International Journal of Innovative Research in Engineering and Management (IJIREM)
ISSN: 2350-0557, Volume-10, Issue-2, April 2023
https://doi.org/10.55524/ijirem.2023.10.2.25
Article ID IJIR2365, Pages 125-129
www.ijirem.org

Android Mobile Devices Context Based Access Control Systems

A.Mounika[1], Choppara Sudheer Babu[2], Bandaru Ananda Babu[3], Ungarala Lohith Kumar[4], K Jaswanth Kumar[5]

[1,2,3,4,5] Department of Computer Science & Engineering, PACE Institute of Technology & Sciences, Ongole, Andhra Pradesh, India

Correspondence should be addressed to A.Mounika; mounika.a@pace.ac.in

ABSTRACT- The Mobile applications designed for Android frequently have access to private information and resources stored on the user's device. The improper use of this data by malicious programs has the potential to result in breaches of privacy as well as the disclosure of sensitive data. An example of this would be a malicious application capturing a confidential business chat in the background without the user's knowledge. The issue is caused by the fact that Android users do not have control over the capabilities of programs once those applications have been granted the desired access during installation. This is what causes the problem. It is necessary for us to have a context-based access control mechanism so that privileges can be dynamically provided or revoked to applications based on the specific context of the user. This is because there are numerous situations in which the possibility that an application will be allowed a privilege is contingent on the specific context of the user. In this article, we suggest an access control system along these lines. Our implementation of context distinguishes between subareas that are geographically adjacent to one another within the same location. We have made certain changes to the Android operating system in order to enable context-based access control limitations, which can then be set and implemented. We have carried out a number of tests in order to evaluate both the efficacy of our access control mechanism and the precision of context recognition.

KEYWORDS- Android Mobile, Smart Phone, Security And Privacy, Context Based Access Control

I. INTRODUCTION

Because of the increasing computing and communication capabilities of smartphones, application developers are beginning to take advantage of these capabilities in order to deliver new or improved services for the applications they create. However, the vast majority of these resources have the potential to acquire sensitive data and put users at great risk in terms of both their privacy and their security if programs use them in an incorrect manner and without the knowledge of the user [1]. The risk is posed when an application on a device behaves maliciously and exploits device resources to spy on the user or leak the user's personal data without the user's permission [2]. Additionally, users who carry their cellphones in public and private locations run the risk of inadvertently disclosing their private information and endangering their personal safety because they are unaware of the presence of such dangerous actions on their devices.

Users need to be able to have a greater control over the capabilities of their devices by decreasing the rights that particular applications have while they are in sensitive circumstances like confidential meetings. This would help avoid attacks like these from occurring. In order to accomplish this goal, smartphone operating systems need to give device owners with changeable rules. These policies should allow users to regulate the amount of system resources and application privileges that their device uses based on the context in which they are operating, specifically location and time. It is imperative that methods for giving customers such control over their devices be researched because popular smartphone operating systems, such as Android, do not yet include such a feature. These operating systems include Android.

The requirement for device restrictions that can be altered dependent on the environment applies not only to high-profile employees but also to common people who use smartphones. For instance, employers in the government, such as those in national labs [3] prohibit their workers from bringing any camera-enabled device to the workplace, including smartphones. This is the case despite the fact that workers may need to have their devices with them at all times because their devices may contain data and services they may require at any time. It is possible that employees will be permitted to use smartphones if context-based device policies are implemented. This is because employees will be able to prevent all applications from using the camera and any other device resources and privileges that employers restrict while they are at work. However, when employees leave the work area, the user device will be able to retain all of its original privileges. Politicians and law enforcement officers require context-based policies as well since they need to be able to turn off their devices' cameras, microphones, and location services during confidential meetings but keep these capabilities while they are in non-confidential areas of the building. Users are able to specify when and where their applications are permitted to access the data and resources on their devices thanks to context-based restrictions. This makes it significantly more difficult for hackers to steal such data.

Previous research on the security of mobile operating systems has focused on preventing applications from gaining access to sensitive data and resources. However, there are not many effective methods for enforcing these restrictions according to fine-grained contexts that differentiate between subareas that are close together. In addition, the majority of this work has been directed toward the development of poli-

cy systems that do not restrict privileges on an application-by-application basis and are only effective on a system-wide scale [4]. In addition, the policy mechanisms that are now in place do not cover all the various ways in which programs might access user data and the resources of a device. Last but not least, the location-based policy systems that are now in use are not precise enough to differentiate between neighboring sites without the use of additional hardware or location devices. The majority of the time, these systems treat the context as if it were already known, but they do not provide or evaluate the context detection mechanisms of mobile devices [5].

Context-based policy systems for mobile devices, such as smartphones, can be difficult to develop since they need to satisfy a number of constraints, including the following:

1) Applications should not be able to trick the device into thinking they are in a different location or time, nor should they be able to circumvent the policy limitations that are enforced on the device based on the context in which they are being used.

2) Because it is presumed that users are mobile, the policy limits ought to be implemented immediately on the device whenever there is a change in the device's position.

3) The accuracy of the location must be greater than the accuracy of the location determined by GPS since we need to apply different policies in different areas or close sub-areas that are located within the same GPS location.

4) The enforcement of context-based policies should not require application developers to modify source code or impose any new requirements on their applications. This is because these regulations are based on the context of the data being analyzed.

5) The policy that is being applied should not result in significant delays in the functionality of the device, as this could have a negative influence on the performance of the system.

In this work, we propose a context-based access control, or CBAC, mechanism for Android systems. This mechanism gives users of smartphones the ability to specify configuration settings that govern how their applications use various resources and services on the device depending on the context in which they are being used. Users have the ability to, for instance, set restricted privileges for device apps when using the device at work. When the device is used at home, however, device applications may regain their original privileges. This capability is made available to users using the CBAC mechanism. This modification to the user's device rights takes effect immediately after the user device is determined to be compatible with a pre-defined context of a user-defined policy. Additionally, the user has the ability to select a default set of policies that are to be applied whenever the user is positioned in a location that has not been previously defined.

The accessible device resources, services, and permissions that are granted to apps as they are being installed are what are used to establish the configured policy limitations that are in place. These policies restrict access to both the user's information and the device itself, thereby defining the services that are made available by the device. The user of the device is responsible for configuring the policy limitations that are associated to the context. We determine context by taking into account both time and place. Additionally, GPS and cellular triangulation coordinates are used to detect location whenever they are available. The primary method for determining location is based on visible Wi-Fi access points and the respective signal strength values of those access points. This allows us to discriminate between close sub-areas within the same work space.

We implement our CBAC policies on the Android operating system, and one of the features we provide is a tool that enables users to designate physical places using the collected Wi-Fi characteristics, such as their homes or places of employment. Users have the ability to be even more specific by differentiating between sub-areas within the same place, such as living rooms and bedrooms in residential settings or conference rooms and offices in commercial settings. Once the user has configured the device policies that specify device and application rights based on context, the rules will be automatically applied whenever the user is within a pre-defined physical location and time interval. This will occur whenever the user is within the pre-defined physical location.

We installed our CBAC policies on the most recent Android device, the Google Nexus 7 tablet, as well as the Google Nexus 4 phone in order to evaluate how effectively these policies may be enforced in a variety of settings. Our analysis included the submission of 250 applications as well as several test cases in a variety of settings across the grounds of our university, where there were also several Wi-Fi access points established. The results of our experiments demonstrate the influence that CBAC policies have on Android applications as well as the impact those policies have on the performance of the system.

The second section presents fundamental ideas and essential background knowledge. In Section 3, we go over the design of our architecture and present the access control framework that we have developed.

II. RELATED WORK

Google's Android is the name of the operating system that powers a wide variety of mobile devices, including smartphones and tablets, all of which are grouped together under the umbrella term "Android mobile devices." Since its conception, Android has grown to become the mobile operating system that is used the most widely all over the world. It now powers billions of devices that come from a broad variety of manufacturers and price points. The following is a list of important characteristics of mobile devices running Android:

Operating System and User Interface: Android's operating system is based on the Linux kernel and is open-source. It also features a customizable user interface. It supports a wide variety of applications, features, and functionalities, and provides a user interface that may be highly customized to suit individual needs. On top of Android, many manufacturers layer their own uniquely crafted user interfaces (UI), which allows for more personalized user experiences. The One UI from Samsung, MIUI from Xiaomi, and OxygenOS from OnePlus are three examples of well-known custom user interfaces. An vast app ecosystem is one of the characteristics that sets Android apart from other mobile operating systems. The Google Play Store is the official app distribution channel for Android, and it houses millions of different apps, including anything from social media and productivity tools to games and multimedia programs. Apps may be quickly downloaded and installed by users, making it simple

for them to customize their devices in accordance with their individual interests [6].

Diversity in Hardware: Android devices can be found in a wide range of form factors, including both dimensions and internal components. The capabilities of the cameras, the display quality, the processing power, and the amount of time a battery may last might differ greatly amongst devices made by various manufacturers. Because of this variety, customers are able to select products that are suitable for their individual requirements and financial constraints.

Personalization and Customization: Android devices are well-known for the high levels of both customization and customization that they provide their users. Users are able to customize the look and feel of their smartphones by altering elements like as the themes, wallpapers, launchers, and widgets. Because of its adaptability, it enables users to craft one-of-a-kind and individualized user experiences.

Connectivity & Integration: Android devices offer a seamless integration with Google services such as Gmail, Google Drive, Google Photos, and Google Maps. Android devices also offer a variety of connectivity options. In addition, Android smartphones offer a broad variety of connectivity options, including Wi-Fi, cellular data, Bluetooth, NFC (Near Field Communication), and a lot more besides.

Updates and Versions: Android's operating system is updated on a regular basis with new versions that each provide a number of enhancements, new capabilities, and heightened levels of security. However, the procedure of updating can be different from one device maker to another and from carrier to carrier. The update cycle may be modified by factors such as hardware compatibility and regional restrictions. Typically, newer devices receive upgrades before older ones do, but the timing of these might vary [7].

Safety and Confidentiality: Google has established a variety of safety precautions to safeguard Android devices against malware and other forms of attack. Enhancing a device's security can be accomplished with the use of features like Google Play Protect, app permissions, and biometric identification (fingerprint or face recognition, for example). Users can also modify their data privacy settings and have control over the permissions that apps have access to. Multimedia and Entertainment: Android devices provide a comprehensive multimedia experience, which includes high-quality displays for the purposes of watching videos and playing video games. Users are able to indulge in a wide variety of kinds of entertainment thanks to the availability of a sizable library full of movies, television series, music, books, and magazines within the Google Play Store.

In a nutshell, mobile devices that run the Android operating system have fundamentally altered the ways in which we carry out our work and pleasure ourselves. Because of their open-source nature, numerous hardware possibilities, huge software ecosystems, and the ability to be customized, they have become a cornerstone of modern technology, serving to the desires and requirements of a wide variety of users [8].

III. PROPOSED WORK

The access control mechanism that manages the access, collection, storage, processing, and utilization of context information and device policies is a component of our framework.

The Context Provider (CP) is responsible for collecting the physical location parameters from the device sensors (GPS, Cell IDs, and Wi-Fi parameters) and storing them in its own database. This links each physical location to a logical location that has been set by the user. When the device is moved to a new location, those parameters are checked for accuracy and then updated.

The Access Controller (AC) is responsible for regulating the permissions granted to applications and preventing illegal access to the resources or services provided by the device. This system is supplemented by the AC with additional control methods and specific fine-grained control permissions that better reflect the application's capabilities and narrow down its accessibility to resources. Even though the Android OS has its own permission control system that determines whether or not an application has the privileges to request resources or services, this system is used by the AC. Because the current Android system has some permissions that, if granted to applications, may provide applications more accessibility than they need, which malicious code can take advantage of, the AC improves the security of the device system. This is because the existing Android system has some permissions. For instance, the READ PHONE STATE permission provides privileged apps with access to a set of data that includes the phone number, the IMEI/MEID identifier, subscriber identification, phone state (busy/available), SIM serial number, and more.

The Policy Manager, often known as PM, is the interface that is utilized in the process of formulating policies, which mostly entails assigning application limitations to contexts. Its primary function is to allow the user complete control over configuring which resources and services are available to apps within a particular context that is provided by the CP. As an illustration, the user can create a policy using the PM to permit location services only when the user is present at their place of employment on weekdays between the hours of 8:00 am and 5:00 pm.

The Policy Executor (PE) is responsible for enforcing device limits by comparing the context of the device with the policies that have been configured. The PE performs a check of the user-configured limitations that have been set at the PM whenever an application makes a request for access to a resource or service. Based on the results of this check, the PE will either grant or refuse the application's request for access. The PE is responsible for resolving any policy conflicts that may arise and implementing the most stringent limitations. In addition, the PE functions as an enforcer of policies by providing the AC with the authorization information needed to process application requests.

Access Control Framework: Users could develop CBAC policies through the use of the PM by configuring application limits and attaching those restrictions to contexts. At the point in time when an application makes a request for a resource or service, the AC checks to see if the application request is approved and then sends the request along to the PE for processing.

After determining whether the request can be granted, the PE will look to see whether any policies exist that are relevant to the application request.

If such a policy is in place, the PE will make a request to the CP to retrieve the context at the time the application makes its request.

The PE will then make a comparison between the retrieved context and the context that has been established by the policy. In the event that a match is found, the PE is responsible for enforcing the relevant policy restrictions by providing

feedback to the AC so that the AC can apply those restrictions to the application request.

We take great effort in the design of the access control framework to ensure that the user-configured policies are reliably implemented with the fewest possible processing steps and in the shortest amount of time possible in order to prevent any substantial delays in providing a response to the application that requested it. In light of the fact that our design should be able to safely manage policy execution, we preserve the context data that is provided by the CP in such a way that it is accurate, precise, and up to date.

IV. RESULTS AND DISCUSSIONS

The Wi-Fi access points (APs) and the signal strengths associated with each one are the primary sources of location-related information that are utilized by our access control system. The location information that is gathered from GPS and cellular towers is also compiled into our context definition; however, this may not be enough for accurate indoor localization due to the fact that GPS and cellular tower signals can become unreliable or nonexistent inside buildings or areas that are contained within building structures. However, location information collected from Wi-Fi characteristics has the potential to be more accurate when attempting to differentiate between sub-areas that are closely placed inside the same GPS location [9].

The combination of GPS coordinates, cellular triangulation location data, and Wi-Fi access points (APs) and signal levels is used to depict a spatial region. By executing the Android LocationManager service, users of Android can retrieve both GPS coordinates and cellular triangulation in a manner that is functionally equivalent. After invoking the LocationManager, we then request location updates by calling the requestLocationUpdates method. This method returns a Location object that has information such as latitude, longitude, a timestamp, and other data.

Wi-Fi operates in a manner distinct from the other two methods of location determination. In order to get the Wi-Fi parameters, we first invoke the WifiManager service in order to retrieve the scans of the Wi-Fi access points. We begin by registering our Broadcast Receiver mWifi receiver with an IntentFiler action in order to receive the broadcasted Wi-Fi scanned intent. After that, we make a request for the actual scanned access points data, which we then proceed to process.

Users of our CBAC policy system have access to a tool that allows them to specify physical locations. This may be done either by manually inputting the area location coordinates or by recording snapshots of location data of the areas that are wanted. In the following sections, we will show our design and implementation of the location capturing phase, which is when users specify and save physical locations; the location detection phase, which is when a device detects its location and matches it with a pre-defined policy context; and the location analyzing phase, which is when a user analyzes their location data to determine what policies should be applied; and the

The purpose of this experiment is to determine how accurate the location detection algorithm that our CBAC mechanism employs is and report our findings. We count both the number of successful and unsuccessful detections in each individual sub-area. The floor plans and wiring diagrams for one of the buildings in which we carried out some of our experiments. The huge rectangles arranged in a grid pattern serve to highlight important sites or regions, which are denoted by numbers. The areas that are not contained inside the rectangles are referred to as "unregistered." The black circles represent the particular sub-areas that were investigated during the phase of location capturing. Every other hue represents a different sub-area that was investigated during the detection phase.

Our tests, on the other hand, were conducted in a variety of locations and buildings. In each room, we selected at least four different areas to take part in the location capturing phase in order to gather location-related data and to build a reliable set of location parameters for each room that will be saved in the database. This was done in order to ensure that the information we collected was accurate. At each of the three different locations, we conducted research on one of the three smaller areas denoted by the letters 'A', 'B', or 'C' and measured the detection rate in each of these smaller areas. We took a picture of the five Wi-Fi access points on that particular level that had the strongest signal strength and ranked them based on how many sub-areas we tested them in. That floor has more than 15 Wi-Fi access points. We carried out fifty location tests in every section of the room and tallied the number of regions in which we were successful in locating the target. According to the findings of our experiments, the effective detection rates were as high as 91%, while the number of incorrect detections we made reached as high as 29% in the worst case scenario. This experimental finding was supplemented by evaluating a few "unregistered" spaces located in close proximity to the registered rooms. We found that 16% of the results were false positives, which are unregistered areas that gave off the impression of being user-defined. The values of the signal strengths of matching Wi-Fi access points were found to fall within the range of signal strengths initially recorded during the phase of location capturing. This was the case within the areas that were registered. However, in the unregistered areas, the values dropped outside of the range that had been saved due to the presence of building structures that interfered with the Wi-Fi signal strengths. This was especially true the further away the device was when the snapshots during the location-capturing phase were taken [10].

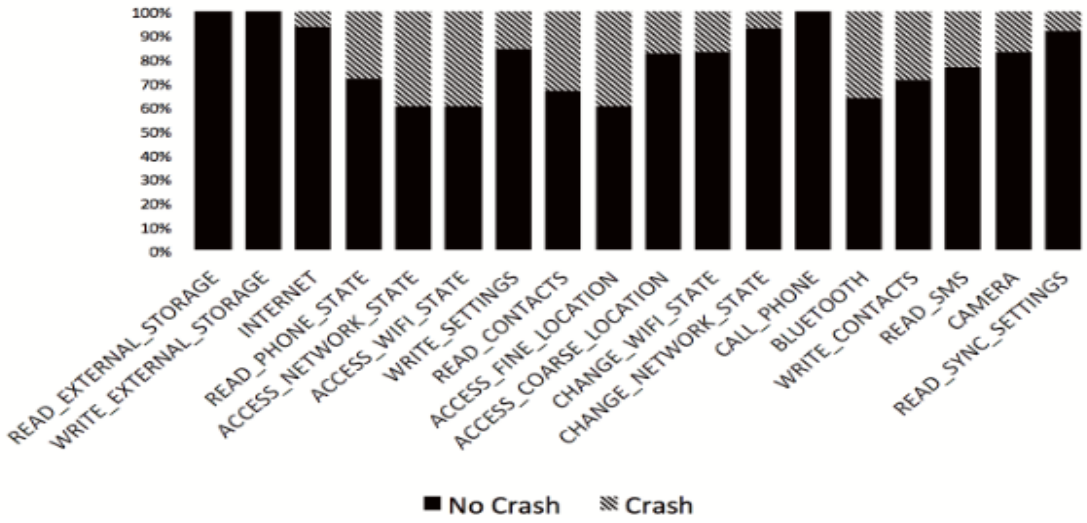

Figure 1: Report of permission revoking of application

V. CONCLUSION

The context-based access control policies was one of the proposals made in this body of work. These policies prevent programs from accessing certain data and/or resources on the basis of the context in which the user is working. As soon as the device being used by the user meets the pre-defined context that is linked with the policy, the policy's associated constraints are immediately put into effect. The outcomes of our experiments demonstrate that these policies have a positive impact on the Android system and applications, as well as the degree of precision with which the device may be located inside a user-defined environment.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] Ali-Gombe, Aisha, et al. "AspectDroid: Android app analysis system." Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. 2016.
[2] Shrivastava, Gulshan, et al. "Privacy issues of android application permissions: A literature review." Transactions on Emerging Telecommunications Technologies 31.12 (2020): e3773.
[3] Cao, Weicheng, et al. "A large scale study of user behavior, expectations and engagement with Android permissions." 30th USENIX Security Symposium (USENIX Security 21). 2021.
[4] Wikipedia, "Samsung galaxy s4 specifications," http://en.wikipedia.org/wiki/Samsung Galaxy S4, May 2013
[5] Talal, Mohammed, et al. "Comprehensive review and analysis of anti-malware apps for smartphones." Telecommunication Systems 72 (2019): 285-337.
[6] A. Kushwaha and V. Kushwaha, "Location based services using android mobile operating system," International Journal of Advances in Engineering and Technology, vol. 1, no. 1, pp. 14–20, 2011.
[7] J. Leyden, "Your phone may not be spying on you now – but it soon will be," http://www.theregister.co.uk/2013/04/24/kaspersky mobile malware infosec/, April 2013.
[8] R. Templeman, Z. Rahman, D. J. Crandall, and A. Kapadia, "Placeraider: Virtual theft in physical spaces with smartphones," CoRR, vol. abs/1209.5982, 2012.
[9] Wang, Chenwei, et al. "GPS 5.0: an update on the prediction of kinase-specific phosphorylation sites in proteins." Genomics, Proteomics & Bioinformatics 18.1 (2020): 72-80.
[10] Demissie, Biniam Fisseha, Mariano Ceccato, and Lwin Khin Shar. "Security analysis of permission re-delegation vulnerabilities in Android apps." Empirical Software Engineering 25 (2020): 5084-5136.