



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.



360UNICORNTTEAM

Who we are? Unicorn Team



- Qihoo360's UnicornTeam 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



360UNICORNTTEAM

YANG Qing

Weibo: [Ir0nSmith](#)

- 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.



360UNICORNTTEAM

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”

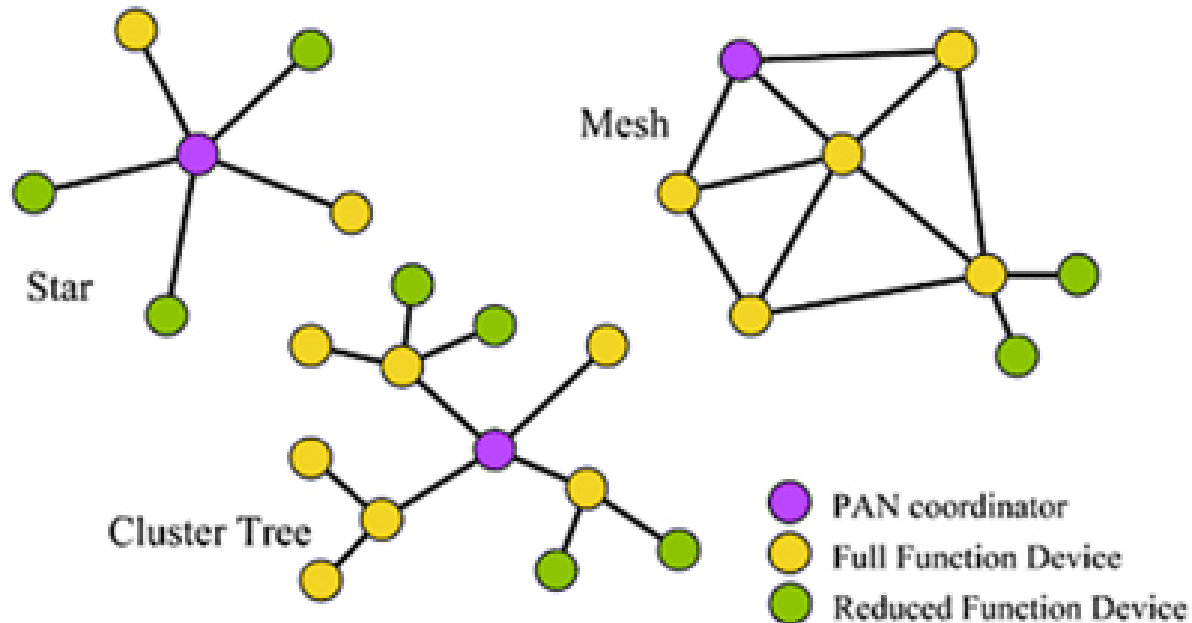


So what is ZigBee ?

- 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.

The screenshot shows the Texas Instruments website interface. At the top, there is a navigation bar with the TI logo and 'TEXAS INSTRUMENTS' text. A search bar is present with the placeholder 'Everything' and a search icon. On the right, there are links for 'Login / Register'. Below the navigation bar, there is a red banner with various menu items: 'Products', 'Applications & designs', 'Tools & software', 'Support & community', 'Sample & buy', 'About TI', 'History', 'Cart', 'English', and 'myTI'. Below this banner, there are three sections: 'My products' with a link to 'A fully compliant ZigBee 2012 solution: Z-Stack', 'My technical documents' with 'No documents in your history', and 'My searches' with 'No Searches in your history'. The main content area shows the breadcrumb 'TI Home > Semiconductors > Wireless Connectivity > A fully compliant ZigBee 2012 solution: Z-Stack' and the title 'A fully compliant ZigBee 2012 solution: Z-Stack (ACTIVE) Z-STACK'. Below the title, there are three tabs: 'Description & Features', 'Technical Documents', and 'Support & Community'. The 'Order Now' section contains a table with the following data:

Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date	Description
Z-STACK-LINUX-GATEWAY: Z-Stack™Linux Ubuntu Gateway installer	Free Download	Alert Me	ACTIVE	1.0.1	09-Jun-2014	Please see below for the descriptions of the ZigBee 2012 solutions downloads.
	Free	Alert Me	ACTIVE	1.2.2	23-FEB-	Please see below for the descriptions of the ZigBee

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

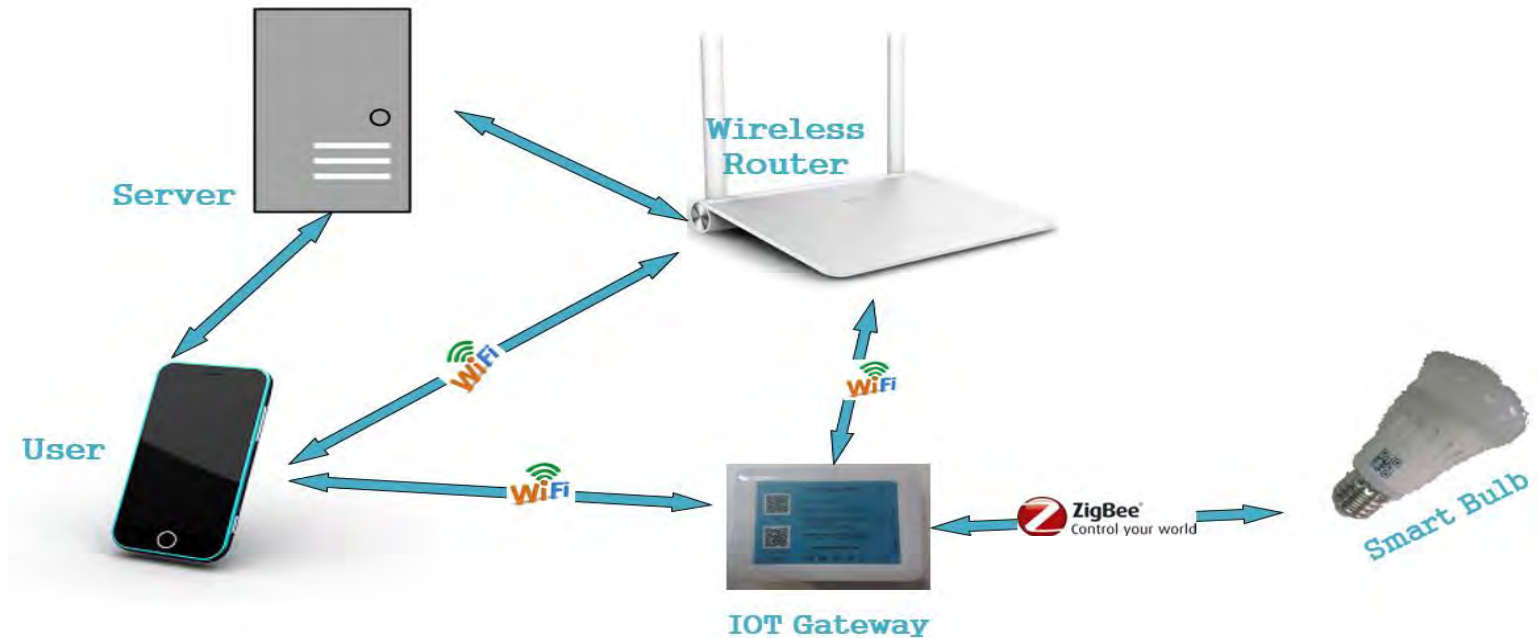
Three key types

- Master key
- Link key
- Network key



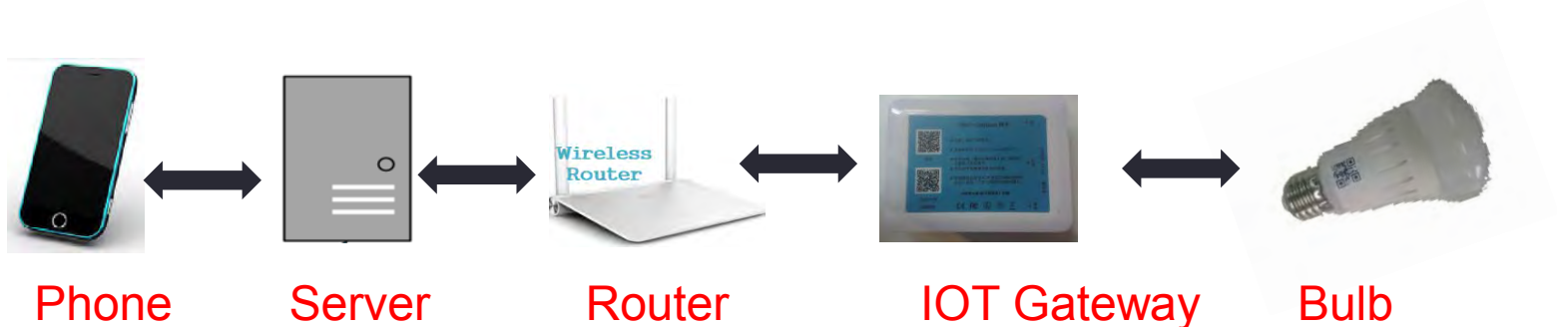
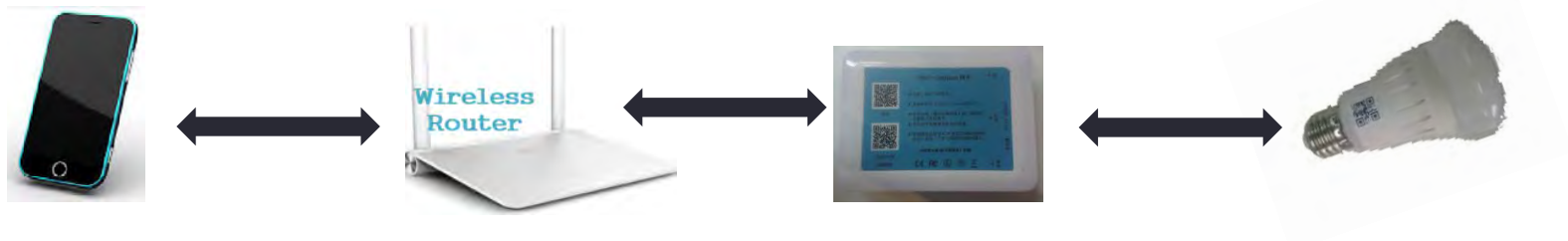
Hacking ZigBee device step by step

Following is a schematic diagram of a smart bulb system:



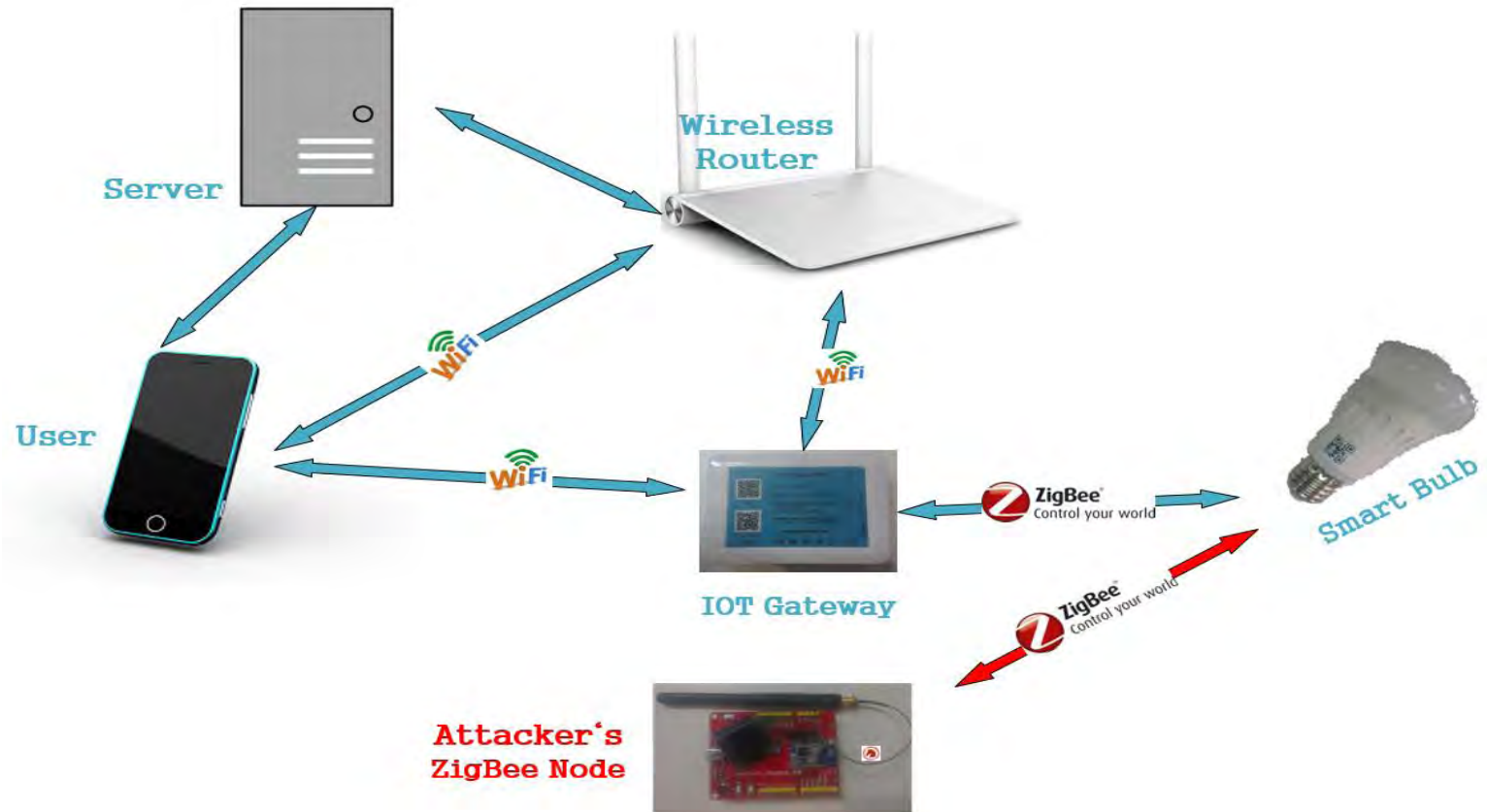
Hacking ZigBee device step by step

3 Normal control flows:



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 blub is harder to disassemble so we chose to extract the keys from the gateway.



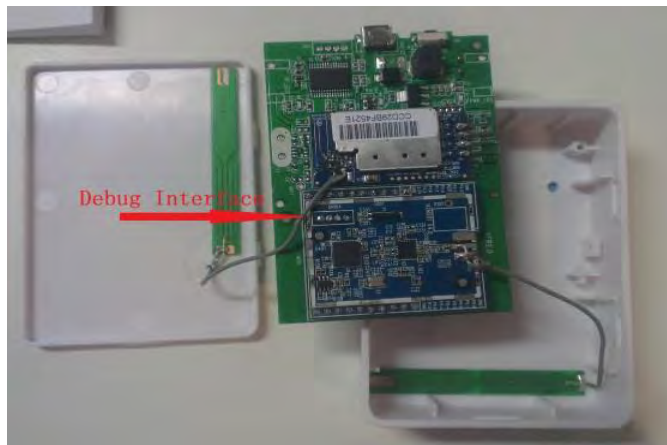
Zigbee Bulb



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.

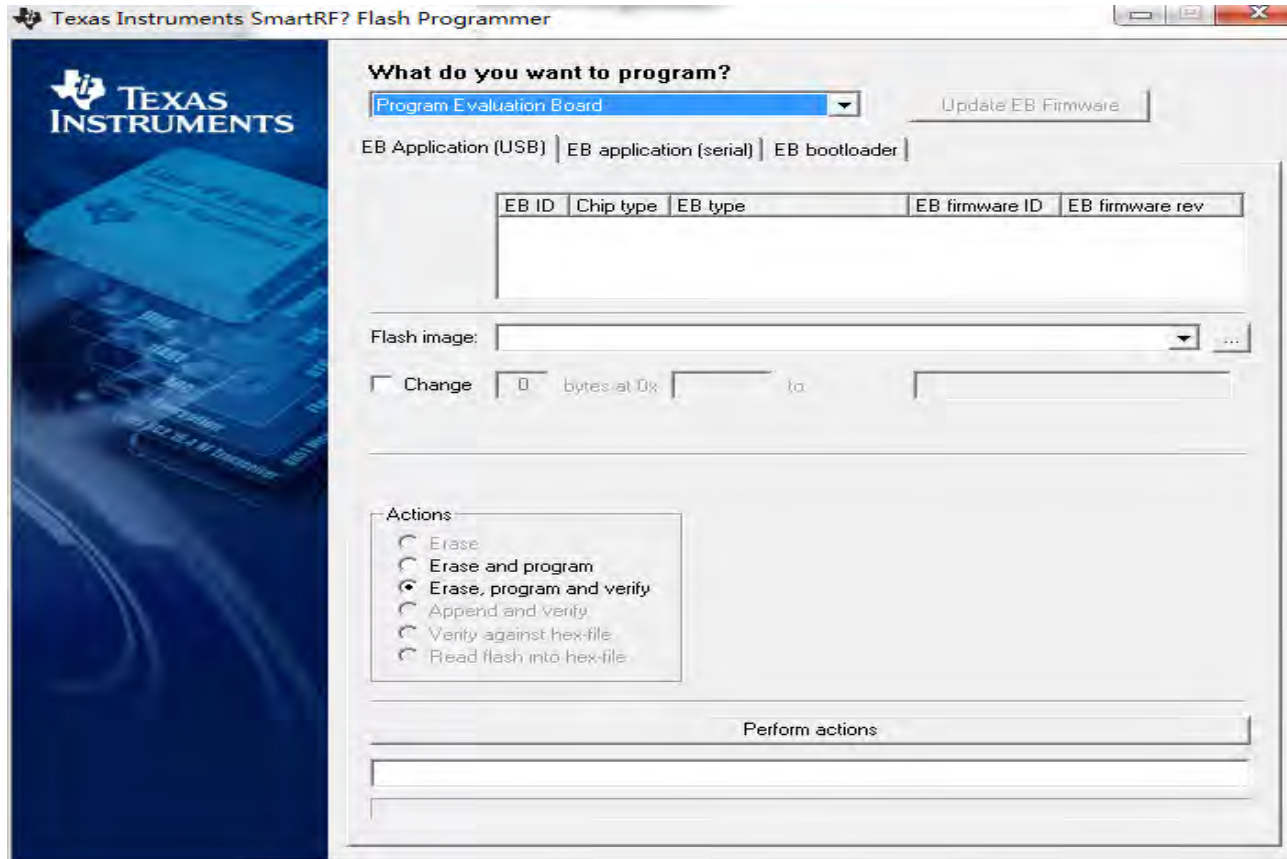


Gateway being prised open



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 ?

```
00000000 27 1b de 94 62 a9 32 b9 f2 05 b1 1c fe aa 7c 57 ' .P"bø2'ò.±.p²|W
00000020 d4 f0 3d e4 6d e8 81 31 3b 99 92 39 31 b6 f9 3d Ôð=ämè 1;™' 91qù=
00000030 e2 fd bf f6 b4 92 c1 a4 4a f2 1d 75 e4 e9 cd 63 âý¿ö''Á*Jò.uâéÍc
00000040 f4 3e 55 5e 49 7f a1 a3 2d 89 4f ca d7 63 4d 2f ô>U^I[];±-%OË×cM/
00000050 b8 59 ef 9c 02 2d 16 06 18 65 c7 61 96 ef bc ed ,Yiæ.-...eÇa-îkí
00000060 ed f9 dd 19 8b bc 1d cb c0 ae 63 34 19 9a 7c 05 iùÝ.<%.ËÄøc4.š|.
00000070 91 f3 cd d9 44 fa 93 4f b6 bf b2 05 6f 52 80 0d 'óíÜDú"Oq¿;°.oRE.
00000080 90 09 84 74 66 48 8c 92 8a 6f 0f 19 a9 9d 8f e7 .„tfHE' Šo..ø ç
00000090 5c 09 9f b5 fc c8 e4 42 b8 ad a0 4b 69 b0 87 3d \.ÿpüËäB,- Ki°+=
00000a00 64 8a 30 f4 2d b4 6f c2 51 b2 96 1a 61 22 ce 58 dŠ0ô-'oÄQ$-.a"ÎX
00000b00 07 03 b2 60 18 b9 ce 4e ba 31 58 c5 3a 41 ae 2e ..*'.îN°1XÄ:Aø.
00000c00 3d c5 3d 71 f6 44 99 05 39 0e 04 48 9d 68 2d 01 =Ä=qøD™.9..H h-.
00000d00 dd d1 16 05 51 3e 7f a6 8e 96 91 57 d6 33 ae 56 ÝÑ..Q>[];ž-'WÖ3øV
00000e00 c0 5f d9 90 03 cd 2b 53 82 7a 5b bb 5f 10 df d0 À_Û .Í+S,z[»_.BD
00000f00 16 3b 2e 2c ba b4 90 4a c5 1b 6b 27 95 5f d4 95 .;.,°' JÄ.k'°_Ô•
00001000 b0 9c 91 f0 cb 27 21 25 9a 40 4d 2f 62 4a 1e 32 °æ'ðË'!šš@M/bJ.2
00001100 bf 79 8d 80 d4 77 52 81 04 18 c3 ec 59 67 2d 97 çy €ÖwR ..ÄiYg—
00001200 28 02 f5 94 b5 aa 59 a1 38 d1 23 88 32 0e 31 86 (.ð"µ²Y;8Ñ#^2.1†
00001300 9f d1 f8 41 34 8a 58 ba 03 69 72 30 b1 49 ac 8e ÝÑøA4ŠX°.ir0±I-ž
00001400 fc 29 36 91 e7 f2 e5 35 b4 41 e9 0e 93 3e 5b 2a ü)6'çòâ5'Áé.">[*
00001500 f5 5b 84 dd 0c ad 2d c5 da b1 32 85 55 d1 2e 74 ô[,Ý.--ÄÛ±2...UÑ.t
00001600 f7 dc 0f 09 fe 85 07 b0 07 ba 1a e3 53 d5 84 c6 =Û..p...°.°.äSÖ„Æ
00001700 b6 07 16 05 0c d9 de fb 0b 57 0f ba f3 3e 36 fe ¶...ÛËû.W.°ó>6p
00001800 a5 9e e9 f0 1d 7b 3f 42 4f a1 c2 79 17 a8 78 f0 ¥žéð.{?BO;Äy.¨xð
00001900 75 62 cc 38 9e 6f 12 d3 5b f0 ed 8f 9c 2e 65 6f ubî8žo.Ó|ðí æ.eo
```



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 patern (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

The screenshot displays the IAR Embedded Workbench IDE interface for a project named 'GenericApp'. The main window shows the source code for 'ZGlobals.c', which defines various network parameters. A red box highlights the 'DDEFAULT_KEY' definition, which is a 16-byte array of hexadecimal values: `{0x08, 0x02, 0x03, 0x05, 0x0A, 0x06, 0x02, 0x0D, 0x03, 0x0B, 0x0C, 0x0F, 0x02, 0x05, 0x06, 0x0F}`. A green arrow points from this definition to the disassembly window, which shows the instruction `osal_memcpy(zgPreConfigKey, defaultKey, SEC_KEY_LEN);` at address 04DD6E. Another green arrow points from the disassembly window to the memory dump window, which shows the contents of memory starting at address 0x31ad. The memory dump shows a sequence of bytes, with a red box highlighting the values `08 0a 05 02 03 05 0a 06 02 0d 03 0b 0c 0f 02 05 06 0f`, which correspond to the 'DDEFAULT_KEY' array. A blue arrow points from the memory dump to the 'Default Network Key' label. Another red box highlights the values `42 65 65 41 66 66 69 61 6e 63 65 30 39 00 00 00` in the memory dump, with a blue arrow pointing to the 'Trust Center Link Key' label. The disassembly window also shows other instructions, including `osal_memcpy(zgPreConfigKey, defaultKey, SEC_KEY_LEN);` at address 04DD85, which is highlighted in green.

```
/* Number of times retry to poll parent before indicating loss of synchronization
 * with parent. Note that larger value will cause longer delay for the child to
 * rejoin the network.
 */
-IMAX_POLL_FAILURE_RETRIES=2

/* The number of items in the broadcast table */
-IMAX_BCAST=9

/* The maximum number of groups in the groups table */
-DAPS_MAX_GROUPS=16

/* Number of entries in the regular routing table plus additional
 * entries for route repair
 */
-IMAX_RTG_ENTRIES=40

/* Maximum number of entries in the Binding table. */
-DNWK_MAX_BINDING_ENTRIES=4

/* Maximum number of cluster IDs for each binding table entry.
 * Note that any value other than the default value may cause a
 * compilation warning but Device Binding will function correctly.
 */
-IMAX_BINDING_CLUSTER_IDS=4

/* Default security key. */
//DDEFAULT_KEY={0x02, 0x03, 0x05, 0x07, 0x09, 0x0B, 0x0D, 0x0F, 0x00, 0x02, 0x04, 0x06, 0x08, 0x0A, 0x0C, 0x0E}
DDEFAULT_KEY={0x08, 0x02, 0x03, 0x05, 0x0A, 0x06, 0x02, 0x0D, 0x03, 0x0B, 0x0C, 0x0F, 0x02, 0x05, 0x06, 0x0F}

Reset when ASSERTI occurs, otherwise Slash LEDs */
//DASSERTI_RESET

/* Set the MAC MAX Frame Size (802.15.4 default is 102) */
-DMAC_MAX_FRAME_SIZE=116
```

Disassembly

```
04DD67 74 F0      MOV     A,#0xF0
04DD69 12 08 FF     LCALL  ?ALLOC_XSTACK8
04DD6C E9          MOV     A,R1
04DD6D FF     MOV     R7,A
osal_memcpy(zgPreConfigKey, defaultKey, SEC_KEY_LEN);
04DD6E 75 08 AD     MOV     V0,#0xAD
04DD71 75 09 31     MOV     V1,#0x31
04DD74 75 0A B7     MOV     V2,#0xB7
04DD77 78 08      MOV     R0,#0x08
04DD79 12 08 B9     LCALL  ?PUSH_XSTACK_I_THREE
04DD7C 7C 10     MOV     R4,#0x10
04DD7E 7D 00     MOV     R5,#0x00
04DD80 74 03     MOV     A,#0x03
04DD82 12 08 D9     LCALL  ?XSTACK_DISP101_8
04DD85 12 16 1A     LCALL  osal_memcpy_?relay
04DD88 74 03     MOV     A,#0x03
04DD8A 12 08 50     LCALL  ?DEALLOC_XSTACK8
status = osal_nv_item_init(ZCD_NV_PREFCFGKEY, SEC_KEY_LEN,
04DD8D A8 18     MOV     R0,XSP(L)
04DD8F A9 19     MOV     R1,XSP(H)
04DD91 88 08     MOV     V0,R0
04DD93 89 09     MOV     V1,R1
04DD95 78 08     MOV     R0,#0x08
04DD97 12 08 3D     LCALL  ?PUSH_XSTACK_I_TWO
04DD9A 7C 10     MOV     R4,#0x10
04DD9C 7D 00     MOV     R5,#0x00
04DD9E 7A 62     MOV     R2,#0x62
04DDA0 7B 00     MOV     R3,#0x00
04DDA2 12 17 4C     LCALL  osal_nv_item_init_?relay
04DDA5 74 03     MOV     A,#0x03
```

Memory

```
00003160 ca 0e 06 f4 0e 08 ee 0e 0b 0c 0f 10 46 0e 02 d0
00003170 0e 05 e2 0e 07 d6 0e 0e 0c 0e 0d be 0e ff 00 00
00003180 3e 13 6e 13 44 13 4a 13 68 13 5c 13 56 13 74 13
00003190 7a 13 80 13 8c 13 50 13 38 13 32 75 89 16 0f 00
000031a0 00 0b 15 0e 02 00 02 22 04 32 00 00 08 02 03
000031b0 08 0a 05 02 0d 03 0b 0c 0f 02 05 06 0a 69 67
000031c0 42 65 65 41 66 66 69 61 6e 63 65 30 39 00 00
000031d0 00 00 00 00 00 89 67 45 23 01 ef cd ab 2e 14 40
000031e0 14 46 14 09 13 4c 14 02 13 0e 13 14 0a e5 d5 c5
000031f0 45 a5 a5 45 45 85 75 65 42 12 3e 12 78 12 7a 12
```

Path

```
C:\Texas Instruments\ZStack-CC2530-2.5.1a\Components\stack\sys\ZGlobals.c
850 osal_memcpy(zgPreConfigKey, defaultKey, SEC_KEY_LEN);
```

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)

```
04DD6D  FF          MOV    R7,A
osal_memcpy( zgPreConfigKey, defaultKey, SEC_KEY_LEN );
04DD6E  75 08 AD     MOV    V0,#0xAD
04DD71  75 09 31     MOV    V1,#0x31
04DD74  75 0A B0     MOV    V2,#0x80
04DD77  78 08        MOV    R0,#0x08
04DD79  12 08 39     LCALL ?PUSH_XSTACK_I_THREE
04DD7C  7C 10        MOV    R4,#0x10
```

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)

The screenshot shows a firmware editor interface. On the left, a file tree lists configuration files like `f8wConfig.cfg`, `f8wCoord.cfg`, `f8wEndev.cfg`, and `f8wRouter.cfg`. The main window displays C-style comments for network configuration:

```
*/  
-DMAX_BINDING_CLUSTER_IDS=4  
  
/* Default security key. */  
//--DDEFAULT_KEY={0x02, 0x03, 0x05, 0x07, 0x09, 0x0B, 0x0D, 0x0F, 0x00, 0x02, 0x04, 0x06, 0x08, 0x0A, 0x0C, 0x0E}  
-DDEFAULT_KEY="{0x08, 0x02, 0x03, 0x08, 0x0A, 0x06, 0x02, 0x0D, 0x03, 0x0B, 0x0C, 0x0F, 0x02, 0x05, 0x06, 0x0F}"  
  
/* Reset when ASSERT occurs, otherwise flash LEDs */  
//--DASSERT_RESET  
  
/* Set the MAC MAX Frame Size (802.15.4 default is 102) */  
-DMAC_MAX_FRAME_SIZE=116
```

Below the code, the 'Go to' field is set to `0x31ad`. The memory dump shows the following data:

Address	Hex	ASCII
00003160	ca 0e 06 f4 0e 08 ee 0e 0b 0c 0f 10 46 0e 02 d0F.....
00003170	0e 05 e2 0e 07 d6 0e 0c ac 0e 0d be 0e ff 00 00>.n.D.J.h.\.V.t.
00003180	3e 13 6e 13 44 13 4a 13 68 13 5c 13 56 13 74 13	z.....P.8.2.....
00003190	7a 13 80 13 8c 13 50 13 38 13 32 13 60 16 0f 002.....Zig
000031a0	00 0b 15 0b 02 00 02 22 0f 32 00 00 06 08 02 03BeeAlliance09.....
000031b0	08 0a 06 02 0d 03 0b 0c 0f 02 05 06 0f 5a 69 67ge#.....@
000031c0	42 65 65 41 6c 6c 69 61 6e 63 65 30 39 00 00 00F...L.....
000031d0	00 00 00 00 00 89 67 45 23 01 ef cd ab 2e 14 40uer < x ~
000031e0	14 46 14 08 13 4c 14 02 13 0e 13 14 0a e5 d5 c5	
000031f0	b5 a5 a5 95 85 85 75 65 47 12 3c 12 78 12 7a 12	

Annotations in the image include a green arrow pointing from the `-DDEFAULT_KEY` code to the `0x31ad` address in the 'Go to' field, and a blue arrow pointing from the same code to the memory dump. A red box highlights the hex values `08 0a 06 02 0d 03 0b 0c 0f 02 05 06 0f 5a 69 67` in the memory dump, which correspond to the ASCII string 'BeeAlliance09'. A legend on the right identifies 'Default Network Key' in blue and 'Trust Center Link Key' in red.

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.

The screenshot shows the IDA Pro interface with the following windows:

- Functions window:** Lists functions with their segments and start addresses.
- Occurrences of binary: 75 08 ? 75 09 ? 75 0a ? 78:** Shows a list of addresses and instructions. A red circle highlights the first address, seg000:000187FE.
- Occurrences of binary: 75 08 ? 75 09 ? 75 0a ? 78:** Shows a list of addresses and instructions. A red circle highlights the first address, seg000:000187FE.

The list of addresses and instructions in the 'Occurrences of binary' window is as follows:

Address	Function	Instruction
seg000:000187FE		
seg000:00018A78		
seg000:00018D62		
seg000:000191F5		
seg000:0001DE3F		
seg000:0001E1BF		
seg000:00021AD4		
seg000:00021AEE		
seg000:00021B20		
seg000:0002460C		
seg000:00024874		

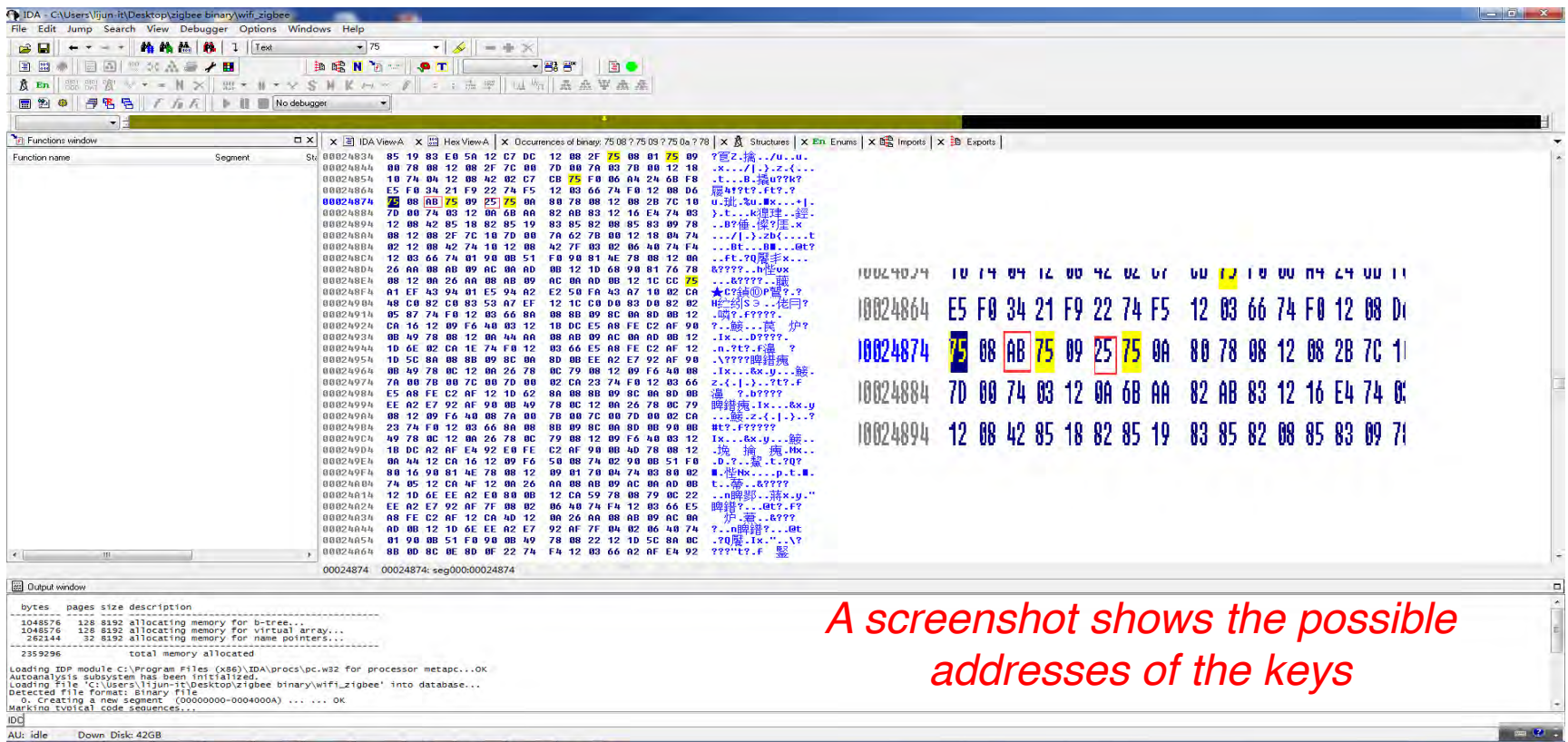
The output window shows the following text:

```
bytes pages size description
-----
1048576 128 8192 allocating memory for b-tree...
1048576 328 8192 allocating memory for virtual array...
262144 32 8192 allocating memory for name pointers...
-----
2359296 total memory allocated

Loading IDP module C:\Program Files (x86)\IDA\procs\pc.w32 for processor metapc...OK
Autoanalysis subsystem has been initialized.
Loading File 'C:\Users\11jun-it\Desktop\zigbee binary\wifi_zigbee' into database...
Detected file format: Binary File
O. Creating a new segment (00000000-0004000A) ... OK
Marking typical code sequences...
IDP
AU: idle Down Disk: 42GB
```

Find the encryption key from firmware

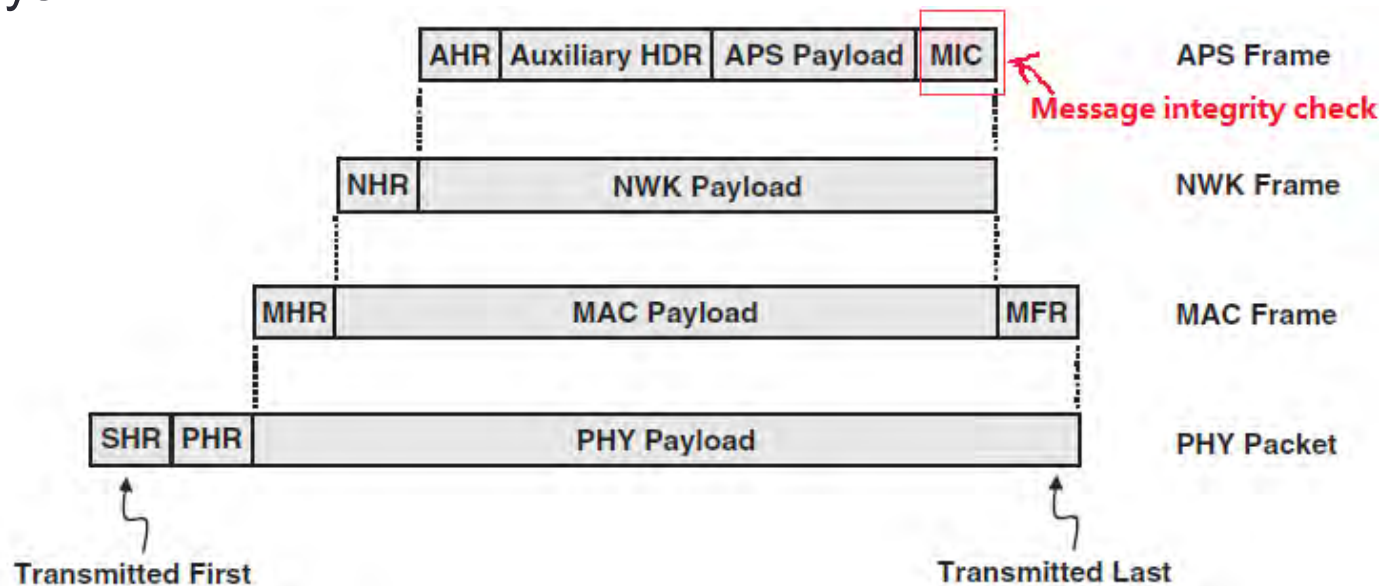
Then we use the four consecutive move instructions' corresponding machine code and operand (75 08 ? 75 09 ? 75 0A) as a filter to search through the firmware for the address of the keys.



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.



ZigBee Packet structure

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.



A sniffer used to capture the packets



360UNICORNTTEAM

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.

The screenshot displays the Wireshark interface for ZigBee 2007/PRO traffic. The packet list on the left shows several frames, including MAC payloads and network layer decoded packets. The main pane shows a detailed view of a network layer decoded packet (Sequence number 13). The packet structure includes fields for Source PAN Address, Source Address, and various control fields. A red box highlights the 'Sec Flag' field, which is set to 1. A red arrow points to this field with the text: "After the receipt of network key the Sec Flag changed from 0 to 1 indicating the payload have been encrypted!". Other annotations include "Anybody here?", "I am here!", "I want to join your network!", "Give me a PANID, please!", "Here is your PANID!", "And here is our network key!", and "Here is the network key".

Find the encryption key by sniffing

Texas Instruments SmartRF Packet Sniffer IEEE 802.15.4 MAC and ZigBee 2007/PRO

File Settings Help

ZigBee 2007/PRO

Pnbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	Source PAN	Superframe specification	GTS fields	Beacon payload	Beacon Payload (NWK Layer Decoded)	TX Offset [symbols]	RSSI (dBm)	FCS
RX 4	+675463 =12261953	10	Type Sec Pnd Ack.req PAN_compr CMD 0 0 0 0	0x4A	0xFFFF	0xFFFF	0x4A	0x0000	BO 30 F.CAP RLE Coord Assoc 15 15 15 0 1 1	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-68	OK
Anybody here ?															
RX 5	+2692 =12814645	28	Type Sec Pnd Ack.req PAN_compr BCN 0 0 0 0	0xF0	0x0080	0x0080	0x0080	0x0000	BO 30 F.CAP RLE Coord Assoc 15 15 15 0 1 1	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-68	OK
I am here !															
RX 6	+840179 =13654824	10	Type Sec Pnd Ack.req PAN_compr CMD 0 0 0 0	0x4B	0xFFFF	0xFFFF	0x4B	0x0000	BO 30 F.CAP RLE Coord Assoc 15 15 15 0 1 1	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-68	OK
RX 7	+3394 =13658218	28	Type Sec Pnd Ack.req PAN_compr BCN 0 0 0 0	0xF1	0x0080	0x0080	0x0080	0x0000	BO 30 F.CAP RLE Coord Assoc 15 15 15 0 1 1	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-69	OK
RX 8	+510317 =14168535	21	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 0	0x4C	0x0080	0x0000	0x0080	0xFFFF	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-72	OK
I want to join your network !															
RX 9	+1056 =14169991	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x4D	0x0080	0x0000	0x0080	0x0000	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-69	OK
OK !															
RX 10	+494703 =14664294	18	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x4D	0x0080	0x0000	0x0080	0x0000	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-72	OK
Data request Give me a PANID ,please !															
RX 11	+963 =14665257	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x4D	0x0080	0x0000	0x0080	0x0000	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-69	OK
OK !															
RX 12	+3094 =14668351	27	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x16	0x0080	0x0080	0x00124B0003D4A912	0x00124B0003D4BD6B	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-68	OK
Here is your PANID !															
RX 13	+1249 =14669600	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x16	0x0080	0x0000	0x00124B0003D4A912	0x00124B0003D4BD6B	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-72	OK
And here is our network key !															
RX 14	+108078 =14777678	56	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x17	0x0080	0x1593	0x0000	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-72	OK
Here is the network key															
RX 15	+2176 =14779854	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x17	0x0080	0x0000	0x00124B0003D4A912	0x00124B0003D4BD6B	0x00124B0003D4A912	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-72	OK
After the receipt of network key the Sec flag changed from 0 to 1 indicating the payload have been encrypted !															
RX 16	+37633 =14817487	57	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x4E	0x0080	0xFFFF	0x1593	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-67	OK
RX 17	+11477 =14828964	57	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x18	0x0080	0xFFFF	0x0000	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-79	OK
RX 18	+400308 =15229272	50	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x19	0x0080	0xFFFF	0x0000	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-67	OK
RX 19	+14653934 =29883206	50	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x4F	0x0080	0xFFFF	0x1593	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-79	OK
RX 20	+3714967 =30258173	50	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x1A	0x0080	0xFFFF	0x0000	0x0000	0x0000	0 0	***K*** K*****	Stk_Prof P_Ver Rtr_Cap Dev_Depth Dev_Cap Ext_PANID 0x2 0x2 0x1 0x0 0x1 0x00124B0003D4BD6B	0xFFFFF	-65	OK

“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:

Byte0	0x04
Byte1	Target PANID
Byte2	
Byte3	Unknown
Byte4	Mode
Byte5	Red
Byte6	Green
Byte7	Blue
Byte8	Illuminance
Byte9	Checksum

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 .

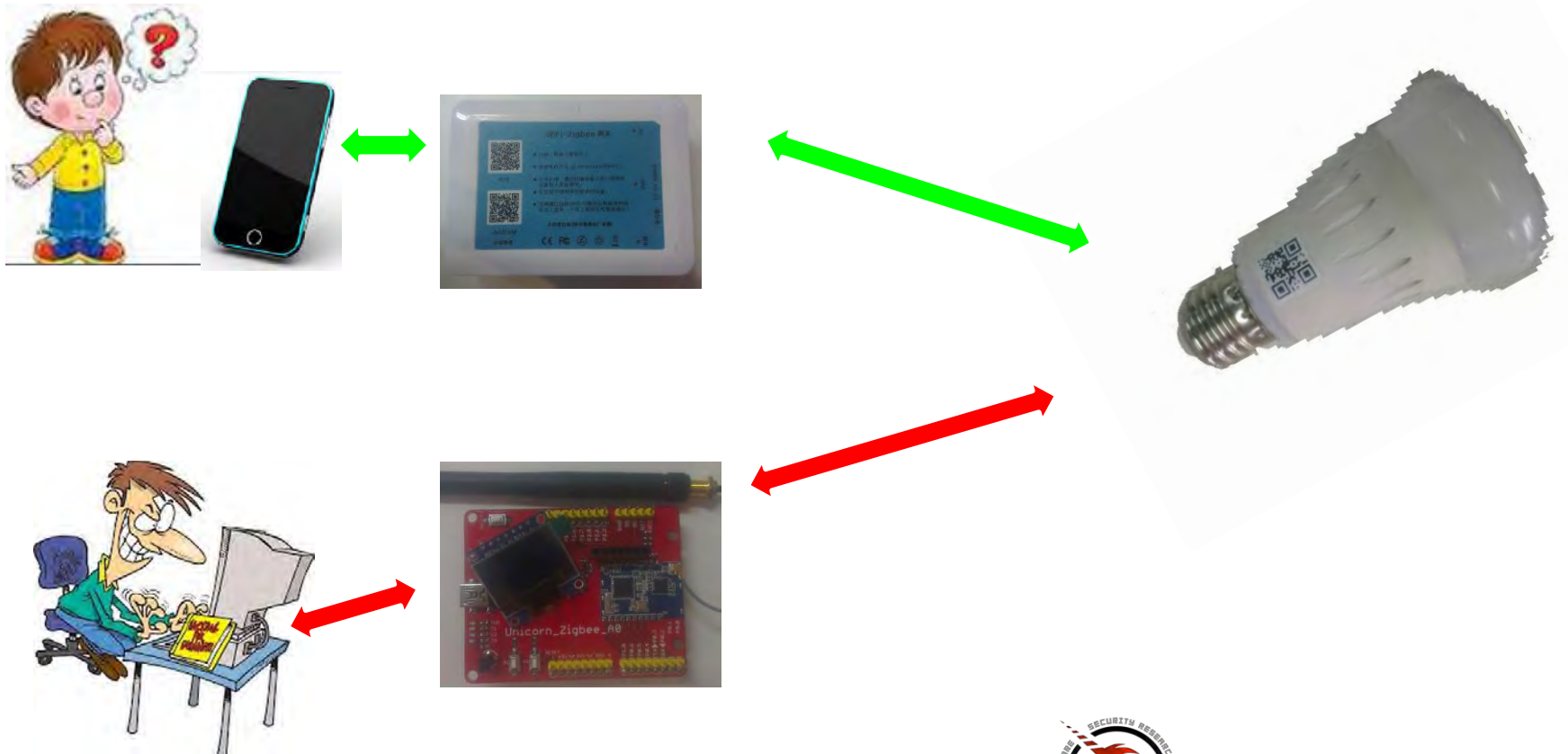
Byte0	0x04
Byte1	Target PANID
Byte2	
Byte3	Unknown
Byte4	Mode
Byte5	Red
Byte6	Green
Byte7	Blue
Byte8	Illuminance
Byte9	Checksum

Take control



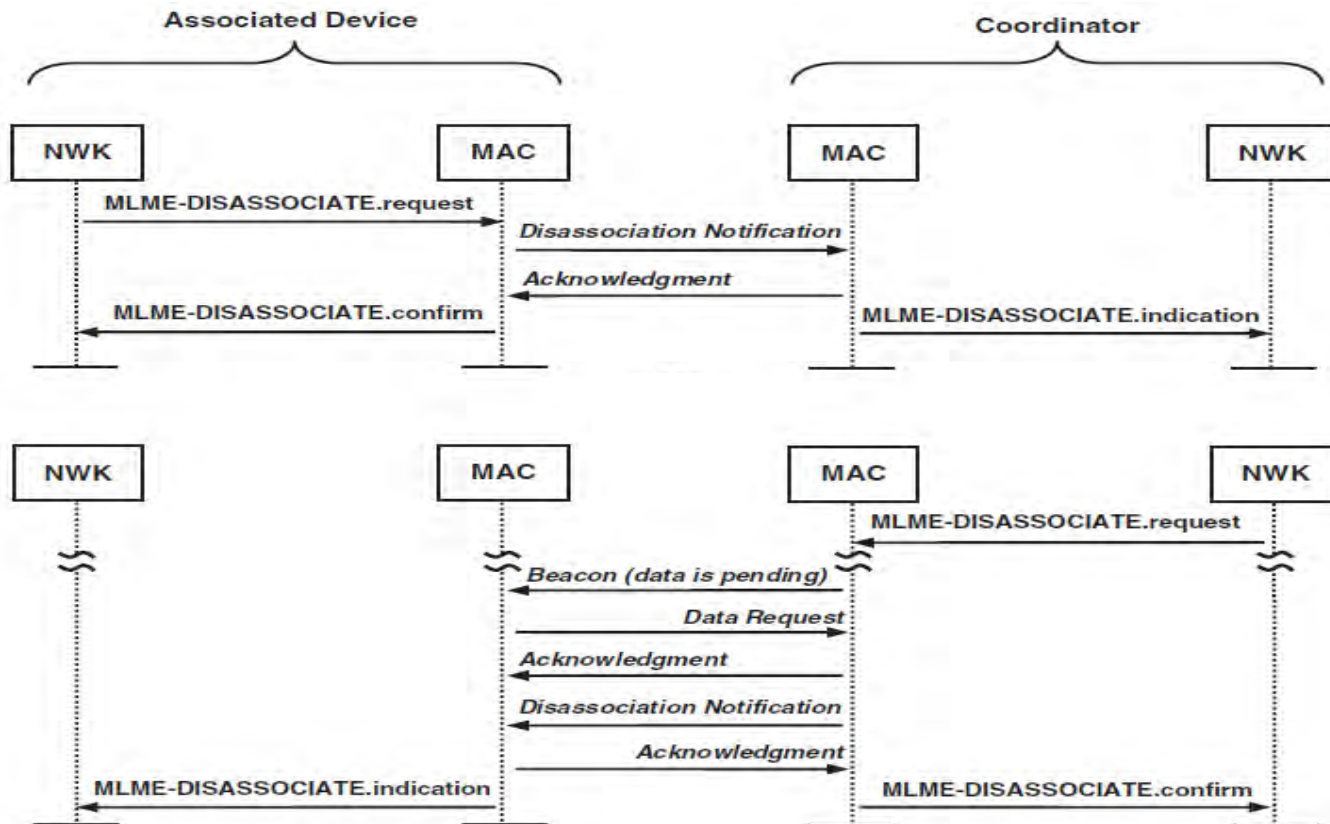
A Zigbee node we made and we used it to control the bulb

Replay & Spoof



360UNICORNTTEAM

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>
- *2) The KillerBee framework :*
- <https://github.com/riverloopsec/killerbee>
- *3) Paperback book entitled “ZigBee Wireless Networks and Transceivers”*
- *2008, Elsevier Ltd . ISBN: 978-0-7506-8393-7*
- *4) Paper “Recommended Practices Guide For Securing ZigBee Wireless Networks in Process Control System Environments”*
- *April 2007.Author:Ken Masica. Lawrence Livermore National Laboratory*
- *5) Paper “ZigBee Security” ©2009 ZigBee Alliance.*
- *Author:Robert Cragie.Chair, ZigBee Alliance ZARC Security Task Group.Principal Engineer, Jennic Ltd.*
- *6) A webpage :*
- <http://www.ciscopress.com/articles/article.asp?p=1823368&seqNum=4>

Acknowledgment

ZHANG Kai

- Twitter: [peekair_zhang](#)



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

Nikita

- Twitter: [Niki7a](#)

Thank you!