A Case Study on Android Malware Analysis using Hindroid

Anu Varghese.[1,2],Jagadeesha S.N.[3]

[1] Research Scholar, College of Computer Science & Information Science, Srinivas University, Mangalore, India.
[2]Assistant Professor, Department of Computer Science, MES M K Mackar Pillay College for Advanced Studies, Aluva , (Mahatma Gandhi University), Aluva, Kerala, India.
ORCIDID: 0000-0001-7779-5436;
[3]Research Professor, College of Computer Science and Information Science, Srinivas University, Mangalore, Karnataka, India. ORCIDID: 0000-0002-5185-2233;

Abstract:
Background/Purpose: The improvement in technology made the smart phone more familiar to common people and also the current situation demands it. Most of the services are digitalized nowadays. This opened up a wide field for the hackers or intruders and the rate of cyber-attacks and cyber-crimes are high. The malware has turned into a major industry as hackers grow more sophisticated and professional. The defenders and hackers are in a race to defeat each other. Machine learning based techniques has shown a higher rate in successful malware detection. In this paper discusses about Hindroid, an intelligent android malware detection system based on structured heterogeneous information network, which uses a static analysis method to identify malware. It analyses the various relationships in API calls and creates higher level semantics.
Design/Methodology/Approach: SWOT framework is being used to analyse and display the information gathered from scholarly articles, web articles, journals and other sources.
Findings/Results: Compared with other detection methods, Hindroid claims to outperform with 98.6% accuracy. It claims 99.01% detection rate compared to other security products like clean master, lookout, Norton etc
Originality/Value: This study gives an overview of Android Malware Analysis based on the various data collected.
Paper Type: Research Analysis based on Case Study

Keywords: Malware, Hindroid, API

1. INTRODUCTION:

With the improvements in hardware and software technology, there is a rapid increase in the usage of smart phones. Android mobile phones are widely used. New statistics says that by 2027 the smartphones will account 76.9% market share (Fig.1). Almost all services like bill payments, recharging, online shopping etc. are digitalized. It opened a new field of thefts for hackers, thereby the number of cybercrimes and criminals has increased rapidly. Malware attacks are at its peak during the pandemic as most of the services were online like online education for the students, work from home for employees, bill payments, online shopping,etc. Android has always been the main target of hackers as it is an opensource software. They use malicious code for transmitting the confidential information, to take up the control of the device, for financial gain, political interest, revenge etc. The malwares create various threats like stealing user's credentials, pushing unwanted apps or advertisements etc, to the smartphone users. Due to the immense growth in the variety of Android malware, the researchers divide the malwares into various families to assist the malware analysis. Malwares are classified based on their similarities in behaviour. The malware defenders need to be more vigilant as the capabilities and knowledge of attackers increases faster.

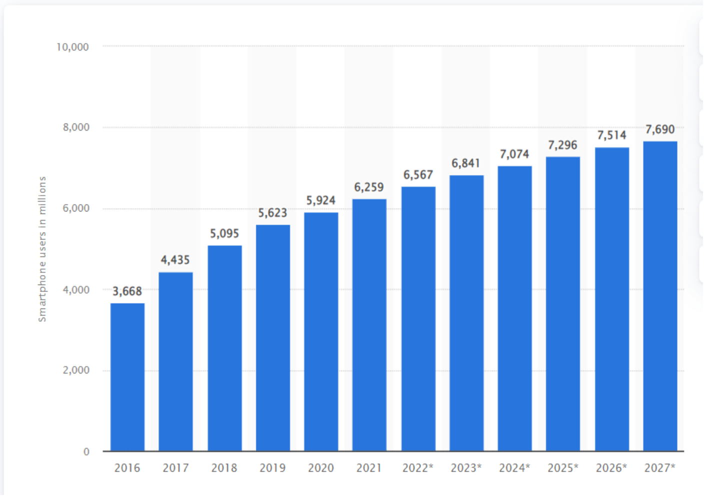

Fig.1: shows number of Smartphone users from 2016-2027. (Source: statistica.com)

2. OBJECTIVES OF THE CASE STUDY:

The main objectives/aim of this study are:

1. To understand the android malware analysis methods.
2. To study the malware detection method using Hindroid
3. To study the effectiveness of heterogeneous graphs in malware analysis.
4. To identify the research gap

3. RELATED WORKS :

Numerous studies have been conducted to evaluate malware attacks.. In order to accelerate the malware analysis, the researchers divide the malwares into various families, based on the similarities in behaviour, to assist the analysis. For example, some silently transmits the confidential information to the remote server, some allows the hackers to control the device remotely. Table 1 summarises relevant research on android malware analysis based on the API Calls and apps, as well as the field, subject, and citations of the work. The following table 1 shows the related works:

Table 1: Related Works

Sl. No.	Work	Type of Analysis	Features Used	Reference
1.	DroidDolphin	Dynamic	APE	[7]
2.	CrowDroid	Dynamic	API System Calls	[9]
3.	CopperDroid	Dynamic	Operating system interactions & intra and inter process communication	[5]
4.	CopperDroid	Static	Analysis of internal components of an App.	[5]
5.	DroidMat	Static	API Calls, Permission intent messages	[8]
6.	DroidMiner	Static	API Calls (associative classifier)	[6]
7.	αCyber	Static	API Calls	[18]
8.	DroidDelver	Static	API Calls	[4]
9.	AiDroid	Static	API Calls	[23]
10.	IntDroid	Static	API Calls	[24]

4. METHODOLOGY :

Journals, conference papers, media articles and various public records were used to gather materials and data for this case study.

5. ANDROID MALWARE ANALYSIS– AN OVERVIEW:

Android is an open-source mobile operation system. It is based on a Linux kernel. The applications are written in Java and are transformed into a slightly different format known as Dalvik. The apps are then run in the Dalvik virtual machine which provides a layer of abstraction over the real hardware. Android applications are packed in the format .apk, which is a ZIP archive containing the AndroidManifest.xml, resources like media files, the actual code as classes.dex and some other optional files.[25]
Malware is any piece of software that is harmful to the systems- worms, trojans, viruses etc. Malware analysis is the practical way of understanding the behaviour and purpose of malicious code or suspicious URLs.
Android malware analysis can be classified as:

1. Static analysis
2. Dynamic analysis
3. Hybrid analysis

5.1 Static Analysis

Static analysis is also called Code analysis. It is performed by viewing the software code of the malware and walking through it, instructions by instructions. It examines the files for signs of malicious intent without running the code. It can be useful to identify malicious infrastructure, libraries, or packed files. Static analysis is signature -based analysis. It is also similar to statistical based analysis. It involves virus scanning, fingerprinting etc. Reverse engineering is used for static analysis.

5.2 Dynamic Analysis

Dynamic analysis improves the static analysis in terms of result delivery. Dynamic analysis also known as behavioural analysis. In dynamic analysis, the malicious code is executed in a safe and controlled environment called sandbox and studies how the malware behaves when executed. It is more difficult to perform dynamic analysis as they may do unexpected changes to the system and also most of the malware can hide their run time activities to an extent.

5.3 Hybrid Analysis

Hybrid analysis combines the techniques of both static and dynamic analysis and thereby covers each other's drawbacks. In other words, it analyses the signature of the malware and then continue the analysis by combining with various behavioural patterns. Hybrid analysis helps to overcome the shortcomings of both static and dynamic analysis.

Android Malware Analysis Tools:

There are various android malware analysis tools. They are classified based on the techniques used for the analysis like static analysis, dynamic analysis, reverse engineering, etc. Most popular ones are classified and listed below:[26]

a) Static Analysis

Tool	Description
ClassyShark	Standalone android apps binary inspection tool
StaCoAn	Mobile application static code analysis tool
SmaliSCA	Smali static code analysis
maldrolyzer	Simple framework to extract "actionable" data from Android malware (C & Cs, phone numbers, etc)
Argus-SAF	Andrid application static analysis framework
DroidRA	Taming reflection to support whole-program analysis of android apps
Androwarn	Static code analyzer for malicious Android applications
PScout	Android permission mapping tool
APK-MiTM	CLI application that automatically prepares Android APK files for HTTPS inspection
Super Android Analyzer	Secure, unified ,Powerful, and Extensible Rust Android Analyze

b) Dynamic Analysis

Tool	Description
AppMon	Automated framework for monitoring and tampering system API calls based on Frida
DroidBox	Dynamic analysis of Android apps
ConDroid	Execute specific code locations with no app manual interaction
Wireshark	Network analysis tool
tcpdump	Network analysis tool
MiTMProxy	An interactive SSL/TLS-capable intercepting HTTP proxy (great for HTTPS inspection)
Burp Suite	The free web proxy for any browser, system, or platform
INetSim	Internet Services Simulation Suite

c) Reverse Engineering

Tool	Description
smali/baksmali	DEX disassembler
AndroGuard	Python-based tool for Android application reverse engineering
Apktool	Tool for disassembling, rebuilding, and reversing in an automated matter
Dex2Jar	DEX to JAR conversion tool
JD-GUI	Graphical utility that displays Java sources from CLASS files

JadX	Dex to Java decompiler (command line and GUI)
Krakatau	Python-based decompiler and disassembler
Procyon	Command-line Java-based decompilation tool
CFR	Command-line Java-based decompiler and disassembler
ndk-gdb	GDB Android debugging
Frida	Dynamic instrumentation framework
Dwarf	Full-featured multi-arch/os debugger built on top of PyQt5 and Frida
JEB Decompiler	Android decompiler
IDA Free/Pro	Disassembler and debugger
radare2	Free and open source disassembler and debugger
Cutter	GUI for radare2
Binary Ninja	A New Type of Reversing Platform

d) Unpacking & Deobfuscation

Tool	Description
Quark Engine	Obfuscation-Neglect Android malware scoring system
DeGuard	Online Android deobfuscation tool
Simplify	Generic Android deobfuscator

e) Forensics

Tool	Description
Andriller	Utility with a collection of forensic tools for smartphones
Mem	Android process memory dumper
dd	Hard drive and SD card forensics acquisition tool
Autopsy	Hard drive and SD card forensics analysis tool
LiME	Memory acquisition tool
dwarfdump	Linux profile creation for Volatility
Volatility	Memory forensics analysis framework

f) Other

Tool	Description
MobSF (Mobile Security Framework)	Malware analysis and security assessment framework capable of performing static and dynamic analysis.
MARA_Framework	Tool that puts together commonly used mobile application reverse engineering and analysis tools.
Cuckoo Sandbox	Free and open-source automated malware analysis sandbox.
Cuckoo-Droid	Cuckoo Sandbox extension for automated Android malware analysis
Android Tamer	VM/Live OS for Android security research and analysis
Vezir-Project	VM/Live OS for mobile security research and analysis

Some popular malwares are:-[27]

1. **Viruses**-- A virus is a type of malware that, when executed, self-replicates by modifying other computer programs and inserting their own code.
2. **Trojan Horse**-- A trojan horse or trojan is any malware that misleads users of its true intent by pretending to be a legitimate program.
3. **Worms** -- A computer worm is a self-replicating malware program whose primary purpose is to infect other computers by duplicating itself while remaining active on infected systems.
4. **Rootkit**-- A rootkit is a collection of malware designed to give unauthorized access to a computer or area of its software and often masks its existence or the existence of other software.
5. **Ransomware**-- Ransomware is a form of malware, designed to deny access to a computer system or data until ransom is paid.
6. **Keylogger**--Keyloggers, keystroke loggers or system monitoring are a type of malware used to monitor and record each keystroke typed on a specific computer's keyboard.
7. **Grayware**—The term grayware describes unwanted applications or files that aren't malware but worsen the performance of the computer and can cause cybersecurity risk.
8. **Fileless malware**-- Fileless malware is a type of malware that uses legitimate programs to infect a computer.
9. **Adware**-- Adware is a type of grayware designed to put advertisements on screen, often in a web browser or popup. It is the most popular malware for mobile phones and least harmful malware.
10. **Malvertising**—It uses advertisements to spread malware. They inject malicious or malware-laden advertisements into legitimate advertising networks and webpages.
11. **Spyware**--It is malware that gathers information about a person or organization, with or without their knowledge, and sends the information to the attacker without the victim's consent.
12. **Bots/botnets**-- A bot is a computer that is infected with malware that allows it to be remotely controlled by an attacker. Botnet is the collection of bots. It is popular in spreading malwares like ransomware, keylogging etc.
13. **Wiper**—A wiper malware is a type of malware which erases the user's data and ensure that it can't recover it.

14. **TrickBot**—It is banking trojan which steals the credentials, data and personal information. It deactivates the antivirus tools and cybersecurity measures.

7. DIFFERENT TYPES OF MALWARE ATTACKS:

a) **Vulnerabilities**: A software security flaw can be exploited by malware to obtain unauthorised access to the computer, device, or network.

b) **Backdoors**: a security vulnerability, whether intentional or accidental, in networks, software, hardware, or systems.

c) **Drive-by downloads**: Unintentional software downloads can happen with or without the end user's knowledge.

d) **Homogeneity**: The likelihood of a successful worm spreading to further computers increases if every system is using the same operating system and is linked to the same network.

e) **Privilege escalation**: An instance where a hacker gains elevated access to a computer or network and then makes use of it to launch an attack.

f) **Blended threats**: Malware packages that contain traits from other malware types are more difficult to identify and stop because they can take use of a variety of vulnerabilities.

8. HINDROID IN MALWARE ANALYSIS [2] :

The malware program also does not need to be run for it to be detected. By analysing its opcode sequences and control flow graphs, we can determine whether it looks like known malware. Machine learning techniques are very successful in identifying the malwares. The defenders and research use different features like API calls, permissions etc for classification of malicious and benign apps.

The HinDroid uses a static analysis method to identify malware. It employs heterogeneous graphs for the analysis. It only analyses the code instead of running a more dangerous dynamic analysis method to monitor the app's behaviour. In essence, we have to take the code out of an Android app. The .apk (Android Application Package) file extension is used for packaging and distributing Android programmes. The unreadable dex code files contained in this package can be decompiled into smali code, which ApkTool can then read and handle. To determine whether an app is malicious or not, parse the smali codes to obtain helpful information.

This paper is focused on is API calls in smali codes. They are a layer of function calls that are abstract and can be inserted into code to perform any task the developer desires. Even though there are low-level APIs for string concatenation and string to integer conversion, the APIs that should alert us are those that require system rights or send HTTP queries to a suspicious IP address.

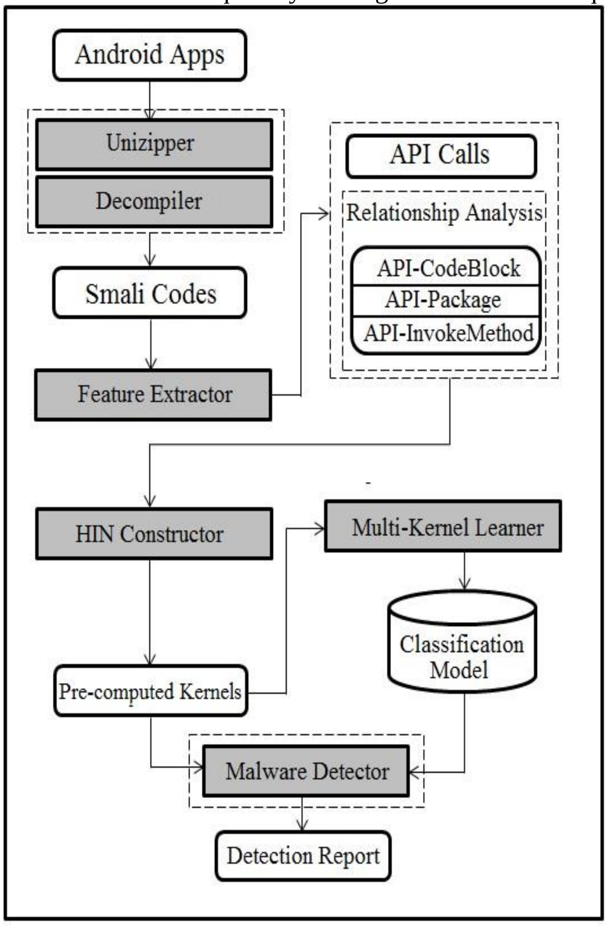

Fig.1: System Architecture of Hindroid.

Interpretation:

The android apps are unzipped and decompiled initially to get the smali code. From this smali code the needed information is extracted to construct a Heterogeneous Information Network with a graph representation. The graph consists of two types of nodes and four types of edges. Every node inside the HIN represents either an App or an API. Every app node must not directly connect to other app nodes and is only allowed to connect to API nodes. The related APIs can be located inside the smali code of the App, according to this sort of edge. Other types of edges only exist between APIs. These edges are characteristics of smali code.

This model calculates similarities between apps and feeds the similarities to a SVM to form a decision boundary between two data classes. It constructs a Heterogeneous Information Network (HIN) to capture the relationships between apps and between APIs.
The network consists of two node types and four edge types.

- Edge type **A** connects apps to APIs if the API is used by that app.
- Edge type **B** connects every pair of APIs that are co-appeared inside the same code block in every app.
- Edge type **P** connects every pair of APIs that are from the same library (package).
- Edge type **I** connects every pair of APIs that are using the same invoke method for present in every app.

Every pair of APIs that co-occur in a code block within an application are connected by edge type B. Every pair of APIs from the same library (package) in every app is connected by an edge type P. Every pair of APIs that each use the identical invoke method and are present in every app are connected by edge type I.
Each type of these edges will be represented by an adjacency matrix. To calculate the similarity, we formalize it as the number of common features that are both present in the HIN. If the similarity between two apps is high, there may be something to be said for it. We define this common feature as the number of metapaths between apps. A metapath starts from an app and ends with an app and it goes through a symmetric node path in the HIN.
For example,
A-A means how many APIs are common within a pair of apps.
A-P-A means how many pairs of APIs (one API from a<sub>i</sub>, one API from a<sub>j</sub>) that use a common library.
To generate the A adjacency matrix, we used a unique ID finder (src.utils.UniqueIdAssigner) that assigns an unique integer ID to each unique API. We then keep a set of API IDs that appear in an app for each app and save these sets as a sparse matrix. The table of the one-to-one association between API and ID is also saved.
The number of meta-paths between the apps can then be calculated by multiplying the A matrix to other adjacency matrices and their appropriate inverses.
Each time the prediction job is run, a new copy of the whole HIN graph is created. Later, we will divide the graph into four adjacency matrices that represent the four different kinds of edges. A matrix of adjacencies will serve as a representation for each type of these edges. The number of shared features that are both present in the HIN is how we formalise the similarity and calculate it. There might be a case for it if two apps have a lot of similarities to one another. The number of meta routes between apps is how we determine this common characteristic. A meta path traverses a symmetric node path in the HIN as it travels from an app to an app and back again.
HinDroid considers that these meta routes signify a special characteristic shared by all programmes. The functionality or aim of two apps should be identical if they both contacted the same exact APIs within a block of source code, like in the case of the meta route A-B-A. Using this insight, we can observe that if an app is more similar to malware in terms of high similarity, we can be more certain that the mystery software should also be malware. In order to create these meta pathways, adjacency matrices are multiplied, and the resulting matrix is employed as a Gram matrix by the SVM method to determine the boundary between the two classes.

9. SWOT ANALYSIS:

a Strengths:

- Hindroid analyses the relationships between the apps and API calls ,and also the relationship among the API calls.
- Heterogeneous graphs used to represent the relationships.

- It uses PathSim an extension to Dot product to represent the similarity defined on HIN. To identify the crucial pathways for entity grouping, PathSim is employed.

b Weaknesses:

- Improvements must be made to the classification's generalisation property.
- It uses the supervised learning approach for the meta path weighting mechanism.
- It fails for the malware familial classification.

c Opportunities:

We can improve the quality of Hindroid by improving the generalization property and by using unsupervised learning for metapath weighting.

d Threats:

- Malware detection becomes more complex as the style of attacks changes rapidly.
- It needs to defend the HG Data from adversarial attack to enhance the HG based classifiers.

10. CONCLUSION:

The defenders and attackers are in a race as the style of attacks varies fastly.The capabilities and knowledge of the attackers improves and increases rapidly [18].The Hindroid uses HIN(Heterogeneous information network) to represent the relationships between Apps and APIs and metapaths are used to link the apps. Multikernel learning is used to aggregate the similarities. The experimental results promises that Hindroid outperforms the other detecting techniques and mobile security products. Hindroid has been incorporated into the Comodo Mobile Security product's scanning feature. [2]. The malware familial classification is lacking using Hindroid. Further studies are going on the basis of heterogeneous graphs and shows it success in detecting malwares.

REFERENCES:

1. Droid Box. https://github.com/pjlantz/droidbox
2. Shibu Hou, Yanfang Ye, Yangqiu Song, and Melih Abdulhayoglu. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. https://www.kdd.org/kdd2017/papers/view/hindroid-an-intelligent-android-malware-detection-system-based-on-structure
3. Android malware analysis tools https://malwareanalysis.co/resources/tools/android
4. Shifu Hou, Aaron Saas, Yanfang Ye, and Lifei Chen. 2016. DroidDelver: An Android Malware Detection System Using Deep Belief Network Based on API Call Blocks. https://www.researchgate.net/publication/309173157_DroidDelver_An_Android_Malware_Detection_System_Using_Deep_Belief_Network_Based_on_API_Call_Blocks
5. K. Tam, S. Khan, A. Fa‹ ori, and L. Cavallaro. 2015. CopperDroid: Automatic Reconstruction of Android Malware Behaviors. https://www.ndss-symposium.org/wp-content/uploads/2017/09/02_4_1.pdf
6. Chao Yang, Zhaoyan Xu, Guofei Gu, Vinod Yegneswaran, and Phillip Porras. 2014. DroidMiner: Automated Mining and Characterization of Fine-grained Malicious Behaviors in Android Applications https://link.springer.com/chapter/10.1007/978-3-319-11203-9_10
7. Wen-Chieh Wu and Shih-Hao Hung. 2014. DroidDolphin: A Dynamic Android Malware Detection Framework Using Big Data and Machine Learning. https://doi.org/10.1145/2663761.2664223
8. D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu. 2012. DroidMat: Android Malware Detection through Manifest and API Calls Tracing https://scholar.google.com/citations?user=ZTqt02EAAAAJ&hl=en
9. Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. 2011. Crowdroid: Behavior-based Malware Detection System for Android. https://www.researchgate.net/publication/245022829_Crowdroid_Behavior-Based_Malware_Detection_System_for_Android
10. N. Peiravian and X. Zhu. 2013. Machine Learning for Android Malware Detection Using Permission and API Calls. https://www.researchgate.net/publication/262221913_Machine_Learning_for_Android_Malware_Detection_Using_Permission_and_API_Calls
11. Lingwei Chen, Shifu Hou, and Yanfang Ye. Securedroid: Enhancing security of machine learningbased detection against adversarial android malware attacks https://www.researchgate.net/publication/321505502_SecureDroid_Enhancing_Security_of_Machine_Learning-based_Detection_against_Adversarial_Android_Malware_Attacks
12. Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: Projected metric embedding on heterogeneous networks for link prediction https://www.kdd.org/kdd2018/accepted-papers/view/pme-projected-metric-embedding-on-heterogeneous-networks-for-link-prediction
13. Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye, and Melih Abdulhayoglu. Gotcha-sly malware!: Scorpion a metagraph2vec based malware detection system https://www.kdd.org/kdd2018/accepted-papers/view/gotcha-sly-malware-scorpion-a-metagraph2vec-based-malware-detection-system
14. Tao-Yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. https://www.researchgate.net/publication/320883245_HIN2Vec_Explore_Meta-paths_in_Heterogeneous_Information_Networks_for_Representation_Learning
15. Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. https://www.researchgate.net/publication/318072021_A_Survey_on_Malware_Detection_Using_Data_Mining_Techniques
16. Yanfang Ye, Tao Li, Shenghuo Zhu, Weiwei Zhuang, Egemen Tas, Umesh Gupta, and Melih Abdulhayoglu. Combining file content and file relations for cloud based malware detection. https://www.researchgate.net/scientific-contributions/Melih-Abdulhayoglu-2081899254
17. Deqing Zou, Yueming Wu, Siru Yang, Anki Chauhan, Wei Yang, Jiangying Zhong, Shihan Dou, and Hai Jin. 2021. IntDroid: Android Malware Detection Based on API Intimacy Analysis. https://dl.acm.org/doi/abs/10.1145/3442588
18. Shifu Hou, Yujie Fan, Yiming Zhang, Yanfang Ye, Jingwei Lei, Wenqiang Wan, and Jiabin Wang, Qi Xiong, Fudong Shao. 2019. αCyber: Enhancing Robustness of Android Malware Detection System against Adversarial Attacks on Heterogeneous Graph based Model https://dl.acm.org/doi/10.1145/3357384.3357875
19. Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. https://www.researchgate.net/publication/220538331_PathSim_Meta_Path-Based_Top-K_Similarity_Search_in_Heterogeneous_Information_Networks
20. Aleksandar Bojcheski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings. https://www.researchgate.net/publication/330008793_Adversarial_Attack_and_Defense_on_Graph_Data_A_Survey

21. Haipeng Cai, Na Meng, Barbara Ryder, and Daphne Yao. 2019. Droidcat: Effective android malware detection and categorization via app-level profiling.
https://ieeexplore.ieee.org/document/8519742
22. Lingwei Chen, Shifu Hou, Yanfang Ye, and Shouhuai Xu. 2018. Droideye: Fortifying security of learning-based classifier against adversarial android malware attacks
https://www.researchgate.net/publication/328525416_DroidEye_Fortifying_Security_of_Learning-Based_Classifier_Against_Adversarial_Android_Malware_Attacks
23. Yanfang Ye, Shifu Hou, Lingwei Chen, Jingwei Lei, Wenqiang Wan, Jiabin Wang, Qi Xiong, and Fudong Shao. 2019. Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection
https://www.ijcai.org/proceedings/2019/576
24. https://tsumarios.github.io/blog/2022/09/24/android-malware-analysis-lab
25. https://malwareanalysis.co/resources/tools/android
26. https://www.upguard.com/blog/types-of-malware