IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CYBER-THREATS AND COUNTERMEASURES IN THE HEALTHCARE SECTOR

Received February 19, 2018, accepted March 26, 2018, date of publication April 12, 2018, date of current version May 2, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2826225

Counter Measuring Conceivable Security Threats on Smart Healthcare Devices

SYEDA MARIAM MUZAMMAL[1], MUNAM ALI SHAH[1], HASAN ALI KHATTAK[1], SOHAIL JABBAR[2], GHUFRAN AHMED[1], SHEHZAD KHALID[3], SHAHID HUSSAIN[1], AND KIJUN HAN[4]

[1]Computer Science Department, COMSATS Institute of Information Technology, Islamabad 45550, Pakistan
[2]Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan
[3]Department of Computer Engineering, Bahria University, Islamabad 75260, Pakistan
[4]Department of Computer Engineering, Kyungpook National University, 37224 Daegu, South Korea

Corresponding author: Kijun Han (kjhan@knu.ac.kr)

This work was supported in part by the BK21 Plus Project (SW Human Resource Development Program for Supporting Smart Life) through the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, South Korea, under Grant 21A20131600005, in part by the Basic Science Research Program through the National Research Foundation of Korea, Ministry of Education, under Grant 2016R1D1A1B03933566, and in part by the Institute for Information & communications Technology Promotion through the Korea Government (MSIP) under Grant 2017-0-00770.

ABSTRACT Smart devices, the carriers of a huge amount of private, sensitive and confidential data are pervasive in today's world with innovative and enhanced functionalities. Smartphones have brought tremendous change in people's lives with the launch of a new platform of communication and an ease of access to a wide range of applications. Due to the swift increase in the users of Android smartphones and the increasing demands based on advanced ease and features, developers are working hard to achieve the needful. Easy access to certain features and applications gave rise to the powerfulness and an efficacy of various threats, risks and vulnerabilities that can victimize users' private data residing in smartphone paradigm. With the developments and enhancements in malware, for Android-based smartphones, attacks continue to occur. In this paper, we investigate one of the possibly most destructive attacks for Android, that is, screenshot attack. We have developed ''ScreenStealer'' application and explored the vulnerabilities which make Android more inclined to risks and threats. Furthermore, we evaluated capture ratio of screenshots, resources consumption and execution time to determine effectiveness, efficiency and stealthiness of such a malicious application.

INDEX TERMS Internet of Things, smart devices, mobile security, information processing, information exchange, electronic healthcare.

I. INTRODUCTION

The use of smartphones has increased tremendously with the enhancement of advanced and innovative features. The adaptation of this technology and innovative services provided by network providers and manufacturers is also growing with the same pace. In the same way, private data of users, which is kept in smartphones to enjoy this mobile revolution, is under great risks and threats. As a result of the developments made by the Android application developers, about 2 million applications have been uploaded to Android Market till February 2016 with billions of reported downloads [1].

Android security mechanisms are overburdened by the developments and downloads of millions of applications. Every day, various malicious applications and security and privacy risks are detected and removed by Google [2]. Additionally, the security experts are working on anti-viruses, anti-malwares like Malwarebytes [3], and several other mechanisms to avoid impish happenings by malware developers and mischievous hackers.

The number of attacks and attack vectors is growing exponentially with an increase in the development of malicious applications in smartphones. This quickly evolving era of smartphones technology as an easy target for malware is still unexplored inside out [4]. Hackers can use different tricks and techniques to extract sensitive user data on smart devices. One of the traditional techniques of cybercrimes is via email phishing scams. A hacker may send a malware link to user via SMS (Short Message Service), requesting to click the link to

get voicemail and the user ends up in infecting his/her smart device. For hackers and cyber criminals, injecting malware into user's smartphone in order to expose private data is not a big deal with the use of advanced techniques, tools and technologies. Consequently, with concerns of major privacy and security issues and to ensure data security in smartphones, the techniques and patterns need to be explored effectively [5]. But due to innovative emergence of smart technologies, this has become a challenging task for researchers.

With increase in the number of Android users to millions and their evolving demands, the Android application developers are struggling to enhance and advance applications to further facilitate lives of people around the globe. To accomplish the targets, developers are hunting to access critical system resources which they are not allowed to obtain directly by Android. One of the major protrusive direction is using ADB (Android Debug Bridge) to escalate privileges. To provide users with smart and avantgarde features in smartphones, a number of applications on Google Play Store are utilizing ADB proxy. This paper analyses the ADB competences and how ADB and Internet Access combined together permit applications to interpret delicate data of an Android smartphone device by screens capturing of other applications on smartphones. Moreover, a malicious Android application, 'ScreenStealer', has been developed and its effects on Android-based smartphones are studied [6].

A. ARCHITECTURE OF ANDROID

The architecture of Android is grounded on Linux Kernel. The primary sections of Android include a working framework, middleware and key applications. Adaptable working framework of Android is depicted in the light of Linux bit. The escalating fervor from the business arises from two focused lookouts: firstly, the open source nature and secondly the engineering model. Mainly, Android has been intended for touch-screen devices like smartphones, ipads, tablets and other alike devices. The open-source nature of Android has invigorated developers and crusaders to make use of the open-source code as the base of the up-to-the-minute applications and acquaint with forward-thinking features for Android users and convince other operating system's users to adapt Android [7].

B. SECURITY IN ANDROID

With the rapidly expanding number of android devices, more newfangled and creatively proceeded, subsequently tougher to elude, attacks are being launched on smart devices. A major part of such assaults is plausible because of the negligence of the end-users and clients. Without opposing the consequences of being under attack, the clients and end-users are struggling more to make the best out of their smart devices.

Android applications run in a secluded sandbox of the system, from which they are unable to attain any system resources, except overtly request for exact permissions and these authorizations must be approved by the user to the application at installation time. On Google Play Store, application developers declare the permissions that the specific application needs to accomplish its work on Android devices. For proper processing of installation, a list of permissions is approved to be granted by consent of user, when an application is clicked to be installed. If user grants the necessary permissions, application is successfully installed, otherwise installation process cannot be initiated if user disagrees.

Additionally, Android makes use of the signature mechanism for application. The distribution of Android files is done in the form of apk. Moreover, both for uploading and installation of an application, validation of the corresponding public key is done as part of Play Store platform [9].

C. ADB - ANDROID DEBUG BRIDGE

ADB is an operative way for application developers to get determined privileges for the Android smart device. It is a multiuse command line tool [9] which is supported with the Android Development Kit [10] and allows the developer to link with the associated Android device. Android application developers can access user data via ADB. It authorizes a developer or a legal agent to communicate with and govern an Android device through USB connection. This means can be used for performing different tasks like application installation and uninstallation, downloading and uploading a file, running of development console, application executions and debugging and many other.

Primarily, ADB client has been just considered to be an application that can be used by developer to control emulator or a device. Nevertheless, its components have now been packaged into a library, that can allow its usage in a Java application [11]. Therefore, a number of opportunities can be attained by developer from ADB to control Android smart devices. There are three key components of ADB: (i) client, (ii) server, and (iii) daemon.

Android developers are emerging with new approaches to work with new safety protections and privacy issues, regardless of the already firm and emerging security mechanisms of Google Play Store and Android. The reason for this enhancement is that Google's API's and services are not enough to fulfil the demands and provide extraordinary capabilities other than the routine and usual tasks like USB tethering and backup etc. Rooting of the device is another approach to accomplish the required tasks.

According to a survey, over, 27.44% users are rooting their phones to take full control over their Android device [12]. But usually rooting is avoided by users and developers due to expiration of warranty as well as rooting can permanently damage the device functionalities.

A feasible and usual substitute to escalate privileges on an Android device is via use of workaround based on ADB. The android device has to be connected to the PC for launching ADB and service instantiation, in order to apply the technique of 'no root'. In this way, the application is able to attain system resources that are not provided in normal scenarios. This resources accessibility is sustained till disconnection of device and computer or the devices is rebooted.

There are a number of applications on Play Store that are utilizing perilous resources and methods to achieve system backup, USB tethering, system sync, and other utilities like taking screenshots. Moreover, this mechanism of privilege escalation is overall considered as legal and authenticated way. Utilization of this technique by several applications over Play Store demonstrates that the authorities are unaware of the security and privacy implications of privileges escalation through ADB workaround.

D. THE LOCAL SOCKET CHANNEL AND ADB SCREENSHOTS

The useful functionality of taking screenshot and capturing the on-screen content as a picture and storing it for later use is demanding among users as part of smart devices. Till Android 4.0 was introduced, this screenshot feature was not introduced in smart devices. For the successive Android versions, screenshots were easy to be taken by pressing Volume up/down and power/menu button together, that is through hardware. Additionally, there was no provided technique or API to enable taking on-the-go screenshots programmatically. The developers and users are making use of this 'no root' alternative, the ADB workaround, rather than damaging their devices with rooting mechanisms.

Since Inter-process communication (IPC) is not supported by Android, between an app and the backend native server, hence a local-socket-channel [13] is used between these two parties to communicate with each other. Such a channel has not been considered while developing Android security design [14], hence opens new ways for malicious attacks and security hazards. With the implementation of this technique, any app with the Internet permission, can connect to the local server in order to take screenshots, and all applications other than screenshot apps, that work using ADB workaround, can have the similar types of privacy exposures. Since this approach is completely legitimate and easy to use, hence lack of thorough understanding of the capabilities and continued utilization of ADB workaround by developers may lead to more harmful privacy and security attacks.

This research work is based on exploration of risks and threats to users' sensitive data on smartphones and how some of the malicious applications such as screenshot attack can affect system resources and users' privacy. The study involves the investigation of ADB (Android Debug Bridge) capabilities and how Internet access and ADB together allow applications to expose sensitive data of an Android smartphone device. Furthermore, a malicious Android application, ''ScreenStealer'', has been developed and its effects on Android-based smartphones are studied. ScreenStealer has been developed to understand the risks and threats of an app that monitors the screen and front-end user activities, then takes screenshots stealthily and sneakily and send to the adversary.

The research contributions include the investigation of the security risks of the ADB workaround and the local-socket channel, new techniques for targeted, stealthy and real-time extraction of user data via screenshots, and implementation and evaluation of a malicious screenshot application. Though there are a number of different attack vectors for Android smartphones, the scope of this paper is primarily based on screenshot attacks, with investigation of innovative vulnerabilities which are caused due to the Android security loopholes. This also includes the study of the results after a successful launch of malicious application on Google Play Store.

II. RELATED WORK

The escalating development in smartphones has led to the advancements in pioneering features offered by different vendors depending on customization of operating systems. This rapid pace has resulted in manufacturing and releasing of low-cost smartphones that are now affordable for common users. The functionalities in these low-cost vendors' smartphones is more or less the same but without a thorough and comparably less secure customization of smartphone Operating System. One such examples is of Android phones. There are a number of Android vendors including Samsung, LG, Motorola, HTC, Huawei and a growing list of companies that are deploying Android on their devices.

Since, Android is an open-source operating system, so there is more Android applications development as compared to any other operating system, hence same is with the applications download and applications piracy; this makes Android, the most vulnerable operating system.

In today's technological era, smart device of a user is somewhat complete sensitive data carrier of an individual, including messages, emails, contacts, important dates, call logs, chats, social media activeness, pictures, documents, whereabouts, bank accounts and other related information. The availability of all this information in a handy smart device invoke roguish hackers to proceed with conceivable attacks on smartphones risking the sensitive, personal and financial information of users.

A. ANDROID PERMISSION MECHANISMS AND APPLICATIONS

Smartphone technologies advancements have provided users with tools and techniques in order to customize and develop innovative applications in alliance with evolving demands. Majority of the smartphone users are not conscious of how the downloaded applications are using private data legally and with their consent. Users incline to ignore the permissions granting and privacy policies at time of applications downloading from App store risking the security and privacy parameters [15], [16]. The downloaded applications can Such applications deliberately or accidentally utilize and distribute sensitive information including photos, location, contacts, identity info etc. and can lead to various vulnerabilities through stalking and theft. Studies have shown that up to 70% of applications on Play Store request for permissions that are irrelevant to the main functions of it [6], [17], resulting in leakage of private information and resources consumption.

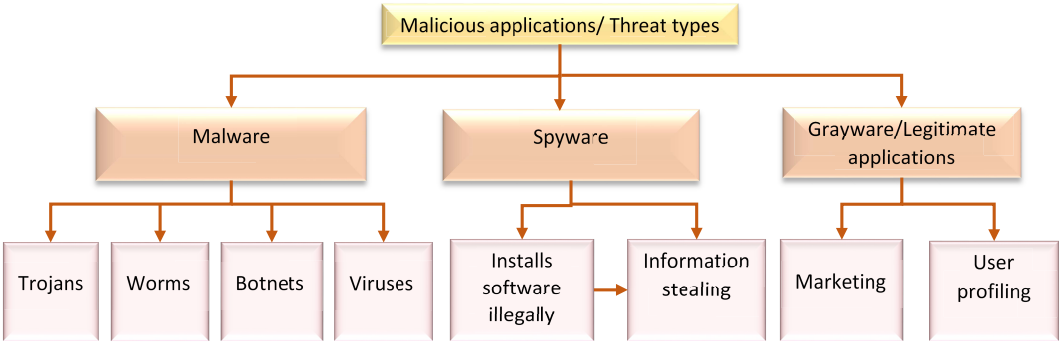

FIGURE 1. Categorization of threat types.

Security alert messages are also neglected by uses while installing applications and granting permissions [18], [19]. The revelation of location data is a common case of such form of data leakage, which is being used by various entertaining social connectivity applications. Many of the social media applications diffuse location data along with photos, videos, messages and status uploads. Majority of the applications allow users to control privacy settings for exposure of location and other data, but a number of users remain unaware or continuously ignore the unintended revelation of whereabouts and personal information.

Likewise, attackers and unethical are developing malware that can be detrimental in several ways, for example, fluctuating or revealing user personal information on smartphone, implementing any random code, exploit user by some pricey jobs like sending SMS or MMS (Multimedia Messaging Service) or perform an annoying call, connect to a network, resources consumption and many more.

B. TYPES OF ATTACK VECTORS

People use their smart devices for communication, storing personal files, images and videos, maintaining a record of important events and dates to remember, browsing over the Internet, entertainment like gaming and music, and many use it for sensitive purposes like banking transactions and online shopping through credit cards. In this way, mobile devices store a lot of private and business information of the users. Because of these and many other attractive uses in routine activities, smartphones have a number of attack vectors including the lost or theft, malware/malicious applications and cyber-crimes. Whereas, users keep using their smart devices, neglecting the security and without realizing the serious security impacts.

According to a survey [20], the data in mobile devices specially in healthcare domain is vulnerable, firstly, due to lost or theft and secondly, due to malicious applications downloaded to the mobile devices whether they are standalone applications or accompany an attractive entertaining app. A simple careless act of user, like downloading an unnecessary app, replying to a malfunctioned SMS, opening an email containing malware etc., can possibly and effectively cause destruction to user's data in smart devices. Some researchers have categorized malicious applications and/or threat types into three categories: Malware, Spyware, and Grayware [21]. This categorization of threat types is shown in Figure 1.

Spyware in Smartphones: Usually the spyware is previously installed in the smartphone or it sneaks in when downloading an application containing malicious phenomenon from Play Store. It is not hard and fast that a spyware targets the private information on smart device, it can also make distribution of sensitive information over unsecure channels, possibly for advertisements or marketing. Spyware can act as a threat to user's personal information with intentional or unintentional installation while stalking user's activities. A research study [22] indicates that there are a number of applications over Play Store that asks user for unnecessary permissions to access private data on device. Such applications can use this personal data for several malicious purposes to attack privacy or to annoy in ways like placing unwanted calls or sending messages through mobile network or over the internet, without users being aware of it.

Similarly, smartphones can be used as a tool for observing an individual's behavior and activities by using certain smart features like GPS, microphones, network connectivity, front and back camera and accelerometer. Attacker can install applications like this deliberately, with provision of access to the user device for a short period of time. This can also be done by fooling user for a fake application that is downloaded from an authenticated source, leading to help attacker to monitor sensitive data about whereabouts, activities and location. Some authentic applications and software can also be stealthily constituted to an eavesdropping implementation.

C. SECURITY AT APPLICATION-LEVEL

The applications can be made secure by developers at application-level. Smartphone integrity and confidentiality can be ensured by application developers by adopting cryptography [23]. This can be implemented either through applications or through APIs. In app-store, there are a number of applications that use cryptographic technologies like Crypto, Cipher tools, Encrypt Editor and other similar applications.

Such applications provide data encryption for user confidential data. Some cryptographic applications also provide hash functions for data integrity; thus, users can secure their smartphones using this type of applications. Application developers are also provided with APIs and libraries that provide security features to be used in implementation of applications.

Similarly, access control has already been provided by Google that enhance user authentication mechanism of the smart devices and limits the access to processes and resources [24].

Millions of users download several applications from Google store every day. Most of the security attacks occur on application layer. Although Android keeps checks on malicious applications via malware scanner that blocks potentially harmful applications. Also, Google Play Store claims to provide security by reviewing the application before it is published on Google Play [25]. Android also provides secure application download by its App Verification feature, which the user can turn on from settings of the device. But this feature can be easily turned off by developers of malicious applications, so this cannot be a reliable way for secure app downloading.

The easiest way to make secure a smartphone is by installing an anti-malware or anti-virus application that are easily available on app store. With the increase in the malware attacks, there is an enormous development of antiviruses, antimalware and spam filter applications as well, for example, Malwarebytes, Eset, Lookout, AVG, Wolfguard and many more. Most of the antimalware applications have their own certain limitations like in terms of cost, timely alerts, automatic updates and lack of advanced and updated features, that hinders the provision of satisfactory protection to users' private stuff in smart devices [26].

D. SCREENMILKER AND ADB

Chia-Chi et al. combined the local-socket channel with ADB workaround to develop Screenmilker that describes the possibility of milking screen for secrets. [14]. With the popularity of screenshot applications over Play Store, millions of such applications are downloaded by users; and ADB proxy is being used by these applications to do the needful. Other than screenshot applications, system backup and sync as well as USB tethering applications also use the ADB workaround to accomplish the tasks.

The purpose of screenmilker is to excerpt data by taking the screenshots from the device instead of just saving the images. It emphasizes on extraction of login and passwords via observation of the keystrokes when password is entered in the target app. A screenshot when taken is processed, guessing and extraction of password is performed on the Android device locally. The capturing of screenshot, image storage and processing can make the user conscious about some suspicious activities since it has been taking screenshots continuously and extracting passwords using resources consuming process of OCR (Optical Character Recognition).

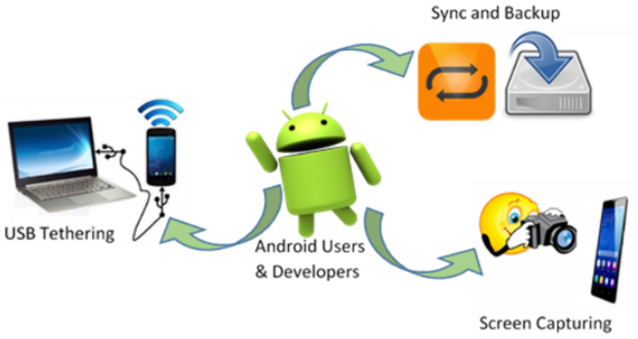

FIGURE 2. Users and developers for apps based on ADB capabilities.

E. GENERALITY OF PROBLEM

As already discussed that Android operating system is open source, giving access to all the enthusiastic developers, multiple opportunities to propel in smartphone application development. The greatest channel for Android applications distribution, Google Play Store, however, has very limited control over the applications being published for users [27]. Hence, there is always possibility of low-quality or malicious applications to go online and penetrate to user device.

In this research work, primarily the focus is on screenshot applications. However, the core of such apps, ADB proxy, a feasible and usual substitute to escalate privileges on an Android device is via use of workaround based on ADB (described in section 1.3). ADB proxy and the local-socket channel, are more generic foundations for various privilege-escalated applications. Android security design does not provide applications direct access to system resources, reading or writing other than its own directory and installing or uninstalling other applications; whereas developers as well as users demand for such applications. There are a number of applications on Google Play Store, like backup and sync applications and USB tethering (Figure 2) expose the ADB capabilities to unauthorized third-party applications that can cause exposure of sensitive user data, also bypassing the firewall protection. The issues regarding this need to be further investigated in detail. The backup applications send request to the ADB proxy using local-socket channel that enables the phone data backup to local storage, SD Card or Google drive. In the similar way, ADB proxy can be leveraged by an unauthorized application with the Internet permission to store and then access user data sneakily. The USB tethering applications work by utilizing an ADB proxy to forward TCP (Transmission Control Protocol) packets to the Internet. This may risk user data for a malicious app to sneak out, without being inspected by firewall applications like No Root Firewall [28] or DroidWall [29]. This indicates that regulation of the use of ADB based workaround and the local socket channel needs to be done in a thoughtful way in order to minimize risks to private and sensitive user data in future.

F. ANDROID PROTECTION MECHANISM

To prevent malicious applications to damage smartphone or misuse user data, Android implements two types of

mechanisms, one is sandboxing and other is permissions [30]. Sandboxing is based on the isolation of the application, that is, each application executes in its own specific instance of DVM (Dalvik Virtual Machine) and each DVM is taken as a unique UNIX userID by the Linux Kernel. This restricts individual applications from interference with each other.

Permission mechanism is based on the protection and control access of the system resources and operations by unauthorized applications. At the time of installation, user is presented with the set of permissions that the installing applications will seek. It is user's choice to accept the permissions or deny. This approach acquired too much criticism [31] as it totally relies on user knowledge and expertise [18]. The main problem with this mechanism is that the user has to accept all the permissions offered by the application in order to proceed installation, there is nothing like accepting a subset of permissions. Furthermore, due to the presence of a large number of permissions, normally users do not understand all the permissions and end up accepting all of them to gain new and advanced features, without considering about the critical permissions granted and potential security threats.

III. OVERVIEW AND IMPLEMENTATION

A. OVERVIEW

The development and deployment of a malicious application, for purpose of stealing user sensitive data sneakily, is a very difficult task practically, even with the use of ADB privilege escalation proficiencies. There will always be some boundaries and restrictions owned by such applications. There can be a number of possibilities for the utilization of Android resources to develop and design an application thoughtfully that can maliciously work with sensitive information over smart device imperceptibly.

''ScreenStealer'' has been developed to demonstrate the development phases of malicious applications conceptually, along with the hypotheses that it is possible to deceive user via authentic application requiring ordinary permissions from user. Once permissions are granted by user and the application is installed, ScreenStealer can monitor user activities, capture screenshots and send to the owner/server, in the same way that it is instructed to do. Eventually, the focus of developing ScreenStealer is to create awareness among the users and developers to consider the application permissions and grant cautiously while usage and development.

When a screenshot is captured, the most important aspect to consider is the size of the image, that will be stored in the SD card. While conspiring image size, quality and resolution of image is also adjusted to reduce the consumption of device memory, and also the image is send directly to the server. User's suspiciousness to the screenshot taking scenario is reduced by setting the time interval and image size of this malicious application rather than randomly taking images. Additionally, ScreenStealer stores the image in encrypted form inside a hidden folder on user's device for a certain time-slot. When the screenshot is successfully uploaded to the server, a new screenshot is taken to replace the previous one immediately. Therefore, to minimize user's consciousness, only one image is stored over device at a time. For optimization, a quite justifiable interval of 20 seconds is kept for taking screenshots for tracking user's information. This interval can be changed by the adversary in accordance with the requirements.

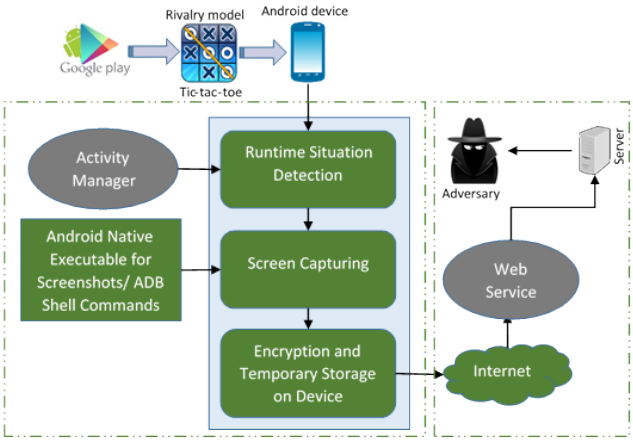

FIGURE 3. The architecture of ScreenStealer.

B. ARCHITECTURE

The architectural aspects of ScreenStealer application are shown in Figure 3. Application mainly consists three components i.e application monitoring with situation discernment, taking screenshot of important user data for real time data collection and the encryption, saving and sending it to server. Moreover, this application is aimed to be used in two different ways, one is for rooted and non-rooted devices while other only works for rooted devices. The developed version is way more authentic and strong than the one which also covers non-rooted android devices, because it doesn't need PC connection for the activation of related android services.

The minimum use of resources, like memory, network, and CPU has been kept in mind to cater stealthiness and develop ScreenStealer intentionally for stealing user's sensitive data from various specified applications. 'Runtime Situation Detection' refers to that the ScreenStealer keeps stalking front-end of user activities and keeps an eye for a number of sensitive application together with the banking, native and social media applications. The screenshot is taken when one of the specified applications is opened on device's screen. This surveillance is carried out by the situation at runtime and application recognition module of the ScreenStealer.

C. WORKFLOW

ScreenStealer has been embedded in an adversary model (described in next section), and uploaded on Play Store, which can easily fool a user to download and install the malicious application. Once the application is installed and run for the first time on the device, it starts monitoring the

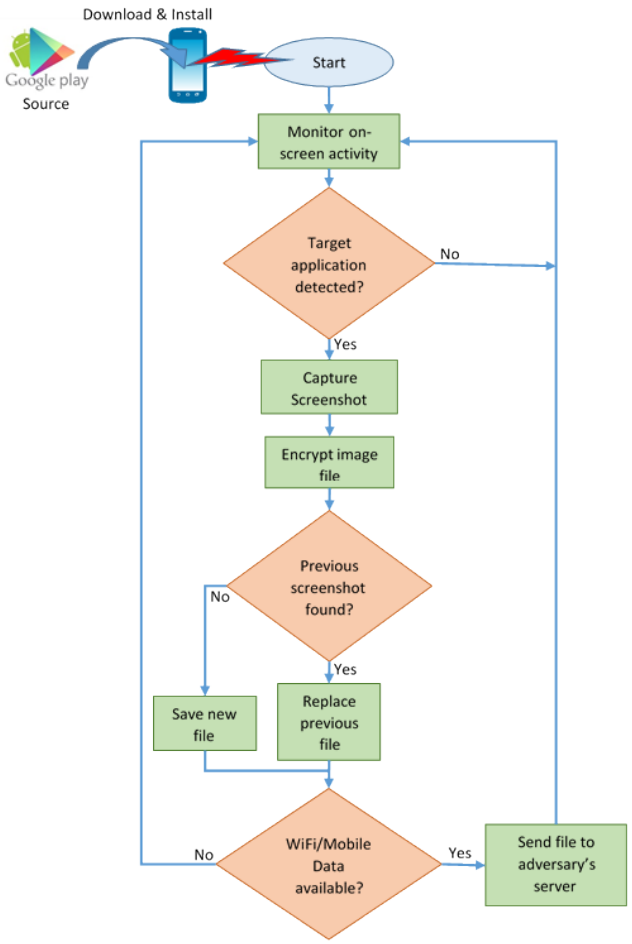

FIGURE 4. Workflow of ScreenStealer.

device screens and front-end activity of user with 5 seconds interval. While it monitors the screen, if a target application is identified at the front-end, it processes the screenshot capturing process. After taking the screenshot firstly, ScreenStealer looks whether any of the screenshots previously taken is already saved in memory, if it is so, it overwrites the previous screen image file; if not, it saves the new screenshot. It then checks whether the Wi-Fi or mobile data is available and connected with the device. If yes, then ScreenStealer transfers the already saved image (the latest screenshot taken) to the attacker via WiFi or mobile data connectivity with the device. In case, there is no network found, then it goes back again to app- monitoring module. If no target app is active at front-end, it keeps going back to the app-monitoring phase, hence keep running at back-end and continue its working cycle. The workflow of ScreenStealer is shown in Figure 4.

D. RIVALRY MODEL

A rivalry model in form of 'Tic-tac-toe' game has been developed to demonstrate the success of screenshot attacks and speculation for how malicious applications work via legitimate resources. Tic-tac-toe game is developed for this research work, to facade ScreenStealer by delude users to

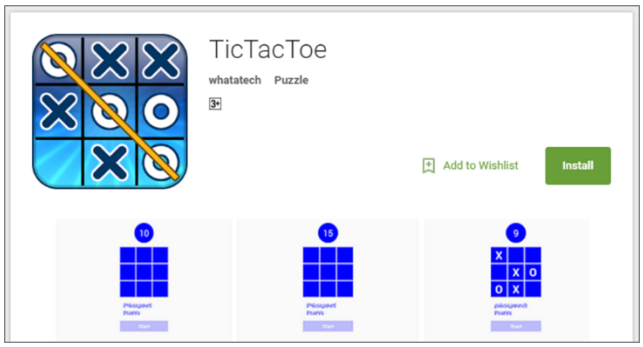

FIGURE 5. ScreenStealer on play store uploaded as 'Tic-Tac-Toe'.

install it as authentic real entertaining application that ask for basic and normal approbations of Internet and storage for installation in devices. Approval for internet is not doubtful, that is, most of the applications on play store ask for internet for various purposes like advertisements and upgradations. Believing it that device will always be connected to internet or mobile data, with user nit paying much attention to the usage of resources and does not use any softwares to obtain any information regarding certain application consistently.

To measure the effectiveness of ScreenStealer application in real-time, it has been uploaded to the Google Play Store (Figure 5) [32]. This malicious application is working as malware under the concealment of a Tic-Tac-Toe game. Furthermore, it was discovered that, the verification and security mechanism of Google Play Store does not cater the validity of data access permissions required by an application already uploaded or updated. This inability to resist unnecessary permissions allocated to an application leads to the freedom of application to run malicious code in cover of a rivalry.

E. SCREENSHOT CAPABILITY OF ADB, DETECTION OF INFORMATION EXPOSURE AND SCREEN CAPTURING

Various strategies are used by ScreenStealer to detect the accessibility of ADB-based screenshot capability. It is quite easy to remodel things at OS level for rooted devices, that is why ScreenStealer uses ADB shell commands for capturing screenshots programmatically. That creates a method using root prerogatives of the device. ScreenStealer has been made possible on non-rooted devices as well by using ASL (Android Screenshot Library), an open source library provided on Google code blog [6], [33]. Taking screenshots programmatically on Android devices without any root prerogatives is been made possible by ASL. It takes ADB to run a backend android related service, that works till the device is rebooted. The privileges provided by rooting the device are not necessary for ASL to capture screenshots programmatically by utilizing the native services running at backend of Android. This Android native service starts by ADB always when the device is rebooted.

F. IMPLEMENTATION

An access control mechanism for android is via specific permissions that applications request. The permissions need

TABLE 1. List of permissions employed by ScreenStealer.

Permission	Purpose
INTERNET	Send screenshots to adversary
WRITE_EXTERNAL_STORAGE	Temporary save the screenshots
GET_TASKS	Get the tasks running on phone
GET_TOP_ACTIVITY_INFO	Get the activity, active in the foreground
RECEIVE_BOOT_COMPLETED	To auto-start ScreenStealer on boot

to be defined deliberately in *AndroidManifest.xml* file in order to access specific resources. *AndroidManifest.xml* file is necessary for every Android project as part of source files, and declare essential information about application for build tools, OS and Play Store. In this file, the user is asked to allow access to resources via permissions, before installation of application. Table 1 summarizes the list of permissions that ScreenStealer seeks to obtain. ScreenStealer makes use of *DeviceAdminReceiver* and *BroadcastReceiver* to attain the essential privileges and permissions to empowering screenshot capture and ScreenStealer to work proficiently. The malware has also been made incapable of 'Force Stop'ed or 'Uninstall'ed by the user. To check whether any of the specified application is running in the background, a backend service has been used to keep stalking for the activities at front-end. Likewise, the screenshots are captured via similar backend service, which are sent to the attacker's server through a web service.

G. REAL-TIME DATA STEALING

With the help of specific services and API's provided by Android, ScreenStealer determines the application currently running at front-end. API *getActivePackages* is used to detect the current application at foreground, making use of the target application declaration beforehand. ScreenStealer screen capturing module is initiated immediately after the target application is detected, and starts capturing screenshots after defined intervals till the life of target application on foreground. The detection module for target application keeps its work going on at backend for stalking user activities. ScreenStealer has been hard-coded with list of social media and banking applications to spot and capture required screenshots.

When ScreenStealer successfully captures a screenshot, it stores it in encrypted form at SD Card or device storage temporarily. The image is not only stored in a hardly locatable manner as well as it is incapable of being opened and viewed. The image that is stored for the time being makes use of web service and Internet and sent to the owner's server. A view of the screenshots on our server from Android devices is captured in Figure 6 below.

H. MALICIOUS ENHANCEMENT OF ScreenStealer

Other than the already implemented and above explained scenario of the ScreenStealer attack, this application can be enhanced maliciously in a number of possible ways, with minor changes and using only a few resources. Along with the screenshots that are being sent to screenshots, the current location data of the device and hence the user in the form of coordinates can be sent to the adversary with a bit development work. In addition to this, phone and device identity can be extracted by this malicious application for the adverse effects like phone blocking or damaging.

In order to reduce the Wi-Fi/3G/4G data usage, attacker can work on the taken screenshots, extract the required data and send some characters or letters over to the server, instead of the whole screenshot image. For this, passwords can be extracted using a local mechanism of characters extraction like what is done by Screenmilker, but it needs to be make more efficient in terms of performance overhead and resources consumption. This can result in sensitive accounts hacking like banking and several social media applications.

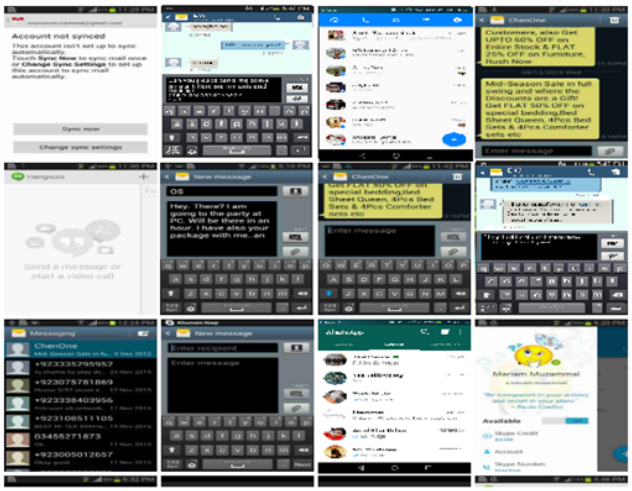

FIGURE 6. Screenshots uploaded on adversary's server.

IV. RESULTS AND EVALUATION

This section encompasses the results and evaluation of ScreenStealer and its effectiveness in stealing sensitive data from user's Android devices. The efficiency and sneaking of the developed malware has been evaluated on the basis of various parameters, like, capture ratio, performance overhead, and consumption of resources, like CPU, memory, network. The evaluation parameters have been selected based on ScreenStealer application features and in accordance with the conceivable consequences it can originate on user device while it is under surveillance.

The experiments and tests are performed on Samsung GT-S7262 Android 4.1.2, Motorola Moto X Android 4.4.2, and Rivo Rhythm RX70 Android 4.4.2. For the purpose of evaluation, PowerTutor, App-TuneUp-Kit, Trepn Profiler, and Android Studio features have been employed to analyze values of parameters. Samsung GT-S7262 has been used for evaluation results that are later explained in this section.

A. EFFECTIVENESS AND EFFICIENCY

This sub-section presents the understanding of effectiveness and efficiency of the developed malware in collecting sensitive user data. The selected parameters to determine the efficiency and effectiveness of ScreenStealer are, display detection, capturing ratio, and data extraction. These factors are very important to be considered regarding the malicious functionality of ScreenStealer, because the malware development has been focused towards capturing a screen of any of the target applications with meaningful information and at a perfect time.

App Monitoring and Display Detection: At first, it was observed whether ScreenStealer can impeccably determine when the target application is active at the foreground. To accomplish this, a number of applications' package names were coded in ScreenStealer to detect, and examinations for all of the target applications were successful. The ScreenStealer's background service runs a loop for checking any of the target applications with an interval of 5 seconds, once it notices any, it rings the module dedicated for capturing screens. The target applications that were hard-coded for ScreenStealer are composed of Messaging, Gmail, Hangout, Facebook, Messenger, WhatsApp, Instagram, Snap Chat, Contacts, Call logs, Viber, Breeze Pakistan, Twitter and Skype. Additionally, ScreenStealer can be programmed to capture data of many other applications specified for screen capturing.

Capturing Ratio: The number of screenshots being taken in a specified time interval while the target application is running at the frontend is determined by Capture ratio ADB workaround and consumption of network limits the rate at which screens are captured by ScreenStealer and sent to the server through web service. When the target application is discovered by ScreenStealer at the front-end, the screen capturing module is alerted at once and with 20 seconds interval screenshots are captured and uploaded to the server, as long as the target application is running at foreground. (Figure 7).

B. STEALTHINESS

In order to evade being sensed by the user, ScreenStealer is developed and designed to reduce the consumption of resources and minimize the use of CPU, memory and network. The stealthiness of the application has been measured in a number of ways, which include the performance overhead in terms of response time, assessment of CPU consumption of the ScreenStealer when it is active and when only spying from the backend, memory usage and the network consumption.

Resources Consumption: A chain of experiments has been performed in order to analyze whether checking the system resources consumption can reveal the presence of malware in the system. According to a study, most of the energy of a smart device is consumed by screen and CPU [34]. The energy consumed by on screen and its brightness is not relevant to our study, but we explicitly examined the CPU usage of ScreenStealer as compared to other applications running

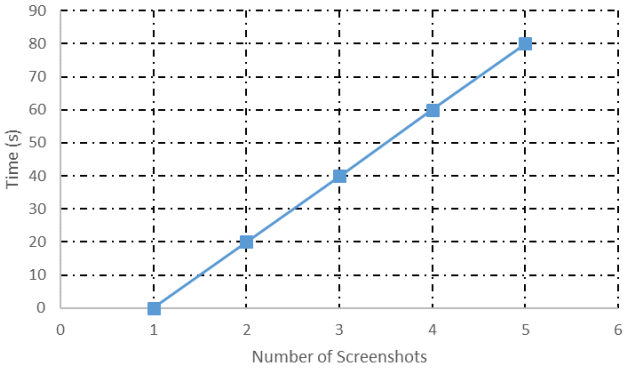

FIGURE 7. Capture ratio of ScreenStealer.

TABLE 2. Average execution time of each of the ScreenStealer's main functions.

ScreenStealer's function	Time (ms)
App-detection	1.52 (1% CPU overhead)
Screen capturing	160.12
Encryption and storage	0.23
Network detection (server upload)	2.43

on the device. As described earlier, the app detection module operates once every 5 seconds in order to detect the running target applications. Each time this module is invoked, it took about 1.52ms to complete its job, which cause less than 1% of CPU overhead, which is very much likely to be ignored.

1) OVERALL EXECUTION TIME

When the objected application is identified, screenshot taking module becomes operational. On the average, the module takes 160.12ms to take screenshot and stored it for the time in encoded form in device storage. Encoding takes less than 0.23ms on average. Also, when the network connection establishes, ScreenStealer takes almost 2.43ms to send the screenshot to the server through web service. Moreover, as the screenshot is transmitted by network, so the time depends on the internet connection. Such level of resource utilization does not cost markable performance distress. Breakdown is presented in Table 2.

2) MEMORY USAGE

Furthermore, evaluation of ScreenStealer is done on the basis of memory usage as compared to other authenticated applications, such as, Skype, Calculator, Calendar, Google Play Music, Gallery, Facebook, WhatsApp, Google Play Store, Temple Run and Subway Surfers. The memory usage of these applications has been monitored using Trepn Profiler and Android Studio, the average results are presented in Table 3.

It can be observed that the ScreenStealer does not consume memory resources in a susceptible way and use relatively a low amount of memory. Moreover, its memory usage is similar to those of small sized applications like calculator.

TABLE 3. Average memory consumption of ScreenStealer as compared to other popular apps.

Application	Memory [Kbyte]
ScreenStealer - App detection	220.8
ScreenStealer - Screen capturing	269.2
ScreenStealer - Sending to Server	297.3
Calculator	279.4
Calendar	301.8
Google Play Music	317.6
Skype	402.8
Gallery	369.6
Facebook	357.1
WhatsApp	303.5
Google Play Store	456.1
Temple Run	435.7
Subway Surfers	329.2

FIGURE 8. Size of screenshots with 320 × 480 resolution and quality factor of 20 in JPEG format.

Other heavy applications like Subway Surfers and Temple Run consume about 55% more memory than ScreenStealer.

3) POWER CONSUMPTION

CPU and Network power usage is examined by using the Trepn Profiler. Evaluation was done by the cross match of values with results from other tools like PowerTutor and AppTune-up Kit. Figure 8 shows the average values over the 5 minutes period compared with other acknowledged applications. ScreenStealer consumes a moderate level of power in comparison with other applications.

4) IMAGE SIZE REDUCTION

To lessen the memory utilization for temporary screenshot storage in device and low usage of network data, we shrink the quality and resolution of image to the level that makes it easy and faster to send it over the server with reduced usage of memory and network resources. JPEG is a lossy format, it may impair the image quality by reducing the size, as image quality is not the concern in our case, so JPEG format is employed to shrink its size to maximum with the settlement of quality and resolution. Therefore, resolution for screenshots is between 3KB to 17KB maximum in size, with 320x480 resolution and quality factor of 20 (Figure 9).

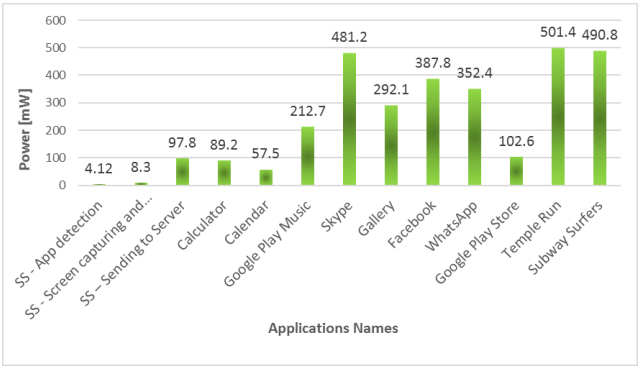

FIGURE 9. Average power usage.

C. DISCUSSIONS

The real-time data stealing technique of ScreenStealer is highly efficient and effective. However, the limitations are still there due to the limited capability of screenshot taking in Android devices and somewhat smart user interface of Android devices which varies by different vendors and Android versions. For example, if ScreenStealer wants to track instant messages sent and received on user device through screenshots, then it is a possibility that the messages are coming and going faster than the rate of screenshots are being taken and uploaded to the server.

Though this strategy is very much effective where messages are displayed in the form of thread, which is usually the scenario in most of the latest and upcoming models of smartphones, this makes ScreenStealer able to capture 3 to 4 messages of about 20 words in one screenshot, however, the number of text messages captured is reduced, when the keyboard is opened in screenshot, that covers about one third of the smartphone screen. Similar types of limitations can be faced when ScreenStealer tries to steal contacts information.

V. CONCLUSION

The ADB based workaround has become an ordinary and legally eligible way for privilege escalation and to obtain signature-level permissions. This also normally require the use of local socket channel to develop a connection between the native service and the application running on device. Android, however, doesn't support any control of access or monitoring of this channel. This can lead to exploitation of this workaround and channel by any third-party application in a malicious way.

Almost all screenshot, sync and backup, and USB tethering application on google play store uses ADB proxy and deficit any methodology for unauthorized access of malicious applications. Whereas, the sensitive applications such as banking, Skype, messaging or any related applications does not use any methodology to avert any bogus applications to examine on-screen activities. We recognized that, security procedures are needed to be upgraded on individual application and OS level and also on Google Play Store. A real-time example is the upload and verification of our malicious screenshot application by Google Play Store, by introducing it as

Tic-Tac-Toe game for users. Our experiment exhibits that verification procedure of Play store needs to be upgraded to detect susceptible activities and approbations by any application without depending on user knowledge only.

Future work for this research will aim to investigate loopholes of already available security procedures in depth and execution and development of the suggested mitigation procedures in practical and constructive ways.

REFERENCES

[1] The Statistics Portal. *Number of Available Applications in the Google Play Store From December 2009 to February 2016*. Accessed: Jun. 2016. [Online]. Available: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/
[2] C. Dehghanpoor. *Brain Test Re-Emerges: 13 Apps Found in Google Play*. Accessed: Jun. 2016. [Online]. Available: https://blog.lookout.com/blog/2016/01/06/brain-test-re-emerges/
[3] Google Play Store. *Malwarebytes Anti-Malware*. Accessed: Jun. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=org.malwarebytes.antimalware&hl=en
[4] M. La Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 446–471, 1st Quart.,2013.
[5] F. Wu, H. Narang, and D. Clarke, "An overview of mobile malware and solutions," *J. Comput. Commun.*, vol. 2, no. 2, pp. 8–17, 2014.
[6] S. M. Muzammal and M. A. Shah, "ScreenStealer: Addressing screenshot attacks on Android devices," in *Proc. Int. Conf. Autom. Comput. (ICAC)*, Sep. 2016, pp. 336–341.
[7] S. Mukherjee, J. Prakash, and D. Kumar, "Android application development & its security," *Int. J. Comput. Sci. Mobile Comput.*, vol. 4, no. 3, pp. 714–719, Mar. 2015.
[8] X.-D. Qu and G. Yu, "Coordinated attack research between Android applications and solutions," in *Proc. 5th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Jun. 2014, pp. 718–722.
[9] Android Developers. *Android Debug Bridge*. Accessed: Apr. 2016. [Online]. Available: http://developer.android.com/intl/zh-CN/tools/help/adb.html
[10] Android Developers. *Download Android Studio and SDK Tools*. Accessed: Apr. 2016. [Online]. Available: http://developer.android.com/intl/zh-CN/sdk/index.html
[11] A. G. Villan and J. Jorba. (2013). "Remote control of mobile devices in Android platform." [Online]. Available: https://arxiv.org/abs/1310.5850
[12] K. Lucic. *Over 27.44% Users Root Their Phone(s) in Order to Remove Built-In Apps, Are You One of Them? Android Headlines*. Accessed: May 2016. [Online]. Available: http://www.androidheadlines.com/2014/11/50-users-root-phones-order-remove-built-apps-one.html
[13] S. Smalley and R. Craig, "Security enhanced (SE) Android: Bringing flexible MAC to Android," in *Proc. NDSS*, vol. 310. 2013, pp. 20–38.
[14] C.-C. Lin, H. Li, X. Zhou, and X. Wang, "Screenmilker: How to milk your Android screen for secrets," in *Proc. NDSS*, 2014, pp. 1–14.
[15] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, "Measuring user confidence in smartphone security and privacy," in *Proc. 8th Symp. Usable Privacy Security*, 2012, Art. no. 1.
[16] A. Mylonas, D. Gritzalis, B. Tsoumas, and T. Apostolopoulos, "A qualitative metrics vector for the awareness of smartphone security users," in *Trust, Privacy, and Security in Digital Business*. Berlin, Germany: Springer, 2013, pp. 173–184.
[17] B. Rashidi and C. Fung, "A survey of Android security threats and defenses," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 6, no. 3, pp. 3–5, 2015.
[18] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proc. 8th Symp. Usable Privacy Security*, 2011, Art. no. 3.
[19] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, "A conundrum of permissions: Installing applications on an Android smartphone," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 68–79.
[20] (Oct. 2014.). *The Impact of Mobile Devices on Information Security: A Survey of IT and Security Professionals*. Accessed: Jun. 2016. [Online]. Available: http://www.checkpoint.com/downloads/products/check-point-mobile-security-survey-report.pdf
[21] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2011, pp. 3–14.
[22] *The Problem with Mobile Phones: Surveillance Self-Defense, A Project of the Electronic Frontier Education*. Accessed: Jun. 2016. [Online]. Available: https://ssd.eff.org/en/module/problem-mobile-phones
[23] W. Jeon, J. Kim, Y. Lee, and D. Won, "A practical analysis of smartphone security," in *Proc. Human Interface Manage. Inf. Interact. Inf.*, 2011, pp. 311–320.
[24] X. Ni, Z. Yang, X. Bai, A. C. Champion, and D. Xuan, "DiffUser: Differentiated user access control on smartphones," in *Proc. IEEE 6th Int. Conf. Mobile Adhoc Sensor Syst.*, Oct. 2009, pp. 1012–1017.
[25] Accounts Help. *Protect Against Harmful Apps*. Accessed: Apr. 2016. [Online]. Available: https://support.google.com/accounts/answer/2812853?hl=en
[26] J. Hindy, (2016). *15 Best Antivirus Android Apps and Anti-Malware Android Apps*. Accessed: Apr. 2016. [Online]. Available: http://www.androidauthority.com/best-antivirus-android-apps-269696/
[27] G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, and D. Sgandurra, "A multi-criteria-based evaluation of Android applications," in *Trusted Systems*. Berlin, Germany: Springer, 2012, pp. 67–82.
[28] Grey Shirts Productivity. (2014). *NoRoot Firewall*. Accessed: Jun. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=app.greyshirts.firewall&hl=en
[29] Z. R. Rodrigo. (2011). *DroidWall - Android Firewall*. Accessed: Jun. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.googlecode.droidwall.free&hl=en
[30] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A. R. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on Android," in *Proc. 1st ACM Workshop Security Privacy Smartphones Mobile Devices*, 2011, pp. 51–62.
[31] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on Android)," in *Trust and Trustworthy Computing*. Berlin, Germany: Springer, 2011, pp. 93–107.
[32] Whatatech. (2016). *TicTacToe*. Accessed: Apr. 2016. [Online]. Available: https://play.google.com/store/apps/details?id=screenshot.screenshot project
[33] Google Code-Archive. *Android Screenshot Library*. Accessed: Apr. 2016. [Online]. Available: https://code.google.com/archive/p/android-screenshot-library/
[34] C. Cortes and A. Krauser. *Android: Resource Consumption in Native and Web Applications*. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A832028&dswid=5995. 2013.

SYEDA MARIAM MUZAMMAL received the B.Sc. and M.Sc. degrees in computer science from the COMSATS Institute of Information Technology, Islamabad, Pakistan, in 2012 and 2016, respectively. Her M. Sc. dissertation topic is based on Security Attacks and User's Privacy Protection in Smartphones. Her research interests include security risks and threats in smart devices, ethical hacking, information security, Internet of Things, and machine learning.

MUNAM ALI SHAH received the B.Sc. and M.Sc. degrees in computer science from the University of Peshawar, Pakistan, in 2001 and 2003, respectively, the M.S. degree in security technologies and applications from the University of Surrey, U.K., in 2010, and the Ph.D. degree from the University of Bedfordshire, U.K., in 2013. Since 2004, he has been a Lecturer with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. He has authored over 50 research articles published in international conferences and journals. His research interests include MAC protocol design, QoS, and security issues in wireless communication systems. He received the Best Paper Award of the International Conference on Automation and Computing in 2012.

HASAN ALI KHATTAK received the B.CS. degree in computer science from the University of Peshawar, Peshawar, Pakistan, in 2006, the master's degree in information engineering from the Politecnico di Torino, Torino, Italy, in 2011, and the Ph.D. degree in electrical and computer engineering from the Politecnico di Bari, Bari, Italy, in 2015. He has been serving as an Assistant Professor of computer science with the COMSATS Institute of Information Technology since 2016. His current research interests focus on distributed system, Web of things and vehicular Ad Hoc networks, and data and social engineering for smart cities. He is involved in a number of funded research projects in the Internet of Things, semantic Web and fog computing while exploring Ontologies, Web Technologies using Contiki OS, NS 2/3, and Omnet++ frameworks.

SOHAIL JABBAR was a Post-Doctoral Researcher with Kyungpook National University, Daegu, South Korea. He has also served as an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology (CIIT), Sahiwal, where he also headed the Networks and Communication Research Group. He is currently an Assistant Professor with the Department of Computer Science, and the Director of the Graduate Programs with the Faculty of Sciences, National Textile University, Faisalabad, Pakistan. He has authored one book, two book chapters, and over 60 research papers. His research work is published in various renowned journals and magazines of the IEEE, Springer, Elsevier, MDPI, Old City Publication and Hindawi, and conference proceedings of the IEEE, ACM, and IAENG. He is on collaborative research with renowned research centers and institutes around the globe on various issues in the domains of Internet of Things, wireless sensor networks and big data. He received many awards and honors from Higher Education Commission of Pakistan, Bahria University, CIIT, and the Korean Government. Among those awards, the Best Student Research Awards of the Year, the Research Productivity Award, and BK-21 Plus Post Doc. Fellowship. He received the Research Productivity Award from CIIT in 2014 and 2015. He has been engaged in many national and international level projects. He has been the reviewer for leading journals (ACM TOSN, JoS, MTAP, AHSWN, and ATECS) and conferences (C-CODE 2017, ACM SAC 2016, and ICACT 2016). He is currently engaged as a TPC member/chair in many conferences. He is the Guest Editor of special issue on *Concurrency and Computation Practice and Experience (Wiley)*, *Future Generation Computer Systems (Elsevier)*, *Peer-to-Peer Networking and Applications (Springer)*, *Journal of Information and Processing System (KIPS)*, and *Cyber-Physical System (Taylor & Francis)*.

GHUFRAN AHMED received the Ph.D. degree from the Department of Computer Science, Capital University of Science and Technology, Islamabad, in 2013, the Post-Doctoral degree from the Department of Computer Science and Digital Technology, Faculty of Engineering and Environment, Northumbria University, Newcastle Upon Tyne, U.K., in 2015, and the Ph.D. degree from the Faculty of Computer Science and Engineering, GIK Institute, Topi, Swabi, KPK. He was a Visiting Scholar with the CReWMaN Lab, Department of Computer Science and Engineering, The University of Texas at Arlington from 2008 to 2009. He is currently serving as an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. His area of research is wireless sensor networks and wireless body area networks.

SHEHZAD KHALID graduated the degree from the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan, in 2000, the M. Sc. degree from the National University of Science and Technology, Pakistan, in 2003, and the Ph.D. degree from the University of Manchester, U.K., in 2009. He is currently a Professor and the Head of the Department with the Department of Computer Engineering, Bahria University, Pakistan. He is a qualified academician and a Researcher with over 60 international publications in various renowned journals and conference proceedings. He is the Head of the Computer Vision and Pattern Recognition Research Group which is a vibrant research group undertaking various funded research projects. His areas of research include but are not limited to shape analysis and recognition, motion-based data mining and behavior recognition, medical image analysis, ECG analysis for disease detection, biometrics using fingerprints, vessels patterns of hands/retina of eyes, ECG, Urdu stemmer development, short and long multi-lingual text mining, and Urdu OCR. He has been the reviewer for various leading ISI indexed journals. He has received the Best Researcher Award from Bahria University in 2014. He has also received the Letter of Appreciation for Outstanding research contribution in 2013 and the Outstanding Performance Award for the academic year 2013-2014.

SHAHID HUSSAIN received the Ph.D. degree from the Department of Computer Science, City University of Hong Kong. He is currently an Assistant Professor of computer science with the COMSATS Institute of Information Technology, Islamabad, Pakistan. He has authored numerous articles in reputed journal and conferences. His major research interests include software design patterns, fault prediction, text mining, data science, and metrics threshold.

KIJUN HAN received the B.S. degree in electrical engineering from Seoul National University, South Korea, in 1979, and the M.S. degree in electrical engineering from the KAIST, South Korea, in 1981, and the M.S and Ph.D. degrees in computer engineering from the University of Arizona, in 1985 and 1987, respectively. He served as a Researcher of Agency for Defense Development from 1981 to 1984. He has been holding the professor position with the School of Computer Science and Engineering, Kyungpook National University, South Korea, since 1988. He headed good number of national and international level projects. His research work is published in various renowned journals and magazines of the IEEE, Springer, Elsevier, MDPI, Old City Publication and Hindawi, and conference proceedings of the IEEE and ACM. His research interests include Internet of Things, Wireless Sensor Networks, and big data.