Title

ACCESSING SMARTPHONE DEVICE USING ANDROID DEBUG BRIDGE (ADB) INTERFACE

BACKGROUND

[0001] Android Debug Bridge (ADB) is a tool that is provided along with the Android framework to facilitate debugging and managing of an Android system. To use ADB, a user typically needs access or login credentials to the device in order to select a setting "Allow USB debugging". However, in some instances, the user may need to use ADB on a device without having access or login credentials to the device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The embodiments of the invention are illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment of the invention in this disclosure are not necessarily to the same embodiment, and they mean at least one. In the drawings:

[0003] Figure 1 is a representational view illustrating an architecture including a client, a server and one or more daemons in which the present invention can be implemented, according to an example embodiment.

[0004] Figure 2 is a representational view illustrating a sequence diagram showing a process for accessing one or more smartphone devices using Android Debug Bridge (ADB) interface, according to an example embodiment.

DETAILED DESCRIPTION

[0005] In the following description, numerous specific details are set forth. However, it is understood that embodiments of the invention may be practiced without these specific details. In other instances, well-known circuits, structures, and techniques have not been shown to avoid obscuring the understanding of this description.

[0006] Android Debug Bridge (ADB) works in a simple client/server architecture, and is made up of three key components, namely, a server such as ADB Server 111, a client such as terminal 112 and studio 113 of Figure 1, and a daemon such as ADB Daemon 102a and ADB Daemon 102b, as shown in architecture 100 of Figure 1.

[0007] The ADB Server 111 runs in the background of the host system or development machine 101 and communicates between the client (112 and 113) and the ADB daemon(s) (102a and 102b) running on an emulator or device. The ADB server 111 also maintains details of the connected device along with its state.

[0008] The client issues commands and can be a terminal 112 or command prompt or a Studio 113 such as Android Studio.

[0009] The ADB daemons 102a and 102b run on the Android devices/emulators as part of the Android USB framework and interact with the ADB server 111 to help manage the Android-powered device. The ADB daemons 102a and 102b enable the ADB server 111 to execute commands on the smartphone device(s).

[0010] ADB uses USB 121 or TCP 122 as its transport layer to communicate with an Android-powered device. The USB 121 can be any USB cable such as standard USB C or microUSB. As shown in Figure 1, the host system 101 can communicate with multiple devices simultaneously. Two devices are shown in Figure 1; however, in other embodiments, one device or more than two devices can communicate with the host system 101 simultaneously.

[0011] Figure 2 is a representational view illustrating a sequence diagram showing a process for accessing one or more smartphone devices (e.g., devices running daemons 102a or 102b) using an ADB interface. As shown in Figure 2, a command of adb kill-server is first sent from the client to the server. This command resets the host system 101.

[0012] As further shown in Figure 2, the client sends an adb start-server command to the server. This starts the server, which will trigger the RSA pop up on the screen of the device as described below.

[0013] The devices are plugged in using the USB cable 121, and the ADB server 111 connects with the daemons 102a and 102b. When the devices are plugged in, the ADB server presents itself as an installer. It can take around 5 seconds for the device(s) to switch into 04e8:6860 if the screen is not interacted with; if the software install on the phone is interacted with, the device(s) will not switch into 04e8:6860. 04e8:6860 is the USB ID shown when the smartphone device is connected in its default state (i.e., with “USB debugging” mode turned off).

[0014] The client then sends a AT+SYSSCOPE=1,0 command to the server which communicates the command to the daemon(s). This sysscope check determines if the device is in the following states: a) scanning, booting and operating system loading; b) normal, the device has not been modified/rooted/or tampered with and has been fully booted into the operating system; or c) modified - the device has been rooted/modified/tampered, and the device security has been compromised.

[0015] In order to move forward, a response message to the AT+SYSSCOPE=1,0 command should be +SYSSCOPE:1,NORMAL OK.

[0016] If the response message is +SYSSCOPE:1,NORMAL OK, then the client sends an AT+USBMODEM command to the server which communicates the command to the daemon(s). The AT+USBMODEM command enables the USB Debug Modem, and triggers a RSA authorization pop up. Again, for the RSA authorization pop up to be triggered, the +SYSSCOPE:1,NORMAL OK response should have been received.

[0017] It is noted that the commands are sent using a serial connection to the device and access the Qualcomm chipset. The ADB server should be running in the background in order for the RSA pop up to appear.

[0018] As shown in Figure 2, the client sends a AT+CTSA=2,170,1209 command to the server which communicates this command to the daemon(s). This command acts as a virtual keyboard, and performs a user press on the device screen that will tap the part of the RSA Dialog that says "Always allow from the computer". In the AT+CTSA=2,170,1209 command, 2,170,1209 are example coordinates on the device screen. These coordinates can be changed to match the device screen size. The range of the coordinates can be as follows: AT+CTSA=[1-4],[1-1500],[1-3000].

[0019] As further shown in Figure 2, the client sends a AT+CTSA=2,880,1420 command to the server which communicates this command to the daemon(s). This command acts as a virtual keyboard, and performs a user press on the device screen that will tap "OK" on the Allow USB Debugging Dialog box on the device screen. This command can be used alone, and does not have to be used in conjunction with the AT+CTSA=2,170,1209 command. In the AT+CTSA=2,880,1420 command, 2,880,1420 are example coordinates on the device screen. These coordinates can be changed to match the device screen size. The range of the coordinates can be as follows: AT+CTSA=[1-4],[1-1500],[1-3000].

[0020] By virtue of the foregoing disclosed arrangement, ADB (Android Debug Bridge) can be enabled without any device modifications or installations. It is also possible to auto authorize RSA security without user interaction, gaining full authorized ADB access to the device. The foregoing disclosed arrangement can bypass the device setup, and go to the home screen. Pin Locks, Pattern Locks, Alpha Numeric Passcodes, Finger Print Locks, Retina Scan Locks, or Mobile Device Management can be bypassed in this process and all locks can be removed if any are detected. Installation of any APK on multiple devices can be performed silently in the background, without detection. APKs, binaries, or any other application can be installed fully authorized. Full Data Clearing of the device can be performed in less than 3 seconds. This can be run against DoD standards and can be fully validated for data sanitization in less than 5 seconds after reboot. This can be done without ADB enabled, and without any user authorization. Passcodes, locks, and MDM are not a factor and device clearing can still proceed. If the device has been locked, the foregoing described arrangement will bypass the firmware flashing lock as well. The foregoing described arrangement has been tested and validated to work for up to 200 devices at a time.

[0021] An embodiment of the invention may be a machine-readable medium having stored thereon instructions which program a processor to perform some or all of the operations described above. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer), such as Compact Disc Read-Only Memory (CD-ROMs), Read-Only Memory (ROMs), Random Access Memory (RAM), and Erasable Programmable Read-Only Memory (EPROM). In other embodiments, some of these operations might be performed by specific hardware components that contain hardwired logic. Those operations might alternatively be performed by any combination of programmable computer components and fixed hardware circuit components. In one embodiment, the machine-readable medium includes instructions stored thereon, which when executed by a processor, causes the processor to perform the method on an electronic device such as a host system 101 as described above.

[0022] In the description, certain terminology is used to describe features of the invention. For example, in certain situations, the terms “component,” “unit,” “module,” and “logic” are representative of hardware and/or software configured to perform one or more functions. For instance, examples of “hardware” include, but are not limited or restricted to an integrated circuit such as a processor (e.g., a digital signal processor, microprocessor, application specific integrated circuit, a micro-controller, etc.). Of course, the hardware may be alternatively implemented as a finite state machine or even combinatorial logic. An example of “software” includes executable code in the form of an application, an applet, a routine or even a series of instructions. The software may be stored in any type of machine-readable medium. While the invention has been described in terms of several embodiments, those of ordinary skill in the art will recognize that the invention is not limited to the embodiments described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting. There are numerous other variations to different aspects of the invention described above, which in the interest of conciseness have not been provided in detail. Accordingly, other embodiments are within the scope of the claims.