

Department of Computer Science

CSE 4820: Wireless and Mobile Security

15. Zigbee Security

Dr. Abdullah Aydeger

Location: Harris Inst # 310

Email: aaydeger@fit.edu

Outline

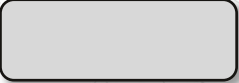
Attacking Zigbee

Recall: Zigbee Overview

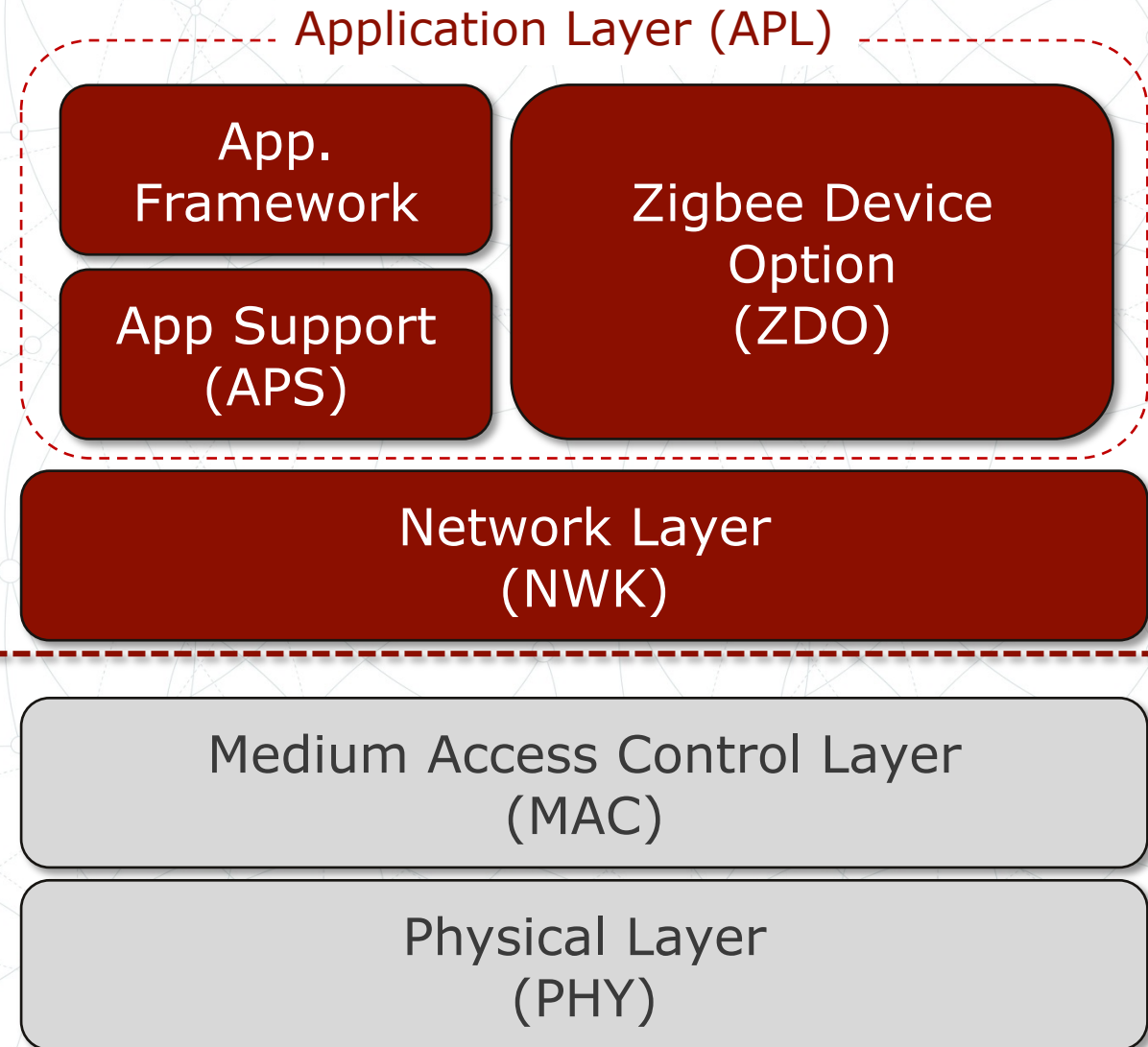
Solution	Description
Network Protocol	Zigbee PRO 2015 (or newer)
Network Topology	Self-Forming, Self-Healing MESH
Network Device Types	Coordinator (routing capable), Router, End Device, Zigbee Green Power Device
Net. Size (theoretical # of nodes)	Up to 65,000
Radio Technology	IEEE 802.15.4-2011
Frequency Band / Channels	2.4 GHz (ISM band) 16-channels (2 MHz wide)
Data Rate	250 Kbits/sec
Security Models	Centralized (with Install Codes support) Distributed
Encryption Support	AES-128 at Network Layer AES-128 available at Application Layer
Communication Range (Avg)	Up to 300+ meters (line of sight) Up to 75-100 meter indoor
Low Power Support	Sleeping End Devices Zigbee Green Power Devices (energy harvesting)
Legacy Profile Support	Zigbee 3 devices can join legacy Zigbee profile networks. Legacy devices may join Zigbee 3 networks (based on network's security policy)
Logical device support	Each physical device may support up to 240 end-points (logical devices)

Recall: ZigBee Layers

 Defined by Zigbee Alliance

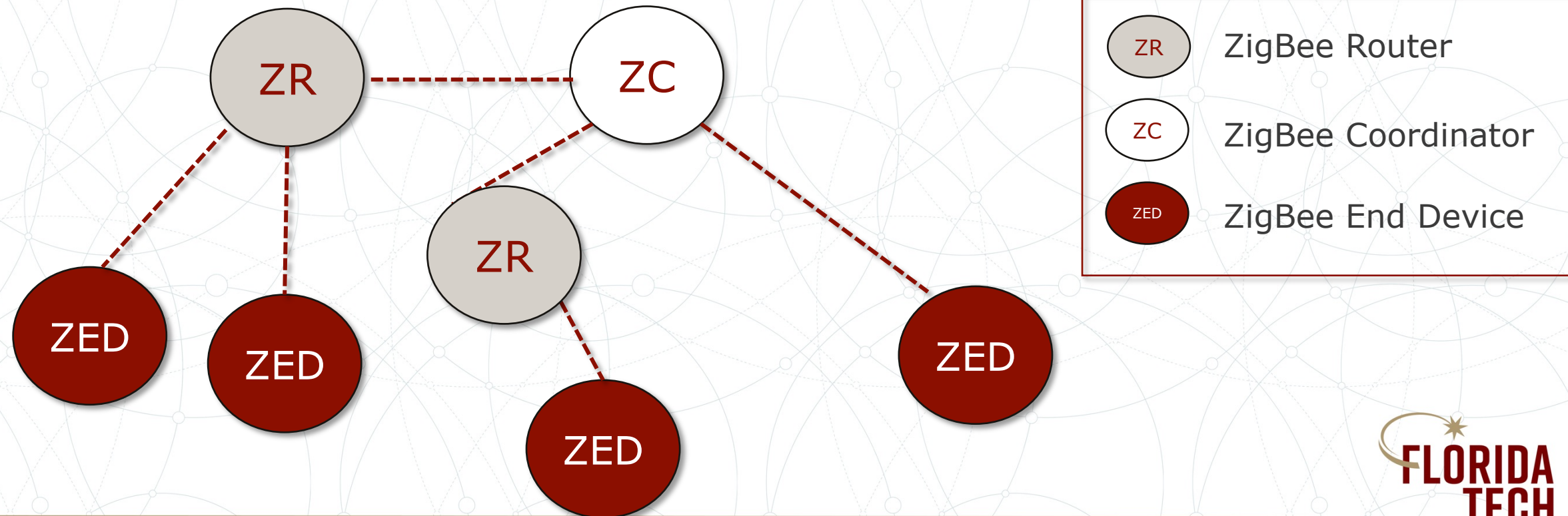
 Defined in IEEE 802.15.4
(Low-rate wireless personal
area network)

<https://csa-iot.org/all-solutions/zigbee/>



Recall: Example Zigbee Network

- One ZC for the network, additional ZRs



Recall: Zigbee: Key Provisioning

- Significant challenge; process of provisioning, rotating, and revoking keys on devices
- Zigbee Pro; Administrator can use the SKKE method to derive the network and link keys on devices
 - Requires devices to have master key provisioned on the TC and device joining the network

Recall: Zigbee Attacks

- KillerBee:
 - Python-based framework for manipulating and penetration testing Zigbee and IEEE 802.15.4 networks
 - Written and tested on Linux, free and open-source
 - Includes support for Scapy
 - Includes a variety of tools including zbwreshark, zbdump, and zbreplay

<https://github.com/riverloopsec/killerbee>

The screenshot shows the GitHub repository page for 'killerbee' by 'riverloopsec'. The repository is a Python-based framework for manipulating and penetration testing Zigbee and IEEE 802.15.4 networks. It has 748,740 commits, 398 commits, and 210 forks. The repository is licensed under the MIT license. The repository includes a README, a LICENSE, and a setup.py file. The repository is also featured in the IEEE 802.15.4/ZigBee Security Research Toolkit.

File	Description	Time
doc	Removed extra level of killerbee folder structure per discussion with...	8 years ago
firmware	Bugfix/cc2531 assorted (#252)	7 months ago
killerbee	Bugfix/cc2531 assorted (#252)	7 months ago
sample	Removed extra level of killerbee folder structure per discussion with...	8 years ago
scripts	implemented goodfet and apimote changes, added mac install instru...	16 months ago
tests	Sewio updates (#233)	16 months ago
tools	Restore APS CMD payload parsing for NWK transport key disclosure	3 months ago
zigbee_crypt	py3 support - merged secureab	3 years ago
.gitignore	Bugfix/cc2531 assorted (#252)	7 months ago
ARCHITECTURE.md	Clean up main readme doc for readability, update other docs with py...	2 years ago
DEVELOPMENT.md	Clean up main readme doc for readability, update other docs with py...	2 years ago
FAQ.md	implemented goodfet and apimote changes, added mac install instru...	16 months ago
LICENSE.txt	Removed extra level of killerbee folder structure per discussion with...	8 years ago
README.md	Goodfet update - bugfixes (#232)	16 months ago
setup.py	Bugfix/cc2531 assorted (#252)	7 months ago

IEEE 802.15.4/ZigBee Security Research Toolkit

www.riverloopsecurity.com

Readme

View license

657 stars

47 watching

210 forks

Releases 5

3.0.0-beta.2 (Latest)

on Jul 14, 2021

+ 4 releases

Packages

No packages published

Contributors 29

Zigbee: Network Discovery

- First assessment is to discover networks within range and enumerate the configuration of devices
 - Simple way; mimic Zigbee network discovery process with Killerbee
- Part of network discovery process in Zigbee Standard, ZDEs transmit beacon request on a given channel
 - All ZR and ZCs receiving beacon -> respond by sending a beacon frame
 - Disclose PAN ID, ZC or ZR source address, stack profile/version, extended IEEE address information
- Using same technique to actively scan for the presence of Zigbee network

Zigbee: Network Discovery

- Killerbee tool zbstumbler (similar to Wifi discovery tool Netstumbler):
 - Channel hops and transmits beacon request frames
 - Every two seconds hopping to a new channel
 - Display useful information from response beacon frames

```
test@test-HP-EliteBook-840-G3:~/killerbee$ sudo zbstumbler
[sudo] password for test:
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:11'
New Network: PANID 0xC762 Source 0xE2DA
Ext PANID: 79:21:70:53:a3:d7:fc:34 Stack Profile: ZigBee Enterprise
Stack Version: ZigBee 2006/2007
Channel: 11
New Network: PANID 0xC762 Source 0xFF4F
Ext PANID: 79:21:70:53:a3:d7:fc:34 Stack Profile: ZigBee Enterprise
Stack Version: ZigBee 2006/2007
Channel: 11
```

Zigbee Network Scanning Countermeasure

- Beacon request mechanism is integral to Zigbee
 - Cannot be disabled
 - Attacker can use it freely
- Best countermeasure is to understand the impact and evaluate your own networks to identify the information attacker can gain

Eavesdropping Attacks

- Zigbee networks are mostly not encrypted
 - Extremely easy to eavesdrop
- Even if it uses encryption
 - Many unencrypted fields useful; MAC header, config of network, node address and PAN ID
 - Might substitute network discovery?
- Killerbee zbdump -> similar to tcpdump

No.	Time	Source	Destination	Protocol	Length	Info
19	11.657403	0x0000	Broadcast	ZigBee	51	Command, Dst: Broadcast, Src: 0x0000
20	13.098522	0x0000		ZigBee	28	Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5
21	15.596531		Broadcast	IEEE 8...	10	Beacon Request
22	15.724666	0x0000		ZigBee	28	Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5
23	16.493441	0x0000		ZigBee	28	Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5
24	16.621446	0x0000		ZigBee	28	Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5
25	16.749531			IEEE 8...	5	Ack
26	16.949566			IEEE 8...	111	Ack, Bad FCS
27	17.021844	0x42c9	0x0000	IEEE 8...	97	Data, Dst: 0x0000, Src: 0x42c9, Bad FCS
28	17.061898	0x0000	0x42c9	IEEE 8...	113	Data, Dst: 0x42c9, Src: 0x0000, Bad FCS
29	17.093841	0x0000	0x42c9	ZigBee	50	Data, Dst: 0x42c9, Src: 0x0000
30	17.132974			IEEE 8...	57	Ack, Bad FCS
31	17.164971	0x0000	0x42c9	IEEE 8...	56	Data, Dst: 0x42c9, Src: 0x0000, Bad FCS
32	17.196972	0x0000	0x42c9	IEEE 8...	51	Data, Dst: 0x42c9, Src: 0x0000, Bad FCS
33	17.237021	0x0000	0x42c9	IEEE 8...	108	Data, Dst: 0x42c9, Src: 0x0000, Bad FCS
34	17.309167	0x0000	0x42c9	IEEE 8...	105	Data, Dst: 0x42c9, Src: 0x0000, Bad FCS
35	17.437044	00:15:5f:00:b4:4d:2...	0x0000	IEEE 8...	27	Association Request, RFD, Bad FCS
36	17.956659	00:15:5f:00:b4:4d:2...	0x0000	IEEE 8...	86	Data Request, Bad FCS
37	18.380939	0x87c4	0x0000	IEEE 8...	90	Data Request, Bad FCS
38	18.420964	0x87c4	0x0000	IEEE 8...	73	Data Request, Bad FCS
39	18.684087	0x87c4	0x0000	IEEE 8...	67	Data, Dst: 0x0000, Src: 0x87c4, Bad FCS

► Frame 27: 97 bytes on wire (776 bits), 97 bytes captured (776 bits)

▼ IEEE 802.15.4 Data, Dst: 0x0000, Src: 0x42c9, Bad FCS

► Frame Control Field: 0x8861, Frame Type: Data, Acknowledge Request, PAN ID Compression, Destination Addressing Mode: Short/16-bit, Frame Ver...

Sequence Number: 11

Destination PAN: 0x872b

Destination: 0x0000

Source: 0x42c9

FCS: 0x6c04 (Incorrect, expected FCS=0xc2aa)

► [Expert Info (Warning/Checksum): Bad FCS]

► Data (86 bytes)

DoS Zigbee

- Silva/Nunes attack exploits a flaw in how recipients process inbound packets with regard to the IEEE 802.15.4 frame counter (FC) value
- When a transmitting node sends a secure packet, it includes a sequential frame counter value in each frame with a range of 0 to 0xffffffff-1
 - FC value is not encrypted but it is included in the calculation of MIC (Message Integrity Check) for a packet

DoS Zigbee

- A receiving node remembers the last observed FC value for all of the nodes on the network
- To defeat replay attacks and avoid reprocessing packet retransmissions, receiving node only accepts packets with greater FC than last observed
- FC is also used to make the 'nonce' unique for each packet transmitted by a specific node

DoS Zigbee

- The unique nonce is important for AES-CTR to avoid initialization vector collision
 - Attacker can use plaintext / ciphertext data to get the key, if collision
- IEEE 802.15.4 specifies that when FC is equal to 0xffffffff, receiving node must stop processing all further data from the device
 - Add the transmitter to a device blacklist
 - Only way to recover, administrator updating the network key on all devices (firmware update)

DoS Zigbee

- Under intended use circumstances in IEEE 802.15.4, devices are not likely to reach max FC value
 - 1 packet per second, a node will need 136 years to reach
- If node receives packet with increasing FC, it will accept and update FC value prior to validating the encrypted packet
 - Attacker forges 0xffffffff-1 and blacklist the legitimate transmitter

Zbscapy

```
$ sudo zbscapy
```

```
>>> kb = KillerBee()
```

```
>>> conf.killerbee_channel = 15
```

```
>>> f = Dot15d4()/Dot15d4Cmd(cmd_id=7)
```

```
>>> kbsendp(f,iface=kb)
```

```
# start killerbee framework
```

```
# acquire killerbee-enabled device
```

```
# set to channel #15
```

```
# craft a IEEE 802.15.4 Command Frame  
# with the CMD_ID = 7 (Beacon Req)
```

```
# send the frame
```


Thank you. Questions?

Dr. Abdullah Aydeger