Security Threats to Mobile Multimedia Applications: Camera-Based Attacks on Mobile Phones

Ramesh G
Associate Professor, Dept. of CSE
National Institute of Engineering, Mysuru

Pooja B R, Shilpa B,Sujatha B, Suma M
Computer Science and Engineering,
National Institute of Engineering, Mysuru

Abstract

Today many smartphone applications areusing wireless multimedia communications and hence in wireless multimedia communications mobile phone security has become an important aspect of security issue. As Android is the most popular operating system the researchers has been extensively studied Android security.However, few works have studied mobile phone multimedia security.Here in this article, we mainly focus on security issues that are related to mobile phone cameras and video based attacks. Here we specifically discover several new attacks that may occur based on the use of phone cameras.We implement the attacks on real smart phones, and demonstrate the effectiveness and feasibility of the attacks. Furthermore, we propose a lightweight defense scheme that can effectively detect these attacks.We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks.The results demonstrate the feasibility and effectiveness of these attacks.

Index Terms- *Cameras, Smart phones, Videos, Network security, Privacy, Real-Time systems Multimedia and Wireless Communication.*

1.INTRODUCTION

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of 2017, the Google Play store features over 3.5 million apps. Simultaneously, a number of Android security and privacy vulnerabilities have been exposed in the past several years by the researcher. Android applications run in a sandbox, an isolated area of the system that does not have access to the rest of the system's resources, unless access permissions are explicitly granted by the user when the application is installed, however this may not be possible for pre-installed apps. It is not possible, for example, to turn off the microphone access of the pre-installed camera app without disabling the camera completely. Even though Android system gives an opportunity for the users to check the permission request before installation of the android application. Many usersdont have knowledge of what all these permission requests stand for. Before installing an application, the Google Play store displays a list of the requirements an app needs to function. After reviewing these permissions, the user can choose to accept or refuse them, installing the application only if they accept, apps are no longer automatically granted all of their specified permissions at installation time. Meanwhile, an increasing number of apps specified to enhance security and protect user privacy have appeared in Android app markets. Most large anti-virus software companies have published their Android-version security apps, and tried to provide a security shield for smartphones by detecting and blocking malicious apps. In addition, there are data protection apps that provide users the capability to encrypt, decrypt, sign, and verify signatures for private texts, emails, and files. However, mobile malware and privacy leakage remain a big threat to mobile phone security and privacy.

While talking about privacy protection, most of the smartphone users pay attention to the safety of private files such as emails, SMS, location information, contact lists, calling histories. They may be surprised that the phone camera could become a traitor. for example, attackers could stealthily take pictures and record videos by using the phone camera when the screen is off, which means that the user is not using the phone and the camera device is idle. Nowadays, various types of camera-based applications have appeared in Android app markets (photography, barcode readers, social networking, etc.). Spy camera apps have also become more popular. As for Google Play Store , there are nearly 100's of spy camera apps available, which allow smart phone users to take pictures or record videos of other people without their permission. Attackers can even implement the spy cameras applications in a malicious apps such that the phone camera is automatically launched without notifying the device owner's, and the captures photos and videos and these can be sent out to these remote attackers. Even worse, according

to a survey on Android malware analysis [1], camera permission ranks 12th of the most commonly requested permissions among benign apps, while it is out of the top 20 in malware. The popularity of camera usage in benign apps and relatively less usage in
malware lower users' alertness to camera-based multimedia application attacks.

People nowadays, carry their phones everywhere and therefore a lots of private information can be seen by these phones. If the mobile phone is attacked by a spy camera app, this may cause a problem related to security and privacy information. For example, the phone camera may record a user's daily activities and conversations such as the user entering his secret password of email or any similar which should be private in the users presidency, and then send these recorded information out via the Internet or multimedia messaging service (MMS). A users phone camera could also provide some benefits if it is controlled well by the device owner. For example, when the owner is willing to check if the mobile has been used by some others without the permission of the owner, the phone camera could be used to record the face of an unauthorized user while accessing the device, which can also be used in finding out the lost mobile device of the owner.

Here , we first conduct a survey on the threats and benefits related to spy cameras. Then we present the basic attack model and t camera based attacks: the remote-controlled real-time monitoring attack, the passcode inference attack, Application-Oriented Attack, Screen Unlocking Attack and Video-Based Eye Tracking. We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks. The results demonstrate the feasibility and effectiveness of these attacks. Finally, we propose a lightweight defense scheme.

2. LITERATURE SURVEY

A number of research have been made for obtaining the information on smartphones using multimedia devices such as microphones and cameras. For example Soundcomber [2] is a stealthy Trojan that can sense the context of its audible surroundings to target and extract high valuedata such as credit card and PIN numbers. Stealthy audio recording is easier to realize since it does not need to hide the camera preview. Xu*et al.* [3] present a data collection technique using a video camera embedded in Windows phones. Their malware (installed as a Trojan) secretly records video and transmits data using either email or MMS. Windows phones offer a function, ShowWindow(hWnd, SW HIDE), which can hide an app window on the phone screen. However, it is much more complicated (no off-the-shelf function) to hide a camera preview window in an Android system. In this work, we are able to hide the whole camera app in Android. Moreover, we implement advanced forms of attacks such as remote-controlled and real-time monitoring attacks. We also utilize computer vision techniques to analyze recorded videos and infer passcodes from users' eye movements.

3. THREATS AND BENEFITS RELATED TO SPY CAMERAS

As mentioned earlier, the way of using spy camera and who is in control of it can mentioned as the main role of the spy camera played in the device. Here we discuss some threats and benefits of using spy camera.

3.1 LEAKING PRIVATE INFORMATION

Here a Spy camera is going to works as a thief, suppose if it steals the information from the phone which is private. First the malware attacks the victims mobile phone to gather the private information, example it appears to be a normal app which has all the permissions of camera and the Internet, it performs its work normally. Second, In the background, it runs as a secret service which silently click pictures and record videos of the user and stores the data in obscure directory that is seldom visited. This secret data is sent outside when the device is connected to the Wi-Fi.

3.2 WATCHDOG

Watchdog is another thing a spy camera can do. Nobody wants other people to use or check his/her phone without permission. A spy camera can stealthily take pictures of the phone user and deter those who use or check other people's phones.

3.3 ANTI-THIEF

On the other hand, a spy camera could play a completely different role if it is used properly. When a user loses his/her phone, the spy cameracould be launched via remote control and capture what the thief looks like as well as the surrounding environment. Then the pictures orvideos along with location information (GPS coordinates) can be sent back to the device owner so that the owner can pinpoint the thiefand get the phone back.

4. ARCHITECTURE OF THE CAMERA BASED ATTACK

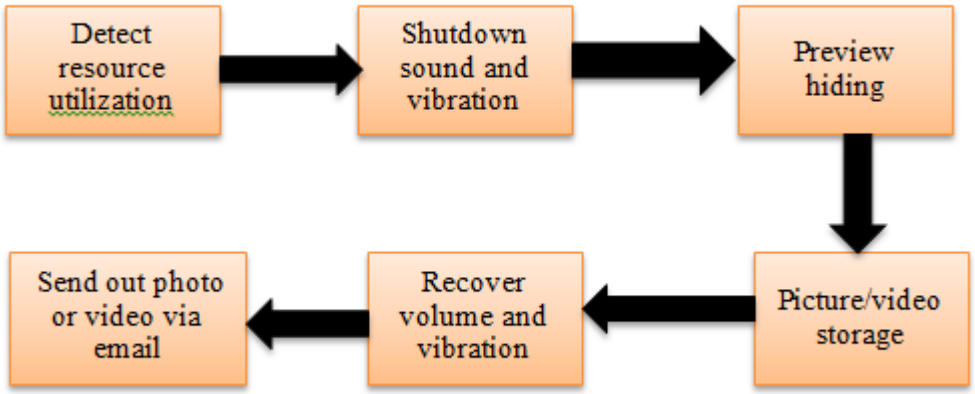

FIG 4.1.ARCHITECTURE

Here we want to discover the possible attacks that can occur based on spy camera. These attacks should appear normally without in notice of the user. The attacks before launching themselves should check the sound and vibration is off. The architecture of the camera based attack is shown in figure 4.1 which mainly contains six parts, here we going to discuss each part briefly.

Step1: To prevent the user from suspecting, the malware should consider the current CPU, memory usage, and battery status. Suppose if before launching the attack if the CPU and memory are already in use which could make performance of phone lower. Users tend to be concerned about the unsmooth experience, and check if any app or service is running in the background. Similar concern happens with energy consumption, especially when the phone's battery is low and is not being charged. A camera attack could drain the battery faster than the user's expectation and cause user suspicion about possible attacks. Hence, before launching the attack, malicious camera apps want to ensure that system resources are plentiful. For Android phones, memory usage could be obtained through the *getMemoryInfo()* function of *ActivityManager*, information related to CPU utilization is available from "*/proc/stat*," while current battery level and charging status can be obtained by registering a BroadcastReceiver with *ACTION_BATTERY_ CHANGED*.

Step2: After ensuring that the sufficient resources are available for launching camera based attacks, a malicious camera app can continue on the remaining actions. When the mobile is off, first the malicious app turn's off the phone's sound and vibration which can be achieved by setting the system sound *AudioManager.STREAM_SYSTEM*to 0 and the flag to *FLAG_REMOVE_SOUND_AND_VIBRATE*. The app can log the current volume level and vibration status, and resume the parameters after the attack.

Step3: The difficult task is to hide the camera preview. At the beginning, the layout containing the *SurfaceView*is inflated into a *view* via *LayoutInflater.inflate()*. Then the app can set the view parameters by changing the attributes of *WindowManager.LayoutParams*. Two importantattributes must be set: *TYPE_SYSTEM_OVERLAY*, which makes the preview window always stay on top of other apps; the other one is*FLAG_NOT_FOCUSABLE*, which disables the input focus of a spy camera app such that input values would be passed to the first focusable window underneath. This would turn the camera preview into a floating and not focusable layer. Then the app changes the size of preview *SurfaceView*) to the minimum pixel (1 pixel), which human eyes cannot notice. This cannot be set directly through *setPreviewSize()*. Instead, theapp needs to get the layout parameter of *SurfaceView*by using *SurfaceView.getLayout-Params()*. Notice that the type of*SurfaceView.getLayoutParams()* is *ViewGroup. LayoutParams*instead of the aforementioned *WindowManager.LayoutParams*. Finally, the app can add the hidden preview dynamically to the window by the *addView*function.

Step4: After setting up the layout, the attack could be launched as follows: initialize the *SurfaceHolder*, choose which camera (front or back) is used, and open the camera to take pictures or record videos. The photo/video data are supposed to be stored in disguises, including using confusing filenames and seldom visited directories. The app releases the camera after the above actions.

Step 5: After the camera attack finishes, the app sets the audio volume and vibration status back to its original values. This way, the device owner would not find any abnormality.
Step 6: The last step of the attack is to transmit the collected data to the outside. Since cellular network usage and MMS may cause extra fees, the best choice is to wait until free WiFi access is available. For example, it could use the *javax.mail*to send the data as an email attachment. Most email systems limit the maximum size of attachments, so the length of a video should have an upper bound specific to the email service.

5. ATTACK TYPES

5.1 THE REMOTE-CONTROLLED REAL-TIME MONITORING ATTACK

The basic camera attack can be further enhanced to more aggressive attacks. For example, the attacker can remotely control the spy camera app such that the time to launch and end the attack is under control of the attacker.

The simplest way to implement the remote control is by socket. After the malicious app is downloaded and installed on a victim's phone, it sends a "ready" message along with the IP address and port number to the attacker's server. Then the attacker can control the app with orders like "launch" and "stop" or specify a time schedule.

There are many Android apps that turn the phone into a security surveillance camera, such as Android Eye [6]. The spy camera can easily be extended to a stealthy real-time monitor based on the way an IP camera is built. NanoHttpd [7] is a lightweight HTTP server that can be installed on a phone. In our case, we can start an HTTP server at a given port which supports dynamic file serving such that the captured videos can be played online upon requests from a browser client.

Camera-based attacks can be detected when multiple apps request the camera device at the same time or if the camera is being used by another app. But this can easily be avoided by selecting the time to launch attack. The malicious camera app can periodically check the screen status and run the stealthy video recording only when the screen is off, which means that the user is not using the phone and the camera device is idle. The status of the phone screen can be obtained by registering two broadcast receivers, *ACTION_SCREEN_ON* and *ACTION_ SCREEN_OFF*.

5.2 THE VIDEO-BASED PASSCODE INFERENCE ATTACK

Since the virtual keyboard in a touch screen smartphone is much smaller than computer keyboards, the virtual keys are very close to each other.
When typing, users tend to keep a short distance to the screen, which allows the phone (front) camera to have a clear view of a user's eye movements. A user's eyes move along with the keys being touched, which means that tracking the eye movement could possibly tell what the user is entering. In this section, we discuss two types of camera attacks for inferring passcodes. We also discuss the computer vision techniques for eye tracking that can be utilized in the attacks.

5.2.1 THE APPLICATION-ORIENTED ATTACK

This type of attack aims to get some information about the apps in phone.Most apps (like Facebook) that require authentication contain letters, pattern and PIN, which needs a complete virtual keyboard. To passcode inference attack, the video must be captured during user authentication. An effective way is to poll the running task list and launch the attack as soon as the target app appears on top of the list. Specifically, using the *getRunningTasks()* function of *Activity-Manager*, we can get the name of the most recently launched app. Meanwhile, the detection service scans the running apps and resource utilization periodically. When attack conditions are met, it opens the camera and secretly takes videos of the user's face (especially the eyes) with a front-face camera for a time long enough to cover the entire authentication process.

To ensure the attack is efficient and effective we need to consider some factors. First, the detection service of a spy camera app must be launched beforehand, by either tempting the user to run the app or registering an *ACTION_BOOT_COMPLETED* receiver to launch when booting is finished. The *RECEIVE_BOOT_COMPLETED* permission is a commonly requested permission that would not be considered dangerous. Second, polling task lists frequently leads to extra consumption of energy resource. To improve the efficiency of scanning, the detection service is active only when a user is using the phone. As mentioned before, this can be determined by screen status. The detection service will cease when the screen is off and continue when the screen lights up again. Moreover, the scanning frequency should be set properly. In a phishing attack [8], a malicious app needs to poll the running task list every 5 ms to prevent the user from noticing that a new window (the fake app) has replaced the

original one. In our phone camera attack, the view is totally translucent to users, so that worry is unnecessary. However, we still need to keep
the frequency at around two scannings per second; otherwise, the attack may happen after the user starts entering the passcode (which makes
the attack unsuccessful).

5.2.2 THE SCREEN UNLOCKING ATTACK

The type of attack is occured when a user is entering a screen unlocking passcode. We categorize this scenario as a different attack model since it is unnecessary to hide the camera preview under this circumstance. To achieve privacy, an Android system would not show the user interface until a visitor proves to be the device owner by entering the correct credential. This accidentally provides a shield for spy camera attacks targeted at the screen unlocking process. Users never know that the camera is working, even though the camera preview is right beneath the unlocking interface.

We demonstrate the screen unlocking passcode inference attack. Its difference from the application-oriented attack is the condition tolaunch the attack and the time to stop. Intuitively, the attack should start as soon as the screen turns on and should end immediately as thescreen is unlocked. This can be achieved in two key steps:

• Registering a BroadcastReceiver to receive *ACTION_SCREEN_ON* when a user lights up the screen and begins the unlocking process

• Registering another BroadcastReceiver to receive *ACTION_USER_PRESENT* when a passcode is confirmed and the screen guard is gone

The second step guarantees that the camera servicewould stop recording and end itself immediatelywhen the user interface is switched on. Inaddition, the attack should consider the situationwith no screen locking passcode. To avoid beingexposed, the spy camera app should check *keyguardmanager*with the *isKeyguardLocked()* functionto make sure the screen is locked beforelaunching the attack.To simplify the screen unlocking process, theAndroid system provides alternative authenticationmethods in addition to the conventionalpassword: pattern and PIN. A pattern is a graphicalpasscode composed of a subset of a 3 × 3grid of dots that can be connected in an orderedsequence. There are some rules for the combinationof dots:

• The number of dots chosen must be at least4 and no more than 9.

• Each dot can be used only once.A PIN is a pure-digit passcode with lengthranging from 4 to 16, and repetition is allowed.Both alternatives are extensively used in screenunlocking of Android phones. The relativelylarger distance between adjacent keys effectivelyrelieves a user's eye fatigue problem. However,this also brings vulnerability tovideo-based passcode inference attacks sincethe larger scale of eye movement makes theattack easier.

5.2.3 VIDEO-BASED EYE TRACKING TECHNIQUES

In the eye tracking field, two types of imaging approaches are commonly used: visible and infrared spectrum imaging. Visible spectrum imaging passively utilizes the ambient light reflected from the eye, while infrared spectrum imaging is able to eliminate uncontrolled specular reflection with active infrared illumination. Although infrared spectrum eye tracking is more accurate, most smartphones today are not equipped with infrared cameras. Hence, we focus on visible spectrum eye tracking. For images captured by visible spectrum imaging, often the best feature to track is the contour between
iris and sclera known as the limbus [9]. Li et al. [9] propose the Starburst eye tracking algorithm, which can track the limbus of the eye. they can locate where the eye is looking in a realtime manner. However, Starburst requires calibration
by manually mapping between eyeposition coordinates and scene-image coordinates.

This can be performed only by the phone owner, which makes it infeasible in spy camera attacks. Aldrian [10] presents a method to extract fixed feature points from a given face in visible spectrum, which is based on the Viola Jones adaboosted algorithm for face detection. But it is able to track pupil movement without scene image and calibration, we adopt this eye tracking algorithm in our research to extract eyes from videos.

6. IMPLEMENTATION

We will discuss two attacks (remote-controlled real-time monitoring and passcode inference attacks). We have implemented the remote-controlled realtime monitoring and passcode inference attacks on real phones including the Nexus S4G (Android 4.1), Galaxy Nexus (Android 4.2), and Nexus 4 (Android4.3 with Security Enhanced support). For passcode inference attacks, a computer vision technique for eye tracking is used to process the captured video. The evaluation of the feasibility and effectiveness of the attacks is based on the experimental results.

6.1 IMPLEMENTATION

Both camera-based attacks are successfully implemented on Android phones equipped with a front-face camera. The spy camera apps

are completely translucent to phone users and work without causing any abnormal experiences. When WiFi access is enabled, the captured data is transmitted to the attacker via a local HTTP server or email. To test the stealthiness of the attacks, we install two popular antivirus apps: AVG antivirus and Norton Mobile Security. Neither of the two antivirus apps has reported warning during the entire video capturing and transmission process. This demonstrates their resistance to mobile antivirus tools.

6.2 FEATURE ANALYSIS OF THE PASSCODE INFERENCE ATTACK

An important feature that enhances the effectiveness of a passcode inference attack is that it can be launched repeatedly, which allows certain
passcodes to be "attacked" many times. In this way, an attacker could get a set of possible passcodes and keep launching attacks until the correct
one is found. The passcode inference attack depends on the victim's eye movement instead of analyzing videos containing the screen [4] or its reflection[5], which makes it harder to achieve high and stable one-time success rates. In addition, there are complex factors that may influence its performance, such as the distance between face and phone, lighting conditions, velocity of eye movements, pause time on each key, and head/device
shaking when typing. Among these experimental conditions, only the lighting condition can be kept constant during our experiments.

To test the effectiveness of the passcode inference attacks with different types of passcodes, we use the conventional password, pattern,
and PIN in our experiments. By comparing the rules of pattern and PIN, we find that pattern combination is actually a subset of PIN. In addition, the outlines of the two passcodes are similar (both are squares). Hence, we present their results and discuss their performance together. Another consideration for experiments is the length of pattern and PIN. In fact, people rarely use long PINs and complex patterns since they are hard to memorize and impractical for frequent authentications such as screen unlocking. This can be best illustrated by Apple iOS's four-digit PIN for screen unlocking. Hence, in our experiments we choose a four-digit pattern/PIN for testing.

7. CONCLUSION

In this article, we study camera-related vulnerabilities in Android phones for mobile multimedia applications. We discuss the roles a spy camera can play to attack or benefit phone users. We discover several advanced spy camera attacks, including the remote-controlled real-time monitoring attack and two types of passcode inference attacks. Meanwhile, we propose an effective defense scheme to secure a smartphone from all these spy camera attacks. In the future, we will investigate the feasibility of performing spy camera attacks on other mobile operating systems

8.REFERENCES

[1] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," *IEEE Symp. Security and Privacy 2012*, 2012, pp. 95–109.
[2] R. Schlegel *et al.*, "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones," *NDSS*,2011, pp. 17–33.
[3] N. Xu*et al.*, "Stealthy Video Capturer: A New Video-Based Spyware in 3g Smartphones," *Proc. 2nd ACMConf. Wireless Network Security*, 2009, pp. 69–78.
[4] F. Maggi, *et al.*,"A Fast Eavesdropping Attack against Touchscreens," *7th Int'l. Conf.Info. Assurance andSecurity*, 2011, pp. 320–25.
[5] R. Ragurame*t al.*, "ispy: Automatic Reconstruction of Typed Input from Compromising Reflections," Proc. 18th ACM Conf. Computer and Commun. Security,
2011, pp. 527–36.
[6] "Android-eye," https://github.com/Teaonly/android-eye, 2012.
[7] "Nanohttpd," https://github.com/NanoHttpd/nanohttpd.
[8] A. P. Felt and D. Wagner, "Phishing on Mobile Devices," Proc. WEB 2.0 Security and Privacy, 2011.
[9] D. Li, D. Winfield, and D. Parkhurst, "Starburst: A Hybrid Algorithm for Video-Based Eye Tracking Combining Feature-Based and Model-Based Approaches,"
IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition — Workshops, 2005, p. 79.
[10] P. Aldrian, "Fast Eyetracking," http://www.mathworks.com/matlabcentral /fileexchange/25056-fast-eyetracking, 2009.