SECURITY ENHANCED AUTHENTICATION FRAMEWORK FOR SECURING ADB ACCESS IN AOSP

[1]SATHYA SUBBIAH, [2]SRIDHAR GOVARDHAN, [3]SHINU ABHI

[1,2,3]REVA Academy for Corporate Excellence, REVA University, Bangalore, India
E-mail: [1]sathya.cs02@reva.edu.in, [2]sridhargovardhan@race.reva.edu.in, [3]shinuabhi@reva.edu.in

Abstract - Android as an operating system (OS) on the Linux platform is designed initially for mobile and tablet devices. The operating system has evolved over time and expanded its device portfolio to include various other devices including smart TV, video conferencing devices, cars, entertainment systems, etc. Debugging is an important aspect of any software development. Debugging software tracks software information including memory allocation, software state, interface calls, processes, etc. Android Platform provides Android Debug Bridge (ADB) which comes along with Software Development Kit (SDK) that helps the developers with the software development cycle during the product or application development. Android Platform provides an opportunity for the end users to switch to development mode in the device by enabling the developer option. End-users can then use the device to build and test their applications. ADB is a client-server tool that helps to connect to the device/emulator over Universal Serial Bus (USB) or Transmission Control Protocol (TCP). The ADB command enables a range of system behavior, such as downloading and debugging applications, and it gives access to a Unix shell that can be used to execute a range of device commands.
ADB very effectively allows application developers to obtain maximum privileges on Android smart devices, which draws the attention of attackers. A lot of attacks (Droid dream, windows malware, juice filming attack, etc.) are executed using the ADB feature. Malware (Crypto mining malware) is transmitted across devices through open ADB Malware. An affected Android-powered device shall stand as an access point for hackers to launch an attack on other devices connected to the same network. The problem causes the Original Equipment Manufacturers (OEMs) designing such devices to disable the feature in production devices, which challenges the device's needed access postproduction, during the warranty period.
Thus, ADB is a pivotal area to focus on, and this research paper aims at throwing light on an approach that can be pursued which would result in enhanced ADB security.

Keywords - ADB security, Encryption, vulnerabilities

I. INTRODUCTION

The Linux kernel-based operating system, Android OS supports a wide range of devices, including smart phones, tablets, smart TVs, cars, entertainment systems, watches, cameras, smart glasses, and more. Debugging and monitoring tools are provided by the Android SDK, along with other tools required for the development of applications and services. The AOSP platform has been included by several OEMs to create their products. Debugging and bug-finding software is used during the implementation, testing, and maintenance phases of the SDLC to fix problems with other software programmes. ADB, a command-line interface tool for software debugging included in the Android SDK, is routinely utilized across all Android software development.

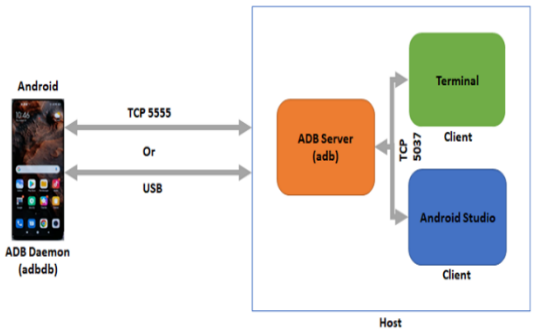

Figure 1 Android Debug Bridge

The ADB tool gives users access to the Unix shell and supports a wide range of commands, including those for installing and removing applications as well as for collecting logs, capturing memory dumps, copying files from devices, and more. As a result, it is quite helpful across the SDLC (Software Development Life Cycle). However, the same capability serves as a convenient entry point for hackers to research the device and insert harmful software into it.
Android is the target of 98 per cent of mobile malware, according to figures that have been released [1]. This forces the equipment manufacturers to disable the feature in the production which affects the access to the system during the manufacturing and the post-production support phase. The aim of this research study is to examine the current implementation and suggest an approach to enhance the security of the ADB feature that could be tailored to the needs of the Original Equipment Manufacturers.

II. ANDROID

Android is a Linux-based operating system that was initially created for touch-screen mobile devices like smartphones and tablets. Starting from version 2.6 of the Linux kernel, core services like security, memory, process management, network stack, and driver models have been migrated. The Android architecture basically consists of five parts [3].

- *Linux Kernel:* The Linux Kernel is a crucial component of Android OS. This provides

essential system capabilities like memory, device, and process management. It serves as a layer of abstraction between the hardware and software stacks.

- *Libraries*: Numerous Java-based and C/C++ core libraries, including Media, Graphics, Surface Manager, SQLite, SSL, OpenGL, etc., are available to help Android development. To make some of these native libraries' functionality available to developers, the Android platform offers Java framework APIs.
- *Android Runtime:* Dalvik Virtual Machine (DVM) and core libraries are core components of the Android run time. DVM is like a Java Virtual Machine (JVM) and was created and optimized specifically for Android. On Android devices, it enables to run many instances of applications efficiently. Threading and memory management is aided by the Linux kernel. The core libraries enable to implementation of Android applications.
- *Application Framework:* The application framework layer is a collection of Java classes and APIs that aid in application development. It is the portfolio of the entire feature set of Android. Various services like activity manager, notification manager, resource manager, etc. are available in the layer. Both system apps and developers use the APIs provided in the layer to develop applications and functionalities.
- *Applications*: The top-most layer of the Android framework is the application layer. Applications that come pre-installed on devices, such as the camera, contacts, gallery, etc. are classified as system apps. Applications created by third parties are installed in this layer when they are downloaded and installed from the Play Store.

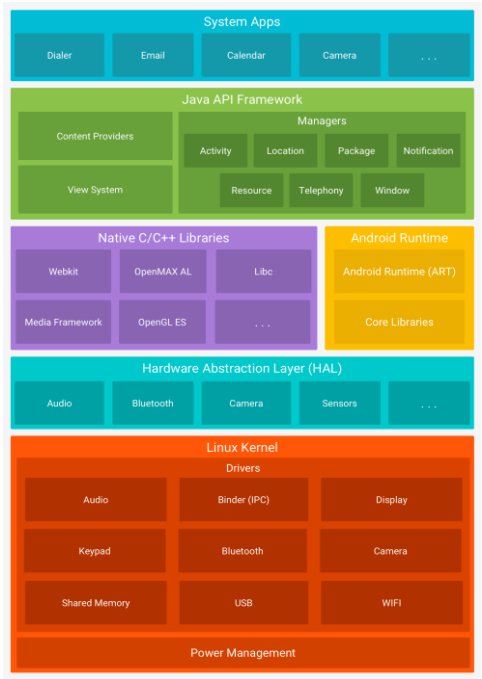

Figure 2 Android Architecture

III. ANDROID DEBUG BRIDGE & AUTHENTICATION

Android Debug Bridge, commonly known as ADB is a tool provided by Google in the Android stack which helps to debug and manage devices [4]. ADB is a command-line tool that allows to communication with Android devices over TCP or UDP protocol.

ADB consists of three components.

- Server runs on the host machine
- Client runs on the host machine
- ADB Daemon runs on Android device

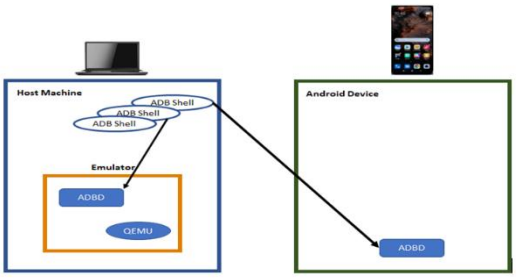

Figure 3 ADB Communication

By design, the Android Open Source Framework has a host authentication mechanism which is enabled by setting the security system property ro.adb.secure in user builds.

The host machine uses the RSA key to authenticate itself to the device's ADB daemon. To accept the key, user approval is needed initially. The user will accept the key for a single session/permanently. In the case of permanent acceptance, the key is stored in memory and can be used for subsequent connections and does not require user interaction, while the key is removed at the end of the session in the case of acceptance for a single session.

The following steps are followed during the ADB authentication process

1. When the host machine tries to connect to the Android device, the device generates a random number (from /dev/urandom) and sends it along with AUTH token message.
2. The host machine generates a hash on the random number and generates a signature over the hashed random number using the private key.
3. The host machine sends the generated SIGNATURE to the device as a response to the AUTH message.
4. If the device has the corresponding public key in the memory, it verifies the signature using the public key stored.
5. In the case of successful verification of the signature, the device responds to the host machine with a connect message and sets the state to ONLINE.
6. In the case of signature verification failure, the device sends an authentication message with the newly generated random number.

7. In the case of first-time authentication, the public key corresponding to the private key will not be present inside the device, and hence always the first authentication fails.
8. In the case of such failure, the host machine sends the public key corresponding to the private key to the device with AUTH RSAPUBLICKEY message.
9. The ADB daemon which runs on the device, generates a MD5 hash on the public key and displays the information to the user for confirmation/acceptance.

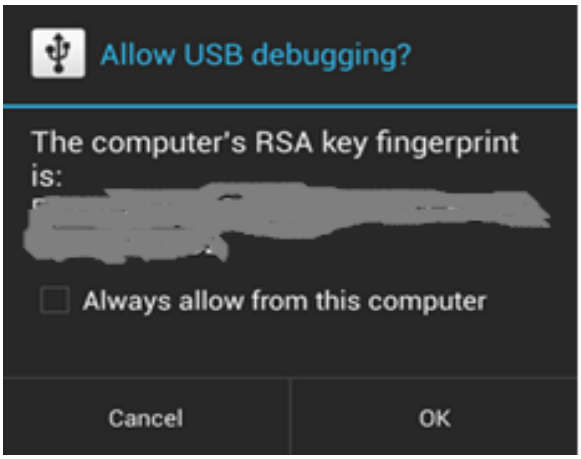

Figure 4 ADB Authorization

10. The public key is used for authenticating ADB session on user acceptance. If the user chooses the option "Always allow from this computer" the public key is permanently added to the device memory.

IV. CAPABILITIES AND CHALLENGES IN ADB

ADB allows to connect to the Unix shell and run a variety of commands. The following actions can be performed using the tool.

- Install applications
- Set up port forwarding
- Copy files to/from the device
- Screen capture
- Issue shell command
- Issue commands to Activity Manager and Package Manager

It is a beneficial tool that is available for use for developers to use both during product and application development [4]. The security of the Android Debug Bridge (ADB) has attracted much attention from researchers because it has a high privilege level and a low level of protection. Many attacks on Android systems have taken advantage of the security holes of ADB.

The features supported by the tool are used by hackers for performing the below.

- Get a list of installed packages
- Obtain device information
- Copy the malicious file to the device
- Copy SD card content from the device
- Install application on the device
- Unlock the device
- Root the device
- Obtain information from the device such as contact list, messages, etc.

Android Rooting allows the user to get root access to the operating system. The root access allows the user to modify the source code of the device and enables the user of the device to install unauthorized software/applications to the device [5] [6]. Gaining root access enables to overcome the security restrictions placed by the operating system and thus malwares like spyware, worms, viruses, etc. can affect the rooted android device.

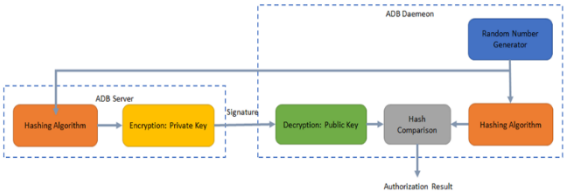

Figure 5 Authentication Flow in ADB

The existing security offered by the ADB feature of Android involves the authentication of the host using the RSA public and private keys. If the host machine is attacked, extracting the keys, and turning on the ADB on the Android device in TCP mode is quite an easy task to perform. The extracted keys can be used to connect from the attacker's PC remotely to the Android device and launch an attack. One author proposes a whitelist approach with a log and an alert feature as solution to handle the security flaw found in the ADB feature [7].

V. ATTACK SIMULATION USING ANDROID DEBUG BRIDGE

Shodan search engine and Ghost framework (an Android post-exploitation framework) are both used to simulate attacks leveraging the ADB. With the help of numerous filters, Shodan's search engine enables users to look for different kinds of servers that are online. We utilized Android Debug Bridge as the search term to look for Android devices that had the ADB option enabled. To understand the pattern of devices with ADB enabled, numerous searches are carried out on different days. It is discovered that, on average, 10K devices have ADB enabled and are accessible over the internet.

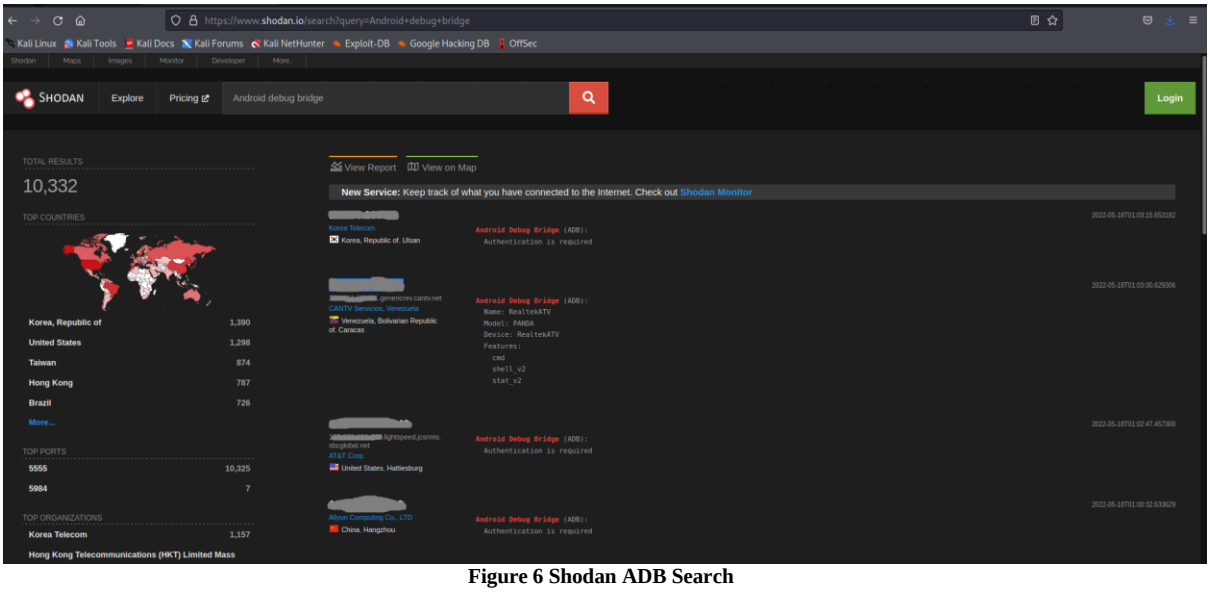

Figure 6 Shodan ADB Search

Ghost is an Android exploit framework which can be installed in Kali Linux for simulating the attack. Remote connection to the ADB-enabled device is enabled using the Ghost tool. The selected IP address is connected using the ghost tool.

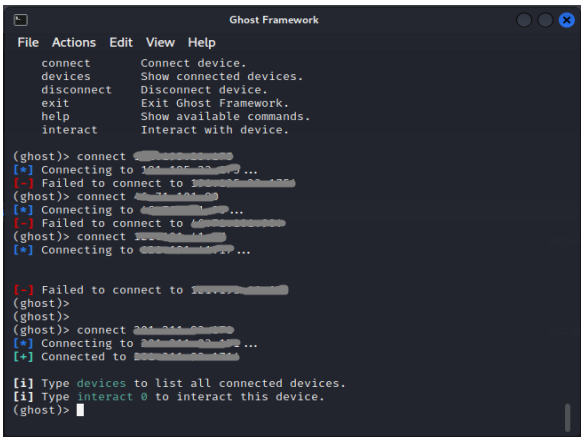

Figure 7 ADB Connection using Ghost Tool

Once the device is connected, shell access is enabled using the tool. Access to the file system of the Android device, the possibility to install applications/scripts, and access and control of the remote device is possible once the connection and shell access is enabled.

Figure 8 Shell Access Using Ghost Tool

Through this simulation, we can understand that on top of the available security in ADB, we are still able to access Android devices remotely.

VI. PROPOSED SOLUTION

The solution proposed as part of this research paper is focused on strengthening the authentication of hosts by Android devices based on digital certificates. The digital certificate-based approach makes the OEM customize, and time limit the ADB session thus improving the security and reducing the attack surface.

As part of the solutions, digital certificates generated by Public Key Infrastructure (PKI) shall be used for authenticating the ADB client (Host Machine) to the ADM Daemon (Android Device).

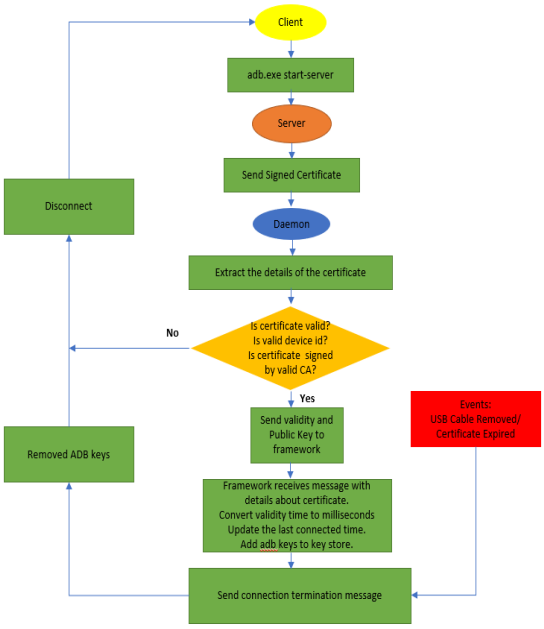

Figure 9 Certificate Enabled ADB Authentication

Digital Certificate: A digital certificate is a file that establishes the legitimacy of a device, server, or user utilizing cryptography and the public key infrastructure (PKI). The digital certificates are used to prove identity and share public keys.

In the case of the proposed enhanced ADB security model, the host server shall send digital certificates to the daemon to prove identity and share the public key. The certificate shall also share other information including validity, purpose, issued to (Example: Engineer name) etc.

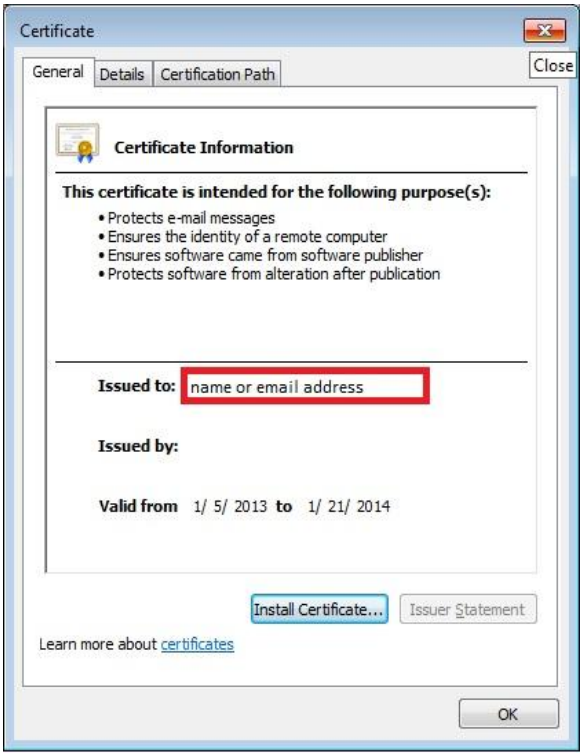

Figure 10 Digital Certificate

Public Key Infrastructure (PKI): PKI is a framework which consists of encryption keys, digital certificates, Certificate Authority (CA) and Registration Authority (RA). PKI makes sure that the issued digital certificate can be trusted.

OEMs have their own PKI which generates encryption keys, certificates and digital signatures which are used for a firmware update or any other authorized communication to the devices produced by OEM.

As part of this proposed implementation, digital certificates shall be issued by PKI for debugging the Android device and shall be used for authenticating the host machine to the Android device.

Figure 11 Public Key Infrastructure (PKI)

Certificate Chain: A certificate chain is an ordered list of certificates. It comprises leaf certificates, intermediate certificates, and root certificates. The root certificate is a self-signed certificate and is on the top of the certificate chain. The root certificate issues intermediate certificates and verifies the digital signature of the intermediate certificates. The intermediate CAs issue the leaf certificates and verifies the leaf certificates' digital signature. Thus, the certificate path is verified from the leaf certificate to the root certificate.

In the proposed solution, the certificates passed to the Android device, shall be verified by the certificate chain stored in the device.

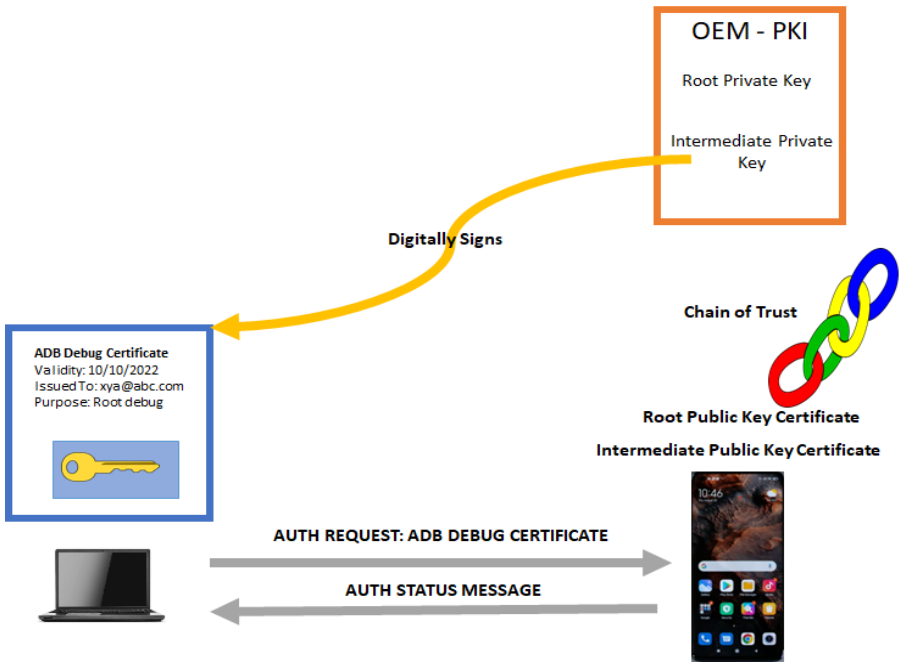

Figure 12 Certificate-Based ADB Authentication

VII. CONCLUSION

Connected devices are the future of the existing global market. We have devices running on various platforms with Android being the most popular Operating System with more than 40% of the market share [7]. ADB tool is the only debug tool provided by the Android platform to develop and debug Android devices. Securing authorization of ADB tool access enables the OEMs to safely enable the ADB bridge in devices with User builds.

REFERENCES

[1] "Cyber Security Statistics," PURPLESEC, 2022. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/.

[2] "Android Platform Architecture," Google Inc, [Online]. Available: https://developer.android.com/guide/platform.

[3] "Android Debug Bridge," Google Inc, [Online]. Available: https://developer.android.com/studio/command-line/adb.

[4] S. Hazarika, "How to Root your Android smartphone," 15 4 2022. [Online]. Available: https://www.xda-developers.com/root/.

[5] L. C. N. T. K. S. J. S. Nguyen-Vu, "Android Rooting: An Arms Race between Evasion and Detection," Hindawi, vol. 2017, no. 4121765 , p. 13, 2017.

[6] M. Mingzhe Xu and Weiqing Sun and Alam, "Security enhancement of secure USB debugging in Android system," in 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2015, pp. 134-139.

[7] "Operating System Market Share Worldwide | Statcounter Global Stats," [Online]. Available: https://gs.statcounter.com/os-market-share.

[8] A. J. E. J. Gonzalez Villan, "Remote Control of Mobile Devices in Android Platform," ResearchGate, no. arxiv.org/abs/1310.5850, 2013.

[9] S. a. M. R. Höbarth, "A framework for on-device privilege escalation exploit execution on Android," ResearchGate, 2011.

[10] T. Kobayashi, ADB (Android Debug Bridge): How it works?., Android Builders Summit, 2012.

[11] R. Regupathy, "Android Debug Bridge (ADB)," in Unboxing Android USB: A Hands-On Approach with Real World Examples, Berkeley, CA, Apress, 2014, pp. 125--138.

★★★