RESEARCH ARTICLE OPEN ACCESS

System call anomaly detection

Marin Aranitasi*, Denis Veliu** Valma Prifti***, Anisa Gjini****

* *(Department Basics of Informatics, Polytechnic University of Tirana, Albania)*
** (Department of Finance, Metropolitan University of Tirana)
***(Department of Production Management, Polytechnic University of Tirana)
****(Department Basics of Informatics, Polytechnic University of Tirana, Albania)

ABSTRACT
Traditional detection techniques use the signature of the tested application and compare it with already stored signatures of previously captured viruses to detect malicious applications. This method is widely used in static devices and now is being used even in mobile devices, mainly the ones that use the Android OS. The malware detection avoidance techniques have improved radically making the traditional techniques obsolete. To solve these problems, this paper proposes a new method, that will increase the detection of malicious applications on Android OS devices. This will happen by monitoring the system calls of the mobile device and detecting any anomaly in their usage. We present the mathematical foundation of our new method, which will be used to learn the normality of the usage of the system calls and their parametrization. Finally, we present future challenges and steps.

Keywords – Malware, System call log, Application, Classification

---

Date of Submission: 08-01-2024 Date of acceptance: 20-01-2024

I. INTRODUCTION

In recent years the popularity of mobile devices has increased exponentially. Android is the "king". He has over 2.8 billion active users. According to Stat counter [1] Android has the 71.62% of the market share compared to 27.61% of IOS.

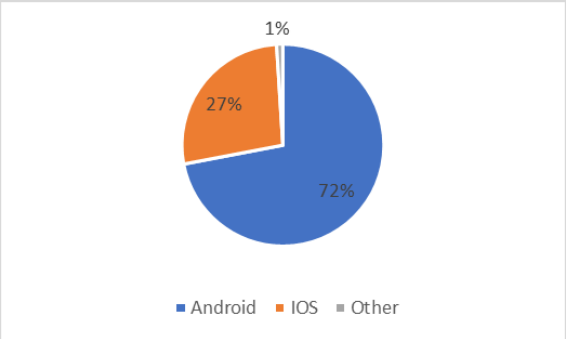

Fig.1 Mobile OS Market Share April 2022

This huge use of Android devices has attracted the attention of malware writers. Kaspersky had prevented 14,465,672 malware, adware, and riskware attacks on 2021 [2]. From those, 24604 packages were mobile banking Trojans.

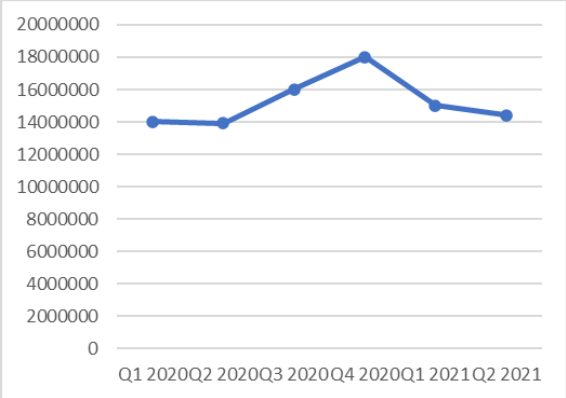

Fig. 2 Number of attacks targeting users of Kaspersky mobile solution 2020-2021

In the mobile world, there are different types of threats. Table 1 outlines some of the most important that are detected by Checkpoint [3] and Kaspersky [4].

TABLE I
Mobile OS threats

Source	*Kaspersky*	*Checkpoint*
Threat	Data leakage	Malicious Apps and websites
	Unsecured WIFI	Mobile Ransomware
	Network spoofing	Phishing
	Phishing Attacks	Man in the middle
	Spyware	Advanced Jailbreaking and rooting techniques
	Broken Cryptography	Device OS exploits

In the mobile world, the variety of cyber-attacks is huge. One of them includes the "Agent Smith" campaign according to Checkpoint [3,5,6]. The apps from this campaign, have been downloaded to 25 million Android devices and were distributed through third-party app stores by a Chinese group. About 300,000 devices were infected in the U.S. The malware was able to copy popular apps on the phone, including WhatsApp and the web browser Opera, inject its malicious code, and replace the original app with the infected version, using a vulnerability in the way Google apps are updated. The infected apps would work just fine, which hid the malware from users. U.S. Department of Homeland Security (DHS) and the Federal Bureau of Investigation (FBI) in a joint report [7][8] detailed the dangers of two Trojan malware packages. It's believed to be the work of Hidden Cobra, also known as the Lazarus Group – threat actors who are connected to the North Korean government. Both malware strains – called Haardrain and Badcall– can install a remote access tool (RAT) payload on Androids. Windows systems are then drawn in as proxy servers, which disguise command and control communications.

As a way to mitigate these concerns, this paper does not add any technological layer in the mobile OS, nor it does not write any new security policy, but it presents a new method for detecting system usage anomalies. This method after observing the usage of the mobile device of different subjects, which here we will assume are normal users, forms a pattern of normality. Every action or event that is outside this pattern triggers a warning flag. The main technical element that we use to form the normal behavior, are the system calls of the mobile kernel.

The paper is organized as follows. Section II describes the Related work in this field, in section III, we present our solution, The section is divided into 3 subsections: the malware used, the system calls (that is our main element of monitoring), and the mathematical foundation of our solution. The last section are the conclusions of this paper.

II . RELATED WORK

The scientific world has made a lot of effort to find solutions for the increasing of Android devices [9].

Chen Da et al. [10] developed a new detection method of the malware that is based on system calls frequency. They use the characteristics of random forest algorithms and data that are gained previously to set up an optimal training model. Their method detected more than 93% of malwares.

Bathia et al. [11] proposed an approach that uses dynamic analysis on Android applications. They built a system that collects the system call traces during their execution. This data is then analyzed to classify the different behaviors of Android applications.

Vikas et al. [12] proposed a behavior-based approach to detect malicious nature of applications in Android. They used events and behavioral activities of an application to generate signature, which then is matched with signature database for detection.

Yan et al. [13] has made a survey on dynamic mobile malware detection approaches. The authors have summarized a large number of criteria and performance evaluation measures of mobile malware detection. In the end, they figured out the open issues in this research field.

Hou et al. [14] has presented a dynamic analysis method named Component Traversal, that automatically executes each routine of the Android apps. Based on the extracted system calls, they built the weighted graphs, and then these graphs are used by a deep learning framework to detect the Android malwares.

Vidal [15] has as a main goal to prevent the installation of malicious software on the victim's system. So he monitored only the system calls during the boot process of the recently installed

applications. This reduces the information that will be used in the analysis. They used three processing layers: monitoring, analysis, and decision-making.

III. SOLUTION

A. Malware

According to Cisco [16] malware is intrusive software that is designed to damage and destroy computers and computer systems. Malware is a contraction for "malicious software." Examples of common malware include viruses, worms, Trojan viruses, spyware, adware, and ransomware.

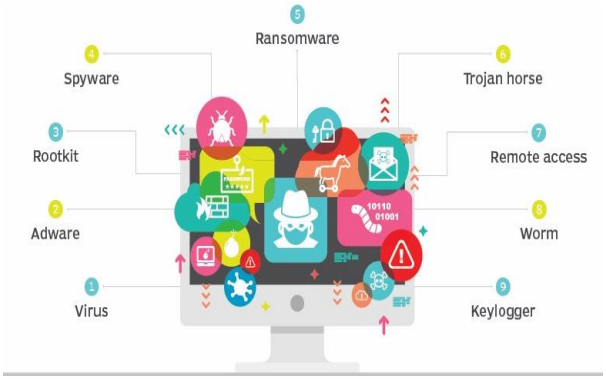

Fig. 3. Types of Malware [17]

In this article, we will focus more on mobile malware, although the solutions presented here can be exported to other computer environments.

Android malware has evolved over the years. The first Android trojans were detected in 2010 and were called DroidSMS, a Fraud SMS app that sent fraud SMS to a premium rate number, and FakePlayer, a Trojan that attempts to send a message without the user's approval to a present number. Also Fig. 4 presents a timeline of the Android evolution from 2010 to 2018 [18].

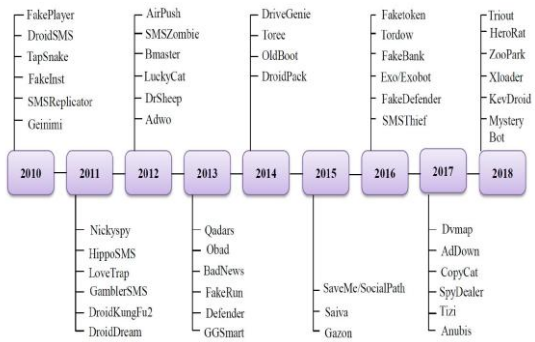

Fig. 4 Android Malware Evolution

Today the number and the variety of malwares, as mentioned earlier, is huge. Their classification is not as simple. After reviewing different research, we decided to use [19] as a base for selecting the malware family to observe and study. The authors have tried to classify the malwares and group them by characteristics into families. Those are presented in fig 5.

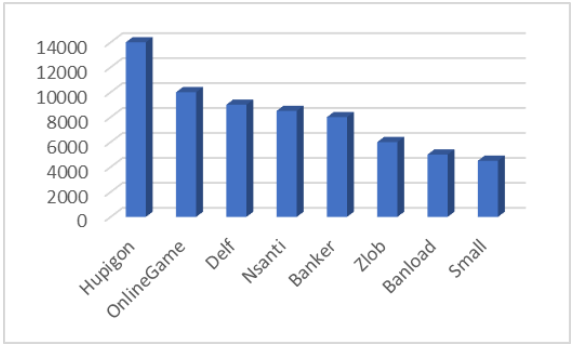

Fig.5 Malware families

This picture shows the top 8 malware families. All have more than 2000 samples.. Some families to mention are:

Banload is a family of Trojans responsible for stealing banking credentials. Buzus is also an information-stealing family and bifrose is a combination of backdoor and trojan allowing remote access to the attacker which is also used for information stealing in most cases.

Currently exist two methods for malware detection. First is static analysis. In this approach, the study of the malware is done without executing the code. Here the functionalities of the application are checked by disassembling and analysing the code. Some of the techniques used include de-compilation, pattern matching, and decryption [20].

The second technique is dynamic analysis. This method observes the behavior of the application while it is being executed. In this case, we have to be careful to run the app in a sandbox environment, in order not to infect any device [21]. The downside of the techniques is resource consumption. Nevertheless, this is the technique that is widely used among researchers, and this is the technique that we are going to use in this paper. So we will observe the behavior of the mobile device, by monitoring the usage of the system call for each user application.

After studying the malware evolution and specifically the malware families and their characteristics we choose the following set of malware to be installed and monitored.

TABLE II

Malwares

Type	Name
Ransomware	Charger
	Jisut
	Simplocker
Adware	dowgin
	feiwo
	kemoge

Charger was found embedded in an app called Energy Rescue. The infected app steals contacts and SMS messages from the user's device and asks for admin
permissions.

```
public static string Fingerprint = "{2D592824-48DE-49F8-8F96-A40B3904C794}";

public static string OperatingSystem = Utility.GetOSInfo();

private static string MachineName = Environment.MachineName;

private static string BotVersion = Utility.GetVersionInfo();
```

Fig. 6 Charger malicious code

The ransom demand for 0.2 Bitcoins. Charger checks the local settings of the device and does not run its malicious logic if the device is in Ukraine, Russia, or Belarus.

The adware malware that we choose has the same basic characteristic, the one that tries to display ad banners during the execution of another program. For each of the malwares, we have selected different samples. These were collected by different antiviruses and antimalware on different devices. This will help us in having different infections of the phone to analyze and test our theory.

B. System calls

According to IBM [22] a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system.

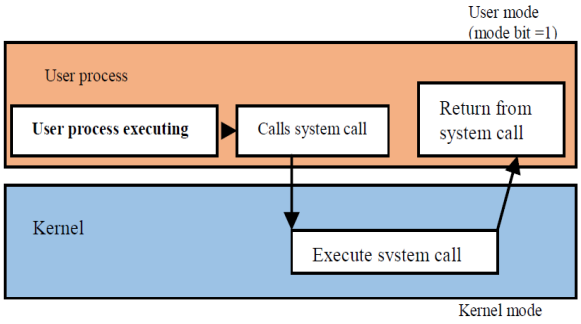

Fig. 7 System calls [23]

When a user program needs something, the operating system generates a system call. System call provides the services of the operating system to the user programs using the Application Program Interface (API). All programs needing resources must use system calls. This includes malicious programs, as their target is to use our resources for bad intentions. And that is why we choose to use the system calls as our main element for malware detection.[22]

There are 5 different categories of system calls [9]:

1. Process control. This type of calls deal with processes. Some of the services that are provided using these calls are:
    a. Create a process
    b. Abort forcefully/normally a running process
    c. Allocate memory to a process.
2. File management**.** These calls are used for file manipulation. The most important services are:
    a. Creating a file
    b. Reading a file
    c. Writing into a file
3. Device management. These calls are responsible for device manipulation. Some of the services are:
    a. reading/writing from/into device buffers
    b. Obtain or modify a specific device attribute
    c. Detach a device from the processor during a system call
4. Information management system call exists for transferring information between the user program and the operating system. Some of the services are:
    a. Obtain system time/date
    b. Configure system data

   c. Set the characteristics of an operating system process
5. Communication. These calls are used for interprocess communication. The services are :
   a. Create/terminate data connection
   b. Send messages
   c. Connect to a remote device via network

As I mentioned earlier we have studied the characteristics of different malware families. There we saw that on the malware data set that we chose, tries to steal info or to change file parameters in order to get privileges. So, we decided that in this paper we are going to monitor 2 categories of system calls

1. Communication
2. File management

In the file management category, we have chosen to monitor and after analyse open (), read (), write (), close (). For communication category we have chosen we accept (), socket (), connect () system calls.

TABLE III
System Calls

Types of system calls	System calls
Communication	accept (), socket (), connect ()
File management	open() read() write() close()

C. Mathematical foundation of the solution

Using the system calls needs a set of preconditions and assumptions. The first is that the system calls that we will monitor will behave in a normal environment. By normal we mean that they will be not part to race conditions, but they will run one at a time from beginning to the end. Also, the set of calls that we choose is assumed to be exhaustive for the purpose of the conclusions of this study. Based on these assumptions a normality pattern can be built by analyzing the past, in other words the system call log. This log file will be extracted using 'strace' Linux command that is integrated in a script, for automation purposes.

Also, this model must take into consideration normal changes to the system that have a probability of being interpreted as abnormal behavior. This includes operating system updates, specific user software updates or just instant change of habits un user application usage. So, the system will be adaptive, and it will alert the user of any abnormal activity outside the normal behavior

A call is by nature a discrete variable that can happen in an interval of time with an estimated probability. For this, if we have just a single call, the probability distribution assigned can be a Bernoullian that assigns a 1 if the call happens with probability p and a 0 with probability 1-p i the call doesn't happen. Thus, since we have more than one possible call, we can treat them as a binomial distribution. So, if X is the number of the calls, and the maximum number of expected call are n, the binomial distribution will be:

$X \sim Bin(n, p)$ where $X=0,1,2,3\ldots\ldots n$

where the probability is calculated as follow. $P(X=x) = {}_nC_x * p^x(1-p)^{n-x}$

If we defined in this way, we know that the probability that a call happens in a certain moment is almost zero. For that we have to consider the number of the calls in an interval of time. In other words, if we do the limit of the binomial distribution for p goes to zero and n goes to infinity, we will obtain a Poisson distribution.

$$P(x = k) = \frac{e^{-\lambda}\,\lambda^{-k}}{k!} \quad k = 0,12,3,\ldots\ldots \text{ for}$$

where λ is the expected value or the mean, i.e. the number of monitored calls in an interval of time. This mathematical model fits wells for the independent events arrival model, can adequately fit the normal session traffic pattern.

The Poisson cumulative distribution function for the given values x and λ is

$$F(x,\lambda) = \sum_{k=0}^{x} \frac{e^{-\lambda}\ \lambda^{-k}}{k!} \quad .$$

In order to work for any distribution, without adapting the lambda λ for the interval of time (or space) , the better switch should be with the Poisson process which is very similar.

$$P(x = k) = \frac{e^{-\lambda t}\ (\lambda t)^{-k}}{k!} \quad \mathrm{k} = 0,12,3 \ldots.$$

and the cumulative

$$F(x,\lambda) = \sum_{k=0}^{x} \frac{e^{-\lambda t}\ (\lambda t)^{-k}}{k!}$$

Also, from the mathematical point of view λt is a Lebesgue measure, so we can benefit for its properties. The distance between two consecutive calls of a Poisson process on the real line will be an exponential random variable with parameter $1/\lambda$. This implies that the calls have the memoryless property: the existence of one event existing (call) in a finite interval does not affect the probability (distribution) of other calls existing, but this property has no natural equivalence when the Poisson process is defined on a space with higher dimensions.

The problem is to fix a threshold for which we a consistent number of calls to trigger the alarm so, if T is the threshold:

$$F(T,\lambda) = \sum_{k=0}^{T} \frac{e^{-\lambda t}\ (\lambda t)^{-k}}{k!} \geq 0.99$$

or by using the complement probability, the number of false alarms that we should ignore should be:

$$(1- \quad F(T,\lambda)) < 0.01$$

So, to find the soil T of the real calls, its complement, false calls, we should do a goal optimization (fmincon function using MATLAB for instance). So, with and estimation of the other parameters, specially λ, and fixing the level of significance and by counting the number of calls, we can distinguish if there is a potential attach if the threshold is reached, or if it is a false alarm.

There are other publications for Poisson Generalized linear models for counting the data and provide the best fit for the response data, the number of intrusions per organization [26].

Similar ideas have been implemented in the web of things (WoT) are inclined to suffer from internal attacks, which are from compromised nodes. [27]

IV. CONCLUSION

Mobile devices are widely used in today's society. As a result of this usage, it has attracted a lot of attention from malicious entities. A lot of effort is done, by the scientific community and the experts, in the prevention and detection of security issues present in the Android mobile OS.

In this paper, we presented a new method to detect malware. We assumed that mobile devices would behave abnormally during the attack. To do that we presented a way to observe and structure the normal behavior of the device and call it normality. Knowing the normal behavior of the OS, we detect any strange or abnormal behavior and categorize it as an anomaly for further study. The implications of detecting anomalies are domain-specific.

In future work, we plan to validate this method with real-world data.

REFERENCES

[1] Statcounter https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-201909-202204-bar
[2] Kaspersky Report https://securelist.com/it-threat-evolution-q2-2021-mobile-statistics/103636/
[3] Checkpoint https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-mobile-security/
[4] Kaspersky https://www.kaspersky.com/resource-center/threats/top-seven-mobile-security-threats-smart-phones-tablets-and-mobile-internet-devices-what-the-future-has-in-store .
[5] Cyber security hub https://www.cshub.com/malware/articles/incident-of-the-week-malware-infects-25m-android-phones

[6] Physio.org https://phys.org/news/2019-07-malicious-apps-infect-million-android.html
[7] Cyber security hub https://www.cshub.com/malware/news/incident-of-the-week-rat-malware-strains-believed
[8] DHS and FBI report https://www.cisa.gov/uscert/sites/default/files/publications/MAR-10135536-F.pdf
[9] https://www.knowledgehut.com/blog/web-development/system-calls-in-os
[10] Chen Da, Zhang Hongmei and Zhang Xiangli, "Detection of Android malware security on system calls," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016, pp. 974-978, doi: 10.1109/IMCEC.2016.7867355.
[11] T. Bhatia and R. Kaushal, "Malware detection in android based on dynamic analysis," 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security), 2017, pp. 1-6, doi: 10.1109/CyberSecPODS.2017.8074847.
[12] Sihag, Vikas & Swami, Ashawani & Vardhan, Manu & Singh, Pradeep. (2020). Signature Based Malicious Behavior Detection in Android. 10.1007/978-981-15-6648-6_20.
[13] Yan, P.; Yan, Z. A survey on dynamic mobile malware detection. Softw. Qual. J. 2017, 26, 891–919
[14] S. Hou, A. Saas, L. Chen and Y. Ye, "Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs," 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), 2016, pp. 104-111, doi: 10.1109/WIW.2016.040.
[15] Vidal, J.M., Monge, M.A.S., Villalba, L.J.G.: A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences. Knowl. Based Syst 150, 198–217 (2018)
[16] Cisco Systems inc https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html
[17] TechTarget https://www.techtarget.com/searchsecurity/definition/malware
[18] Dhalaria, M., Gondotra, E.,: Android Malware Detection Techniques: A Literature Review. Recent patent on Engineering, Vol 15, issue 2, 2021
[19] N. Aman, Y. Saleem, F. H. Abbasi, and F. Shahzad, "A hybrid approach for malware family classification," in Applications and Techniques in Information Security. ATIS 2017. Communications in Computer and Information Science, vol. 719, pp. 169–180, Springer, Singapore, 2017
[20] Y. Chang, S. Wang, 'The Concept of Attack Scenarios and its
Applications in Android Malware Detection', IEEE 18th
International Conference on High Performance Computing and Communications 2016, pp. 1485-1492.
[21] T. Isohara, K. Takemori, A. Kubota, 'Kernel-based Behaviour Analysis for Android Malware Detection', Seventh International Conference on Computational Intelligence and Security', 2011, pp. 1011-1015.
[22] IBM https://www.geeksforgeeks.org/introduction-of-system-call/
[23] S. De Capitani, S. Foresti, and P. Samarati, "Data Security Issues in Cloud Scenarios", In Proceedings of the 11th International Conference on Information Systems Security, 2015.
[24] M. Aranitasi, M. Neovius, ―Anomaly Detection in Cloud Based Application using System Calls‖ CLOUD COMPUTING, The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 2017
[25] M. Aranitasi, B. Byholm and M. Neovius, "Quantifying Uncertainty for Preemptive Resource Provisioning in the Cloud," 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), 2017, pp. 127-131, doi: 10.1109/DEXA.2017.42.
[26] Nandi O. Leslie, Richard E. Harang, Lawrence P. Knachel, and Alexander Kott. "Statistical Models for the Number of Successful Cyber Intrusions" U.S. Army Research Laboratory, Adelphi Laboratory Center
[27] Weidong Fang, Wuxiong Zhang, Wei Chen, Li Yi and Weiwei Gao. 2021. PDTM: Poisson Distribution-based Trust Model for Web of Things. In Companion Proceedings of the Web Conference 2021 (WWW '21 Companion), April 19-23, 2021, Ljubljana, Slovenia. ACM, New York, NY, USA, 7 Pages. https://doi.org/10.1145/3442442.3451144
[28] Checkpoint https://blog.checkpoint.com/2017/01/24/charger-malware/