Journal of Information Security, 2019, 10, 91-102
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Android Security and Its Rooting—A Possible Improvement of Its Security Architecture

Nick Rahimi[1], John Nolen[2], Bidyut Gupta[2]

[1]Department of Computer Science, Southeast Missouri State University, Cape Girardeau, MO, United States
[2]Department of Computer Science, Southern Illinois University, Carbondale, IL, United States
Email: srahimi@semo.edu, Tennotsukai87@siu.edu, bidyut@cs.siu.edu

How to cite this paper: Rahimi, N., Nolen, J. and Gupta, B. (2019) Android Security and Its Rooting—A Possible Improvement of Its Security Architecture. *Journal of Information Security*, **10**, 91-102.
https://doi.org/10.4236/jis.2019.102005

Received: August 5, 2018
Accepted: March 29, 2019
Published: April 1, 2019

Abstract

The advent of technology brought forth a myriad of developments that have streamlined the manner through which people operate. With the growing need to be at the forefront of communication and information, people have resorted to the use of mobile phones with a great percentile preferring android oriented systems. Similarly, the systems are susceptible to the various threats posed by technology with due summations showing that security flaws and unauthorized access to sensitive data pose a huge threat to the overarching efficacy of the android systems. The research presented lays a primal focus on how users can improve intrinsic android features through the use of Google services, rooting, custom kernels and ROM techniques. The research also focused on how Android security features can be improved when using or installing applications. Results indicate that the rooting process is the most conclusive and safest scheme. Summations drawn are indicative of the fact that system security is a moot research topic that requires further research into how it can be improved.

Keywords

Android Security, Sandbox, Bricking, Firmware, Rooting

1. Introduction

For the past decade and a half, the incorporation of Android software into mobile phones has taken over the contemporary market. Summative figures drawn from a myriad of research show that at least 52.8% of smartphone users own an Android device with the numbers increasing by the day [1]. This inadvertently means that with the rise in numbers comes the rise in attacks and threats to the operating systems [2]. Prior to highlighting the various issues faced by Android systems, one has to understand the inherent structure of the software addendum to their levels of protection and system architecture [3].

Android developers have focused on ensuring that the security architecture of the systems is dependent on a Linux Based Kernels which ensures for the isolation of applications from each other in order to ensure for cohesion [4]. The isolation technique requires signatures and various permissions in order to run accordingly with the security model being intricate by design. The intricacies are developed in a manner through which the user is able to revamp and change the preferences as they deem fit. An example can be given by going through the set security settings so as to change the preferences as the user deems fit. Most times, the tasks may seem to be daunting although the Android operating system comes intact with descriptions on how one can change the settings [5].

Various scholars have presented musings that are solely focused on how such attacks can be mitigated which sets the precedent for this research. The most popular scheme of mitigating the attacks is through the rooting of the devices which helps unlock various features hidden on Android devices for better user experience [6]. It is imperative for users however to understand that rooting an Android device comes with its own demerits and ethical issues with the topmost being that the warranty is rendered null and void. In most cases also, the Android device can become unresponsive in a situation called "bricking" [7] [8].

The subsequent tenets of the paper will follow through giving a description of the security architecture, the security model of the android systems secure and the user security roles. The fifth section highlights the efficacy of the Google services systems and how they mitigate the inconsistencies that come with the system. The 6th section lays a focus on rooting and its efficacy addendum to its demerits with conclusions being accorded in the final chapter.

2. Security Architecture

Being an open platform, Android accords both its normal and knowledgeable users the ability to place multi-layered security features on every device which helps boost their experience. The intrinsic framework used by the Android operating system is shown in **Figure 1** in which every layer works in tandem with the other layers to ensure that the system is secure. The layers are all sandboxed into a parent layer which bases its ideas from the Linux security component of isolating them to allow for cohesion [9].

2.1. Linux Kernel

The first layer which is the Linux kernel is embedded in the device and provides many hardware applications and software. Derivatives from research show that the hardware drivers that are used by the kernel are inclusive of the display data, the camera, and the Bluetooth devices. This makes it hard for one to try and revamp it as any mistakes can lead to the device structure crashing. Android developers have mitigated the occurrence of such situations through the minimization of permission access to the structural tenets [10].

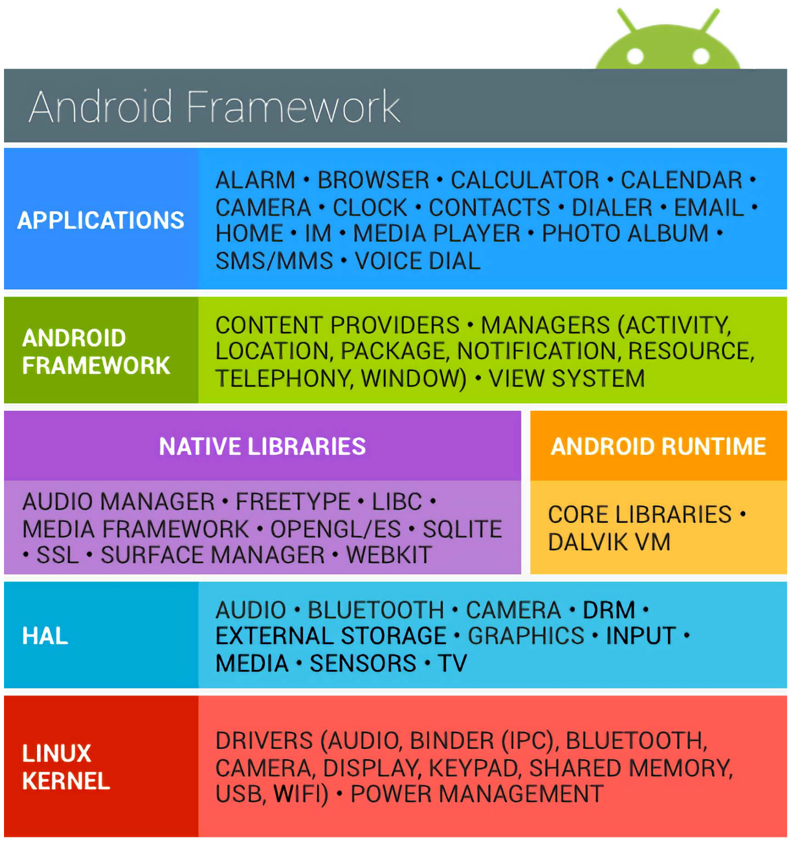

Android Framework
APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

Figure 1. Android security framework [2].

2.2. HAL Component

The subsequent layer is the hardware abstraction layer which creates a nexus between the device software and the hardware. Users are also limited from configuring this layer on their own as the systems depend on it for data analysis and efficacy of the system. The HAL component works in thrall with the Linux Kernel to ensure that the user experience is maximized [11].

2.3. Native Libraries and Android Runtime

The HAL layer comes intact with two isolated tenets that include the Android runtime and the native libraries. Owing to the fact that the Android device is formulated through the use of a Linux Kernel, a good chunk of the C and C+++ libraries are required for the development process. The Android runtime feature contains a milieu that allows for the system to run effectively. Previously known as Dalvik, the runtime feature helps in the compilation of the system codes when installing into the mobile device [12].

2.4. Android Framework Library and Application Layer

This component of the Android system contains the required APIS used by Java systems that can help in the building of the android systems. The application layer is vital in the provision of the data and graphical user interface requirements of the devices. The application layer, on the other hand, comes intact with system applications which are separated from the user installed apps. The applications depend highly on the resources accorded by the system from the lower end of the framework.

3. Security Model

There are several concepts to the security model that allow the system to provide better security for all users. The model consists of a kernel-based application sandbox, secure IPC, system services with reduced privileges, code signing, and application permissions.

3.1. Kernel-Based Application Sandboxing

The kernel-based application sandbox gives each application separate UIDs (User IDs) and GIDs (Group IDs). This prevents applications from accessing other applications' data. As such, a downloaded application will not have access to system.

3.2. Secure Inter-Process Communication (IPC)

Seeing as certain applications do need access to other applications, secure IPC gives access to the important applications to communicate with one another. This is done via local sockets, binders, and intents with most applications utilizing the local sockets to securely communicate with other apps. The binders are also key in the implementation of the code into the Android environment and they are only activated whenever the application is running. Research indicates that the intents help in the collection of the information on the operations that have to be performed addendum to the previous events [13].

3.3. Reduced Privileges

As noted prior, there is a myriad of native codes that are contained within the application framework of the Android systems. This leads to the Android developers being forced to make most of the system services to run by utilizing fewer privileges in a bid to help in the prevention of malicious use of the system resources. The native codes are accorded various signatures which are similar to certificate authorities as shown in the next section.

3.4. Code Signatures

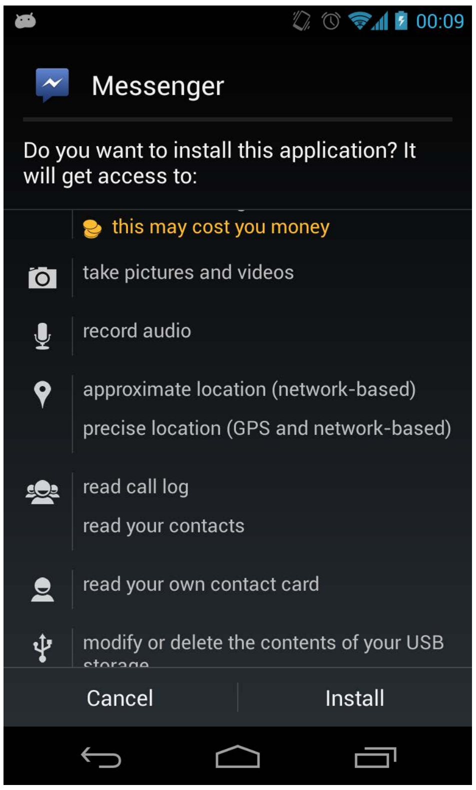
Compared to IOS systems, Android systems contain a different set of code signage in which they do not require authorization from the user during the signing process. The default process, however, requires the code signing of the applications which is later on applied on the security model. The signatures also use public keys for the cryptographic process addendum to X.509 certificates. Code signatures differ depending on the Java JAR signing scheme. Code signatures provide authenticity for said application with an example of a code signature being shown below:

```
    <permissions>
    ..
    <item name =
“com.google.android.googleapps.permission.ACCESS_GOOGLE_PASSWORD”
    package = “com.google.android.gsf.login” protection = “2”/>
    ...
    </permissions> [14].
```

There are several reasons as to why code signatures are important with the first being making sure updates for an app are coming from the same author (same origin policy) addendum to establishing trust relationships between applications. The aforementioned roles are both implemented through the drawing of comparisons between the certificates signed on the installed target application against the update certificate [14]. Over-the-air (OTA) updates, as well as application patches, depend on code signatures to ensure authenticity and identity.

3.5. Android Application Permissions

Finally, Android application permissions have been implemented to aid the user in preventing certain applications from gaining access to important system resources. Figure 2 is a representation of the installation process of an application in which the application requests for permission and limits access to other applications. The Android system allows the user to choose whether to install the


00:09

Messenger

Do you want to install this application? It will get access to:

this may cost you money

take pictures and videos

record audio

approximate location (network-based)

precise location (GPS and network-based)

read call log

read your contacts

read your own contact card

modify or delete the contents of your USB storage

Figure 2. Android permissions [14].

application after which they cannot revoke the privileges. The Linux kernel sorts the process of privilege accordance and giving access to the system files and other applications.

4. How to Keep Android Applications Secure

There are several approaches to take when securing Android apps and programs and from two different standpoints. As a developer, there are certain actions one can do to better ensure security for their users. One such example would be through penetration testing in which the developer identifies and mitigates any risks that the user is susceptible to. The penetration testing process takes into account a number of processes which include attack surfacing, analysis of the application interactions with the files, the storage capability of the system and the communication between the entities. The second standpoint comes from the securing of the application by the average user in order to enhance its productivity.

4.1. Attack Surface

The attack surface focuses on the primary functions of the application. Activities like the generation of random characters and fuzz testing can help in the prevention of the applications from undue crashing. More time should be spent on application components that deal with key system components.

4.2. Interactions with Devices/Applications

Interacting with other devices or applications can be done via IPC mechanics as previously discussed. A remote procedure call can work with the socket-based communications to enhance the interactions. The intents and broadcast features, on the other hand, work to limit the provision of unnecessary permissions to various applications thus reducing the risk factor [15].

4.3. Communication

Communications outside of a system can be another risk factor. Usually, most popular applications will utilize some form of cryptography to limit unwarranted access. When developing an application, it is quite imperative for the user to never hardcode a crypto-key, as applications on Android are open source meaning intrusion is imminent [16].

4.4. Storage

Testing an application which assesses sensitive data is vital to fixing all loopholes that may be apparent. Ensuring the proper read/write permissions are allowed with maximum restrictions when pertaining to system files can also limit the number of applications gaining access to the user data. When penetration testing data applications the developer has to assess, review what information is being stored to ensure needless data is being accumulated.

5. User Optional Security Roles

A user has his or her own responsibilities to utilize the correct security features needed to complete the CIA triad [17]. The following security tips may require some technical expertise; regardless, these are some tips for any user to keep from dealing with malicious attacks. These steps may require additional research within the given references as there is a dearth of knowledge existent in contemporary research.

Security Tips

- Storing Data—By default, most data go into internal storage of an Android device which allows for fast access. You may also encrypt files with a key using the KeyStore which protects a file using a password. External storage of applications or data is the least secure method like take, for example, an SD card inserted into a phone are globally readable/writeable by default; however, with newer versions of Android, it is possible to encrypt external SD cards. For the safest storage, use cryptography schemes as most applications work with a Java Cryptography Architect in which they generally use 256-bit AES, 256-bit public keys, CBC, CTR, or GCM modes, ensure integrity with HMAC-SHA1/256/512 or GCM [18]. Using full or file disk encryption is an exemplary way to provide confidentiality for one's Android device which uses AES (128), CBC, and SHA256 to encrypt information.
- Screen Lock—Whether it be from a key lock, iris scanner, fingerprint scanner, PIN lock, or facial recognition software, this is one of best ways to keep an unwanted passerby from accessing data on your device.
- Use Application Lockers—There are thousands of applications within the Google Play Store which can be used in preventing users from accessing certain apps such as the Gallery or Camera. It is however best to check reviews and the popularity of said application to ensure the safety of your data. Not all applications on the Play Store are genuine.
- Keep OS Up-to-date—This should really go without saying, but many users disregard OTA updates for their phones and generally find them annoying to deal with. It only takes a couple of minutes, and OTA patches usually contain several security features and fixes that could prevent malicious actions on a device.
- Manage Permissions—As stated prior, the user has to ensure that the applications they are installing are not frivolous with its required permissions to be granted. With Android's Marshmallow firmware, a user can now prevent an application from using permission. This may be contradictory to what was stated earlier, but the application will not run correctly with this in mind and will result in explicitly asking the user for granted permissions to run. However, this may ensure certain third-party applications do not handle certain hardware or files if the feeling of concern is there. **Figure 3** shows good layered control of what happens before and during installing an application.

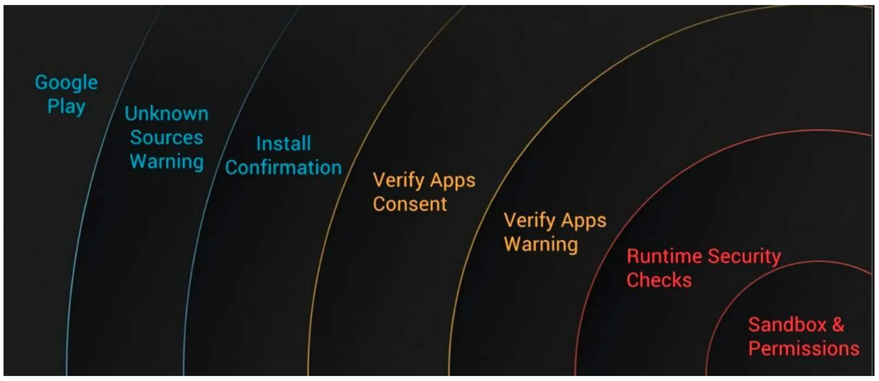


Figure 3. Multiple layers of defense [21].

- Remote Wipe and Tracking—Another feature Android provides is remote wiping and tracking in case a thief grabs your phone.
- Backing Up Files—Users are able to back-up their applications and files to a hard drive or to the cloud.
- Malware Prevention—A good way to prevent malware is to not click on anything that looks too good to be true. In other words, be careful what websites you visit, pop-up ads that may seem desirable to click on, and knowing the applications you download are trusted.
- Encrypted Messaging—There are several apps on the Play Store that will use point-to-point encryption as a means to message another party. Again, these apps come with their own storage and likely store every message you send. If a server becomes compromised in which you utilize, there is a good chance your messages are out in the open afterward [19] [20].

6. Google Services

Most Android devices use Google services as pre-installed software packages to enhance user experiences. Google Play is a collection of applications that may be purchased and/or downloaded. Google services provide a variety of tasks to aid users filter through their cloud database. Actually, Google Services does quite a bit more for Android devices in terms of security. Google Services verifies application licenses within its Play Store, delivers the OTA updates, cloud storage, and scans their Play Store for harmful applications. Due to Android's popularity, this does not mean that Google Services finds all malicious applications in their cloud. There are several precautions a user must take before installing any application as was discussed earlier.

7. Rooting

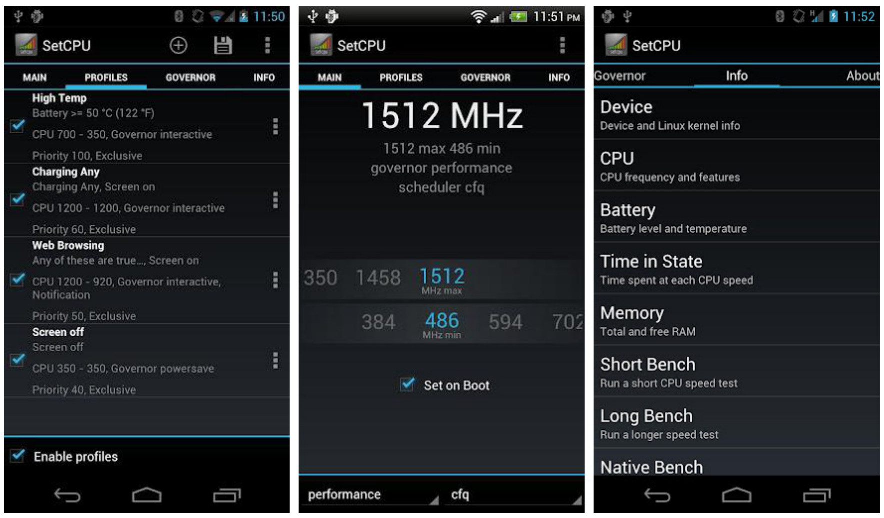
Rooting is something many Android developers originally sought-after Android became popular. Prior Android versions did not allow as much customization as they do today. Much of the information contained in this document related to rooting is provided through personal experiences and literal research knowledge. There is a lot of speculation whether rooting a phone is worth it anymore as there are many downsides to rooting as soon discussed. Even though rooting is not as prevalent today as it used to be, there are many benefits to rooting.

7.1. Benefits

One of the most sought-after benefits of rooting an Android device is to remove bloatware from it. Bloatware can be said to be applications pre-installed on phones by Google, the device developer, and carrier. There are times these pre-installed applications may become problematic with device batteries and resources as some of them will act as background processes. Today, this is less of an issue, as newer versions of Android allow a user to disable pre-installed applications which prevent any events that the application can incur such as auto-updates and notifications. However, disabling an application does not delete the application, so the app just stays in storage for no good use. Rooting a device gives the user complete say on whether an application is significant or insignificant.

Another reason to root one's device and is a big oversight is to provide root access to all files within the device. Root access can be worrisome for most, but, in the right hands, it can be beneficial to the proficient ones. Root access allows users to increase the core speeds of their processors by overclocking them to desired amounts. The negative is also true; one can lower CPU speeds to increase battery life. Figure 4 shows an example root application called SetCPU [22] that provides the change to the CPU governor's minimum and maximum performances. The application is also able to change a multitude of other aspects of an Android device such as memory utilization, temperature control, and show benchmarks or speed tests. The SetCPU will not run on any device as a device needs to be rooted in order to run it.

With that said, there are a plethora of applications on the Play Store that provide several impressive tasks such as SetCPU that require root access to utilize. There are root applications that will let the user automate any tasks, apps that



High Temp
Battery >= 50 °C (122 °F)
CPU 700 - 350, Governor interactive
Priority 100, Exclusive

Charging Any
Charging Any, Screen on
CPU 1200 - 1200, Governor interactive
Priority 60, Exclusive

Web Browsing
Any of these are true..., Screen on
CPU 1200 - 920, Governor interactive, Notification
Priority 50, Exclusive

Screen off
Screen off
CPU 350 - 350, Governor powersave
Priority 40, Exclusive

✓ Enable profiles

1512 MHz
1512 max 486 min
governor performance
scheduler cfq

350 1458 1512 MHz max
384 486 MHz min 594 702

✓ Set on Boot

performance cfq

Figure 4. Set CPU [22].

back up data in all the current states they are in which includes a game and the progress you have made within it, and apps that block ads that freeware come with. The final advantage is that a rooted device gives the user the added advantage of installing a custom kernel and custom ROM.

7.2. Disadvantages

First and foremost, the greatest disadvantage to any rooted device revolves around ethical issues of warranty. When rooting an Android device one of the steps involves unlocking the boot loader. Many times, the boot loader will go through a continuous loop, and this happens often. A very simple step for most devices in most cases, will void the warranty of the said device. The second demerit that stems from the rooting process is the device becoming bricked which leads to them becoming insignificant. Suffice to say that if a device is completely rooted but happens to receive a physical device related issue, there are ways to un-root a device.

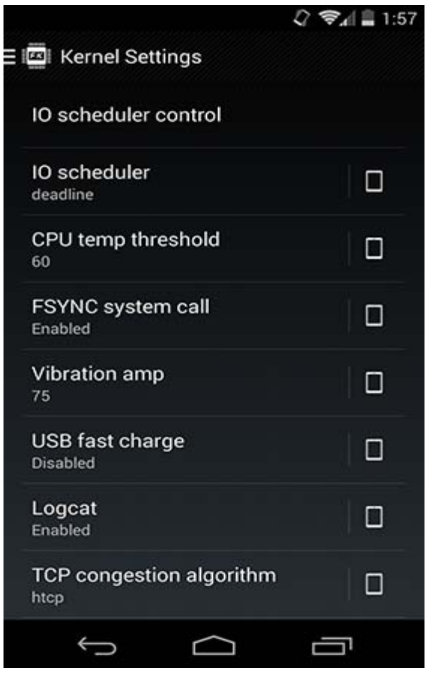
There are times an incompetent root user may mess with system settings too much like in the application SetCPU and cause the device to brick. This situation is no good. There are no warranties for such recklessness. Minor tweaks to system files may completely brick a device, so it is very important to do any research towards a goal before accomplishing any of these tasks.

Security is another issue that comes with rooting the devices. Come to think of it, there are many times a rooter will need to downgrade his or her device firmware to unlock the boot loader. Once the root is complete, installing any OTA updates is out of the question. Root users tend to turn off those notifications immediately upon completely rooting a device. The problem with this shows that root users must use a downgraded version of Android, usually containing public security flaws and custom bugs.

7.3. Custom ROMS and Kernels

Better defined as custom read-only memory, the systems are vital in maximizing the security and productivity of an android system. This is shown especially in cases where the android firmware downgrades as a result of the rooting process with Custom ROMS updating the system with bug and security fixes. Another desirable aspect of installing a custom ROM is that many ROM developers are incredibly fast to release newer versions of their ROMs to be on par with the newest stock firmware released. This combined with the fact that one can modify the taskbar or revamp the intrinsic designs of the systems show that the Custom ROMS are vital components of the android systems. Finally, all the features of many root apps and a custom kernel, custom ROMs provide better battery durations, better benchmark performances, and removal of bloatware.

A custom kernel provides everything the normal Linux-based kernel provides but with many tweaked settings to hardware settings. Basically, it is a revamped version of the stock kernel. It also provides many of the benefits as many root applications provide such as Set CPU as shown in **Figure 5**.



1:57

Kernel Settings

IO scheduler control

IO scheduler
deadline

CPU temp threshold
60

FSYNC system call
Enabled

Vibration amp
75

USB fast charge
Disabled

Logcat
Enabled

TCP congestion algorithm
htcp

Figure 5. Kernel Menu [23].

8. Conclusion

Security is an amorphous concept that has to be upheld in a bid to limit any cases of data breaches addendum to low user satisfaction rates. The intricate features of the Android systems have made it easy for hackers to gain access to data which calls for the formulation or usage of previous security techniques such as rooting. The preceding analysis shows that the various layers of the Android system are all susceptible to interference with the merits and demerits of rooting being explained succinctly. Further research is however required into the importance of the technique in upholding the integrity of the system.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] ComScore Reports January 2016 US Smartphone Subscriber Market Share (2016).

[2] Enck, W., Ongtang, M. and McDaniel, P. (2009) Understanding Android Security. *IEEE Security & Privacy*, **7**, 50-57. https://doi.org/10.1109/MSP.2009.26

[3] Davi, L., Dmitrienko, A., Sadeghi, A.R. and Winandy, M. (2010) Privilege Escalation Attacks on Android. *International Conference on Information Security*, Chengdu, 17-19 December 2010, 346-360.

[4] Gunasekera, S. (2012) Android Security Architecture. In: Gunasekera, S., Ed., *Android Apps Security*, Apress, Berkeley, 31-45. https://doi.org/10.1007/978-1-4302-4063-1_3

[5] Blasing, T., Batyuk, L., Schmidt, A.D., Camtepe, S.A. and Albayrak, S. (2010) An

Android Application Sandbox System for Suspicious Software Detection. 5*th International Conference on Malicious and Unwanted Software*, Nancy, 19-20 October 2010, 55-62. https://doi.org/10.1109/MALWARE.2010.5665792

[6] Sun, S.T., Cuadros, A. and Beznosov, K. (2015) Android Rooting: Methods, Detection, and Evasion. 5*th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, Denver, 12-16 October 2015, 3-14. https://doi.org/10.1145/2808117.2808126

[7] Duffy, J. (2011) A Concise Guide to Android Rooting. Pcmag. https://uk.pcmag.com/feature/112875/a-concise-guide-to-android-rooting

[8] Gaikar, V. (2013) Android Rooting and Risks Involved.

[9] Brickell, E.F., Hall, C.D., Cihula, J.F. and Uhlig, R. (2011) US Patent No. 7,908,653. US Patent and Trademark Office, Washington DC.

[10] Android Open Source Project (2017).

[11] Siddha, V., Ishiguro, K. and Hernandez, G.A. (2012) US Patent No. 8,254,285. US Patent and Trademark Office, Washington DC.

[12] Georgiev, A.B., Sillitti, A. and Succi, G. (2014) Open Source Mobile Virtual Machines: An Energy Assessment of Dalvik vs. ART. *IFIP International Conference on Open Source Systems*, San José, 6-9 May 2014, 93-102. https://doi.org/10.1007/978-3-642-55128-4_12

[13] Dubey, A. and Misra, A. (2013) Android Security Attacks and Defenses. 51.

[14] Nelenkov.blogspot.com (2017) Code Signing in Android's Security Model.

[15] Elgamal, T. and Hickman, K.E. (1997) US Patent No. 5,657,390. US Patent and Trademark Office, Washington DC.

[16] Rahimi, N., Reed, J.J. and Gupta, B. (2018) On the Significance of Cryptography as a Service. *Journal of Information Security*, **9**, 242-256. https://doi.org/10.4236/jis.2018.94017

[17] Cherdantseva, Y. and Hilton, J. (2013) A Reference Model of Information Assurance & Security. 2013 *International Conference on Availability, Reliability and Security*, Regensburg, 2-6 September 2013, 546-555. https://doi.org/10.1109/ARES.2013.72

[18] Security Tips/Android Developers. https://developer.android.com/training/articles/security-tips

[19] Brewis, M. (2017) How to Secure Android: 14 Top Tips for Securing Your Phone or Tablet. PC Advisor. http://www.pcadvisor.co.uk/how-to/google-android/how-secure-android-14-top-tips-for-securing-your-phone-or-tablet-whatsapp-3637549/

[20] Rahimi, N., Sinha, K., Gupta, B., Rahimi, S. and Debnath, N.C. (2016) LDEPTH: A Low Diameter Hierarchical p2p Network Architecture. 14*th International Conference on Industrial Informatics*, Poitiers, 19-21 July 2016, 832-837. https://doi.org/10.1109/INDIN.2016.7819275

[21] Lifehacker.com (2017). http://lifehacker.com/how-secure-is-android-really-1446328680

[22] Android Authority (2017) Best Speed Booster Apps for Android. http://www.androidauthority.com/best-speed-booster-apps-android-108889/

[23] (2017) How to Install Official CyanogenMod 11 ROM on Android One Devices. International Business Times, UK. http://www.ibtimes.co.uk/how-install-official-cyanogenmod-11-rom-android-one-devices-1482622