VERSION 0.1
May 8, 2015

BTstack for RugGear/MediaTek Chipsets

Getting Started

Dr. sc. Milanka Ringwald
Dr. sc. Matthias Ringwald
contact@bluekitchen-gmbh.com

Contents

This documents describes how BTstack[1] can be installed and used on RugGear devices with MediaTek chipset. It also presents measurements of the reception of Advertising reports from a remote devices, e.g. beacons or peripherals, as this is the crucial step for discovering and connecting to them.

1. Hardware Setup

To install BTstack on a RugGear mobile phone, connect the RugGear device to a Mac or Linux system using a micro USB cable. The installation might also work on Windows with Cygwin[2] and/or MSYS[3] installed. The RugGear device is connected to an USB port during setup and development.

2. General Tools

- Cydia Impactor[4] to get root access.
- Google's Android Developer Tools[5] (ADT) to develop an Android LE Client.
- The Android Debug Bridge[6] (adb) to communicate with a connected Android device via a command line. It comes as a part of the Android Developer Tools.
- Apple's PacketLogger[7] (available to the registered developers as part of the Hardware IO Tools for Xcode download) or Wireshark[8] to look and analyze Bluetooth packet logs.
- Some LE devices that send Advertising reports to test your LE Client. Here, we used the nio Tag[9] and a BTstack-based LE Peripheral on a desktop machine using the BCM20702A0 module.

3. Rooting the RugGear Device

There are various ways to root an Android device. We recommend the Cydia Impactor tool. It is available for all major platforms, it works with most Android devices and it comes from a trustworthy source[10].

To root the device, start Impactor, and press the "Start" button as shown in Fig.1

[1] http://bluekitchen-gmbh.com/btstack

[2] https://www.cygwin.com

[3] www.mingw.org/wiki/msys

[4] http://www.cydiaimpactor.com

[5] http://developer.android.com/tools/index.html

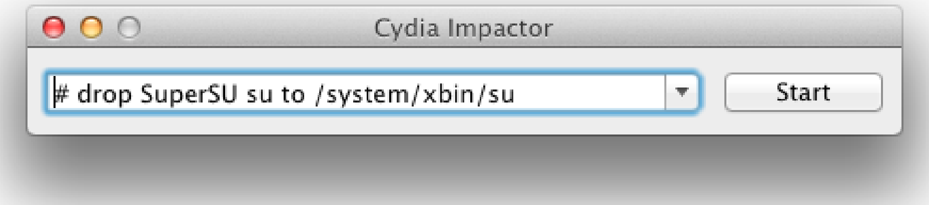
[6] http://developer.android.com/tools/help/adb.html

[7] http://adcdownload.apple.com/Developer_Tools/hardware_io_tools_for_xcode__june_2014/hardwareiotools_june_201

[8] http://www.wireshark.org

[9] https://www.bluenio.com/products/accessories/niotag

[10] It was created by Jay Freeman, who has supporting the use iOS devices outside of Apple's walled garden since the device came out



drop SuperSU su to /system/xbin/su

Figure 1. Cydia Impactor running on OS X.

4. Installing BTstack on RugGear devices with MediaTek Chipsets

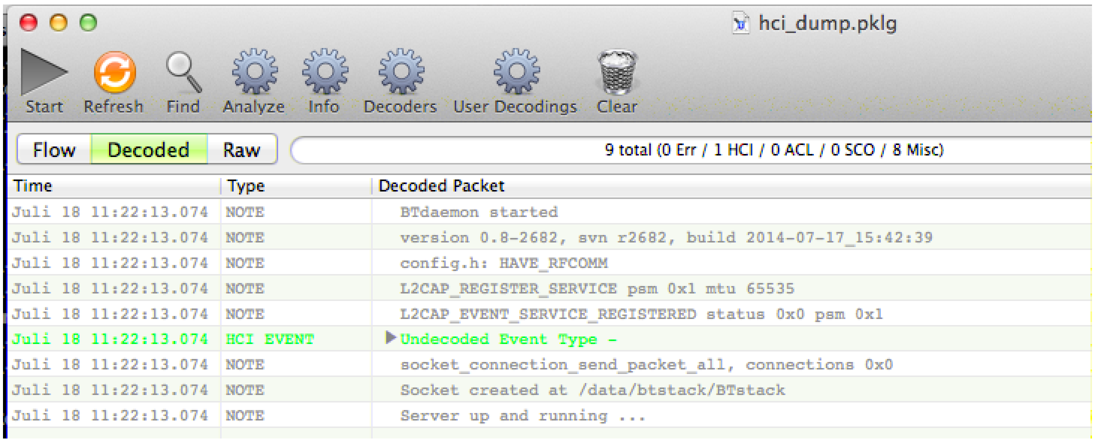
First, extract the provided `btstack-android-mtk-VERSION.tar.gz` archive. Please make sure that the path to the extracted archive does not contain spaces as the installer may fail in this case. Then, start the provided `./installer.sh` in the `mtk` folder. This should look similar to this listing.

```
$ mkdir btstack
$
```

The `make hci_dump` command assumes that the underlying system is OS X and that PacketLogger is already running. The two`adb` commands, shown in the following Listing, are used to fetch the packet log (hci_dump.pklg) and it can be run on any platform. Wireshark can then be used to open the packet log.

```
$ adb shell su root chmod 666 /data/btstack/hci_dump.pklg
$ adb pull /data/btstack/hci_dump.pklg 2> /dev/null
```

The initial packet log should look like in Figure 2.



9 total (0 Err / 1 HCI / 0 ACL / 0 SCO / 8 Misc)

Time	Type	Decoded Packet
Juli 18 11:22:13.074	NOTE	BTdaemon started
Juli 18 11:22:13.074	NOTE	version 0.8-2682, svn r2682, build 2014-07-17_15:42:39
Juli 18 11:22:13.074	NOTE	config.h: HAVE_RFCOMM
Juli 18 11:22:13.074	NOTE	L2CAP_REGISTER_SERVICE psm 0x1 mtu 65535
Juli 18 11:22:13.074	NOTE	L2CAP_EVENT_SERVICE_REGISTERED status 0x0 psm 0x1
Juli 18 11:22:13.074	HCI EVENT	▶ Undecoded Event Type -
Juli 18 11:22:13.074	NOTE	socket_connection_send_packet_all, connections 0x0
Juli 18 11:22:13.074	NOTE	Socket created at /data/btstack/BTstack
Juli 18 11:22:13.074	NOTE	Server up and running ...

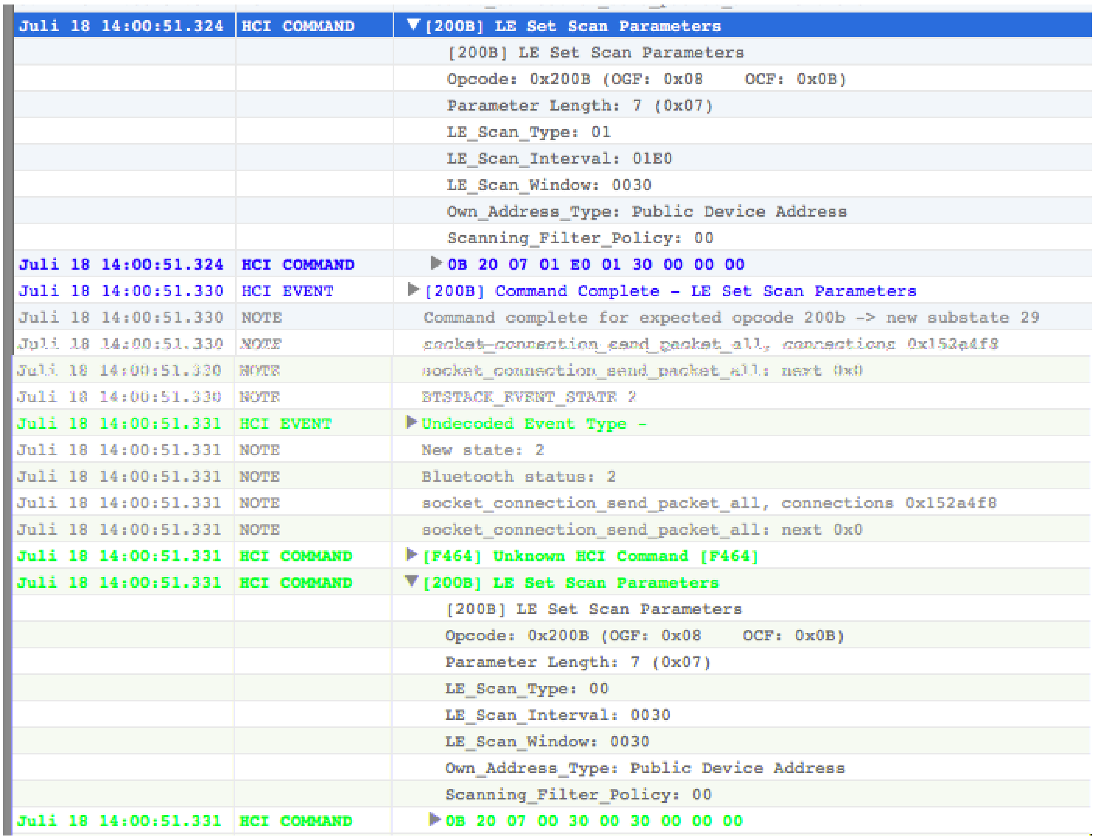
FIGURE 2. hci_dump.pklg right after BTstack daemon was installed.

5. Running the First Example

As first, make sure that BTstack was installed properly and it is running, i.e. by checking the packet log as explained in the previous section. Now, let's do an LE Scan using a test program written in C against libBTstack.

```
$ adb shell
$ le_scan
le_scan started
– connecting to BTstack Daemon
– connected
– send power on
– btstack state 1
– btstack state 2
– start LE scan
– ADV: 3E 0F 02 01 00 01 F2 01 8F 45 16 66 03 02 01 1A B3
```

The `le\_scan` just dumps the data of each received advertisement. It uses the default parameters used by iOS, which are scan window 30 ms, scan interval 300 ms. More on these parameters is in Section 7. The packet log looks like in Figure 3.



```
Juli 18 14:00:51.324  HCI COMMAND  ▼[200B] LE Set Scan Parameters
                                       [200B] LE Set Scan Parameters
                                       Opcode: 0x200B (OGF: 0x08    OCF: 0x0B)
                                       Parameter Length: 7 (0x07)
                                       LE_Scan_Type: 01
                                       LE_Scan_Interval: 01E0
                                       LE_Scan_Window: 0030
                                       Own_Address_Type: Public Device Address
                                       Scanning_Filter_Policy: 00
Juli 18 14:00:51.324  HCI COMMAND      ▶0B 20 07 01 E0 01 30 00 00 00
Juli 18 14:00:51.330  HCI EVENT    ▶[200B] Command Complete - LE Set Scan Parameters
Juli 18 14:00:51.330  NOTE          Command complete for expected opcode 200b -> new substate 29
Juli 18 14:00:51.330  NOTE          socket_connection_send_packet_all, connections 0x152a4f8
Juli 18 14:00:51.330  NOTE          socket_connection_send_packet_all: next 0x0
Juli 18 14:00:51.330  NOTE          BTSTACK_EVENT_STATE 2
Juli 18 14:00:51.331  HCI EVENT    ▶Undecoded Event Type -
Juli 18 14:00:51.331  NOTE          New state: 2
Juli 18 14:00:51.331  NOTE          Bluetooth status: 2
Juli 18 14:00:51.331  NOTE          socket_connection_send_packet_all, connections 0x152a4f8
Juli 18 14:00:51.331  NOTE          socket_connection_send_packet_all: next 0x0
Juli 18 14:00:51.331  HCI COMMAND  ▶[F464] Unknown HCI Command [F464]
Juli 18 14:00:51.331  HCI COMMAND  ▼[200B] LE Set Scan Parameters
                                       [200B] LE Set Scan Parameters
                                       Opcode: 0x200B (OGF: 0x08    OCF: 0x0B)
                                       Parameter Length: 7 (0x07)
                                       LE_Scan_Type: 00
                                       LE_Scan_Interval: 0030
                                       LE_Scan_Window: 0030
                                       Own_Address_Type: Public Device Address
                                       Scanning_Filter_Policy: 00
Juli 18 14:00:51.331  HCI COMMAND      ▶0B 20 07 00 30 00 30 00 00 00
```

Figure 3. HCI dump of the le_scan program.

6. BTstack Java API

BTstack on RugGear/MediaTek provides its own Bluetooth stack and its own interface for using it. While BTstack has been certified for Classic SPP and LE Peripheral, it does not provide a complete Java API for this. At the moment, the functionality to turn Bluetooth on/off, scan for LE devices, connect, and make use of Services and Characteristics is provided. This covers the use of BTstack as LE Central. Please note that the Security Manager is not implemented yet. Use of the Security Manager is necessary when devices require explicit pairing before some services can be used.

The Java classes that make up the BTstack API are split into three folders. Please add all of them to your Android Java project:

(1) `java/src` - main code for BTstack client, i.e., socket communication with BTstack server that runs as a daemon.
(2) `java/android` - Android specific Client/Server communication.
(3) `java/gen` - code for available commands and events, auto-generated from the BTstack C source.

6.1. **BTstack GATT Client for Android example.** More documentation on the Java API is needed. For now, please have a look at the Anrdoid example in `java/example/com/bluekitchen/GATTClientTest.java`. It connects to BTstack Server and turns Bluetooth on. On success, it starts an LE Scan for devices. When it finds a device, it connects to it and queries the available Services. Then, it requests the list of Characteristics for the first service. Finally, it performs some read/write operations on the found Characteristics.

BTstack server runs as a daemon. In the event of a crash, the Java client will get notified and can restart the daemon, to provide a continuous use of the Bluetooth services.

7. BCM20702A0 and RugGear RG500 Reception of Advertisements

During the first tests of BTstack on RugGear/MediaTek, we have seen that it receives less Advertisements than other devices.

Advertisements serve three main purposes:

(1) They provide information that a particular device is in range and turned on.
(2) They can provide some information without connecting to a particular device (e.g. its name, provided services, or even manufacturer specific data).
(3) They allow to connect to a particular device, i.e., an LE device listens for incoming connection requests only a short moment after sending an Advertisement (to save energy).

A setup with one device that sends Advertisements and one device that receives them, can be completely described by three parameters:

(1) the *Advertisement Interval* - the period between two consecutive Advertisements being sent.
(2) the *Scan Interval* - it determines how frequent the receiving device listens for Advertisements.
(3) the *Scan Window* - the time window in which the receiving device listens for Advertisements.

7.1. **Setup.** To evaluate the MediaTek Bluetooth chipsets used in the RugGear RG500, we compared them to a common Bluetooth USB Dual-Mode Dongle that uses the Broadcom BCM20702A0. We used two LE Peripherals. As the first LE Peripheral, we setup another device with the BCM20702A0 and configured it to advertise every 30 ms. This is very often, but it helps to establish how good Advertisements are received, if they occur frequently. As a second more real-world example, we used the nio Tag without further configuration. Based on the measured results, the nio Tag advertises roughly every second.

We measured two cases:

(1) continuous scanning (Scan Window 30ms, Scan Interval 30ms),
(2) intermittent scanning (Scan Window 30ms, Scan Interval 300ms) as performed by iOS.

7.2. **Process.** For the continuous and intermittent scanning measurements on the RugGear, we used the `le_scan` test. We let it run for 15 minutes in continuous, and then 15 min in intermittent scanning setup. For each of the two measurement, we retrieved the packet logs from the device. We processed it an analyzed it using two Python scripts `process_scan.py` and `plot_scan.py`.

The measurements with the BCM module have been carried out with the GATT Browser example of BTstack on OS. Hence, we have up to four experiments for each setup:

- BCM/BCM - BCM20702A0 receives Advertisements from second BCM 20702A0 module
- BCM/nio - BCM20702A0 receives Advertisements from nio tag
- RugGear/BCM - RugGear RG500 receives Advertisements from BCM 20702A0 module
- RugGear/nio - RugGear RG500 receives Advertisements from nio tag

7.3. **Measurements with continuous scanning.** For this experiment with set the Scan Interval to 30 ms and the Scan Windows to 30 ms, which translates to continuous scanning.

The plot in Figure 4 shows how many Advertisements have been received for both receivers from the two senders. In the BCM/BCM combination, 27 out of max 33 (1s /30 ms) Advertisements have been received. On the RugGear device, only about 1 in 15 Advertisements have been received.

Figure 5 shows the distribution of the time between two received Advertisements over the full range.

All setups but the RugGear vs. nio Tag, received an Advertisement every second, which is sufficient for most interactive applications. We further analyzed the RugGear vs. nio Tag setup in more detail. For this setup, we calculated the expected delay between the start of a scan and the first received Advertisement based on the previous measurements as shown in Figure 7. An alternative and easier to read representation of the same is given in Figure 8. From this Figure we can, for example, see that we have 20 percent probability of receiving an Advertisement with less then 1 second.

7.4. **Measurements with normal scanning.** For this experiment with set the Scan Interval to 300 ms and the Scan Windows to 30 ms, which translates to scanning for 1/10 of the time. As the scanning time is reduced by a factor of 10, we also expect the number of received Advertisements to be lower by 1/10. We didn't had the nio Tag around this time.

The plot in Figure 9 shows the number of Advertisements received for both receivers. The BCM/BCM received 2.9 Advertisements per second (adv/s) which is roughly 1/10 of the 27 adv/s received in the continuous scanning experiment. For the RugGear/BCM combination, we received 0.6 adv/s, which is roughly 1/3 of the 1.8 adv/s received in the continuous scanning experiment and is unexpected.

We conclude from this data, that the MediaTek chipset didn't do continuous scanning in the first experiment. Instead, it looks like it was scanning for only 1/3 of the time, potentially due to internal resource limitations or scheduling problems. In the intermittent scanning setup, the RugGear device received 1/5 of the Advertisements received by the BCM chipset.

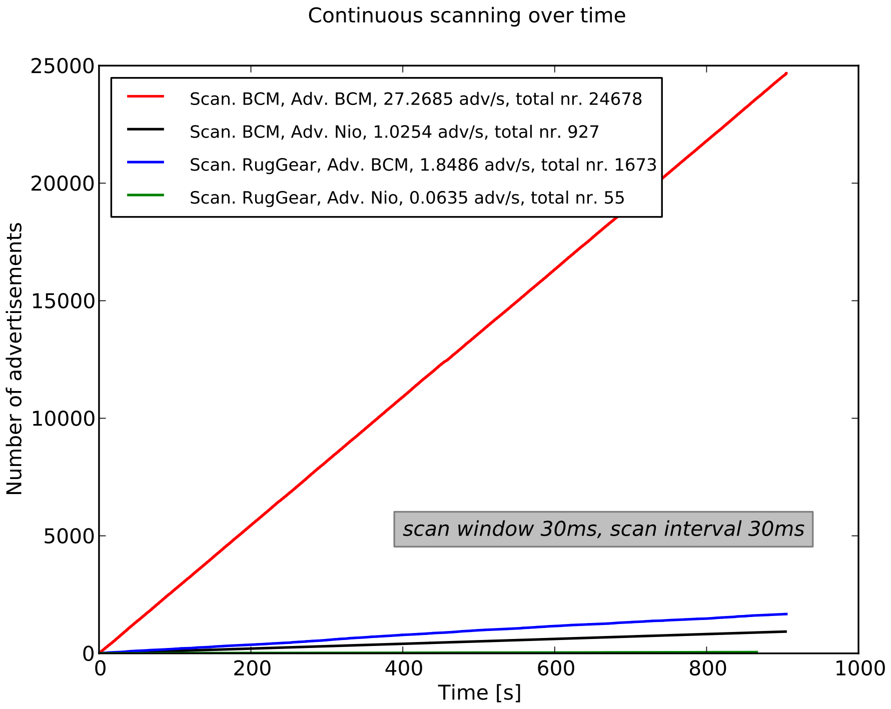


FIGURE 4. Advertising reports accumulated over time, continuous scanning.

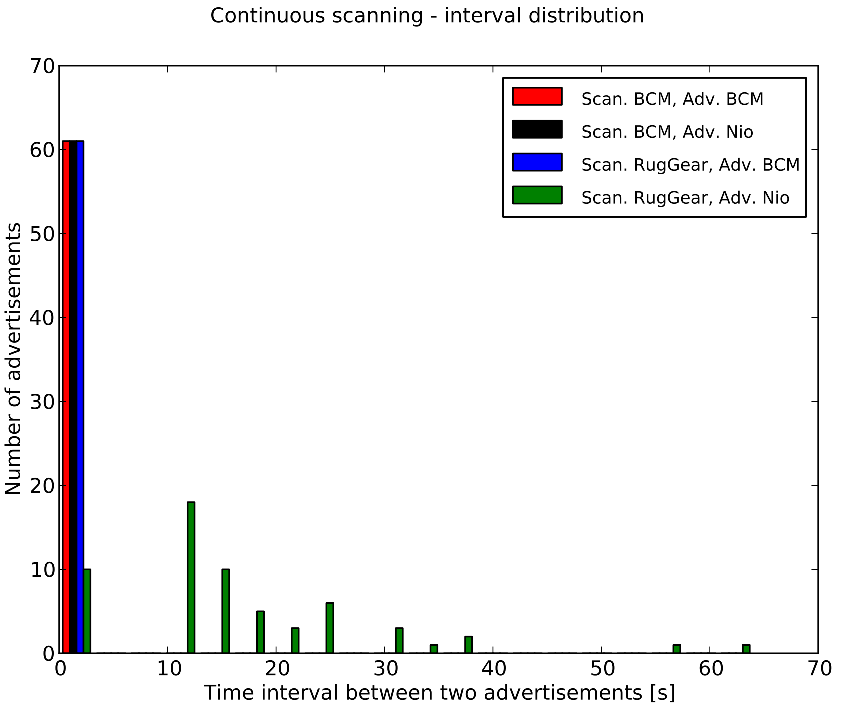


FIGURE 5. Time delay histogram, continuous scanning.

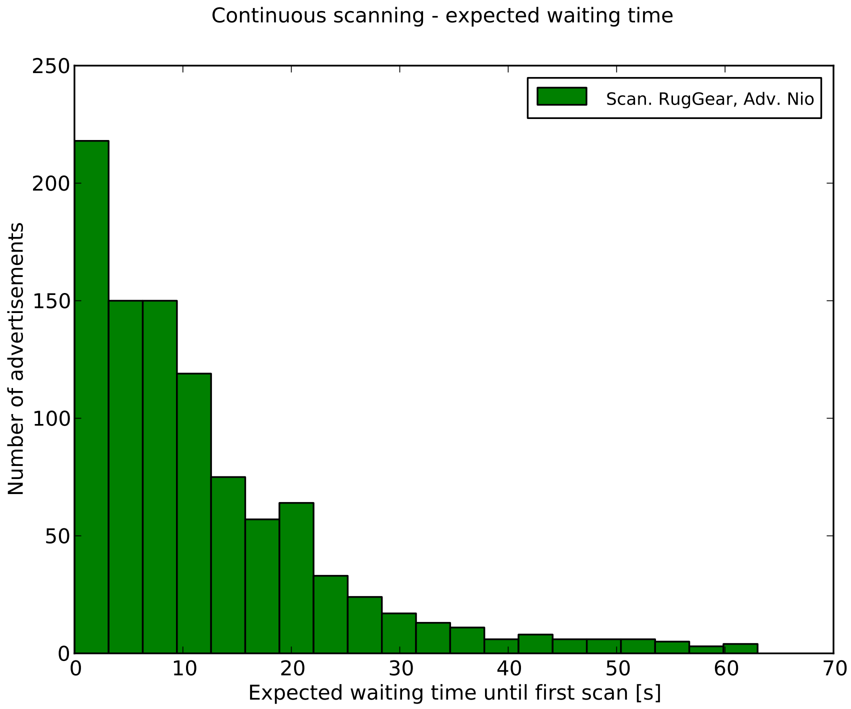


FIGURE 6. Expected time until first Advertisement, continuous scanning.

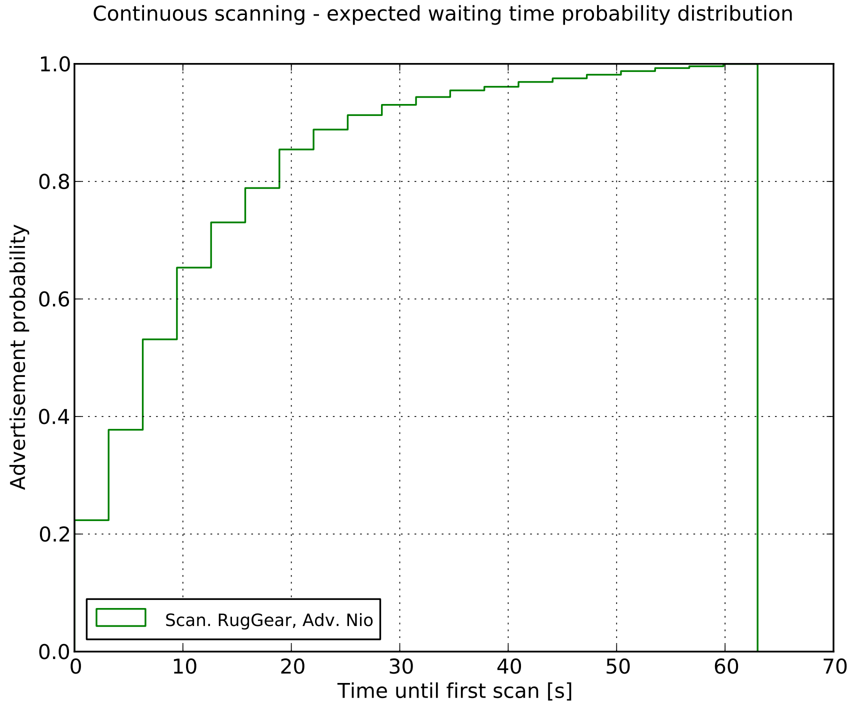


FIGURE 7. Cumulative distribution of expected time until first Advertisement, continuous scanning.

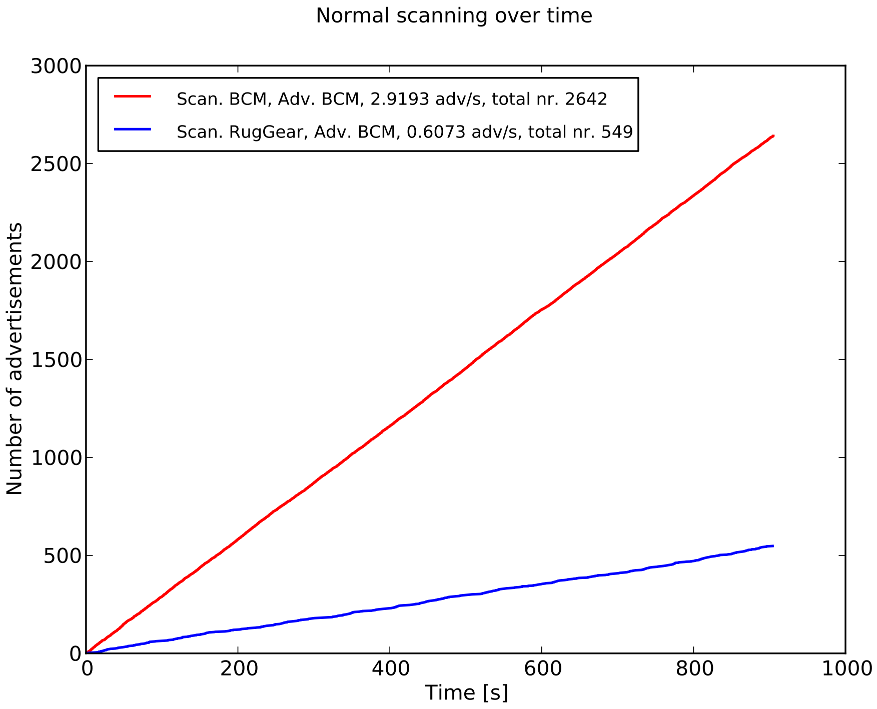


FIGURE 8. Advertising reports accumulated over time, normal scanning.