

Report on Qubo IoT Device Vulnerability

Ayyappan Rajesh
University of Massachusetts Dartmouth
Justin Justin

July 3, 2023

1 Introduction

As we step further into the interconnected era of the Internet of Things (IoT), the security implications of these devices come to the forefront of the discussion. This comprehensive study zeroes in on the Qubo IoT doorbell, a pioneering product designed and manufactured in India. This device, however, is not exempt from the security vulnerabilities that often plague IoT devices, as highlighted by the issue tracked as CVE-2023-22906.

Navigating the intricacies of IoT security, we aim to throw light on the consequences of this specific vulnerability for the end-users, illustrating the potential threats and breaches to their digital safety. Our investigation reaches beyond the identification of these risks, striving to propose strategic and effective countermeasures.

Underpinning this exploration is an emphasis on the critical need for stringent security protocols in the IoT landscape. By assessing their incorporation during the essential stages of IoT development and deployment, we highlight the protective role they play in safeguarding digital ecosystems.

By unveiling the intricacies of this particular vulnerability, this study offers essential insights into IoT security. Beyond diagnosing potential pitfalls, it paves the way for developers, manufacturers, and end-users to proactively engage with these threats, thereby fostering a more secure digital environment.

2 Background

IoT is a network of interconnected devices that can communicate and exchange data over the internet without human intervention. This technology has seen a rapid increase in adoption across various industries, from healthcare to smart homes, due to the promise of increased efficiency, convenience, and improved decision-making. However, the proliferation of IoT devices has also introduced significant security risks, including data breaches, malware attacks, and unauthorized access.

Given the growing adoption of IoT technologies and the potential risks associated with their use, it is critical to conduct thorough security assessments and implement robust security measures in the development and deployment of these devices. The following sections of this research paper will provide a detailed analysis of the Qubo IoT doorbell vulnerability and its potential impact on users, as well as recommendations for mitigation.

The Qubo IoT doorbell is a device that brings advanced (sometimes, not so advanced) security features, including facial recognition, two-way audio, motion detection, and more, to households across India. These features were once only available in high-end products due to their high cost. However, the Qubo IoT doorbell's affordability has made these features accessible to every household in India, making it a popular choice among homeowners. Nevertheless, a significant security vulnerability assigned CVE-2023-22906 has been discovered, which could compromise the confidentiality, integrity, and availability of user data and the device itself.

According to a market research report published by Markets and Markets, the global IoT security market size is expected to grow from USD 8.2 billion in 2018 to USD 35.2 billion by 2023, at a CAGR of 33.7 percent during the forecast period¹. Additionally, Smart home security camera shipments in India grew 48 year-over-year in Q1 2023 according to Counterpoint Research.²

¹<https://www.marketsandmarkets.com/PressReleases/iot-security.asp>

²<https://www.counterpointresearch.com/indias-smart-home-security-camera-shipments-48-yoy-q1-2023/>

3 Discovery of the Vulnerability

During a visit to India, one of our researchers stumbled upon the Qubo doorbell and, in a playful attempt to trick their neighbor, managed to capture the RF signal of the device using a Flipper Zero. The subsequent replaying of this signal triggered the neighbor's chime unit to ring. Intrigued by this incident, we joined forces to scrutinize the device's architecture, aiming to thoroughly understand and evaluate the security standards of this Made in India technology.

Before embarking on a physical disassembly of the device, we opted for an initial virtual exploration. We conducted a comprehensive NMap scan, which revealed a surprising vulnerability - the device's TELNET was not only open but lacked password protection, effectively offering root access to the device. This discovery underscored the importance of our investigation into the device's security framework.

4 Technical Details

4.1 Reconnaissance

Device Specifications

- Firmware : HCD01_02.V1.38_20220125
- Operating System : Linux 4.9.84
- Processor : Cortex A7
- Architecture : ARM v7l 32-bit

Nmap scan

Below are the results of the Nmap scan, indicating that port 23(TELNET) is open

```
$ Nmap scan report for 192.168.68.126
Host is up (0.025s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
8080/tcp  open  http-proxy?
MAC Address: 14:C9:CF:XX:XX:XX (Unknown)
Nmap done: 1 IP address (1 host up) scanned in 2.46 seconds
```

4.1.1 Reverse Engineering QR Code for Wi-Fi Pairing

It was observed that during the setup process, the user is required to use the Qubo application on their smartphone to pair the Qubo device with their Wi-Fi network using a QR code generated by the app. The QR code is scanned by placing the phone screen in front of the camera during the setup process. It was also noted that the Wi-Fi settings can be reset at any time by inserting a sim card tool in the reset slot on the Qubo. This will activate Wi-Fi pairing mode and allow anyone to display the appropriate QR code to connect the device to the network.

The format for the QR code is as shown below.

```
{"a":1,"w":"SSID","p":"password","z":"UTC time"}
```

For instance, an example of pairing it to a Wi-Fi with SSID "Qubo" which has the password "password" and is located in Boston would be

```
{"a":1,"w":"Qubo","p":"password","z":"-4"}
```

4.2 Exploitation

After successfully establishing a TELNET connection to the Qubo device, an attacker with malicious intent could potentially use this access to remotely execute code on the device. This could involve exploiting vulnerabilities or weaknesses in the device's software or configuration, allowing the attacker to gain further access or control over the device.

4.2.1 Qubo Scanner

A simple python3 script that was built to scan for the device on a local network and connect to it. This script can be modified to also act as a payload delivery mechanism, where the payload could be anything from a simple reverse shell to the Mirai malware.

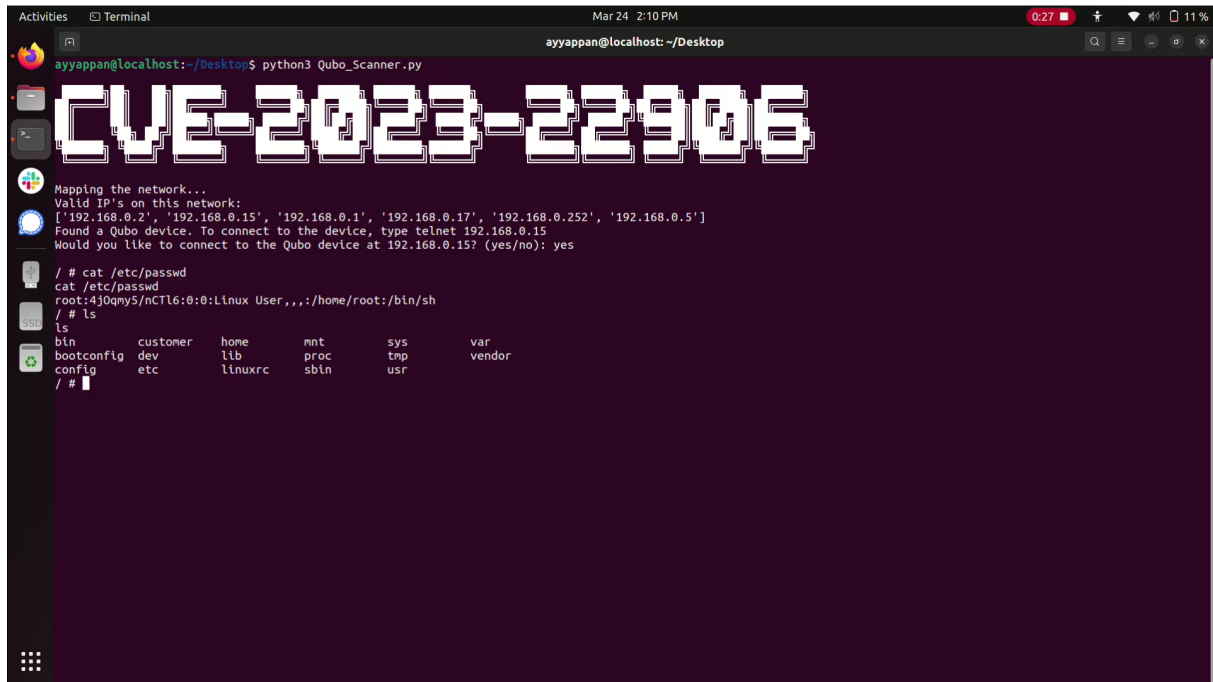


Figure 1: PoC of /etc/passwd

4.3 Post Exploitation

Once the exploitation phase is complete, the post exploitation phase focuses on gathering sensitive data, maintaining persistence, and most importantly, changing the chime of the doorbell!

4.3.1 Reverse Shell

Motivated by the intention to test the device's ability to connect to external command-and-control (C2) servers that can remotely execute commands, effectively turning the device into a potential cog in a botnet or exposing it to over the internet, rather than limiting control to those on the local network. A reverse shell is a type of shell³ that enables attackers to gain remote access to a compromised system, giving them complete control over the system and allowing them to carry out malicious activities.

IoT devices typically have limited processing power, memory, and storage compared to traditional Linux machines. They are also designed to run only specific programs needed to perform their intended functions. These constraints present a unique set of challenges when attempting to execute a reverse shell on an IoT device, thus underlining the significance of our testing.

To test a reverse shell on an IoT or embedded device, tools such as Metasploit may be used, which includes a wide range of exploits and payloads for various targets. Alternatively, tools like Netcat can be used to establish a connection with the device and execute commands remotely.

³a command-line interface that allows users to interact with the operating system of a computer.

4.3.2 Netcat

Netcat, often hailed as the 'TCP/IP Swiss Army Knife', is a flexible networking utility capable of reading and writing data across network connections using TCP or UDP protocols. Its vast array of capabilities justifies the Swiss Army Knife metaphor, given its ability to handle a multitude of network-related tasks, thus making it a versatile tool in the realm of networking.

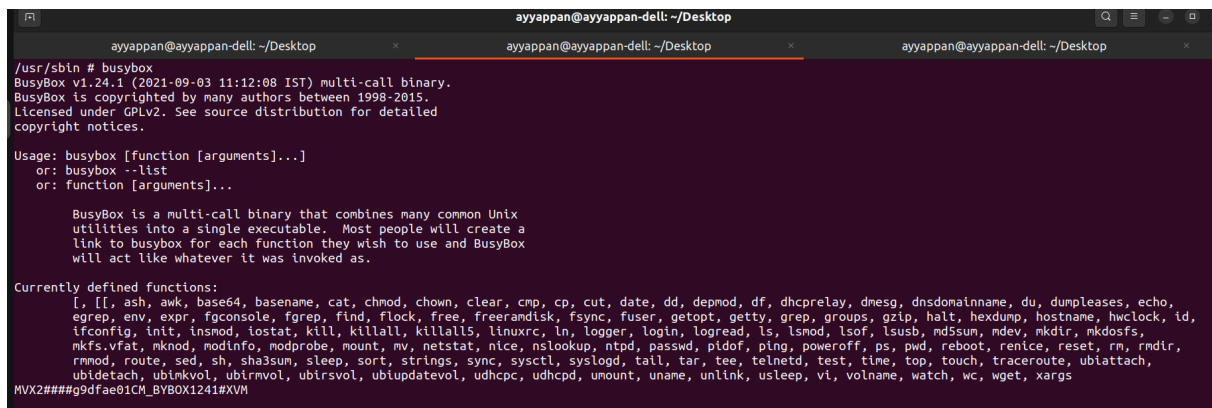
The value of Netcat is particularly highlighted in its ability to create a reverse shell. This feature is of great importance in system penetration testing and security auditing, as it allows for remote command execution, which in turn provides users with a shell environment on the target system. This further underscores Netcat's crucial role in comprehensive system validation and security assessments.

4.3.3 Netcat on Busybox

Embedded Linux systems, like the Qubo IoT device, often have to work within limited functionalities to manage resource utilization effectively. However, in order to expand the capabilities of these devices and provide crucial tools like netcat and FTP, a software suite like BusyBox is incorporated.

BusyBox, widely recognized as the Swiss Army Knife of Embedded Linux, is an extremely versatile collection of Unix utilities combined into a single executable file. This configuration makes it an excellent fit for embedded systems and is a common feature in streamlined Linux distributions. Its wide array of functions, combined with its widespread use, has cemented its position as a preferred tool in the embedded software development arena.

However, the version of BusyBox binary present on the target device did not have Netcat as it was not included by the developer during compilation. This can present a challenge if commands like nc (netcat) are not available, which would be highly beneficial for a tester or an attacker. To overcome this, a precompiled BusyBox binary can be obtained from the official site. This version includes all the additional features that could aid in thoroughly testing the device.



```
ayyappan@ayyappan-dell: ~/Desktop
/usr/sbin # busybox
BusyBox v1.24.1 (2021-09-03 11:12:08 IST) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
or: busybox --list
or: function [arguments]...

BusyBox is a multi-call binary that combines many common Unix
utilities into a single executable. Most people will create a
link to busybox for each function they wish to use and BusyBox
will act like whatever it was invoked as.

Currently defined functions:
[I, [f, ash, awk, base64, basename, cat, chmod, chown, clear, cmp, cp, cut, date, dd, depmod, df, dhcprelay, dmesg, dnsdomainname, du, dumpleases, echo,
egrep, env, expr, fgconsole, fgrep, find, flock, free, freerandisk, fsync, fuser, getopt, getty, grep, groups, gzip, halt, hexdump, hostname, hwclock, id,
ifconfig, init, insmod, iostat, kill, killall, killall5, linuxrc, ln, logger, login, logread, ls, lsmod, lsof, lsusb, md5sum, mdev, mkdir, mknod, mkfs.vfat,
mknod, modinfo, modprobe, mount, mv, netstat, nice, nslookup, ntpd, passwd, pidof, ping, poweroff, ps, pwd, reboot, renice, reset, rm, rmdir,
rmmod, route, sed, sh, sha3sum, sleep, sort, strings, sync, sysctl, syslogd, tail, tar, tee, telnetd, test, time, top, touch, traceroute, ubiattach,
ubidetach, ubienvol, ubirmvol, ubirsvol, ubiupdatevol, udhcpc, udhcpd, umount, uname, unlink, usleep, vi, volname, watch, wc, wget, xargs
MVX2###g9dfe81CM_BYBOX1241#XVM
```

Figure 2: Default BusyBox on the Qubo with limited commands

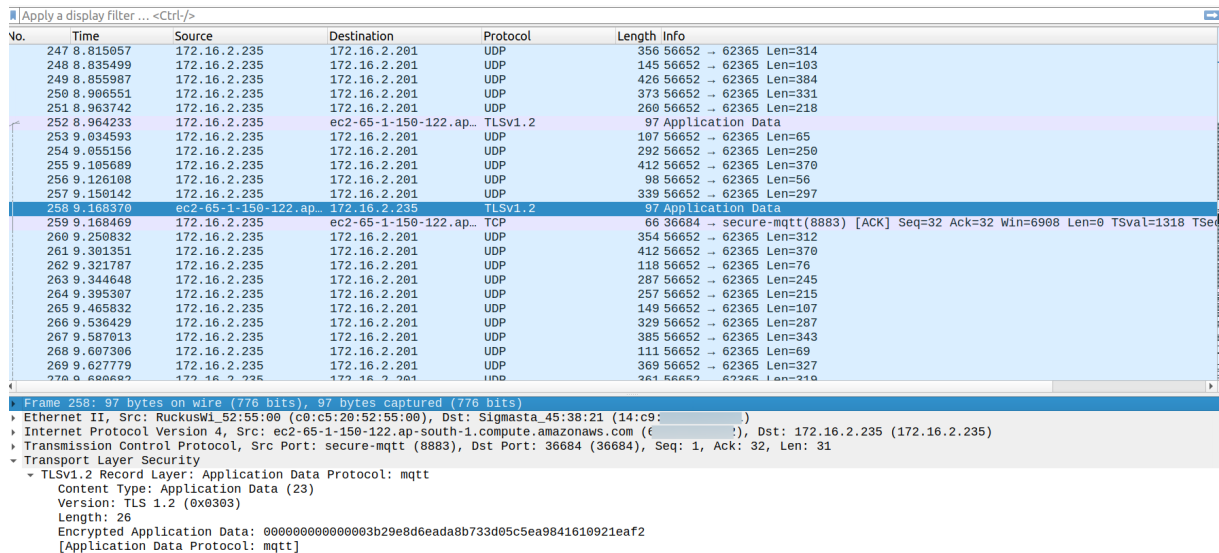
4.3.4 Capturing & Understanding Network Traffic

In order to gain a deeper understanding of how a device communicates with the server or other devices on the network, various methods can be employed. This includes establishing a video call or utilizing the MQTT⁴ protocol. To analyze the communication process, the network traffic can be intercepted either wirelessly or directly on the device itself. This interception technique is commonly known as a Man-in-the-Middle Attack.

Wireless traffic interception can be achieved by utilizing tools such as Ettercap. Ettercap leverages attacks like ARP spoofing to redirect the traffic flow through the attacker's machine instead of the router. By positioning the attacker as an intermediary, this method allows for monitoring and analysis of the data exchange.

Another approach to capturing network traffic involves using a program called tcpdump. Tcpdump is a command-line packet analyzer that enables the real-time capture and analysis of network traffic. As we shell access, we can download a version of tcpdump compiled for ARM architecture and execute it on the target device, which will save the data as a .cap file.

To analyze the captured data, we can utilize Wireshark, a widely used network protocol analyzer. Wireshark allows for detailed inspection and interpretation of captured network traffic. By opening the .cap file in Wireshark, we can examine the packet-level details, dissect protocols, and gain valuable insights into the communication process between the device and the server or other network devices as shown below.



No.	Time	Source	Destination	Protocol	Length	Info
247	8.815057	172.16.2.235	172.16.2.201	UDP	356	56652 → 62365 Len=314
248	8.835499	172.16.2.235	172.16.2.201	UDP	145	56652 → 62365 Len=103
249	8.855987	172.16.2.235	172.16.2.201	UDP	426	56652 → 62365 Len=384
250	8.906551	172.16.2.235	172.16.2.201	UDP	373	56652 → 62365 Len=331
251	8.963742	172.16.2.235	172.16.2.201	UDP	260	56652 → 62365 Len=218
252	8.964233	172.16.2.235	ec2-65-1-150-122.ap...	TLSv1.2	97	Application Data
253	9.034593	172.16.2.235	172.16.2.201	UDP	107	56652 → 62365 Len=65
254	9.055156	172.16.2.235	172.16.2.201	UDP	292	56652 → 62365 Len=250
255	9.105689	172.16.2.235	172.16.2.201	UDP	412	56652 → 62365 Len=370
256	9.126188	172.16.2.235	172.16.2.201	UDP	98	56652 → 62365 Len=56
257	9.150142	172.16.2.235	172.16.2.201	UDP	339	56652 → 62365 Len=297
258	9.168370	ec2-65-1-150-122.ap...	172.16.2.235	TLSv1.2	37	Application Data
259	9.168469	172.16.2.235	ec2-65-1-150-122.ap...	TCP	66	36684 → secure-mqtt(8883) [ACK] Seq=32 Ack=32 Win=6908 Len=0 TSval=1318 Tseq=...
260	9.250832	172.16.2.235	172.16.2.201	UDP	354	56652 → 62365 Len=312
261	9.301351	172.16.2.235	172.16.2.201	UDP	412	56652 → 62365 Len=370
262	9.321787	172.16.2.235	172.16.2.201	UDP	118	56652 → 62365 Len=76
263	9.344648	172.16.2.235	172.16.2.201	UDP	287	56652 → 62365 Len=245
264	9.395307	172.16.2.235	172.16.2.201	UDP	257	56652 → 62365 Len=215
265	9.465832	172.16.2.235	172.16.2.201	UDP	149	56652 → 62365 Len=107
266	9.536429	172.16.2.235	172.16.2.201	UDP	329	56652 → 62365 Len=287
267	9.587013	172.16.2.235	172.16.2.201	UDP	385	56652 → 62365 Len=343
268	9.607306	172.16.2.235	172.16.2.201	UDP	111	56652 → 62365 Len=69
269	9.627779	172.16.2.235	172.16.2.201	UDP	369	56652 → 62365 Len=327
270	9.680602	172.16.2.235	172.16.2.201	UDP	261	56652 → 62365 Len=219

Frame 258: 37 bytes on wire (776 bits), 37 bytes captured (776 bits)	
Ethernet II	Src: RuckusW1.52:55:00 (c8:c5:20:52:55:00), Dst: Sigmasta_45:38:21 (14:c9:...
Internet Protocol Version 4	Src: ec2-65-1-150-122.ap-south-1.compute.amazonaws.com (66...), Dst: 172.16.2.235 (172.16.2.235)
Transmission Control Protocol	Src Port: secure-mqtt (8883), Dst Port: 36684 (36684), Seq: 1, Ack: 32, Len: 31
Transport Layer Security	
TLSv1.2 Record Layer: Application Data Protocol: mqtt	
Content Type: Application Data (23)	
Version: TLS 1.2 (0x0303)	
Length: 26	
Encrypted Application Data: 000000000000003b29e8d6eada8b733d05c5ea9841610921eaf2	
[Application Data Protocol: mqtt]	

Figure 3: TCPDump traffic capture

⁴Message Queuing Telemetry Transport is an extremely simple and lightweight messaging protocol (subscribe and publish) designed for limited devices and networks with high latency, low bandwidth, or unreliable networks.

4.3.5 linuxrc

Linuxrc is a fundamental component of the Linux kernel, responsible for the initial configuration and setup during the boot process of the system. It is a user-space program executed by the kernel after loading the initial ramdisk (initrd) or the initial ramfs (initramfs). Linuxrc serves as a bridge between the kernel and the main user-space applications, facilitating the integration of essential drivers and modules required for the successful operation of the system.

Executing the linuxrc file resulted in sensitive information disclosure, where the log details the process of an IoT device initiating a peer-to-peer (P2P) streaming session and subscribing to an MQTT topic, amongst other things. The following information may be found in the logs, which can be seen in the figure below.

- **Device UUID:** Unique identifier for the device, which can be used to track or target the device.
- **MAC address:** Physical address of the device's network interface, which can be used to track or target the device.
- **Session ID:** Unique identifier for the P2P streaming session, which can be used to intercept or manipulate the session.
- **AES encryption key and IV:** Secret encryption key and initialization vector, which can be used to decrypt the data.
- **Candidate addresses (host and reflexive):** IP addresses and ports used for establishing the P2P connection.

```
3286 2733132723 24-03 15:20:24.429 V GwControlMonitorIot : Subscribed MQTT topics. Count = 1
3286 2425482741 24-03 15:20:33.094 V GwControlMonitorIot : ::sendFirebaseStackEvent:[1693] Entry
3286 2425482741 24-03 15:20:33.108 V FirebaseEventHelper : ::sendFirebaseStackEventIfEnabled:[465] Entry
3286 2425482741 24-03 15:20:33.122 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[22] Entry deviceUUID REDACTED
3286 2425482741 24-03 15:20:33.125 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[35] macAddress 14:C9:CF:XX:XX:XX
3286 2425482741 24-03 15:20:33.142 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[54] Exit
3286 2425482741 24-03 15:20:33.158 V FirebaseEventHelper : ::sendFirebaseStackEventIfEnabled:[474] Exit
3286 2425482741 24-03 15:20:33.159 V GwControlMonitorIot : ::sendFirebaseStackEvent:[1696] Exit
3286 2425482741 24-03 15:20:50.744 V GwControlMonitorIot : ::sendFirebaseStackEvent:[1693] Entry
3286 2425482741 24-03 15:20:50.760 V FirebaseEventHelper : ::sendFirebaseStackEventIfEnabled:[465] Entry
3286 2425482741 24-03 15:20:50.774 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[22] Entry deviceUUID REDACTED
3286 2425482741 24-03 15:20:50.776 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[35] macAddress 14:C9:CF:XX:XX:XX
3286 2425482741 24-03 15:20:50.853 V FirebaseEventHelper : ::isFirebaseEnabledForCurrentDevice:[54] Exit
3286 2425482741 24-03 15:20:50.855 V FirebaseEventHelper : ::sendFirebaseStackEventIfEnabled:[474] Exit
3286 2425482741 24-03 15:20:50.858 V GwControlMonitorIot : ::sendFirebaseStackEvent:[1696] Exit
3286 3280590915 24-03 15:20:52.966 V GwControlMonitorIot : UNKNOWN UNKNOWN ::mqttNotificationFromCloud:[1092] received mqtt notification for topic /control/REDACTED/REDACTED/p
3286 3280590915 24-03 15:20:52.967 V GwControlMonitorIot : UNKNOWN UNKNOWN ::mqttNotificationFromCloud:[1094] received mqtt data for topic {"command": : {"device
3286 3280590915 24-03 15:20:52.971 V GwControlMonitorIot : ::mqttNotificationFromCloud:[1140] isTest command 0
3286 3280590915 24-03 15:20:52.977 V GwControlMonitorIot : [mqttNotificationFromCloud][1160]DeviceUUID = REDACTED
3286 3280590915 24-03 15:20:52.980 V GwControlMonitorIot : UNKNOWN UNKNOWN ::mqttNotificationFromCloud:[1172] received mqtt data for Topic Name[/control/REDACTED/REDACTED/p2pS
3286 3280590915 24-03 15:20:52.994 V GwControlMonitorIot : [mqttNotificationFromCloud][1189]&lt;mqtt&gt; Case2_0 P2p Stream detected
3286 3280590915 24-03 15:20:53.009 V GwControlMonitorIot : [mqttNotificationFromCloud][1210]&lt;mqtt&gt; Case2_1 P2p Stream detected
3286 3280590915 24-03 15:20:53.012 V p2pManager : P2PManager_PostMqttMessage:[1188] Enter
3286 3732362744 24-03 15:20:53.013 V p2pManager : P2PManager_MainThread:[253] Enter Action[ACTION_MQTT_MESSAGE]
3286 3732362744 24-03 15:20:53.013 V p2pManager : P2PManager_MainThread:[266] p2p Active session [0][0]
3286 3732362744 24-03 15:20:53.020 V p2pManager : P2PManager_ProcessCommands:[1424] Enter Active[0][0]
3286 3732362744 24-03 15:20:53.020 V p2pManager : <time_p2p&gt; Start P2p call received; Line[1457] Session[e2dfa6e6-44f1-4e88-abe4-fbb008e5c845]
3286 3732362744 24-03 15:20:53.023 V p2pManager : [P2PStreaming] Command [startP2pCall]
3286 3732362744 24-03 15:20:53.023 V p2pManager : [P2PStreaming] Session [e2dfa6e6-44f1-4e88-abe4-fbb008e5c845]
3286 3732362744 24-03 15:20:53.026 V p2pManager : [P2PStreaming] pttEnabled [false]
3286 3732362744 24-03 15:20:53.026 V p2pManager : [P2PStreaming] protocol [udp]
3286 3732362744 24-03 15:20:53.044 V p2pManager : [P2PStreaming] aesKey [409F4204F82DFEDE84AD8D522E68D052]
3286 3732362744 24-03 15:20:53.063 V p2pManager : [P2PStreaming] aesIV [AE656FFDBEBE0393E2407536A781995]
3286 3732362744 24-03 15:20:53.065 V p2pManager : [P2PStreaming] streamSourceQuality [low]
3286 3732362744 24-03 15:20:53.068 V p2pManager : [P2PStreaming] mediaStreams [both]
3286 3732362744 24-03 15:20:53.069 V p2pManager : [P2PStreaming] offerData [
{
  "host": "192.168.0.14:65404";,
  "protocol": "UDP";,
  "srflx_0": "REDACTED:65404";
}
]
```

Figure 4: linuxrc output

4.3.6 EMBA - Embedded device firmware security analyzer

Using EMBA, all the firmware and files that were on the Qubo were analyzed. EMBA is a central firmware analysis tool designed for penetration testers, supporting the complete security analysis process, including firmware extraction, static analysis, dynamic analysis via emulation, and web report generation. EMBA automatically identifies possible weak spots and vulnerabilities in firmware, such as insecure binaries, outdated software components, potentially vulnerable scripts, or hard-coded passwords. The analysis process was carried out through EMBA's command line interface, and a user-friendly web report was generated for further analysis.

The report is available [here](#).

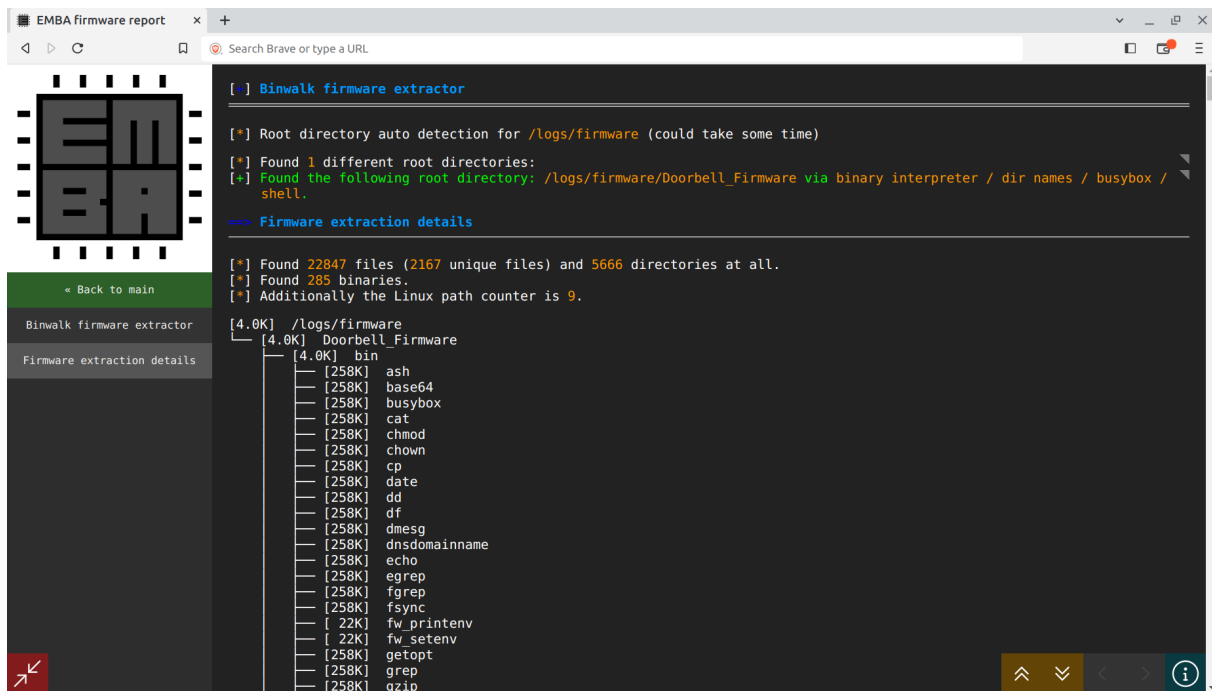


Figure 5: EMBA Web Report

4.3.7 More Information Disclosure

As data was exfiltrated from the device, several files were discovered that led to sensitive information disclosure, such as the local SQL database on the device, revealing sensitive information such as user email addresses, streaming credentials, Wi-Fi passwords, and API keys for services like Alexa, as shown in Figures 8 & 9. Along with which, a file named "Input.txt" in the same directory contained sensitive information as shown below.

```
"alexaURL" : "https://alexa.platform.quboworld.com",
"cameraName" : "Video Doorbell",
"entityUUID" : "REDACTED",
"esi_url" : "https://srvc.platform.quboworld.com",
"httpURL" : "srvc.platform.quboworld.com",
"mqttURL" : "mqtt.platform.quboworld.com:8883",
"ntp_url" : "",
"provisionToken" : "REDACTED",
"spId" : "REDACTED",
"stun1URL" : "stun1.platform.quboworld.com:5349",
"stun2URL" : "stun2.platform.quboworld.com:5349",
"stun3URL" : "stun3.platform.quboworld.com:5349",
"stun4URL" : "stun4.platform.quboworld.com:5349",
"stun5URL" : "stun5.platform.quboworld.com:5349",
"unitUUID" : "REDACTED",
"weather_info_url" : "https://srvc.platform.quboworld.com/weather/"
```

4.3.8 Changing the doorbell chime

As another Proof of Concept that code can be executed, the doorbell chime audio was changed as shown here.

Audio file requirements

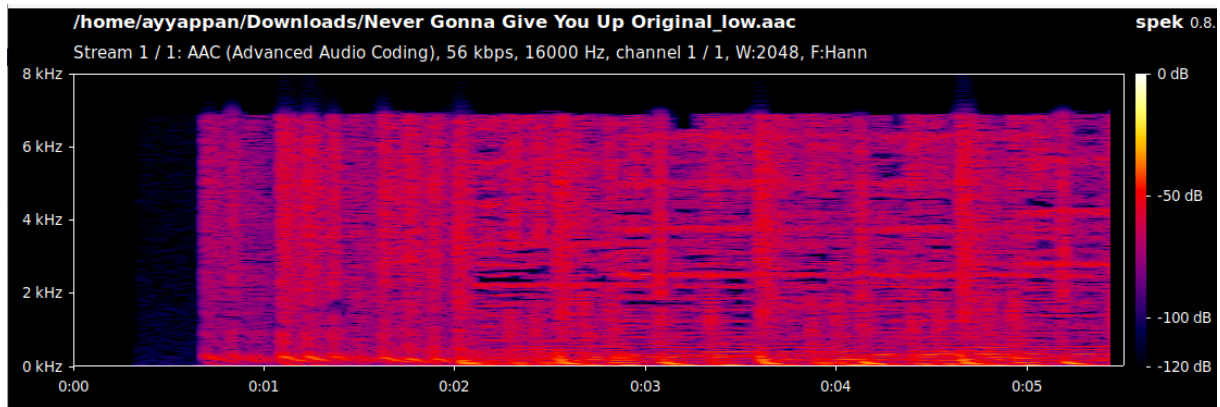


Figure 6: Information Disclosure : Audio File Parameters

- Bitrate : below 70 kbps
- Sample Rate : 16 kHz
- Audio Channels : Mono
- Format : AAC

Instructions to change the chime

- Rename the file to dingdong.aac
- Navigate to /customer/audio/
- Replace dingdong.aac with your file
- Enjoy!

4.3.9 Root cause of the vulnerability

```
GNU nano 6.2                ./etc/init.d/rcS
#!/bin/sh

mount -a
mkdir -p /dev/shm
mkdir -p /dev/pts
mount devpts

echo /sbin/mdev > /proc/sys/kernel/hotplug

/sbin/mdev -s

/sbin/sysctl -p

telnetd -l sh
mkdir -p /dev/pts
mount -t sysfs none /sys
mount -t tmpfs mdev /dev
mkdir -p /var/lock
mdev -s
mount -t ubifs ubi0:miservice /config; mount -t ubifs ubi0:customer /customer;
mkdir -p /dev/pts
mount -t devpts devpts /dev/pts

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```

Figure 7: Telnet service initialization

As seen in Figure 2, this shell script is executed on startup, and runs the command "telnetd -l sh" which starts a telnet server with no password being required.

5 Analysis and Impact

The impact of this vulnerability on users and the device itself includes:

- Unauthorized access : RCE allows attackers to bypass security measures, access sensitive information, and potentially take control of the affected IoT device. This can lead to privacy breaches and loss of data integrity.
 - Mirai botnet: In 2016, the Mirai botnet targeted IoT devices such as routers, cameras, and DVRs, allowing attackers to gain unauthorized access and launch massive DDoS attacks.
- Espionage: RCE vulnerabilities and leaking of the Wowza RTSP credentials can enable attackers to eavesdrop on sensitive information, monitor user activities, or live stream while recording audio and video from the device.
 - Weeping Angel : Targeted Samsung Smart TVs, turning them into covert listening devices.
- Botnets: Attackers can use the compromised IoT devices to create botnets, which are networks of infected devices used for coordinated cyberattacks, such as DDoS attacks, spam campaigns, and cryptocurrency mining.
 - In 2016, the Mirai botnet targeted IoT devices such as routers, cameras, and DVRs, allowing attackers to gain unauthorized access and launch massive DDoS attacks.
- Device malfunction: Attackers can exploit RCE vulnerabilities to disrupt the normal functioning of IoT devices, causing them to malfunction or become unresponsive.
 - St. Jude Medical pacemaker vulnerability: In 2017, security researchers discovered vulnerabilities in St. Jude Medical's pacemakers, which could have allowed attackers to cause the devices to malfunction, potentially endangering patients' lives.
- Network compromise: An IoT device with RCE vulnerability can serve as an entry point for attackers to compromise an entire network. Once they gain access, they can move laterally within the network, potentially compromising other devices and systems.
 - Target data breach: In 2013, attackers gained access to Target's network through an HVAC system, compromising the payment data of millions of customers and leading to significant financial and reputational damage for the company.
- Data tampering: Attackers can manipulate or delete data stored on IoT devices, leading to incorrect or falsified information being transmitted between devices and systems.
 - Maroochy Water Services attack: In 2000, a disgruntled former employee used a radio transmitter to remotely tamper with the data and controls of an Australian sewage treatment plant, causing significant environmental damage.

6 Mitigation Strategy

- Disabling telnet on devices in the production environment.
- Verify integrity of files on the device through checksums to ensure it has not been infected already.

To address the vulnerability and protect users, the vulnerability was reported to CERT-In, the agency in charge of who coordinated with the Qubo security team, who were rapid in their response, to address the vulnerability and protect users. The Qubo security team worked diligently to develop a patch to fix the vulnerability and were able to release it within 72 hours. Users were advised to update their Qubo devices to the latest firmware version that includes the patch as soon as possible. Qubo's quick and efficient response helped to prevent any potential harm to users and demonstrates their commitment to ensuring the security of their IoT devices.

The Qubo team has released an OTA update (firmware version HCD01.02.01.65.SYSTEM) that fixes the issue, and upgrading to the latest firmware requires **no action from the user**.

7 Conclusion

This report highlights the risks and challenges associated with the widespread adoption of IoT technologies and emphasizes the critical importance of robust security measures in the development and deployment of these devices. By addressing this vulnerability and offering effective mitigation strategies, the Qubo security team has set an excellent example for the industry, demonstrating their commitment to ensuring the safety and reliability of their IoT devices. As IoT technologies continue to evolve and become more pervasive in our daily lives, it is crucial that manufacturers prioritize security measures to ensure the privacy and safety of their users.

7.1 Important Note

All the material shown is available [here](#).

Acknowledgments

This research and disclosure process would not have been possible without:

- EMBA
- Qubo Security team and CTO
- CERT-In

I wish to express my profound gratitude to all those who have supported me, specifically my esteemed professors and mentors.

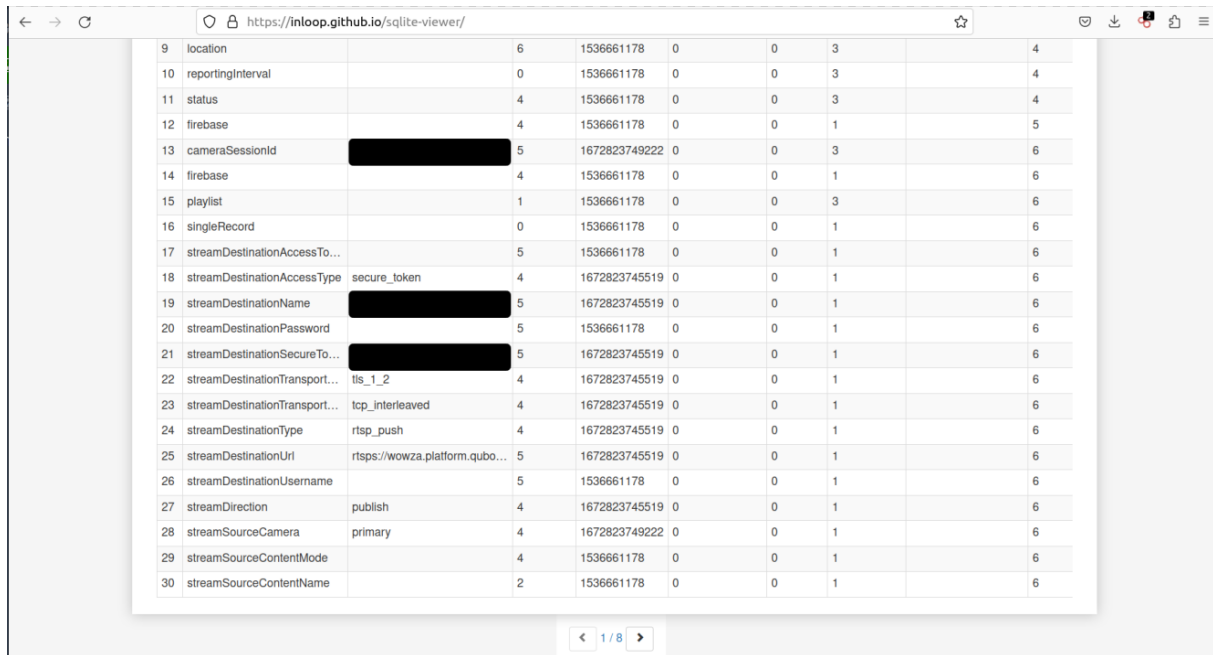
- Prof. Lance Fiondella
- Prof. Dayalan Kasalingam
- Prof. Hong Liu
- Sam Curry
- Prof. Aanjhan Ranganathan
- Dr. Harshad Sathaye
- Maryam Motallebighomi
- Dr. Balsing Rajput

References

- SigmaStar Docs
- tcpdump for ARM
- Compiling BusyBox for ARM
- BusyBox binaries

A Appendix

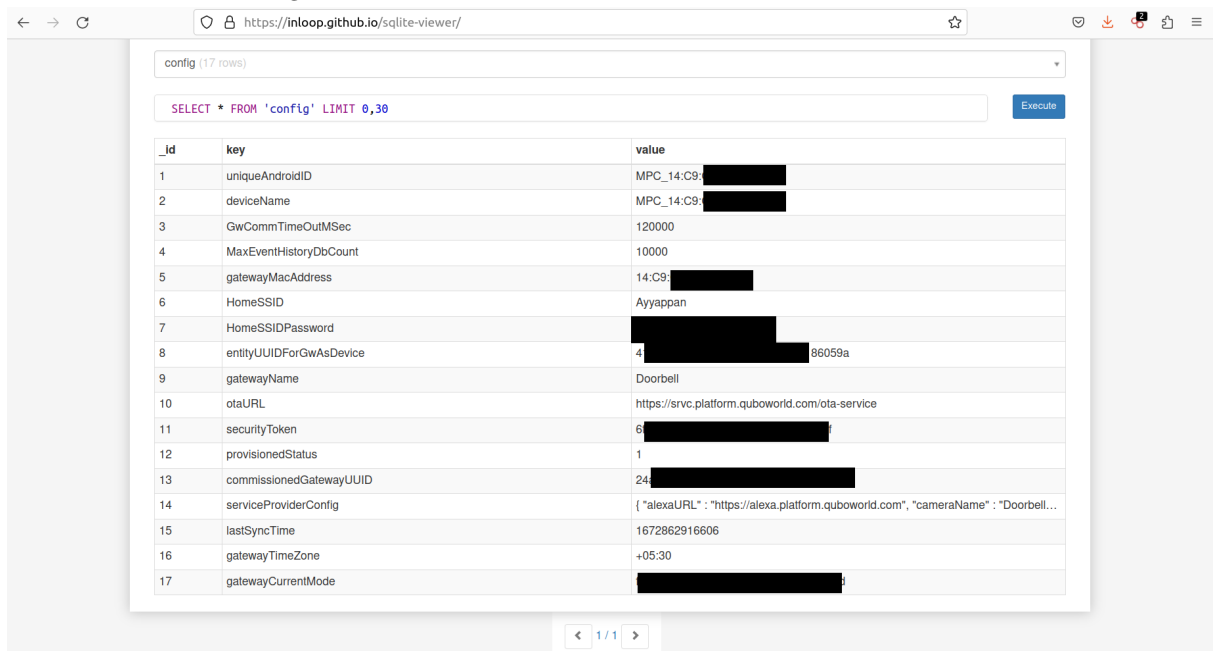
Here are some additional images that support the findings of the report:



The screenshot shows a SQLite viewer interface with a table containing 22 rows of stream destination configurations. The table has columns for id, location, reportingInterval, status, firebase, cameraSessionId, playlist, singleRecord, streamDestinationAccessTo..., streamDestinationAccessType, streamDestinationName, streamDestinationPassword, streamDestinationSecureTo..., streamDestinationTransport..., streamDestinationTransport..., streamDestinationType, streamDestinationUrl, streamDestinationUsername, streamDirection, streamSourceCamera, streamSourceContentMode, and streamSourceContentName. Some values are redacted with black boxes.

id	location	reportingInterval	status	firebase	cameraSessionId	playlist	singleRecord	streamDestinationAccessTo...	streamDestinationAccessType	streamDestinationName	streamDestinationPassword	streamDestinationSecureTo...	streamDestinationTransport...	streamDestinationTransport...	streamDestinationType	streamDestinationUrl	streamDestinationUsername	streamDirection	streamSourceCamera	streamSourceContentMode	streamSourceContentName
9		6		1536661178	0	0	3														
10		0		1536661178	0	0	3														
11		4		1536661178	0	0	3														
12		4		1536661178	0	0	1														
13		5		1672823749222	0	0	3														
14		4		1536661178	0	0	1														
15		1		1536661178	0	0	3														
16		0		1536661178	0	0	1														
17		5		1536661178	0	0	1														
18		4		1672823745519	0	0	1														
19		5		1672823745519	0	0	1														
20		5		1536661178	0	0	1														
21		5		1672823745519	0	0	1														
22		4		1672823745519	0	0	1														
23		4		1672823745519	0	0	1														
24		4		1672823745519	0	0	1														
25		5		1672823745519	0	0	1														
26		5		1536661178	0	0	1														
27		4		1672823745519	0	0	1														
28		4		1672823749222	0	0	1														
29		4		1536661178	0	0	1														
30		2		1536661178	0	0	1														

Figure 8: Information Disclosure : RTSP stream credentials



The screenshot shows a SQLite viewer interface with a table containing 17 rows of device configuration. The table has columns for _id, key, and value. Some values are redacted with black boxes.

_id	key	value
1	uniqueAndroidID	MPC_14:C9: [redacted]
2	deviceName	MPC_14:C9: [redacted]
3	GwCommTimeOutMSec	120000
4	MaxEventHistoryDbCount	10000
5	gatewayMacAddress	14:C9: [redacted]
6	HomeSSID	Ayyappan
7	HomeSSIDPassword	[redacted]
8	entityUUIDForGwAsDevice	4 [redacted] 86059a
9	gatewayName	Doorbell
10	otaURL	https://srcv.platform.quboworld.com/ota-service
11	securityToken	6 [redacted]
12	provisionedStatus	1
13	commissionedGatewayUUID	24 [redacted]
14	serviceProviderConfig	{ "alexaURL" : "https://alexa.platform.quboworld.com", "cameraName" : "Doorbell..."
15	lastSyncTime	1672862916606
16	gatewayTimeZone	+05:30
17	gatewayCurrentMode	[redacted]

Figure 9: Information Disclosure : Device configuration