Privasec
Governance and Information Security Partners

Breaching Physical Security & Generally Causing Mayhem, with Wireless Signals

T.J. Acton | BSidesAU 18 March 2017

Who is this guy?

- Security Researcher & Penetration Tester
- Work at Privasec – BSidesAU Sponsor
- Founded Cyberspectrum Sydney (an SDR/RF meet-up group)

- Father of one…….

Objective

Get into your house, protected by a wireless alarm.

- ~~Option 1 – Jam the signal from the sensors to the base.~~
- ~~Option 2 – Cut the power lines.~~
- Option 3 – Reverse engineer the alarm remote and disable the wireless alarm system
  - Less risky than jamming
  - Infiltrate the target site without alarms sounding
  - Reactivate alarm on the way out to avoid suspicions

The Target Device

The Target Device

Black Box Approach

- I could have started with access to device.
- Attack scenario/narrative altered here to emulate a black-box attack.
- Also makes the narrative suitable to describe a physical penetration test approach

The Attack

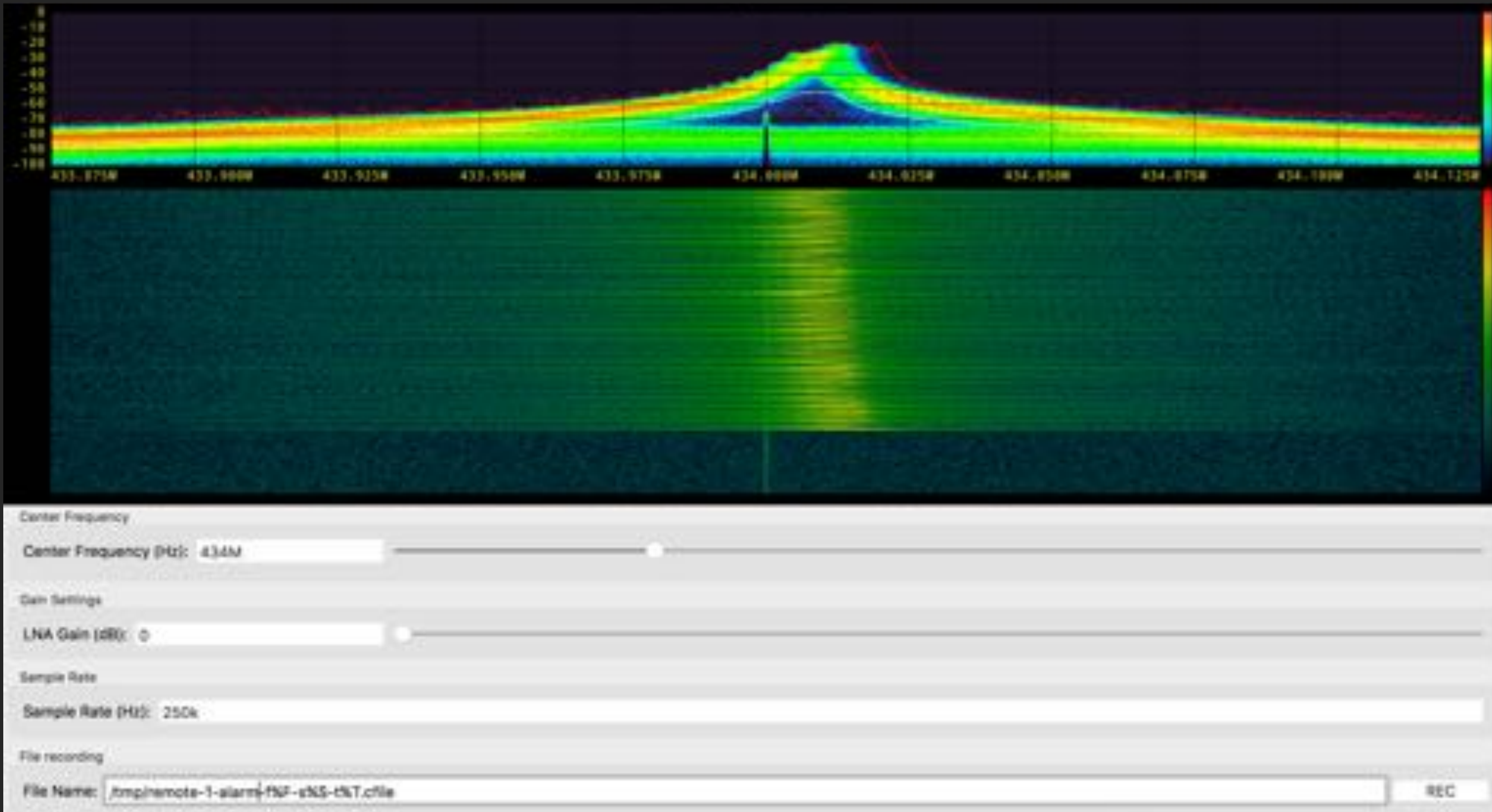
Step 1 - Surveillance & Initial Recon

- Conduct initial surveillance around the target site.

→ Determine that there's a wireless alarm system in place.

Step 2 –Capture wireless signal

- Find the frequency & capture the signal



Step 3 – Identify Captured Signal

Looking at the captured signal in Inspectrum, we can see:

- OOK-PWM (On-Off Keying, Pulse Width Modulation)
- 25 bits per packet
- Other info such as the baud rate (transmission speed)

Step 4 – Analyse captured Signal

OSINT (Open Source Intelligence) gathering can be useful to identify the system based on the metadata we now have:

- We know the frequency
- We know the number of bits in a packet
- We know the modulation type
- We conducted recon IRL and may have been able to take a picture (reverse image search?)

- Option 1 - If we can fingerprint (identify) the alarm system, we can buy our own to reverse engineer
- Option 2 - If we can't, we can still reverse engineer it, we'll just need to rely on our ability to make logical/plausible assumptions

Step 5 - Reverse Engineer captured signal

Privasec
Governance and Information Security Partners

- To have a good sample of data to work with, we ideally should capture the following at minimum:
  - Remote 1 - Unlock 1
  - Remote 1 - Lock 1
  - Remote 1 - Unlock 2
  - Remote 2 - Unlock 1
  - Remote 2 - Lock 1

Step 5 - Reverse Engineer captured signal

Now that we have our data, we need to try to understand it.

- We already know it's PWM (Pulse Width Modulation), so it's easy to demodulate it:
  - Demodulation effectively means converting the raw radio waves we captured into binary
  - Demodulation is usually very easy, but can be time consuming if performed manually/visually
- We then need to compare our captures, along with the context we have for them, and look for patterns

Step 5 - Reverse Engineer Captured Signal

- Remote 1:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Unlock 2	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0

- Remote 2:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

Step 5 - Reverse Engineer Captured Signal

- Bytes 1 & 2 (bits 0-7, and 8-15) both change between the two remotes, but are otherwise static across functions from the same remote. So we'll assume it's a device ID.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Unlock 2	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

Step 5 - Reverse Engineer Captured Signal

Privasec
Governance and Information Security Partners

- Byte 3 (bits 16-23) changes depending on function pressed, but is the same regardless of the device used. It changes reliably based on button press. It must be the function ID.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Unlock 2	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

Step 5 - Reverse Engineer Captured Signal

- The last bit never changed to anything other than 0. We'll continue to ignore it for now.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Unlock 2	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

Step 5 - Reverse Engineer Captured Signal

We need a better way of representing the numbers. Integers will be easier to express / remember.

- The unlock function binary is: 00001100
  - Converting that to an integer gives us: 12
- The lock function binary is: 00110000
  - Converting that to an integer gives us: 48

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

Step 5 - Reverse Engineer Captured Signal

Continuing with our attempt to assign easier to remember/digest values, we now convert the two device-id bytes from each remote to integers:

- Remote 1:
  - Binary: 01000001 01010001 = 16721
- Remote 2 (pictured below):
  - Binary: 01011111 01110001 = 24433

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Unlock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1	0	0	0
Lock 1	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
B	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0

What we now know

Privasec
Governance and Information Security Partners

- We know the function codes for each button
- We know the device IDs for two remotes
- We can reliably predict each section of the transmission, seems pretty static
- We know what frequency to transmit on
- We know how to modulate our signals
- We know the baud rate

→ We can now combine all of this knowledge to impersonate a valid remote, and trigger any action we desire, at our leisure.

Step 6 - Putting it all together

We can now send an unlock signal, impersonating remote 2 (id: 24433):

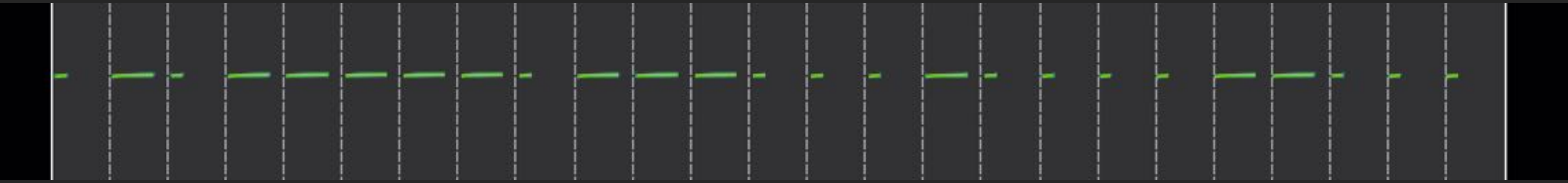
- 24433 converted to binary is: 01011111 01110001
- The disarm command (12) converted to binary is: 00001100

- Let‘s not forget the extra 0 that we‘ve been ignoring at the end...

Step 6 - Putting it all together

Privasec
Governance and Information Security Partners

- The packet we need to transmit looks like this (if our transmitter knows how to PWM):
  - 0101111101110001000011000



- Otherwise, the packet we need to transmit as ‘vanilla’ OOK, would be:
  - 10001110100011101110111011101110100011101110111010001000100011101000100
    010001000111011101000100001000

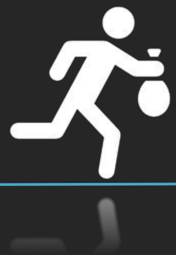
Step 6 - Putting it all together

We could have just 'replayed' the capture for remote 2's disarm function, as we already have it. However….. because we reverse engineered it:

- We can approach any site in future, that happens to use this alarm system, and very easily craft a disarm message, just by intercepting a single key-press (such as someone arming the alarm system as they leave their home).
- The total attack time (with a little automation in place) is just a few short seconds, and the only pre-condition is to capture 1 transmission from a valid remote key-press.

Step 7 - Attack Scenario

1. Hide in the bushes, or in a seedy van
2. Capture an ‘arm system’ message
3. Retrieve the code
4. Construct our own ‘disarm system’ message
5. Wait for site to be vacant
6. Disarm the alarm (send our constructed message)
7. ?????
8. Re-arm the alarm on the way out (to delay theft discovery)
9. Profit



Step 7 - Attack Scenario

What if we can't capture a transmission?

What if you don't want to wait?

What if there is no where to hide/camp without raising suspicion?

How about brute forcing the signal?

Brief Digression:

Brute forcing the signal.

Brief Digression: Brute-force

Privasec
Governance and Information Security Partners

[Blackbox Scenario] The device ID is comprised of a maximum length of 2 bytes (2^16)

= 65,536 possibilities

[Whitebox Scenario] The Device ID is a product of a tri-state address, so the maximum pool of valid IDs is actually (3^8):

= 6,561 possibilities

Brief Digression: Brute-force

This tells us two things:

- If we are unable to capture a valid key-press, but have some time, we can bruteforce the device ID at a modest rate of 2 attempts per second:
  - Blackbox perspective: 65,536 possibilities, achievable in under 10 hours
  - Whitebox perspective: 6,561 possibilities, achievable in under 1 hour

Brief Digression: Brute-force

Privasec
Governance and Information Security Partners

This tells us two things:

- If we are unable to capture a valid key-press, but have some time, we can bruteforce the device ID at a modest rate of 2 attempts per second:
  - Blackbox perspective: 65,536 possibilities, achievable in under 10 hours
  - Whitebox perspective: 6,561 possibilities, achievable in under 1 hour
- Device ID collision is likely

Brute-force possibilities:

Get list of tri-state permutations:

```
2.2.2 :001 > permutations = [8].flat_map{|n| ['1','0','X'].to_a.repeated_permutation(n).map(&:join)}
 => ["11111111", "11111110", "1111111X", "11111101", "11111100", "1111110X", "111111X1", "111111X0", "1:
```

Total # possibilities (sanity check):

```
[2.2.2 :002 > permutations.count
 => 6561
```

Example of one possible device ID, expressed as a trinary address:

```
[2.2.2 :003 > permutations[499]
 => "11X11000"
```

Brief Digression: Brute-force

Conversion to actionable data for brute-forcing

The "formula" for trinary address to device ID conversion:

Tri	PWM	OOK
1	11	11101110
0	00	10001000
X	01	10001110

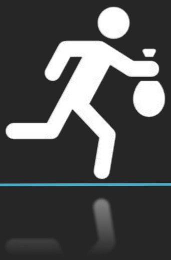
The results of conversions:

Trinary Byte	Device ID (pwm binary to int)	PWM Binary - 2 Bytes	Raw OOK Representation
11X11000	63424	11110111 11000000	1110111011101110100011101110111011101110111....

End of digression

Step 7 - Attack Scenario

1. Hide in the bushes, or in a seedy van
2. Capture an ‘arm system’ message
3. Retrieve the code
4. Construct our own ‘disarm system’ message
5. Wait for site to be vacant
6. Disarm the alarm (send our constructed message)
7. ?????
8. Re-arm the alarm on the way out (to delay theft discovery)
9. Profit



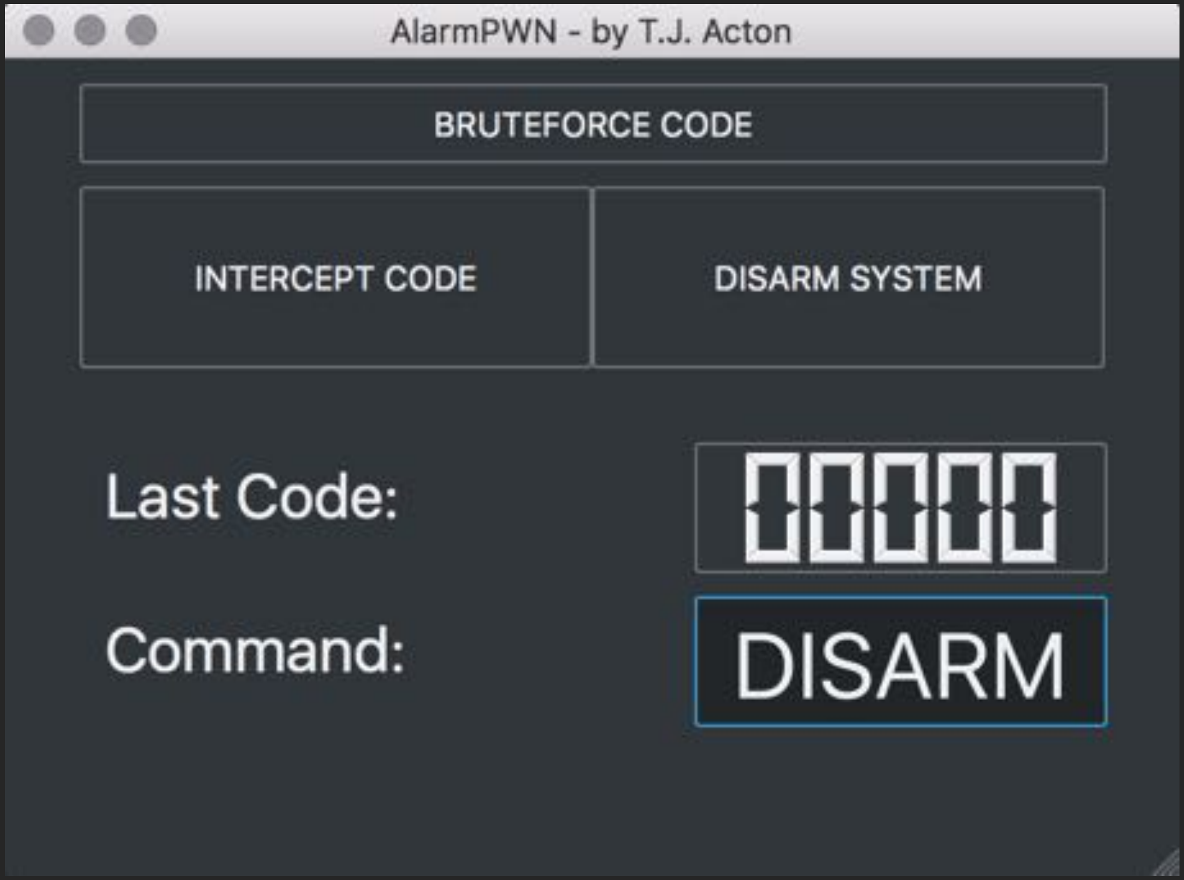
Where we are now

- We have our attack scenarios ☺
- We know how to execute the attack ☺
- It is a manual process
- There is a technical knowledge barrier ☹

Let's up the game a little…

Step 8 – Attack Automation

AlarmPWN - an app, with a GUI:



Step 8 – Attack Automation

The app can be deployed on a Raspberry Pi for discrete, code interception – with or without technical skills

Step 8 – Attack Automation

The code (device ID) is displayed on the screen & stored to the SD Card

You can then either:

- Disarm the system from a laptop with a Yard Stick One (YS1)
- Disarm the system from a purpose built Arduino or similar
- Disarm the system straight from the 'interception device' with a YS1
- Disarm the system from your Android/IOS phone using a PandwaRF (similar to YS1, but controllable via Bluetooth from your phone)

Step 8 – Attack Automation

AlarmPWN – intercepting a code and the button pressed

Step 8 – Attack Automation

Privasec
Governance and Information Security Partners

The app can be deployed on a Raspberry Pi for discrete, code interception – with or without technical skills.

We can plant the device instead of waiting/camping near the target site.

Leave/hide and wait for the Raspberry Pi to intercept the code

(Maybe even add a sim card so you can be notified when it's complete?)

Step 8 – Attack Automation

This is what the attack looks like in practice:

Reflection

&

Usefull Tips

Reflection

Where they went wrong:

- No rolling codes (the codes / device IDs were static)
- No encryption
- No CRC / validation
- Device IDs too small (brute-force / collision attacks)
- No attempts to detect jamming of signals from sensors

Where they went right:

- The alarm CAN send text message alerts when system is disarmed…

Reflection

Privasec
Governance and Information Security Partners

Where they went wrong:

- No rolling codes (the codes / device IDs were static)
- No encryption
- No CRC / validation
- Device IDs too small (brute-force / collision attacks)
- No attempts to detect jamming of signals from sensors

Where they went right:

- ~~The alarm CAN send text message alerts when system is disarmed…~~

  nope… It only supports 2G

Tips for Consumers

Privasec
Governance and Information Security Partners

How can you distinguish between products with security and without security?

- Not by looking at it, there are no external physical appendages to lookout for
- You could try asking the vendor (but most lack the technical understanding to be able to address your concerns)
- Prioritise products with a published FCC ID for the device or internal components
  - FCC submissions include specifications for the chips and are accessible online
  - Open the specification documents and look for mentions of:
    - Rolling/Hopping codes
    - Encryption
    - Time based codes
    - Frequency hopping (though this is only a mild barrier to penetration and is easily thwarted)

Make an informed decision and vote with your $$$

Any questions?

Privasec
Governance and Information Security Partners