What Is This Talk About

27Mhz keyboards & analyzing RF Signals

Design and build an “Encryption Validation Device” aka known as keystroke sniffer with some special features
Warning!

• Verify the security of someone else's data transmission without permission could send you to jail in some countries :-)

![Warning Image](image-url)
About Us

Thorsten Schroeder & Max Moser
Dreamlab Technologies AG, Switzerland

remote-exploit.org

<<back|track
Why Do We Continue Hacking This Stuff?

• Full disclosure
• POC technique was not usable in practice
  • Was neither portable nor handy
  • Depending on certain drivers and software (e.g. Sound card, filters )
• Going open-source
• Finishing the job :-(
Technical Background
Involved Components
Receiver

- Receives, demodulates and processes the RF signal
- Most implementation are using dedicated receiver/transceiver chips to accomplish the demodulation task
- Micro-controller decodes data signal and generate the relevant USB-HID or Key scan-codes
- Persistently stores connection and encryption details
Microsoft Receiver

Micro-Controller

EEPROM

Receiver Chip
Sniff The Signal

- RF scanner
- GNUradio / USRP
- Tap the original receiver
- Build your own receiver
Visualize The Signal

- Sound-card + Audacity (Soft-scope)
- (USB) Oscilloscope
Identify Signal Encoding

• It is important to know how the binary data is modulated onto a signal

• Most communication is using standardized ways to encode binary data

• There are a lot of well-known methods available and even more variations of each

• NRZ, Miller and Manchester are some of the most common ones

• All keyboards we have analyzed were Miller encoded
Miller

- Aka delay encoding - RFID, Serial RF protocols
- So a typical Miller signal has same signal level for a length of 1 bit period, 1.5 bit period and 2 bit period of time
Sequence Patterns

- Look at signals to find sequence boundaries
- Do they repeat per keystroke?
- Are they similar/identical when using different keyboards?
<table>
<thead>
<tr>
<th>Key</th>
<th>State</th>
<th>Keyboard ID</th>
<th>Keystroke</th>
<th>Key State</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(down) Keyb 1</td>
<td>000000100</td>
<td>10001001001</td>
<td>00000011110</td>
<td>1</td>
<td>00000</td>
</tr>
<tr>
<td>a(down) Keyb 2</td>
<td>000000100</td>
<td>100111001111</td>
<td>0000011110</td>
<td>1</td>
<td>0001000</td>
</tr>
<tr>
<td>a(up) Keyb 1</td>
<td>000000100</td>
<td>10001001001</td>
<td>0000011110</td>
<td>0</td>
<td>00000</td>
</tr>
<tr>
<td>a(up) Keyb 2</td>
<td>000000100</td>
<td>100111001111</td>
<td>0000011110</td>
<td>0</td>
<td>0001000</td>
</tr>
<tr>
<td>b(down) Keyb 1</td>
<td>000000100</td>
<td>10001001001</td>
<td>0000000101</td>
<td>1</td>
<td>0101</td>
</tr>
<tr>
<td>b(down) Keyb 2</td>
<td>000000100</td>
<td>100111001111</td>
<td>0000000101</td>
<td>1</td>
<td>0100000</td>
</tr>
<tr>
<td>b(up) Keyb 1</td>
<td>000000100</td>
<td>10001001001</td>
<td>0000000101</td>
<td>0</td>
<td>0101</td>
</tr>
<tr>
<td>b(up) Keyb 2</td>
<td>000000100</td>
<td>100111001111</td>
<td>0000000101</td>
<td>0</td>
<td>0100000</td>
</tr>
</tbody>
</table>
Data Details Logitech 2

- Unencrypted per default
- Logitech drivers for windows are able to enable encryption
- Secure connect (new tech) has encryption on per default (Fixed identifier on RFID)
- Decoding not implemented in Keykeriki right now, but ready to be ported from first POC codes, just a value table lookup

Snipplet from lookup table

```
"0000111101"=>" ",
"0001110001"=>"[ENTER]\n",
"0001101001"=>"[SHIFTL]",
"0000110101"=>"[SHIFTR]",
"0000011101"=>"[CTRLL]",
"0000000011"=>"[CTRLR]",
"0000111011"=>"[WINL]",
"0001111011"=>"[WINR]",
"0001011101"=>"[ALT]",
"0001111101"=>"[ALTGR]",
"0000000111"=>"[WINMENU]",
"0001110010"=>"[TAB]",
"0001101110"=>"[CAPSL]",
"0000011110"=>"a",
"0000000101"=>"b",
"0000111001"=>"c",
"0000111110"=>"d",
"0000101010"=>"e",
"0001111110"=>"f",
"0000000001"=>"g",
"0001000001"=>"h",
"0001111010"=>"i",
"0001000101"=>"j",
"0001110001"=>"k",
"0000010001"=>"l",
"0000100101"=>"m",
```


Motivation / Threats

• „I forgot my bank officers password!“
• „Screen Sharing“
• Seriously…

• Many public accessible offices with computers in front of customers are using wireless equipment to reduce the rat‘s nests onto the desk
• Malicious people might want to collect passwords, CC numbers, PII, etc
• Access to those desks is easy…
Getting Data Access

- Extending range using an antenna & amplifiers
- Get as close to the sender (keyboard) as possible
  - Souvenirs (Concealments)
  - Duck-tape
  - …
• No one throws them away
• Or just use some duck-tape and stick it somewhere
Getting Data Access

• Or simply make an appointment with the target person and keep it in your jacket
Design Considerations

• External antenna connector
• Small, Stand-alone / battery powered
• Platform / PC independent
• Data logging/storage desired
• Flexible interfacing with HW/SW extensions
The µC

• Micro-controllers are small, cheap, handy, easy to use, less power consuming, ...

• Programming is very easy (C,ASM)

• Hardware support for many useful items like detecting edges, timer, communication via different HW bus systems (I²C/TWI, SPI, USART) etc

• Fast enough to compete with the user’s typing skills (*)
(*) Timing

- Well, we have a lot stuff to be processed, we’ll discuss some problems and limitations later
Tasks

- 1. Capture keystrokes
- 2. Decode keystrokes
- 3. Capture or crack crypto keys
- 4. Decrypt data and translate HID codes
- 5. Process and store or forward decrypted data
Task 1: Capture

- We can use different approaches to capture the signals using a micro-controller:
  - Using a input capture interrupt which detects falling or raising edges and interrupt code execution
  - Using two timers, we can act like an oscilloscope and measure times between edges as well as detect edges
Task 2: Decoding

• Once we captured the raw, digital signal, we have to decode it properly

• It’s Miller Time!!

• But… Microsoft did some modifications to the miller decoding standard (surprise!)
Microsoft Vs. Miller

- Binary decoding depends on the value of the last decoded binary value. Depending on the variation it starts with “0” or “1”

<table>
<thead>
<tr>
<th>Standard Miller</th>
<th>Microsoft’s Way</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Duration</strong></td>
<td><strong>Binary decoding</strong></td>
</tr>
<tr>
<td>1</td>
<td>1 bit (same as lastbit)</td>
</tr>
<tr>
<td>1.5</td>
<td>1 bit “1” when lastbit == 0 2 bit “00” when lastbit == 1</td>
</tr>
<tr>
<td>2</td>
<td>2 bits with values “01”</td>
</tr>
</tbody>
</table>
More Pitfalls

• The packet sequence boundaries are different depending on the type speed of the users typing-skills/speed!
Task 3: Crack

- Store raw data and perform offline-bruteforce
- Gather current crypto key (XOR) in real time
  - Capture within keyboard SYNCH Sequences
- Perform an On-The-Fly Cryptoanalysis and exploit design issues in the communication protocols
On-The-Fly Crypto Analysis??

- Freaking simple… For this release we followed and implemented two approaches:
  - Meta keys are unencrypted
    - If a Shift key is pressed, we go back in our data buffer and assume last typed key is a whitespace
    - Than we XOR the last data byte with the HID code for Whitespace and assume we have got the right key
    - We check if the key is correct by applying an XOR using the key to the previous byte. If the Plaintext equals an HID code of a sentence mark we assume to have the right key for the session
Crypto Analysis (Cont‘d)

• Second approach is to check whether a key is pressed three times in a row.

• If so, we assume it was the HID code for the character ‘w‘ (i guess you are aware of the term „www“ ;-)  

• After XORing the triple data byte with the HID code for ‘w‘ we assume to have the current XOR key.

• We check if the key is valid by XORing the key with the next cipher-byte – if the result is the HID code for the character ‘.’ we have successfully gathered a session key.
Task 4: Decryption & Translation

- After we have successfully gathered the Crypto Key, we can optionally perform an On-The-Fly decryption and translation of data.
- Captured data is stored in raw mode as well as in deciphered mode.
- Decrypted data can be used to be displayed on a small LCD screen or on a computer (via USB).
Task 5: Process & Store

- Data is written to the SD Card in Raw and Plaintext.
- Text data can be transferred to an LCD display.
- Data can be sent to a computer via USB cable (no special drivers necessary, our device works well with default FreeBSD, Mac OSX, Windows XP, Vista, Linux, maybe IPhone(*) …) 😊
- We can also send data (buffered or unbuffered) via USART to a cellphone which has an SMS flat-rate or GPRS.
Our HW Solution
Atmel Atmega 64

• Pro’s
  • Cheap
  • Flexible, easy to handle, well known
  • Built-in features meet our design considerations
  • Pin & footprint compatible with larger microcontrollers when more memory is required
Atmel Atmega 64

- **Con’s**
  - 8-bit only
  - Limited amount of resources
  - Small pitch (TQFP 64) makes it difficult for beginners to handle
TI TRF7900 Receiver Chip

- **Pro’s**
  - All in one IC solution
  - Can handle all commonly used frequencies
  - Able to be configured using I2C bus
  - Built-in Signal Strength Measurements (RSSI)
  - Dual channel capable
  - Relatively cheap
  - Low power consumption
TI TRF7900 Receiver Chip

- **Con’s**
  - Differential antenna input with 5kOhm input impedance
  - Public documentation could be better
  - 5V only
FTDI FT232RL

• Pro’s
  • USB to RS 232 converter
  • Driver included within all major os ‘s
  • Supports USB bus powered design
  • Integrated 3.3v regulator output
  • Bitbang modus
  • Open-source code available from vendor

• Con’s
  • Relatively expensive
SD Card

- Pro’s
  - Cheap
  - Larger storage capacity
  - Easy to use compared to other storage types
  - Requires very few external components
  - Standard SPI bus used for communication

- Con’s
  - 3v only!
External Antenna Connector vs. PCB Loop Antenna

- Pro’s
  - Arbitrary antennas
  - Larger receiving range
  - Smaller

- Con’s
  - Directional antennas would be very very very large (27Mhz ≈ 11 Meter)
  - Expensive
Original Receiver Vs. Keykeriki
Worth To Be Mentioned Pitfalls
Problem Error Correction

- Input capture is not optimal for error correction
- Error propagation to later part of decoding
- Errors in Start/Stop patterns are hard to distinguished from noise
Problem Antenna Matching

- Design recommendations available for differential loop pcb antennas
- TI: “Unfortunately we did not make any such design”
- TRF7900 Chip got 5kOhm input impedance
- Common CB (11 Meter) antennas have 50Ohm
- Hughe gap! Small solution is needed
- Proper balun + match difficult in that size footprint
Current Antenna Matching
Future

- Keystroke injection
- Range extension using amplification
- Port Logitec decoding to Keykeriki
- Automatic channel-hopping, Kismet-NG Plugin?
- Analysis of Logitec encryption
- Decoding for other devices
- Inspection of 2.4 Ghz devices
That’s It!

• Our white-paper “27_Mhz_keyboard_insecurities.pdf”
• http://www.remote-exploit.org
• Yes, we are doing complete sets
• Price is not clear jet. Guess will end up somewhere around ~30-40 Euros