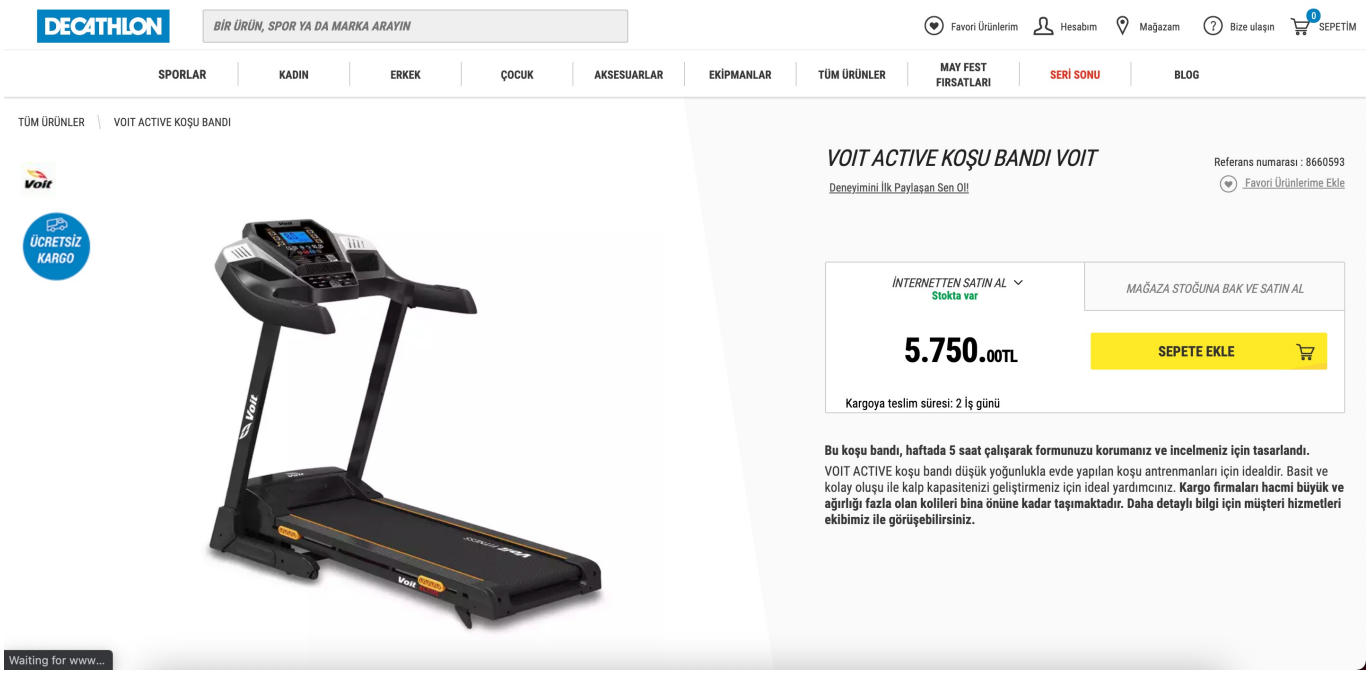
Run Mert Run

written by Mert SARICA | 1 June 2022

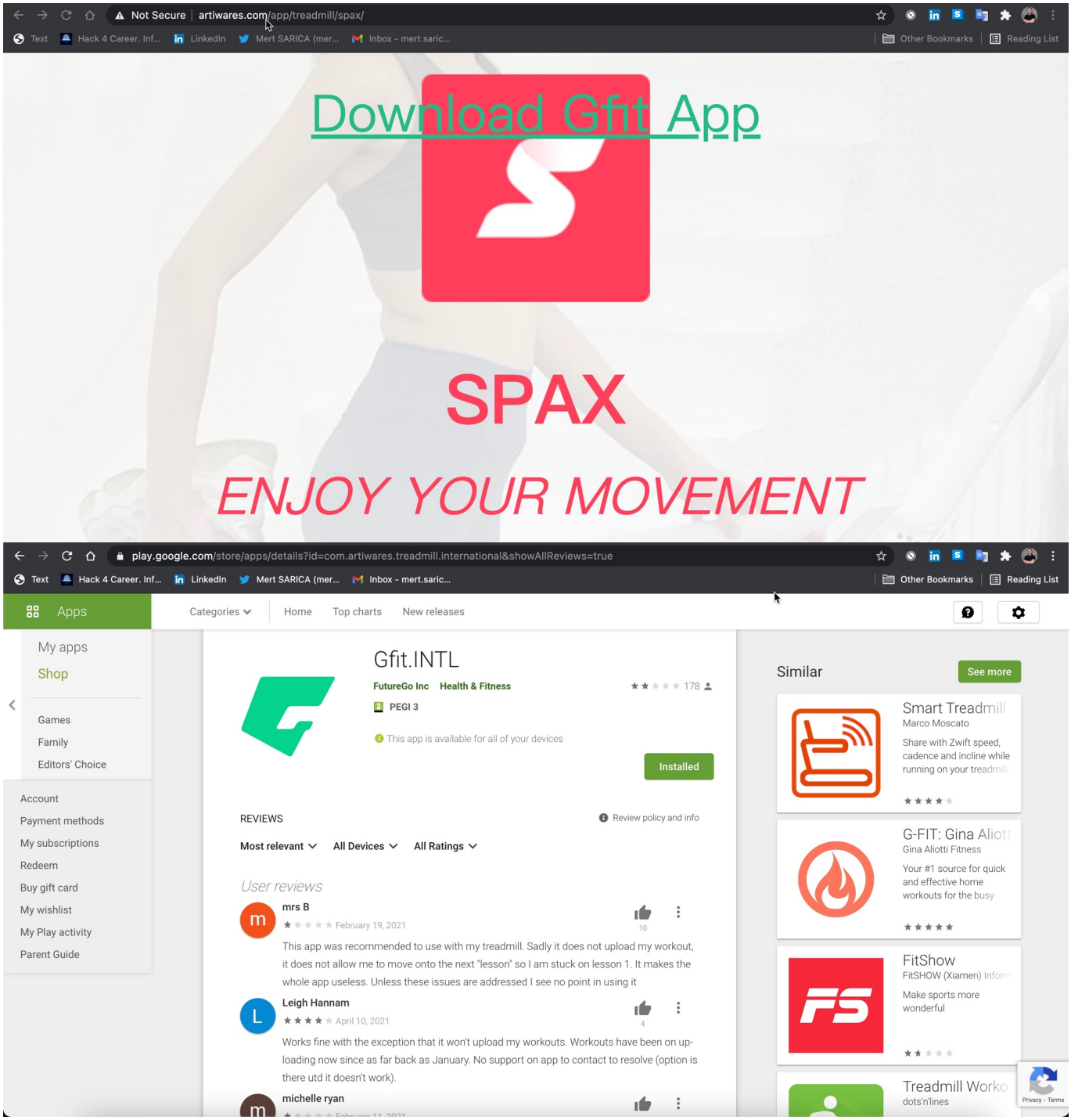
Starting in 2020, due to the increasing impact of the Covid-19 pandemic in our country, I began doing sports with my sports coach through WhatsApp instead of going to the gym. Over time, my coach directed me to purchase various sports equipment such as a pull-up bar, weight set, and bench, and due to the ongoing pandemic, I had to expand to include a treadmill. Finally, the Voit Active treadmill with Bluetooth function took its place among my sports equipment.



As a security researcher who has tried to hack various electronic devices that I have purchased (such as “Don’t Just Say Printer!“, “It’s a Bird… It’s a Plane… It’s Drone!“, “and you, CPCR-505?“, “Spy Mouse“, “Escape from Imprisonment“), it did not occur to me to apply disproportionate power to the innocent treadmill. However, as the time I spent walking on the treadmill increased, the QR code on the treadmill’s panel began to catch my attention.

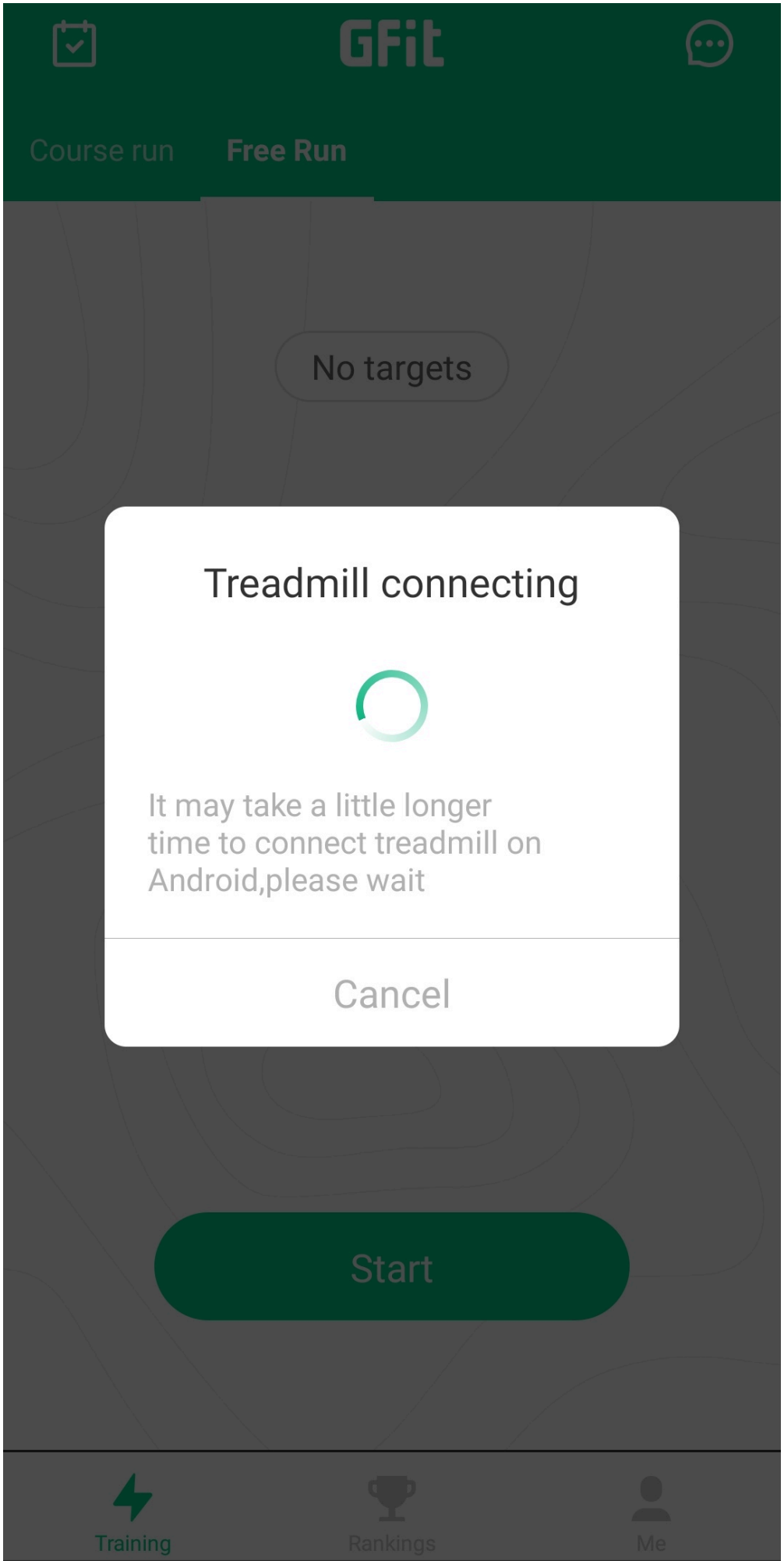


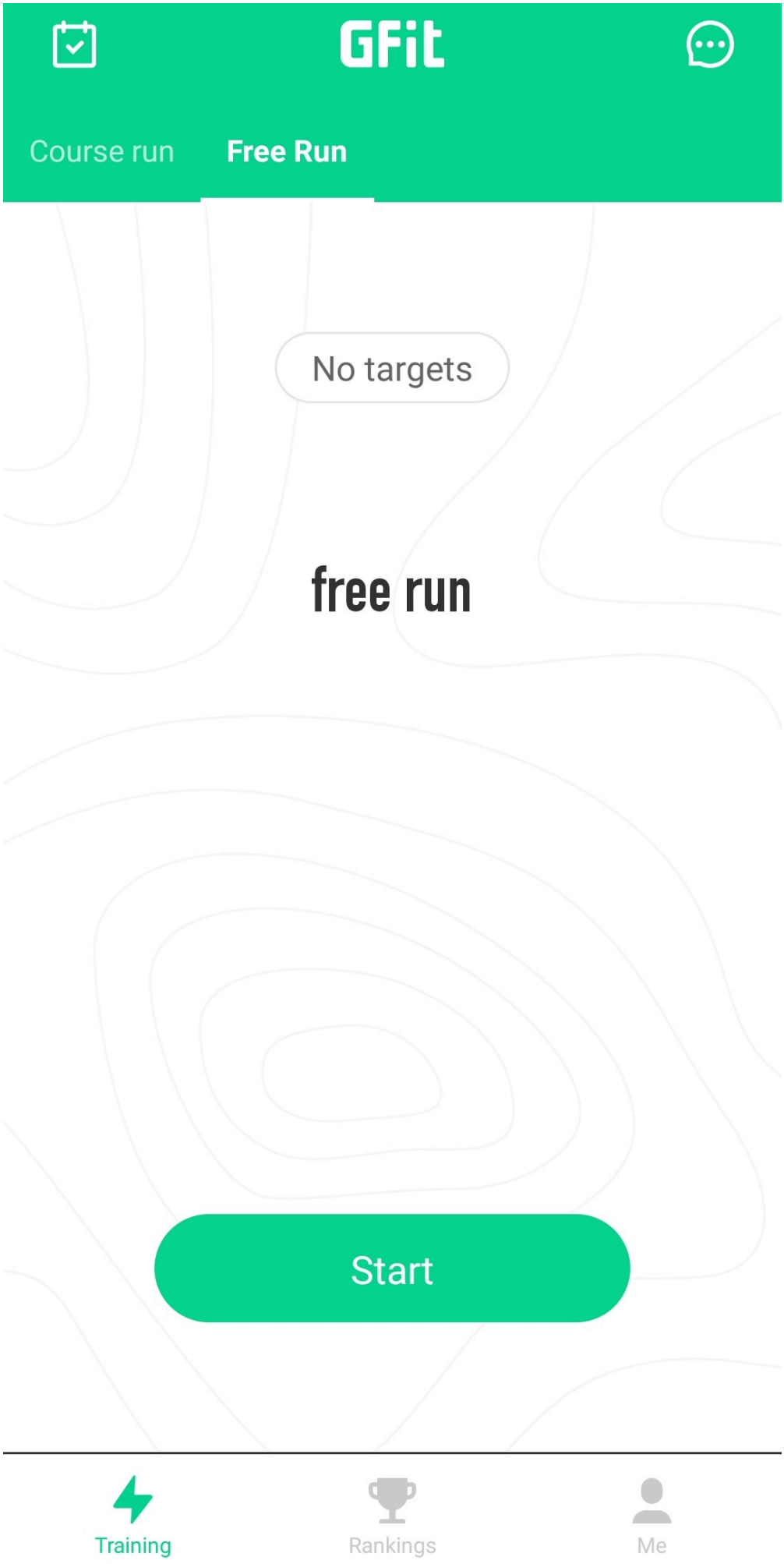
When I scanned the QR code with an app, I was directed to the address http://www.artiwares.com/app/treadmill/spax/, and from there to the page of the Gfit.INTL application on Google Play, which had many negative reviews and was developed by an unknown Chinese company. Since the treadmill supports Bluetooth, I decided to install the app, examine what commands could be sent, and identify potential malicious use scenarios.



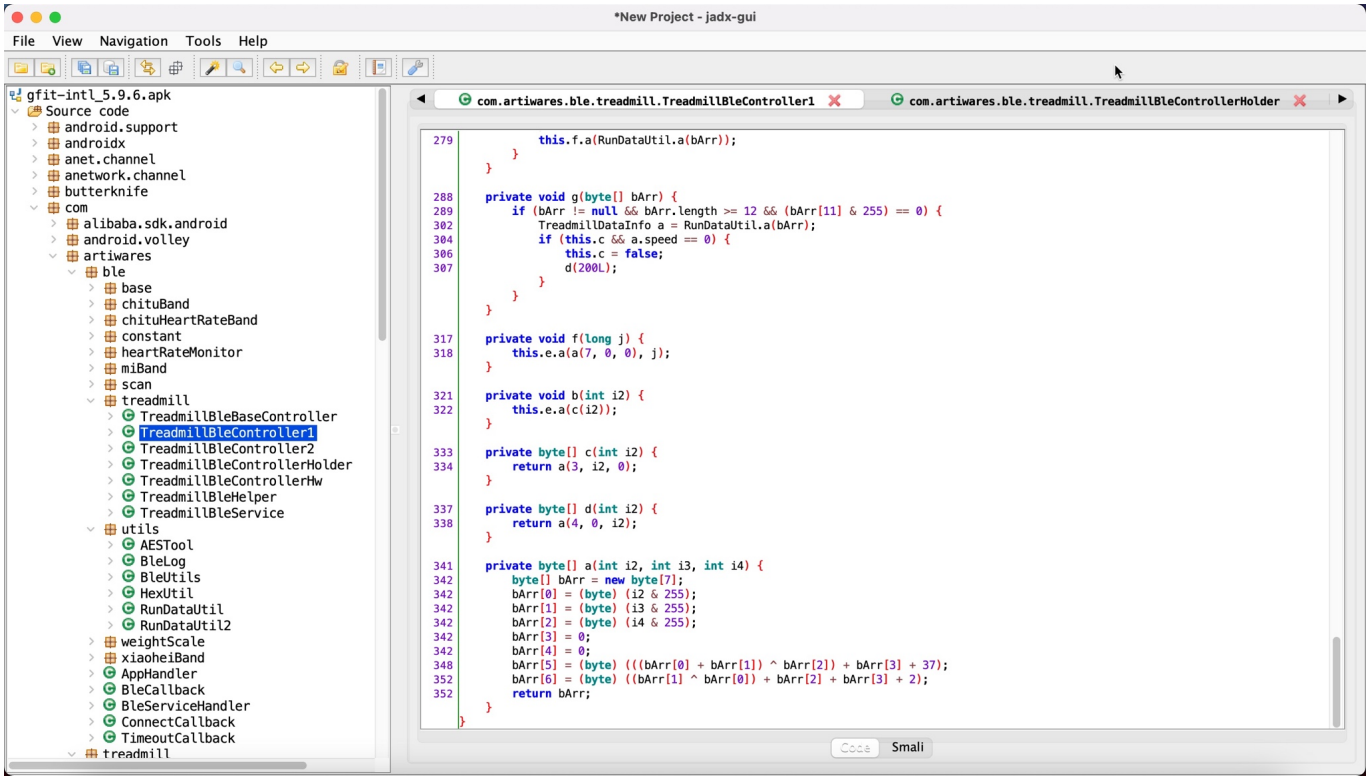
During this research, I was aware that I would not be able to benefit from emulator-based dynamic analysis and various resources, as my favorite tool, the Genymotion emulator, does not support Apple M1 processors with macOS, as I mentioned in my article “Hooking on Android”

After installing the Gfit application on my phone and pairing it with my RunnerT treadmill, which has Bluetooth name, I saw that I could easily perform the basic functions of the treadmill, such as starting, increasing speed, decreasing speed, and stopping, through the app.


GFit
Course run
Free Run
No targets
Treadmill connecting
It may take a little longer time to connect treadmill on Android,please wait
Cancel
Start
Training
Rankings
Me


GFit
Course run
Free Run
No targets
free run
Start
Training
Rankings
Me

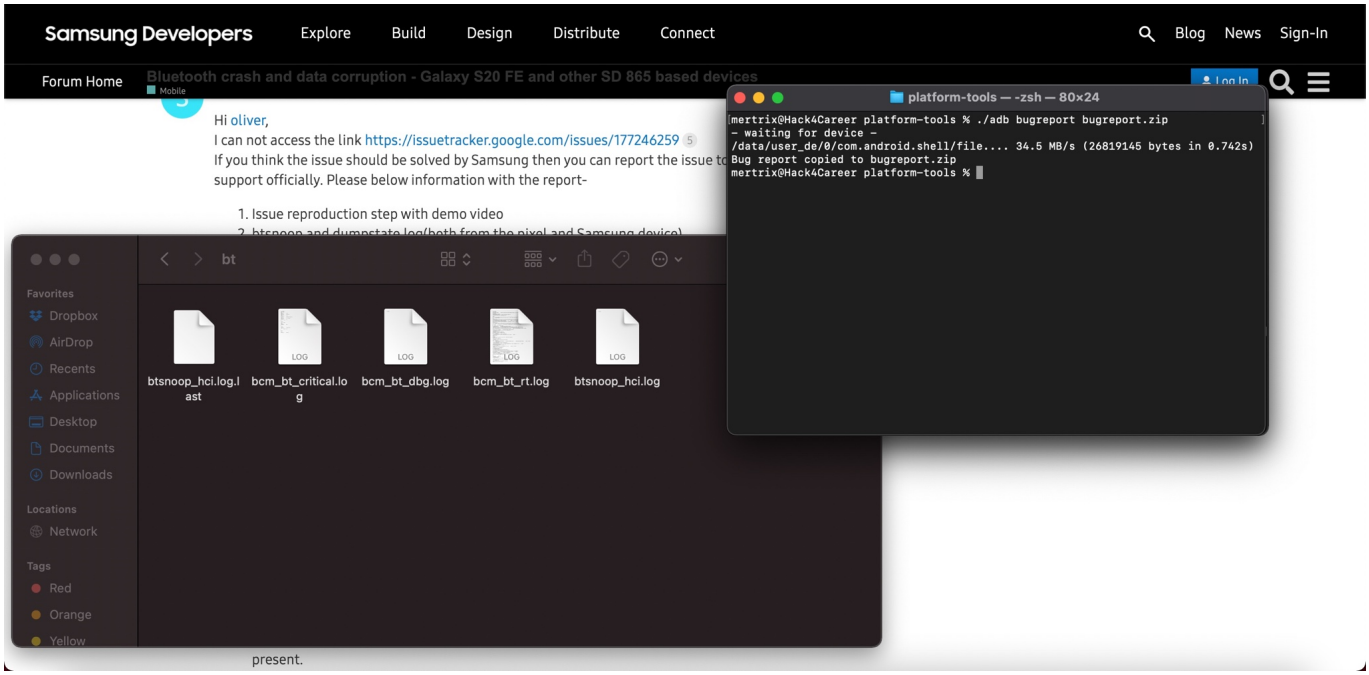
To learn about the commands sent to the treadmill by the app, I had to choose static code analysis or use the phone on which the app was installed. Since static code analysis seemed more practical, I started to examine the Gfit app with the jadx tool.



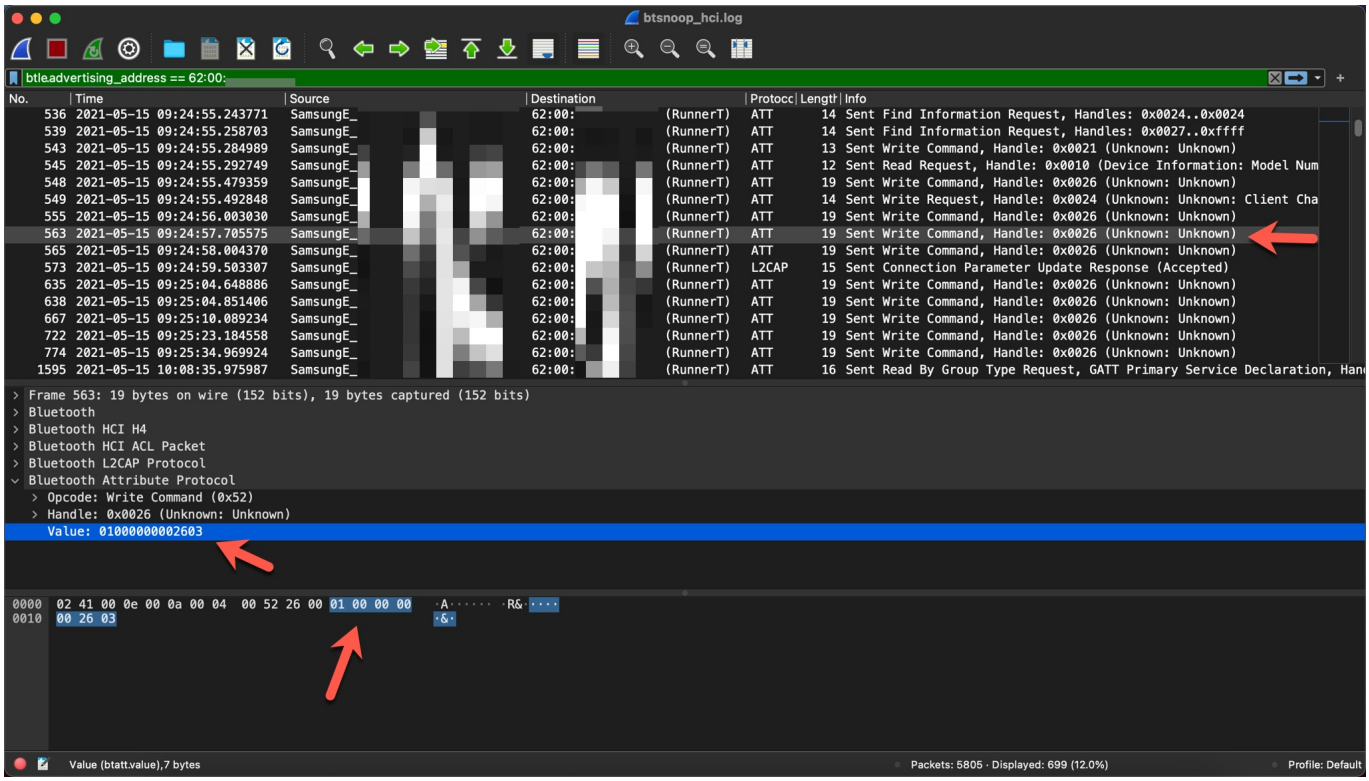
Since the code is hidden (obfuscated) throughout the app and I had no chance of doing dynamic code analysis on the emulator under the current conditions, I immediately gave up and saw what I could do with the phone.

I began to follow the steps in response to a message from someone experiencing a problem with Bluetooth packet sources on Samsung’s support page.

When I reached step 6, I sent start, increase speed, decrease speed, and stop commands to the treadmill through the Gfit application, and then moved on to the other steps and began analyzing the btsnoop_hci.log file with Wireshark.



When I looked at the first command that reached the treadmill using the filter btle.advertising_address == 62:00:a1:18:b5:22 on WireShark, I saw the value 01000000002603, which is the command to start the treadmill.



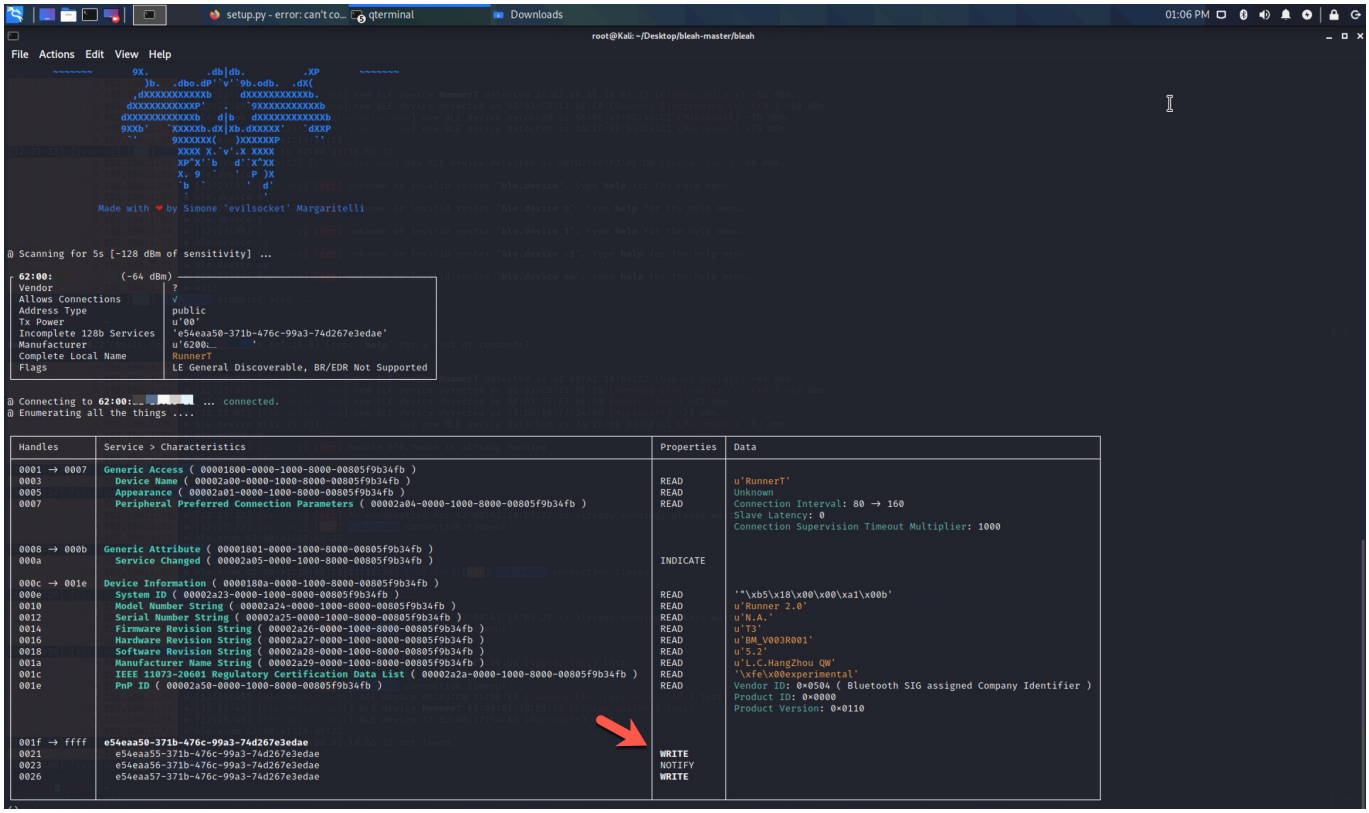
Next, I sent the treadmill the commands Hold/Pause (05000000002a07), set speed to 9 km/h (035a000000825b), set speed to 5.2 km/h (03340000005c39), and Stop (02000000002704) and saw the other values in parentheses on WireShark.

First, I decided to find out if the treadmill was vulnerable to a replay attack. To do this, I had to decide on the USB device and tool that would

send the Bluetooth packets that I had obtained for the BLE-supported treadmill. For the USB device, the Parani-UD100 came to my aid, as I had used it in my article titled “The Blue Threat”

It was time to find a tool to send packets, and I decided to proceed with bleah among the gatttool, bleah, and nRF Connect tools.

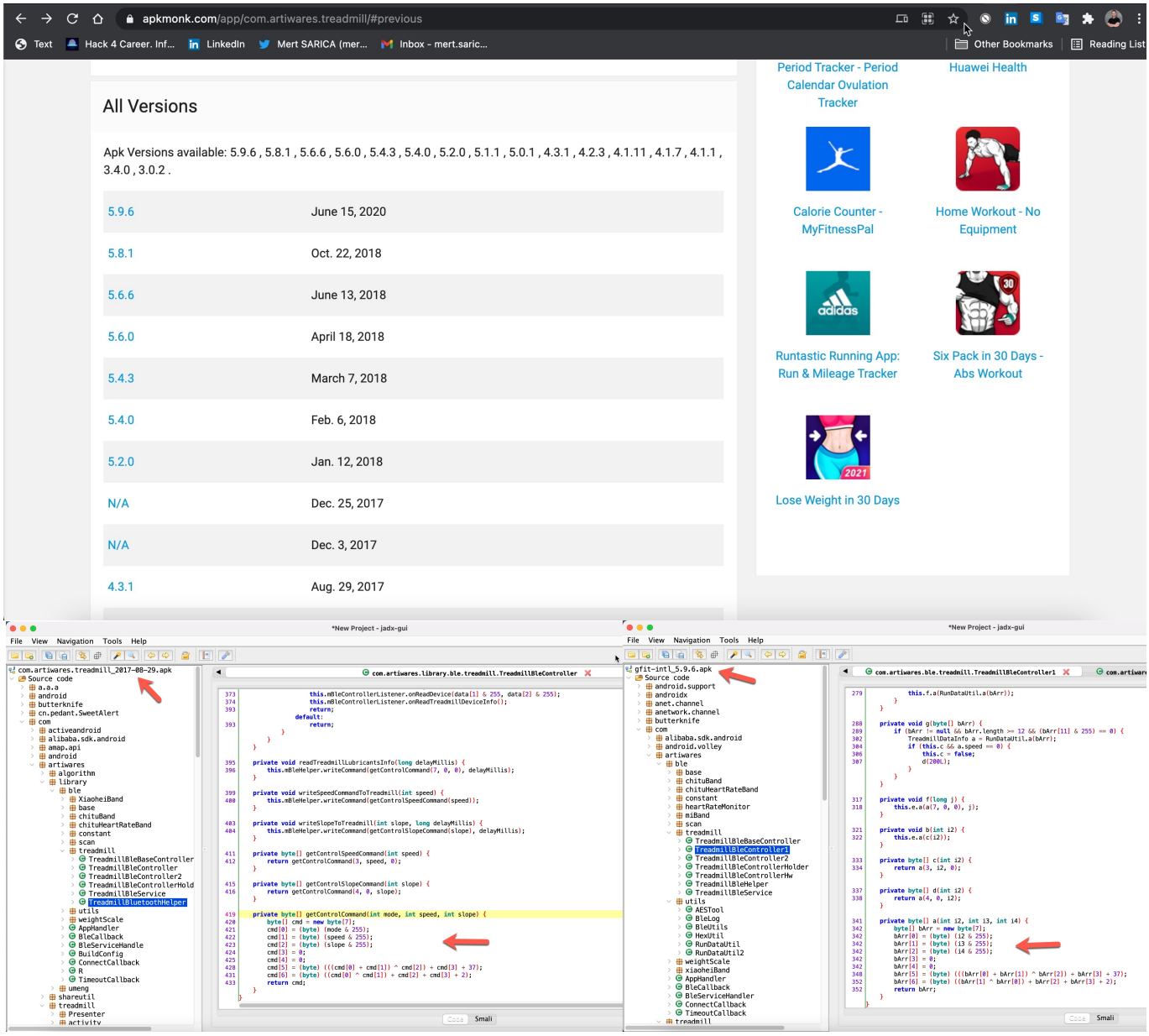
On the Kali operating system, I quickly listed the Services and Characteristics that I would need to send packets to the treadmill using the Generic Attribute Protocol (GATT) with the bleah -b “62:00:xx:xx:xx:xx” -e command.



After seeing the WRITE feature in the e54eaa57-371b-476c-99a3-74d267e3edae characteristic information, I sent the bleah -b “62:00:xx:xx:xx:xx” -u “e54eaa57-371b-476c-99a3-74d267e3edae” -d “0x01000000002603” command to the treadmill using bleah and saw that the treadmill started!

At this point, after learning that the treadmill was vulnerable to replay attacks, I decided to find out how these commands were constructed. Since the source code was hidden using the obfuscation technique and I was unable to perform dynamic code analysis, I downloaded and examined older versions of the Gfit application, and soon found that the obfuscation technique was not used in the 2017 version. When I compared the source code from 2020 and 2017

side by side, it was very easy for me to learn how the commands sent from the Gfit application to the treadmill were constructed.

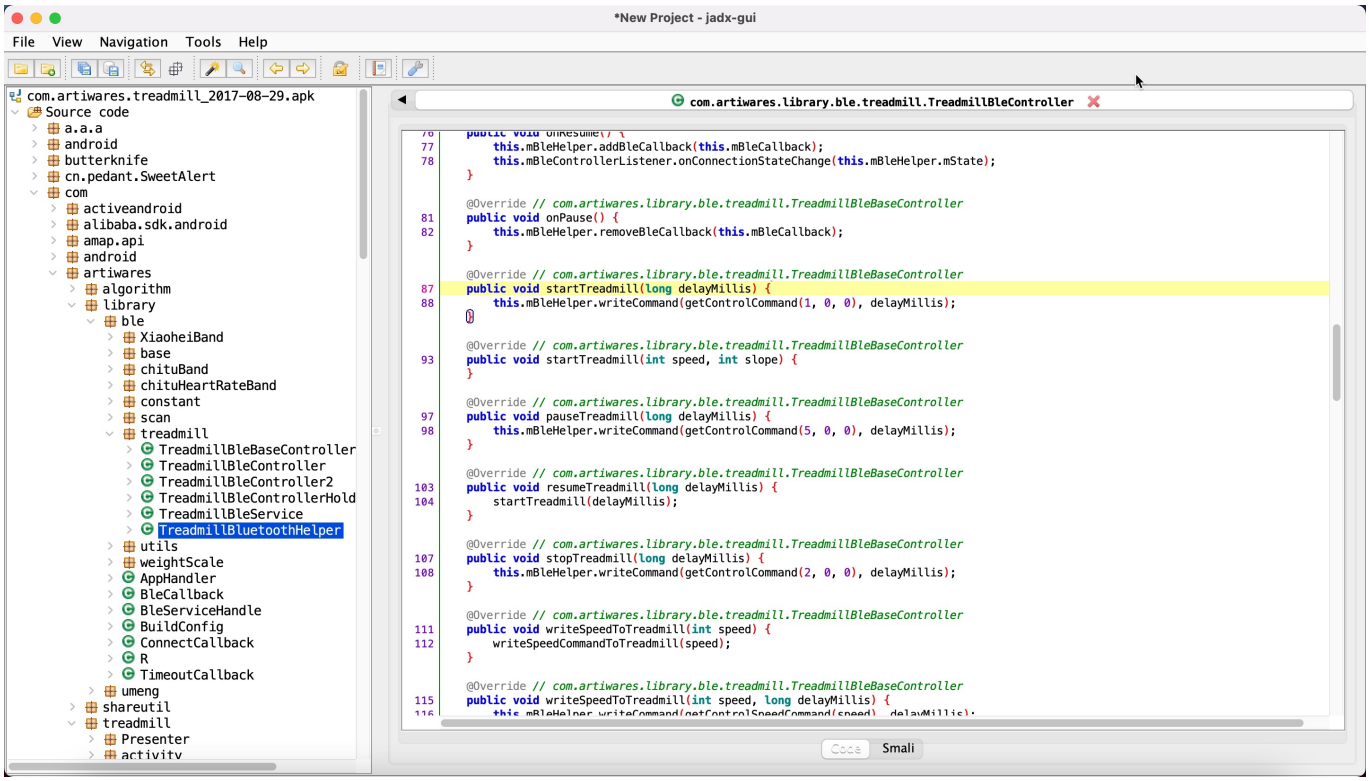


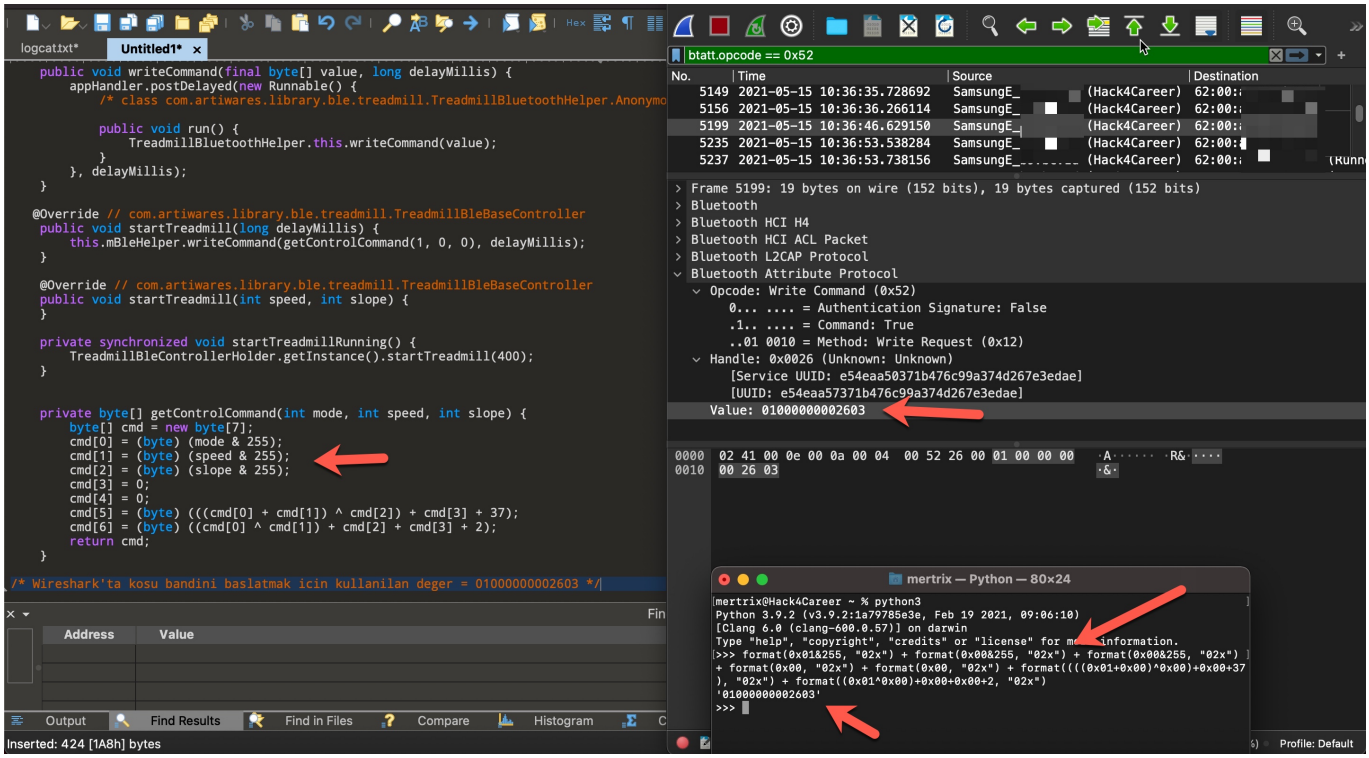
When I quickly looked at the source code, the startTreadmill(long delayMillis) function was called to start the treadmill, then the mBleHelper.writeCommand(getControlCommand(1, 0, 0), delayMillis) function was called, and finally, the getControlCommand(int mode, int speed, int slope) function was called, which creates the 7-byte packet to be sent to the treadmill.

```
private byte[] getControlCommand(int mode, int speed, int slope) {
byte[] cmd = new byte[7];
cmd[0] = (byte) (mode & 255);
```

```
cmd[1] = (byte) (speed & 255);
cmd[2] = (byte) (slope & 255);
cmd[3] = 0;
cmd[4] = 0;
cmd[5] = (byte) (((cmd[0] + cmd[1]) ^ cmd[2]) + cmd[3] + 37);
cmd[6] = (byte) ((cmd[0] ^ cmd[1]) + cmd[2] + cmd[3] + 2);
return cmd;
}
```

I was able to create the output 01000000002603 by using the getControlCommand function with the variables mode = 1, speed = 0, and slope = 0 using Python. This way, I was able to generate all the commands from starting the treadmill (getControlCommand(1, 0, 0)) to increasing speed (getControlCommand(3, 5, 0)) using Python and send them to the treadmill using the Parani-UD100 and the bleah tool without the Gfit application.





When I came to try out the malicious use scenarios that came to mind,

1. I tried to increase the speed of the treadmill, which is 14 km/h, to 20 km/h (bleah -b “62:00:xx:xx:xx:xx” -u “e54eaa57-371b-476c-99a3-74d267e3edae” -d “0x03c8000000f0cd”) and found that it could only be set to a maximum of 14 km/h. (good news)
2. In the second scenario, when I sent the command to stop the treadmill while it was running at a speed of 14 KM per hour (bleah -b “62:00:xx:xx:xx:xx” -u “e54eaa57-371b-476c-99a3-74d267e3edae” -d “0x02000000002704”), I saw that the treadmill slowed down to 0 KM in the time it took to reach its current speed (if it was running at 14 KM per hour, it stopped in 14 seconds). However, when I sent the same packet twice in a row, it slowed down in 4-5 seconds and I thought that such a sudden drop in speed while running at high speeds could lead to an accident for the runner. (bad news)
3. Finally, when I sent a command to the treadmill to increase the speed of a person walking or running at low speed to 14 KM per hour in an infinite loop, as shown in the video below, I saw that the person tried to decrease the speed in a panic but failed, and therefore the possibility of an accident occurred. And when I think about a malicious person going to a gym that has purchased and made these treadmills available for use and sending this command to all the treadmills in order, I didn’t even want to imagine the crisis that would occur. (bad news)

In conclusion, I highly recommend considering the risks before purchasing or using this treadmill developed by an unknown Chinese company that cannot turn off its Bluetooth function. Hope to see you in the following articles.