년 10월~2008년 8월: 삼성종합기술원 수석연구원
2008년 9월~현재: 숭실대학교 IT대학 소프트웨어학부 부교수
〈관심분야〉 모바일 보안, 컴퓨터 보안, 네트워크 보안