My Arduino can beat up your hotel room lock

- Cody Brocious

trappedorbit research

Intro

- This talk is all about the Onity HT lock system for hotels
  - Over 4 million locks are installed in hotels
  - On the market since 1993
  - Every one is vulnerable

Intro

If you've stayed at a hotel, you've probably seen this lock

trappedorbit research

Design

- Primary components:
  - Encoder – Makes keycards, loads data into the Portable Programmer
  - Portable Programmer (PP) – Loads data into the lock, opens locks
  - Lock – In this talk, we'll be focused on standard guest room door locks

Design

- Sitecodes are 32-bit unique values that identify a property (hotel)
  - All equipment in the hotel knows it
  - Used primarily as an encryption key
  - Hidden, even from property owners

Portable Programmer

- The portable programmer does the following
  - Initialize – Load data into lock for the first time
  - Update – Update the time and data in the lock
  - Test – Shows diagnostic data about the lock
  - Read openings – Reads the audit report from the lock
  - Open – Opens the lock

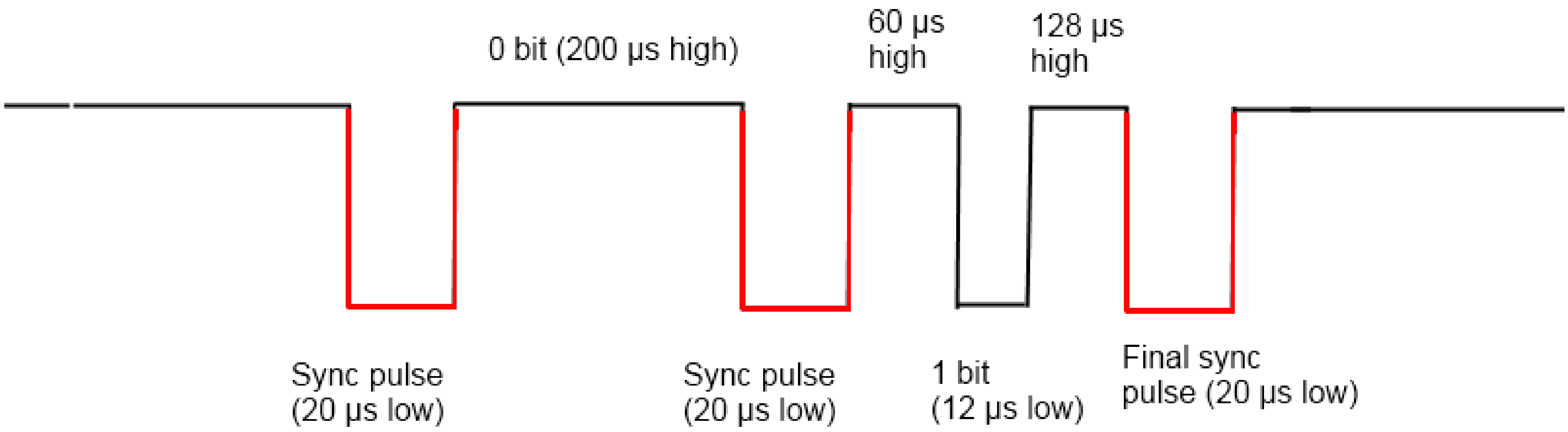
Lock communications

- The PP uses a DC barrel-type connector
  - It attaches to the bottom of the front face of the lock
  - The port is accessible without removing any hardware
- Communication happens over a one-wire protocol with the other being a shared ground

Lock communications

- The master (PP) drives the communication
  - Sends pulses at regular intervals while communicating
  - If one side wants to transmit a bit, it's done by pulling the line low between those pulses
    - That indicates a 1 bit

Lock communications



- This is a case of the lock sending data to the master
  - A zero and one, specifically
  - Red pulses are from the master, the black pulse is from the lock

Hardware

- To communicate with the lock physically, you'll need the following
  - An Arduino or other microcontroller
  - A 5.6k pull-up resistor from the 3.3v line to your data line
  - The DC barrel plug to physically mate with the lock
- This, depending on the board you get, can be $20 or less from Radioshack

But what can we *do*?

- There are a few key commands in the protocol
  - Reading memory
    - Given an address, the lock will send back 16 bytes of memory from that point
  - Opening the lock
    - Given the sitecode for the property, the lock will open

But what can we *do*?

- But it'd be crazy to just let anyone do this
  - If you can read the memory, the keys to the castle are there
- How do we deal with authentication?

Authentication

. . .

trappedorbit
research

Authentication

- Reading memory requires *no authentication*
  - Send it an address, it sends you memory
  - That's it

trappedorbit research

Memory

- Knowing how to read memory is irrelevant if you don't know what to read
  - But every guest room lock has their data at the same addresses
    - Exterior entry doors are different, but you can detect the type and act based on that

trappedorbit research

Memory

- The most obvious piece of data is the sitecode
  - Given that, you can decrypt or encrypt your own cards
  - Or you can go the direct route, and just use it with the open command on the lock

Open command

- All you need is the sitecode
  - We got that from memory
- Complete time for reading the memory and opening the lock is about 200 milliseconds
  - This can be longer if you need to try different addresses, due to supporting multiple door types
- Creates an entry in the audit report that shows the PP having been used to open the lock
  - But it doesn't alter any data on the lock or inhibit normal functioning

Memory

- But there's more:
  - Guest code
    - Make your own guest card for the door
  - Master codes
    - Make copies of any master card programmed into the lock
    - This won't necessarily get you into every lock at the property
      - Not all masters are assigned to all doors

trappedorbit research

Programming cards

- Also in memory is the programming card code
- Truly magical cards
  - One code is loaded into every lock at the property
  - Used for cases where the encoder is out of service
    - A programming card is put into the lock
    - Then a 'spare' card is put into the lock
      - That spare card is now the actual guest card
    - Hotels keep dozens of these on file in case of front desk system issues
  - We can read this code from memory and make a skeleton key

trappedorbit research

Card cryptography

- As mentioned previously, the sitecode is *the* crypto key for cards
  - As a reminder, this is only 32-bit
  - A naïve implementation of the crypto algorithm gives you 2 million card encrypts or decrypts per second trivially
  - That means that trying every sitecode on a key would take about 35 minutes on a normal desktop using one core
    - If you wanted to do it in a minute, it would cost less than a dollar on Amazon EC2

Card cryptography

- Brute force is obviously viable
- The crypto algorithm is proprietary
  - It works in a linear fashion from beginning to end
  - Each step is a rotate and an XOR
  - Key material is poorly distributed
- If you know plaintext in the card, it's trivial to determine the sitecode used to encrypt it

Card cryptography

- Let's look at the card format
  - 16-bit ident value
    - Identifier for the door combined with the card copy field
  - 8-bit flags byte
  - 16-bit expiration date
  - 8-bit authorizations byte
  - 24-bit zeros
  - 24-bit code key value

trappedorbit research

Card plaintext

- Ident values may be predictable
  - We do know the card copy field that takes up a few of the lower bits of the ident field
  - And when the doors are added to the encoder, they're added in a specific order and spaced out logically
    - Very possible that this could be guessed, though validating it is next to impossible without outside info
- We can't know the code key value
  - 24-bit space, effectively randomly distributed
- But we know the expiration date and the zero bytes

Card plaintext

- If you get two cards when you check into a hotel
  - The ident value will be separated by one
- If you get a card for a room, then get a new card for it (e.g. lost the old one)
  - The code key value will be incremented by one

Card plaintext

- All of this gives us enough plaintext to determine the sitecode
  - Read in a couple cards with known properties
  - Bruteforce the sitecode and decrypt the cards
    - Check to see that those properties are upheld in the plaintext
- Given the properties of the crypto, full brute force should not be necessary
  - Should be able to figure out which bits of the sitecode are correct and which are not

Audit reports considered harmful

- Given all the vulnerabilities present in this system, the audit report is unquestionably untrustworthy
  - And this is all assuming that it isn't also possible to write to memory, in addition to reading

Demonstration

Opening a lock with an Arduino

trappedorbit
research

Release

- The paper is being released in a beta form
  - It will be available and updated at http://daeken.com/
  - Full details on the opening device, as well as protocol specifications, crypto code, etc are included
  - There's 3 years of work to release
    - This talk only shows a tiny section of it
    - The paper includes a lot already and will get bigger and bigger as time goes on

Mitigation

- At the moment there's no mitigation, but there are possibilities
  - Direct memory access
    - Redesign lock to provide safe interface for programming
    - Update portable programmer to be compatible
  - Cryptography
    - Switch to a larger key and industry standard algorithm like AES
    - Update encoders and locks

trappedorbit research

Mitigation

- Biggest impediment to mitigation is that the locks are not upgradeable
  - At the very least, the circuit boards in over 4 million locks would have to be replaced
- The PP is not much better off, but the EPROM can be changed
- Given the substantial changes that would be required, it would be impossible to replace the locks without replacing all of the equipment at the front desk as well
  - And all of the locks at a property would have to be replaced at the same time
  - This all adds up to a very substantial cost

Future work

- There's a lot of work still to be done
  - Cryptography
    - A cryptographer would likely be able to make significant progress towards simplifying and breaking the crypto algorithm beyond what was presented here
  - Protocol
    - It is believed that the PP initializes/updates the lock via direct memory writes, but this is not reversed
  - Memory
    - The complete memory maps of all of the locks are not available
  - CT locks
    - The Onity CT (commercial) locks may be vulnerable to the same sort of issues detailed here, but this has not been tested

Recap

- Arbitrary memory access
  - Gives us the sitecode
    - Open the lock instantly
    - Or create cards to open the other locks at the property
      - Including the programming card skeleton keys
- Completely unauthenticated
- Cryptography is broken
  - Tiny keyspace
  - Proprietary algorithm leaks data

trappedorbit research

Questions?

trappedorbit
research