>>> Insteon, Inste-off, Inste-open?

Caleb Mays & Ben Ramsey

MERCULITE SECURITY

>>> whoami

* Caleb Mays
  - Researcher, Merculite Security
  - BS in Comp Sci
  - Tech hobbyist, Soccer enthusiast
  - Prior work: Network Admin & Project Manager
  - Currently focused on Insteon network security

* Ben Ramsey
  - Research Director, Merculite Security
  - PhD in Comp Sci
  - Wireless geek
  - Recent work: Z-Wave attacks
    -DerbyCon 2015
    -ShmooCon 2016
    -PoC||GTFO 0x12

>>> Overview

1. Goals
2. Review: What is Insteon?
3. Previous reverse engineering & security tools
4. How I made them better
5. Demo
6. Questions

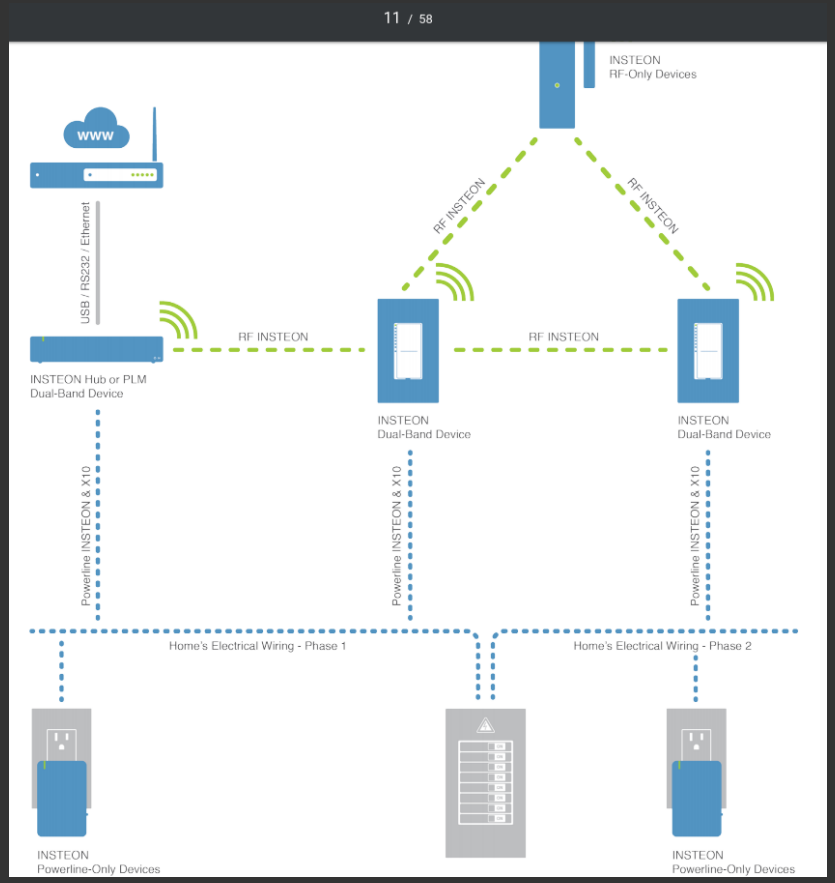
>>> Goals

* Validate and improve any previous work
* Create Wireshark capability
* Create Insteon network scanner / enumerator

"If you don't set goals,
you can't regret not reaching them."
-Yogi Berra

>>> What is Insteon?

* Home automation devices
* Sensors, automation, lights, sprinklers, garage doors, etc.
* Mesh network via RF and power-line protocol
* Can connect from anywhere (even your Apple watch!)
* Integrates with many 3rd party devices (i.e. Amazon Echo).



INSTEON®

>>> What is Insteon?

* Publicized protocol information!(??)

RF Specification	Value
Center Frequency	915 MHz
Data Encoding Method	Manchester
Modulation Method	FSK
FSK Deviation	64 KHz
FSK Symbol Rate	76,800 symbols per second
Data Rate	38,400 bits per second

INSTEON Standard Message – 10 Bytes				
3 Bytes	3 Bytes	1 Byte	2 Bytes	1 Byte
From Address	To Address	Flags	Command 1, 2	CRC[3]

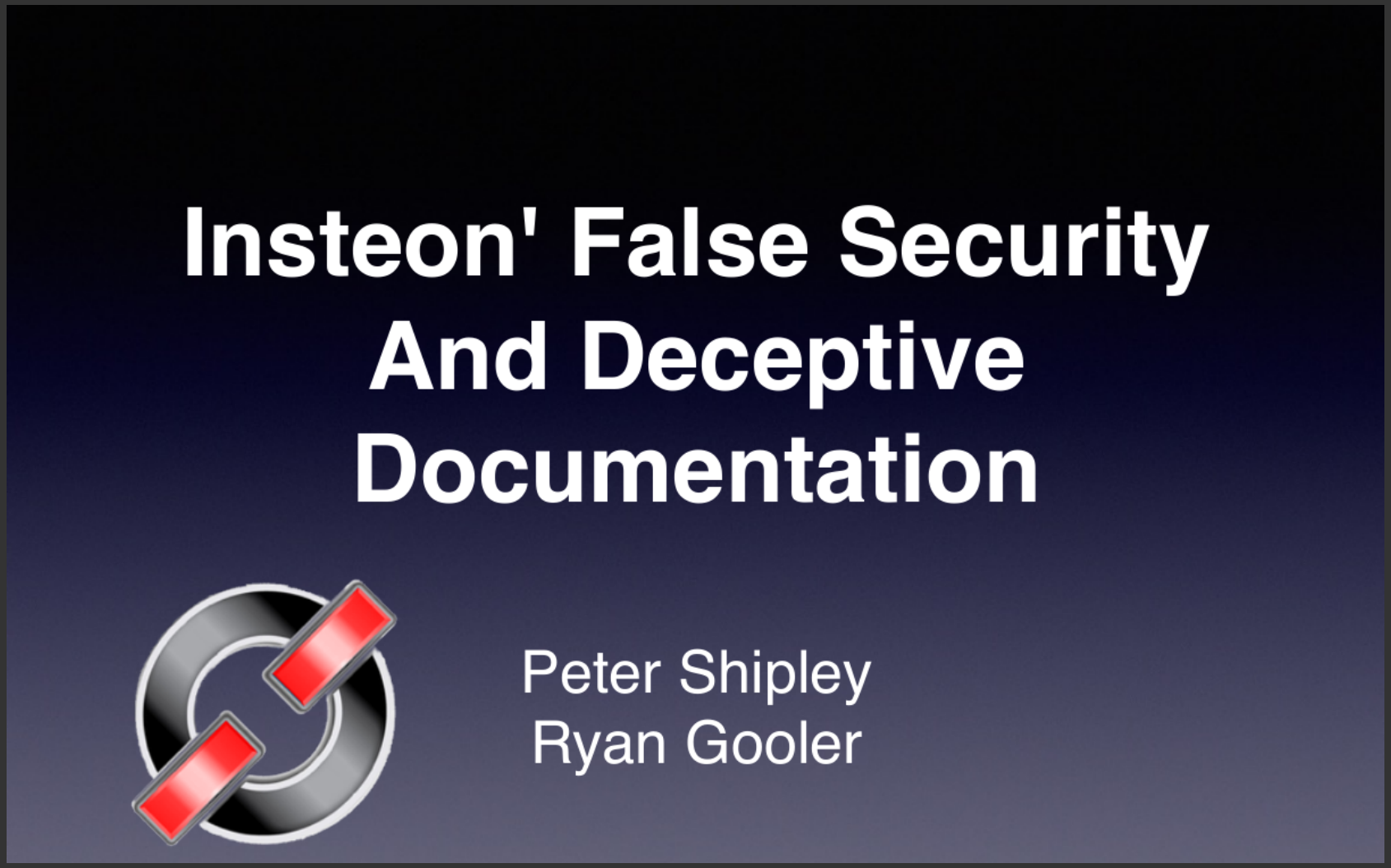
INSTEON Extended Message – 24 Bytes					
3 Bytes	3 Bytes	1 Byte	2 Bytes	14 Bytes	1 Byte
From Address	To Address	Flags	Command 1, 2	User Data	CRC[3]

>>> Had anything been done already?

* Protocol seemed easy enough to reverse
* Had anybody already created any tools?
* YES! Peter Shipley at DEFCON 23.

>>> Previous Reverse Engineering

* Peter Shipley presented Insteon reverse engineering and basic listener and sender at DEFCON 23



>>> Previous Reverse Engineering

* From DEFCON 23...(Peter Shipley)

Published RF Specification
(Layer 2)

Center Frequency	915Mhz	Bullshit
Encoding Method	Manchester	Bullshit
Modulation	FSK	TRUE!!
Deviation	64,000 Hz	Bullshit
Symbol Rate	76,800 sym/s	Bullshit
Data Rate	38,400 bits/s	Bullshit

*** 83% BULLSHIT ***

* WHITEPAPER: The Details

>>> Previous Reverse Engineering

* From DEFCON 23...(Peter Shipley)

Actual RF Specification (Layer 2)

Center Frequency	~~915Mhz~~	914.975 Mhz
Encoding Method	~~Manchester~~	"Tokenized" Manchester
Modulation	FSK	FSK (inverted)
~~Deviation~~ (Shfit)	~~64,000 Hz~~	150,000 Hz
Symbol Rate	~~76,800 sym/s~~	9125 sym/s
Data Rate	~~38,400 bits/s~~	2600 bit/s

>>> Previous Reverse Engineering

* From DEFCON 23...(Peter Shipley)

Actual Packet Order

X	X	X	X	X	X	X	X	X	X
Flag	To Addr			From Addr			cmd	opt	crc

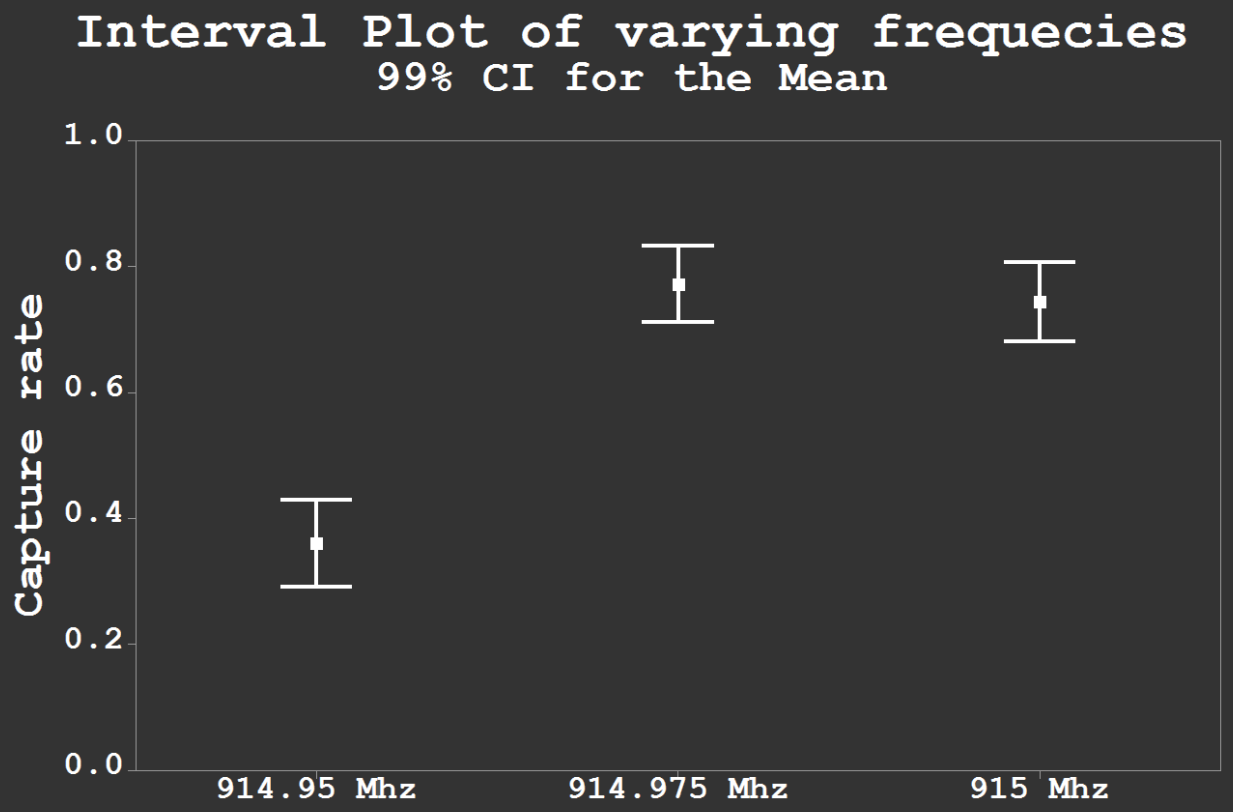
Each byte (X) is encoded as 26 bits:

"11" followed by

+ 5 bit index number (manchester encoded, LSB)
+ 8 bit byte (manchester encoded, LSB)

>>> Making the listener better

* Shipley got it mostly right, however...
* Frequency testing proved freq=915 Mhz
* Shipley claimed freq=914.975 Mhz
  (but had freq=914.95 Mhz in code)
* Using YardStick One, no noticeable difference.



>>> A (minor) correction to packet structure

* Shipley got it mostly right, however...
* Packet structure changes!

Insteon advertised packet structure					
3 Bytes	3 Bytes	1 Byte	2 Bytes	14 Bytes	1 Bytes
Source	Destination	Flags	Command 1, 2	Data (optional)	CRC

>>> A (minor) correction to packet structure

* Shipley got it mostly right, however...
* Packet structure changes!

Insteon advertised packet structure					
3 Bytes	3 Bytes	1 Byte	2 Bytes	14 Bytes	1 Bytes
Source	Destination	Flags	Command 1, 2	Data (optional)	CRC

Peter Shipley's claimed packet structure					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Destination	Source	Command 1, 2	Data (optional)	CRC

>>> A (minor) correction to packet structure

* Shipley got it mostly right, however...
* Packet structure changes!

Insteon advertised packet structure					
3 Bytes	3 Bytes	1 Byte	2 Bytes	14 Bytes	1 Bytes
Source	Destination	Flags	Command 1, 2	Data (optional)	CRC

Peter Shipley's claimed packet structure					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Destination	Source	Command 1, 2	Data (optional)	CRC

Observed packet structure (most commands)					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Destination	Source	Command 1, 2	Data (optional)	CRC

>>> A (minor) correction to packet structure

* Shipley got it mostly right, however...
* Packet structure changes!

Insteon advertised packet structure					
3 Bytes	3 Bytes	1 Byte	2 Bytes	14 Bytes	1 Bytes
Source	Destination	Flags	Command 1, 2	Data (optional)	CRC

Peter Shipley's claimed packet structure					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Destination	Source	Command 1, 2	Data (optional)	CRC

Observed packet structure (most commands)					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Destination	Source	Command 1, 2	Data (optional)	CRC

Observed packet structure (broadcast, group broadcast commands)					
1 Byte	3 Bytes	3 Bytes	2 Bytes	14 Bytes	1 Bytes
Flags	Source	Source Info or Group Address	Command 1, 2	Data (optional)	CRC

>>> Quick Recap

* Goal #1: Validate & improve previous work

>>> Quick Recap

* Goal #1: Validate & improve previous work
    - Fixed frequency
    - Corrected some packet structure details

>>> Quick Recap

* ~~Goal #1: Validate & improve previous work~~
  - ~~Fixed frequency~~
  - ~~Corrected some packet structure details~~
* How can I make this better?

>>> Quick Recap

* ~~Goal #1: Validate & improve previous work~~
  - ~~Fixed frequency~~
  - ~~Corrected some packet structure details~~
* How can I make this better?
  - Goal #2: Output to Wireshark
  - Goal #3: Network enumerator

>>> Insteon .pcap-ization

* Wireshark is the gold standard for network traffic analysis.
* Modify /etc/wireshark/init.lua
* Create /usr/share/wireshark/Insteon-dissector.lua

init.lua

```
652
653 dofile(DATA_DIR.."console.lua")
654 dofile(DATA_DIR.."dtd_gen.lua")
655 dofile(DATA_DIR.."Insteon-dissector.lua") -- new line here.
```

About Wireshark

Wireshark | Authors | Folders | Plugins | Keyboard Shortcuts | License

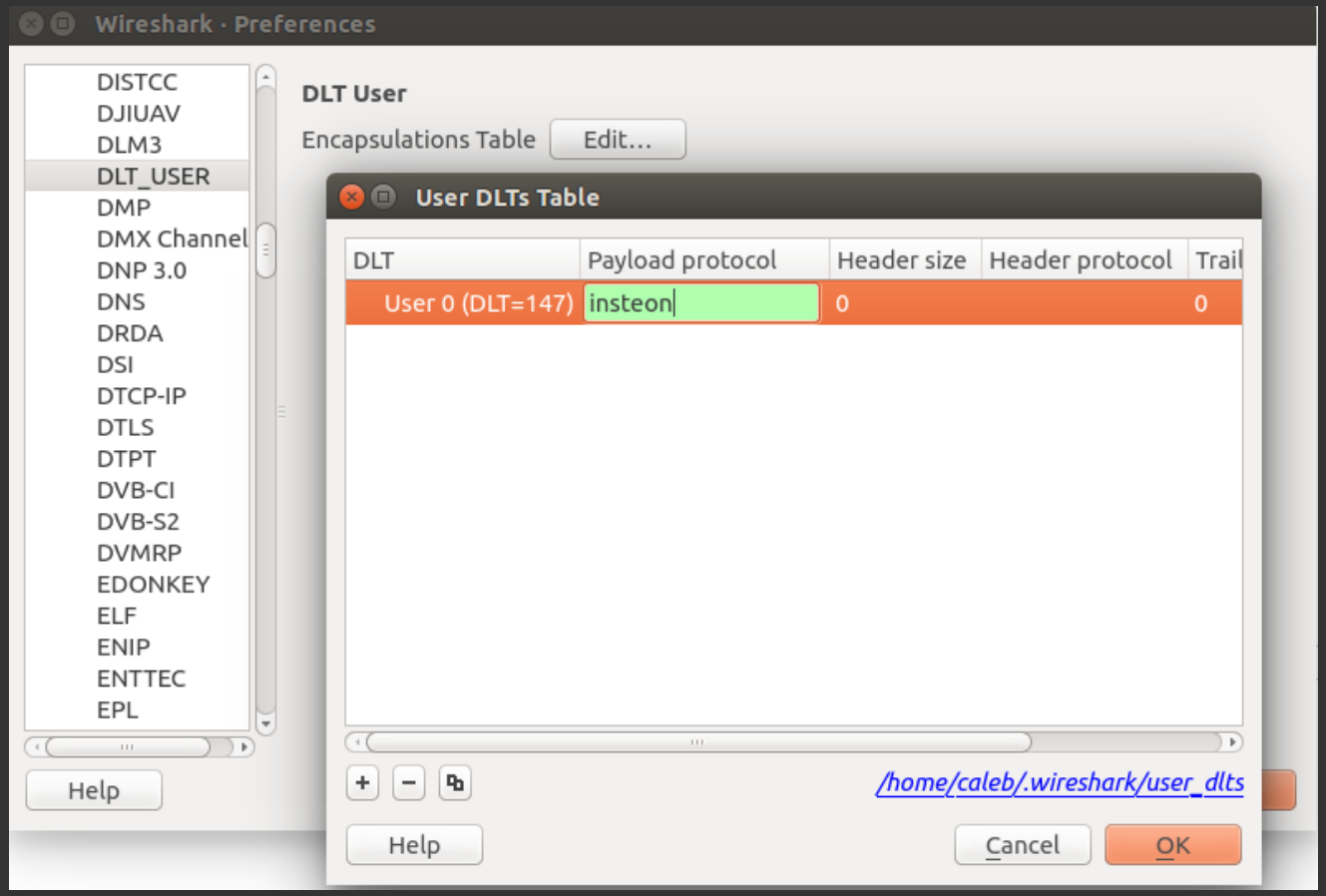
Name	Location
"File" dialogs	/home/caleb/Insteon/myinsteon/
Temp	/tmp
Personal configuration	/home/caleb/.wireshark/
Global configuration	/usr/share/wireshark
System	/etc
Program	/usr/bin
Personal Plugins	/home/caleb/.wireshark/plugins
Global Plugins	/usr/lib/i386-linux-g...reshark/plugins/2.0.4
GeoIP path	/usr/share/GeoIP
Extcap path	/usr/lib/i386-linux-gnu/wireshark/extcap/

steon-dissctor.lua

```
3  ----------------------------------------
4  -- creates a Proto object, but doesn't register it yet
5  local insteon = Proto("Insteon","Insteon")
6
7  -- start declaring some fields.
8  -- first, the flags.
9  local pf_flags              = ProtoField.new   ("Flags", "insteon.fl
10
11 local pf_flag_broadcast     = ProtoField.new   ("Broadcast/NAK", "in
12     {"this is a broadcast","this is not a broadcast"}, 8, 0x80, "is
13
14 local pf_flag_group         = ProtoField.new   ("Group/Not-Group", "
15     {"this is a group message", "this is not a group message"}, 8, 0
16
17 local pf_flag_acknowledge   = ProtoField.new   ("Acknowledge", "inst
18     {"acknowledgement", "not an acknowledgement"}, 8, 0x20, "is this
```

>>> Insteon .pcap-ization

* Edit->Preferences



>>> Insteon .pcap-ization

* Convert packets to .pcap data format
* Can write data to a file or a named pipe
* Live capture using wireshark -i <named pipe>

No.	Time	Destination	Source	Protocol	Length	Info
21	9.000000	dec233	8cde36	Insteon	10	Direct Message
22	9.000000	dec233	8cde36	Insteon	10	Direct Message
23	9.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message
24	9.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message
25	9.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message

```
▸ Frame 21: 10 bytes on wire (80 bits), 10 bytes captured (80 bits)
  DLT: 147, Payload: insteon (Insteon)
▾ Insteon
  ▾ Flags: 0x07
      0... .... = Broadcast/NAK: this is not a broadcast
      .0.. .... = Group/Not-Group: this is not a group message
      ..0. .... = Acknowledge: not an acknowledgement
      ...0 .... = Message Type: Standard Message
      .... 01.. = Hops Left: 1
      .... ..11 = Max Hops: 3
    Destination: dec233
    Source: 8cde36
    Command 1: 0f
    Command 2: 00
    CRC: 53
```

* Credit to "RPGillespie" here.

>>> Another Quick Recap

* ~~Goal #1: Validate & improve previous work~~
  - ~~Fixed frequency~~
  - ~~Corrected some packet structure details~~
* How can I make this better?
  - Goal #2: Output to Wireshark
  - Goal #3: Network enumerator

>>> Another Quick Recap

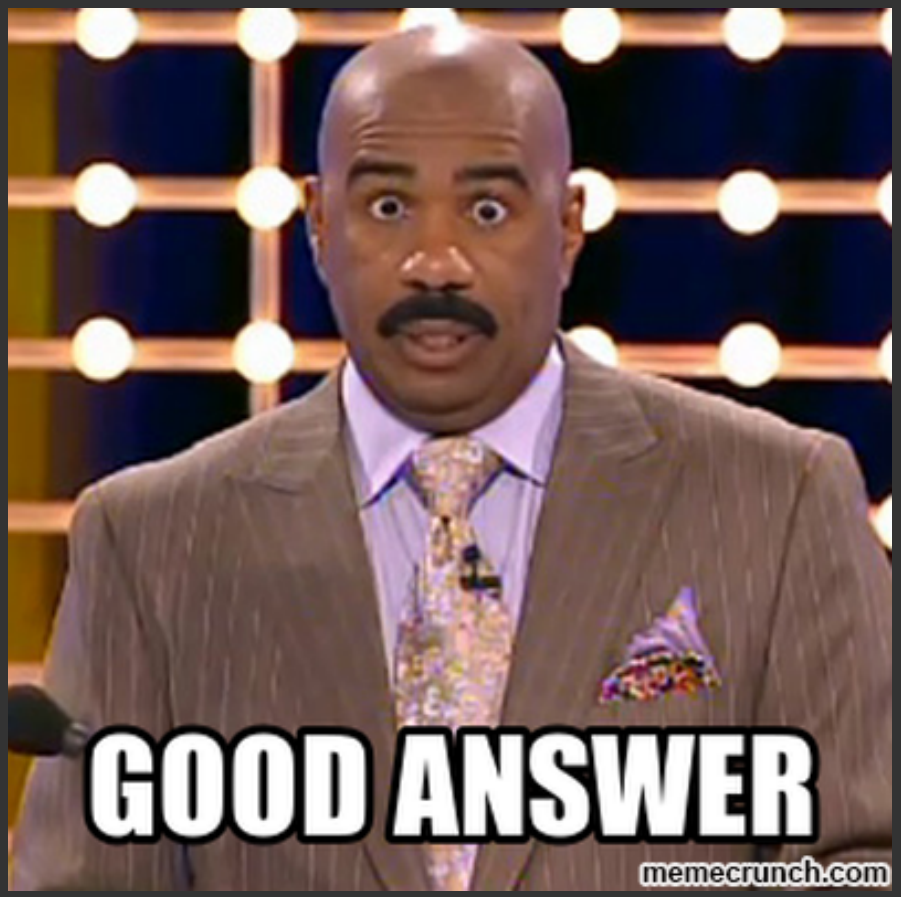
* ~~Goal #1: Validate & improve previous work~~
  - ~~Fixed frequency~~
  - ~~Corrected some packet structure details~~
* How can I make this better?
  - ~~Goal #2: Output to Wireshark~~
  - Goal #3: Network enumerator

>>> Towards an Insteon Network Enumerator

* Question:
  Can I scan for & enumerate Insteon devices?

>>> Towards an Insteon Network Enumerator

* Question:
  Can I scan for & enumerate Insteon devices?
* Answer:
  Yes, but...I need device IDs...(so far)



>>> Towards an Insteon Network Enumerator

* Need commands that would be helpful to ID devices

>>> Towards an Insteon Network Enumerator

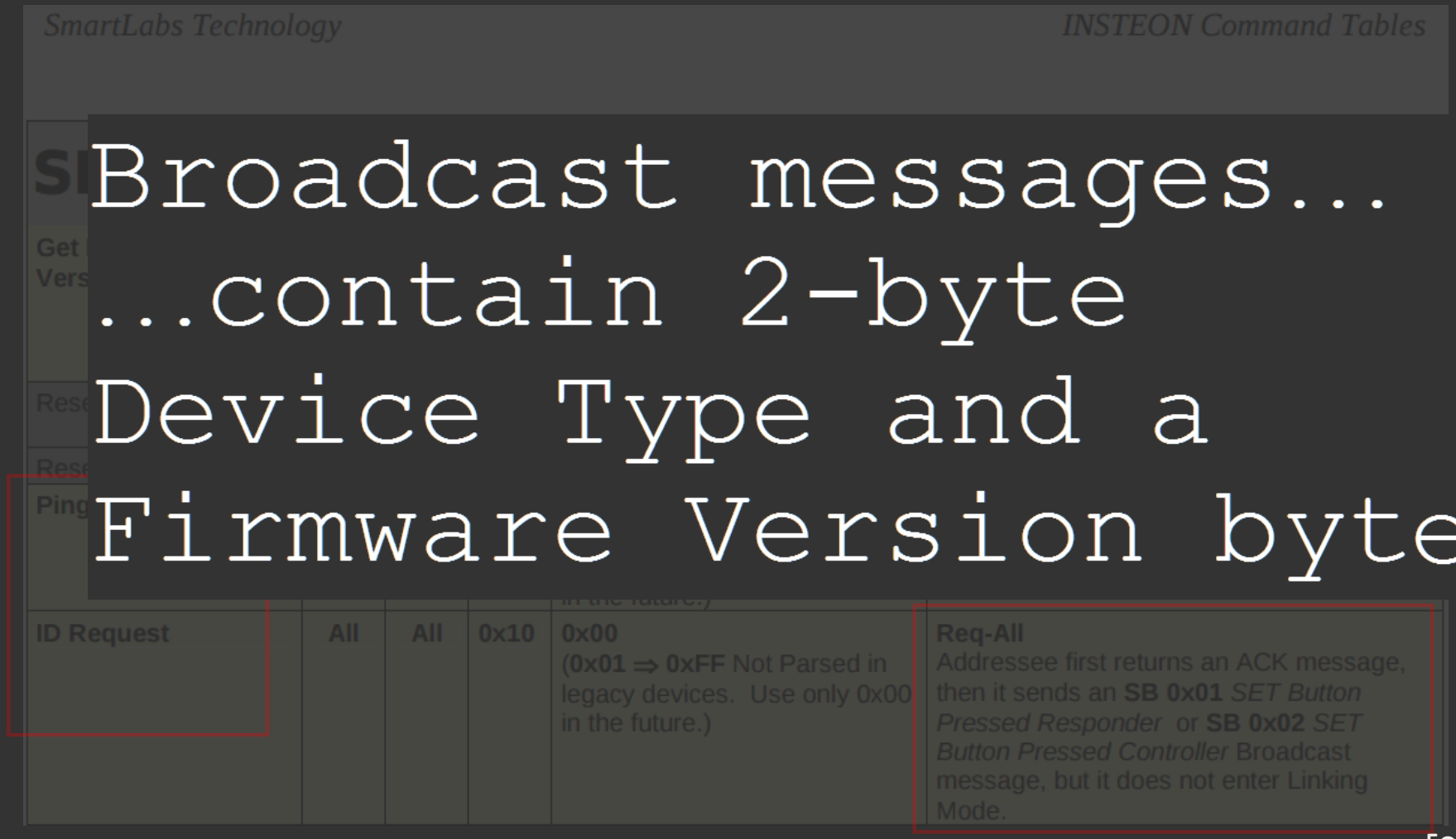
* Need commands that would be helpful to ID devices
* Found (circa 2007) command list: ping and ID request commands

SmartLabs Technology *INSTEON Command Tables*

SD Commands	**Dev Cat**	**Sub Cat**	**Cmd 1**	**Cmd 2**	**Note Keys, Description**
Get INSTEON Engine Version	**All**	**All**	**0x0D**	**0x00**	**Req-All** Returned ACK message will contain the INSTEON Engine Version in Command 2. **0x00** = i1 (default echo for legacy devices) **0x01** = i2
Reserved			**0x0D**	**0x01 ⇒ 0xFF**	Do not use so that legacy devices will echo **0x00** in Command 2
Reserved			**0x0E**		
Ping	**All**	**All**	**0x0F**	**0x00** (**0x01 ⇒ 0xFF** Not Parsed in legacy devices. Use only 0x00 in the future.)	**Req-All** Addressee returns an ACK message but performs no operation.
ID Request	**All**	**All**	**0x10**	**0x00** (**0x01 ⇒ 0xFF** Not Parsed in legacy devices. Use only 0x00 in the future.)	**Req-All** Addressee first returns an ACK message, then it sends an **SB 0x01** *SET Button Pressed Responder* or **SB 0x02** *SET Button Pressed Controller* Broadcast message, but it does not enter Linking Mode.

>>> Towards an Insteon Network Enumerator

* Need commands that would be helpful to ID devices
* Found (circa 2007) command list: ping and ID request commands



SmartLabs Technology *INSTEON Command Tables*

ID Request	All	All	0x10	**0x00** (**0x01** ⇒ **0xFF** Not Parsed in legacy devices. Use only 0x00 in the future.)	**Req-All** Addressee first returns an ACK message, then it sends an **SB 0x01** *SET Button Pressed Responder* or **SB 0x02** *SET Button Pressed Controller* Broadcast message, but it does not enter Linking Mode.

Broadcast messages...
...contain 2-byte
Device Type and a
Firmware Version byte

>>> Towards an Insteon Network Enumerator

* Ping and ID request are still valid commands!

No.	Time	Destination	Source	Protocol	Length	Info
202	56.000000	dec233	8cde36	Insteon	10	Direct Message
203	56.000000	dec233	8cde36	Insteon	10	Direct Message
204	56.000000	dec233	8cde36	Insteon	10	Direct Message
205	56.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message
206	56.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message
207	56.000000	8cde36	dec233	Insteon	10	Acknowledgement of Direct Message
208	56.000000	Firmware & Dev Type	dec233	Insteon	10	Broadcast Message
209	56.000000	Firmware & Dev Type	dec233	Insteon	10	Broadcast Message
210	56.000000	Firmware & Dev Type	dec233	Insteon	10	Broadcast Message

```
▶ Frame 208: 10 bytes on wire (80 bits), 10 bytes captured (80 bits) on interface 0
  DLT: 147, Payload: insteon (Insteon)
▼ Insteon
  ▶ Flags: 0x8f
    Source: dec233
    Firmware: 9e
    Device Subtype: 33
    Device Type: 03
    Command 1: 01
    Command 2: 00
    CRC: 0f
```

Source: dec233

Firmware: 9e

Device Subtype: 33

Device Type: 03

>>> Towards an Insteon Network Enumerator

* Oh by the way...here are some device categories (circa 2008)

Dev Cat #	Device Category Name	Examples of Devices
0x00	Generalized Controllers	ControLinc, RemoteLinc, SignaLinc, etc.
0x01	Dimmable Lighting Control	Dimmable Light Switches, Dimmable Plug-In Modules
0x02	Switched Lighting Control	Relay Switches, Relay Plug-In Modules
0x03	Network Bridges	PowerLinc Controllers, TRex, Lonworks, ZigBee, etc.
0x04	Irrigation Control	Irrigation Management, Sprinkler Controllers
0x05	Climate Control	Heating, Air conditioning, Exhausts Fans, Ceiling Fans, Indoor Air Quality
0x06	Pool and Spa Control	Pumps, Heaters, Chemicals
0x07	Sensors and Actuators	Sensors, Contact Closures
0x08	Home Entertainment	Audio/Video Equipment
0x09	Energy Management	Electricity, Water, Gas Consumption, Leak Monitors
0x0A	Built-In Appliance Control	White Goods, Brown Goods
0x0B	Plumbing	Faucets, Showers, Toilets
0x0C	Communication	Telephone System Controls, Intercoms
0x0D	Computer Control	PC On/Off, UPS Control, App Activation, Remote Mouse, Keyboards
0x0E	Window Coverings	Drapes, Blinds, Awnings
0x0F	Access Control	Automatic Doors, Gates, Windows, Locks
0x10	Security, Health, Safety	Door and Window Sensors, Motion Sensors, Scales
0x11	Surveillance	Video Camera Control, Time-lapse Recorders, Security System Links
0x12	Automotive	Remote Starters, Car Alarms, Car Door Locks

>>> Towards an Insteon Network Enumerator

* Overall Method:
  - Collect IDs
  - Spoof ping and ID requests between devices
  - Generate "map" by tracking & labeling devices

```
8C DE 36 is a:  Security, Health, Safety
DE C2 33 is a:  Network Bridge
33 D3 32 is a:  Dimmable Lighting Control
C2 EA 31 is a:  Security, Health, Safety
FD C9 36 is a:  Security, Health, Safety


Controllers are:
8C DE 36  controls:  ['DE C2 33', '33 D3 32']
DE C2 33  controls:  ['33 D3 32']
33 D3 32  controls:  ['DE C2 33']
C2 EA 31  controls:  ['DE C2 33']
FD C9 36  controls:  ['DE C2 33']


Responders are:
DE C2 33  responds to:  ['8C DE 36', '33 D3 32', 'C2 EA 31', 'FD C9 36']
33 D3 32  responds to:  ['8C DE 36', 'DE C2 33']


Group controllers are:
8C DE 36  controls group(s):  ['group 01', 'group 04']
C2 EA 31  controls group(s):  ['group 01']
FD C9 36  controls group(s):  ['group 01']


Group responders are:
DE C2 33  responds to group(s):  ['group 01', 'group 04']
33 D3 32  responds to group(s):  ['group 01']
```

>>> In summary

* Goal #1: validate & improve previous work
  - Fixed frequency
  - Corrected some packet structure details
* I made it better by:
  - Goal #2: Output to Wireshark
  - Goal #3: Network enumerator

>>> Copyright / Credits

* Peter Shipley - https://github.com/evilpete/insteonrf
* Insteon Whitepaper: The Details - http://cache.insteon.com/documentation/insteon_details.pdf
* Insteon: Command Tables - http://cache.insteon.com/pdf/INSTEON_Command_Tables_20070925a.pdf
* Insteon: Device Categories and Product Keys - http://cache.insteon.com/pdf/INSTEON_DevCats_and_Product_Keys_20081008.pdf
* INSTEON is a trademark of INSTEON

  16542 Millikan Ave., Irvine, CA 92606-5027
  866-243-8022, www.insteon.com

Wireless Demo

>>> Questions?

Code: github.com/merculite/Insteon-Tools

Contact us: team @ merculite.net

>>> Hardware used

* Yet Another Radio Dongle [YARD] Stick One
  - Created by: Great Scott Gadgets
  - Half duplex transmit and receive
  - Freqs: 300-348 MHz, 391-464 MHz, and 782-928 MHz
  - Modulations: ASK, OOK, GFSK, 2-FSK, 4-FSK, MSK
  - Website: https://greatscottgadgets.com/yardstickone/