2013년 동계 한국정보기술융합학회 융합 IT 학술대회

SVM을 통한 안드로이드 악성코드 탐지 기법

함효식[1], 김환희[1], 김명섭[2], 최미정[1*]

[1]강원대학교 컴퓨터과학과, [2]고려대학교 세종캠퍼스 컴퓨터정보학과
[1]{ manhhs, hwanhee0920, mjchoi }@kangwon.ac.kr, [2]tmskim@korea.ac.kr

Abstract. 모바일 단말이 다양한 서비스 및 컨텐츠를 지원하게 되면서 단말 내에 사용자의 중요한 개인정보들이 산재해 있다. 따라서 공격자들은 기존 PC 및 인터넷 환경뿐만 아니라, 모바일 단말까지 공격 범위를 확대하고 있다. 본 논문에서는 안드로이드 환경에서 발생되는 악성코드를 탐지하기 위하여 리소스 정보를 모니터링하고 이를 자동적으로 탐지하기 위해 기계학습 분류기 중 높은 분석 성능을 보이는 SVM(Support Vector Machine)을 적용하여 악성코드를 탐지하는 방법을 제안한다. 실험 결과를 통해 제안하는 방법론의 우수성을 검증하였다.

Keywords: 안드로이드, 악성코드 탐지, 기계학습, SVM(Support Vector Machine)

1. 서론

스마트폰은 모바일 산업의 발전을 촉진시키고 있으며, 사용자 또한 기하급수적으로 증가하고 있다. 현대인들은 스마트폰으로 언제 어디서나 웹 검색, SNS(Social Network Service), 모바일 뱅킹 서비스 등 다양한 서비스를 이용하고 있으며, 이는 현대인의 생활 패턴과 산업의 패러다임까지 변화시키고 있다. 이러한 스마트폰의 발전은 현대인에게 편리함과 유용성을 가져다 주었지만 그 이면에는 보안 취약점이라는 커다란 위협이 도사리고 있다. 스마트폰에는 사용자의 위치정보, 연락처, 공인인증서 등 다양한 개인중요정보들이 산재해 있기 때문에 해커의 위협적인 공격으로부터 심각한 피해를 입을 가능성이 있다. 현재, 해커들은 기존 PC 환경의 공격에서 스마트폰으로 공격지를 확대하고 있다.

그림 1은 핀란드의 보안업체 F-secure의 보고서로써, 2012년 발생된 301개의 모바일 악성코드 샘플 중 238개의 악성코드가 안드로이드 플랫폼을 대상으로

이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2013R1A1A3011698)

* Correspondence to Mi-Jung Choi, Dept. of Computer Science, KNU, Chuncheon, Republic of Korea.

하고 있다고 한다[1]. 다른 모바일 플랫폼들은 1분기에서 4분기로 시간이 지나면서 악성코드의 수가 점점 감소하고 있는데 비해, 대조적인 결과이다. 안드로이드 악성코드 증가 원인은 오픈소스 정책 및 마켓의 앱 검증이 까다롭지 않은 안드로이드 플랫폼의 특성 때문이며, 악성코드를 정상 어플리케이션에 삽입하는 리패키징 방식으로 마켓에 배포하기 유리하기 때문이다.

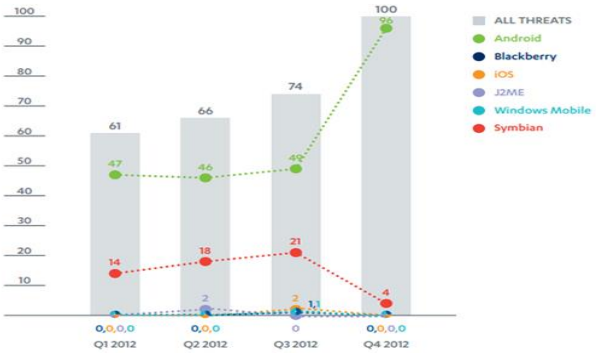

Fig. 1. 안드로이드 악성코드의 증가

기존 연구에서는 모바일 악성코드를 탐지하기 위해 시그니처 기반 기법(Signature-based detection)[2,3,4,5], 행위 기반 기법(Behavior-based detection)[6,7,8,9,10], 동적 분석 기반 기법(Taint analysis-based detection)[11,12] 등의 접근이 있었다. 본 논문에서는 기존 연구들의 문제점을 파악하고 효율적으로 안드로이드 플랫폼에서 악성코드를 탐지하기 위해 어플리케이션 실행시 사용되는 리소스를 모니터링하고 이를 기계학습 알고리즘 중 높은 성능을 보이는 SVM(Support Vector Machine)[13]을 통해 탐지하는 방법에 대해 제안한다.

2. 관련 연구

본 장에서는 모바일 악성코드를 탐지하기 위한 기존 연구 동향을 살펴보고 제안하는 SVM(Support Vector Machine) 기법에 대해 설명하고자 한다.

2.1 모바일 악성코드 탐지 기법 연구 동향

기존 모바일 환경에서 발생하는 비정상행위(악성코드, 바이러스, 웜 등)을 탐지하기 위하여 시그니처 기반 기법, 행위 기반 기법, 동적 분석 기반 기법이 있다.

시그니처 기반 기법[2, 3, 4, 5]은 PC 환경에서 전통적으로 사용되는 방식을 적용한 기법이다. 시그니처를 정의하기 위해서는 주로 정적 및 동적 분석 방법을 사용한다. 정적 분석(Static analysis)은 프로그램을 실제로 실행해보지 않고 분석하는 방법으로, 프로그램의 원시코드 및 목적코드를 대상으로 한다. 악성코드의 소스코드를 디컴파일하여 명령어, 선언문 등에서 발생하는 취약점 등을 발견하면 이를 시그니처를 정의한다. 동적 분석(Dynamic analysis)은 실제로 프로그램을 실행하면서 나타나는 메모리 누수, 트래픽 흐름, 데이터 흐름에서 나타나는 일정한 패턴을 찾아 시그니처로 정의하는 방법이다.

행위 기반 기법[6, 7, 8, 9, 10]은 시스템에서 발생하는 프로세스 행동과 미리 정해진 공격의 패턴을 비교분석하여 침입여부를 탐지하는 방법으로 시그니처 기반 기법의 제한된 악의적인 행동 탐지로 인하여 최근 가장 주목 받고 있는 연구 중 하나이다. 주로 스마트폰 사용 시 발생되는 블루투스, SMS, 배터리 소모량 등의 이벤트 정보를 모니터링하여 비정상적인 패턴을 탐지한다. 기기 내부에서 직접 탐지하는 호스트 기반 탐지 (Host-based detection) 방법과, 네트워크 기반 탐지(Network-based detection) 방법이 많이 사용된다. 하지만 호스트 기반 탐지 방법은 스마트폰의 배터리 및 메모리 사용을 가중시키므로 기기 내부에서 데이터를 수집하고 외부로 데이터를 전송하여 탐지하는 방식을 주로 사용한다.

동적 분석 기반 기법[11, 12]은 Taint analysis으로 불리는 기법으로 특정 데이터에 마킹을 하고, 어플리케이션 코드 내에서 데이터가 전파되는 과정을 모니터링하여 데이터의 흐름을 추적하는 기법이다. 스마트폰은 가상 머신 상에서 실행되므로 본 기법을 적용하기에 적합하다는 평가를 받았지만 낮은 수준까지 데이터의 흐름을 추적하기 위한 오버헤드로 인해 실제 환경에 적용하기에는 어려움이 있어 현재는 연구가 더 이상 진행되지 않고 있다.

2.2 SVM 기법을 통한 악성코드 탐지

본 논문에서는 안드로이드 환경에서 리소스를 모니터링하여 수집한 데이터를 기반으로 악성코드를 판단한다. 이와 같은 행위 기반 기법은 분석자가 수 많은 Feature로 이루어진 데이터를 보고 악성코드 감염유무를 판별해야 하는 불편함이 있다. 따라서 행위 기반 기법은 자동화된 악성코드 분류 및 판별과 정확도를 보장하기 위해 기계학습 기법을 사용한다.

기계학습 기법은 해당 기기로부터 수집한 데이터를 학습데이터로 입력하여 학습모델을 생성하고, 일부 데이터를 학습모델에 적용하여 탐지율을 분석하는 방법이다.

기존 연구[14,15]에서는 RF(RandomForest) 기계학습 알고리즘으로 악성코드를 탐지하는 접근이 있었다. RF는 독립적으로 샘플링 된 랜덤 백터들에 의해 형성된 Decision tree들의 조합으로, 비교적 높은 탐지율을 보였다. 본

논문에서는 악성코드를 탐지하기 위해 SVM(Support Vector Machine) 기법을 적용한다. SVM은 최근 가장 각광받는 기계학습 분류기 중 하나로 우수한 성능으로 인하여 다양한 변형과 응용이 이루어 지고 있다. 또한 SVM은 비선형적인 데이터의 분류문제를 해결할 수 있으며, 입력된 Feature 중 필요없는 Feature를 필터링하는 기능이 포함되어 있어 좋은 성능을 기대할 수 있다.

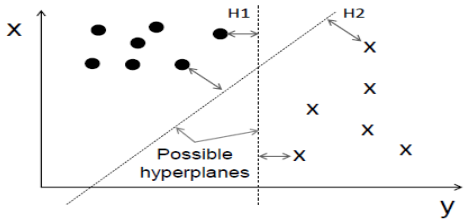

Fig. 2. SVM(Support Vector Machine)의 데이터 분류 방법

그림 2는 SVM이 학습과정을 거쳐 데이터를 분류하는 기준인 초평면(Hyperplanes)을 구하는 방법이다. SVM의 가장 큰 장점은 데이터 간의 거리인 마진(Margin)을 최대화하여 일반화 능력을 극대화하는데 있다. 따라서 데이터가 애매모호한 경계에 위치해 있을 때, SVM의 성능은 다른 분류기보다 더욱 극대화 된다. 본 논문에서는 기존 연구에서 적용되었던 RF(RandomForest)와 제안하는 방법인 SVM 분석 기법의 실험결과 비교를 통해 제안하는 방법의 우수성을 검증하고자 한다.

3. 악성코드 탐지를 위한 리소스 정보 수집

본 장에서는 안드로이드 환경에서 분석대상 악성코드를 탐지하기 위해서 리소스 정보를 수집하는 방법에 대하여 제시한다. 안드로이드 기기 내부에서 리소스 정보를 수집하기 위해 설계 및 구현한 에이전트와 수집한 리소스 데이터에 대한 Feature들을 설명 하고자 한다.

3.1 악성코드 탐지를 위한 리소스 Feature

분석대상 악성코드를 탐지하기 위해서는 정상어플리케이션 및 악성코드가 삽입된 악성어플리케이션을 사용자가 사용하였을 때, 기기에서 발생하는 리소스 정보들을 모니터링 한다. 기존 연구[10]에서는 안드로이드 기기에서 발생하는 모든 리소스와 이벤트들을 정의하였고 이를 통해 악성코드를 분석하는 연구를 수행하였다.

하지만 Feature의 개수가 88개로 너무 많고, 대부분은 상관관계가 적어 성능 및 시간적인 오버헤드가 크다는 단점이 있었다. 본 논문에서는 표 1과 같이

분석대상 악성코드와 높은 관계가 있는 32개의 Feature를 7가지 형태의 리소스 유형별로 나누어 정의하였다. 특히 메모리 유형을 단순히 어플리케이션 수행으로 변화하는 메모리 사용량만을 모니터링 하는 것이 아니라, 안드로이드 플랫폼의 특성을 고려하여 각 어플리케이션 실행마다 할당되는 Dalvik Machine(가상 머신) 영역의 메모리와 Native 영역으로 분류하여 메모리의 사용량을 세분화하여 모니터링한다.

Table 1. 악성코드 탐지를 위한 Feature 선정

리소스 유형	리소스 Feature	
네트워크	RxBytes, TxBytes, RxPacket, TxPacket	
전화	Send/Receive Call	
메시지(SMS)	Send/Receive SMS	
CPU	CPU Usage	
배터리	Level, Temperature, Voltage	
프로세스	Process ID, Process Name, Running Process, Context Switches	
메모리	Native	Total Size, Shared Size, Allocated Size, Physical page, Virtual Set Size, Free Size, Heap Size, Dirty Page
	Dalvik	Total Size, Shared Size, Allocated Size, Physical page, Virtual Set Size, Free Size, Heap Size, Dirty Page

3.2 리소스 정보 모니터링 에이전트

선정한 리소스 Feature를 모니터링 하기 위해서는 기기내부에서 지속적으로 해당 Feature를 모니터링 할 수 있는 에이전트가 필요하다. 그림 3은 리소스 정보를 모니터링 하기 위한 에이전트의 구조이다.

리눅스 커널은 안드로이드 플랫폼의 최하위에서 운영체제 및 하드웨어를 관리하며 이를 위한 드라이버들이 포함되어 있다. 또한 리소스 정보를 모니터링 하는 시작점이 되며 각각의 Collector 들은 각 유형별 리소스를 독립적으로 수집하여 모니터링 상태를 유지해주며, 실시간으로 모니터링 된 데이터를 취합하여 데이터 관리 모듈로 전송된다. 리소스 정보는 API 호출 및 리눅스 명령어를 통해 얻을 수 있다. 데이터관리 모듈은 수집된 리소스 데이터를 기계학습 모델에 적용하기 위해 각각의 데이터를 벡터화 하며, 통신 모듈을 통해 외부 서버에 전송한다. 이와 같이 데이터를 외부서버의 전송하는 이유는 기기에서 발생하는 성능적인 오버헤드를 줄이기 위함이다. 또한 서버에서 기계학습을 통해 결정된 악성코드 탐지여부를 알람 모듈을 통해 사용자에게 알려준다. 마지막으로 사용자 레벨에서는 모니터링 상태 및 악성코드 탐지 여부를 시각화하여 보여준다.

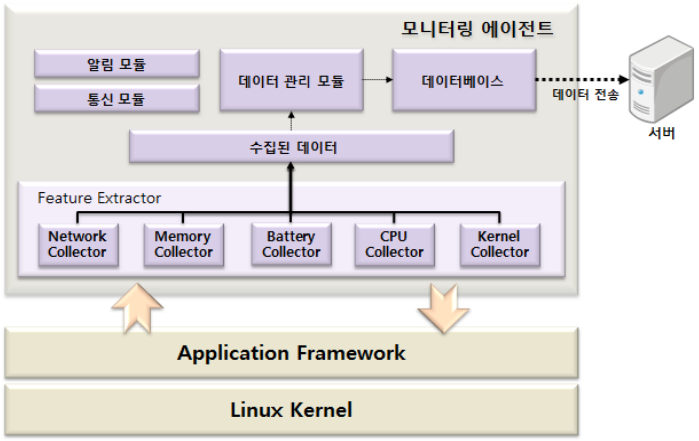

Fig. 3. 리소스 정보 모니터링 에이전트의 구조

4. 실험 및 결과

본 장에서는 제안한 SVM(Support Vector Machine) 기법을 적용하여 악성코드 탐지유무를 검증하기 위하여 RF(RandomForest) 분류기와의 비교를 통해 우수성을 검증하며, 실험 방법과 결과에 대해 서술한다.

4.1 실험 방법

본 논문에서는 악성코드를 탐지하기 위하여 정상어플리케이션 14개와 악성코드가 삽입된 악성어플리케이션 14개로 구성하여 데이터셋을 구성하였다. 또한 실제 모바일 환경에서 적용가능성을 검증하기 위하여 두 가지 실험환경을 구성하였다.

첫 번째 실험환경은 5개의 안드로이드 기기에서 수집된 데이터 중 80%를 학습데이터로 사용하여 학습과정을 거쳐 모델을 생성하고 생성된 모델을 바탕으로 20% 데이터를 테스트데이터로 입력하여 성능을 평가하는 방법이다. 정상데이터와 악성코드 데이터의 비율을 1:1로 거의 같은 양의 데이터로 실험하였다. 본 실험의 목적은 같은 기기에서 학습데이터와 테스트데이터를 구성하였을 경우에 각 분류기가 악성코드를 탐지할 수 있는지를 평가하는 일반적인 방법이며, 실험 1을 통해 수집한 Feature의 정확성을 검증할 수 있다.

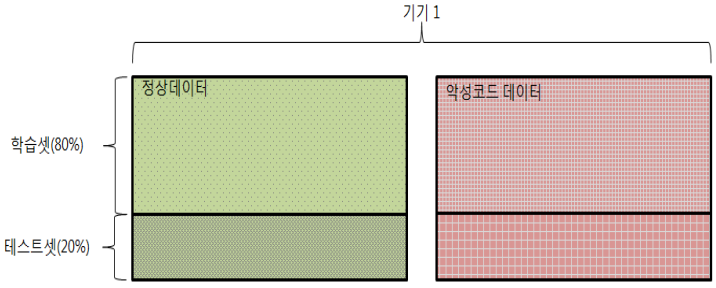

Fig. 4. 같은 기기에서 데이터를 수집하는 방법(실험 1)

두 번째 실험환경은 기기 1, 2, 3, 4에서 수집된 데이터를 학습데이터로 입력하여 학습모델을 생성하고 기기 5에서 수집된 데이터를 테스트데이터로 입력하여 평가하는 방법이다. 본 실험의 목적은 각각 다른 기기에서 학습데이터와 테스트데이터를 수집하는 방법으로써, 데이터셋을 분리하여 분류기의 성능을 평가하는 방법이다. 따라서 실험 2는 실제 모바일 환경에 적용 가능성 여부를 판단하는 실험이다.

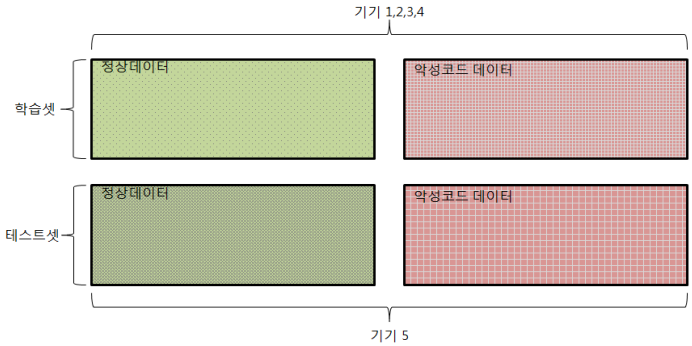

Fig. 5. 다른 기기에서 데이터를 수집하는 방법 (실험 2)

4.2 실험 결과

그림 6은 수집한 데이터를 정상/악성 어플리케이션으로 나누었을 때, 각 분류기에 대한 성능을 평가지표로 비교한 그래프이다. 두 가지 분류기 모두 높은 성능을 보이지만 TPR(True Positive Rate) 과 Accuracy 관점으로 보았을 때, RF 분류기가 더욱 좋은 성능을 보였다. 하지만 실험 1의 케이스는 같은 기기에서 수집한 데이터를 기반으로 학습/테스트 과정을 거쳤기 때문에 실제 악성코드를 탐지하는 환경에 적합한 실험방법이 아니며, SVM 분류기 또한 FPR(False Positive Rate)과 Precision 평가지표 관점에서는 더욱 좋은 성능을 보였기 때문에 각 분류기의 우수성을 검증하기 에는 어려움이 있다.

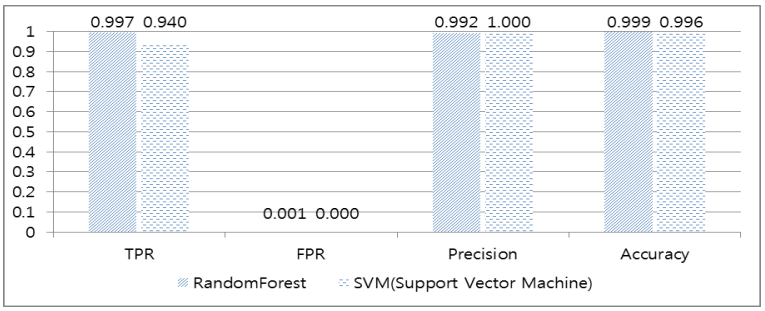

Fig 6. 각 분류기의 성능 비교 (실험 1)

그림 7은 다른 기기에서 수집된 데이터를 학습모델에 적용하였을 때, 각 분류기에 대한 평가지표 비교 그래프이다. RF는 실험 1에서 TPR/ Precision/Accuracy가 0.99 이상으로 높은 수치를 나타냈으나 실험 2에서는 TPR을 제외하고 모두 낮은 탐지 성능을 보였다. 또한 악성코드 탐지 시 중요한 성능지표로 작용되는 FPR도 0.135로 높은 수치를 나타내어 좋은 분류기라고 할 수 없다. 하지만 SVM의 경우에는 실험 1과 비슷한 결과를 보여 실험환경이 바뀌어도 높은 성능을 유지하였다. 따라서 환경에 영향을 받지 않는 SVM 분류기가 악성코드를 탐지하는데 더 우수하다고 판단할 수 있다.

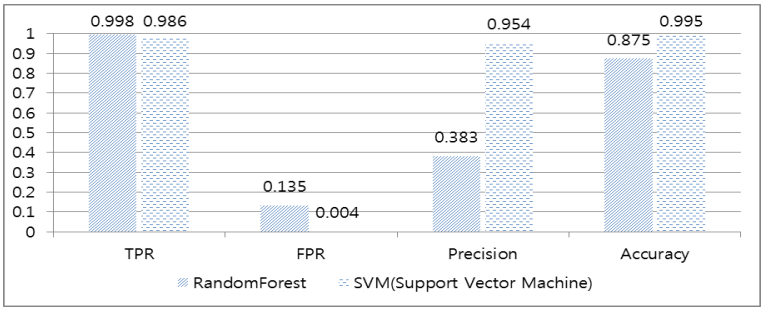

Fig. 7. 각 분류기의 성능 비교 (실험 2)

5. 결론 및 향후 연구

본 논문에서는 안드로이드 환경에서 악성코드를 탐지하기 위하여 리소스 정보를 모니터링하고 이를 SVM을 통해 탐지하는 기법을 제안하였다. 제안하는 방법의 우수성 검증을 위하여 SVM과 기존 연구에서 좋은 성능을 보인 RF와 실험결과를 비교하였고, 두 가지 실험환경 구성을 통해 실험결과를 도출 하였다. 실험1,2의 결과를 분석한 결과 SVM 분류기가 TPR = 0.986, FPR = 0.004의 결과를 보여 14개의 분석대상 악성코드를 거의 정확히 탐지하였다. 기계학습

기법은 학습데이터가 많이 주어질수록, 성능이 향상하는 특성이 있으므로 실험에서 제외된 악성코드도 제안하는 방법으로 탐지가 가능할 것으로 보인다.

안드로이드 악성코드는 수 많은 변종 및 새로운 악성코드가 발견되고 있으므로, 학습되지 않은 악성코드를 탐지할 수 있는 방안에 대한 추가적인 연구를 수행할 예정이다.

References

1. F-Secure, "Mobile Threat Report" Q4 2012.
2. Aubrey-Derrick Schmidt, Ahmet Camtepe, and Sahin Albayrak. "Static smartphone malware detection. ", In proceedings of the 5th Security Research Conference (Future Security 2010), ISBN: 978-3-8396-0159-4, page 146, 2010.
3. Thomas Bläsing, Aubrey-Derrick Schmidt, Leonid Batyuk, Seyit A. Camtepe, and Sahin Albayrak. "An android application sandbox system for suspicious software detection", In 5th International Conference on Malicious and Unwanted Software (Malware 2010) (MALWARE'2010), Nancy, France, France, 2010.
4. Xiaoming Kou, Qiaoyan Wen, "Intrusion detection model based on android", Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on, 624 – 628
5. Abhijit Bose, Xin Hu, Kang G. Shin, Taejoon Park, "Behavioral Detection of Malware on Mobile Handsets", MobiSys '08 Proceedings of the 6th international conference on Mobile systems, applications, and services.
6. Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Jan Clausen, Kamer Ali Yüksel, Osman Kiraz, Ahmet Camtepe, and Sahin Albayrak. "Enhancing security of linux-based android devices". In in Proceedings of 15th International Linux Kongress. Lehmann, October 2008.
7. Jerry Cheng, Starsky H.Y. Wong, Hao Yang, Songwu Lu, "SmartSiren Virus Detection and Alert for Smartphones", MobiSys '07 Proceedings of the 5th international conference on Mobile systems, applications and services.
8. Lei Liu, Guanhua Yan, Xinwen Zhang and Songqing Chen, "VirusMeter Preventing Your Cellphone from Spies", RECENT ADVANCES IN INTRUSION DETECTION Lecture Notes in Computer Science, 2009, Volume 5758/2009, 244-264.
9. Iker Burguera, Urko Zurutuza, Simin Nadjm-Tehrani, "Crowdroid Behavior-Based Malware Detection System", SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices.
10. Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer and Yael Weiss, "`Andromaly` a behavioral malware detection framework for android devices", JOURNAL OF INTELLIGENT INFORMATION SYSTEMS Volume 38, 2012.
11. Adam P. Fuchs, Avik Chaudhuri, and Jeffrey S. Foster, "Scan –Droid Automated Security Certification of Android Applications", 2011.
12. William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth, "TaintDroid An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation.
13. Burgesm C. j, C. , “A Tutorial om Support Vector Machines for Pattern Recognition”,”submitted to Data Mining and Knowledge Discovery, http://svm.research.bell-labs.com/SVMdochtml, (1998).
14. Hyo-sik Ham, Hwan-hee Kim, Mi-jung Choi, "Analysis of Android Malware Detection Performance using Machine Learning Classifiers", International Conference on ICT Convergence 2013, 2013.
15. Taehyun Kim, Yeongrak Choi, Seunghee Han, Jae Yoon Chung, Jonghwan Hyun, Jian Li, and James Won-Ki Hong, "Monitoring and Detecting Abnormal Behavior in Mobile Cloud Infrastructure" , 2012

IEEE/IFIP International Workshop on Cloud Management (CloudMan 2012), Maui, Hawaii, USA, April 20, 2012, pp. 1303-1310.