International Journal of Industrial Engineering
2023;7(2):65-72.
ISSN: 2456-8449

Android Spy Camera System to Enhance Security Using Super Resolution Images

T Sivakumar[1*], K. Mahalakshmi[2], P. Shobha Rani[3], B. Maruthu Kannan[4]

[1]Department of Computer Science and Engineering, KPR Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.
[2]PG & Research Department of Commerce, Thanthai Hans Roever College (Autonomous), Perambalur, Tamil Nadu, India.
[3]Department of Computer Science and Engineering, R.M.D. Engineering College, Chennai, Tamil Nadu, India.
[4]Department of Computer Science and Engineering Sphoorthy Engineering College, Hyderabad, Telangana, India.

[*]Corresponding author: Sivakumar.t@kpriet.ac.in

Abstract. Studied the possibility of a "transplantation attack," in which malicious apps could take privacy-harming images to spy on users without the Android Application Programming Interface (API) auditing being aware of it, based on the observation that spy-on-users attack by calling Android API would be detected out. Typically, applications will need to make API calls to the Android Camera Service, a media server process, to snap a photo. To carry out a transplantation attack, a malicious program must copy the code responsible for taking photos from the media server process and then use these codes to take photos in its own address spaces, by passing the need for inter-process communication (IPC). This allows one to avoid being audited by the API. The results of studies show that the transplanting assault is real. The spy-on-user assault is also made more covert by the transplanting attack. Found that approximately half of the 69 Smartphones evaluated (produced by eight manufacturers) were vulnerable to the transplanting attack developed. In addition, the assault may hide from seven different antivirus scanners and the Android Device Administration APIs, which are utilized for enterprise-level mobile device administration. The transplanting attack prompted us to find a minor flaw in Android's security architecture and code using a super-resolution algorithm. The results show the scalability scores for Android versions.

Keywords: Android, Spy on Users, Transportation Attacks, Android Camera Services, Super Resolution Algorithm

INTRODUCTION

The police in Malaysia have been engaged in several cases involving bullying, school absences, and the disappearance of children, yet these problems persist. Children who are left off at school or on the playground without adult supervision run the danger of being abducted, bullied, or sexually assaulted. There is a necessity for parents to watch their kids and even know their child's whereabouts while they are outside. Therefore, the goals of the proposed systems are to assist parents in keeping tabs on their child's whereabouts in potentially dangerous situations. This article looked at how to create an Android app for parents called Closed-Circuit Mobile TV (CCMTV) [1]. Since many parents today are working long hours and can't devote much attention to their children, this app was designed to fill that gap. Android parental surveillance software is a monitoring tool that records audio and video and keeps tabs on a child's precise Global Positioning System (GPS) position. Both programs (parent and child) in this system need to be run when connected to the internet. To function as spying software that relies on Google API but doesn't need rooting, it is an Android development project that takes advantage of the access permissions of Android devices [2].

The top five parental control apps of 2019 served as the basis for research of comparable monitoring applications to identify areas for improvement. Firebase is being used as the server in a client-server architectural setup, with the client acting as both the parent and the child. Live streaming is another feature of this system that depends on the media server. This system was designed using structured evolutionary prototyping with increments approach, and the only necessary hardware is an Android smartphone and laptop. The creation of these applications was utilizing Android Studio platforms. Parental supervision over their child's Smartphone use is limited to passive, covert surveillance only [3]. It is becoming more common for tiny covert surveillance cameras to be placed in sensitive spots like hotels and toilets. Due to their modest size, these concealed cameras are readily

Received: 21.08.2023 Revised: 05.10.2023 Accepted: 24.10.2023

available for purchase and are almost impossible to detect with the human eye. Current methods for detecting these cameras are insufficient since they rely on expensive and niche technology and have poor detection rates. Studies of the wireless traffic generated by covert cameras have been proposed in recent scholarly publications. These solutions, however, are similarly restricted since they presume wireless video streaming while only being able to identify the existence of the concealed camera and not their positions.

Using the times-of-flight (ToF) sensors found in inexpensive Smartphones, introduce a revolutionary concealed camera detection and localization method. Implement a mobile app that sends out laser signals from the ToF sensors and then utilizes computer vision and Machine Learning (ML) to track down the specific reflection captured by covert cameras. Extensive real-world trials with 379 people were conducted to assess and find that it has an 88.9% success rate in detecting concealed cameras, whereas the success rate for using simply one's naked eye is only 46.0% [4]. Consumer mobile spyware applications secretly track a user's movements and broadcast that data over the internet so that remote monitoring may be carried out. So-called "stalker ware" applications expose a far greater spectrum of victims to intrusive surveillance since they are mass-marketed to customers at retail, unlike theoretically identical apps employed for state espionage. The market for such applications has grown to the point where dozens of rivals may thrive, with some providers apparently keeping tabs on hundreds of thousands of mobile devices. While the academic world is aware of the existence of such applications, knowledge of how they work is still patchy at best [5].

This paper provides an in-depth technical examination of 14 separate popular mobile spyware programs targeting Android phones. Catalog the numerous methods of tracking user actions (pictures, texts, live microphone access) via the inventive use of Android APIs. Also, uncover previously unreported techniques used by these applications for evading detection and establishing a persistent presence. Also, keep track of the safety precautions each program takes to guard its users' personal information and reveal several shortcomings on the side of spyware providers (such as private information being sent in the open or kept in an unencrypted cloud) [6]. Massive privacy concerns have been brought to light using hidden cameras in public spaces, including motels, hotels, homestays (i.e., Airbnb), and bathrooms. Wifi spy cameras are widely employed by many opponents due to their cheap cost, ease of installation, and small size. Most research has shown that detecting wireless cameras using video traffic analysis is insufficient to avoid privacy intrusions and that additional synchronous data from external sensors or stimulus devices is necessary to prove the user's movements. Users experience discomfort, and more time and effort are needed to adjust to such supplements. In this research, we offer DeepDeSpy, a system that can identify the recording of spy cameras with zero input from the users. The goal is to see whether the user's motions are being recorded by the wireless camera by analyzing the channel state information (CSI) and the camera's network traffic.

Because of this, it might be difficult to identify motion in a massive quantity of CSI data in real time. To solve this issue, use the deep learning techniques of convolution neural networks (CNNs) and bidirectional long short-term memory networks (BiLSTMs). By automatically extracting significant characteristics from the sequential raw CSI data, quick and accurate detection is made possible by such synergistic CNNs and BiLSTM deep learning models. DeepDeSpy was tested in real-world circumstances (including a range of room sizes and user motions) to confirm its viability and provide a smooth transition from theory to practice. Across a variety of real-world scenarios, accuracy averaged about 96%, peaking at 98.9% during vigorous exercise in a big space. Real-time applications are feasible because of the capability of quick detections on Smartphone within just a one-second reaction time [7]. The problem statement is discussed below. Discreet and effective monitoring solutions are in high demand in the field of security and surveillance. Developing a Super Resolution Algorithm–integrated Android Spy Camera System to enhance security is a matter of concern here. We need to apply sophisticated technologies to fix existing surveillance systems that have problems with low-resolution photography, lack of discretion, and inefficient data processing. It is difficult to acquire and identify important features in surveillance films with conventional spy cameras due to the low-resolution pictures they typically create. The surveillance system's ability to detect and react to security threats might be hindered by poor picture quality. There can be lags in recording, sending, and analyzing video due to inadequate processing of surveillance data on Android devices.

Missed chances to identify and react to security events in real time might be caused by inefficient data processing. It may be difficult to extract useful information from recorded videos using older surveillance systems due to a lack of sophisticated picture enhancement capabilities. The clarity and detail of surveillance photography may be greatly enhanced by applying a Super Resolution Algorithm to the photos. Improved picture quality with the use of a Super Resolution Algorithm is essential for improved object, person, or activity recognition in surveillance film. The real test will be in getting this algorithm to work flawlessly with an Android-based spy camera system without sacrificing speed. To avoid unwanted access or interception of critical information, it is vital to ensure the safe transfer of surveillance data from the Android device to a central monitoring system. The project's goal is to increase overall security measures by developing an Android spy camera system that is strengthened with a super resolution algorithm. This system will solve these critical difficulties and deliver a discreet, high-resolution, and efficient surveillance solution. Improving the quality of recorded images, making sure monitoring is happening in real time, and optimizing resource utilization are all parts of this process for effective threat identification and response. The following are the contributions.

Developing and deploying a Super Resolution Algorithm-based Android Spy Camera System may greatly improve security measures in a range of contexts. Major benefits to productivity from such a system include the following: Surveillance video now has better picture quality thanks to the use of a Super Resolution Algorithm. Security surveillance is made more effective with higher-quality images since people, objects, and actions can be more accurately identified. With its stealthy form, the Android Spy Camera System can fit in with any setting, allowing for covert and subtle observation. The technology may be used in situations when overt monitoring would not be feasible or successful because of its enhanced discreteness. Prompt identification and reaction to security issues are made possible by the system's real-time monitoring of acquired video. To better handle security issues as they emerge, the system may instantly create warnings based on established parameters. By producing sharper and more detailed pictures, the Super Resolution Algorithm helps security staff see possible dangers more precisely, which in turn leads to better threat identification. Improved picture quality helps spot important features that low-resolution surveillance systems could overlook. To prevent unauthorized access or interception of surveillance data, the system employs encryption technologies to prioritize safe data transfer. Ensuring the integrity and secrecy of the acquired video during transmission to a central monitoring system is the primary goal of secure data transfer. Security staff can make better judgments with clearer images, reducing the chance of misinterpretation, thanks to the Super Resolution Algorithm's enhanced picture quality and powerful analytics, which help to reduce false alarms. Put simply, the Android Spy Camera System, which has been upgraded with a Super Resolution Algorithm, helps to tighten security by offering covert, high-quality monitoring in real-time. All these improvements make conventional surveillance systems more effective and provide guards with better means of detecting and responding to threats.

The following section will be a survey discussed in section 2, and the proposed system using the Super-resolution algorithm for the Android spy camera system will be discussed in section 3. Then, the results and discussion for the given dataset are discussed to enhance security in section 4. Finally, the conclusion provides the overall performance of the Android spy camera system and future work.

LITERATURE SURVEY

There is no denying the importance of Internet of Things (IoT) technology to daily life. Smartwatches, refrigerators, lights, and locks are just a few examples of the many ways that technology has improved daily lives. Smartphone applications, which are becoming more popular, are often paired with such devices for the benefit of easier monitoring and administration and are sometimes even necessary to run the device. However, installing and using such applications may increase the IoT device's own attack surface, leaving the user vulnerable to security and privacy issues. Therefore, a crucial issue emerges: do these programs restrict themselves to the bare minimum, and are they also safe against commonly exploited vulnerabilities and flaws? [8]. To address the issue, this paper analyzes the top 40 Android applications in six different popular categories of IoT devices. After carefully connecting each app with actual IoT devices, do a comprehensive static analysis on all of them and a dynamic analysis on almost half of them. Trackers manifest data, shared software, and other concerns are only a few examples of the many dimensions over which the gathered findings extend. The brief response to the presented question is that a wide variety of security and privacy vulnerabilities continue to plague most of such applications, which, in turn, represents the overall tendency in this ecosystem [9].

Android as an operating system is currently progressively being implemented in industrial information systems, notably with cyber-physical systems (CPS). As a result, there is a growing number of polymorphic and metamorphic malicious programs that target Android devices, putting them on the front lines of managing security-related data and executing sensitive activities. More precise detection and monitoring of sensitive Android app activities is crucial to the safety of CPS and IoT devices using Android because of the prevalence of malware threats like these. However, because of limitations in the Android security and privacy paradigm, enabling dynamic app activity tracking and identification on actual CPS is difficult.This essay explores how recent developments in deep learning could provide a more precise solution to this security issue. This paper presents a deep learning engine for detecting malicious app activity by analyzing system-wide parameters like free storage and network traffic volume and categorizing them using deep neural networks trained on the Encoders and ResNets models. Meanwhile, sparse learning is employed to decrease the number of valid parameters in the trained neural networks, which is useful for dealing with the resource constraints of conventional CPS and IoT devices. Despite the possibility of overlap between background noise and the intended behaviors, evaluations reveal that the suggested model is superior to a set of predefined baselines on time series categorization [10].

Logic bombs are a common technique used to conceal malicious behavior during dynamic test campaigns by delaying the execution of harmful operations until a predetermined set of circumstances has been met. There is still no clear solution in the literature for neutralizing logic bombs. As a first step in logic bomb triage, I advise looking into Suspicious Hidden Sensitive Operations (SHSOs) in this study. To find SHSOs, which are believed to be logic bomb implementations, creates a unique hybrid method that combines static analysis with anomaly detection approaches. Difuzer, by combining an instrumentation engine with a study of the flow of data across procedures, can pinpoint potential SHSO entrance locations. The SHSOs are then characterized by characteristics extracted from the triggers and an unsupervised learning model for identifying aberrant triggers is implemented using One-Class Support Vector Machine (SVM). Conduct an evaluation of the prototype and demonstrate that it is 99.02% accurate in detecting SHSOs, of which 29.7% are logic bombs. By exposing more logic bombs and producing fewer false positives in about an order of magnitudes less time, Difuzer exceeds the state-of-the-art. Give out all the items to the public [11].

Unauthorized sensing and surveillance of routines are real concerns in this age of ubiquitous IoT devices. A major privacy concern/threat is the disclosure of photographs captured by wireless spy cameras in private areas, including hotels, airports, public bathrooms, and public showers. Proposed a Spy Camera Finders (SCamF) that utilizes pervasive Smartphones to identify and localize wireless spy cameras by analysis of encrypted wifi network data, mitigating/addressing this important topic. SCamF accurately verifies the presence of wireless cameras on wifi networks and determines whether they are recording users' activities by characterizing the network traffic pattern of the wireless camera and then reconstructing encoded video frame size from encrypted traffic [12]. By evaluating the sizes of reconstructed video frames, SCamF can also correctly pinpoint the location of hidden cameras. Using a real-world testbed with twenty different kinds of wireless cameras, have successfully built SCamF on AndroidsSmartphone and analyzed its performances. Based on experiments, SCamF is able to (1) correctly categorize wireless cameras with an accuracy of 0.98; (2) detect spy cameras among the classified wireless cameras with a true positive rate (TPR) of 0.97; (3) incur low false positive rates (FPRs) of 0 and 0.031 for non-camera device and camera not recording the users' activities, respectively; (4) locates spy cameras with centimeters-level distances error [13].

There has been a lot of worry about the security and safety concerns with doors and their construction, which is ironic given that doors are supposed to protect people, places, and property and must be shut when not in use to have a protected house. Most doors nowadays use mechanical locks and keys that are insufficient to prevent access even to authorized personnel. This security solution, a smart door lock system with built-in spy cameras, may be used to monitor who is at the door and make sure that the visitor is safe before allowing entry. The door locking and unlocking procedure is implemented by using an ESP-32 (Artificial Intelligence (AI)-enabled interfaced with an Arduino microcontrollerATMEGA328P [14]. Using Global System for Mobile Communication (GSM) technology, a surveillance camera, an alarm system, and a web app, developed and deployed a locking system for doors. The device employs a camera for spying on things, and it sends the footage to a phone or computer over wifi, so you may lock and unlock the door from anywhere you happen to be. The door unlocks, and the authorized user is granted entry when the system detects a call from a registered mobile phone, a command from its mobile apps, or the input of the Internet Protocol (IP) addresses and passwords. After a short delay (a few microseconds), the front door locks to keep out intruders. Once again, the system will ask for one of the two methods to provide access whenever a new visitor comes. The system was created through real-time testing, where it performed comparably to similar works that didn't use the methodologies [15].

PROPOSED SYSTEM

Camera Service offers the functionality of utilizing the camera devices to, e.g., capture a photo, record a video, etc. Between versions 2. x and 4. x, Camera Service has a drastically altered process. Here, takes a closer look at how Android 4. 's Camera Service works since this version of Android has risen to prominence in recent months, according to a study conducted by Google. The Android operating system is built on top of Linux. Daemon processes, the media server process, the system server process, the service manager process, and the Android application processes are all examples of processes (address spaces) that make up the Android system. Any app that makes use of the Camera Service is a client. The client process is organized in a normal five-layer stack, from the application layers to the framework layers, to the Android Applications Runtime layers, to the hardware abstract layer (HAL), and to the Linux kernel layer. Java is used for both the Application and Framework layers. For running Java programs, the Runtime layers provide Dalvik Virtual Machines (DVM). This layer also contains various native libraries to take care of the IPC demands. Since the client processes do not need to communicate with the Camera devices directly, there is no such code present in the HAL layer.

Since the Linux kernels layer's memory page containing the kernel codes is shared by all processes, the camera driver code is also located in the client address spaces. The server media process runs in its native environment. The media server process does not need DVM since native processes do not include Java code. The media server operation simply consists of three stages. System libraries written in native codes (so libraries) are the only contents of the top layer, which is why they are referred to as the systems library layers. The Library System layers, so libraries are used to process a client's request, send it on to the HAL layer, get the answer, and send it back to the client. There are. So, libraries are on the HAL layer as well. Different from the Systems Library layers, libraries on these layers are utilized to interact with the camera drivers. Most of the tasks for the media server are being handled by the HAL layer. The camera operator is in the Linux kernels layer of the media server process. Before a client may exchange data with the Camera Service via binder IPC, it must first contact the service management process to get the Camera Service's reference.

The service manager is a native process that keeps track of all the system services that have been registered and the references to those services. It is a standard IPC operation to ask the service manager for a reference to the Camera Service, but this is not included in the diagram. Once the Camera Service reference has been received, the following is a high-level description of how the Camera Service is meant to be used. The client must establish a connection with the Camera Service before it can do any camera-related tasks. Once the connection is made, the client has access to all features of the Camera Service. Figure 1 shows the system architecture of the proposed system.

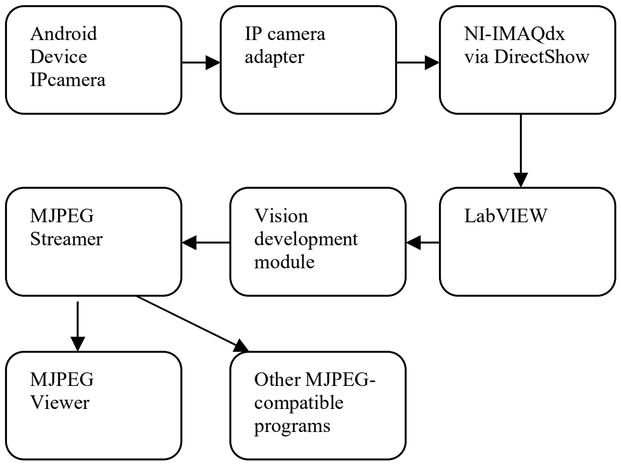

FIGURE 1. System architecture of the proposed system

The principle of super-resolutions is to create a high-resolution picture or series of pictures by combining many low-resolution (noisy) photos of a scene. Therefore, it uses a collection of lower-quality observed photos to try to recreate the original scene image with higher resolution. The standard method treats the low-resolution pictures as if they were produced by re-sampling a high-resolution original. After that, using the input photos and the imaging models to resample will provide the low-resolution observed images; the aim is to recover the high-resolution images. Therefore, super-resolutions rely heavily on an accurate imaging model, and further degradation of the picture might result from inaccurate modeling, for example, of motion. To become connected, the clients first access an Android API from its Java codes. The CONNECT binder request is sent from the Android API to the media server process using a system library (lib-camera clients). The Camera Service in the System Library layers extracts the request types from the binder data structures when they are received by the media server process. The Camera Service component will defer to the system server process for authorization if the requested action is a CONNECT. Go over how to utilize this to do an API audit later. The camera hardware interface in the System Library layer will be called by Camera Services when the client has successfully passed the permission check.

To communicate with the camera drivers in the Linux kernels, the Camera Hardware Interface routines will make calls to the HAL layers functions. A notifications thread in the HAL layer awaits camera events like "Camera has focused" or "focus has moved," among others. The camera driver will notify the thread when the camera is done taking a photo. The picture data will be sent back to the System Library layers of the media server processes via the callback routines when the notification thread gets the event. The Camera Service component may provide the image data to the client process via Binders IPC after it has been compressed (by the camera drivers) into a specific image format (for example, jpeg). After the Camera Client in the Runtime layer of the client's process has received the picture data, it will be sent to the Framework layer. The picture data is then shown on the screen after being sent by the Framework layer. In the Application layer, programmers have the option of saving the image data as an image file. Android's APIs allow developers to access the phone's hardware (such as the camera), wifi and networks, user data, and settings. There are restricted access points for certain of the APIs.

These APIs may be found in a framework's source codes or in systems libraries. When a protected API is called, the implementation codes of the API will make a request to the system server processes to validate the caller's authorization. Binder IPC is used to communicate with the Package Manager, thread of the system server process and complete the request. This thread has access to the API being called, the API being called the thread making the API request. So, this thread is useful for auditing Android APIs. The audit log also allows for intrusion detection. The media server process in Camera Service will initiate a request to the Package Manager thread whenever a request to take a photo is made to ensure that the caller has the necessary permission to do so. A review of the photo-taking API may be performed in the meantime.

RESULTS AND DISCUSSIONS

Assess the transplanting assault from the perspectives of real-world success, Antivirus (AV) detection rate, and Android Device Administration. Test the malicious software on real-world devices running a variety of Android versions to gauge the efficacy of the transplanting attack. Select eight manufacturers, 69 models, and seven releases of Android. Fourteen of the phones come from fellow researchers, while the other sixty-two come from the Baidu app test platform. Seven of the 62 phones are compatible with some of the 14 since they share the same model and Android version. So, the total number of various phones is 69. All phones used in the app testing process on the Baidu platform are actual phones and not simulators. Put the malicious software through its paces in the lab, capturing pictures, concealing the preview pane, and emailing them. Gave the rogue software permission to transfer the hidden photos to another device. Count the assault as successful if the target can access and see the attached images. The captured pictures may be stills from a video sequence, or they might be captured by several different cameras. There must be a common reference frame to which these pictures are mapped. This action is known as registration. After the aligned composite picture is ready, a specific area may be processed using the super-resolution technique. Creating a suitable forward picture model and performing precise alignment, also known as registration, are the two most important steps in achieving effective super-resolution. The spy camera dataset is shown in Table 1.

TABLE 1. Spy Camera System Dataset

Parameter	Value
Dataset Title	Android Spyware
Data Type	PCAP files, CSV files
Data class	Multivariate
Data source	Android-based spyware tools

If the malicious program can take a photo, and the wifi connection is active, the image will always be sent. Because we do not have physical access to the phones on the Baidu test platform, we simply assess the camera's functioning. Allow the malicious program to display the preview window for testing purposes. Therefore, checking if a phone's screenshot (produced by the test platforms) includes the preview windows can verify whether the image capture is successful or not. The transplanting attack is a complete success on phones running 4.1.1 from 5 different manufacturers. For version 4.1.2 phones from 5 suppliers, the success percentage of transplanting attacks is 75%. Also, track the vendor-by-vendor success rate. The total success rate is 46.38 percent, based on 69 phones. That implies this spy-on-user hack might affect almost half of all phones on the globe. The experiments on version 4.0.3 are quite confusing, so I examined this version's photo-capturing method and found there are no differences between this version and the previous 4. x versions. Baidu offers access to all 4.0.3 mobile devices. We are unable to determine the cause of failure for phones running 4.0.3 and 4.2.1 since Baidu's test platform does not provide us with the necessary failure information (adb log). However, this is a side advantage of personalization according to the failures caused by the phones utilized by lab mates, which would be studied as follows. Figure 2 shows the average search interest, and Figure 3 shows the vulnerability scale scores.

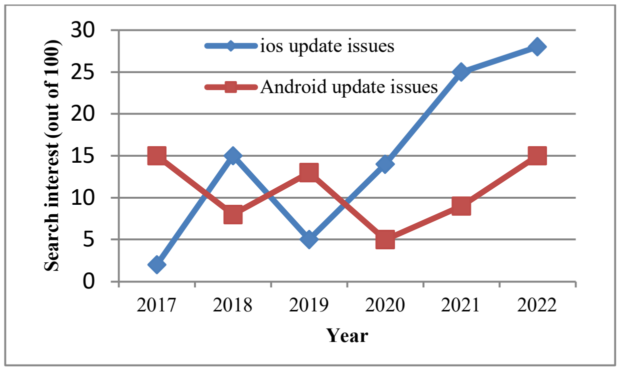

FIGURE 2. Average search interest

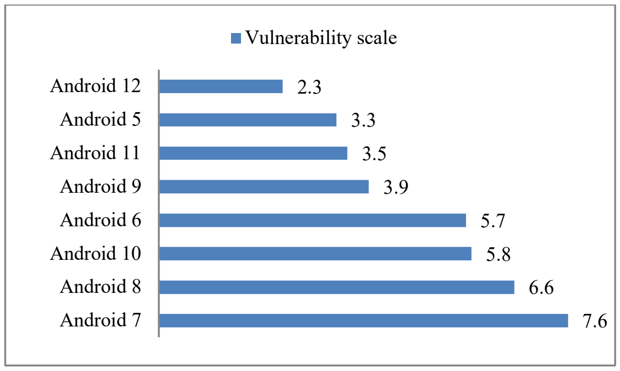

FIGURE 3. Vulnerability scale scores

Six of the phones failed to survive the transplant onslaught. Each phone has its own unique list of potential causes of failure. To determine what went wrong with each device, collect the adb log of a failed photo capture from both the stock Camera app and the malicious software. The results are summed up here. Since Google Galaxy Nexus is an AOSP (Android Open-Source Project)-compatible device, examine its photo-taking process source code. Discover that the media server process opens the /dev/proc user device, which is part of the drmrpc group, before it opens the camera device. Applications are unable to get the drmrpc group ID, in contrast to the camera group ID. Therefore, applications are unable to access the /dev/proc user device because they cannot join the drmrpc group. This means the strike is useless. For the Samsung SHV-E160S, the primary distinction between the Camera apps and the malicious software is whether the app is "unable to find matching camera info" for the provided camera ID.

CONCLUSIONS

This work presents the transplanting attack, an exploit that circumvents Android's API auditing to allow a malicious app to secretly collect (perhaps privacy-harming) images whenever it likes. The transplanting assault, when being implemented to accomplish the spies-on-users purpose, results in the stealthiest and unobserved image capturing. Test many forms of assault in a controlled environment. Of the 69 Smartphone's (from 8 different manufacturers) examined, 46.38 percent allowed the transplanting assault to proceed. The transplanting attack also reveals a minor flaw in the Android system's design or implementation. The latest research shows that although the issue has been addressed, most manufacturers still have not done so. Consequently, the malicious app is unable to access the camera, and an "operation not permitted" error of -1 is returned. Since we do not have access to the phone's source code, I can only speculate that the manufacturer may have changed the procedure for accessing the camera.

REFERENCES

[1]. M. N. Abdullah, and N. Baidilah, 2022, "CCMTV: Android parental spying apps utilizing child's phone camera and microphone," *In AIP Conference Proceedings,* 2617**(1)**, pp. 1-5.

[2]. S. Sami, S.R. Tan, B. Sun, and J. Han, 2021, "Lapd: Hidden spy camera detection using smartphone time-of-flight sensors," *In Proceedings of the 19th Conference on Embedded Networked Sensor Systems*, pp. 288-301.

[3]. E. Liu, S. Rao, S. Havron, G. Ho, S. Savage, G. M. Voelker, and D. McCoy, 2023, "No Privacy Among Spies: Assessing the Functionality and Insecurity of Consumer Android Spyware Apps," *Proceedings on Privacy Enhancing Technologies*, **1**, pp. 1-8.

[4]. D. Dao, M. Salman, and Y. Noh, 2021, "DeepDeSpy: A Deep Learning-Based Wireless Spy Camera Detection System," *IEEE Access*, **9**, pp. 145486-145497.

[5]. E. Chatzoglou, G. Kambourakis, and C. Smiliotopoulos, 2022, "Let the cat out of the bag: Popular android iot apps under security scrutiny," *Sensors*, **22 (2)**, pp. 1-41.

[6]. H. Ma, J. Tian, K. Qiu, D. Lo, D. Gao, D. Wu, C. Jia, and T. Baker, 2020, "Deep-learning–based app sensitive behavior surveillance for Android powered cyber–physical systems," *IEEE Transactions on Industrial Informatics*, **17(8)**, pp. 5840-5850.

[7]. J. Samhi, L. Li, T.F. Bissyandé, and J. Klein, 2022, "Difuzer: Uncovering suspicious hidden sensitive operations in android apps," *In Proceedings of the 44th International Conference on Software Engineering*, pp. 723-735.

[8]. J. Heo, S. Gil, Y. Jung, J. Kim, D. Kim, W. Park, Y. Kim, K.G. Shin, and CH. Lee, 2022, "Are There Wireless Hidden Cameras Spying on Me?" *In Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 714-726.

[9]. A.S. Falohun, B.O. Makinde, O.A. Adegbola, T.H. Akin-Olayemi, A.E. Adeyege, A.E. Adeosun, and B.D. Akande, 2021, "Design and construction of a smart door lock with an embedded SPY-camera," *Journal of Multidisplinary Engineering Science and Technology*, **8(7)**, pp. 14521-14528.

[10]. K. Kollnig, A. Shuba, R. Binns, M. Van Kleek, and N. Shadbolt, 2021, "Are iPhones really better for privacy? comparative study of iOS and Android apps," *proceedings on privacy Enhancing Technologies*, pp. 1-19.

[11]. K. Hariharan, R. R. Jain, A. Prasad, M. Sharma, P. Yadav, S.S. Poorna, and K. Anuraj, 2021, "A Comprehensive Study Toward Women Safety Using Machine Learning Along with Android App Development," *In Sustainable Communication Networks and Application: Proceedings, Springer Singapore*, pp. 321-330.

[12]. M. Suleman, T.R. Soomro, T.M. Ghazal, and M. Alshurideh, 2021, "Combating against potentially harmful mobile apps," *In the International Conference on Artificial Intelligence and Computer Vision. Cham: Springer International Publishing*, pp. 154-173.

[13]. M. Hatamian, S. Wairimu, N. Momen, and L. Fritsch, 2021, "A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps," *Empirical Software Engineering*, **26**, pp. 1-51.

[14]. F. Maasmi, M. Morcos, H. Al Hamadi, and E. Damiani, 2021, "Identifying applications' state via system calls activity: a pipeline approach," *In 28th IEEE International Conference on Electronics, Circuits, and Systems*, pp. 1-6.

[15]. S. Garg, and N. Baliyan, 2021, "Comparative analysis of Android and iOS from a security viewpoint," *Computer Science Review*, **40**, pp. 1-7.