Home Insecurity: No alarm, False alarms, and SIGINT

Logan Lamb
lamblm@ornl.gov

Agenda

- Motivation
- Models and Methodology
- Attack Primitive Implementation
- Application to three security systems
- Observations
- Conclusion

Who am I?

- Researcher for Center for Trustworthy Embedded Systems at ORNL
- Focus on V2X currently
- Ongoing privacy research involving intelligent transportation systems

Home Security System Value

- Ostensibly protects your home and occupants from intruders!
  - Previous hacks
    - Disable Sensors
    - Control GSM
    - Z-Wave (Home Automation)
- Lower insurance premiums!

Motivation

- Complete dominance of the security system
  - Render it useless
  - If possible, make owning a security system a liability

Motivation

- Covert Infiltration and Exfiltration
- Monitor Behavior
- Induce Behavior

Motivation

- Covert Infiltration and Exfiltration
  - Monitoring Company
  - Occupants
- Monitor Behavior
- Induce Behavior

Motivation

- Covert Infiltration and Exfiltration
- Monitor Behavior
  - Particular Occupants (better for homes)
  - Aggregate (better for businesses)
- Induce Behavior

Motivation

- Covert Infiltration and Exfiltration
- Monitor Behavior
- Induce Behavior
  - Monitoring Company
  - Occupants

MODELS AND METHODOLOGY

Adversary Model

Desires….

- General solution
- High Yield
- Cheap

Adversary Model

Desires…. A **WIRELESS** hack!

- General solution
- High Yield
- Cheap

Adversary Model

Desires…. A **WIRELESS** hack!

- General solution
  - Bet the sub GHz RF is similar across manufacturers and super vuln ☺
- High Yield
- Cheap

Adversary Model

Desires…. A **WIRELESS** hack!

- General solution
- High Yield $$$
  - Everything is going wireless!
- Cheap

Adversary Model

Desires…. A **WIRELESS** hack!

- General solution
- High
- Cheap
  - SDRs are getting cheaper, software is ‘cheap’

Adversary Model

Desires….

- Covert Infiltration and Exfiltration
- Monitor Behavior
- Induce Behavior

Adversary Model

Desires….

- Covert Infiltration and Exfiltration
  - Attempt with **Jamming**
- Monitor Behavior
- Induce Behavior

Adversary Model

Desires….

- Covert Infiltration and Exfiltration
  - Attempt with **Jamming**
- Monitor Behavior
  - Attempt with **SIGINT**
- Induce Behavior

Adversary Model

Desires….

- Covert Infiltration and Exfiltration
  - Attempt with **Jamming**
- Monitor Behavior
  - Attempt with **SIGINT**
- Induce Behavior
  - Attempt with **Replay**

Adversary Model

Desires….

- Covert Infiltration and Exfiltration
  - Attempt with **Jamming**
- Monitor Behavior
  - Attempt with **SIGINT**
- Induce Behavior
  - Attempt with **Replay**

Adversary Model

- Only use Software Defined Radio
  - No rom dumping (black box testing)
- Will not craft custom messages
  - No protocol fuzzing
  - No packets of death

Adversary Model

- Why so many constraints?
  - Easy to commodify these attacks if successful
  - Relax the restrictions if the adversary needs to be more sophisticated ☺

Security System Model

- Build the Model based on the Adversary's capabilities
- Intra-system communications are the focus

Security System Model

Types of Intra-Home Communications

- Vulnerable
  - Legacy sub GHz communications
- Secure
  - Everything else

Security System Model

Types of Devices in a System

- Sensors
- Alarm Devices
  - Alert occupants and/or monitoring company
- Bridges
  - Convert one communication type to another
- Other

Security System Model

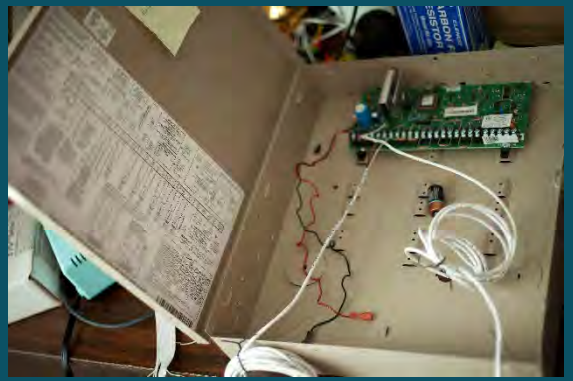
Interesting Properties

- Sensors trigger their events even when the system is disarmed
- Sensors have one way communication
- Only alarm devices can alert the stakeholders

Security System Model

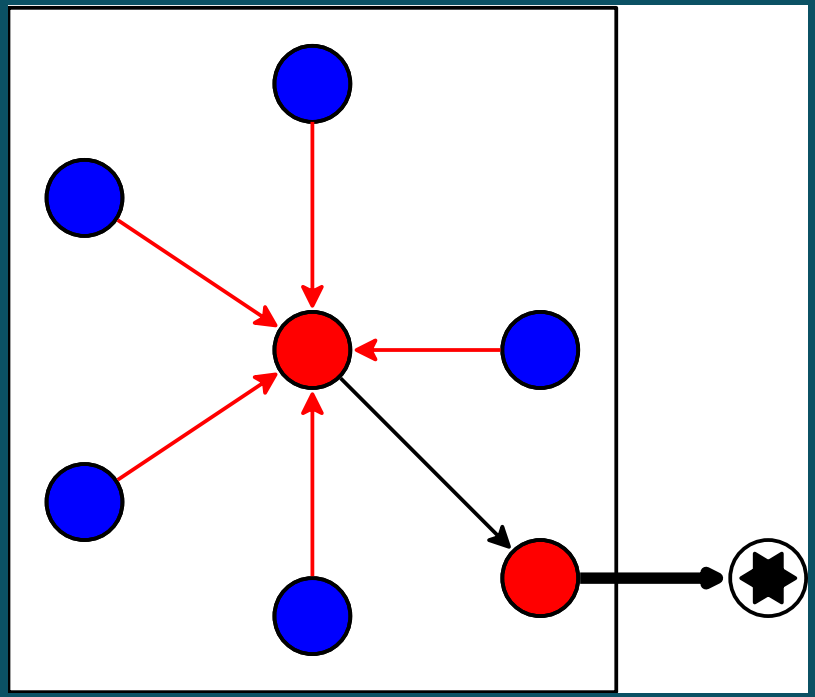
- Directed Graph
  - Vertices are devices (Sensors, Alarm Devices, Bridges)
  - Edges are communication channels (Vulnerable wireless, everything else)
  - Transmissions flow from source (sensors) to sinks (alarm devices)

Honeywell Devices



Honeywell Digraph

- 5 Sensors
  - 2 Door
  - 3 Motion
- 2 Alarm Devices
  - 1 Keypad
  - 1 Control Panel



Methodology

1. Identify all devices and their communication type(s)
2. Generate a digraph from sources to sinks
3. If there are any wireless communications, attempt the SIGINT attack primitive
4. If a path exists from source to sink that involves a wireless communication channel, attempt the Jamming and Replay attack primitives
5. Evaluate the attained level of control and situation awareness

ATTACK PRIMITIVE IMPLEMENTATION

Prerequisites

- Software Defined Radio, USRP N210
- GNU Radio
- Tuned Antenna
- System to test with

Prerequisites

- Software Defined Radio, USRP N210
- GNU Radio
- Tuned Antenna
- System to test with

Prerequisites

- Software Defined Radio, USRP N210
- GNU Radio
- Tuned Antenna
- System to test with

Prerequisites

- Software Defined Radio, USRP N210
- GNU Radio
- Tuned Antenna
- System to test with
  - Honeywell

Tuning In

- Spectrum Analyzer
  - Dedicated
  - Build with SDR
  - Consult FCC documentation

Jamming

- Spot Jamming
  - Blast noise! :D
  - It….works? Really?
- Manufacturers are aware of the threat
  - Introducing ‘RF Jam’
  - Once enabled, the spot jammer fails

Periodic Jamming

- At what point does the interference go from benign to malicious?
  - Noise floor
  - Number of malformed transmissions

Noise Floor Testing

- How long can the spot jammer be used?
  - About a minute
- Noise floor is checked

Malformed Packet Testing

- In GRC, layout flow chart that flips bits
  - Induce errors
  - Low duty cycle

How quickly can we turn simple jamming off and on?

- Pretty quick, about ¼ of a second
- Is that good?
  - Yup
  - Supervisory transmission requires 0.77 s
  - Alarm transmission requires 3.54 s

What does this get us?

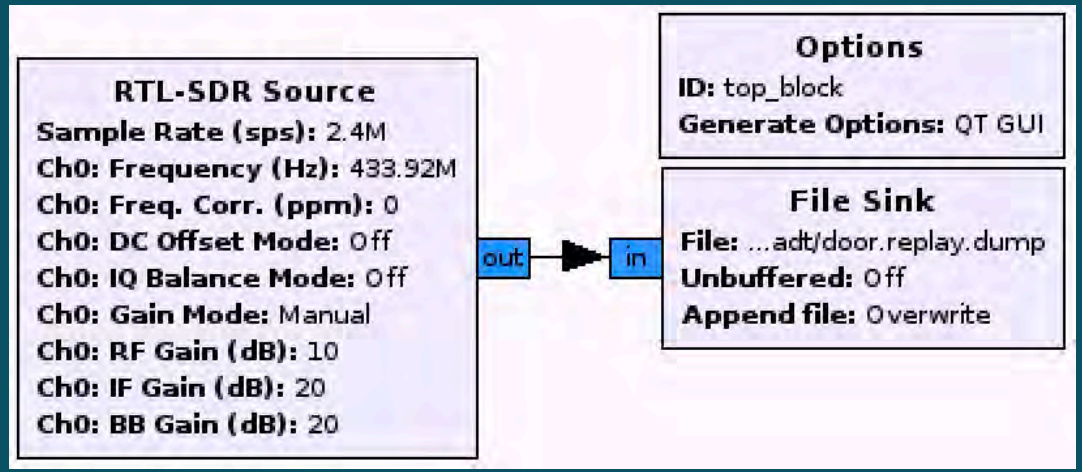
- RF Jam Disabled
  - Covert infiltration and exfiltration
- RF Jam Enabled
  - Covert infiltration, exfiltration, and alarm triggering
  - When enabled, RF Jam is a liability

SIGINT

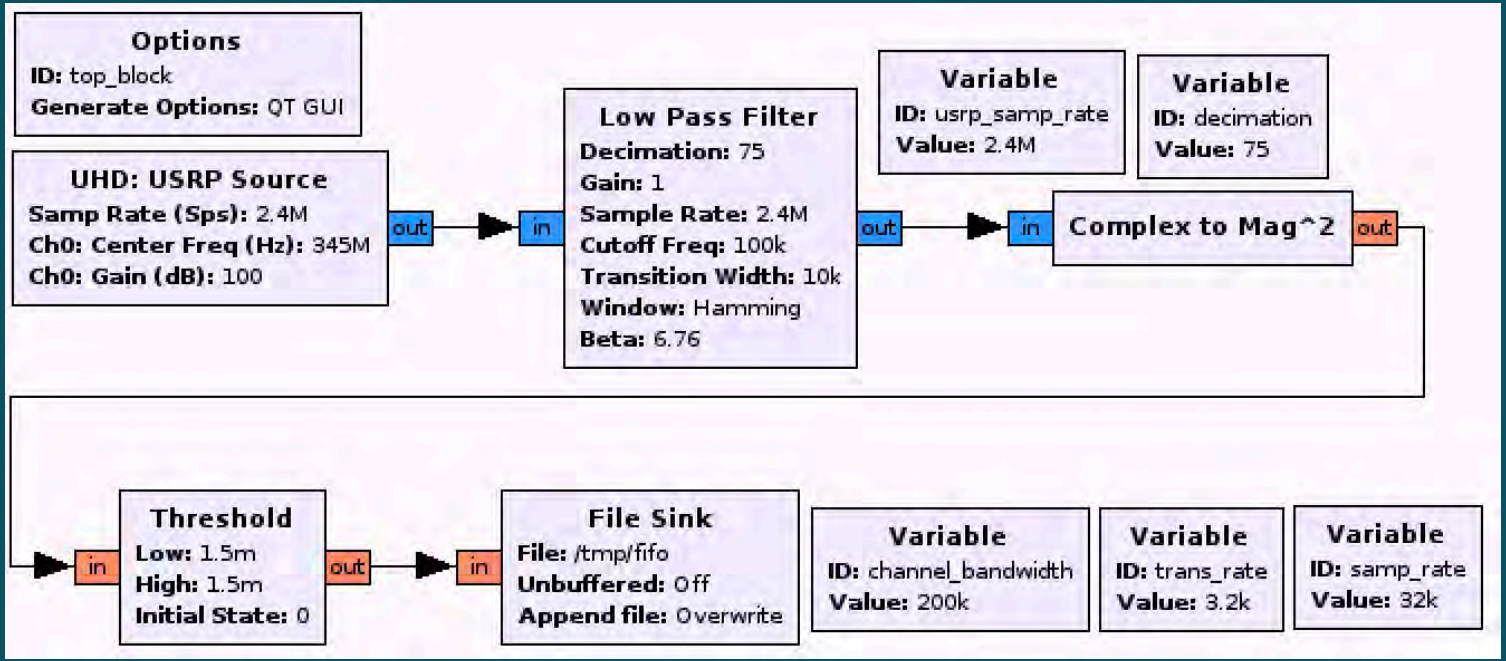
- Tiers of complexity
  - RF Capture
  - Bitstream
  - Protocol Capture
    - We know what that means

RF Capture

- Simple in GRC
  - Useful if more intel is available



Bitstream Capture



Bitstream -> Packets

- Helpful if more intel is available
  - From the FCC
    - Manchester encoded
    - 3200 Baud
    - Word length 64 bits
    - Packets are repeated to form a transmission

Bitstream -> Packets

- Just Software
  - Read bitstream from stdin
  - Figure out the number of samples per bit
  - Convert samples to bits
  - Manchester decode and print

Honeywell Door Packets

- 0xfffe 84d4 0280 512c
- 0xfffe 84d4 02a0 d1ef
- 0xfffe 84d4 02e0 506c
- 0xfffe 8faa 8380 4d3d
- 0xfffe 8faa 83a0 cdfe
- 0xfffe 8faa 83e0 4c7d

Reverse Engineering

- 0xfffe 84d4 0280 512c
- 0xfffe 84d4 02a0 d1ef
- 0xfffe 84d4 02e0 506c
- 0xfffe 8faa 8380 4d3d
- 0xfffe 8faa 83a0 cdfe
- 0xfffe 8faa 83e0 4c7d

Device Serial:
A 031-6418

Device Serial:
A 102-6691

Reverse Engineering

- `0xfffe`
  - In every packet
  - Looks like a preamble and sync bit

Reverse Engineering

- 0x{80, a0, e0}
  - All three appear for both sensors
  - 0xa0 – Open Event
  - 0x80 – Closed Event
  - 0xe0 – Tamper Event

Reverse Engineering

- 0x{84d402, 8faa83}
  - Unique to each sensor, in every packet
  - 0x84d402 No significance, but
  - 0x4d402 316,418 in decimal
  - 316,418 -> A 031-6418
  - 0x8faa83 -> A 102-6691

Reverse Engineering

- `0x{512c, d1ef, 506c, 4d3d, cdfe, 4c7d}`
  - What is this? Different for each packet seen
  - Probably a CRC, time to break out…
  - REVENG

CRC Reversing with REVENG

- Arbitrary-precision CRC calculator and algorithm finder
- Search every packet for a one byte or two byte CRC
- Easy bash script…

CRC Reversing with REVENG

```
while read hex_line; do
  let len=${#hex_line}-2
  for ix in $(seq 0 2 $len); do
    val=$(./reveng -w8 -s ${hex_line:ix} 2> /dev/null)
    ret=$?
    if [ $ret -eq 0 ]; then
      echo ${hex_line:ix}" "$val
    fi
  done
  let len=${#hex_line}-4
  for ix in $(seq 0 2 $len); do
    val=$(./reveng -w16 -s ${hex_line:ix} 2> /dev/null)
    ret=$?
    if [ $ret -eq 0 ]; then
      echo ${hex_line:ix}" "$val
    fi
  done
done
```

CRC Reversing with REVENG

```
l5o@k:/tmp/crc_check$ cat all_reveng.txt
fffe84d40280512c
fffe84d402a0d1ef
fffe84d402e0506c
fffe8faa83804d3d
fffe8faa83a0cdfe
fffe8faa83e04c7d
fffe8cf96c00944e
fffe8cf96c021441
fffe8cf96c80174d
fffe8abec9003728
fffe8abec902b727
fffe8abec980b42b
fffe8cf91e00384b
fffe8cf91e80bb48
l5o@k:/tmp/crc_check$ time cat all_reveng.txt | ./crc_check.sh
84d40280512c width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
84d402a0d1ef width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
84d402e0506c width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8faa83804d3d width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8faa83a0cdfe width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8faa83e04c7d width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8cf96c00944e width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8cf96c021441 width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8cf96c80174d width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8abec9003728 width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
b727 width=8 poly=0x9b init=0x00 refin=true refout=true xorout=0x00 check=0x25 name="CRC-8/WCDMA"
8abec902b727 width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8abec980b42b width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8cf91e00384b width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"
8cf91e80bb48 width=16 poly=0x8005 init=0x0000 refin=false refout=false xorout=0x0000 check=0xfee8 name="CRC-16/BUYPASS"

real    0m0.701s
user    0m0.214s
sys     0m0.315s
```

Reverse Engineering

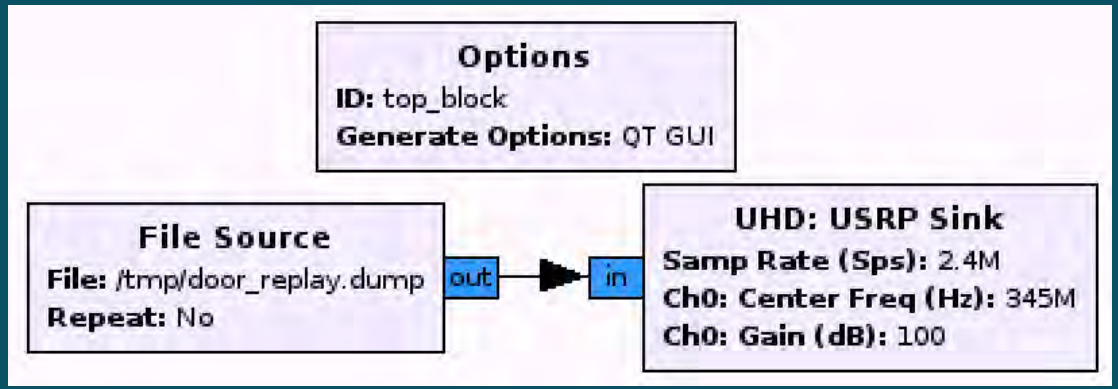
- 0xfffe 84d4 0280 512c
- 0xfffe – Preamble and sync bit
- 0x84d402 – Serial
- 0x80 – Event type
- 0x512c – CRC-16/BUYPASS

What does this get us?

- Monitoring capability
  - Helps with Situational Awareness
- How?
  - Different sensors transmit different events
  - Sensors are installed in logical locations

Replay

- What does this get us?
  - Induce behavior with false alarms



APPLICATION TO THREE SYSTEMS

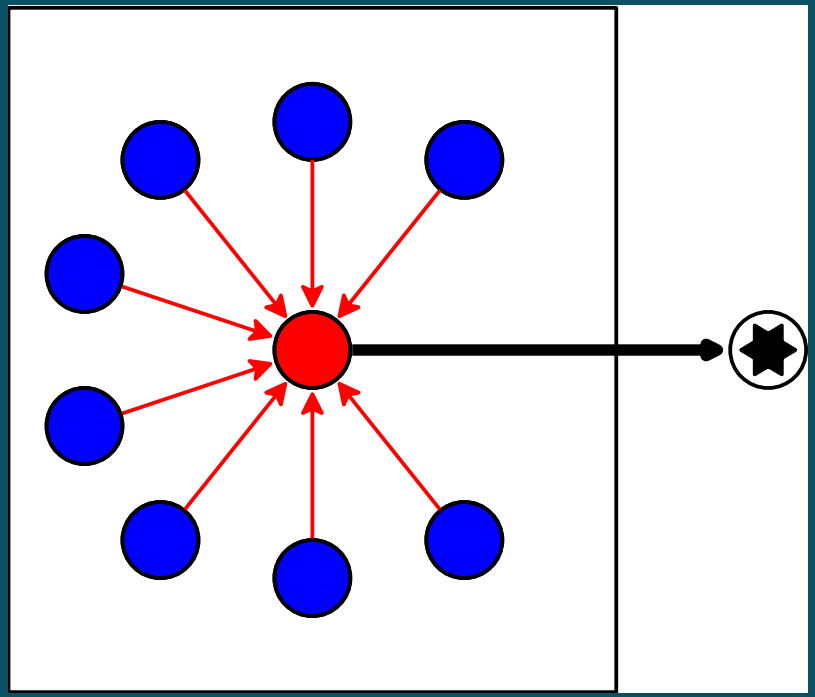
Honeywell

- Covered in the attack primitive implementation section
- Summary
  - Covert Infiltration and Exfiltration ✔
  - Induce Behavior ✔
  - Monitor Behavior ✔

ADT Devices

ADT Digraph

- 8 Sensors
  - 4 Door
  - 3 Glass Break
  - 1 Motion
- 1 Alarm Devices
  - 1 Panel (GSM out)



ADT Specifics

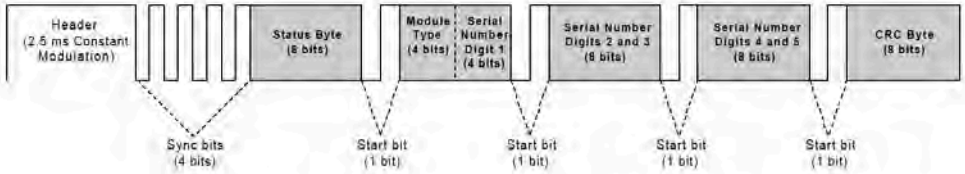
- Completely Wireless
- RF Jam Detection capable, but disabled
- Unable to get Installer Code
  - Yeah, there's a fee for that
  - Thanks ADT

ADT Changes

- Simple Jammer and Replay
  - Center Frequency change to 433.96
- SIGINT
  - Center Frequency change to 433.96
  - Reverse Engineering not implemented, but all info is given in FCC Documentation…

ADT Changes

Figure 2 – Data Packet Format



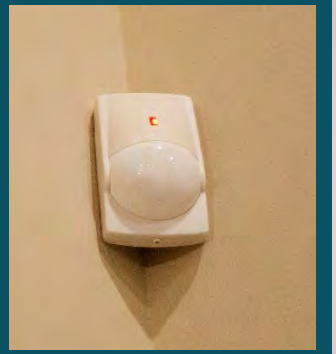
Just Needs to be Implemented!

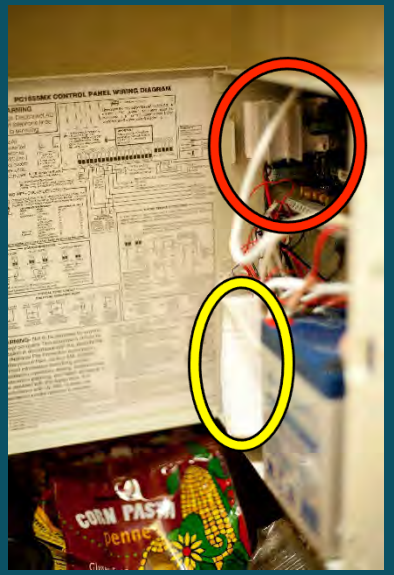
Packet Component	Description
Header	2.5 ms Of Carrier Frequency To Indicate Start of Packet
Sync Bits	4 Logic '1' Bits For Synchronization
Status	Status Information: Minimum Valid Value = 80 hex (1000 0000 Binary) Maximum Valid Value = FF hex (1111 1111 Binary)
Start Bit	1 Logic '1' Bit For Synchronization
Module Type	Valid Module Types Currently Used Are: 2 Hex (0010 Binary), 3 Hex (0011 Binary) 4 Hex (0100 Binary), 5 Hex (0101 Binary) 6 Hex (0110 Binary), 9 Hex (1001 Binary) Minimum Valid Value = 2 or 4 Hex Maximum Valid Value = 3,5,6 or 9 Hex
Serial # Digit 1	Minimum Valid Value = 0 Hex (0000 Binary) Maximum Valid Value = F Hex (1111 Binary)
Start Bit	1 Logic '1' Bit For Synchronization
Serial # Digit 2&3	Minimum Valid Value = 01 Hex (0000 0001 Binary) Maximum Valid Value = F0 Hex (1111 1110 Binary)
Start Bit	1 Logic '1' Bit For Synchronization
Serial # Digit 4&5	Minimum Valid Value = 01 Hex (0000 0001 Binary) Maximum Valid Value = F0 Hex (1111 1110 Binary)
Start Bit	1 Logic '1' Bit For Synchronization
CRC	Cyclic Redundancy Check Value
	CRC Byte Calculated From Above Minimum Values = 39 Hex (0011 1001 Binary)
	CRC Byte Calculated From Above Maximum Values = 91 Hex (1001 0001 Binary)

ADT

- Summary
  - Covert Infiltration and Exfiltration ✔
  - Induce Behavior ✔
  - Monitor Behavior ✔
    - Not currently implemented

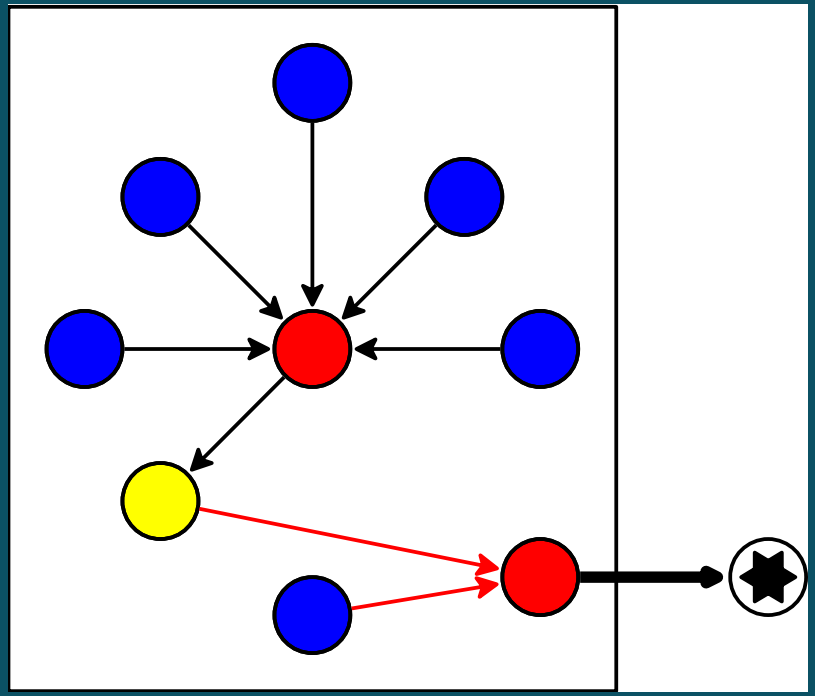
2GIG Devices



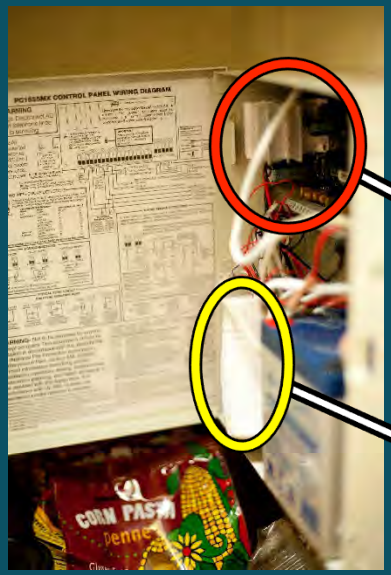


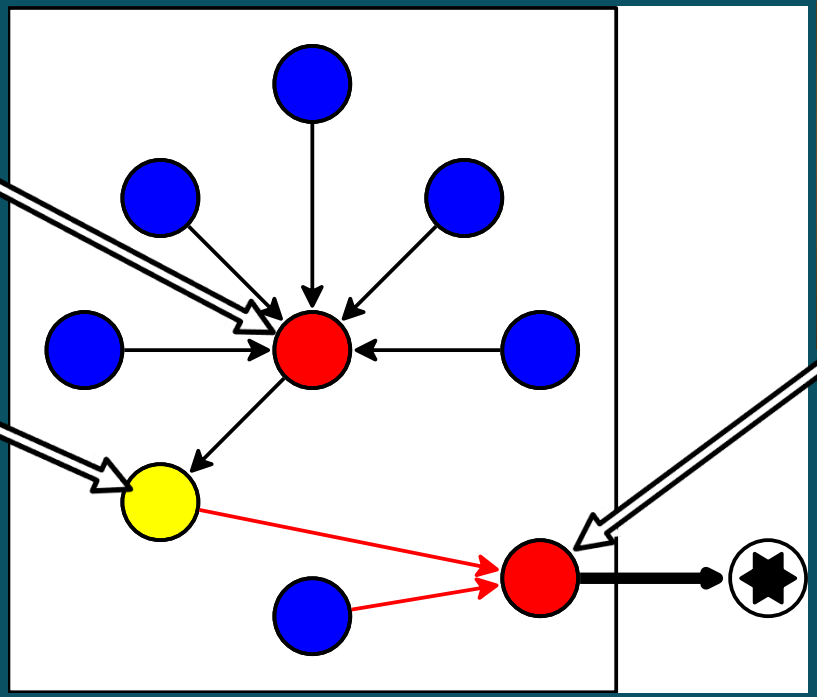
2GIG Digraph

- 6 Sensors
  - 5 Door
  - 1 Motion
- 2 Alarm Devices
  - 1 Go!Control Panel
  - 1 12V Control Panel
- 1 Bridge Device
  - 2GIG Takeover Module

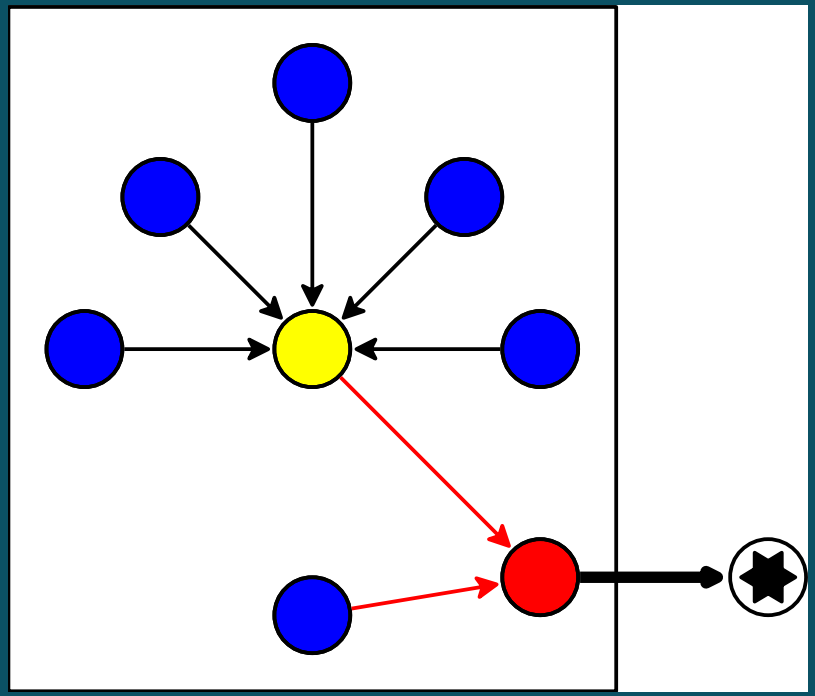


2GIG Digraph





2GIG Equivalent Digraph



2GIG Specifics

- Hybrid System
  - Wired and wireless devices
  - RF Jam Detection capable, but disabled
    - Sooo, we enabled it ☺

2GIG

- Summary
  - Covert Infiltration and Exfiltration ✔
  - Induce Behavior ✔
  - Monitor Behavior ✔

Observations

- Full control and monitoring on all systems
- Simple communications
- Legacy communications

Thanks!

Logan Lamb
lamblm@ornl.gov