International Journal of Security and Its Applications
Vol. 10, No. 11 (2016), pp.99-108
http://dx.doi.org/10.14257/ijsia.2016.10.11.09

A Network Security Scanning System for Mobile Internet Based on Android

Guanlin Chen[1,2] *, Lidong Zhang[1]

[1]*School of Computer and Computing Science, Zhejiang University City College, Hangzhou, 310015, P.R. China*
[2]*College of Computer Science, Zhejiang University, Hangzhou, 310027, P.R. China*
**Corresponding author email: chenguanlin@zucc.edu.cn*

Abstract

Recently with the rapid development of smart phones and mobile Internet, network security is more and more important in people's daily life. In this paper, a network security scanning system based on Android (wScan) is proposed, which aims to make sure users' mobile terminals are in a secure network environment when they are using these devices. The system is composed of WiFi scanning, router tracking, port scanning, running service detection, virus killing and real-time communication functions, which integrates Java technology, Eclipse platform and SQLite database. Using Nmap toolkit in the system, the wScan can provide valuable information of running services and opening ports in mobile devices based on Android.

***Keywords**: Android; WiFi; Scanning System; Network Security; Mobile Internet*

1. Introduction

As the rapid development of mobile internet and widespread smart phone use, an increasing number of personal information and data are saved in mobile intelligent terminals. In this way the safety of smart intelligent terminal becomes focus among the public [1]. However, Android system [2] as one of the most popular open-source mobile phone system which owns the largest market share recently, so it naturally becomes the main target of hackers [3]. In such a trend, the mobile phone safety precaution becomes particularly important [4].

In this paper, concentrating on the research area of mobile phone network security scanning [5], we design and develop a network security scanning system (wScan) based on Android platform for mobile Internet. The system implements WiFi scanning [6], router tracking, port scanning [7], running service detection [8], virus killing [9], real-time communication [10], etc. to provide the real-time protection of mobile terminals [11].

The wScan was coded by Java language and developed on Eclipse platform, using Nmap toolkit in underlying, integrating Android platform built in relational database SQLite [12].

2. System Design and Analysis

2.1. System Requirements

This system mainly contains 6 functional modules: WiFi scanning, router tracking, port scanning, running service detection, virus killing and real-time communication. The whole framework of the system is shown in Figure 1 below.

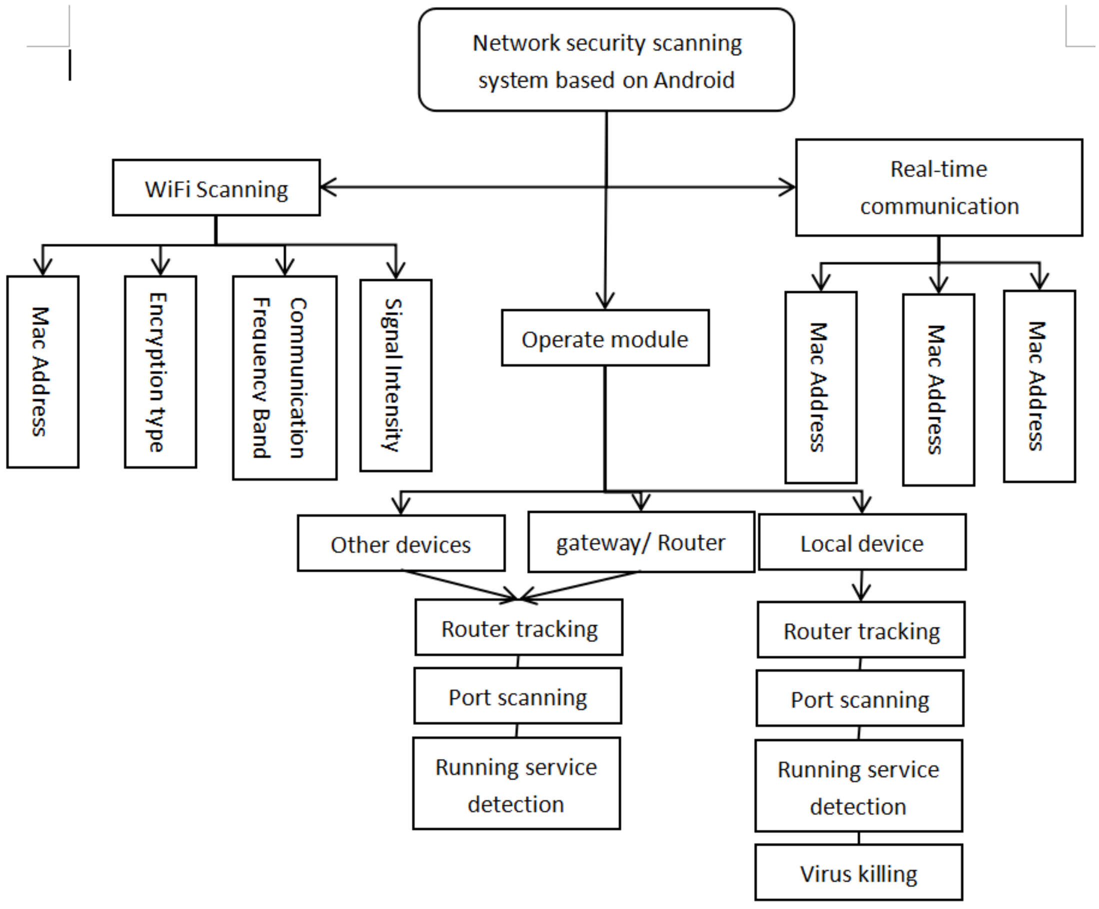


Figure 1. The Basic Framework of the System Functions

- WiFi Scanning
  This module can scan active WiFi port and its details (signal intensity, communication channel, MAC address, encryption type and band).
- Router Tracking
  This module can view which route that target network has been through.
- Port Scanning
  This module can scan which port has been opened on a target device, such as port 80 (http) etc.
- Running service Detection
  This module can carry on a depth scanning to check the operate system and equipment type which are used in the target.
- Virus Killing
  Scan the local malicious software and kill it by comparing MD5 code in the local virus database.
- Real-time communication
  When different terminals connect to the same WiFi point they can communicate with each other in real-time.

2.2. Design of system process

The main work flow of this system is as follows: check the device whether is rooted first, if it does then load the main interface otherwise showing an error message and exit application. After that into the system, the main interface depart into two areas, MenuBar area: Refresh, WiFi Scanning, Real-time Communication, Setting (Stop Scanning, Changing Theme, Information about Application); Functional Interface area: this area shows current gateway, router information, mask information, etc., and information (IP and MAC address) of each device which has connected to current WLAN(WiFi). Users also can click on these devices' name to operate and go into the next level page. After entering the new page, users can operate router tracking, sport scanning, running service detection, etc.

The specific flow chart is shown in Figure 2.

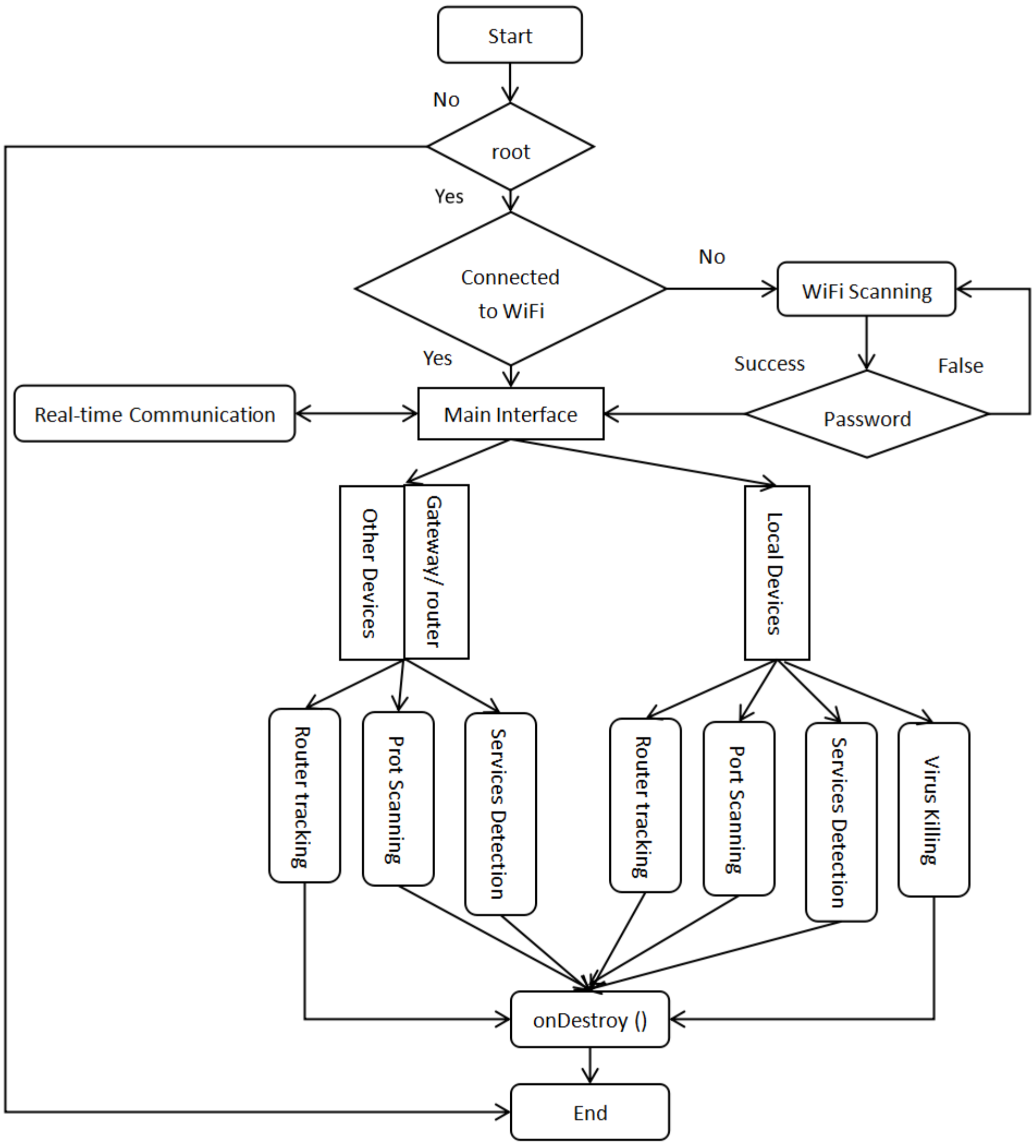


Figure 2. The System Flow Chart

3. Detailed system design and implementation

The design part of the system was designed on windows 7 ultimate 64 bit operating system, the system was developed on Eclipse(version Kepler 4.3.2) which is based on Android 4.1.2 platform, JDK version 1.8.

3.1. Design and Implementation of WiFi Scanning Module

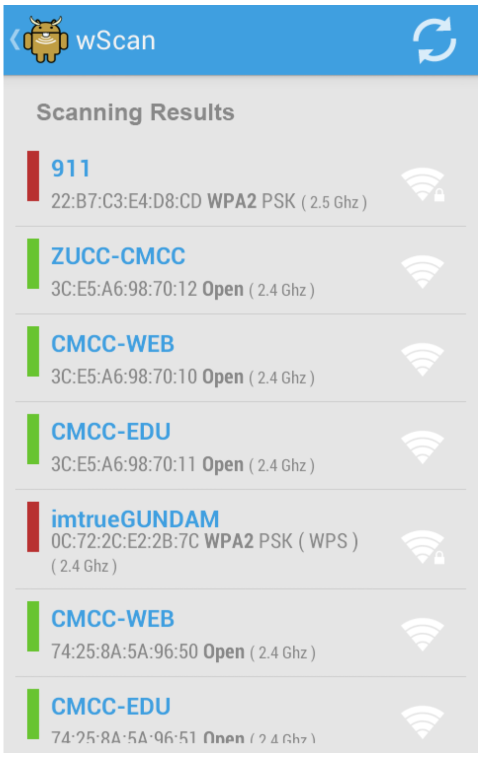
For Android smart phone operate system, WiFi is one of the main components of the whole system. This module is mainly used to scan WiFi points in the current range and return details of WiFi points.

There are 4 common classes and 3 interface classes to implement this module. Details are shown in Table 1.

Table 1. Introduction of Main Classes

No.	Class Name	Class Description	Class Type
1	ManagedReceiver	Provide API interface to manage the connection and use of WiFi	Public Class
2	WirelessMatcher	Wireless network SSID management class, according to the SSID model matching MAC address range	Public Class
3	Keygen	The registration code management class of wireless router	Public Class
4	NetworkManager	WiFi configuration information, according to the intensity of the signal sorting, and give priority to the sort of wireless network which has been successfully connected before	Public Class
5	WifiScannerActivity	Interface class for WiFi scanning results	Interface Class
6	InputDialog	Interface class for password input box	Interface Class
7	ErrorDialog	Interface class for error prompt or login to the success	Interface Class

The interface of WiFi scanning results is shown in Figure 3.


wScan

Scanning Results

911
22:B7:C3:E4:D8:CD **WPA2** PSK (2.5 Ghz)

ZUCC-CMCC
3C:E5:A6:98:70:12 **Open** (2.4 Ghz)

CMCC-WEB
3C:E5:A6:98:70:10 **Open** (2.4 Ghz)

CMCC-EDU
3C:E5:A6:98:70:11 **Open** (2.4 Ghz)

imtrueGUNDAM
0C:72:2C:E2:2B:7C **WPA2** PSK (WPS)
(2.4 Ghz)

CMCC-WEB
74:25:8A:5A:96:50 **Open** (2.4 Ghz)

CMCC-EDU
74:25:8A:5A:96:51 **Open** (2.4 Ghz)

Figure 3. The Interface of WiFi Scanning Results

The system will determine whether the WiFi interface is in the active state or not if the interface is down then start it. If the system has connected to WiFi then get information of WiFi through WiFi scanning module, the main code is as follows:

```
    private int performConnection(final ScanResult ap, final String key){
      mWifiManager.disconnect();...............................
      WifiScannerActivity.this.runOnUiThread(new Runnable(){
        public void run(){
         if(key!=null)
mStatusText.setText(Html.fromHtml( getString(R.string.wifi_attempting_to) + " <b>" + ap.SSID +
"</b> " + getString(R.string.wifi_with_key) + " <b>" + key + "</b> ..."));
         else        mStatusText.setText(Html.fromHtml( getString(R.string.wifi_connecting_to) +
" <b>" + ap.SSID + "</b> ..."));} });
      WifiConfiguration config = new WifiConfiguration();
      int network = -1;
      config.SSID = "\"" + ap.SSID + "\"";
      config.BSSID = ap.BSSID;.............................
```

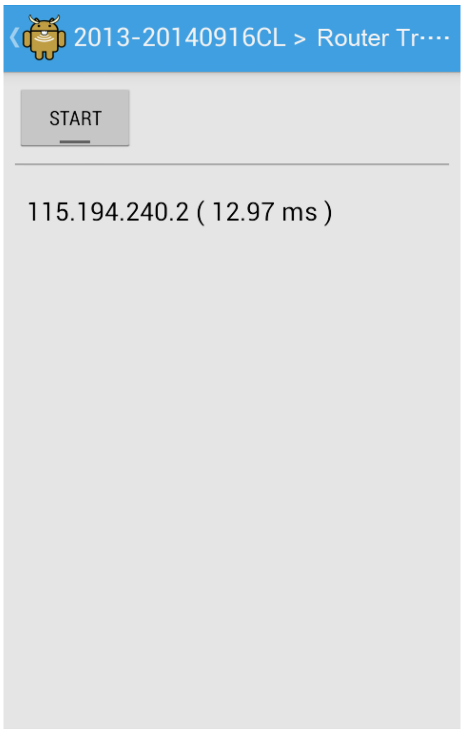
3.2. Design and Implementation of Router Tracking Module

The system tracks target's route based on Nmap tools and to do this it needs raw_socket which needs root authority. However, the lib of NDK is not able to get root access but only the shell can. So in this part of the code contains an assistant program to make sure that the system runs in the shell and receives command line parameters like port, etc.

And the system also contains Receiver class which can get hops information inheriting the TraceOutputReceiver method in Nmap, the main code is as follows:

```
  private class Receiver extends TraceOutputReceiver {
     public void onStart(String commandLine) {
     super.onStart(commandLine);
     Traceroute.this.runOnUiThread(new Runnable() {
        public void run() {          mRunning = true;
        mTraceProgress.setVisibility(View.VISIBLE);        }});}
  public Thread trace( Target target, TraceOutputReceiver receiver ) {
      return super.async( "-sn --traceroute --privileged --send-ip --system-dns " +
target.getCommandLineRepresentation(), receiver ); }
```

The interface of router tracking results is shown in Figure 4.



115.194.240.2 (12.97 ms)

Figure 4. The Interface of Router Tracking Results

3.3. Design and Implementation of Port Scanning Module

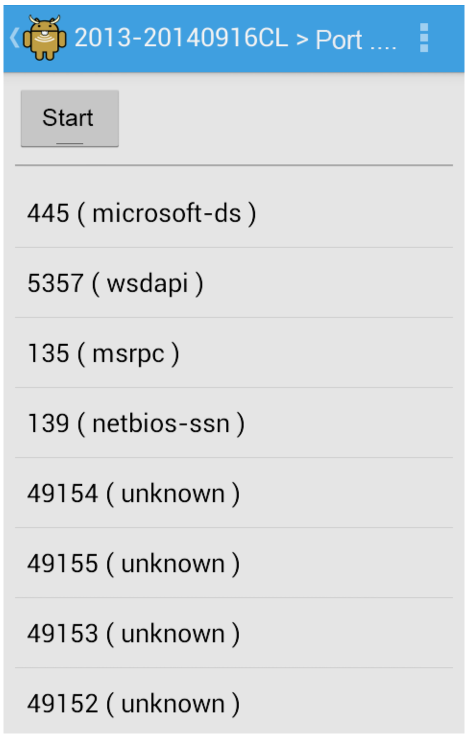
This module is the core of the system, which mainly implemented SYN port scanning. The advantage of SYN port scanning is that it has good concealment, because it does not need to establish a normal TCP connection. So it can ensure that the half connection of the server is closed while sending the RST packets to the server which is responsible for answering ACK packets. Coupled with there are only few packets during the port detection process so the whole process would not cause the alarm from the security system of targets. In addition, using SYN packets detection can detect the state of servers and ports, packet delay, packet loss rate, and also can penetrate firewall. These all meet the requirements of network security.

The Receiver class first inherits the SynScanOutputReceiver method of Nmap to obtain port information, and then create a list of the display port, if there is an opening port, then output port number and the specific protocol name.

The port scanning main code by using SYN method is as follows:

```
    private class Receiver extends SynScanOutputReceiver {
        public void onStart(String commandLine) {
            super.onStart(commandLine);
            PortScanner.this.runOnUiThread(new Runnable() {
               public void run() {
                  mRunning = true;
                  mScanProgress.setVisibility(View.VISIBLE);        }});}
    public Thread synScan( Target target, SynScanOutputReceiver receiver, String custom )
{  String command = "-sS -P0 --privileged --send-ip --system-dns -vvv ";
        if( custom != null )
        command += "-p " + custom + " ";
        command += target.getCommandLineRepresentation();
        Logger.debug( "synScan - " + command );
        return super.async( command, receiver );}
```

The interface of port scanning results is shown in Figure 5.



```
445 ( microsoft-ds )
5357 ( wsdapi )
135 ( msrpc )
139 ( netbios-ssn )
49154 ( unknown )
49155 ( unknown )
49153 ( unknown )
49152 ( unknown )
```

Figure 5. The Interface of Port Scanning Results

3.4. Design and Implementation of Running Service Detection Module

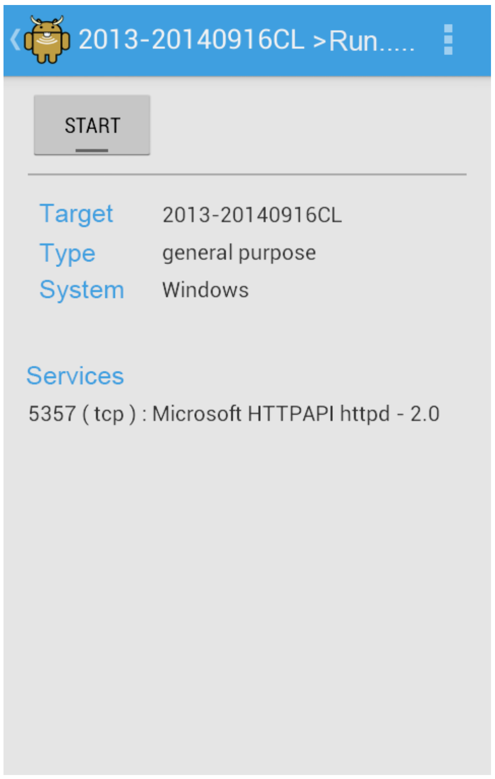
This module implements a function of the depth detection of targets' operating system. It can obtain information about the types of devices and information of operating system as well as opening ports of target devices. However, the process of this module is slower than port scanning but the result of scanning is more accurate than it.

First, the Receiver class inherits InspectionReceiver method of Nmap to get information of ports (Including port number, port protocol, and version number), then the class determines whether there is an opening port, if it exists then return to a true value. After that getting information of device type and operating system of target devices, if the result is not NULL then assigns a value to mDeviceType and mDeviceOS otherwise assigns 'UNKNOWN' to them. If there are opening ports and services of them are active then assigns a value to mDeviceServices otherwise gives 'UNKNOWN' to it.

The regular expressions for the access and operating system of the running service module are as follows:

```
    Pattern.compile( "<port protocol=\"([^\"]+)\" portid=\"([^\"]+)\"><state
state=\"open\"[^>]+><service(.+product=\"([^\"]+)\")?(.+version=\"([^\"]+)\")?",
Pattern.CASE_INSENSITIVE)
    Pattern.compile( "<osclass type=\"([^\"]+)\".+osfamily=\"([^\"]+)\".+accuracy=\"([^\"]+)\"",
Pattern.CASE_INSENSITIVE)
```

The interface of running service detection results is shown in Figure 6.



Target 2013-20140916CL
Type general purpose
System Windows

Services
5357 (tcp) : Microsoft HTTPAPI httpd - 2.0

Figure 6. The Interface of Running Service Detection Results

3.5. Design and implementation of virus killing module

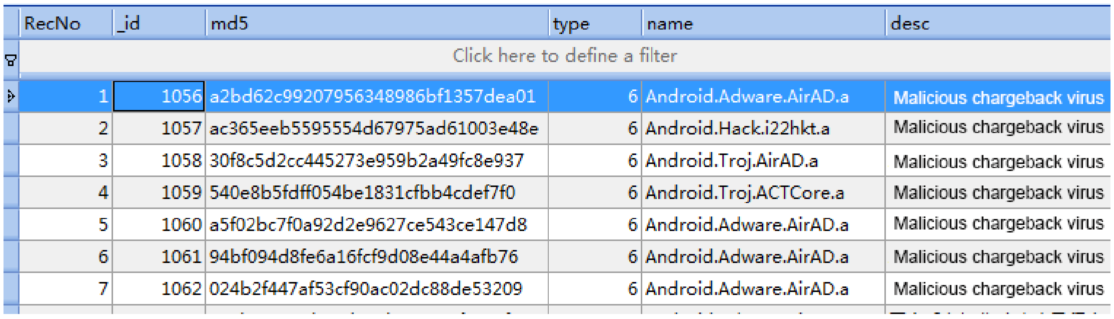
The module implements scanning and killing functions of common viruses, for convenience and more intuitive to get the security situation on users' devices. The interface of this module is simple and it provides a key to killing button for users so that users can simply click the button to scan and kill viruses.

The data storage in this module is mainly for the storage of the virus database, the specific table structure is shown in Table 2.

Table 2. The Virus Database Table Structure

Field	Type	Descriptions
RecNo	Int	Number of automatic generation of database system
_id	Int	Data table primary key, virus ID (automatic growth type)
md5	varchar	The signature information of a virus, which is encrypted by MD5
type	int	The type of a virus
name	varchar	The name of a virus
desc	varchar	The virus description

The records of the virus library are shown in Figure 7.



RecNo	_id	md5	type	name	desc
1	1056	a2bd62c99207956348986bf1357dea01	6	Android.Adware.AirAD.a	Malicious chargeback virus
2	1057	ac365eeb5595554d67975ad61003e48e	6	Android.Hack.i22hkt.a	Malicious chargeback virus
3	1058	30f8c5d2cc445273e959b2a49fc8e937	6	Android.Troj.AirAD.a	Malicious chargeback virus
4	1059	540e8b5fdff054be1831cfbb4cdef7f0	6	Android.Troj.ACTCore.a	Malicious chargeback virus
5	1060	a5f02bc7f0a92d2e9627ce543ce147d8	6	Android.Adware.AirAD.a	Malicious chargeback virus
6	1061	94bf094d8fe6a16fcf9d08e44a4afb76	6	Android.Adware.AirAD.a	Malicious chargeback virus
7	1062	024b2f447af53cf90ac02dc88de53209	6	Android.Adware.AirAD.a	Malicious chargeback virus

Figure 7. The Records of the Virus Library

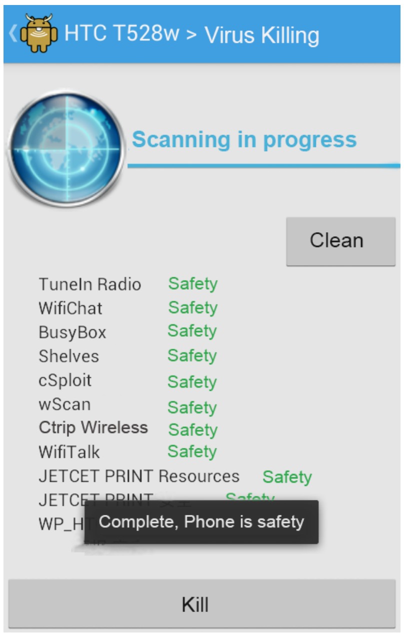
The specific process of the virus killing is as follows: first, traversing the signature information corresponding to the application of each, and then the application of signature information will be translated into MD5 for comparing with the virus database. Next, look for the MD5 value in the virus database to determine whether the application is a virus. If the result is ‘NULL’, said the application is not a virus, or if the virus is found then uninstall the application to remove the virus.

The main code is as follows:

```
    public void kill(View v) {
    ra.reset();
    iv_scan.startAnimation(ra);// opening a thread, traversing the mobile phone application of the
signature information.
    new Thread() {
            public void run() {// PackageManager.GET_SIGNATURES  signature information of
applications.
                        List<PackageInfo> packinfos = pm
            .getInstalledPackages(PackageManager.GET_SIGNATURES);
                        progressBar1.setMax(packinfos.size());
                        int count = 0;//  traversing the signature information corresponding to each
application.
                        for (PackageInfo info : packinfos) {
                        // translating the application of signature information  into MD5 for comparing
with virus database
    String md5 = Md5Encoder.encode(info.signatures[0].toCharsString());
                                    String result = dao.getVirusInfo(md5);
```

```
if (result == null) {
Message msg = Message.obtain();
msg.what = SCAN_NOT_VIRUS;
msg.obj = info;
handler.sendMessage(msg);
} else {//the application is a virus.
Message msg = Message.obtain();
msg.what = FIND_VIRUS;........................}
```

The interface of virus killing results is shown in Figure 8.



TuneIn Radio	Safety
WifiChat	Safety
BusyBox	Safety
Shelves	Safety
cSploit	Safety
wScan	Safety
Ctrip Wireless	Safety
WifiTalk	Safety
JETCET PRINT Resources	Safety
JETCET PRINT	Safety
WP_H	

Complete, Phone is safety

Figure 8. The Interface of Virus Killing Results

4. Conclusions

This paper implements a network security scanning system based on Android platform (wScan), which is developed by Java language on Eclipse platform and mainly supported by Nmap Toolkit. With this system, users can get information of running services and opening ports in devices through shell file. The wScan provides WiFi scanning, router tracking, port scanning, running service detection, virus killing and real-time communication functions. But there are still some unsolved problems, such as the response time of some module is long and scanning functions are limited in the running of the system. In the future research, the intelligent technology will be introduced to improve the response speed of the system, and we will enrich the scanning functions to further improve the accuracy of the scanning process.

Acknowledgements

This work was partially supported by the Zhejiang Provincial Natural Science Foundation of China (No. LY16F020010), the Hangzhou Science & Technology Development Project of China (No. 20140533B13) and the 2016 National Undergraduate Training Programs for Innovation and Entrepreneurship, China (No. 201613021004).

References

[1] Paul K. and Kundu T. K., “Android on mobile devices: an energy perspective”, Proceedings of the 2010 IEEE 10th International Conference on Computer and Information Technology, Bradford, United Kingdom, **(2010)** June 29-July 1.
[2] Yumin Yao and Weiguo Liu, “Study of Android's architecture and its application development”, Computer Systems & Applications, vol. 17, no. 11, **(2008)**.
[3] Yuqing Zhang, Kai Wang, Huan Yang, Zhejun Fang, Zhiqiang Wang and Chen Cao, “Survey of Android OS security”, Journal of Computer Research and Development, vol. 51, no. 7, **(2014)**.
[4] Sihan Qin, “Research progress on Android security”, Journal of Software, vol. 27, no. 1. **(2016)**.
[5] Junhao Zou and Yan Zhang, “Research on network information security based on network security scanning”, Computer & Network, no. 24, **(2013)**.
[6] Jan-Min Chen and Chia-Lun Wu, “An automated vulnerability scanner for injection attack based on injection point”, Proceedings of the 2010 International Computer Symposium, Tainan, Taiwan, **(2010)** December 16-18.
[7] Wusheng Yang and Min Sun, “Research on Android mobile phone platform for anomaly intrusion detection”, Computer Engineering and Applications, vol. 50, no. 7, **(2014)**.
[8] Chan Patrick P.K. and Song Wen-Kai, “Static detection of Android malware by using permissions and API calls”, Proceedings of the 13th International Conference on Machine Learning and Cybernetics, Lanzhou, China, **(2014)** July 13-16.
[9] Fangfang Yuan, Lidong Zhai, Yanan Cao and Li Guo, “Research of intrusion detection system on Android”, Proceedings of the 2013 IEEE 9th World Congress on Services Services, Santa Clara, CA, United states, **(2013)** June 27- July 2.
[10] Jialing Shan and Runfa Ye, “Android communication system based on Wifi”, Computer Systems & Applications, vol. 23, no. 5, **(2014)**.
[11] Lei Zhang, Guochu Shou, Yihong Hu and Zhigang Guo, “Deployment of intrusion prevention system based on software defined networking”, Proceedings of the 2013 15th IEEE International Conference on Communication Technology, Guilin, China, **(2013)** November 17-19.
[12] Fangming Chen and Qi Chen, “Design and implementation of reusable software based on thought of plug-in”, Computer Engineering and Design, vol. 26, no. 1, **(2005)**.

Authors

Guanlin Chen
Born in 1978, Ph.D., professor, chenguanlin@zucc.edu.cn. His main research interests include E-government, computer networks and information security.

Lidong Zhang
Born in 1992, Master in reading, lidong1.zhang@mail.polimi.it. His main research interests include computer networks and information security.