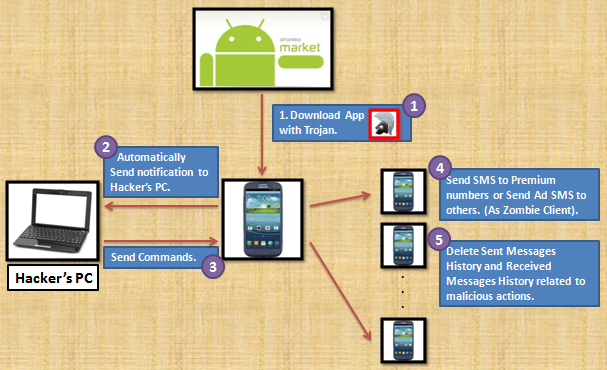
SMS Trojan Malware

*Abstract*—In this lab, we will develop an Android Trojan from scratch to demonstrate the concept of Mobile Malware on Android platform. The main functionality of this Android Trojan is sending text messages to others according to a hacker's commands without knowledge of the phone user. The diagram below illustrates the work flow of this Android Trojan.

[](https://sites.google.com/site/mobilesecuritylabware/4-mobile-malware/malware_lab_activities/lab-1-mobile-malware-attack-trojan/Trojan%20Diagram.png?attredirects=0)

# Introduction

An attack of mobile malware usually involves three phases: the infection of a host, accomplishments of its goal, and spread of the attack. It should be noted that not all the mobile malware conduct the third phase of attack that is spreading the attack.

The infection phase refers to the infection of a malware into a target device. Users may be infected when downloading a malicious email attachment or visiting a phishing website. Peer-to-Peer sharing applications, shared links on mobile social networking can also bring malware into your phone.

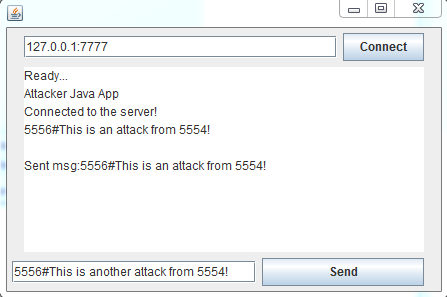
Trojan is a popular malware that steals confidential information such as credit card information. In this lab, the Trojan and the hacker are able to set up a TCP/IP communication channel, via which the hacker can send commands to the Trojan on victim's device. When receive a command from the hacker, the Trojan will analyze the data packet, extract the target user’s phone numbers and the content of the malicious message, and finally send the malicious messages to the target user.

For example, you have 2 Android virtual devices running: phone1 with ID 5554 and phone2 with ID 5556. Phone1 is infected with the Trojan malware, while Phone2 is a regular phone not infected by it. The Trojan on phone1 sets up a ServerSocket to listen on an unused port, say 7777. The Hacker’s PC (Attacker App) opens a TCP socket connection to phone1 (Emulator 5554 infested with Trojan) without the user’s knowledge, then sends a message through the socket connection to phone1 with the content “5556#This is an attack from 5554”. The Trojan on phone1 extracts the target phone2’s ID of 5556 and message string “This is an attack from 5554”, and sends a SMS to phone2 with content “This is an attack from 5554”. Please note that you need to set the MalwareSMS app as the Default SMS App, similar to Lab 6.

在手机A （5554） 中植入木马程序，然后通过计算机端连接手机A 中木马程序的服务器端口，发送相关信息，手机A 的木马程序当收到信息的时候，会进行过滤，如果信息不是一开始设定的“攻击”短信（包含attack string “#”），那么就忽略此短信而不做任何操作；如果是“攻击”短信，就给对应攻击对象(attack string “#”前面的那个号码，例如5556）发送骚扰短信。

I implemented a simple Java application (not Android app) as the attacker program, with the following UI. The attacker first establishes a TCP/IP connection to phone1 (localhost, port 7777) by clicking the “Connect” Button, then sends a message to it by clicking the “Send” button. (For Java socket programming:

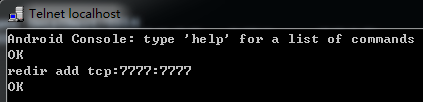
<http://examples.javacodegeeks.com/android/core/socket-core/android-socket-example/>)



Ideally, the attacker application should run on a remote host and connect to the host running Emulator #5554. But for convenience, we run the attacker app on the same machine as the Emulator #5554. In order to allow attacker’s program on the host to establish a TCP/IP connection to the Android virtual device phone1 on port 7777, you need to use telnet to connect to phone1 and run “redir add tcp:7777:7777”, which redirects all connections to host port 7777 to the virtual device port 7777. First enable telnet by Control Panel-打开或关闭Windows功能, check “telnet客户端“：

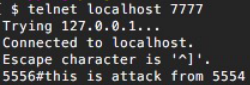


Win-R, telnet to bring up the telnet window. Keep the virtual device Phone1 running, enter “open localhost:5554” to connect to it. Then enter

“redir add tcp:7777:7777”:

This means the PC’s local port 7777 is now connected to the simulator’s port 7777. You can use “redir list” to see the connected ports. Now the attacker Java app can connect to Phone1 at localhost:7777, and instruct phone1 to send attack SMS to Phone2.

You can also choose to use telnet to connect to Phone1 5554 instead of the Java app.



You need to implement one function startServer() in MalwareSMS.java

MalwareSMS.java:

|  |
| --- |
| **public** **class** MainActivity **extends** Activity {  **private** ServerSocket serverSocket;  **public** **static** **final** **int** *SERVERPORT* = 7777;  Thread serverThread = **null**;  @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  **this**.serverThread = **new** Thread(**new** ServerThread());  **this**.serverThread.start();    IntentFilter filter = **new** IntentFilter("android.provider.Telphony.SMS\_RECEIVED");  SMSReceiver receiver = **new** SMSReceiver();  registerReceiver(receiver,filter);  }    **class** ServerThread **implements** Runnable {  **public** **void** run() {  startServer();  }  }  **public** **void** startServer() {  //TODO: Create a socket and wait for incoming attacker connection. Upon receiving a message from the attacker, parse it and try to find the “#” pattern. If found, send attack SMS to phone2 with sendSMS(); If not, simply ignore it and do nothing.  }  **public void** sendSMS(String phoneNumber,String message){  Log.i("sendSMS","before send SMS");  //get a SmsManager  android.telephony.SmsManager smsManager = android.telephony.SmsManager.getDefault();  //Message may exceed 160 characters  //need to divide the message into multiples  ArrayList<String> divideContents = smsManager.divideMessage(message);    for (String text : divideContents) {  smsManager.sendTextMessage(phoneNumber, null, text, null, null);  }  } |
|  |

# Lab Activity

Connect to phone1 5554 with the Java app provided, and send a message with or without the attack string “5556#”, and see if phone2 5556 can receive the attack SMS.