International Science and Technology Journal
المجلة الدولية للعلوم والتقنية
Volume32العدد
Part 2 المجلد
July 2023 يوليو
المجلة الدولية للعلوم والتقنية
ISTJ
تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Android Malware Detection and Protection Systems

Hanan Mohammed Esmail
University of Aljufra, Faculty of Education -Waddan
Department of Computer Science

hhanan22015@gmail.com

ABSTRACT

Android, an open-source and Linux-based mobile operating system, is the most widely used mobile operating system in the world according to recent reports. Due to its popularity and open-source structure, Android has become a target for malicious attacks and attackers. According to Cisco's 2014 security report, 99% of mobile malware targets the Android operating system. Android applications are usually obtained from the official app market, Play Store, which publishes applications uploaded by various developers without subjecting them to security scanning. In addition, when a user attempts to install an application, Android presents the permissions requested by the application during installation and leaves all responsibility to the user afterward. Studies have shown that a large majority of Android users are unaware of these permissions or the effects they have. Therefore, there is a need for a security scan to determine whether applications contain malicious content and to inform users. Various approaches and methods are available in the literature to meet this need. In this study, various Android malware detection/protection systems were examined under four headings: static, dynamic, signature-based analysis approaches, and protection with encrypted data communication. The methods used, including manifest analysis, API call tracking, signature databases, secure data exchange, and machine learning features, were presented comparatively**.**

Key Words: Android, malware detection, smartphone, mobile security, mobile application security.

International Science and Technology Journal
المجلة الدولية للعلوم والتقنية
Volume32العدد
Part 2 المجلد
July 2023 يوليو
المجلة الدولية للعلوم والتقنية
ISTJ

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

أنظمة الكشف عن البرامج الضارة وحمايتها لأجهزة الأندرويد

حنان محمد اسماعيل

جامعة الجفرة – كلية التربية – ودان – قسم علوم الحاسوب

الملخص:

أندرويد، نظام التشغيل المحمول مفتوح المصدر والمبني على نواة لينكس، هو أكثر أنظمة التشغيل المحمول استخدامًا في العالم وفقًا للتقارير الأخيرة. ونظرًا لشهرته وهيكله المفتوح المصدر، أصبح أندرويد هدفًا للهجمات الخبيثة والمهاجمين. ووفقًا لتقرير أمن سيسكو لعام 2014، تستهدف 99% من برامج الضارة المحمولة نظام التشغيل أندرويد. عادةً ما يتم الحصول على تطبيقات أندرويد من سوق التطبيقات الرسمي، متجر Play Store ، الذي ينشر التطبيقات المرفوعة من قبل مطورين مختلفين دون خضوعها لفحص الأمان. بالإضافة إلى ذلك، عندما يحاول المستخدم تثبيت تطبيق، يقدم أندرويد الأذونات المطلوبة من قبل التطبيق أثناء التثبيت ويترك كل المسؤولية على المستخدم بعد ذلك. وقد أظهرت الدراسات أن غالبية كبيرة من مستخدمي أندرويد غير مدركين لهذه الأذونات أو التأثيرات التي تحدثها. ولذلك، هناك حاجة إلى فحص أمان لتحديد ما إذا كانت التطبيقات تحتوي على محتوى خبيث وإبلاغ المستخدمين. تتوفر مختلف الأساليب والطرق في الأدب لتلبية هذه الحاجة. في هذه الدراسة، تمت مراجعة أنظمة الكشف عن البرامج الضارة وحمايتها في أندرويد بمختلف المقاربات: الكشف الثابت، الكشف الديناميكي، التحليل القائم على التوقيع، والحماية مع تبادل البيانات المشفرة. تم تقديم الأساليب المستخدمة، بما في ذلك تحليل الملفات التعريفية، تتبع استدعاءات واجهة برمجة التطبيقات، قواعد البيانات الموقعية، تبادل البيانات الآمن، وميزات التعلم الآلي بشكل مقارن.

الكلمات الدالة: أنظمة كشف البرامج الضارة في أجهزة الأندرويد، الهواتف الذكية، أمان المحمول، أمان تطبيقات المحمول.

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Introduction

Due to its open-source nature, Android has gained a significant market share in the smartphone market. In late 2013, Google announced that there were over one billion Android users. According to research conducted by Strategy Analytics in the fourth quarter of 2013, Android increased its market share in the smartphone market from 75% to 81.3% compared to the previous year, with a user base of 204.4 million [1]. In July 2013, Google announced at an event that the number of applications available on the official Android app store, Play Store, had surpassed one million [2].According to Cisco's 2014 security report, 99% of mobile malicious attacks conducted in 2013 targeted Android [3]. Similarly, the Mobile Security Threat Report prepared by Sophos in 2014 revealed a six-fold increase in Android malware in the past year [4]. Android-based malicious software aims to collect users' personal information (**gray ware**) and perform unauthorized actions (malware). The rate of malicious software on the Play Store has shown a significant increase in recent years, according to studies [5,6]. The main reasons for this increase can be attributed to the open-source nature of Android and its passive protection method. Android has a permission-based security mechanism [7,8]. Permissions are the approvals required for applications to use various resources of the phone, such as Short Message Service (SMS) sending and location reporting. When users installapplications, Android lists the permissions requested by the application. These permissions are used to inform the user about the capabilities the application will have [9]. After this stage, the user can continue with their own evaluation and proceed with the installation or cancel it. Moreover, these permissions are not presented to the user again after the installation is complete [10]. If permissions not requested during installation are used by the application, the application fails [11]. A study conducted by Felt et al. revealed that only 17% of users pay attention to these permissions, and 42% of users have no knowledge about them [10].

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Other studies have also shown that Android permission notifications are ignored by users [8], [12-15].

There are various Android malicious software detection and protection systems in the literature that aim to address these shortcomings. These systems are categorized into four main approaches: static analysis approach, dynamic analysis approach, signature-based analysis and protection, and protection through encrypted data communication. The article is structured as follows:

1. The second section provides information about the Android system architecture to better understand the permission mechanism.
2. The third section discusses the mechanisms for detecting and protecting against malicious software through the static analysis approach, dynamic analysis approach, signature-based analysis and protection, and protection through encrypted data communication.
3. In the fourth section, these methods are presented comparatively.
4. The fifth section presents the results and evaluations, and proposes possible protection methods that can be further developed.

2. Android System Architecture

Android has a layered system architecture. The layers and components of this system architecture are presented in Figure 1.

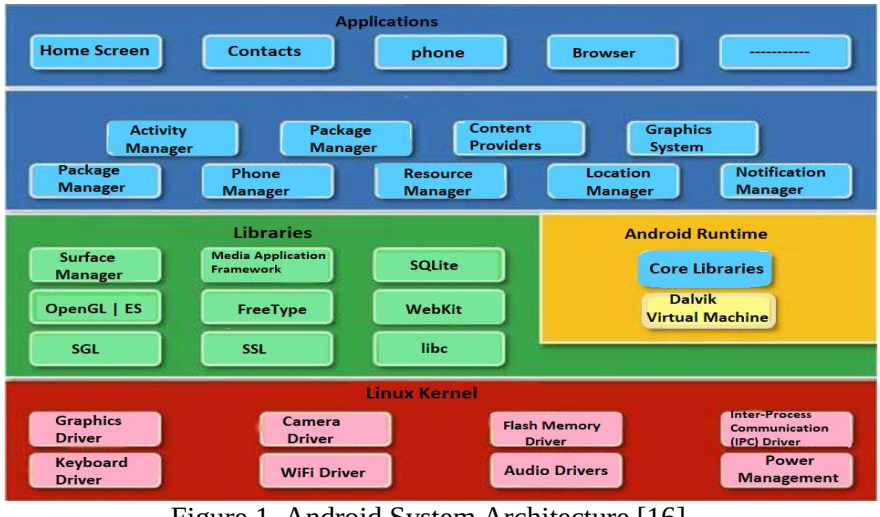

Figure 1. Android System Architecture [16].

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Android has a customized Linux operating system (Linux Kernel) at the bottom layer, which communicates with the phone hardware. The middle layer consists of Java-based libraries and the application framework. The top layer, the application layer, contains software applications that interact with end users, developed using the Android Application Programming Interface (API). Permissions such as SMS sending, location reporting, and external storage read/write, which are requested by applications, are authorized and communicated with system resources through the application framework. For example, in order to perform location reporting on Android, at least one of the permissions, eitherACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION, needs to be requested by the applications. If the user grants this permission/permissions, the Location Manager service, which is one of the components in the middle layer application framework, provides access to location information.Android applications are delivered as package files with the .apk extension. These package files contain all the resources related to the application, such as source code, image files, and constant value definitions. Each Android application contains a manifest file (AndroidManifest.xml) that includes basic information about the application, permissions requested, activities, services, and receivers owned by the application. Android applications are developed using the Java programming language but run on a virtual machine called the Dalvik Virtual Machine (DVM).Figure 2 illustrates the software stack structure of Android applications. After Java classes are compiled, a .class file is created for each class, and Java Virtual Machine (JVM) bytecode is generated. The Dalvik dx compiler recompiles these codes and creates a single .dex file that contains all the classes. This file is executed on the DVM. Each application runs on its own DVM.Vidas and his friends [20] have developed an extension of Eclipse, a widely used Java Integrated Development Environment (IDE), to perform application permission analysis on source code. However, this tool does not

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

question the purpose of permission usage, thus it does not highlight potential malicious content.

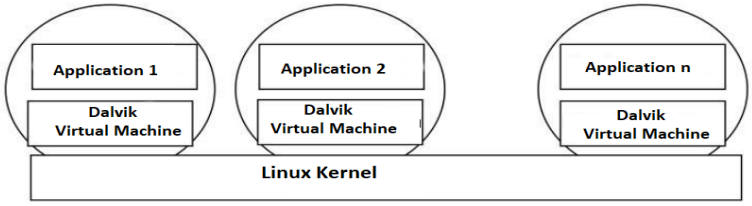

Figure 2. Software Stack Structure of Android Applications [18]

3. Android Malware Detection and Protection Mechanisms

As the variety and number of Android malware increase, protection methods also increase and diversify accordingly. Malware detection and protection systems in the literature are classified under four headings: static analysis approach, dynamic analysis approach, signature-based analysis and protection, and encrypted data communication.

3.1. Static analysis approach

The static analysis approach enables the detection and protection of malicious software in applications through the data they provide without being installed on devices. The most significant advantage of this approach is that it allows for the detection and protection of malicious software without installing it on the device. Thus, the device remains unaffected by the malicious content.

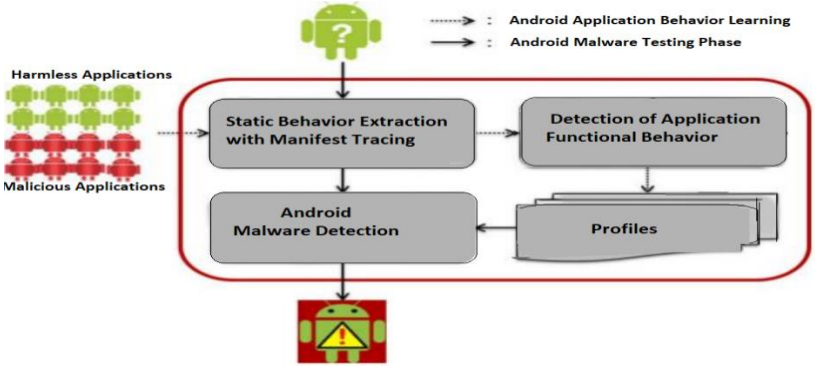

Figure 3. DroidMat Architecture [19]

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

DroidMat [19] is a tool that enables the detection of Android malware through API calls related to the manifest file and permissions. The architectural structure of DroidMat is presented in Figure 3. As shown in the figure, static behavioral analysis is derived through the manifest file. After obtaining static behavioral analyses, the application proceeds to the functional behavioral analysis phase. Once these analyses are obtained, malicious software models are created using the K-means algorithm. The number of clusters to be formed is determined using the Singular Value Decomposition (SVD) algorithm. Finally, using the kNN (k Nearest Neighbours) algorithm with k=1, it is determined whether the analyzed application is malicious or not.Drebin [21] is a tool that attempts to detect Android malware by combining static analysis and machine learning approaches. Drebin collects various features such as permissions, API calls, and network addresses using the application's source code and manifest file. These features are embedded in a unified feature vector and used for Android malware detection through various patterns.Felt et al. [22] proposed a system called Stowaway that detects Android applications that request excessive permissions. Stowaway identifies the set of API calls used by an application and matches these calls with permissions. Stowaway consists of two parts: the first part identifies the API calls used by the application, and the second part matches permissions to each API call to determine the required permissions. This way, applications that request excessive permissions are identified through this system. Additionally, the study highlights the "most common developer mistakes" asfollows: (1) incorrect use of permission names, (2) misuse of proxies, (3) violation of protection levels during the use of related methods, (4) inclusion of deprecated permissions, (5) use of signature and/or system permissions, (6) forgetting changes made during testing, and (7) unnecessary permission requests through copy-pasting.

3.2. Dynamic analysis approach

The most significant difference between the dynamic analysis approach and the static analysis approach is that application analysis

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

is performed at runtime. The key advantage of the dynamic analysis approach is its ability to uncover complexities, deficiencies, or vulnerabilities that cannot be revealed through static analysis [23].

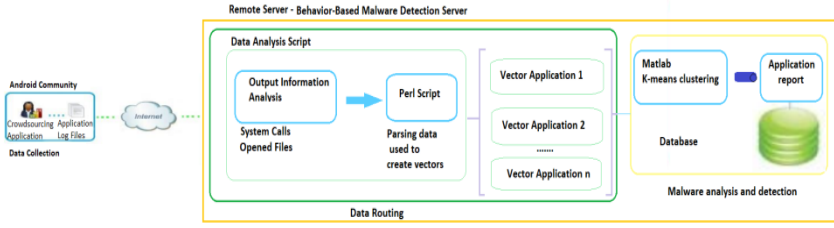

Figure 4. Crowdroid Android Malware Detection Mechanism [24]

Burguera et al. [24] present Crowdroid, an application framework for detecting abnormal behavior in Android applications. Crowdroid utilizes the Strace command, a Linux-based tool, to capture system calls made by the Android kernel and classify Android applications as "benign" or "malicious". Figure 4 illustrates the Android malware detection mechanism of Crowdroid. This mechanism involves data collection (device information, installed application list, and monitoring logs), data processing, malicious software analysis, and detection stages.

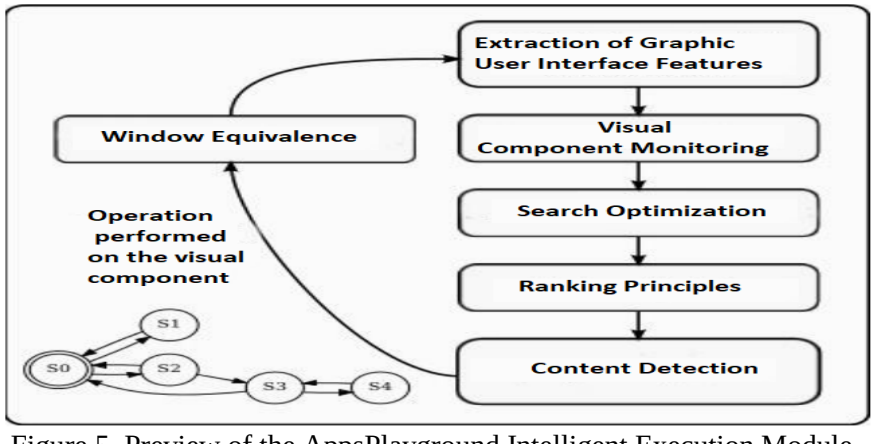

Figure 5. Preview of the AppsPlayground Intelligent Execution Module [25]

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Rastogi et al. [25] developed an automated dynamic analysis tool called AppsPlayground for smartphone applications. AppsPlayground employs various detection, tracing, and masking techniques such as kernel-level tracing, API call tracing, event triggering, and intelligent execution to provide an effective analysis environment. Figure 5 presents a preview of the AppsPlayground intelligent execution module.

Zhou and Jiang [26] propose a systematic approach to Android malware detection. They use a dataset consisting of 1,260 Android malware samples from 49 different malware families. Zhou and Jiang characterize malicious software based on various behavior patterns, including installation, activity, and data transmission.

RiskRanker [27] is a proactive system that analyzes whether applications exhibit dangerous behaviors such as gaining high-level privileges and sending SMS messages in the background. It aims to detect potentially harmful behaviors in applications.DroidMOSS [28] is a system designed to detect attacks on applications by repackaging them with malicious content in application markets. It focuses on identifying repackaged applications.Paranoid Android [29] conducts security scans on Android applications using virtual copies of smartphones in a virtual environment. By using servers, it allows for the simultaneous application of multiple detection techniques. Additionally, by performing virus scans on servers instead of mobile devices, Paranoid Android reduces the software cost.

CopperDroid [30] performs dynamic behavior analysis by characterizing low-level operating system-specific and high-level Android-specific application behaviors on an emulator called QEMU. The application behavior analyses are observed through system call requests. This allows CopperDroid to analyze behavior from Java programs, Java Native Interface (JNI), or native code executions.

3.3. Signature-based analysis and protection

In signature-based analysis and protection systems, analyzed applications are stored in a signature database, following a

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

continuous learning-based approach. This approach typically involves the use of a central server and a signature database. The central server handles the analysis and protection processes, while the database server stores the obtained analyses for future use.

Guido et al. [31] developed a service called Tractor Beam to run on smartphones and transfer information to a central server via Wi-Fi. The service periodically sends bit changes and offset values to the central server over Wi-Fi. Tractor Beam stores only the bit changes, represented by Secure Hash Algorithm-256 (SHA-256) hashes, which only show the differences in bits instead of storing entire blocks. This approach minimizes the storage space required. The central server stores the received bit changes and offset values in a normalized relational database. The central server creates system images and processes them through an analysis framework. The analysis framework consists of detectors and loggers. Detectors house malware detection techniques, while loggers record all activities that may be malicious.

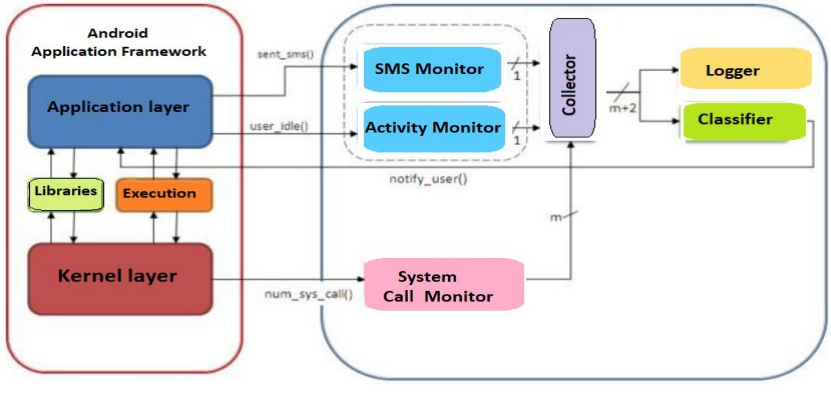

Figure 6. Functional blocks of MADAM [32]

MADAM [32] is a multi-level anomaly detection tool for Android malware. MADAM performs malware detection at both the kernel and application levels. The system leverages system calls to monitor user activities, file and memory accesses, incoming and outgoing data traffic, power consumption, and sensor status at the kernel level. At the application level, detections are based on extracted

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

features such as user idle status, the number of sent/received SMS messages, and Bluetooth/Wi-Fi analysis. Figure 6 presents the functional blocks of MADAM, and Table 1 provides the level-specific features of MADAM.

Level	Feature
Kernel User/Applications	System calls Executed processes CPU usage Free RAM Keyboard operations Dialed numbers Sent/received SMS messages Bluetooth/Wi-Fi analyses

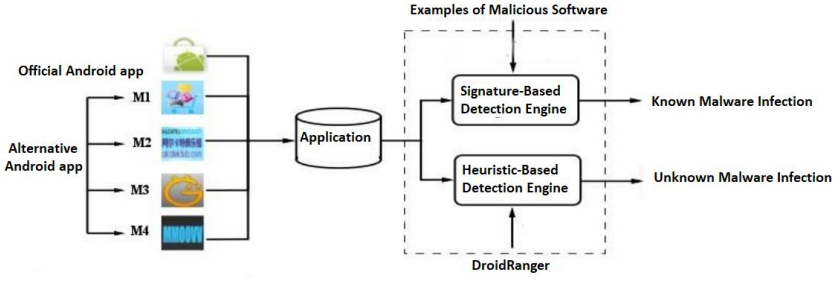

Figure 7. DroidRanger Software Architecture [6]

DroidRanger [6] aims to detect new samples of known malware families through its permission-based behavior tracing scheme. Subsequently, a heuristic-based filtering scheme is applied to reveal the behavioral patterns of unknown malware families. Figure 7 presents the software architecture of DroidRanger.

3.4. Protection Approach with Encrypted Data Communication

The protection approach with encrypted data communication aims to ensure secure data transmission. By doing so, it aims to prevent

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

potential security vulnerabilities that are frequently exploited by malicious software.

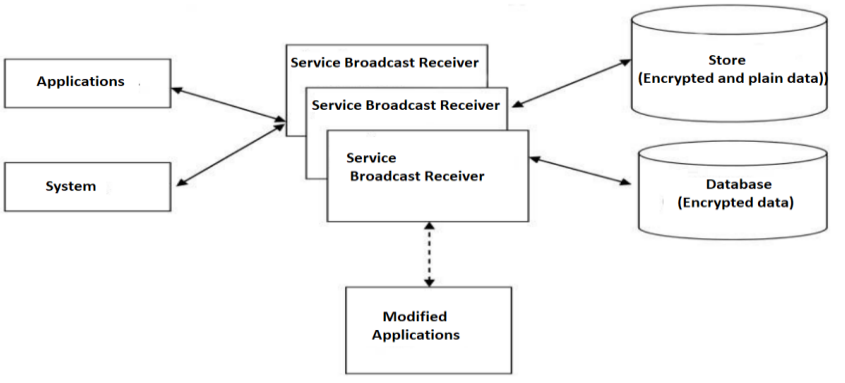

Figure 8. Cryptographic system architecture proposed by Pocatilu [18].

In the approach proposed by Pocatilu [18], sensitive user data such as SMS information, emails, and files are encrypted and stored in a database. When a user receives a message from the system or applications, an interception takes place, and the message content is encrypted and stored in the database. When another application needs access to the stored data, the data is retrieved from the SQLite database, decrypted, and then forwarded to the relevant applications. Figure 8 illustrates the proposed system architecture.The javax.crypto package in the Android application programming interface provides classes for symmetric key encryption (AES, DES), public-key encryption (RSA, DH), and message digest functions. Figure 9 presents an example Java class that demonstrates the encryption and decryption of character strings using the DES algorithm.Vidas and Christin [33] introduce a tool called AppIntegrity, which assists in cryptographic verification between software developers and users.

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

```
import javax.crypto.*;
import org.kobjects.base64.Base64;

class SecCD
{
 public static String decripteazaDES(String sir, SecretKey cheie)
 {
       String sirDecriptat = null;

       try
       {
        Cipher cifDecriptare = Cipher.getInstance("DES/ECB/PKCS5Padding");
        cifDecriptare.init(Cipher.DECRYPT_MODE, cheie);
        sirDecriptat = new String(cifDecriptare.doFinal(Base64.decode(sir)));
       }
       catch(Exception ex)
       { Log.e("PDM1", ex.getMessage()); }
       return sirDecriptat;
}

 public static String cripteazaDES(String sir, SecretKey cheie)
 {
       String sirCriptat = null;

       try
       {
        Cipher cifCriptare = Cipher.getInstance("DES/ECB/PKCS5Padding");
        cifCriptare.init(Cipher.ENCRYPT_MODE, cheie);
        sirCriptat = Base64.encode(cifCriptare.doFinal(sir.getBytes("UTF8")));
       }
       catch(Exception ex)
       { Log.e("PDM1", ex.getMessage()); }
       return sirCriptat;
 }
}
```

Figure 9. Example Java Class for Encrypting and Decrypting Character Strings using the DES Algorithm.

This tool aims to strengthen the authentication mechanisms provided in application markets. It is believed that this can make application repackaging, a popular method for spreading malicious software, more difficult.

Khalil et al. [34] have proposed a new architectural design called "Consolidated Identity Management System" by addressing the vulnerabilities of existing identity management systems in mobile cloud computing. It is believed that this new architecture addresses the identified server vulnerabilities, mobile device vulnerabilities, and network traffic interception vulnerabilities.APK files of applications available on the Play Store can be obtained and modified to incorporate malicious code. Subsequently, these applications are presented under a different name through the market or another accessible webpage. Apart from these activities, the main tools used for recompiling and modifying Android applications include:- APKTool: Used for compiling and

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

recompiling APK files.- smali: Acts as a translator and reverse translator for DEX files.- dex2jar, jad: Enables the extraction of JAR files from DEX files.- JD-GUI: An interface program that converts compiled Java classes (class files) into readable Java source code.

4. System Comparisons

Due to the wide variety of malicious software, a single protection and detection method is not sufficient for detecting and protecting against all types of malware. Each approach presented in Section 3 encompasses a unique method for malware protection and detection. The malware detection and protection features of these approaches are comparatively presented in Table 2.

Table 2. Feature comparison of malware detection and protection systems.

Feature	Drebin	Crowdroid	MADAM	DroidMat	Julia
ManifestAnalysis	YES	NO	YES	YES	YES
API call Monitoring	YES	YES	YES	YES	YES
Signature database	NO	YES	YES	YES	NO
Secure data exchange	NO	NO	NO	NO	NO
Machine learning	YES	YES	YES	YES	NO

As seen in Table 2, none of the examined systems possess all five defined features. Among these systems, only MADAM and DroidMat have four out of the five features we considered. However, none of the malware detection and protection systems provide secure data exchange. On the other hand, all of these systems have the feature of API call monitoring and detection. Additionally, it can be observed that 4 out of 5 systems utilize machine learning in their detection and protection mechanisms.

5. Conclusion and Evaluations

Android continues to maintain its status as the most widely used mobile operating system worldwide. The official application market for Android, Play Store, sees thousands of new applications being uploaded every day. This popularity has made Android a prime target for malicious software developers. Research indicates that

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

99% of mobile-based malicious attacks are targeted toward Android. The key factors identified in Android's vulnerability to malicious software include: (1) Passive protection exhibited towards applications uploaded to Play Store, (2) End users being responsible for approving permissions requested by applications, with a large majority of users being unaware of these permissions, (3) Open-source nature of Android, and (4) Support for Java, a popular programming language.In this study, various systems involved in the detection and protection against Android malicious software were comparatively examined from different perspectives. The examination results revealed that these systems lack features related to secure data exchange. However, they tend to incorporate machine learning methods for malware detection and protection or prefer utilizing machine learning.These systems generally operate after an application has been installed on the system, rather than having pre-installation detection and protection mechanisms. Therefore, there is a need for more comprehensive malware detection and protection systems that can identify malicious software before Android applications are installed, thereby guiding users and taking necessary precautions Developing such systems is crucial for analyzing the usage purpose of permissions, which forms the foundation of security vulnerabilities, by examining the application source code. Through such an application, it becomes possible to determine whether requested permissions have malicious intent and to ensure that users or the system take necessary measures.

References:

[1] Zhou, Y., Jiang, X., & Zhou, W. (2012). Detecting repackaged cellphone packages in 1/3-birthday celebration android marketplaces. In complaints of the ninth worldwide conference on cellular systems, programs, and offerings (MobiSys), 1-14.

[2] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-based malware detection system for

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (SPSM), 15-26.

[3] Grace, M., Zhou, Y., Wang, Z., & Jiang, X. (2012). Unsafe exposure analysis of mobile in-app advertisements. In Proceedings of the 5th ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec), 101-112.

[4] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. (2014). DREBIN: Effective and explainable detection of Android malware in your pocket. In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS), 23-26.

[5] Tam, K. P., & Ho, K. K. (2015). A survey of mobile malware in the wild. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 255-260.

[6] Wang, T., Niu, Y., & Chen, B. (2013). SmartDroid: An automatic system for revealing UI-based trigger conditions in Android applications. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), 181-191.

[7] .Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-based malware detection system for Android. Journal of Trust Management, 1(2), 91-102.

[8] Wang, T., & Chen, B. (2014). DeDexer: A static analysis tool for Android application. In Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME), 460-463.

[9] Enck, W., Ongtang, M., & McDaniel, P. (2009). Understanding Android malware. In Proceedings of the ACM Workshop on

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

Security and Privacy in Smartphones and Mobile Devices (SPSM), 3-14.

[10] Zhou, Y., Wang, Z., Zhou, W., & Jiang, X. (2013). Division of labor: Tools for discovering split-personality malware. In Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS), 1-15.

[11] Enck, W., Ongtang, M., & McDaniel, P. (2009). Understanding Android malware. In Proceedings of the ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 3-14.

[12] Shabtai, A., Kanonov, U., Elovici, Y., & Glezer, C. (2010). Andromaly: A behavioral malware detection framework for Android devices. Journal of Intelligent Information Systems, 38(1), 161-190.

[13] Zhou, Y., & Jiang, X. (2012). Dissecting Android malware: Characterization and evolution. In Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P), 95-109.

[14] Bilge, L., Demir, E., Davi, L., & Sadeghi, A.-R. (2014). Squashing the pufferfish: Detecting encrypted botnet traffic. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS), 1040-1051.

[15] Wei, F., & Roy, A. (2016). Automating Android malware detection using deep learning techniques. In Proceedings of the 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 245-255.

[16] Xu, Y., Qi, Y., Zhang, Y., & Bissyande, T. F. (2016). DroidFlow: Efficient taint analysis of whole-system Android applications. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), 888-900.

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

[17] Wang, T., & Chen, B. (2015). In the direction of automatic detection of permission specification violations in Android apps. In court cases of the twenty second ACM SIGSOFT international Symposium on Foundations of software Engineering (FSE), 575-586.

[18] Ren, Y., Kang, M., Lyu, M. R., & Yang, J. (2013). Apposcopy: Semantics-primarily based detection of Android malware through static analysis. In proceedings of the twenty second USENIX safety Symposium (USENIX security),519-534.

[19] Zhu, T., Martini, B., & Rrushi, J. (2016). A comparative analysis of static, dynamic, and hybrid analysis for Android malware detection. In Proceedings of the 9th International Conference on Security of Information and Networks (SIN), 102-109.

[20] Xu, Y., Zhang, Y., & Yin, H. (2013). Semantics-aware Android malware classification using weighted contextual API dependency graphs. In Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC), 369-378.

[21] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., & Jung, J. (2014).TaintDroid: An information-flow monitoring system for actual-time privacy tracking on smartphones. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 1-6.

[22] Zhou, Y., Wang, Z., Zhou, W., & Jiang, X. (2013). Hare hunting in the Android market. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS), 795-806.

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

[23] Gascon, H., Arp, D., Rieck, K., & Sadeghi, A.-R. (2013). Structured analysis of Android malware families. In Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (AISec), 49-60.

[24] Zhauniarovich, Y., Canfora, G., & Aciicmez, O. (2014). Repecker: Automatic detection of repackaged applications on Android. In Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC), 382-391.

[25] Li, L., & Bartel, A. (2017). I realize why you went to the hospital: risks and attention ofHTTPS site visitors evaluation. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), 1617-1631.

[26] Backes, M., Bugiel, S., Skoruppa, M., & Weisgerber, S. (2016). COPDroid: Automatic detection of code reuse vulnerabilities in Android apps. In Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS), 1-14.

[27] Pearce, P., Armando, A., Li, L., Li, N., & Etalle, S. (2015). Enhancing Android malware detection through analysis of resource utilization. In Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security (CCS), 1295-1306.

[28] Fredrikson, M., Jha, S., & Ristenpart, T. (2014). Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), 1322-1333.

[29] Liu, Y., Sun, X., & Wei, T. (2015). Falcon: Recognizing Android app GUI through feature extraction. In Proceedings of

تم استلام الورقة بتاريخ: 2023/6/15م وتم نشرها على الموقع بتاريخ: 24/ 7/2023م

the 10th ACM Symposium on Information, Computer and Communications Security (ASIACCS), 477-482.

[30] Tapiador, J. E., Suarez-Tangil, G., & Peris, E. M. (2013). Evolution, detection, and analysis of malware for smart devices. In Emerging Trends in ICT Security (pp. 21-34). Springer, Berlin, Heidelberg.

[31] Spreitzenbarth, M., Echtler, F., Schreck, T., & Freiling, F. C. (2013). Mobile-sandbox: Having a deeper look into Android applications. In Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC), 1283-1290.

[32] Bilge, L., Demir, E., & Balzarotti, D. (2013). A comprehensive survey of malware in the Android market. In Proceedings of the 2013 European Symposium on Research in Computer Security (ESORICS), 363-380.

[33] Kan, Y., Fong, S., & Zhang, L. (2014). A review of machine learning techniques for Android malware detection. In Proceedings of the 2014 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 63-68.