A Lightbulb Worm?
A teardown of the Philips Hue.

Colin O’Flynn
(with special appearance by Eyal Ronen)
ChipWhisperer-Lite: A New Era of Hardware Security Research

Embedded security - is it an oxymoron? Learn the truth through a series of hands-on labs targeting computer and electrical engineers.

Created by
Colin O’Flynn

331 backers pledged $88,535 CAD to help bring this project to life.
HACKS?

2. Move bulb onto unavailable network, or control bulb.
3. Hack into bridge, access ethernet.
4. Malware in bulbs to do #3?
Understanding ZLL

1. BRIDGE
   - Detection

2. Random Link Key
   - ZLL Master Key

3. ZLL Traffic

New Bulb
- Detection
- Decrypt using ZLL Master Key

ZLL Traffic
6.4.4 Stealing a Node

A node that is already part of a ZLL network can be taken or ‘stolen’ by another ZLL network using Touchlink (in which case, the stolen node will cease to be a member of its previous network). This transfer can only be performed on a node which supports one or more Lighting devices (and not Controller devices).

The node is stolen using an initiator in the new network, e.g. from a remote control unit. The ‘stealing’ process is as follows:

1. The initiator sends a Scan Request to nodes in its vicinity. The required function is:
   
   ```
   eCLD_ZLLCommissionCommandScanReqCommandSend()
   ```

2. A receiving ZLL node replies to the Scan Request by sending a Scan Response. The required function is:

   ```
   eCLD_ZLLCommissionCommandScanRspCommandSend()
   ```

3. The initiator receives Scan Responses from one or more nodes and, based on these responses, selects a node (containing a Lighting device) that is already a member of another ZLL network.

4. The initiator then sends a Reset To Factory New Request to the desired node. The required function is:

   ```
   eCLD_ZLLCommissionCommandFactoryResetReqCommandSend()
   ```

5. On receiving this request on the target node, the event `E_CLD_COMMISSION_CMD_FACTORY_RESET_REQ` is generated and the function `ZPS_eApiZdoLeaveNetwork()` should be called. In addition, all persistent data should be reset.

6. The node can then be commissioned into the new network by following the process in Section 6.4.2 from Step3.
8.1.2 Channels

A ZLL device shall be able to operate on all channels available at 2.4GHz, numbered from 11 to 26. When operating on channel 26, the transmission power may be reduced in order to comply with FCC regulations.

Within this range, two sets of channels shall be defined. The primary ZLL channel set shall consist of channels 11, 15, 20 and 25 and shall be used in preference for commissioning and normal operations. The secondary ZLL channel set shall consist of channels 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24 and 26 and can be used as a backup to allow the ZLL device to connect to a non-ZLL network.
Demo by Eyal Ronen

See http://www.wisdom.weizmann.ac.il/~eyalro/
LIGHT BULB MALWARE OR (BRICKS)

Cloud

BASE

Encrypted with link key

Encrypted?
LIGHT BULB MALWARE

1) ZLL key leaked. We know it's possible to "steal" bulbs.

2) Custom FW on bulbs could turn bulb into "bridge" that searches for & steals nearby bulbs.

3) If could cause other bulbs to perform OTA FW update → WORM
CHEAP BULBS
TX

+3.3V in (avoid killing yourself)
[Log, Info, ConnectedLamp, MCUCR=0x00, LockBits=0xFC, LowFuse=0xF6, HighFuse=0x9A, ExtFuse=0xFE]
[Log, Info, ConnectedLamp, devsig=0x1EA803]
[Log, Info, S_DeviceInfo, Booting into normal mode...]
[Log, Info, S_DeviceInfo, DeviceId: Bulb_A19_DimmableWhite_v2]
[Log, Info, N_Security, LIB4.5.75]
[Log, Info, N_Security, KeyBitMask, 0x0012]
[Log, Info, ConnectedLamp, Platform version 0.41.0.1, package_ZigBee 117, package_BC_Stack 104, svn 26632]
[Log, Info, ConnectedLamp, Product version WhiteLamp-Atmel 5.38.1.15095, built by LouvreZLL]
[Log, Info, A_Commissioning, Factory New at Ch: 11]
[TH, Ready, 0]
[Log,Info,ConnectedLamp,MCUCR=0x00,LockBits=0xFC,LowFuse=0xF6,HighFuse=0x9A,ExtFuse=0x0E]
[Log,Info,ConnectedLamp,devsig=0x1EA803]
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: Bulb_A19_DimmableWhite_v2]
[Log,Info,N_Security,LIB4.5.75]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,ConnectedLamp,Platform version 0.41.0.1,package_ZigBee=117,package_BC_Stack=104,svn 26632]
[Log,Info,ConnectedLamp,Product version WhiteLamp-Atmel 5.38.1.15095,built by LouvreZLL]
[Log,Info,A_Commissioning,Factory New at Ch: 11]
[TH,Ready,0]
[Sys,test,1]
[SYS,Error,Incorrect format]

Working serial input too!
See white-paper for JTAG pin-out connections.
a. Hold SPI line low, notice ASSERT printed matches same name-types used (NVs)

b. Can find same print statements

[TH, Ready, 0]
[Log, Info, N_Connection, Starting discovery for updated networks]
[Log, Info, N_Connection, Discovery for updated networks completed]
/
*/
16 byte sector header used in flash located at the start of the active sector. */

typedef struct SectorHeader_t
{
    /*
     ** Is this sector active. Written with 0x0000 at the end of the compact operation. */
    uint16_t isActive;
     ** Signature to detect valid sectors. Must have the value "0x0000". */
    uint8_t signature[6];
    /*
     ** Counter, decreased each time a new sector becomes the active sector. */
    uint32_t sequenceNumber;
    /*
     ** Parity bits for the sequenceNumber field = sequenceNumber ^ 0xFFFFFFFFFULL. */
    uint32_t sequenceParity;
} SectorHeader_t;
Damn.

December 03, 2014

Lamp software version: 66013452
- Related products are hue A19 and BR30 downlight bulbs and Friends of hue
- Hue Tap range is extended if lamp in between Tap and bridge is powered
- Faster start-up when using the wall switch
- Bug fixes and stability improvements

Not Atmel bugged.
BRIDGE 1.0 HACKING

Bridge

Dumb hub

Run App
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>8500</td>
<td>11:17</td>
<td>192.368.0.23</td>
<td>5.70.62.91</td>
<td>TCP</td>
<td>60</td>
<td>40640 &gt; [FIN, ACK] Seq=1621 Ack=873 Win=1800 Len=0</td>
</tr>
<tr>
<td>8501</td>
<td>11:17</td>
<td>192.368.0.23</td>
<td>102.168.0.23</td>
<td>DNS</td>
<td>79</td>
<td>Standard query Oxsi1 A fes.cpp.http/1.1</td>
</tr>
<tr>
<td>8502</td>
<td>11:17</td>
<td>192.368.0.23</td>
<td>102.168.0.23</td>
<td>DNS</td>
<td>122</td>
<td>Standard query response Oxsi1 CNAME a20149.g2.ajcm21.net A 173.237.125.64 A</td>
</tr>
<tr>
<td>8503</td>
<td>11:17</td>
<td>192.368.0.23</td>
<td>173.237.125.64</td>
<td>TCP</td>
<td>60</td>
<td>40641 &gt; [FIN, ACK] Seq=14 Win=294 Len=0 MSS=384</td>
</tr>
<tr>
<td>8504</td>
<td>11:17</td>
<td>173.237.125.64</td>
<td>102.168.0.23</td>
<td>TCP</td>
<td>64</td>
<td>http: 43041 [FIN, ACK] Seq=0 Ack=1 Win=1460 Len=0 MSS=336 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]</td>
</tr>
<tr>
<td>8505</td>
<td>11:17</td>
<td>173.237.125.64</td>
<td>102.168.0.23</td>
<td>TCP</td>
<td>60</td>
<td>40641 &gt; [TCP] Seq=4 Ack=3 Win=234 Len=0</td>
</tr>
<tr>
<td>8506</td>
<td>11:17</td>
<td>173.237.125.64</td>
<td>102.168.0.23</td>
<td>TCP</td>
<td>500</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
</tbody>
</table>

http://xxx/firmware/HUE0100/66013452/ConnectedLamp-Target_0012_13452_8D.sbl-ota

http://xxx/firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw
Output from CC2530

[Log, Info, S_DeviceInfo, Booting into normal mode...]
[Log, Info, S_DeviceInfo, DeviceId: IpBridge]
[Log, Info, N_Security, LIB4.4.52]
[Log, Info, N_Security, KeyBitMask, 0x0012]
[Log, Info, A_Bridge, Platform version 0.25.0, package_ZigBee 8720, package_Z_Stack 8720, built by LouvreZLL]
[Log, Info, A_Bridge, Product version 5.7.1, SmartBridge 11393, built by LouvreZLL]
[Bridge, Version, 5.7.1, SmartBridge 11393, built by LouvreZLL]
[Bridge, GroupRange, 0x5357, 0x5367]
[Log, Info, D_Led, dc 16]
[Bridge, NetworkSettings, False, 0xB163, 26DF52A183D85889, 11, 0, S=0x0001]
[Log, Info, A_Bridge, NwkAddr: 0x0001, Ch: 11, Pan: 0xB163, NwkUpdId: 0, ExtPanID: 26:DF:52:A1:83:D8:58:89]
[Log, Info, D_Led, dc 16]
[TH, Ready, 0]
[Connection, A]
[Connection, GetAddress, L=00:17:88:01:01:07:BF:FC, S=0x0001.0]
[Bridge, StoreGroupRange, 0]
[Log, Info, N_ConnectionRouter, Startup network discovery...]
<table>
<thead>
<tr>
<th>Action</th>
<th>Device ID</th>
<th>Command ID</th>
<th>Cluster ID</th>
<th>Attribute ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection, GetAddress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bridge, StoreGroupRange</td>
<td>0x5357, 0x5367</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zcl, S, S=0x0002.11, 6, 00000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing, ClearEntry, 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing, SendMtoRR, True</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zcl, S, S=0x0003.11, 6, 0001000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing, ClearEntry, 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing, SendMtoRR, True</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zcl, S, S=0x0002.11, 6, 0002000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zcl, S, S=0x0003.11, 6, 0003000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zcl, S, S=0x0002.11, 6, 0004000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Serial Data During Bootload

Data
Larger Delay during Page erase

Each data block = 64 bytes

First Page
Second Page

One page = 512 Bytes
Extracting Keys from Second Generation Zigbee Chips

Travis Goodspeed
1933 Black Oak Street
Jefferson City, TN, USA
travis@radiantmachines.com

ABSTRACT
First generation Zigbee chips were SPI slaves with no internal processing beyond cryptographic acceleration. Extracting a key was as simple as spying on the SPI transactions. The second generation chips, typified by the CC2430 from Texas Instruments and the EM250 from Ember, contain both a microcontroller and a radio, making the SPI sniffing attack all but irrelevant. Nevertheless, both chips are vulnerable to local key extraction. This paper describes techniques for doing so, focusing on the CC2430 as the EM250 has no protection against outside access. Recommendations are made for defending CC2430 firmware by using compiler directives to place sensitive information in flash memory, rather than in RAM. All Chipcon radios with 8051 cores released prior to the publication of this paper are expected to be vulnerable.

Keywords
Zigbee, CC2430, EM250, System on a Chip (SoC)

1. GENERATIONS
First generation Zigbee chips, such as the CC2420, were simply digital radios with SPI interfaces and a bit of hardware-accelerated cryptography. They could not run a Zigbee stack themselves, but rather relied upon an external microcontroller cores were added for convenience, not security, as will be explained below.

The third generation of chips will include more powerful microprocessors and—hopefully—a lot more security. The offering from Texas Instruments is the CC430 family, based upon the MSP430X2 processor. Ember will be using the Arm Cortex M3 in its EM300 series. These chips are out of scope for this paper, as they are not yet commercially available. Also, Freescale’s line of radios have not yet been examined by the author, but they will be in the near future.

2. CONCERNING THE EM250
The Ember EM250 contains a 16-bit XAP2b microprocessor from Cambridge Consultants Ltd.[3] Debugging support is provided by that firm’s proprietary SIF protocol, with hardware and software available only through Ember. SIF itself is a variant of JTAG.

While the datasheet and various piece of marketing literature claim “The EM250 employs a configurable memory protection scheme usually found on larger microcontrollers.”, this refers not to a debugging fuse or bootloader password, but rather to protection from accidental self-corruption of memory. This is in the form of Application/System separation, allowing the EmberZNet stack to defend certain regions from modification.
Good Things

• ZLL master key not in regular SRAM

• Tried AES-128 CBC to decrypt bootloader image, where key = \{every possible 16-byte block\}

  LDNo success, key not in SRAM?
TX BUFFER ATTACK

for(uint8 c=0; c < data_to_send; c++) {
    uart_write(tx_buf[c]);
}

Tx Buffer
for(uint8_t i=0; i < data_to_send; i++) {
    uart_write(tx_buf[i]);
}

Glitch Attack!
Custom PCB

CC2530 from Bridge
Clock Glitching
Original Clock
7.37 MHz

Width = 10%
Offset = +15%

Clock XOR'd with Glitch
Appears section of SRAM is erased after use.

This is good practice!

May be possible with more glitches.
Glitch Attacks To Firmware

- Appears we can use glitching to dump SRAM.
- Careful timing required to get decrypted data.
R.I.P.
ZigBee Soc

USB Serial

QCA4005

DDR

Qualcom

Boot Flash

SFF Nano Flash
Short boot ~3s after starts

https://www.youtube.com/watch?v=hi2D2MnwiGM
Or: http://www.oflynn.com
eth1: 00:17:88:24:15:8e
athrs27_phy_setup ATHR_PHY_CONTROL 0 : 1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 0 : 10
athrs27_phy_setup ATHR_PHY_CONTROL 1 : 1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 1 : 10
athrs27_phy_setup ATHR_PHY_CONTROL 2 : 1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 2 : 10
athrs27_phy_setup ATHR_PHY_CONTROL 3 : 1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 3 : 10
eth1 up
eth0, eth1
Qualcomm Atheros SPI NAND Driver, Version 0.1 (c) 201
ath_spi_nand_ecc: Couldn't enable internal ECC
Setting 0x181162c0 to 0x4b9fa100
Hit any key to stop autoboot: 0

** Device 0 not available
ath>
Use "mkpasswd"

ath> setenv bootdelay 3
ath> printenv security

***COPY THE DEFAULT VALUE THAT WAS PRINTED & SAVE THIS SOMEWHERE***

ath> setenv security '$5$wbgtEC1iF$ugIfQUoE7SNg4mplDI/7xdfLC7jXoMAkupeMsm10hY9'
ath> printenv security
security=$5$wbgtEC1iF$ugIfQUoE7SNg4mplDI/7xdfLC7jXoMAkupeMsm10hY9
ath> saveenv
ath> reset

https://www.youtube.com/watch?v=hi2D2MnwiGM
http://colinoflynn.com/?p=706
• Master binary seems to “do it all” (webserver, parsing requests, etc.) at /usr/sbin/ipbridge

• FW Update routine at /usr/sbin/swupdate
  • References AES-CBC-256 decryption routine, which references encryption key at /home/swupdate/certs/enc.k

• Two different bridges used same AES key (not really a big deal, as we already have unencrypted binaries since we have root).
Previous slide: power signature of first 64-byte block sent (sign-on info?).
This slide: Power signature for remaining 64-byte blocks (delay varies).
64 Byte Decryption

ECB
CBC
CTR
BR30
Downlight
Using Salae Pro-16
to capture 8MHz SPI traffic
SRAM Dump during Bootload

That block from previous page.

Reset (entering debug)
SECURITY CONCLUSIONS
1. Huge risk to Philips if worm designed.

2. Good security practices in place to prevent this:
   - Encrypted FW
   - Keep Keys out of SRAM
   - Signed FW (Linux only)
   - Clear memory when done.

3. Trade-offs may cause future problems:
   - Same Key decrypts FW updates across many devices.
   - ZLL master key leak opens up lamp-stealing.
   - Huge Linux binary does a lot, vulnerabilities.
   - See White Paper for more!
Eyal Ronen
http://www.wisdom.weizmann.ac.il/~eyalro/

Colin O’Flynn
@colinoflynn
oflynn.com
newae.com
coflynn@newae.com

Also, read the W.P. for details