Dangerous Pyrotechnic 'Composition': Fireworks, Embedded Wireless and Insecurity-by-Design

Andrei Costin, Aurélien Francillon

EURECOM
Sophia Antipolis

DefCamp 2014

guest@con:~/$ whoami

Andrei Costin DefCamp'14

guest@con:~/$ whoami

Andrei Costin DefCamp'14

guest@con:~/$ whoami

guest@con:~/$ whoami

guest@con:~/$ whoami

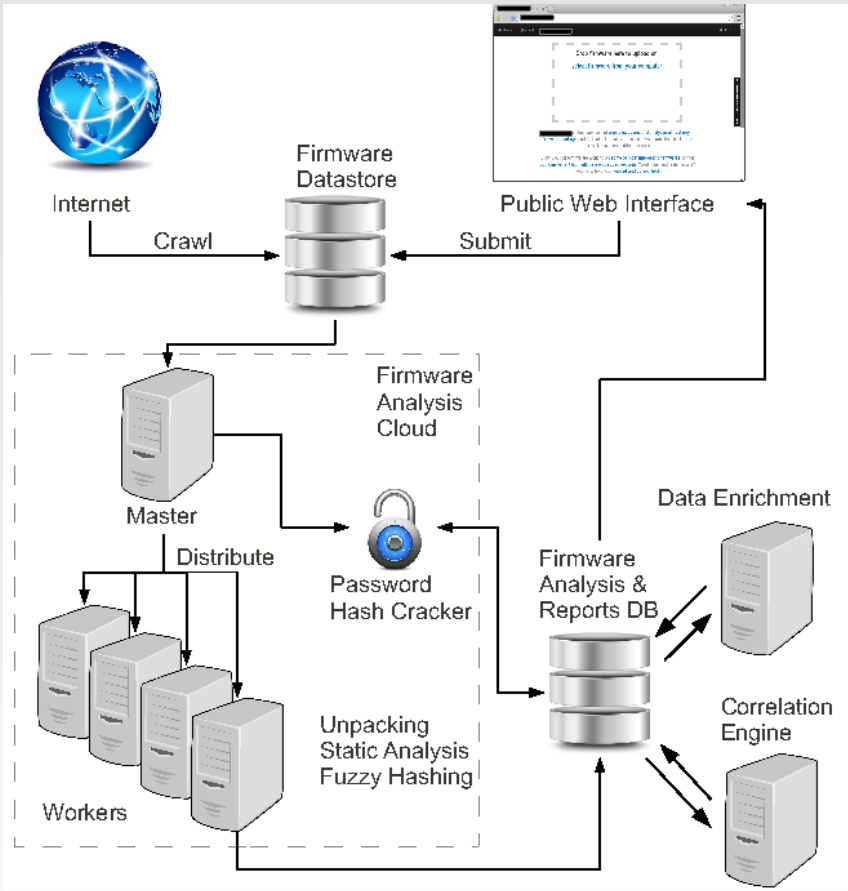
guest@con:~/$ whoami

Disclaimer

- Educational/research purposes only
- No liability whatsoever on the speaker, affiliates
- WARN: High accident risk and life hazard
- DO NOT replicate unless qualified & equipped

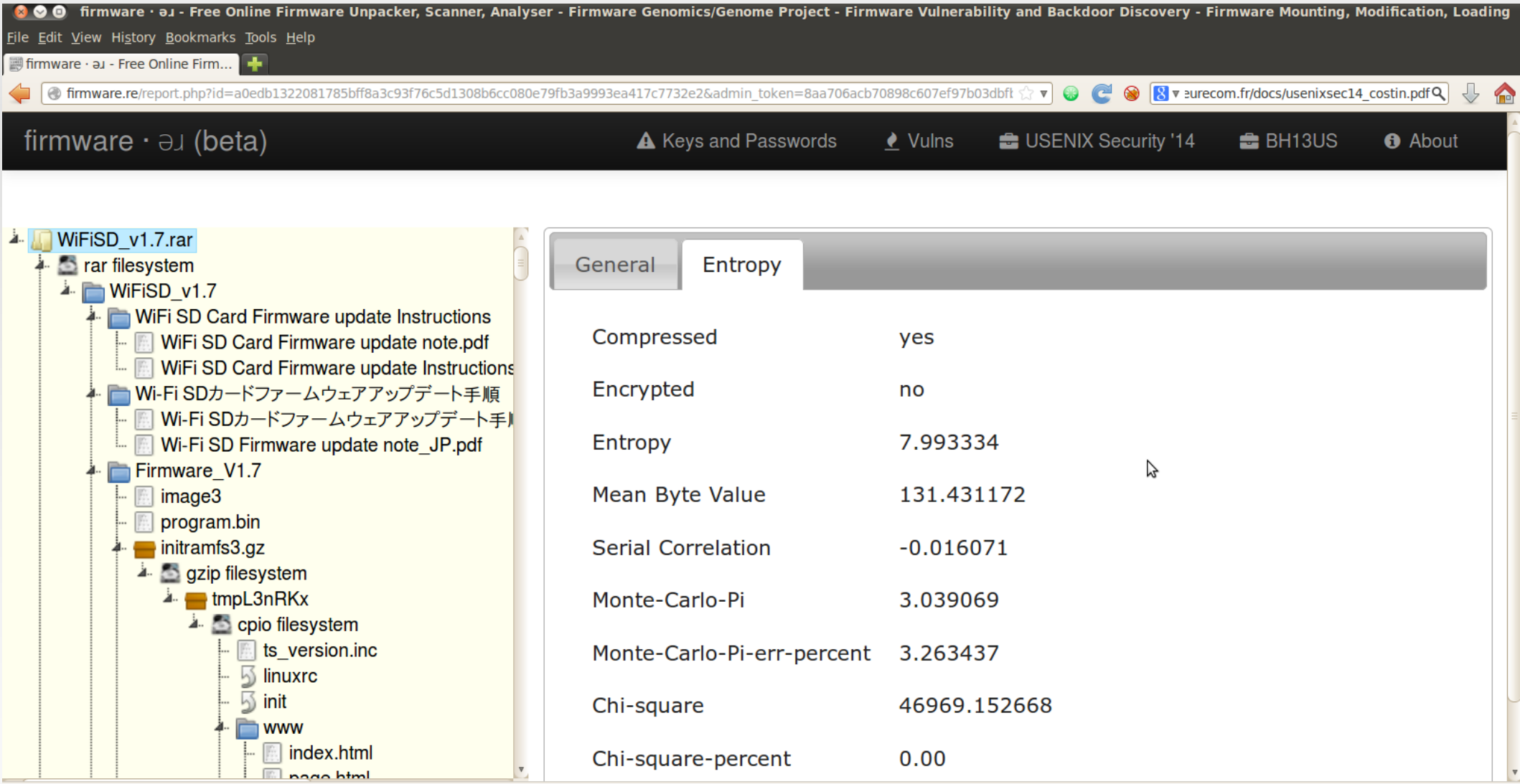
Firmware Analysis

- Large-scale analysis of embedded firmwares [1]
  - crawled 172K firmwares
  - analyzed 32K firmwares
  - found 38 vulnerabilities
  - in over 693 firmwares
  - 140K online devices

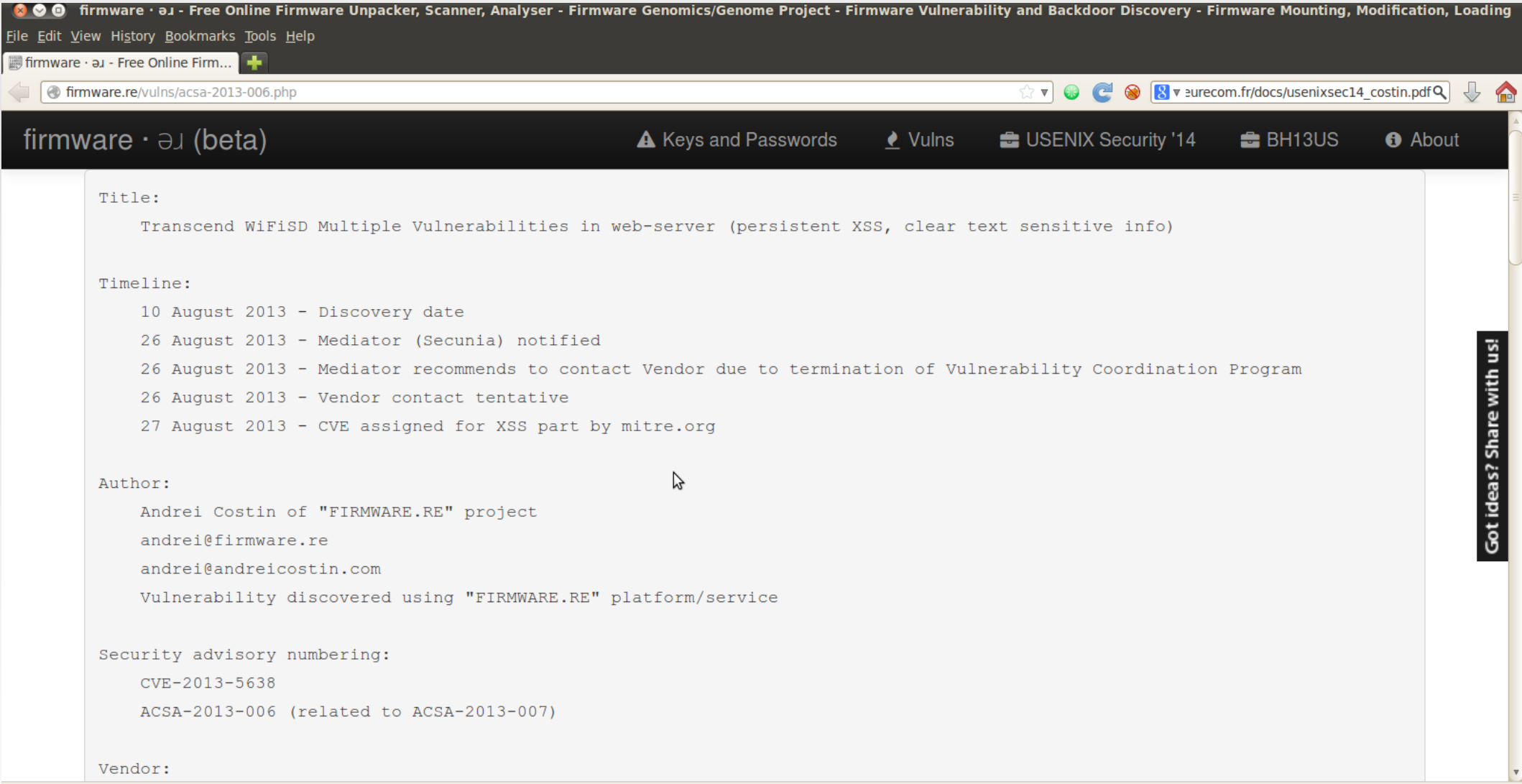


[1] Costin et al., "A Large-Scale Analysis of the Security of Embedded Firmwares", USENIX Sec '14

Firmware Analysis



Firmware Analysis



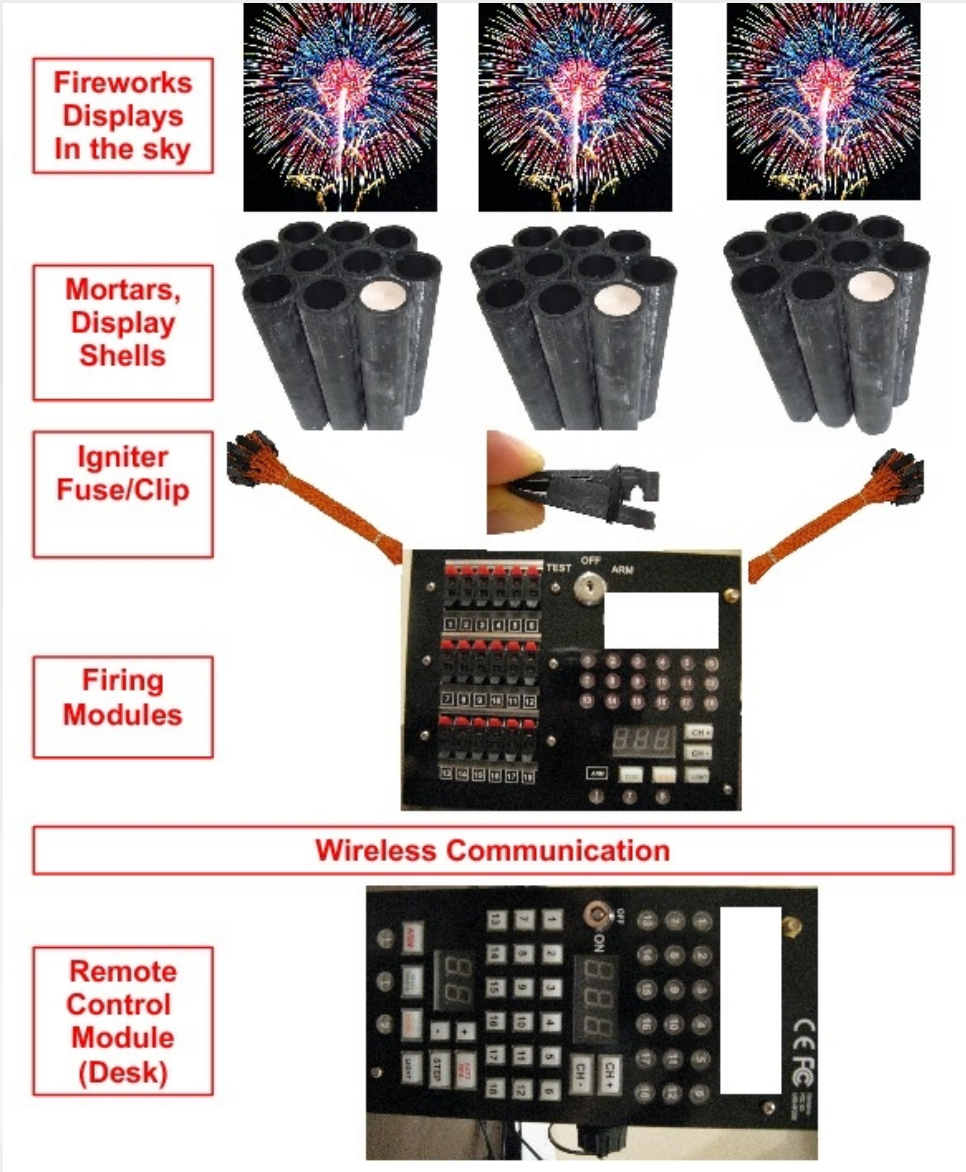
Firmware Analysis

- The firmwares of the firing system:
  - found by our crawlers
  - in *.ihex* format
  - *unencrypted*
- Our framework detected:
  - m68k-based code
  - debugging features (strings)
  - wireless protocols (strings)

Agenda

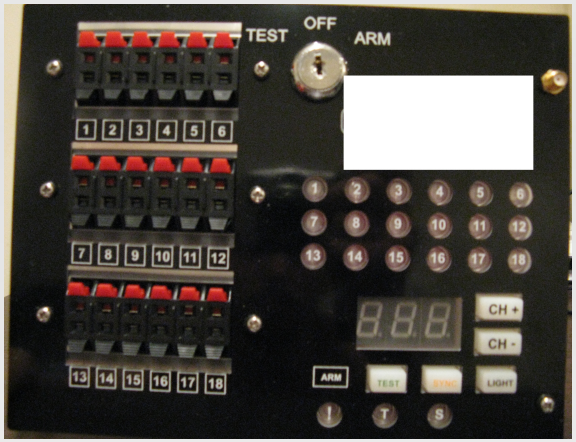
- Pre-Intro
  - It all starts with a good piece of firmware
- Intro
  - What are the *wireless firing systems*?
- Methodology
  - System analysis
  - Attack development
- Results
  - Attacks summary
  - Disclosure process

Wireless Firing Systems



Wireless Firing Systems

- Normal (safe) mode – diagram



1. Connect Firing Module to pyrotechnics and wiring
2. Turn the *physical key* to TEST
3. Perform the continuity test
4. Turn the *physical key* to ARM
5. Firing Module awaits *digital FIRE* command
6. Depart to safety distance

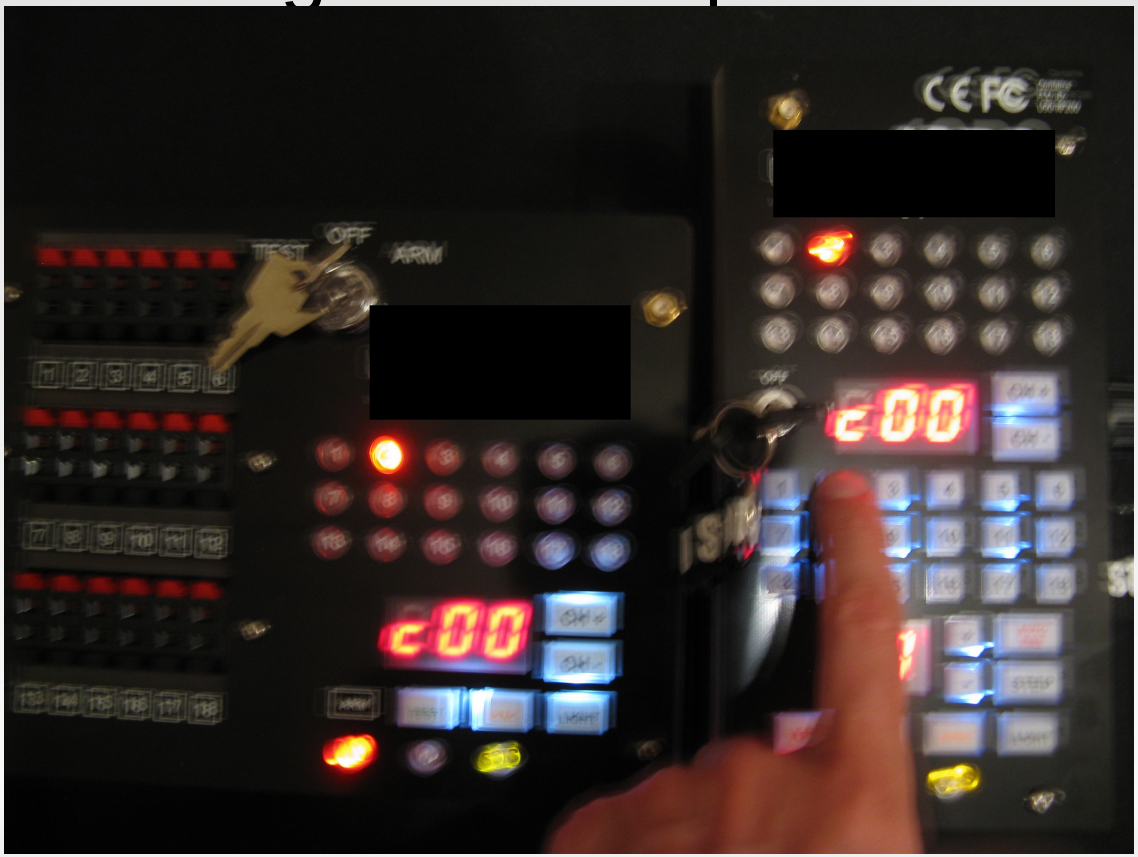
SAFETY DISTANCE BY REGULATION

1. Turn the *physical key* to ARM
2. Press the FIRE keys
3. Remote Control sends *digital FIRE* command

Wireless Firing Systems

- ARM/FIRE operation example

Firing Module | Remote Control



Wireless Firing Systems

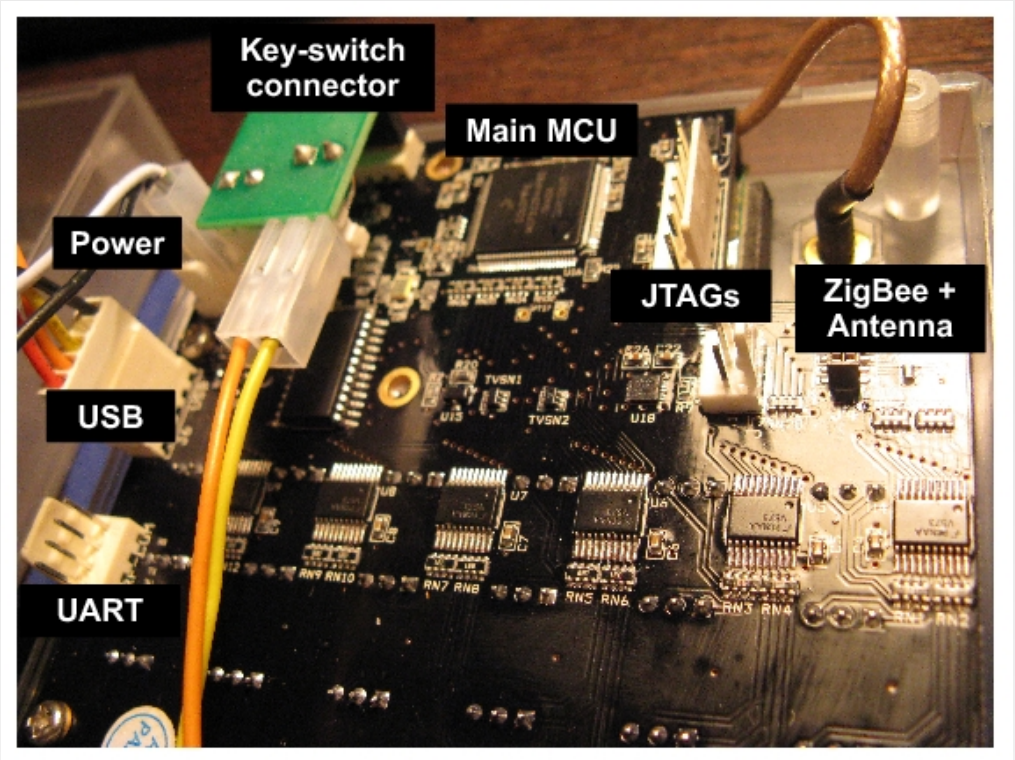
- A very good example of:
  - Wireless Sensors Actuators Network (WSAN)
  - Cyber Physical System (CPS)
- With their properties, challenges and *flaws*

- Used for:
  - Fireworks
  - Building demolition
  - Military-like trainings/simulations

Agenda

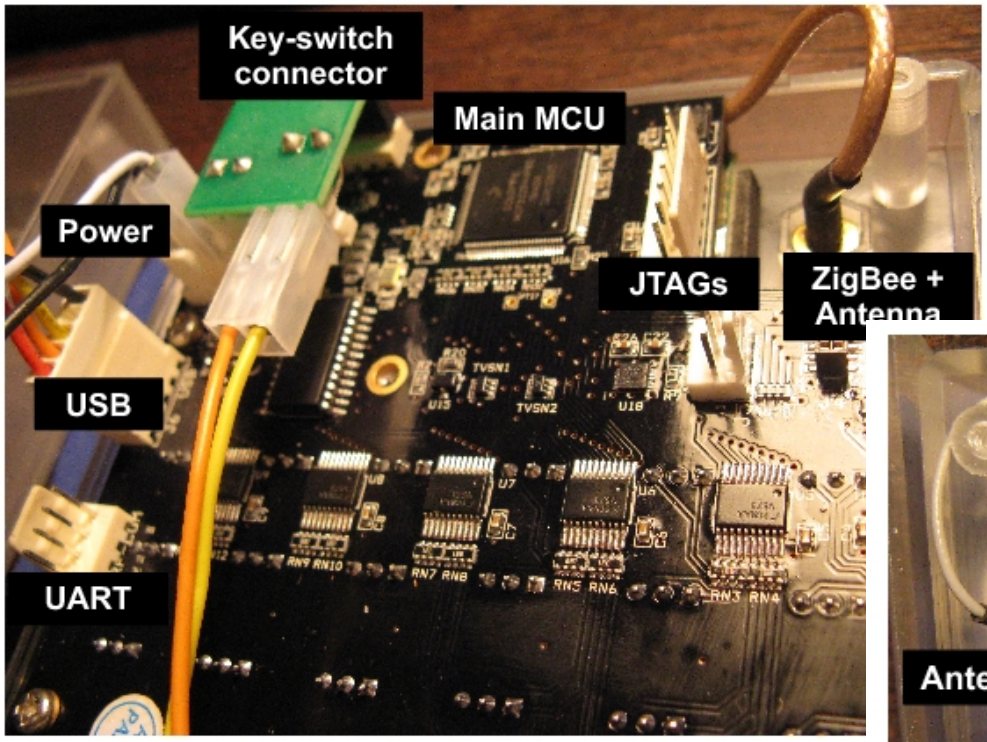
- Pre-Intro
  - It all starts with a good piece of firmware
- Intro
  - What are the *wireless firing systems*?
- Methodology
  - System analysis
  - Attack development
- Results
  - Attacks summary
  - Disclosure process

Methodology – System Analysis



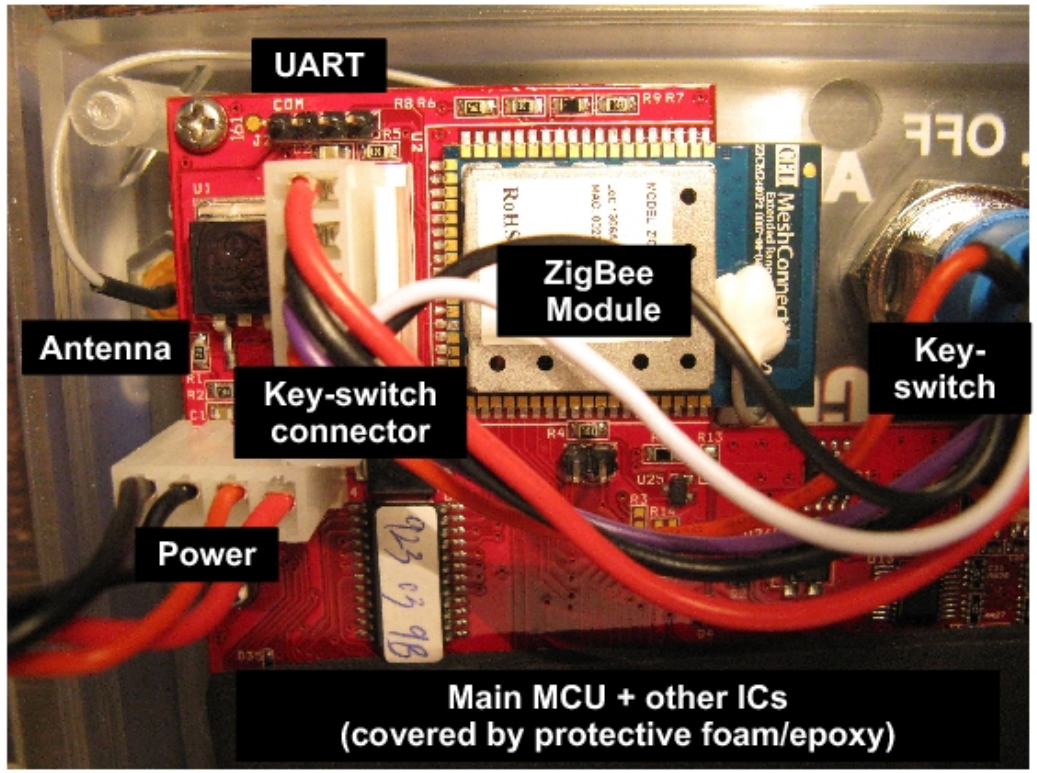
Firing Module

Methodology – System Analysis



Firing Module

Remote Control

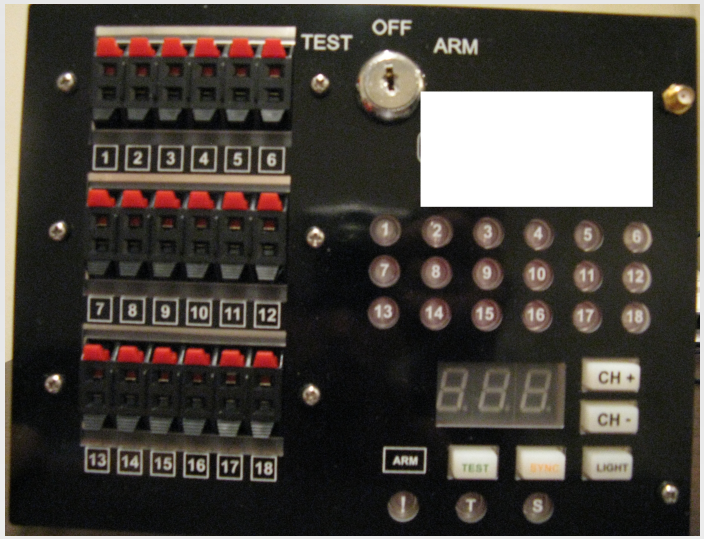


Methodology – System Analysis

- Main MCU running main firmware
  - Freescale ColdFire MCF52254
- 802.15.4 MCUs (*ATmega128RFA1*)
  - Synapse's SNAP Network Operating System
  - API for running Python on the wireless chips
  - AES is supported (802.15.4 standard)
- This system *does not* use AES!!!

Methodology – Attack Explained

- Attacker (unsafe) mode – diagram



1. Connect Firing Module to pyrotechnics and wiring
2. Turn the *physical key* to TEST
3. Perform the continuity test
4. Turn the *physical key* to ARM
5. Firing Module awaits *digital FIRE* command
6. Staff not yet departed

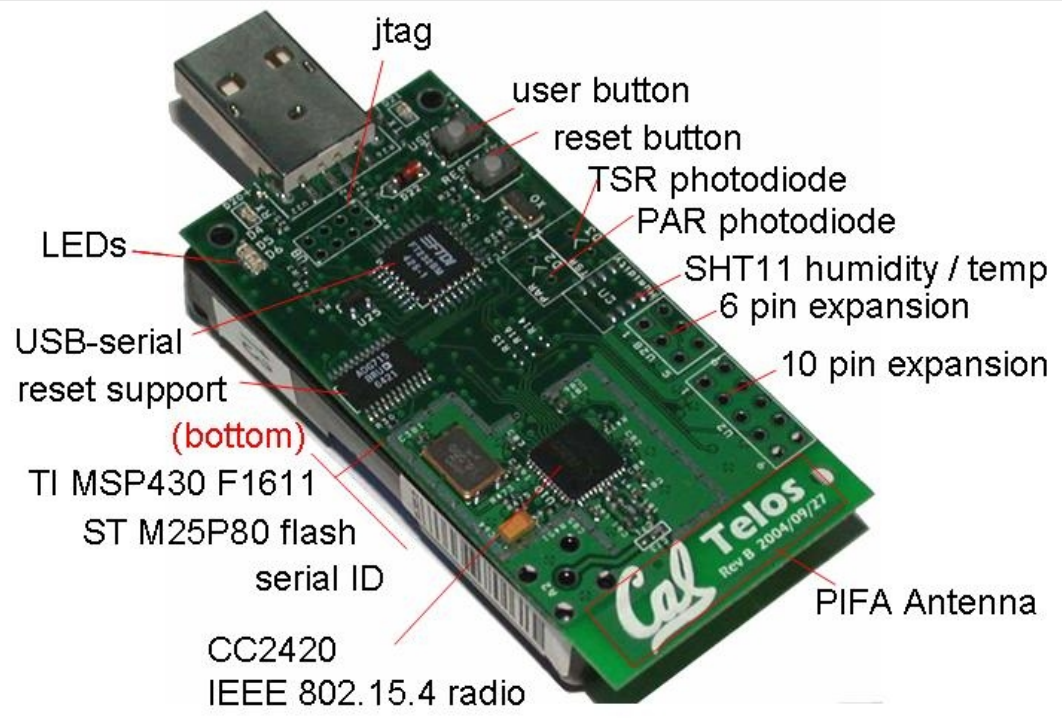
UNSAFE DISTANCE (STAFF NEAR PYROTECHNIC LOADS)

1. {Sniff, replay, inject} loop

1.x Attacker sends *digital FIRE* command

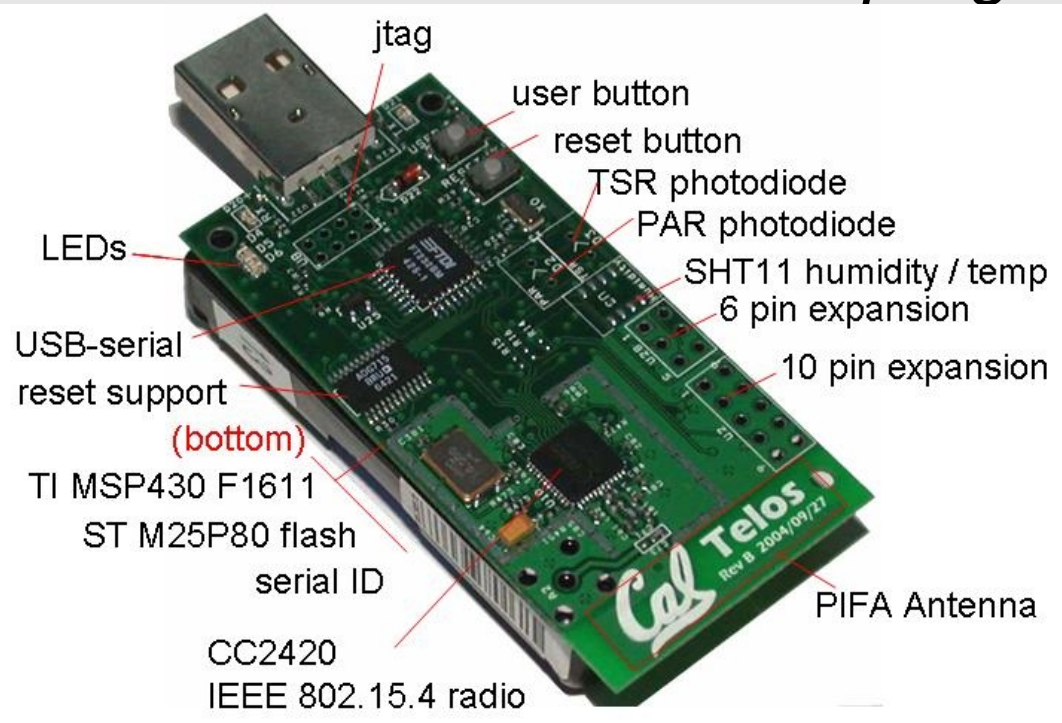
Methodology – Attack Dev

- Sniffers – TelosB and SS200-001
  - TelosB: Default GoodFET / KillerBee firmwares



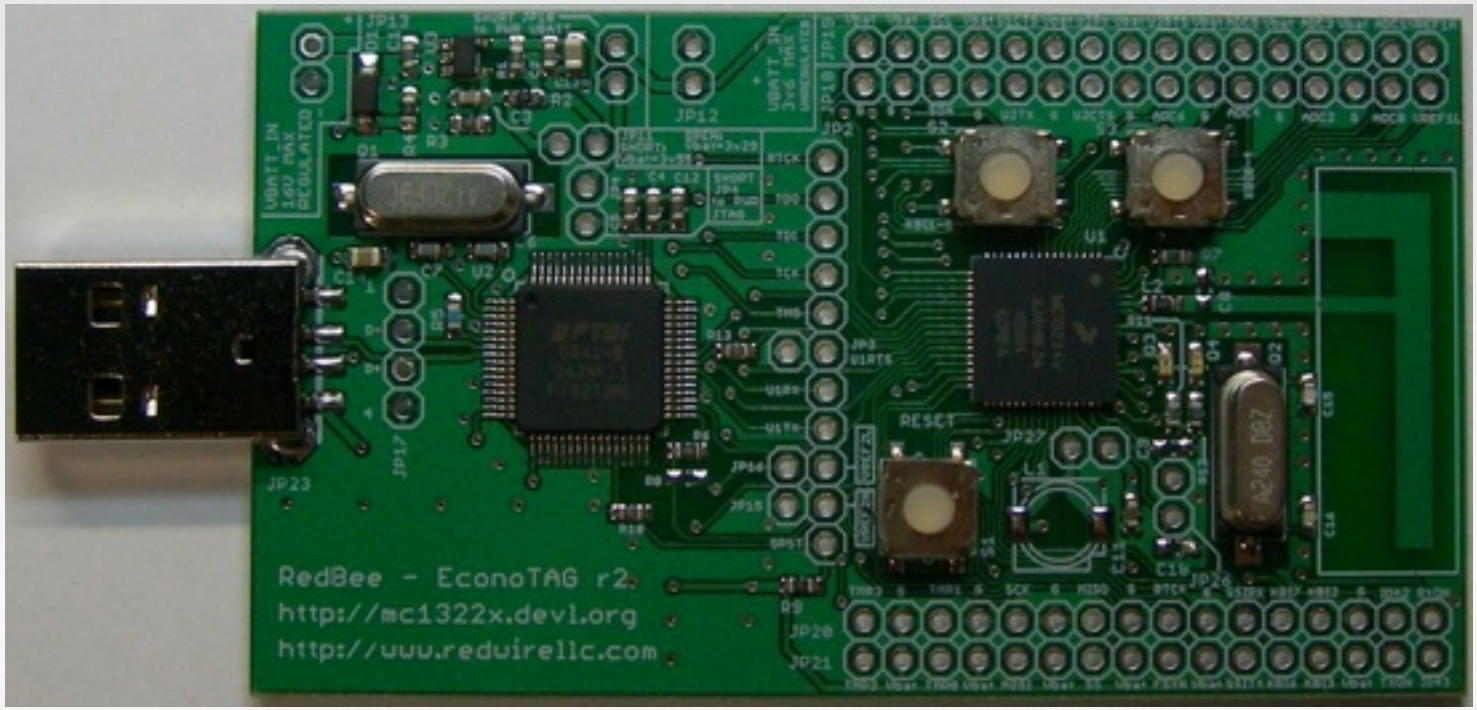
Methodology – Attack Dev

- Sniffers – TelosB and SS200-001
  - TelosB: Default GoodFET / KillerBee firmwares
  - SS200: Wireless *reprogrammer* and *sniffer*



Methodology – Attack Dev

- Injector – Econotag
  - Used as general purpose 802.15.4 device
  - We developed custom replay/inject firmware



Agenda

- Pre-Intro
  - It all starts with a good piece of firmware
- Intro
  - What are the *wireless firing systems*?
- Methodology
  - System analysis
  - Attack development
- Results
  - Attacks summary
  - Disclosure process

Attack Summary

- Sniffing with TelosB the raw packets

```
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export board=telosb
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export platform=telosb
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export mcu=msp430f1611
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export config='monitor ccspi spi'
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ goodfet.ccspi info
ON: /dev/ttyUSB0
Found   CC2420
Freq:   2405.000000 MHz
Status: XOSC16M_STABLE TX_ACTIVE LOCK
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ goodfet.spiflash info
Ident as Numonyx/ST M25P80
Manufacturer: 20 Numonyx/ST
Type: 20
Capacity: 14 (1048576 bytes)
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ goodfet.ccspi sniff 15
ON: /dev/ttyUSB0
Listening as 00deadbeef on 2425 MHz
# 33 37 cd 2e 08 00 01 01 04 4d 8a c1 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c2 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e5 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c3 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 e9
# 33 37 cd 2e 08 00 01 01 04 4d 8a c4 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c5 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ea
# 33 37 cd 2e 08 00 01 01 04 4d 8a c6 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 e9
# 33 37 cd 2e 08 00 01 01 04 4d 8a c7 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ec
# 33 37 cd 2e 08 00 01 01 04 4d 8a c9 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ea
# 33 37 cd 2e 08 00 01 01 04 4d 8a ca 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a cb 04 4d 8a 06 02 00 15 01 00 00 01 00 01 05 00 00 05 00 00 01 00 01 01
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 e9
```

Attack Summary

- Sniffing with the SNAP device/decoder

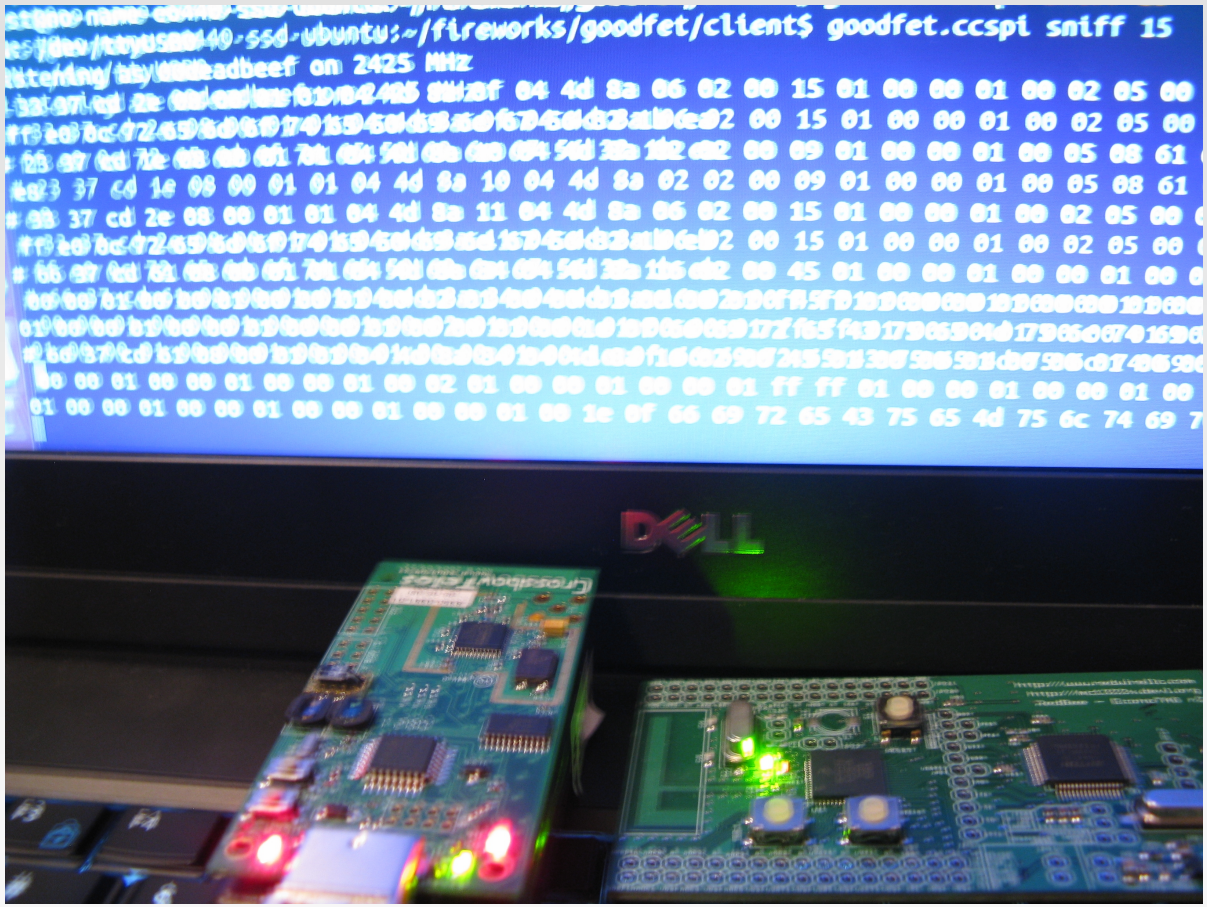
```
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export board=telosb
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export platform=telosb
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export mcu=msp430f1611
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ export config='monitor ccspi spi'
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ goodfet.ccspi info
ON: /dev/ttyUSB0
Found    CC2420
Freq:    2405.000000 MHz
Status: XOSC16M_STABLE TX_ACTIVE LOCK
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client$ goodfet.spiflash info
Ident as Numonyx/ST M25P80
Manufacturer: 20 Numonyx/ST
Type: 20
Capacity: 14 (1048576 bytes)
test@no-name-e6440-ssd-ubuntu:~/fireworks/goodfet/client
ON: /dev/ttyUSB0
Listening as 00deadbeef on 2425 MHz
# 33 37 cd 2e 08 00 01 01 04 4d 8a c1 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c2 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e5 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c3 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 e9
# 33 37 cd 2e 08 00 01 01 04 4d 8a c4 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a c5 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ea
# 33 37 cd 2e 08 00 01 01 04 4d 8a c6 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 e9
# 33 37 cd 2e 08 00 01 01 04 4d 8a c7 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ec
# 33 37 cd 2e 08 00 01 01 04 4d 8a c9 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e7 ea
# 33 37 cd 2e 08 00 01 01 04 4d 8a ca 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 eb
# 33 37 cd 2e 08 00 01 01 04 4d 8a cb 04 4d 8a 06 02 00
 24 b2 0c 72 65 6d 6f 74 65 50 69 6e 67 56 32 e8 e9
```

9c	00039b	0001 TTL=3	Multicast RPC	Method: pingReplyV2(5, 0, 67)
c2	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 1, False, False, 0, -27637)
9d	00039b	0001 TTL=3	Multicast RPC	Method: pingReplyV2(5, 0, 67)
c3	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 1, False, False, 0, -27637)
9e	00039b	0001 TTL=3	Multicast RPC	Method: pingReplyV2(5, 0, 67)
c4	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 1, False, False, 0, -3655)
c5	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, True, False, 0, -3655)
9f	00039b	0001 TTL=3	Multicast RPC	Method: pingReplyV2(5, 0, 66)
c6	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, False, False, 0, -3655)
a0	00039b	0001 TTL=3	Multicast RPC	Method: pingReplyV2(5, 128, 255)
c7	044d8a	0001 TTL=1	Multicast RPC	Method: ackArmed(0, 5)
c8	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, False, False, 0, -3655)
c9	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, False, False, 0, -3655)
ca	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, False, False, 0, -3655)
cb	044d8a	0001 TTL=1	Multicast RPC	Method: remotePingV2(0, 2, False, False, 0, -3655)
79	044d9a (024d8a)	0001 TTL=1	Multicast RPC	Method: fh"eCœMuti@le(1536, 0, 16, 0, 0, -8191, 0, 0, 259, 0, 0, 11942, 0, 0, 37
7a	e44d8a (044d8a)	fa01 TTL=209	Multicast RPC	
70	044dba (044f8a)	0061 TTL=5	Multicast RPC	
80	04bd8a (044d8a)	0001 TTL=1	Multicast RPC	
85	944d8a (044d8a)	0601 TTL=65	Multicast RPC	Method: (0,\n 4096,\n 0,\n 1,\n 0,\n 0,\n None,\n -28659,\n 2,\n 0,\n 0,\n -1,\n 19
86	044d8a (044d4a)	0001 TTL=65	Multicast RPC	
87	044c8a (044d8a)	0001 TTL=33	Multicast RPC	Method: fûreOueMultiple(0, 0, 1, 0, 0, 1280, 7, 0, 24576, -1, 48, 0, 0, 0, 0, 0, 0, 0
89	f24d5a (044f8a)	3001 TTL=1	Multicast RPC	
8d	048d8a	d00101	Mesh RTR	Type: unknown

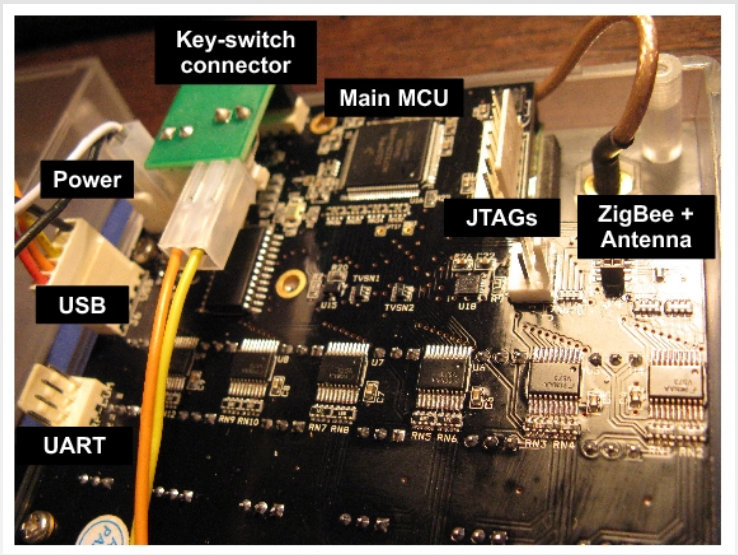
```
4 4D 8A 02 02 00 09     ......M...M.....
3 41 72 6D 65 64        .......ackArmed
```

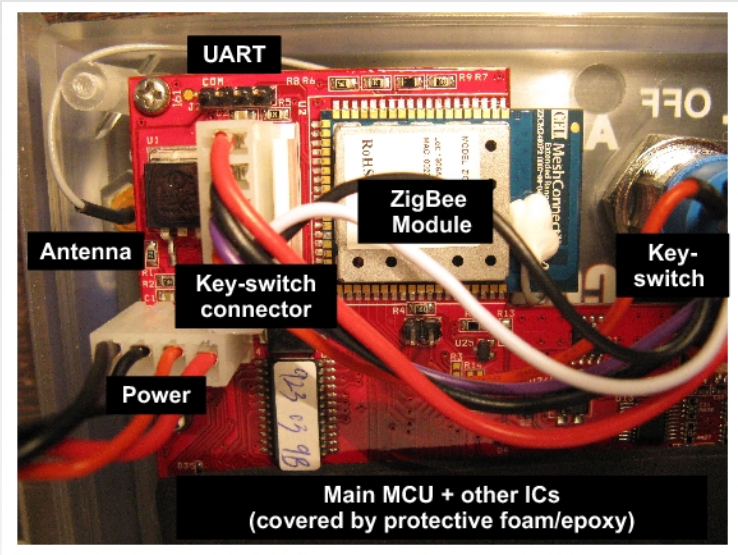
Attack Summary

- Replay/Inject

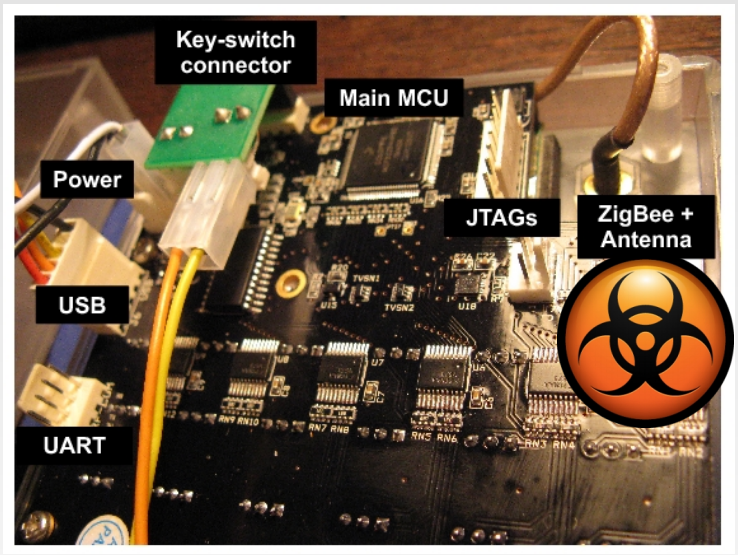


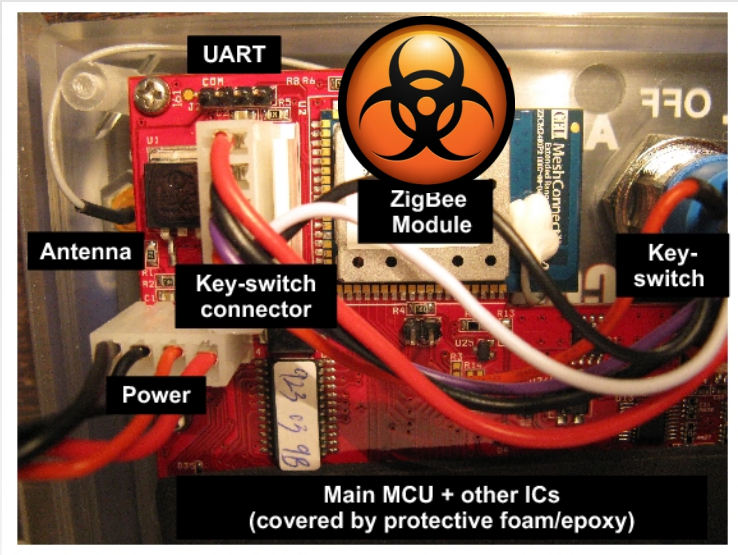
Ultimate Attack Scenario - Theory



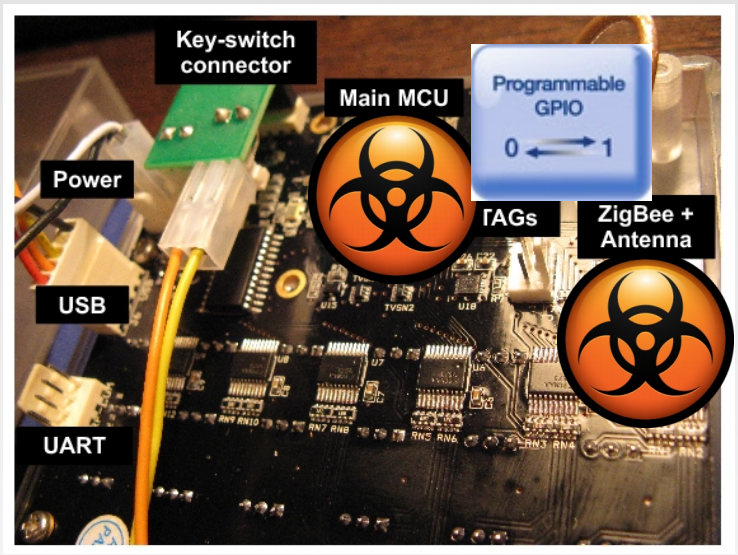


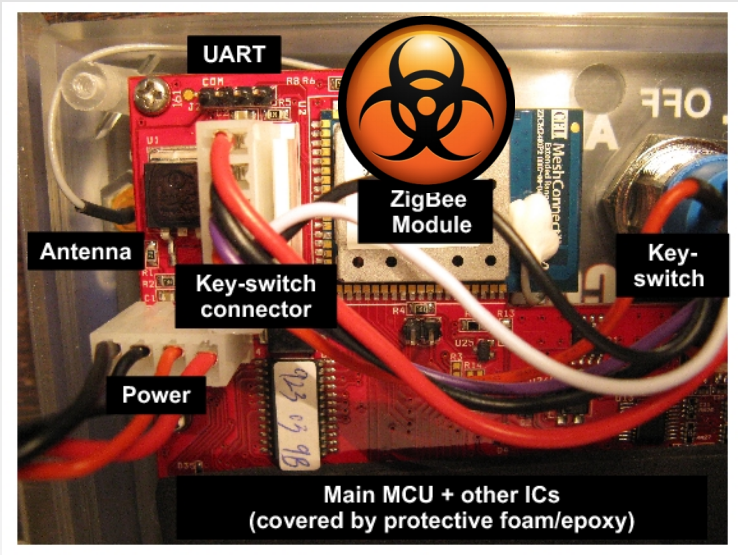
Ultimate Attack Scenario - Theory



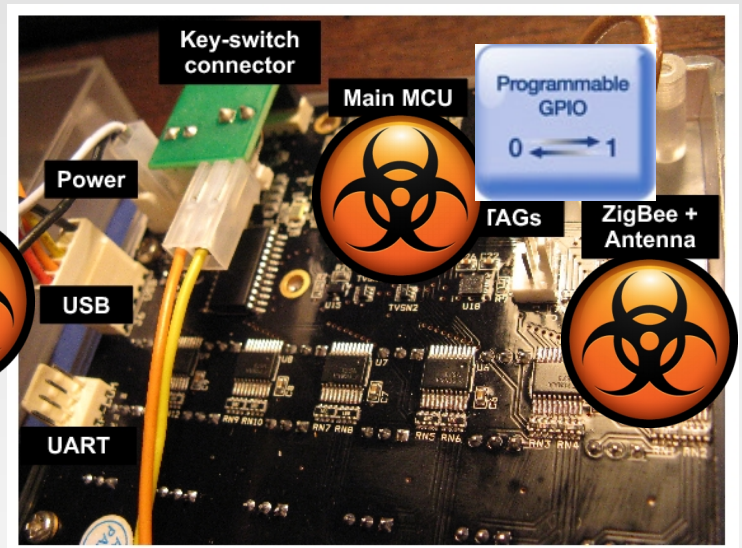


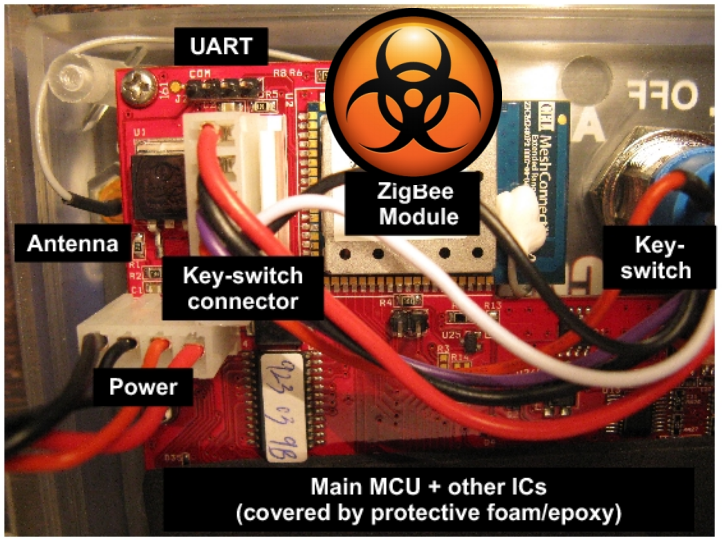
Ultimate Attack Scenario - Theory



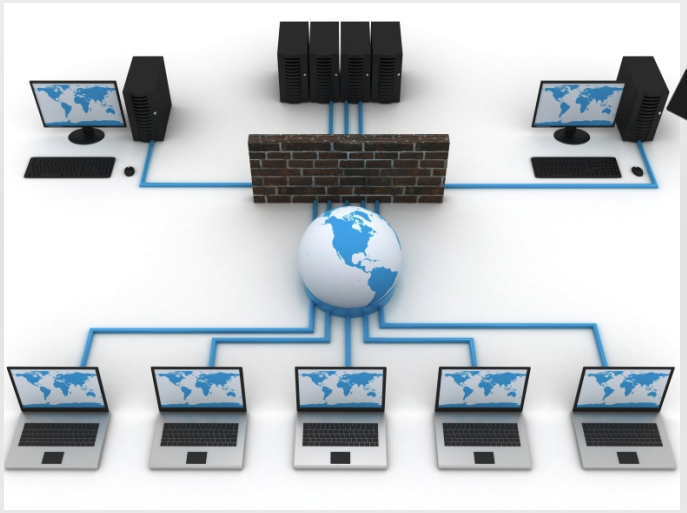


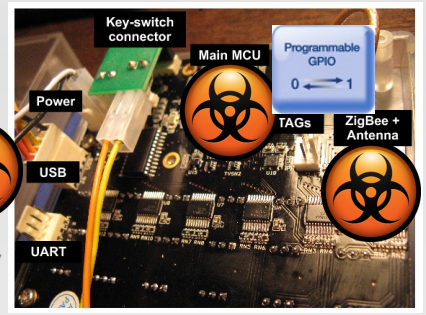
Ultimate Attack Scenario - Theory

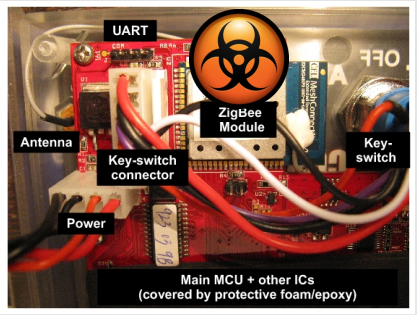




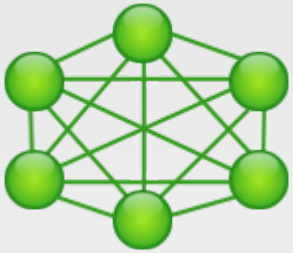
Ultimate Attack Scenario - Theory







Wireless Mesh Sensor Network



Disclosure Process

- We took vulnerabilities very seriously
  - Responsible disclosure
  - Contacted the vendor
  - Coordinated the content and paper release
- Vendor
  - Confirmed the issues
  - Had security improvements being deployed
  - Many of the issues now fixed
  - Shipping updates and communicates to customers

Agenda

- Pre-Intro
  - It all starts with a good piece of firmware
- Intro
  - What are the *wireless firing systems*?
- Methodology
  - System analysis
  - Attack development
- Results
  - Attacks summary
  - Disclosure process

Future Work

- Solutions for this kind of devices exist
  - Secure firmware upgrades
  - Authenticated communications
  - Secure restore and debug chains
  - Practical key distribution
  - *Latency control, secure positioning?*
- How to get those actually used?
  - Vendor communicates to regulators/industry groups
  - We contacted certification bodies

Future Work

- Implement some other attacks
  - Main MCU firmware upgrade via 802.15.4 (remote)
  - UART-based exploitation (local)

Conclusions

- Firmware analysis gets *better and faster*
  - **Large-scale automated analysis=great results!**
  - **Firmware.RE** is a push in a right direction
- Wireless security is an issue in many products
  - Even for life critical systems
  - Vulnerable to basic attacks!
- Firing systems' security must be taken *seriously*
  - *Solution probably involves certification, regulation*

Conclusions

Fireworks/CPS Hacking:

Putting the real meaning of *weapon* into the *weaponized exploits* concept

Conclusions

- Fireworks whitepaper:
- http://s3.eurecom.fr/docs/wisec14_Costin.pdf

- Firmware whitepaper:
- http://s3.eurecom.fr/docs/usenixsec14_costin.pdf

Thank You!
Questions/Concerns?

@costinandrei, @FirmwareRE

andrei.costin@eurecom.fr

aurelien.francillon@eurecom.fr

References

- [1] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, *"A Large-Scale Analysis of the Security of Embedded Firmwares"*, In Proceedings of the 23rd USENIX Conference on Security (to appear)
- [2] A. Costin, J. Zaddach, *"Poster: Firmware.RE: Firmware Unpacking and Analysis as a Service"*, In Proceedings of the ACM Conference on Security and Privacy in Wireless Mobile Networks (WiSec) '14