Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.4, Aug. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.4.817

안드로이드 모바일 악성 앱 탐지를 위한 확률적 K-인접 이웃 분류기*

강 승 준,† 윤 지 원‡
고려대학교 정보보호대학원

Probabilistic K-nearest neighbor classifier for detection of malware in android mobile*

Seungjun Kang,† Ji Won Yoon‡
Graduate School of Information Security, Korea University

요 약

현대인은 스마트폰과 매우 밀접한 관계를 가지고 있으며 이로 인한 수 많은 보안 위협에 노출되어 있다. 실제로 해커들은 스마트폰에 악성 프로그램을 은밀하게 설치하여 장치 이용 제한 및 개인정보 유출 등의 보안 위협을 야기하고 있다. 그리고 그러한 악성 프로그램은 일반적인 프로그램과 다르게 필요 이상의 권한을 요구한다. 본 논문에서는 이 같은 문제를 바탕으로 사용되는 안드로이드 기반 앱들이 요구하는 권한 데이터를 이용하여 주성분 분석(Principle Component Analysis:PCA)과 확률적 K-인접 이웃(Probabilistic K-Nearest Neighbor:PKNN) 방식을 사용하여 효과적으로 악성 프로그램과 일반 프로그램을 분류하고자 한다. 이뿐 아니라 이를 k-묶음 교차 검증(K-fold Croos Validation)을 통해 PKNN의 정확도를 측정하였다. 그리고 일반적으로 사용되는 K-인접 이웃(K-Nearest Neighbor:KNN) 방식과 비교하여, KNN이 분류하기 힘든 부분을 확률적으로 해결하는 PKNN방법을 제안한다. 최종적으로 제안한 방식을 최적화하는 k와 β 파라미터를 구하는 것을 목표로 한다. 본 논문에서 사용된 악성 앱 샘플은 Contagio에 요청하여 이용하였다.

ABSTRACT

In this modern society, people are having a close relationship with smartphone. This makes easier for hackers to gain the user's information by installing the malware in the user's smartphone without the user's authority. This kind of action are threats to the user's privacy. The malware characteristics are different to the general applications. It requires the user's authority. In this paper, we proposed a new classification method of user requirements method by each application using the Principle Component Analysis(PCA) and Probabilistic K-Nearest Neighbor(PKNN) methods. The combination of those method outputs the improved result to classify between malware and general applications. By using the K-fold Cross Validation, the measurement precision of PKNN is improved compare to the previous K-Nearest Neighbor(KNN). The classification which difficult to solve by KNN also can be solve by PKNN with optimizing the discovering the parameter k and β. Also the sample that has being use in this experiment is based on the Contagio.

Keywords: Malware Detection, Android Permissions, Principal Component Analysis, KNN, PKNN

접수일(2015년 4월 1일), 수정일(1차: 2015년 6월 15일, 2차: 2015년 7월 14일), 게재확정일(2015년 7월 15일)
* 본 연구는 미래창조과학부의 지원으로 한국 연구 재단의 기초 과학 연구 프로그램의 일환으로 시행 되었음(NRF-2013R1A1A1012797)
† 주저자, kangren@korea.ac.kr
‡ 교신저자,jiwon_yoon@korea.ac.kr(Corresponding author)

I. 서 론

전 세계의 스마트폰 사용자는 2014년 기준으로 17억 5천명으로 조사되었고[1], 스마트폰 시장은 계속해서 성장하고 있다. 이와 더불어 앞으로의 시대는 운영체제가 컴퓨터에서 한정 되지 않고 항시 휴대가 가능한 작은 스마트폰 디바이스에서 실행될 것이다[2]. 이처럼 현대인들의 삶에서 스마트폰의 중요도가 높아짐에 따라 스마트폰 사용자의 개인정보가 보안 위협에 노출될 것으로 예상된다.

스마트폰에서 사용되는 운영체제로는 Android, iOS, Windows, Blackberry등이 있다. NETMARKETSHARE에 따르면 Android 운영체제는 46.38%의 시장 점유율을 가지는 것으로 나타났다[3]. 안드로이드 앱들은 개발자 사이트에서 개발자 정보를 입력한 후 마켓에 안드로이드 앱을 등록할 수 있다. 이와 같은 안드로이드 마켓의 취약한 시스템은 악성 앱이 주입될 위험성을 가진다[4]. 즉, 악성코드를 유포하는 방식으로 악성코드가 포함된 앱을 받은 사용자의 개인정보는 위협받게 된다. 그러므로 악성 코드를 탐지하는 많은 연구들이 현재 진행중에 있다.

악성 앱을 프로그램으로 분석하는 방법으로는 프로그램을 실행하지 않고 그 파일 자체를 분석하는 정적 분석 방법과 프로그램 실행 과정을 단계적으로 분석하는 동적 분석 방법으로 나뉜다. 하지만 이러한 방법은 현재 구글플레이(Google Play)에 올라온 안드로이드 앱이 약 150만개가 올라 왔을 때[5], 이를 정적 분석이나 동적 분석을 위해서는 현실적으로 많은 처리시간이 요구된다. 이와 같이 아직까지 알려진 안드로이드 악성 코드가 성장함에 따라 미래에는 권한 기반의 탐지 기술에 대한 고려가 필요하다[16]. 그러므로 본 논문에서는 앱들을 다운 받을 때, 필요 이상의 권한을 요구하는 악성 앱의 특징을 통해 악성 앱을 판별하고자 했다. 분류 방법으로는 확률적인 k개의 인접 이웃을 비교하여 분류하고, 악성 앱으로 판별된다면 사용자들이 앱을 다운받을 시 경고를 주는 목적으로 연구하였다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로 시그니처 기반 탐지, 행위 기반 탐지와 기계 학습 방식에 대해서 기술하였다. 3장은 제반 연구로 악성 앱을 분류하기 위한 분류기로 일전적인 KNN과 주성분 분석(PCA), k-묶음 교차 검증 방법에 대해서 설명하며, 4장은 본 논문에서 제안하는 방식인 PKNN대해서 설명하였다. 5장은 실험을 통해 KNN과 PKNN의 정확도를 비교하고 최적화된 k와 β 파라미터를 찾고 이에 대해 분석하였다. 본 논문에 대한 요약으로 마무리 하였다.

II. 관련 연구

2.1 시그니처 기반 탐지

시그니처 기반 탐지(Signature Based Detection)는 오용 탐지(Misuse Detection) 또는 지식 탐지(Knowledge Detection)라고도 불린다. 악성 코드를 정적 분석 하여 특정 시그니처를 발견하게 되면 이를 저장하고 이 후에 같은 시그니처가 발견되면 악성 코드로 탐지하는 방식이다. 여기서 시그니처에 기록되는 내용은 DDL 이름, API 이름 또는 의심스러운 정규표현 식 등 이전의 분석 과정에서 의심스러운 부분을 간략하게 저장한다.

이러한 시그니처 기반 탐지 기술은 악성 코드가 아님에도 불구하고 악성코드라 인식하는 경우가 적고 많이 알려져 있는 악성코드에 대해서는 전부 탐지할 수 있는 장점이 있다. 하지만 제로데이 공격과 같은 시스템 취약점을 이용한 악성코드에 대해서는 탐지하기가 어렵다[18]. 또한 만약 비교해야 할 시그니처가 많으면 모바일 장비의 메모리 자원을 많이 사용해서 부하를 많이 줄 것이다[19]. 이러한 특징으로 시그니처 기반의 악성 코드 탐지는 미래의 변화된 악성코드를 탐지해 내는 데는 한계가 있다.

2.2 행위 기반 탐지

행위 기반 탐지 방식(Behavior Based Detection)은 동적 분석 방법으로, 수집한 정보를 분석하여 악성 행위를 탐지하는 방법이다. 즉, 실행 파일을 직접 실행하여 시스템 콜과 시스템 로그에 대한 정보를 분석하여 악성 행위에 대한 패턴을 찾아 악성 코드를 분류한다. 이와 같은 행위 기반 탐지 방식은 시그니처 기반 탐지에서 찾지 못하는 변종된 악성 코드를 탐지해 낼 수 있다. 또한 서포트 벡터 머신(Support Vector Machines)기반의 행위 기반 탐지 방식으로 탐지 정확도가 96%이다[17].

최근에는 이와 같은 동적 분석을 피하기 위해서 공격자는 악성 코드에 중간에 실행을 중지해서 동적 분석을 못하는 경우가 있다. 그러므로 동적 분석 방

법은 악성코드 정상적인 실행가능 여부가 중요하다[18]. 이와 같이 정적 분석과 마찬가지로 동적 분석도 한계가 있다.

2.3 기계 학습 방식

기계 학습 방식은 기계로 하여금 주어진 데이터 샘플을 학습 및 분석을 통해서 패턴을 찾고 기계 스스로 문제에 대한 해결을 하는 방법이다. 악성 코드 분석에서는 악성 코드 샘플들을 트레이닝을 통해서 분석하여 이들의 패턴을 이용해서 악성 코드를 분류한다. 분류 기술로는 서포트 벡터 머신, 의사 결정 트리(Decision Tree), KNN등이 있다.

이 중에서 요구 권한을 이용하여 기계학습 중 의사 결정 트리를 사용한 탐지 정확도는 약 90%이다. 이는 200개의 데이터 셋에 대해서 각각 J48가 90.72%, 랜덤 포레스트(Random Forest)가 91.75%, CART(Classification and Regression Tree)가 90.72%의 정확도이다[21]. 이는 높은 확률이지만 최적의 의사 결정 트리는 NP-완전 문제로 최적화가 힘들다. 이러한 점에서 다른 기술보다 성능은 조금 낮지만 간단한 KNN의 최적화를 통해서 그 이상의 정확도를 보일 것이다. 특히 확률적 계산을 통해서 확률 계산이 없는 기술의 한계를 극복하고자 한다.

III. 제반 연구

3.1 안드로이드 보안

안드로이드 운영체제는 API level 15을 기준으로 165개의 권한들이 존재한다[6]. 이러한 많은 권한에 대해서 사용자는 인식을 하지 못한다. 그래서 사용자가 원하는 앱을 다운 받을 때, 앱에 필요한 권한을 승인하는 메시지 박스에 대해서 어떠한 권한을 요구하는지 모르고 승인해버린다. 그래서 Adrienne는 안드로이드 마켓에서 얻은 940개의 샘플 앱중 1/3정도가 권한을 과도하게 요청하고 있으며, 안드로이드 개발자 문서에서 설명하는 권한 정보를 포함하고 있다고 말한다[7]. 이러한 취약점은 사용자가 악의적으로 작용될 수 있어서 더욱 주의가 필요하다.

서드 파티 어플리케이션이란 해당 앱이 사용하는 기존의 자원이 아닌 추가적인 기능에 대한 권리로 안드로이드 고유의 보안 체계로, 카메라 기능이나 GPS와 같은 기능이 이에 해당한다. 만약 사용자가 그에 승인한다면 승인한 권한은 공격자가 그 앱에 국한되지 않고 다른 응용 프로그램 내에서의 함수 호출을 가능하게 만든다. 이와 같이 안드로이드 앱에서의 서의 파티 어플리케이션은 취약점이 존재한다.

이러한 권한 요구는 사용자가 앱을 다운받기 직전에 이루어진다. 해당 앱이 요구하는 권한을 통해서 이 앱이 어떤 정보를 필요로 하고 어떤 기능을 사용할 수 있을지를 예상 할 수 있다. 특히 악의적인 공격자는 필요 이상의 권한 요구에 대한 승인을 받아 사용자의 개인정보를 획득하거나 손쉽게 관리자 권한을 획득할 수 있을 것이다. 즉, 서드 파티 어플리케이션이 사용자 자신이 검증하는 방식을 사용하기 때문에 악성 코드에 자유롭지 못하다고 말할 수 있다[17].

Fig.1은 일반 앱(왼쪽), 악성 앱(오른쪽)에 대한 요구 권한을 그래프로 나타낸 그림이다. 그림에서 볼 수 있듯이 악성 앱이 일반 앱보다 더 많은 권한을 요구하는 것을 확인 할 수 있으며 이는 악성 앱이 필요 이상으로 더 요구하는 권한으로 악의적인 목적이 있다고 판단 할 수 있다. 그림과 같이 본 논문에서는 일반 앱과 악성 앱이 요구하는 권한의 차이를 통해서 두 클래스를 분류하는 것을 목적으로 한다.

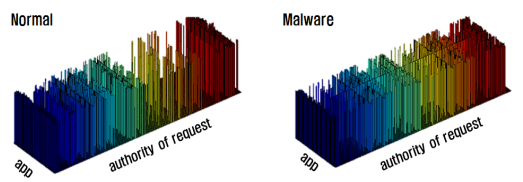

Fig. 1. normal vs malware (authoity of request)

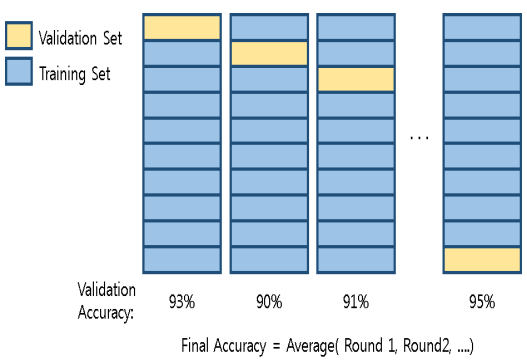

Fig. 2. k-Fold Cross Validation

3.2 분류 기술 모델

3.2.1 특성 추출: 주성분 분석

본 논문에서는 안드로이드 앱이 특정 권한을 요구하면 1, 특정 권한을 요구하지 않으면 해당 특징에 대해서 0을 가지도록 설정했다. 이러한 특성 벡터 F의 형태는 다음의 수식과 같다.

$$F=(f_1,f_2,\cdots,f_N),\\ where\ f_n=\begin{cases}1 & \text{if } f_n\ request\ permision\\ 0 & otherwise\end{cases} \quad (1)$$

안드로이드 권한은 API level 15를 기준으로 165의 권한을 가지고 있다. 이로 인해 차원의 저주가 발생하며, 두 클래스를 비교하는데 더 많은 샘플 데이터가 필요할 수 있다. 그래서 본 논문에서는 앱으로부터 얻어진 특성 벡터가 효율적으로 사용될 수 있도록 주성분 분석(Principle Component Analysis)을 이용하여 새로운 특성 벡터를 추출하였다[9]. 주성분 분석은 고차원에서 사용되며 데이터들 간의 구분에 있어서 특성이 적은 것들을 제거하여 차원을 감소시킨다. 그리고 감소시킨 의미가 있는 값을 이용하는 방법이다.

주성분 분석의 알고리즘은 다음과 같다. 먼저 특정 벡터 $F_\mu(\mu=1,2,\cdots,N)$ 일 때, 전체 데이터의 평균값 m, 공분산 행렬 S를 계산한다. 그 후 공분산 행렬 S에서 대한 고유 벡터와 고유 값을 찾아서 고유 값의 크기 순서대로 상위 M개의 고유 값에 대한 고유 벡터 $E=[e^1,e^2,\cdots,e^M]$를 구한다.

$$m=\frac{1}{N}\sum_{\mu=1}^{N}F_\mu \quad (2)$$

$$S=\frac{1}{N-1}\sum_{\mu=1}^{N}(F_\mu-m)(F_\mu-m)^T \quad (3)$$

각 데이터 요소 f_μ에 하위 차원인 차원의 축소에 대한 표현은 다음과 같이 식 (4)의 y_μ로 나타낸다.

$$y_\mu=E^T(f_\mu-m) \quad (4)$$

그래서 식 (4)에서 계산했던 y_μ를 이용하여 특징 데이터 f_μ에 대한 재구성 값은 다음과 같이 구할 수 있다.

$$f_\mu\approx m+Ey_\mu \quad (5)$$

3.2.2 KNN(K-인접 이웃 분류기)

데이터 셋의 앱은 정상(normal)과 악성(malware)의 2개의 클래스로 분류가 된다. 임의의 특성 벡터 F_μ에 대한 클래스는 다음과 같다.

$$C(F_\mu)\in normal,\ malware \quad (6)$$

KNN 분류는 기준이 되는 테스트 데이터에 대해서 k개의 가까운 이웃을 찾는 방식으로 두 특성 벡터의 차이로 이웃을 정한다. 즉, 두 특성 벡터 $F_x=[f_1,f_2,\cdots,f_N]$와 $F_y=[f_1,f_2,\cdots,f_N]$의 유클리디안 거리를 이용하여 차이를 계산하며 이는 다음 식 (7)과 같다.

$$dist(F_x,F_y)=\|F_x-F_y\|\\ =\sqrt{(x_1-y_1)^2+\cdots+(x_N-y_N)^2} \quad (7)$$

기존 트레이닝 데이터들을 기준으로 새로운 데이터의 특성 F_x의 가장 가까운 k개의 특성 벡터들의 집합을 $F_{n_k}(x)$라고 할 때 알고리즘은 다음과 같다.

```
1 : input : F_x,  x ∈ 1,2,⋯,n
2 : if N(C(F_{n_k}(x)) = normal) ≥ N(C(F_{n_k}(x)) = malware) then
3 :     C(F_x) = normal
4 : else
5 :     C(F_x) = malware
6 : end if
```

$N(C(F_{n_k})=normal)$ 는 $C(F_{n_k})=normal$ 인 F_x의 개수이며, 반대로 $N(C(F_{n_k})=malware)$ 는 $C(F_{n_k})=malware$ 인 F_x의 개수를 나타낸다.

이러한 KNN은 간단한 방식으로 많이 사용되어지고 있다. 하지만 두 개의 클래스를 분류할 때 k가 홀수로 정해주어야 하는 점과 두 클래스를 가지는 샘플의 수가 비대칭을 때는 성능이 감소하는 단점이 존재한다. 그러므로 KNN 분류기의 단점은 확률적인 관점으로 바라보는 PKNN을 통하여 극복 가능하다.

3.2.3 k-묶음 교차 검증 방법

성능 측정 방법 중에 대표적인 방법은 k-묶음 교차 검증 방법이다. k-묶음 교차 검증 방법은 샘플 데이터가 적을 때, 적용되는 방식이다. 첫째, k-묶음 교차 검증 방법은 전체 샘플 데이터들을 k등분으로 나누어 처음 묶음부터 검증 데이터로 사용한다. 둘째, 테스트 데이터 묶음을 제외한 나머지 묶음을 트레이닝 셋으로 사용해서 총 k번의 라운드를 수행을 한다. 이러한 방법을 다른 말로는 leave-one-out이라고도 하며 각 라운드에서 계산한 정확도의 평균값으로 전체 정확도를 구할 수 있다. 본 논문은 Contagio에 요청해서 받은 데이터 200개로 k-묶음 교차 검증 방법을 적용한다.

3.2.4 정확도 측정

분류된 앱이 실제 데이터 클래스와 비교하여 분류 기술의 정확도를 측정하는 방법은 다음 Table 1.과 같다. 여기서 TP(True Positive)는 실제 일반 앱일 때 일반 앱으로 맞게 분류했을 경우, TF(True Negative)는 실제 악성 앱일 때 악성 앱으로 맞게 분류했을 경우, FP(False Positive)는 실제로는 일반 앱이지만 분류기술이 오탐하여 악성 앱으로 분류하였을 경우이며 FN(False Negative)는 실제로는 악성 앱이지만 일반 앱으로 오탐 했을 경우를 나타낸다.

식(8)과 같이 TPR(True Positive Rate)은 일반 앱으로 분류한 것 중 실제와 일치하게 식별한 비율을 나타내고, 반대로 식(9)와 같이 FPR(False Positive Rate)는 악성 앱으로 분류한 것 중 실제와 일치하게 식별한 비율을 나타낸다. 이러한 값들은 이 후에 실험 결과를 분석 하는데 용이한 값이 될 것이며 본 논문에서는 k-묶음 교차 검증으로 측정한 값으로 식 (10)과 같이 분류 정확도를 계산 하였다.

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

Table 1. Accuracy Classification Table

		Inferenced Classification Class	
		Normal	Malware
Actual Class	Normal	TP(true positive)	FP(false positive)
	Malware	FN(false negative)	TN(true negative)

$$FPR = \frac{FP}{TN + FP} \tag{9}$$

$$\text{정확도}(Accuracy) = \frac{TP + TN}{(TP + FP + TN + FN)} \tag{10}$$

IV. 제안 방법

4.1 권한 기반 탐지

앞서 연구에서는 시그니처 기반과 행위 기반을 통해서 악성 앱을 탐지하였다. 이러한 악성 코드 탐지는 정적/동적 분석 방식은 병행해서 사용하지만 한계가 있다. 공격자는 이러한 두 가지 탐지 방식을 피해서 악의적인 행위를 해오고 있다. 그러므로 안드로이드의 서드 파티 어플리케이션 특징을 이용하여 APK 파일을 분석할 필요 없이 앱 설치 시 요구하는 권한을 분석하여 일반 앱과 악성 앱을 분류한다.

4.2 PKNN(확률적 K-인접 이웃 분류기)

일반적인 KNN인 경우 확률적 관점이 없기 때문에 데이터를 예측하는 측면에서 손실 발생이 가능하다[10]. 그래서 PKNN(Probabilistic K-Nearest Neighbor)인 경우 k개의 인접한 이웃을 찾은 후 그 이웃들이 선택될 확률 값을 추가적으로 계산하여 최대 우도(Maximum Likelihood:ML)를 구하여 새로운 데이터의 클래스를 예측한다.

$$p(Y|X, \beta, k) = \prod_{i=1}^{n} \frac{\exp\left\{\beta(1/k) \sum_{j \sim k_i} \delta_{y_i y_j}\right\}}{\sum_{q=1}^{Q} \exp\left\{\beta(1/k) \sum_{j \sim k_i} \delta_{q y_j}\right\}} \tag{11}$$

위 식 (11)에서 Y는 새로운 데이터의 클래스, y_x는 x의 클래스를 뜻한다. X는 트레이닝 데이터 셋을 나타내고, k는 이웃의 개수, β는 회귀계수로 이웃 간의 관계의 비중을 나타낸다. δ_{ab}는 Dirac Function으로 a와 b가 같을 때는 1, 다를 때는 0을 반환하는 함수이다. Q는 전체 클래스를 나타내는 변수이다. 식 (11)에서 분모는 새로운 데이터 이웃 값들의 클래스 전체 확률이다. 분자는 이웃인 값들이 각 클래스가 될 확률로써, 다시 말하면 해당 클래스가 선택 될 확률을 이웃들의 전체 확률로 나눠서 그

각각의 클래스일 우도를 계산한다. 이후 그 중 우도(likelihood)가 가장 큰 값의 클래스가 새로운 데이터의 클래스라고 추론하는 방법이다.

PKNN방식은 KNN을 비교했을 때 2가지 장점이 있다. 첫 번째 장점은 Fig. 3.과 Fig. 4를 비교하여 설명할 수 있다. Fig. 3.는 2개의 클래스를 이용하여 KNN을 설명한 그림이다. 그림 왼쪽은 k가 홀 수 일 때의 일반적인 이웃 근접 분류방식의 예로 파란 원이 2개로 많기 때문에 파란 공으로 새로운 데이터는 분류된다. 반면, 이웃의 개수인 k가 짝수일 때 각 이웃의 클래스 개수가 서로 같다면 새로 들어온 데이터 분류할 방법이 없다.

Fig. 4.는 2개의 클래스를 PKNN으로 설명한 그림이다. 이 그림에서 왼쪽은 $P(Y=\text{파란 원})=0.2$, $P(Y=\text{노란 삼각형})=0.3$ 으로 파란 원이 0.4의 확률로 더 높은 확률을 가지기 때문에 새로운 데이터는 파란 원으로 분류된다. 반면에 그림에서 오른쪽은 k가 4일 때 경우, 짝수일 때 파란 원일 확률은 총 0.4, 노란 삼각형이 될 확률은 총 0.6으로 노란 삼각형이 될 확률이 높다. 이 때문에 k가 짝수 일 때도 새로운 값의 클래스를 분류할 수 있다. 또한, 분류하고자 하는 모든 클래스들이 동등한 개수를 가지지 않고 각 클래스 총 개수의 차이가 불균형한 상태일 때 각 클래스가 서로 다른 확률 값을 가진다. 나아가 클래스 별로 다른 비중을 줘서 더 좋은 성능을 기대할 수 있다.

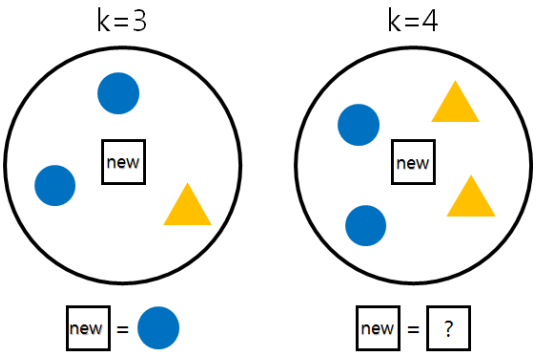

Fig. 3. Case of KNN(Problem)

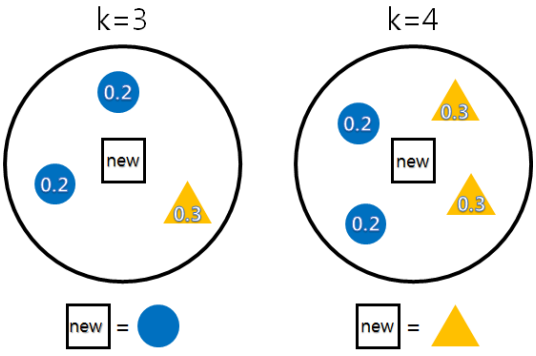

Fig. 4. Case of PKNN(Problem)

V. 실 험

5.1 KNN vs PKNN

본 논문에서 사용한 데이터는 200개의 안드로이드 앱으로 100개는 일반 앱이고 나머지 100개는 악성 앱이다. 이 앱들은 286가지의 권한 특징을 0과 1로 표현한다. 근접 이웃들이 의미를 가지기 위해서는 10~15차원이하가 되어야[11]하기 때문에 286개의 차원을 주성분 분석 방법을 통해서 차원 감소를 시킨다. 이러한 이유로 본 논문에서는 15차원으로 주성분 분석을 한 후, 5개의 인접 이웃을 이용하여 10개 묶음 교차 검증 방법으로 10개의 묶음으로 나눈다. 처음 묶음은 검증 데이터로 사용하고, 그 다음 묶음은 테스트 데이터 사용한다. 이와 같은 방법으로 총 10라운드를 수행하여 정확도의 평균을 계산하였다.

먼저 KNN 분류 방식으로 검증 데이터에서 가까운 이웃인 트레이닝 검증 값들을 5개 찾아서 그 이웃들이 악성 앱이 많다면 해당 검증 데이터는 악성 앱으로 예상할 수 있다. 이와 다르게 PKNN은 해당 검증 데이터의 이웃 데이터를 그대로 사용하는 것이 아니라 확률 값으로 계산하여 우도가 큰 값이라고 판단하는 방법으로 실험 결과는 Fig. 5와 같다.

본 실험에서는 보유하고 있는 데이터를 랜덤하게 순서를 바꿔서 검증 데이터와 트레이닝 데이터를 사용하였다. Fig. 5과 Fig. 6.에서 그래프의 가로축은 테스트 횟수(Count), 세로축은 정확도(Accuracy)를 나타내며, 파란 점선은 KNN, 주황 실선은 PKNN의 정확도를 나타낸다. 분석 결과 PKNN이 높은 정확도를 가지는 것을 확인 할 수 있다. Fig. 5.는 k가 홀 수 일 때의 그래프로 횟수가 4번째, 9번째 일 경우 PKNN이 더 낮은 정확도를 가지는데 이는 PKNN이 확률적인 분류 방식이기 때문에 발행하는 현상이다. 하지만 각 시도마다 정확도의 변화가 많은 KNN에 비해서 전체적으로 균일한 정확도를 보인다.

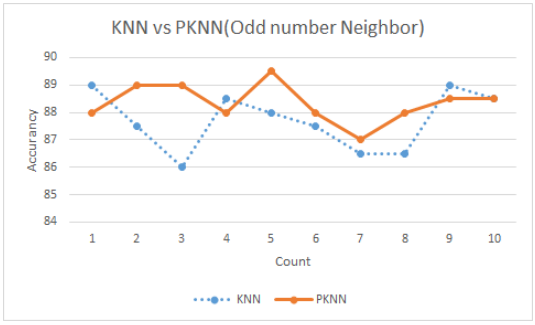

Fig. 5. KNN vs PKNN (Odd number Neighbor)

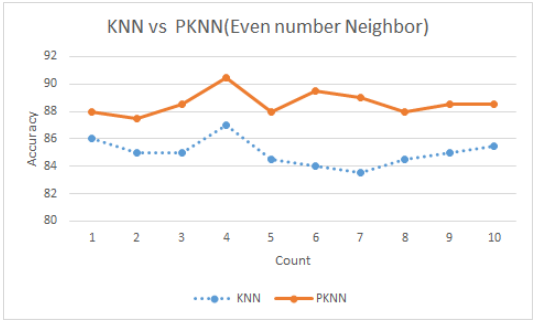

Fig. 6. KNN vs PKNN (Even number Neighbor)

그리고 Fig. 6은 인접 이웃의 개수를 짝수로 설정한 결과이다. 먼저 KNN은 짝수 k일 때, 85%정도의 정확도를 나타내서 성능이 떨어지는 것을 확인할 수 있다. 하지만 PKNN은 k가 짝수라도 홀수일 때와 같은 정확도를 보여준다. 이러한 이유는 KNN 알고리즘은 이웃인 클래스의 수가 일반과 악성이 같을 때 일반 앱으로 분류하지만 실제 데이터는 악성 앱일 경우가 있기 때문이다. 그래서 k에 변화에 관계없이 성능 변화가 없는 PKNN이 KNN보다 성능이 더 좋다고 할 수 있다. 또한 본 실험에서는 클래스가 2개뿐이지만 다중 클래스인 경우는 k가 홀수라도 이웃인 클래스인 개수가 같은 경우가 발생하기 때문에 다중 클래스 환경에서도 더 좋은 정확도를 기대할 수 있다.

5.2 k ,β 파라미터 최적화

앞의 실험에서는 파라미터 k는 5, β를 1으로 고정했다. 하지만 PKNN을 더 좋은 정확도를 가지는 탐지 분류기로 사용하기 위해서는 이 파라미터를 최적화하는 작업이 필요하다. 이를 위한 실험으로는 정해진 범위에서 k와 β를 변경, 모든 경우의 값에서 likelihood가 가장 높은 값으로 추론하는 방식이다. 이 방식을 위해서는 k-묶음 교차 검증 방법으로 검증하고 이를 매번 무작위로 섞은 10번의 횟수로 실험을 하여 그 평균값으로 계산한다.

먼저 Fig. 7는 k를 1에서 15까지 범위에서변화시킬 때 정확도(Accuracy) 나타내는 것으로 β를 0.1로 고정하였다. 그림에서 볼 수 있듯이 k가 9일 때 91.5% 정확도로 가장 높은 것을 볼 수 있다. 인접한 이웃의 개수가 9이상일 때 일반 앱과 악성 앱을 판단하는데 있어 정확성이 감소하는 이유는 자신과 관련 없는 클래스도 이웃으로 되어 오탐을 야기하기 때문이다.

β값을 기준으로 k가 1~15, β가 0.1~9(그래프에서 1 = 0.1)에 대해 표현한 그래프는 Fig. 8.과 같다. 그래프를 봤을 때, β는 정확도에 큰 영향을 주지 않고 0.1에서 8.0까지의 값에서는 일정한 정확도를 가지다 그 이상인 8.1의 β값에서는 서서히 정확도가 떨어지는 결과를 확인할 수 있다. 본 논문에서 총 200개의 앱 샘플들 중 일반 앱이 100개, 악성 앱이 100개로 균등한 양을 가지고 실험에서 랜덤으로 샘플들을 섞은 후에 k-묶음 교차 검증 방법으로 완전히 똑같은 수로 검증이 되는 것은 아니지만 두 클래스 간에 전체 개수의 차이가 크지 않기 때문에 β값이 정확도에 영향을 끼치지 않기 때문이다. 하지만 실제 환경에서는 일반 앱과 악성 앱의 수가 같지 않기 때문에 β값의 최적화도 필요하다.

최종적으로 Table 3.은 k가 1~15까지 β는 1~10까지 앞의 Fig. 7.과 Fig. 8.의 그래프를 조

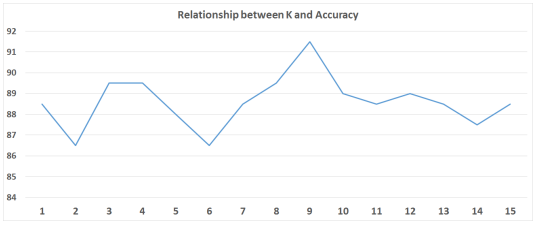

Fig. 7. Relationship between k and Accuracy

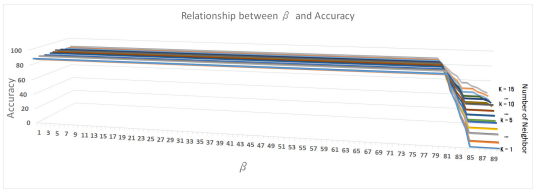

Fig. 8. Relationship between β and Accuracy

합한 정확도 표이다. 총 10번의 실험을 하여 정확도도의 평균값과 표준편차 값을 계산하였다. 앞의 내용을 종합해 볼 때, k값이 9이고 β가 8.0이하인 값일 때, 악성 앱을 탐지할 정확도가 90.1%로 가장 최적의 k, β 파라미터 값이다. 또한 표준 편차 값이 0.747으로 전체 표준 편차의 평균 1.531보다 작기 때문에 다른 파라미터보다 평균값이 고르다. β값을 통해서 정확도가 변화하는 부분이 있지만 현재 Contagio에서 받은 200개의 샘플 데이터는 비슷한 같은 비율을 섞어서 하기 때문에 검증 데이터로 뽑힌 데이터에 의해서 β에 영향을 받는다고 볼 수 없으며 실험결과 이웃한 k 파라미터의 값이 큰 영향을 준다.

여기서 k와 β가 최적화 되었을 때 오탐율 9.9%에 대한 분석을 위한 표는 Table 2.와 같다. 여기서 TPR은 96%로 일반 앱에 대한 분류 정확도가 높다. 여기서 일반 앱중에서 특별히 많은 권한을 요구하는 앱으로 오탐률 4%가 발생한다. 반면 FPR은 87%로 TPR보다 상당히 떨어지는 것으로 보아 권한 요구는 일반 앱과 같지만 실제로는 악의적인 행위를 하는 앱으로 판별된다. 그러므로 권한 요구가 적은 악성 앱을 탐지할 수 있는 추가적인 요소가 필요하다.

Table 2. Optimal Accuracy Table

	Actual Normal	Actual Malware	Accuracy
Inferenced Normal	86	3	96%(TPR)
Inferenced Malware	14	97	87%(FPR)

VI. 결 론

일반적으로 안드로이드 OS 기반의 앱들은 해당 앱에 필요한 권한을 다운 받기 직전에 해당 사용자에게 승인을 확인 받는다. 이 과정에서 대부분의 사용자들은 이 절차를 중요하게 생각하지 않고 있으며, 이 같은 점은 악의적으로 이용될 위험성을 갖고 있다. 이러한 이유로 사용자는 앱을 다운받기 전 요구하는 권한을 분석하여 승인된 권한인지 아니면 악의적인 의도를 가지고 있는지 판단할 수 있어야 한다.

본 논문에서 각 앱들이 요구하는 권한 데이터의 특징을 이용하여 KNN 이웃 분류 방식에 확률적인 요소를 추가한 향상된 성능의 확률적인 KNN분류 방식을 제안하였다. 즉, KNN분류 방식에서의 단점인 k값이 반드시 홀수가 되어야하는 점과 각 클래스가 균등한 분포가 아닌 경우에서 각 클래스의 가중치를 주지 못하는 점을, 확률적인 KNN분류를 통해서 극복할 수 있음을 보였다. 또한 이 분류 방식을 통해 현재 안드로이드 앱 다운 시 가지는 취약점을 판단하여 악의적인 공격자가 마켓에 올린 악성 앱을 판별하는 일에도 도움을 줄 수 있을 것이다.

본 연구를 통해서 시그니처 기반 방식과 행위 기반 방식으로 탐지하지 못하는 악성 앱을 권한 기반 방식을 통해서 탐지할 수 있음을 보였다. 또한 고정된 값을 쓰지 않고 교차 검증 방법을 이용하여 k와 β 파라미터를 최적화하였으며, 악성앱을 분류하기 이전에 트레이닝을 하는 부분에서 미리 계산된 PKNN이 최적화된 파라미터 값을 이용해 정밀하게 악성 앱을 분류할 수 있다.

향 후 연구로는 악성 앱 분류에 대한 정확도 향상을 위한 요구 권한과 API call을 조합하여 효과적인 악성 앱 탐지를 할 수 있을 것이다[20]. 또한 KNN과 마찬가지로 PKNN도 큰 데이터를 처리할 때 많은 처리시간이 요구되는 문제점을 해결하기 위해서 KNN보다 연산속도가 빠른 알고리즘인 k-d tree나 LSH(Locality Sensitive Hashing)를 PKNN에 결합을 하여, 빅데이터 기술에도 접목할 수 있는 분류 기술에 대하여 연구할 예정이다.

References

[1] eMarketer, [Online] "Smartphone Users Worldwide Will Total 1.75 Billion in 2014" URL{http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536} Jan. 2014

[2] DENCK, William, et al. "Understanding Android Security," IEEE security & privacy, pp. 50-57, Jul 2009.

[3] NETMARKETSHARE. [Online] "Mobile/Tablet Top Operating System Share Trend" URL{http://www.netmarketshare.com/} Oct.2014

[4] ORTHACKER, Clemens, et al. "Android security permissions-can we trust them?." In: Security and Privacy in

Mobile Information and Communication Systems. Springer Berlin Heidelberg, pp. 40-51. 2012.
〔5〕 AppBrain. 〔Online〕 "Android Statistics - Google Play stats", URL{http://www.appbrain.com/stats/number-of-android-apps}, Mar 2015.
〔6〕 WEI, Xuetao, et al. "Permission evolution in the android ecosystem.," In: Proceedings of the 28th Annual Computer Security Applications Conference. ACM, pp. 31-40. 2012.
〔7〕 FELT, Adrienne Porter, et al. "Android permissions demystified.," In: Proceedings of the 18th ACM conference on Computer and communications security. ACM, pp. 627-638. 2011.
〔8〕 WOLD, Svante; ESBENSEN, Kim; GELADI, Paul. "Principal component analysis." Chemometrics and intelligent laboratory systems, 2.1:pp. 37-52. 1987,
〔9〕 BARBER, David. "Machine Learning A Probabilistic Approach.2006.5 August." URL{http://files.is.tue.mpg.de/hjhuang/ebook/mlgm_epfl_book.pdf}
〔10〕 HOLMES, C. C.; ADAMS, N. M. A "probabilistic nearest neighbour method for statistical pattern recognition.," Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64.2: pp. 295-306. 2002,
〔11〕 BEYER, Kevin, et al. When is "nearest neighbor," meaningful?. In: Database Theory—ICDT'99. Springer Berlin Heidelberg, pp. 217-235. 1999.
〔12〕 Barber, David. "Machine Learning and Pattern Recognition Principal Component Analysis. 2001. 5 August." URL{ http://www.inf.ed.ac.uk/teaching/courses/mlpr/lectures/mlpr-dim-red.pdf}
〔13〕 Zhang, Min-Ling, and Zhi-Hua Zhou. "A k-nearest neighbor based algorithm for multi-label classification." Granular Computing, 2005 IEEE International Conference on. Vol. 2. pp.718-721, IEEE, 2005.
〔14〕 Hyelim Lee, J. W. Yoon, "Efficient Malware Detector for Android Devices," Korea institute of information security and cryptology, 24(4), pp. 617-624. 2014
〔15〕 Yoon, Ji Won; FRIEL, Nial. "Efficient model selection for probabilistic K nearest neighbour classification." Neurocomputing, 149: pp. 1098-1108. 2015,
〔16〕 FELT, Adrienne Porter, et al. "A survey of mobile malware in the wild." In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011. pp. 3-14.
〔17〕 BOSE, Abhijit, et al. "Behavioral detection of malware on mobile handsets." In: Proceedings of the 6th international conference on Mobile systems, applications, and services. ACM, 2008. pp. 225-238.
〔18〕 Jae-sung Yun, Jae-wook Jang, Huy Kang Kim. "Andro-profiler: Anti-malware system based on behavior profiling of mobile malware." Korea institute of information security and cryptology. 2014, 24.1: pp. 145-154.
〔19〕 VENUGOPAL, Deepak; HU, Guoning. "Efficient signature based malware detection on mobile devices.," Mobile Information Systems, 2008, 4.1: pp. 33-49.
〔20〕 PEIRAVIAN, Naser; ZHU, Xingquan. "Machine learning for android malware detection using permission and api calls." In: Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on. IEEE, 2013. pp. 300-305.
〔21〕 AUNG, Zarni; ZAW, Win. "Permission-based Android malware detection." International Journal of Scientific and Technology Research, 2013, 2.3: pp. 228-234.

Table 3. Accuracy Table of β, k Parameters Combination(mean(standard deviation))

k \ β	1	2	3	4	5	6	7	8	9	10
1	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	88.2 (0.677)	0 (0)	0 (0)
2	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	86.2 (3.011)	5.5 (2.990)	5.5 (2.990)
3	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	89.0 (0.724)	13.3 (0.783)	13.3 (0.783)
4	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	87.9 (1.822)	18 (2.160)	18 (2.160)
5	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	88.2 (0.948)	23.6 (1.355)	23.6 (1.355)
6	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	86.6 (2.108)	24 (2.134)	16.8 (4.589)
7	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	88.3 (0.918)	28.7 (1.475)	18 (0.881)
8	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	88.6 (1.528)	33.1 (2.389)	20.7 (1.918)
9	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	90.1 (0.747)	38.8 (1.375)	24.1 (1.125)
10	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	88.3 (1.564)	35.3 (4.655)	24.7 (2.395)
11	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	88.7 (0.889)	37.9 (4.846)	25.4 (2.654)
12	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	87.3 (1.510)	37.9 (4.135)	23 (4.062)
13	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	87.7 (0.714)	38.3 (5.662)	25.4 (2.633)
14	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	87.5 (1.257)	41.4 (6.584)	24.6 (1.395)
15	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	88.5 (0.707)	43.8 (5.840)	28.2 (1.399)

〈저자소개〉

강 승 준 (SeungJun Kang) 학생회원
2013년 8월: 경성대학교 컴퓨터공학사 졸업
2014년 9월~현재: 고려대학교 정보보호 대학원 석사과정
〈관심분야〉 정보보호, 사이버 보안을 위한 전기 전력 시스템, IoT 보안, 다중 라벨 분류

윤 지 원 (Ji Won Yoon) 종신회원
2003년 2월: 성균관 대학교 정보공학사 졸업
2005년 2월: University of Edinburgh, 정보학과 석사 졸업
2008년 11월: University of Cambridge 전자공학과 박사 졸업
2008년 2월~2009년 5월: University of Oxford 로봇연구소 박사후과정
2009년 5월~2011년 5월: University of Dublin 통계학과 연구원 및 강사
2011년 7월~2012년 8월: IBM 연구소 정규 연구원
2012년 9월~현재: 고려대학교 정보보호대학원 조교수
〈관심분야〉 신호정보처리, 응용통계, 빅데이터 분석 기술, 도감청 탐지 기술