UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido
https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A CASE-STUDY VISION

BlueLock a tool to prevent Bluetooth attacks

BlueLock una herramienta para prevenir ataques en Bluetooth

Gerardo Alberto Castang-Montiel[1], ***Fernando Betancourt-Duque***[2], ***Luz Adriana Peña-Salazar***[3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 10/02/2020
Recibido: 17/02/2020
Aceptado: 16/04/2020

Keywords:
Android
Attacks
Bluetooth
Security

ABSTRACT

Digital devices are presented several vulnerabilities because when making a connection your information can be compromised. Bluetooth is a means of wireless communication that allows you to transfer files with distances of up to 100 meters, but its security has been seen as malicious users, attacking special cell phones. The Bluetooth manufacturers have improved their security but it has not been enough to control the vulnerabilities. The security of the Bluetooth connection has been directed by the manufacturers, but the user has not been able to carry out a control of this security.

Palabras clave:
Android
Ataques
Bluetooth
Seguridad

RESUMEN

Los dispositivos digitales presentan vulnerabilidades ya que al realizar una conexión su información puede verse comprometida. Bluetooth es un medio de comunicación inalámbrico muy reconocido ya que permite transferencia de archivos con distancias de hasta 100 metros, pero su seguridad se ha visto afectada por usuarios maliciosos, atentando en especial dispositivos celulares. Los fabricantes de bluetooth han mejorado su seguridad, pero no ha sido suficiente para controlar las vulnerabilidades. La seguridad de conexión de bluetooth ha sido gestionada por los fabricantes, pero el usuario final no puede llevar un control de esta seguridad.

[1]BSc. In Electronic Engineering, Universidad Distrital Francisco José de Caldas, Colombia. Current position: Professor at Universidad Distrital Francisco José de Caldas, Colombia. E-mail: gacastangm@udistrital.edu.co
[2]BSc. in Telematic engineering, Universidad Distrital Francisco José de Caldas, Colombia. Current position: Insoftar. E-mail: duquefernandobetancourt@gmail.com
[3]BSc. in Telematic engineering, Universidad Distrital Francisco José de Caldas, Colombia. E-mail: adriana-0225@hotmail.com

1. Analysis stage

1.1. Attack Deployment

Having identified several types of Bluetooth attacks that have existed in a system, it is possible to test those by pentesting. The tests performed are of the white box (tandem) type, since the attack environment was implemented using the Kali Linux pentesting platform, documenting each tool and designing the Bluetooth hacking tests.

To get started, using Kali Linux ensure that Bluetooth is active [1] by using the following commands:

```
root@kali:~# service bluetooth start
root@kali:~# service bluetooth status
● bluetooth.service - Bluetooth service
  Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset
  Active: active (running) since Sun 2018-11-11 11:35:52 -05; 5h 33min ago
        Docs: man:bluetoothd(8)
 Main PID: 622 (bluetoothd)
  Status: "Running"
        Tasks: 1 (limit: 4915)
  Memory: 4.0M
  CGroup: /system.slice/bluetooth.service
        └─622 /usr/lib/bluetooth/bluetoothd
```

The Bluetooth adapter's interface is then validated:

```
root@kali:~# hciconfig
hci0:    Type: Primary  Bus: USB
    BD Address: C8:FF:28:A6:15:F1  ACL MTU: 820:8  SCO MTU: 255:16
    UP RUNNING PSCAN ISCAN
    RX bytes:3857 acl:0 sco:0 events:602 errors:0
    TX bytes:28371 acl:0 sco:0 commands:388 errors:0
```

It can be used the BlueLog tool which is native to Kali Linux and allows viewing and registering detectable devices on the Bluetooth network and runs as follows:

```
root@kali:~# bluelog
Bluelog (v1.1.2) by MS3FGX
---------------------------
Autodetecting device...OK
Opening output file: bluelog-2018-11-11-1655.log...OK
Writing PID file: /tmp/bluelog.pid...OK
Scan started at [11/11/18 16:55:26] on C8:FF:28:A6:15:F1.
Hit Ctrl+C to end scan.
```

BlueScanner is another Kali Linux tool that allows scanning through Bluetooth network adapter, capturing the connectivity features of the networked devices and runs as follows:

```
root@kali:~# btscanner
```

With "i" command, a network device scan is performed generating a directory with MAC address of the device and a file with all the collected information.

Bluejacking Attack: To carry out the attack a contact must be created, only the name box will contain the message in the device's address book that will act as an attacker saving the contact. Then, an area with a large number of mobile devices is searched, the option "send via Bluetooth" is selected and it is sent to the target devices.

Bluesmack Attack: It requires a command line executed directly on the attacking device's console, which makes a certain amount of calls in a time interval that results in a bluetooth service drop by the host or victim device. The script consists as follows:

```
root@kali:~# while read r; do l2ping -s 50 84:C7:EA:57:36:D7; done < numscans
```

First, a file called numscans is created with a counter from 1 to 100, to have it as a reference in the script and avoid memory overflow or overload in the attacking host [2].

```
root@kali:~# while read r; do l2ping -s 50 84:C7:EA:57:36:D7; done < numscans
Ping: 84:C7:EA:57:36:D7 from C8:FF:28:A6:15:F1 (data size 50) ...
0 bytes from 84:C7:EA:57:36:D7 id 0 time 5.91ms
0 bytes from 84:C7:EA:57:36:D7 id 46 time 63.44ms
Send failed: Connection reset by peer
Ping: 84:C7:EA:57:36:D7 from C8:FF:28:A6:15:F1 (data size 50) ...
```

1.2. Compilation of Attack Results

It can be seen that the Bluetooth protocol is designed to provide interconnection facilities for devices and this is a necessary approach to IoT device communication, so that through the software solution designed it is not planned to limit or block the protocol functionality in an automated way by an application or service, this is relegated to the user, designing an application that allows the user to browse the connectivity events, paired devices and can take the decision to lock or not a device.

2. Design Stage

2.1. Software Architecture

The Class diagram is show in Figure 1.

2.2. Iteration Diagram

The diagram of the Figure 2 shows the application running on the device, engaged in different communications with other devices, thus the device is in a susceptible environment where it receives different bluetooth attacks. The device has a bluetooth interface which communicates with the application (BlueLock) when a bluetooth adapter is instantiated.

The application (BlueLock), has an event filter that specifically detects changes in bluetooth adapter status, such as incoming connections and recording everything in the database, as well as in the application log.

Figure 1: Class diagram.

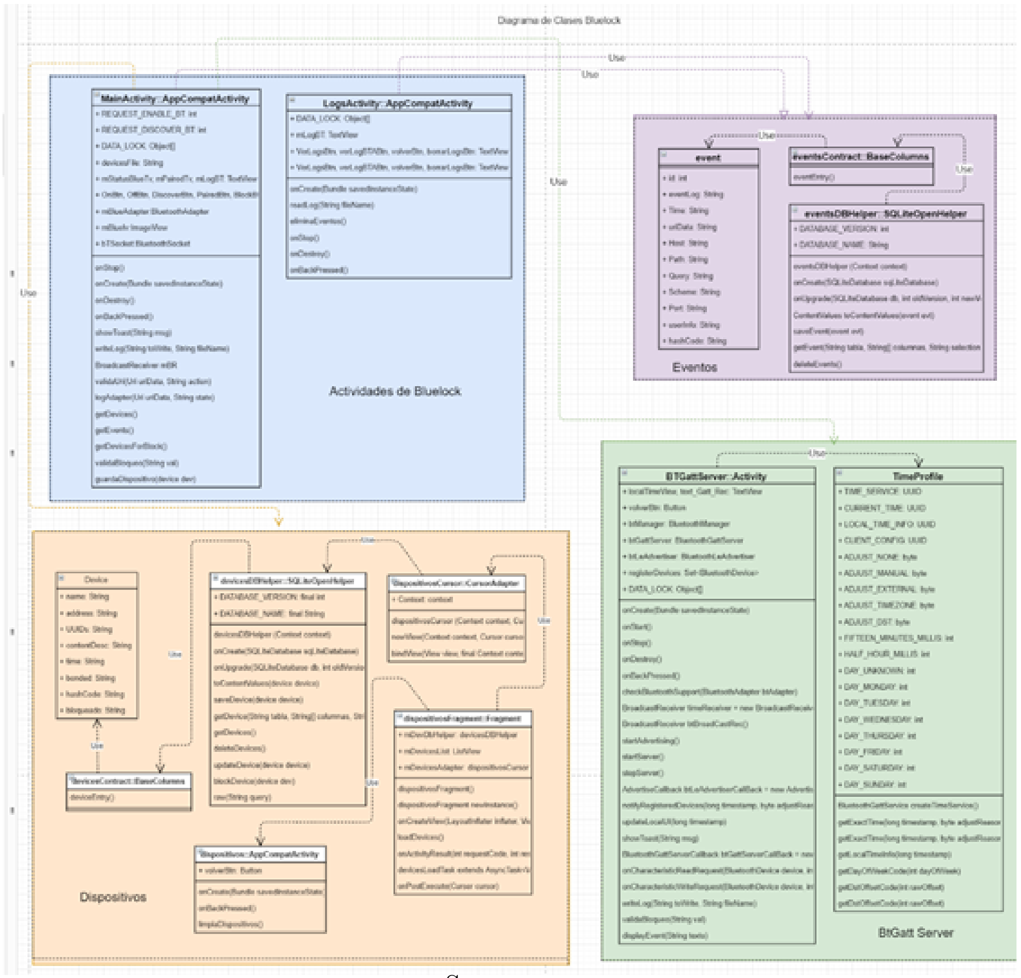
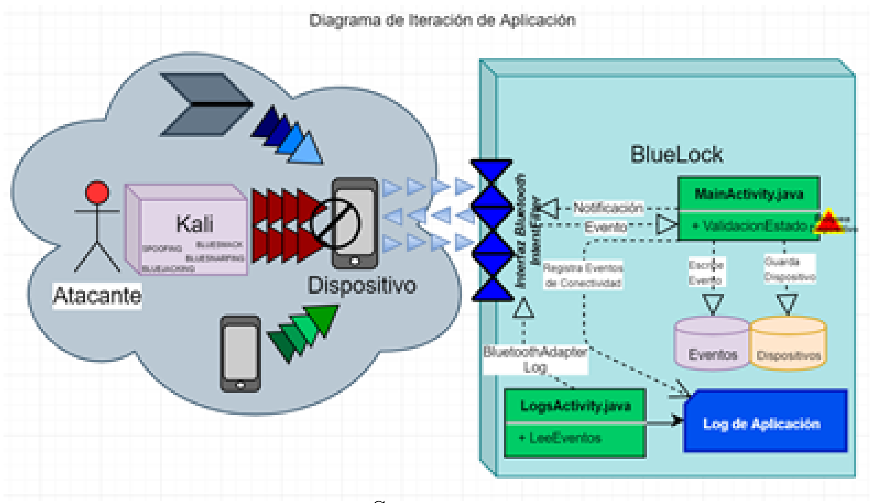


Diagrama de Clases Bluelock
Actividades de Bluelock
MainActivity::AppCompatActivity
LogsActivity::AppCompatActivity
Eventos
event
eventsContract::BaseColumns
eventsDBHelper::SQLiteOpenHelper
Dispositivos
Device
devicesDBHelper::SQLiteOpenHelper
DispositivosCursor::CursorAdapter
dispositivosFragment::Fragment
DevicesContract::BaseColumns
Dispositivos::AppCompatActivity
BtGatt Server
BTGattServer::Activity
TimeProfile
Use

Source: own

Figure 2: Application iteration diagram.



Source: own

3. Development Stage

3.1. *Application Development*

It is proposed to develop an Android mobile application that allows to control the Bluetooth communication, turn on/off the controller, enable device searching, consult paired devices, Bluetooth event log, locked devices and access to communication feature through Bluetooth with Gatt Server [3].

Application features:

- Bluetooth user interface to control the device.
- Bluetooth event log generation detected by the application.
- Device locking interface through bluetooth interface.
- Gatt Server allowing communication event detection with several devices connected through Bluetooth.
- Non-relational database generation allowing the device Bluetooth connection data store and its related events.

3.2. *Code Documentation*

Bluetooth package filtering: The main application feature is the BroadcastReceiver function, which allows to receive and manage communication events sent by the bluetooth adapter in Context.sendBroadcast. This function filters and controls the states detected by the application and writes in log these events [4, 5].

```
//BroadcastReceiver
private final BroadcastReceiver mBR = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final String action = intent.getAction();
        String evento = "";
        boolean bloqueo = false;
        final Uri uriData = intent.getData();
        event evt = validaUri(uriData, action);
        eventsDBHelper dbHelper = new eventsDBHelper(getApplicationContext());
        String log = "", showToastLog = "";
        boolean logger;
        if(action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)
                ||action.equals(BluetoothAdapter.ACTION_DISCOVERY_STARTED)
                    ||action.equals(BluetoothAdapter.ACTION_DISCOVERY_FINISHED)
                        ||action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
            final int estado = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
BluetoothAdapter.ERROR);
            switch (estado) {
                case BluetoothAdapter.STATE_OFF:
                    evento = "<font color='blue'>BluetoothAdapter.STATE_OFF";
                    log += logAdapter(uriData,evento);
                    break;
                case BluetoothAdapter.STATE_ON:
                    evento = "<font color='blue'>BluetoothAdapter.STATE_ON";
                    log += logAdapter(uriData,evento);
                    break;
```

Device connection status validation in the "Connecting" event. If the device is in a locked state it generates a removeBond event, which removes the paired device from the list of Bluetooth devices, which generates a lock at connection attempt.

```
case BluetoothAdapter.STATE_CONNECTING:
   evento = "<font color='purple'>BluetoothAdapter.STATE_CONNECTING";
   log += logAdapter(uriData,evento);
   BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
   bloqueo = validaBloqueo(device.getAddress());
   UUID uuid = UUID.randomUUID();
   if(bloqueo){
       try {
           Log.w("ACTION_ACL_CONNECTED","Entró a removeBond");
           Method m = device.getClass()
                   .getMethod("removeBond", (Class[]) null);
           m.invoke(device, (Object[]) null);

           evento = "<font color='red'>Dispositivo Bloqueado ("+uuid+"): "+device.getAddress()+";
"+device.getName();
           log += logAdapter(uriData,evento);
           showToast("Dispositivo bloqueado ("+uuid+") : "+device.getAddress()+" : "+ device.getName());
       } catch (Exception e) {
           Log.w("ACTION_ACL_CONNECTED", e.getMessage());
           evento = "<font color='red'>Error de Dispositivo Bloqueado ("+uuid+"):
"+device.getAddress()+"; "+device.getName();
       }
       log += logAdapter(uriData,evento);
   }
   break;
```

The next validated state is when an available device is found in the range, then we store the MAC and NAME of the found device, having the information of unpaired devices available in the application persistence.

```
if(action.equals(BluetoothDevice.ACTION_FOUND)){
   BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
   int rssi = intent.getShortExtra(BluetoothDevice.EXTRA_RSSI, Short.MIN_VALUE);
   Log.w("ACTION_ACL_CONNECTED","Entró a
ACTION_ACL_CONNECTED"+device.getName()+"\n"+device.getAddress()+"\n"+rssi);
   device dev = new device();
   dev.setId(0);
   dev.setName(device.getName());
   dev.setAddress(device.getAddress());
   dev.setContentDesc(String.valueOf(rssi));
   dev.setBloqueado(String.valueOf(false));
   dev.setTime(String.valueOf(Calendar.getInstance().getTime()));
   dev.setBonded(String.valueOf(false));
   dev.setHashCode(String.valueOf(device.hashCode()));
   boolean result = guardaDispositivo(dev);
   evento = "<font color='green'>BluetoothDevice.ACTION_FOUND";
   log += logAdapter(uriData,evento);
   evt.setEventLog(evento);
   dbHelper.saveEvent(evt);
   logger = writeLog(log, "BluetoothAdapter.txt");
   showToastLog = "BluetoothAdapter.ACTION_FOUND";
   showToast(showToastLog);
}
```

Device locking: In the application it is very important the state in which the device is, therefore, if the application finds a device and this is active to make a connection, you can block its status to prevent it from performing clandestine actions to our device. The following method blockDevice validates the status of this device and updates the locked column [6].

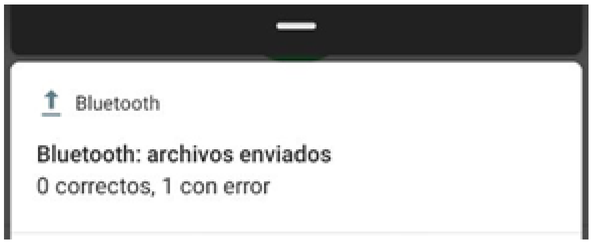
```
public boolean blockDevice(device dev){
    boolean result = false;
    String estado = "bloqueado";
    try {
        String args [] = new String[1];
        args[0] = dev.getAddress();
        Cursor cursor =
this.getDevice(devicesContract.deviceEntry.tableName,null,devicesContract.deviceEntry.address+"=?
",args,null,null,null);
        if (cursor.getCount() > 0){
            cursor.moveToFirst();
            String bloqueado =
cursor.getString(cursor.getColumnIndex(devicesContract.deviceEntry.bloqueado));
            if(bloqueado == null){
                estado = "bloqueado";
            }
            else if(bloqueado.equals("bloqueado")) {
                estado = "desbloqueado";
            }
        }else{
            Log.e("deviceDBHelper", "Error bloqueando Dispositivo no emparejado:"+dev.getAddress() );
        }
        ContentValues cv = new ContentValues();
```

4. Test Stage

4.1. *Bluejacking Attack Test*

Attack Deployment: To begin with using the BlueLock application on target device 2 with MAC A0:8D:16:88:A5:BD and block all devices that have Bluetooth enabled. Then it proceeds to create a new contact on attacker device 1 with MAC C0:8C:71:84:94:A1 and sends or shares it with victim device (Figure 3).

Figure 3: Bluejacking Attack - File Send Error Message.


Bluetooth
Bluetooth: archivos enviados
0 correctos, 1 con error

Source: own

Bluejacking attack result: When running the test it can be seen that the application blocks correctly the contact sending preventing that there will be some communication channel between the two devices.

4.2. *Bluesnarfing Attack Test*

Attack deployment: Remember that this attack is similar to bluejacking, but with the difference that it enters and steals information from the attacked device [7,8]. Executing the same commands used before, the result is shown in the console:

```
root@kali:~# sdptool browse --tree --l2cap C0:8C:71:84:94:A1
Failed to connect to SDP server on C0:8C:71:84:94:A1: Connection timed out
root@kali:~# sdptool browse --tree --l2cap C0:8C:71:84:94:A1
Failed to connect to SDP server on C0:8C:71:84:94:A1: Operation already in progress
root@kali:~# sdptool browse --tree --l2cap C0:8C:71:84:94:A1
Failed to connect to SDP server on C0:8C:71:84:94:A1: Connection timed out
root@kali:~#
```

Bluesnarfing attack result: The victim device, having the attacker in a locked state, avoids service description reading and remains inaccessible to the attacking device.

4.3. *Result Analysis*

- When applying Bluejacking and Bluesnarfing attacks using BlueLock application, it gives a positive result, since the blocking of an attacking device did not allow to deploy the attack, with the Bluejacking, it does not allow notification entry with the annoying contact, and Bluesnarfing was completely stopped since the device tries to initiate communication to make some request.
- Logging allows the user to access the validation of communication events of Bluetooth adapter in a range of time and to easily detect unusual events that can be performed by devices that are within the physical range of the Bluetooth interface [9,10].

5. Conclusions

Software solutions for mobile devices focused on connection and data packet validation can improve Android device security in places like apartments and buildings where there are a lot of active Bluetooth devices therefore being more likely to be hacked. BlueLock is a connectivity event-oriented security application that allows detection of available devices and provides the option to block unwanted ones. This is achieved by validating the status of Bluetooth interface through an Intent filter, which is sensitive to status changes of the Bluetooth adapter.

References

[1] D. Santo, "KALI LINUX". Madrid: Ra-Ma Editorial, 2018.

[2] A. Ramos, C. Barbero, R. Martínez, A. García, and J. González, "Hacking y seguridad de páginas web", Colombia: Ediciones U, Ra – Ma, 2015.

[3] J. Nolasco, "Desarrollo de aplicaciones móviles con Android. Segunda edición", Colombia: Ediciones U, Ra – Ma, 2016.

[4] M. Ciampa, "Security+ Guide to Networks Security Fundamentals", Boston: Cengage, 2018.

[5] A. Lonzetta, P. Cope, J. Cambell, B. Mohd and T. Hayajneh. "Security Vulnerabilities in Bluetooth Technology as Used in IoT", *Sensors and Actuators*,

vol. 7, no. 3, pp 26, july 2018. https://doi.org/10.3390/jsan7030028

[6] J. Hoo, "Study on the Security Threats Factors of a Bluetooth Low Energy", *Convergence Security Journal*, vol. 17, no. 4, p. 9, october 2017.

[7] T. Pandey and P. Khare, "Bluetooth Hacking and Its Prevention", *L&T Technology Services*, vol. 12, p. 8, 2017.

[8] N. Minar and M. Tarique, "Bluetooth Security Threats and Solutions: A Survey", *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 3, no. 1, pp. 127- 148, 2012. https://doi.org/10.5121/ijdps.2012.3110

[9] Android, "Documentation, android.bluetooth", 2019. [Online]. Available https://developer.android.com/reference/android/bluetooth/package-summary.

[10] Clarín, "Advierten sobre los peligros de utilizar el Bluetooth de tu celular", 2019. [Online]. Available in: https://www.clarin.com/tecnologia/advierten-peligros-utilizar-bluetooth-celular_0__WpofinjL.html