Automated Android Penetration Testing Toolkit (AAPTT) Manual.

Contents

[Setting up the Toolkit 2](#_Toc133262721)

[Setting up a Linux-derivative Virtual Machine (VM) 2](#_Toc133262722)

[Adding the Toolkit to the Virtual Machine 3](#_Toc133262723)

[Installing Toolkit dependencies 3](#_Toc133262724)

[Using the Toolkit 4](#_Toc133262725)

[Connecting to the Network & Finding the Target IP Address 4](#_Toc133262726)

[Launching the Toolkit & Understanding the Process 5](#_Toc133262727)

[Using Tools That Require User Intervention 5](#_Toc133262728)

[Viewing & Understanding Toolkit Results 9](#_Toc133262729)

***DISCLAIMER:*** ALL TEXT IN **BOLD** ARE THE BASIC STEPS THAT NOVICE OR NEW USERS SHOULD FOLLOW TO BE ABLE TO USE THE TOOLKIT CORRECTLY.

# Setting up the Toolkit

The first step in using this Android penetration testing toolkit is to set up the environment for the toolkit to execute within. The toolkit has been designed to execute commands upon the terminal of a Linux-derivative Operating System. This means that the user must be using either a PC running a Linux-derived Operating System, like Kali Linux or Ubuntu, or they must have access to a Virtual Machine running one of these Operating Systems.

## Setting up a Linux-derivative Virtual Machine (VM)

For users that do not currently have access to a physical device that is running a Linux-derivative Operating System, their only option for running this toolkit is to create or launch a VM running a Linux-derivative Operating System to launch the toolkit script from. The simplest way to do this would be to install a Virtual Machine workstation program and import the pre-prepared Virtual Machine image of the testing and development environment used to create the toolkit. This is the simplest way as the VM used during testing and development already has the required tools, packages, and the toolkit script installed. Installation of this VM requires a slightly different process including using the **“Import”** option rather than “New” and selecting the **“AAPTT.ova”** file supplied. The username & password for this VM are **“kali” & “kali”.**

If the user decides not to use the pre-prepared VM image and chooses to create their own VM with a Linux-derived Operating System, the user will still need to **choose a Virtual Machine Workstation program, download, and install the workstation**, before **downloading an Operating System** from official websites to setup the virtual device. The user has many different workstations and Operating Systems to choose from, all of which are free to download and use, many of which are listed below. The recommended combination for running this toolkit is to use Oracle VirtualBox to run a Kali Linux VM but any combination of VM workstation and Linux-derived Operating System should work as intended.

Table - Virtual Machine Workstation Programs.

|  |  |
| --- | --- |
| ***Workstation Program Name:*** | ***Official Link:*** |
| VMWare Workstation Player | <https://www.vmware.com/uk/products/workstation-player/workstation-player-evaluation.html> |
| Oracle VirtualBox | <https://www.virtualbox.org/wiki/Downloads> |
| Microsoft Hyper-V | WINDOWS ONLY – Search > “Windows Features” > Select “Hyper-V” > Select “OK” > Restart Device. |

Table 2 - Linux-derivative Operating Systems.

|  |  |
| --- | --- |
| ***Operating System Name:*** | ***Official Link:*** |
| Kali Linux | <https://www.kali.org/get-kali/#kali-virtual-machines> |
| Ubuntu | <https://ubuntu.com/download/desktop> |
| Debian | <https://www.debian.org/distrib/netinst> |
| Fedora | <https://getfedora.org/en/workstation/> |

Creating a new Virtual Machine is usually a simple task but steps can vary drastically between VM workstations. Each of the suggested workstations provides a basic VM setup mode that allows new/inexperienced users to create a VM with the user only required to manually choose their downloaded Operating System image while other VM settings are automatically detected and set on the user’s behalf. The general path a user needs to take when creating a new VM is to select the **“New VM”** option and select or stick to the **“Basic” or “Simple” VM creation mode**. This mode then automatically sets the necessary display, memory, and processing settings before asking the user for an **ISO Operating System image file** which they should have already downloaded to install the desired Operating System onto the VM. Finally, the user can select the **“Finish” or “Done”** button to create the VM before selecting **“Power On” or “Start”** to start the VM.

## Adding the Toolkit to the Virtual Machine

To add the Toolkit script into a new VM, the user must **activate “Guest Additions”** in Oracle VirtualBox, via opening the VM, clicking the **“devices”** tab, and **inserting the guest additions cd** image, or **“VMWare Tools”** in VMWare Workstation Player. This will allow the user to either copy the script from their PC to the VM using the drag-n-drop (dragging the file into the VM to copy the file over) or copy and paste (copying the file on the PC and moving to the VM where they select to paste the file) method, or to copy the script to a shared folder between their PC and VM.

## Installing Toolkit dependencies

Once the script has been added to the Virtual Machine, the tools used by the script, also known as the “dependencies”, must be installed on the VM so that the script can be executed successfully. Some of these dependencies include Nmap, enum4linux, Andriller and more. Many different tools and packages are used by this toolkit, but some need not be installed if using the Kali Linux Operating System. This is due to Kali Linux featuring many of the most popular penetration testing tools already included within all installations of the Operating System. This means that Kali Linux users can skip the steps needed to install the Nmap and Enum4linux packages.

Before attempting to install any of these tools, the user should type the command ***“sudo apt-get update”***,before pressing enter, into the **“terminal emulator”** to ensure software packages on the VM are up to date. The user should enter the following commands into a terminal on their Linux-derivative machine answering **“yes” or “y”** to any questions asked by the terminal during the installation process. The “sudo” keyword allows the command to run with root/administrator privileges, the user may be asked to re-enter their password to continue the process.

Table 3 - Installing tools used by the toolkit.

|  |  |
| --- | --- |
| ***Tool Name:*** | ***Terminal Command(s):*** |
| Nmap | ***“sudo apt-get install nmap”*** – Install Nmap.  ***“nmap --version”*** – Verify installation. |
| Enum4Linux | ***“sudo apt-get install enum4linux”*** – Install Enum4Linux.  ***“enum4linux -h”*** – Verify installation, returns instructions to use tool. |
| Python3 | ***“sudo apt-get install python3”*** – If not installed, install Python interpreter.  ***“sudo apt-get install python3-pip”*** – Install Python3 package manager.  ***“sudo apt-get install python3-tk”*** – Install tkinter (for ALEAPP).  ***“python3 –version”*** – Verify installation. |
| ADB | ***“sudo apt-get install android-tools-adb”*** – Install ADB. |
| ALEAPP | ***“sudo apt-get install git”*** – Install git.  ***“sudo git clone https://github.com/abrignoni/ALEAPP.git”*** – Clone ALEAPP.  ***“sudo pip3 install -r ALEAPP/requirements.txt”*** – Install ALEAPP modules. |
| Andriller | ***“sudo pip3 install andriller -U”*** – Install Andriller. |
| Ettercap | ***“sudo apt-get install Ettercap-graphical”*** – Install Ettercap. |
| Evil-Droid | ***“sudo git clone https://github.com/M4sc3r4n0/Evil-Droid.git”*** – Clone tool.  ***“chmod +x Evil-Droid/evil-droid”*** – Give the script execution permissions. |

# Using the Toolkit

With all the toolkit’s dependencies installed, the toolkit can now be used to find any weaknesses/vulnerabilities in their Android mobile device.

## Connecting to the Network & Finding the Target IP Address

Before running this toolkit, the user should make sure that **the device that they are to run the toolkit from, and their Android target device are both on the same Wi-Fi network** so that the toolkit can reach and communicate with the target device. For example, the developer tested this toolkit using a desktop PC and Android smartphone both connected to a Wi-Fi network named “TP-Link\_7172”. After connecting to the same Wi-Fi network on each device, the user should then make sure that they are aware of, and may even wish to **take note of, their Android target device’s IP address**. It is important to do so as the toolkit will first ask the user for the IP address of the target device.

Figure - Finding the IP address of the Android target device.

Text

Description automatically generated

***DISCLAIMER:*** THE USER OF THIS TOOLKIT SHOULD ONLY TEST DEVICES THAT BELONG TO THEM OR THAT THEY HAVE EXPLICIT PERMISSION TO TEST SO AS NOT TO BREAK THE COMPUTER MISUSE ACT 1990, THE USER SHOULD MAKE EVERY EFFORT TO ENSURE THE IP ADDRESS THEY ARE PASSING INTO THE TOOLKIT IS ACCURATE TO THE DEVICE THEY ARE TESTING.

## Launching the Toolkit & Understanding the Process

With a note of the IP address now taken, the user can execute the script by opening a terminal, the “terminal emulator”, in the same directory as the toolkit script and typing the command ***“sudo python3 aaptt.py”***, before pressing enter. A terminal can be opened in a specific directory by opening the desired folder using the file explorer, **right clicking**, and clicking **“Open Terminal Here”.** The user will then be prompted to enter the IP address of the target device. If they are using the pre-prepared VM, they will be asked to enter the password **“kali”.** After the user has entered the IP address of their target device, the toolkit will carry out a quick ‘ping’ scan to determine whether there is an active and reachable device at the given IP address. If there is an active device at the given address, the toolkit will begin to automatically scan the target device, if not, an error message will be displayed to inform the user.

The next step taken by the toolkit is to probe the target device for access to a shell and to the filesystem using the Android Debug Bridge (ADB) through port 5037, formerly port 5555. The toolkit does this in stages by starting the server, setting up wireless TCP/IP connectivity on port 5037, before running a shell command to copy the Android filesystem. This copy of the filesystem can then be zipped and fed into ALEAPP’s Command Line Interface (CLI) which produces a folder including discovered files and a report of the findings. If access to the filesystem is denied by the target device, the ALEAPP report and folder of findings will all appear empty.

## Using Tools That Require User Intervention

After this, other exploits that cannot be carried out programmatically, and that will require user intervention, will be executed starting with an ADB shell on the target device. However, the device may not be vulnerable to this attack and produce an error saying the connection failed. If this occurs skip to the next paragraph. Since the Android Operating System is a Linux-derivative, the ADB shell uses many of the same commands to navigate through the terminal as the device running the toolkit like **‘cd’, ‘ls’**, and more, whilst also including some Android-specific commands. A table of some commands the user may find interesting to use, but are not necessary for testing, can be found below. Each of these commands are safe and are only to provide the user proof of certain dangers.

Table 4 - Android Debug Bridge Shell Commands.

|  |  |
| --- | --- |
| ***Command:*** | ***Description:*** |
| ***“am start -W -c android.intent.category.HOME -a android.intent.action.MAIN”*** | Emulates pressing the home button, returns the device to the home screen. |
| ***“pm grant [packageName] [Permission]”*** | Grant a chosen permission (“[permission]”) to the user’s chosen app (“[package]”). |
| ***“screenrecord --verbose”*** | Records the screen of the Android target device, use “Ctrl+C” to stop recording. |
| ***“am start android.intent.action.CALL -d tel:[phoneNumber]”*** | Start a new activity, the “CALL” activity starts a new call to the chosen telephone number (“[phoneNumber]”). |
| ***“exit”*** | Closes the ADB shell, allows the toolkit to proceed with closing ADB and launching other tools. |

After closing the ADB shell by typing **“exit”**, the toolkit can then disconnect from the target device and shutdown the ADB server. This is then followed by opening Andriller if the prior ADB shell attempt to extract Android filesystem data failed. Andriller uses a Graphical User Interface (GUI) to extract files from an Android device. Andriller requires the target device to be connected to the host via a USB cable and for the user to allow the backup when the request pops up on the screen of the target device. Once this has been allowed, Andriller uses the Android backup function to copy data from the filesystem and can carve out data like images, videos, documents, etc. The user should begin by clicking **“Output”** and selecting a folder to store any discovered information in. The user can then click **“Check”** to ensure their Android target device is recognised by the tool, the device is recognised if a serial number is displayed next to the button, and **“Extract”** to begin to pull data from the target device. This tool creates a report of its findings, which is opened automatically, to search through the data captured and creates archives that can be passed into ALEAPP for further investigation. With this, Andriller can now be closed by clicking the **“x”** in the top-right corner.

Figure - Andriller GUI, Extraction Tab.

Graphical user interface, text, application

Description automatically generated

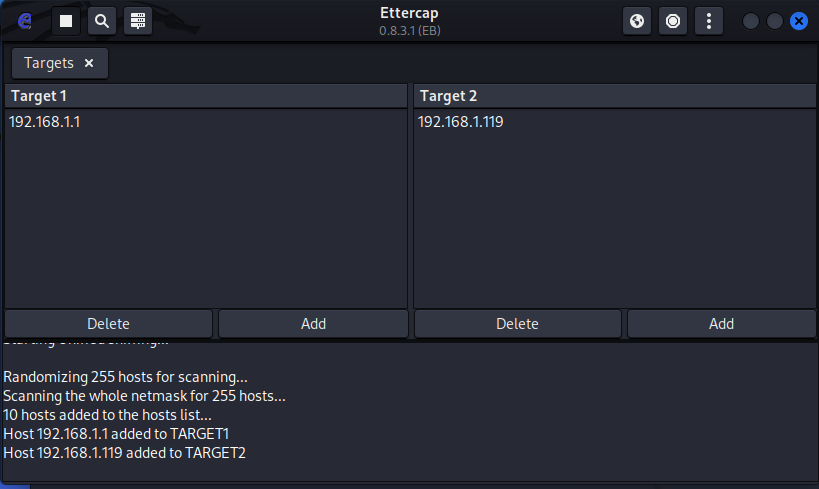
The next tool launched is the Ettercap tool that can gather information from the target device and the network’s router using Address Resolution Protocol (ARP) poisoning. This tool requires that a VM running Ettercap uses a “bridged” network adapter. This can be changed from the VM’s settings page and changing the network adapter type. Once the tool has launched, the user can proceed with the default settings, **click the tick**, before searching the network for hosts. They can do this by **clicking the magnifying glass** icon to fill the host list with targets. The **host list** icon can be found to the right of the magnifying glass and should be filled with IP addresses. The user should select the address of the **network’s router as target 1** and the **address of the Android target as target 2**. From here, the user can navigate through the **man-in-the-middle (MITM) attack menu** and through the **“view”** menu to perform attacks like ARP spoofing or view connections between the targets with the option to even kill some of these connections, respectively. The user should always refer to the toolkit output if certain tools don’t work as this may be a sign that the device is not vulnerable to these attacks/tests.

Figure - Ettercap GUI, startup settings can be left as default.

Graphical user interface

Description automatically generated

Figure - Ettercap GUI, main screen to select targets and attacks.



The final tool run by the toolkit is Evil-Droid which injects a backdoor script into a legitimate Android Package Kit (APK) file, the file responsible to install an Android app. It can then create a website that an Android user can navigate to and download the compromised APK before opening a reverse shell listener that will wait for the malicious app to be launched. No dependencies need to be installed before using the tool as these will all be automatically verified, and downloaded as the tool is launched before producing the menu.

There are many options in this menu, the first option, APK MSF, can be used to create a simple Proof-of-Concept (PoC) backdoor APK as it will not take a real APK file from the user and instead creates an APK with the sole purpose of producing a backdoor when executed. The next 2 options will take a real APK file of the user’s choice and install a backdoor within. The user should select option 1 by **typing “1” and hitting enter**. The tool will then ask for an IP address and port, both of which are automatically filled by the tool, and a name for the new backdoor APK. The user can **click “OK” for each** as no changes should be required. Finally, the user should **select the “android/meterpreter/reverse\_tcp” option and click “OK”**. Once these have been entered and the APK has been created, the tool will ask the user which type of attack they wish to use. The **“Attack-Vector”** option asks for a **website URL** to be cloned and a **page name** before producing a webpage that can allow the Android target device to download the APK.

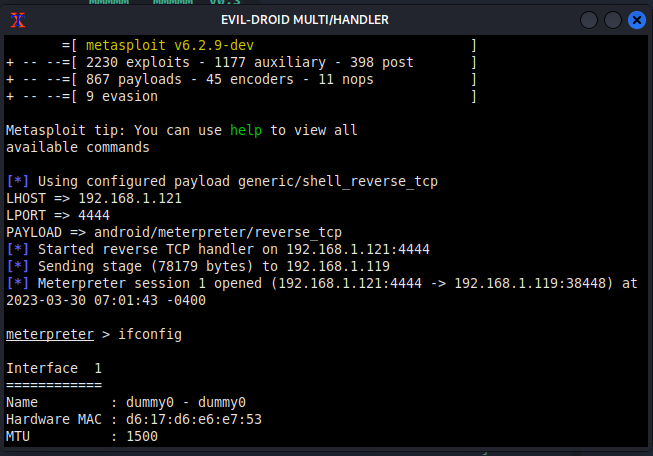
Figure - Evil-Droid menu.

Text

Description automatically generated

Once the APK has been created and an attack option has been chosen, the user can **navigate to the cloned webpage** on the Android device using the link displayed on screen. Once they open this page, the user can **download and install** the backdoor application whilst also being able to see the webpage they chose to clone on-screen. After installing and opening the malicious app on the Android target device, the user should see an update on their Evil-Droid Multi/Handler screen showing a new meterpreter session has opened and they can now enter commands. Some common, safe commands the user may wish to play around with include **“ifconfig”** (this shows the IP address of the remote device), **“sysinfo”** (displays system information), and **“app\_list”** (list the app packages installed on the target device). Finally, the user should **type “exit” and hit enter twice** to return to the Evil-Droid menu where they will be asked if they wish to return to the main menu or exit the tool. At this point the user is free to **exit** the Evil-Droid tool and finish using the toolkit or try out Evil-Droid’s other options.

Figure - Evil-Droid, meterpreter reverse TCP shell after launching "evilapk" on Android target device.



## Viewing & Understanding Toolkit Results

Each command executed by the toolkit will print its output both to the terminal and a file in the directory where the toolkit script is stored called **“aapttResults.txt”.** This allows the user to view more detailed information from the toolkit as it is produced on the terminal, which may be a little more confusing to some more novice users, or in a much more user-friendly and structured text file.

The text file generated will also include some recommendations to mitigate any uncovered vulnerabilities and threats alongside instructions on how to implement these to help the user improve their security after discovering issues. Reasons as to why some tools or exploits could not be carried out successfully will also be included in this file.