I'M A NEWBIE YET I CAN HACK ZIGBEE
Take Unauthorized Control Over ZigBee Devices

LI Jun, YANG Qing
Unicorn Team – Radio and Hardware Security Research
Qihoo 360 Technology Co. Ltd.
Who we are? Unicorn Team

- Qihoo360’s Unicorn Team consists of a group of brilliant security researchers. We focus on the security of anything that uses radio technologies, from small things like RFID, NFC and WSN to big things like GPS, UAV, Smart Cars, Telecom and SATCOM.

- Our primary mission is to guarantee that Qihoo360 is not vulnerable to any wireless attack. In other words, Qihoo360 protects its users and we protect Qihoo360.

- During our research, we create and produce various devices and systems, for both attack and defense purposes.
LI Jun

Twitter: bravo_fighter
Weibo: GoRushing

- Hardware security intern in Unicorn Team of Qihoo360, China.

- Second year graduate student at Chengdu University of Information Technology, China. He received his bachelor’s degree from University of Electronic Science and Technology of China.

- Interested in the security of the internet of things and the security of automobile electronics.
• YANG Qing is the team leader of Unicorn Team.

• He has rich experiences in wireless and hardware security area, including WiFi penetration testing, cellular network interception, IC card cracking etc. His interests also cover embedded system hacking, firmware reversing, automotive security, and software radio.

• He is the first one who reported the vulnerabilities of WiFi system and RF IC card system used in Beijing subway.
Why is this talk relevant to you?

- Cause hackers might be able to control your ZigBee enabled appliances without authorization, this talk will teach you how to prevent it.
What will you learn from the talk?

You will learn, step by step, how to hack ZigBee enabled devices, and you will also learn some techniques to protect your ZigBee appliance from being hacked.
So what is ZigBee?

“ZigBee is the only open, global wireless standard to provide the foundation for the Internet of Things by enabling simple and smart objects to work together, improving comfort and efficiency in everyday life.”
So what is ZigBee?

“ZigBee is the wireless language that everyday devices use to connect to one another. In fact, ZigBee could be at work in your home right now.”
So what is ZigBee?

- Technological Standard Created for Control and Sensor Networks
- Based on the IEEE 802.15.4 Standard
- Low-power, low data rate wireless protocol
- Widely used in the Internet of Things
- Widely adapted in applications that require low power consumption
- Flexible network topology
So what is ZigBee?

ZigBee network topology
And then what is Zstack?

An specific implementation of ZigBee Stack from Texas Instruments based on its CC2530 (which is IEEE80.15.4 enabled)chip, in other words, ZigBee standard is written in plaintext while Zstack is written in code.
Security in ZigBee

ZigBee security is based on symmetric keys and both originator and recipient of a protected transaction need to share the same key.

Key distribution schemes
- Pre-installation
- Transport
- Establishment

Three key types
- Master key
- Link key
- Network key
Hacking ZigBee device step by step

Following is a schematic diagram of a smart bulb system:
Hacking ZigBee device step by step

3 Normal control flows:

Phone → Server → Router → IOT Gateway → Bulb
Hacking ZigBee device step by step

What we want is to directly control the bulb via our own ZigBee node:
Find the encryption key from firmware

The keys are stored in every node in the network, as the bulb is harder to disassemble so we chose to extract the keys from the gateway.

Zigbee Bulb

Gateway
Find the encryption key from firmware

As the red arrow indicates, the debug interface is right there, we solder on a few wires, connect it to a debugger, and used TI’s SmartRF Flash Programmer to dump the firmware.

Gateway being praised open

Debugger used to extract firmware
Find the encryption key from firmware

Screenshot of TI’s SmartRF Flash Programmer
Find the encryption key from firmware

Now we got THE firmware, what is next?
Let’s do “Firmware Diving”
Searching through firmware for keys
Find the encryption key from firmware

First set the keys to have distinct signature, then find it in the firmware and see if we could discover something interesting.

As the key is used to encrypt the packets, why not try to find the instructions that manipulate the keys?

**Bingo!** We found that the instructions used to manipulate the keys have relatively fixed pattern (shown in the next slide) and the four consecutive move instructions could be used as a *filter* (or signature) for the address of the keys.
Find the encryption key from firmware.
Find the encryption key from firmware

On the upper right corner is the instructions that manipulate the network key. The $0x31$, $0xAD$ is the memory address that stores the key (shown on the lower left corner)
Find the encryption key from firmware

On the upper right corner is the instructions that manipulate the network key. The 0x31, 0xAD is the memory address that stores the key (shown on the lower left corner).
Find the encryption key from firmware

Then we use the four consecutive move instructions’ corresponding machine code and operand (75 08 ? 75 09 ? 75 0A) as a filter to search through the firmware for the address of the keys.
Find the encryption key from firmware

Then we use the four consecutive move instructions’ corresponding machine code and operand (75 08 ? 75 09 ? 75 0A) as a filter to search through the firmware for the address of the keys.

A screenshot shows the possible addresses of the keys.
Verify The Keys

In order to verify the keys, we utilized the MIC (message integrity check) contained in the packet, and if the deciphered packet can pass the MIC, we can assert that we find the right keys.
Verify The Keys

In order to verify the keys, we utilized the MIC (message integrity check) contained in the packet, and if the deciphered packet can pass the MIC, we can assert that we find the right keys.
Find the encryption key by sniffing

The following screenshot shows the process of a new node joining the network, and the figure is quite self-explanatory. The network key is sent from the coordinator to the joining device in plaintext, and after receiving the network key the communication is immediately encrypted.
Find the encryption key by sniffing.
“Utilize” the keys found

I wouldn’t say that after we found the key we could do some data mining to find the users habit etc, cause that would be a little bit farfetched, but the following are some very practical attacks we can perform:

- Analysis of the deciphered data
- Replay & Spoof
- Intercept
- Disassociation attack
### Analysis of the deciphered data

After we deciphered the data, in order to take control over the target device, we have to analyze the application level data and the results are as following:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte0</td>
<td>0x04</td>
</tr>
<tr>
<td>Byte1</td>
<td>Target PANID</td>
</tr>
<tr>
<td>Byte2</td>
<td></td>
</tr>
<tr>
<td>Byte3</td>
<td>Unknown</td>
</tr>
<tr>
<td>Byte4</td>
<td>Mode</td>
</tr>
<tr>
<td>Byte5</td>
<td>Red</td>
</tr>
<tr>
<td>Byte6</td>
<td>Green</td>
</tr>
<tr>
<td>Byte7</td>
<td>Blue</td>
</tr>
<tr>
<td>Byte8</td>
<td>Illuminance</td>
</tr>
<tr>
<td>Byte9</td>
<td>Checksum</td>
</tr>
</tbody>
</table>
Analysis of the deciphered data

The payload is 10 byte in length, with the last byte being the xor checksum of the foregoing bytes, the byte1 and byte2 is the PANID of the target device (the bulb in our case). Now we can control the bulb with our own node.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte0</td>
<td>0x04</td>
</tr>
<tr>
<td>Byte1</td>
<td>Target PANID</td>
</tr>
<tr>
<td>Byte2</td>
<td>Unknown</td>
</tr>
<tr>
<td>Byte3</td>
<td>Mode</td>
</tr>
<tr>
<td>Byte4</td>
<td>Red</td>
</tr>
<tr>
<td>Byte5</td>
<td>Green</td>
</tr>
<tr>
<td>Byte6</td>
<td>Blue</td>
</tr>
<tr>
<td>Byte7</td>
<td>Illuminance</td>
</tr>
<tr>
<td>Byte8</td>
<td>Checksum</td>
</tr>
</tbody>
</table>
Take control

A Zigbee node we made and we used it to control the bulb
Replay& Spoof
Disassociation attack

Disassociation Sequence
Hacking for Protection

Only talking about attacks and leaving protection aside goes against the ethos of Defcon and the whole hacker community I guess, so here come tips for prevention or, at least, mitigation:

- Store hash of the encryption key instead of plaintext.
- Don’t use OTA (over the air) key provisioning scheme, use preinstall or key negotiation instead.
- Blow the fuse to prevent the firmware from being dumped.
- Employ some light weight encryption on the application data to make the analysis of application data harder after key compromise.
References

- Below are references related to the topics discussed.
- 1) The ZigBee Alliance homepage:
  - [http://www.ZigBee.org](http://www.ZigBee.org)
- 2) The KillerBee framework:
  - [https://github.com/riverloopsec/killerbee](https://github.com/riverloopsec/killerbee)
- 3) Paperback book entitled “ZigBee Wireless Networks and Transceivers”
  - April 2007. Author: Ken Masica. Lawrence Livermore National Laboratory
  - Author: Robert Cragie. Chair, ZigBee Alliance ZARC Security Task Group. Principal Engineer, Jennic Ltd.
- 6) A webpage:
Acknowledgment

ZHANG Kai

- Twitter: peekair_zhang

Former reversing engineer at Qihoo360. (protocol, fireware, binary format & crack).
He loves cats.

Nikita

- Twitter: Niki7a
Thank you!