Raf. J. of Comp. & Math's., Vol. 15, No. 2, 2021
DOI:

User Centric Android Application Permission Manager

Belal Amro *bilala@hebron.edu*
Zaid Abu Znaid *21611416@students.hebron.edu*
Hebron University

Received on: 08/03/2021 **Accepted on: 23/03/2021**

ABSTRACT

Mobile malware has become a very hot research topic in the last few years, and this was due to the widespread usage of mobile devices all over the world. Like other systems, mobile devices are prune to different attacks that might invade user's privacy and lead to private data leakage. Millions of Mobile application have been developed and used Worldwide, most of them are requiring permissions to work properly. The permission management problem is more apparent on Android systems rather than other mobile systems such as iOS. Some of these permissions might lead to successful security attacks on Android systems and hence lead to privacy leakage. To reduce the possibility of such attacks, many researchers have proposed mobile applications that help users to manage access permissions for their mobile applications. Most of the proposed systems lack the ability to profile users according to their preferences and do not provide automatic follow up with temporary granted permissions. In this research, we propose a User Centric Android Application Permission Manager tool called (UCAAPM), that provides an efficient and flexible way for managing permissions and profiling these permissions for each user, these profiles can be used on any Android device. UCAAPM will automatically follow up users permissions and grant/deny the permission on a scheduling basis defined by the user's profile and according to his preferences. Experimental results showed that the tool works efficiently in terms of CPU, RAM, and power consumption, furthermore users are highly satisfied with using it.

Keywords: Android, permission control, access control, security, privacy.

1. Introduction

According to McAfee Labs Threats Report [1], mobile malwares have spread rapidly in the last few years. This fact was also confirmed by Kaspersky Labs that have reported that the number of mobile malware attacks has doubled in 2018[2]. In addition, security researchers reported that the number of malware files has decreased, leading the researchers to conclude that the malware types have become more robust and immune to different Anti-Malware detection techniques.

Based on this, mobile malware has become an important threat that must be counterfeited. Recently, there has been many attacks aimed at breaking security of Android systems and are directly related to access permissions, such attacks are escalation attacks [3][4], in this type of attacks a component in one application might be able to indirectly access another component in another application via a third component that has access permissions. Man-in-the-middle attack [5][6] is another reported Android attack, here an attacker can relay and may also alter the communication between two parties. Inter-process communication call vulnerabilities [7][8] is another attack that allows a component to access the data and transfer it to

other components either locally – within the same device- or to an external server. Inter-process communication might be a basis for different security vulnerabilities including: hijacking, spoofing, as well as collusion.

The malware detection techniques -either static or dynamic- showed a very good ability to detect mobile malwares, a summary of theses techniques and their drawback are summarized in Table 1.

Table 1: Malware Detection Techniques

Technique	Description	Disadvantages
Signature Based Approach [9]	This method extracts the semantic patterns and creates a unique signature	IT can only identify the existing malwares and fails against the unseen variants of malwares
Permission Based Analysis [10]	Analyzes permissions required by applications and detect abnormal requirements	Permission based methods require second pass to provide efficient malware detection.
Virtual Machine Analysis [11]	Tests the application behavior and analyzes control and data flow which in sake of detecting dangerous functionalities	Analysis is performed at instruction level and consumes more power and storage space.
Anomaly based [12]	Based on watching the behavior of the device by keeping track of different parameters and the status of the components of the device	The larger the parameters engaged the more the calculations required.
Taint analysis [13]	Tracks multiple sources of sensitive data and identifies the data leakage in mobile applications	Does not perform control flow tracking.
Emulation based [14]	dynamically analyze applications based on Virtual Machine Introspection	Cannot detect new malwares

As seen from Table 1, all techniques have some disadvantages that prohibits them from perfectly doing the job. In fact, some of the security enhancements burden is put on the shoulders of the users of the systems. Therefore, some efforts should be done to enhance the user's ability to tackle security threats.

As mentioned before, some of these attacks are based on exploiting some vulnerabilities related to access control permissions granted to the applications, where users are often not aware about these vulnerabilities. To protect users against these attacks and help them manage their permissions efficiently, researchers has proposed many permission management tools and mechanisms. In our work, we propose a user-centric permission control tool for Android devices that relies on user's preferences and will build a portable permission profile for the Android user where he can use it with any Android device.

The rest of this paper is organized as follows, a literature is provided in section 2. Explanation of the design of the proposed application and its features are provided in Proposed model and specification section. The implementation details are described in Implementation and Testing section. The last section is the Conclusion.

2. Literature Review:

Access control has been improved in different Android Mobile operating systems, however it is still far away from achieving the desired solutions for end users. The latest

works on improving the Android access controls relied on either a virtual layer for managing access control or leveraging the security of the Android system itself. I this section we will spot the light on the most popular research solutions related to managing Android access controls and hence leveraging the Android security.

Most of the proposed solutions relied on providing a virtual layer for managing access control in Android systems, Peng et. al. proposed a three-tier Android safety subsystem that aims at protecting the Android security control subsystem. The proposed system uses the security control subsystem to control the other main parts of the Android system [15].

Liu et. al. proposed a fine-grained access control framework for managing access control on sensitive information relying on eXtensible Access Control Markup Language data flow model. Users are able to define their access policies for each application and access request to sensitive information is granted/denied according to these policies [16].

An access control application called Saint was proposed by Ongtang [17], the application controls the permission usage in the application runtime. A similar application called Apex was proposed by Nauman et. al.[18]. Apex protects privacy by allowing the user to selectively grant permissions when applications are installed.

Inverardi et al [19], proposed a user-managed approach that allows end users to specify their desired privilege levels. A similar work by Aron and Hanacek [20] was proposed and is built on a dynamic permission mechanism for Android applications. This mechanism prevented the possibility of privacy leakage by enforcing permissions to the applications on the basis of the files they work with.

S Bugiel et al [21], proposed FlaskDroid application that implements mandatory access control (MAC) on different layers in Android. The authors also proposed a policy language that is used as a middleware semantics of Android. RecDroid [22], is a crowdsourcing framework for recommending users regarding smartphone permission controls best practices.

Scoccia et. al. proposed an enhancement on trustability of Android applications by implementing a user centric flexible permissions manager [23]. it allows users to selectively grant the desired level of permissions according to the specific features of the application that requires these permissions.

Away from implementing applications, Morles et. al. [24] proposed a new categorization for permissions in Android systems that might improve the permission management.

Most of the proposed systems use automatic techniques to control permissions based on different criteria including the user's preferences. However, they do not provide flexibility in choosing preferences that made it hard for users to accommodate with. The user centric permission control techniques and their disadvantages in Table 2

Table 2: comparison of user centric permission management techniques

No	Method	description	Disadvantages
1	Apex	Apex protects privacy by allowing the user to selectively grant permissions when applications are installed	• User grants permissions only when application is installed; will not be able to manage permission efficiently after that • No user profiling • No automatic follow up with permission management
2	RecDroid	Is crowdsourcing framework	• No user profiling

		for recommending smartphone users with the suitable permissions control best practices.	• Online system • Affected by user's opinions • No automatic follow up with permission management
3	Scoccia et. al	Enhancement on trustability of Android applications by implementing a user centric flexible permissions manager	• No user profiling • No automatic follow up with permission management

According to Table 2, we find that the current user centric permission management systems do not offer a flexible techniques based on user's preferences, this might be important in permission control and differs from one user to another. Besides, these systems are not designed to automatically follow up permissions and grant/deny permissions on a scheduling basis. In our work, we propose a flexible user-centric permission control tool for Android devices that relies on user's preferences. The system automatically profiles these preferences and enables users to edit their profiles and to use them with other Android devices. The tool automatically follows the permissions being updated and performs grant/deny on a predefined basis.

3. Proposed Model And Specification

UCAAPM is a User Centric Android Application Permission Manager that will be used to manage permissions of different Android applications and their behavior according to the user's preferences. To better explain UCAAPM and how does it work, we first explain the components of UCAAPM, then we explain the work flow and the interaction between different components, at last, we will explain some of the innovative properties of UCAAPM.

Components of UCAAPM:

UCAAPM consists of 6 interactive components as shown in Figure 1. These components are described below:

1) Application : It refers to Android applications that run on the user's device, either newly or previously installed. UCAAPM works for all applications installed on the device and enables users to manage permissions and update these permissions on a need basis.
2) Package manager: The Android Framework API which is responsible for application updates, installation, and uninstallation. UCAAPM will invoke the package manager to get a list of all installed applications on the Android device, their icons, permissions… etc.
3) Personalized Mobile Malware Guard (PMMG): An interface that interacts with the user and android package manager, the role of this component is fetching metadata of installed and newly installed applications from package manager, and providing the user with an interface to decide what are the permissions that an application could have. PMMG will then provide the Permit-Granter component with the user's feedback, which in turn will store user decisions regarding permissions in the rule database.
4) Permit Granter : The role of this component is to provide the PMMG interface with the metadata of each application, every application has its own metadata that is stored in the Rule Database, so Permit Granter can perform queries on this database. It also forms a gateway between the PMMG interface and Rule Database.

The idea behind this architecture is to hide the Rule Database from the PMMG interface for the sake of better security.

5) Rule database**:** A database that contains metadata about applications that are installed on the user's mobile. It will help building the user's preferences profile and enables the profile to be transferred to other Android devices.
6) User: The Android device user who will be interacting with the mobile applications and will grant/deny permissions for different applications. The user's feedback and actions will be stored in the rule database and will be used for future interactions and access control decisions.

UCAAPM functionalities:

Figure 2 shows the use case diagram of our UCAAPM. As seen from the diagram, the user of UCAAPM will be able to perform four functions listed below:

1) Login and log out : This security feature will prohibit others from changing the permissions' status of the user.
2) Manage permissions: This will enable the user to manage his permissions by the following functions:

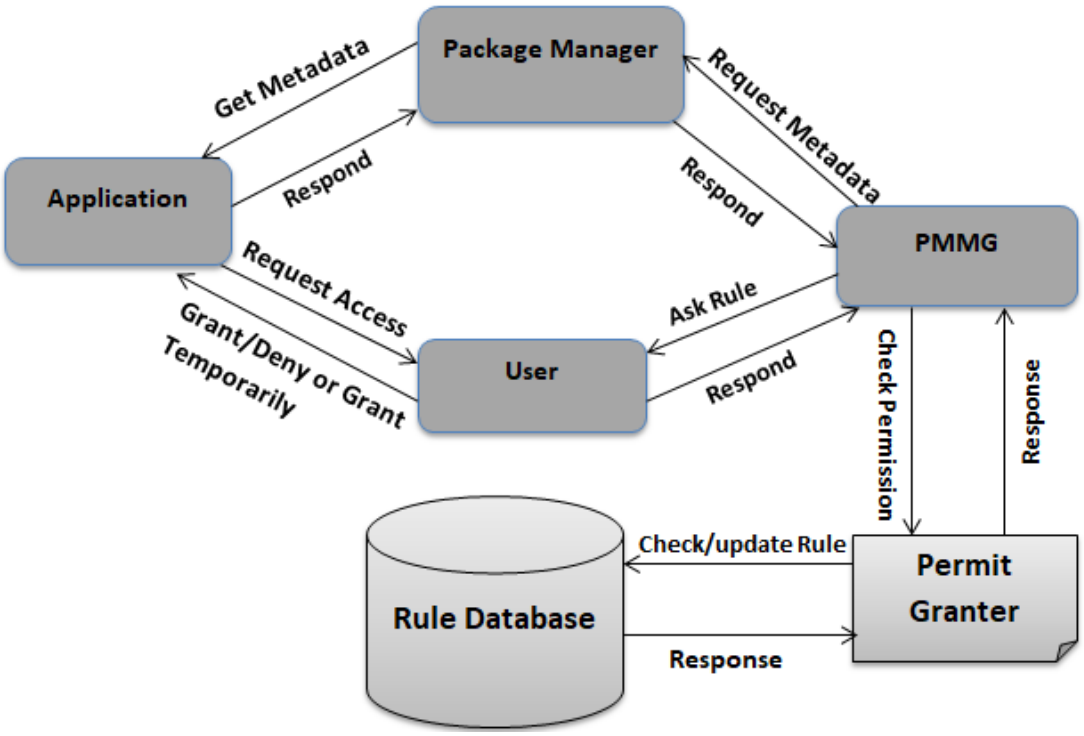

Figure 1: Components of UCAAPM

a. View permissions
b. Grant permission
c. Deny permission
d. Grant permission temporarily

3) And scan packages: : This will enable the user to perform some security services from trusted third parties such as scanning packages against viruses and vulnerabilities.

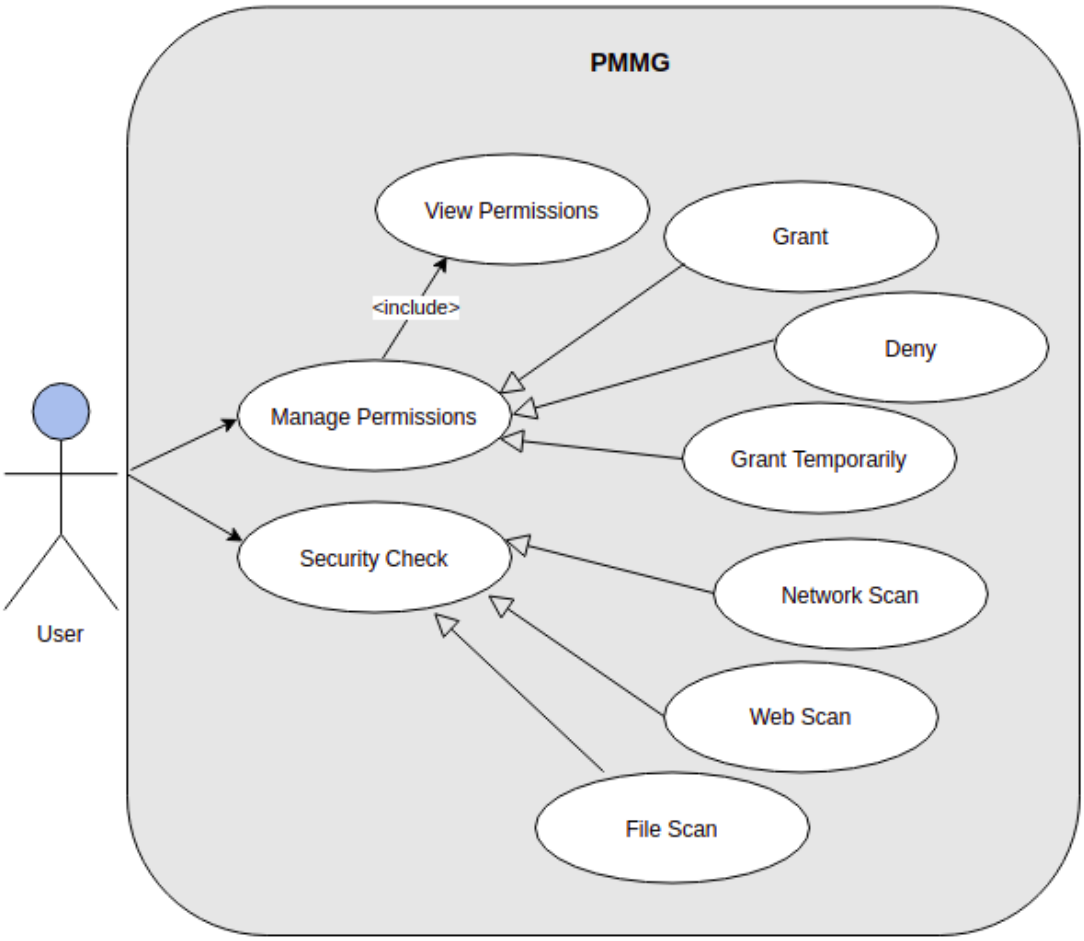

Figure 2: UCAAPM use case diagram

4. Implementation And Testing:

In this section, we will explain the implementation and the testing results of our UCAAPM tool. The complete workflow of UCAAPM is provided in Figure 3. The figure shows the complete process that UCAAPM follows:

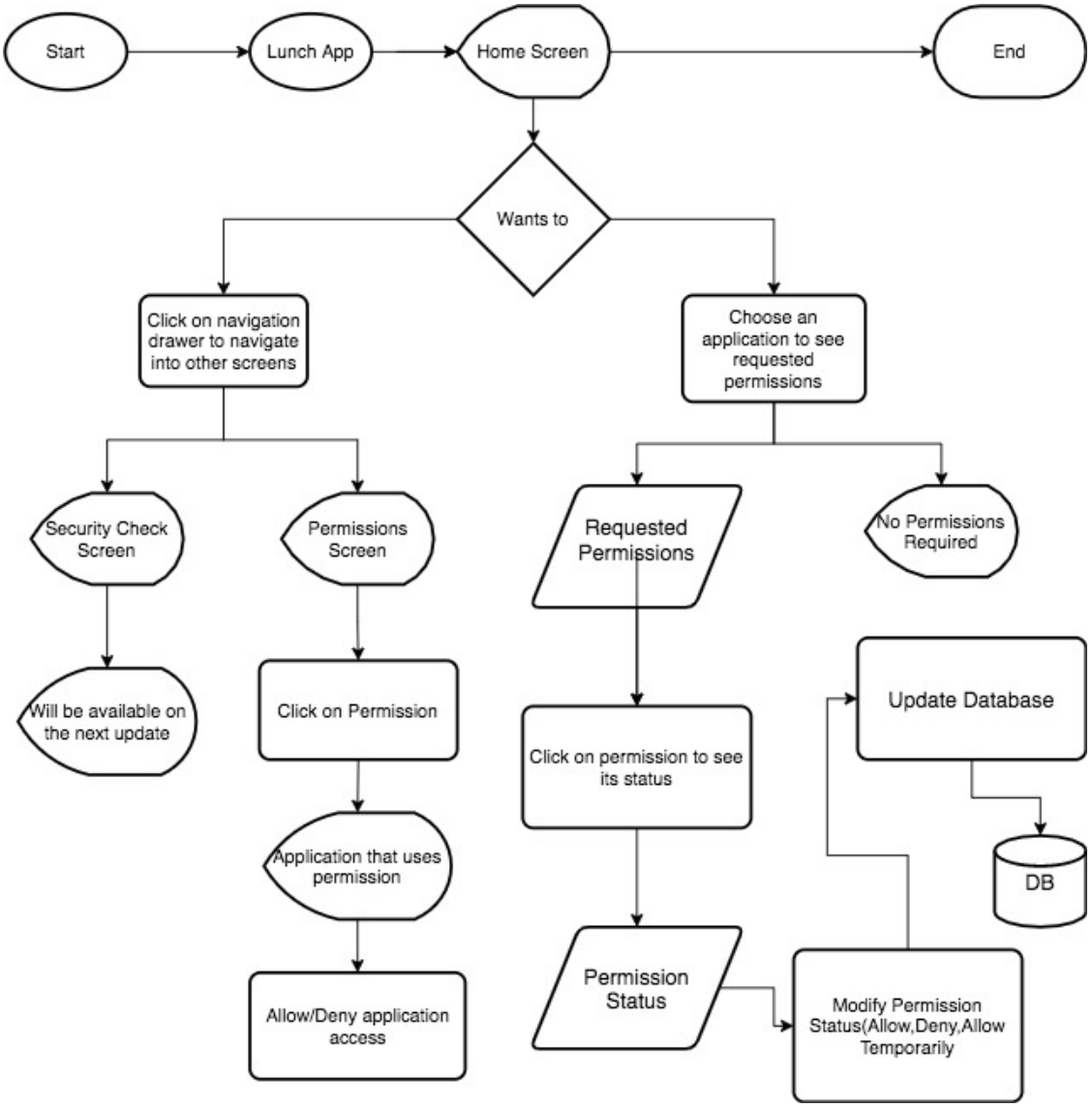

Figure 3: Detailed UCAAPM process

Implementation:

We used the Iterative model of software development. According to the functional requirements described before, and following the software development process in designing the tool by developing the use case diagram, the sequence diagrams, the class diagrams, the activity diagrams, and the package diagrams. UCAAPM was developed using Android Studio; the official IDE for Google's Android operating system, built on JetBrains IntelliJ IDEA software and designed specifically for Android development.

In UCAAPM, we used Room database system to store permissions. Android comes with built in SQLite database implementation, where Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. The use of the database is to store the user's preferences in a rule based transactions and to use the profile on other devices without having to build a new dataset.

The interfaces where designed on a user friendly basis, the screens were designed to fit automatically for different layouts and the navigation between screens is done smoothly. Figure 3 shows the home screen and the logo of UCAAPM:

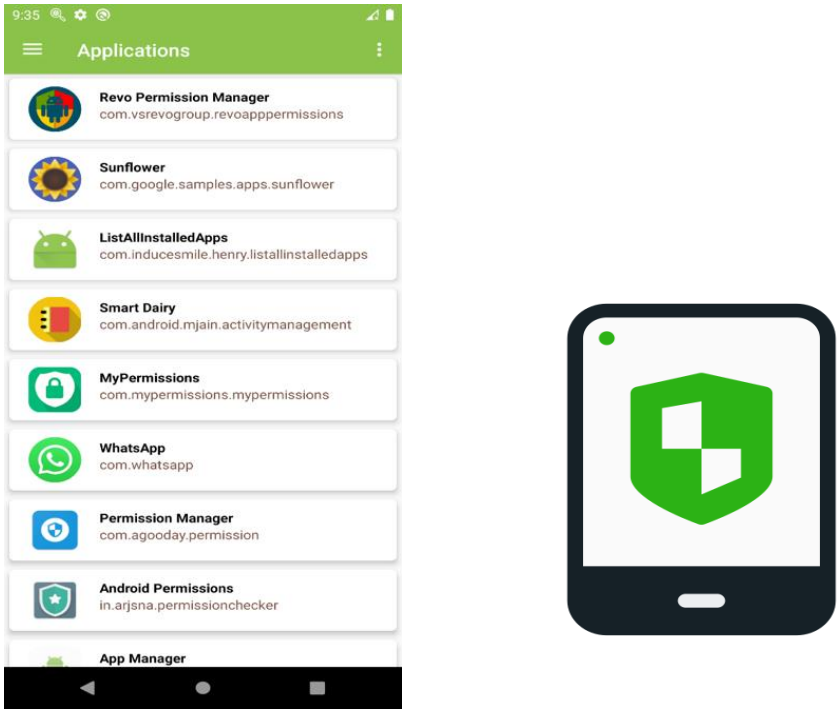

Figure 4: Home screen and logo of UCAAPM

Testing:

The testing phase is important to make sure that the product is according to the specified requirements and user friendly. A performance testing was performed to check the feasibility of required mobile phone resources (CPU, memory, and Energy). Usability testing was also carried out to measure usability of UCAAPM by users.

Performance Testing

During implementation phase and after it, different test types were carried out to make sure that all components are working correctly and efficiently. The code was optimized to gain the best achievable performance. Table 3 shows the performance test of UCAAPM while it is running. These measurements were done using Android Profiler tool embedded in Android Studio, the tool calculates the average CPU, Memory, and Energy usage for UCAPPM both in Foreground and Background.

Table 3: UCAAPM performance measurements

Resource	Foreground	Background
CPU	40%	**5%**
Memory	90 M	**35M**
Energy	Low-Medium	**Low**

Usability Testing:

For the user acceptance test, we distributed the first version of UCAAPM on 50 users and asked them to use the application for a month and then we collected their feedback about their usage. The following questions were presented to users:

1) Was UCAAPM easy to use?
2) Are you willing to continue using UCAAPM?

3) Do you recommend UCAAPM for others?
4) How do you rate the performance of UCAAPM?
5) How much do you trust UCAAPM?

They were asked to rate the application in a Likert scale from 1 which is the worst to 5 which is the best. The Likert scale is shown in Table 4.

Table 4: Likert scale

Strongly disagree	disagree	Neutral	agree	Strongly agree
1	2	3	4	5

The descriptive analysis of results is shown in Table 5:

Table 5: descriptive analysis of usability test

	Q1	Q2	Q3	Q4	Q5
Mean	4.6	4.4	4.8	4.1	4
Standard deviation	0.503	0.503	0.410	0.552	0.562

As shown by table 5, 92% of respondents found that UCAAPM is easy to use, 96% are recommending it for others. A lower percentage of 80% reported that they do trust the system; this means that UCAAPM is acceptable by users and trust might be increased by frequent use of the application by users.

Based on our performance and usability test, UCAAPM achieved the design requirements and the required specifications to manage user's permission in a flexible manner while maintaining user's preferences to adjust the permission handling process.

5. Conclusions

Android permission management has been exploited to launch some attacks such as escalation and Man in the middle attacks. These attacks might be mitigated if efficient access control management mechanism is enforced. In this work, we proposed a User Centric Android Application Permission Manager (UCAAPM) that helps users to effectively manage permissions for their applications. UCAAPM enables users to temporarily grant permissions and automate the grant/deny process. It also profiles user's preferences and enables them to transfer their profile to any other Android device without having to reconfigure it again. Performance measures showed that UCAAPM works efficiently when considering memory, CPU, and power consumption. And the user's feedback about using the tool were very positive and they are willing to continue using the tool and are recommending it for others.

REFERENCES

[1] McAfee Labs. (2018). McAfee Labs Threats report. McAfee Labs Teams.

[2] Kaspersky Lab. (2019, March 5). usa.kaspersky.com. Retrieved 10 31, 2019, from Kaspersky: https://usa.kaspersky.com/about/press-releases/2019_number-of-mobile-malware-attacks-doubles

[3] Malley, S, Craig, R. Security enhanced (SE) Android: bringing flexible MAC to Android. Auckland, New Zealand: NDSS, 2013

[4] Marforio, C, Francillon, A, Capkun, S. Application collusion attack on the permission-based security model and its implications for modern smartphone systems, vol. 19. Zurich: Department of Computer Science, ETH Zurich, 2010, pp.1–18.

[5] Heuser, S, Negro, M, Pendyala, PK. DroidAuditor: forensic analysis of application-layer privilege escalation attacks on android (Short paper). In: Grossklags,J, Preneel, B (eds) Financial cryptography and data security. Berlin: Springer, 2017, pp.260–268 ,

[6] Conti, M, Dragoni, N, Lesyk, V. A survey of man in the middle attacks. IEEE Commun Surv Tut 2016; 18(3): 2027–2051

[7] Wei, X, Wolf, M. A survey on HTTPS implementation by Android apps: issues and countermeasures. Appl Comput Inf 2016; 13(2): 101–117. ,

[8] Yagemann, C, Du, W. Intentio ex machina: Android intent access control via an extensible application hook. In: International conference on ubiquitous intelligence & computing and international conference on autonomic & trusted computing, Fukuoka, Japan, 4–7 September 2012. New York: IEEE.

[9] Feng, Y., Aiken, A., Anand, S., & Dillig, I. (2014). Apposcopy: Semantics-Based Detection of Android. Hong Kong: Fung Seng Enterprises.

[10] Felt, A., Song, D., Wagner, D., & Hanna, S. (2012, April 4). https://ptolemy.berkeley.edu/projects/truststc/pubs/848.html. Retrieved November 20, 2019, from ptolemy.berkeley.edu: https://ptolemy.berkeley.edu/

[11] Y. Aafer, W. Du, & H. Yin. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. In, Security and Privacy in Communication Networks (pp. 86-103). 2013 New York: Springer,

[12] Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012, February). "Andromaly": A behavioral malware detection framework for android devices. Journal of Intelligent Information Systems, pp. 161-190.

[13] Enck, W., Gilbert, P., Chun, B.-G., Cox, L., Jung, J., McDaniel, P., et al. (2014, June). TaintDroid: An Information-Flow Tracking System for Realtime Privacy. ACM Transactions on Computer Systems, pp. 1-6.

[14] Yan, L., & Yin, H. (2012, August 8-10). https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/yan. Retrieved November 20, 2019, from usenix: https://www.usenix.org

[15] Y. Peng, M. Zhang, J. Zheng and Z. Qian, "Research on Android Access Control Based on Isolation Mechanism," 2016 13th Web Information Systems and Applications Conference (WISA), Wuhan, 2016, pp. 231-235, doi: 10.1109/WISA.2016.53.

[16] Liu, G., Zhang, G., Wang, Q., Ji, S., & Zhang, L. (2019). Fine-grained access control method for private data in android system. International Journal of Distributed Sensor Networks. https://doi.org/10.1177/1550147719840232

[17] Ongtang, M, Mclaughlin, S, Enck, W. Semantically rich application-centric security in Android. In: Computer security applications conference, Honolulu, HI, 7–11 December 2009. New York: IEEE.

[18] Nauman, M, Khan, S, Zhang, X. Apex: extending android permission model and enforcement with user-defined runtime constraints. In: 5th international symposium on ACM symposium on information, computer and communications security, Beijing, China, 13–16 April 2010. New York: ACM.

[19] Inverardi, P, Salle, AD, Autili, M. User-centric Android flexible permissions. In: International conference on software engineering, Buenos Aires, 20–28 May 2017, pp.365–367. New York: IEEE.

[20] Aron, L, Hanacek, P. A concept of dynamic permission mechanism on android. AIP Conf Proc 2016; 1705: 020022.

[21] Bugiel, S, Heuser, S, Sadeghi, AR. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies. In: Proceedings of the 22nd USENIX conference on Security, Washington, DC, 14–16 August 2013. Berkeley, CA: USENIX Association.

[22] Rashidia, B, Funga, C, Vub, T. Android fine-grained permission control system with real-time expert recommendations. Pervasive Mob Comput 2016; 32: 62–77.

[23] Scoccia, Gian & Malavolta, Ivano & Autili, Marco & Di Salle, Amleto & Inverardi, Paola. (2019). Enhancing Trustability of Android Applications via User-Centric Flexible Permissions. IEEE Transactions on Software Engineering. PP. 1-1. 10.1109/TSE.2019.2941936.).

[24] Morales, LV, Rueda, SJ. Meaningful permission management in Android. IEEE Lat Am T 2015; 13: 1160–1166.