atlas0fd00m / **rfcat**

RfCat - swiss-army knife of ISM band radio

543 commits | 13 branches | 3 releases | 11 contributors | View license

Branch: master | New pull request | Find file | Clone or download

atlas0fd00m print out the SerialException — Latest commit 109f887 on 23 Apr

CC-Bootloader	print out the SerialException	7 months ago
etc/udev/rules.d	whitespace cleanup	10 months ago
firmware	Merge pull request #34 from roaldnefs/feature/py3compat	9 months ago
rflib	missed a spot.	7 months ago
tests	unittest enhancements. my own version of a fix for the bytes[] problem.	7 months ago
vstruct	RfCat Server for Metasploit Hardware Bridge	3 years ago
.gitignore	add `firmware/.revision` to gitignore	9 months ago
.vimloader	Remove cc1111client.py and update references	10 months ago
FAQ	RfCat Server for Metasploit Hardware Bridge	3 years ago
LICENSE	RfCat Server for Metasploit Hardware Bridge	3 years ago
LICENSE.CC-Bootloader	RfCat Server for Metasploit Hardware Bridge	3 years ago
MANIFEST	show_banner fix-addendum	last year
MANIFEST.in	fixed setup.py strip list errors	2 years ago
README.immesniff	RfCat Server for Metasploit Hardware Bridge	3 years ago
README.md	documentation fixes	7 months ago
README.msfrelay	Added README Documentation for msfrelay	3 years ago
README.nonroot	RfCat Server for Metasploit Hardware Bridge	3 years ago
README.rst	Merge pull request #34 from roaldnefs/feature/py3compat	9 months ago
TODO	RfCat Server for Metasploit Hardware Bridge	3 years ago
package.sh	RfCat Server for Metasploit Hardware Bridge	3 years ago
revision.sh	replace hg revision script with a rough git equiv	10 months ago
rfcat	Update print statement to function in rfcat	10 months ago
rfcat_msfrelay	Add the required future import in rfcat_msfrelay	9 months ago
rfcat_server	Update print to function in rfcat_server	9 months ago
setup.cfg	fixed setup.py strip list errors	2 years ago
setup.py	Add dependency on the future package	9 months ago

README.md

Welcome to the rfcat project

Table of Contents

GOALS

The goals of the project are to reduce the time for security researchers to create needed tools for analyzing unknown targets, to aid in reverse-engineering of hardware, and to satiate my rf lust.

REQUIREMENTS

RfCat currently requires Python 2.7. the only suspected incompatabilities with Python 3.x are minimal, mostly print("stuff") versus print "stuff" and other str/bytes issues.

Other requirements

- python usb
- libusb - should be able to work with either 1.x or 0.1 versions. please let us know if you run into issues.
- pyreadline (especially for Windows)
- PySide2 (for Spectrum Analyzer GUI): (Ubuntu 18.10+: python-pyside2) $ sudo pip install PySide2

Build requirements

- Make
- SDCC (code is kept up-to-date with the current Ubuntu release, as of this writing: 3.4.0+dfsg-2ubuntu1)

DEVELOPMENT

New development efforts should copy the "application.c" file to "appWhateverMyToolls.c" and attempt to avoid making changes to other files in the repo if at all possible. that is only a recommendation, because future bug-fixes in other libraries/headers will go much more smoothely for you.

Gotchas

A couple gotchas to keep in mind while developing for the cc1111

- The memory model includes both "RAM" and "XDATA" concepts, and standard RAM variables and XDATA variables have different assembly instructions that are used to access them. this means that you may find oddities when using a function written for XDATA on a standard RAM variable, and vice-versa.
- Variables should be defined in a single .c file, and then "externs" declared in a .h file that can be included in other modules. this is pretty standard for c programs, but both this and the previous point caused me difficulties at some points, and i found myself unsure what was causing my troubles.
- RAM memory is not cheap. use it sparingly.
- You need to set the radio into IDLE mode before reconfiguring it
- You need to set the radio into TX mode *before* writing to the RFD register (firmware) as it is a 1-byte FIFO.

INSTALLING ON HARDWARE

Installing and getting up to speed with rfcat...

First things first. Using rfcat requires that you either use the python client in root mode (sudo works well), or configure udev to allow non-root users full access to the dongle. you must also have one of the supported dongles flashed with the necessary application firmware.

allowing non-root dongle access

```
sudo cp etc/udev/rules.d/20-rfcat.rules /etc/udev/rules.d
sudo udevadm control --reload-rules
```

This tool is created, maintained, and used primarily on linux. make and sdcc must be installed for creating new firmware and some of the helper functions we provide through make.

supported dongles

- YARDStick One
- cc1111emk (aka DONSDONGLES)
- chronos watch dongle (aka CHRONOSDONGLE)
- imme (limited support for both IMME and IMMEDONGLE)
  - imme dongle is not really usable as of 1/31/2012

GoodFET

```
        ---------------------------------
        |                         1  2 |
        |                         3  4 |
   ------                         5  6 |
   | USB                          7  8 |
   ------                         9 10 |
        |                        11 12 |
        | GoodFET                13 14 |
        ---------------------------------
```

Chronos Dongle

```
        ---------------------------------
        |                              |
        |            RST 1  2 TP      ------
        |            GND 3  4 VCC      USB |
        |         DC/P2_2 5  6 DD/P2_1 ------
        | Chronos                      |
        ---------------------------------

           GoodFET            Chronos
             PIN                PIN

              1 <----- DD -----> 6
              2 <----- VCC ----> 4
              5 <----- RST ----> 1
              7 <----- DC -----> 5
              9 <----- GND ----> 3
```

EMK Dongle

```
        ---------------------------------
        | 2 4 6 8 10   2 4 6 8 10      |
        | 1 3 5 7 9    1 3 5 7 9       |
        |-TEST-PINS----DEBUG-PINS------|
        |                              |
   ------                              |
   | USB                               |
   ------                              |
        | Don's Dongle (EMK)           |
        ---------------------------------

           GoodFET             EMK
             PIN            DEBUG PIN

              1 <----- DD -----> 4
              2 <----- VCC ----> 2
              5 <----- RST ----> 7
              7 <----- DC -----> 3
              9 <----- GND ----> 1
```

YARD Stick One

Pogo pads on the back are clearly marked, but if you want to use the header...

```
        ----------------------------------------
        | YARD Stick One      2 4 6 8 10 12 14  |
        |                     1 3 5 7 9  11 13  ------
        |                                        USB |
        |                                       ------
        |                                        |
        ----------------------------------------

           GoodFET           YARD Stick One
             PIN                  PIN

              1 <----- DD -----> 1
              2 <----- VCC ----> 2
              5 <----- RST ----> 5
              7 <----- DC -----> 7
              9 <----- GND ----> 9
```

Your build environment

Intended development model is using a GoodFET although one of our developers uses the chipcon debugger from Texas Instruments.

- install sdcc
- install make
- make sure both are in the path
- cd into the `rfcat/firmware/` directory
- `make testgoodfet` will read info from your dongle using the GoodFET. you should see something like:

```
SmartRF not found for this chip.
Ident   CC1111/r1103/ps0x0400
Freq         0.000 MHz
RSSI    00
```

- `make backupdongle` will read the current firmware from your dongle to the file `.../bins/original-dongle-hex.backup` . (`make restoredongle`) to revert to the original firmware.
- `make clean installRfCatChronosDongle` will clean, build, and install the RfCat (`appFHSSNIC.c`) firmware for a Chronos dongle.
- `make clean installRfCatDonsDongle` will clean, build, and install the RfCat (`appFHSSNIC.c`) firmware for a cc1111emk.
- `make clean installimmesnifffw` will clean, build, and install the RfSniff firmware for the IMME girls toy from girltech

INSTALLING WITH BOOTLOADER

Dependencies: Fergus Noble's CC-Bootloader (slightly modified). For your convenience, hex files are provided in the CCBootloader sub-directory in firmware.

Source can be found here

- https://github.com/AdamLaurie/CC-Bootloader

Which is branched from here

- https://github.com/fnoble/CC-Bootloader

To install

We need permanent symlinks to the USB serial devices that will communicate with the CHRONOS, DONSDONGLE or YARDSTICKONE bootloader when required. If you haven't done this step already (see above), then run:

```
sudo cp etc/udev/rules.d/20-rfcat.rules /etc/udev/rules.d
sudo udevadm control --reload-rules
```

Next, your user must have read/write access to the dongle when it shows up to the operating system. For most Linux distros, this means you have to be a member of the "dialout" group.

To prepare your dongle for the first time, you'll need to hook up your debugger as described above and do:

(install `rfcat_bootloader` from the CC-Bootloader subdirectory to somewhere on your execution path)

`cd firmware`

for EMK/DONSDONGLE: `make installdonsbootloader`

for CHRONOS: `make installchronosbootloader`

for YARDSTICKONE: `make installys1bootloader`

now unplug the debugger and plug in your USB dongle.

If you have just installed the bootloader, the dongle should be in bootloader mode, indicated by a solid LED.

If you are re-flashing a dongle that is already running rfcat, the Makefile targets will force it into bootloader mode for you, but you can manually put it into bootloader mode either by holding down the EMK/DONS button as you plug it into USB (on the CHRONOS or YARDSTICKONE jumper P2_2/DC to GROUND), or by issuing the command `d.bootloader()` to rfcat in interactive mode (`rfcat -r`), or by issuing the command `rfcat --bootloader --force` from the command line.

Once you have a solid LED, or if you're running an rfcat dongle, you can do the following:

`cd firmware`

for EMK/DONSDONGLE:

- `make installRfCatDonsDongleCCBootloader`

for CHRONOS:

- `make installRfCatChronosDongleCCBootloader`

for YARDSTICKONE:

- `make installRfCatYS1CCBootloader`

The new version will be installed, and bootloader exited.

Installing client

Dependencies

- python-usb
- libusb

Install rfcat onto your system. on most linux systems, this will place `rfcat` and `rfcat_server` in `/usr/local/bin/` and `rflib` into `/usr/*/lib/python2.x/dist-packages`

Installation

- cd into the rfcat directory (created by unpacking the tarball or by hg clone)
- sudo python setup.py install
- I highly recommend installing `ipython`
  - For deb/ubuntu folk: `apt-get install ipython`

Using rfcat

If you have configured your system to allow non-root use:

- type "rfcat -r" (if your system is not configured to allow non-root use, prepend "sudo" or you must run as root) you should have now entered an interactive python shell, where tab-completion and other aids should make a very powerful experience i love the raw-byte handling and introspection of it all.
- try things like:
  - d.ping()
  - d.discover()
  - d.debug()
  - d.RFxmit('blahblahblah')
  - d.RFrecv()
  - print(d.reprRadioConfig())
  - d.setMdmDRate(19200) # this sets the modem baud rate (or DataRate)
  - d.setPktPQT(0) # this sets the preamble quality threshold to 0
  - d.setEnableMdmFEC(True) # enables the convolutional Forward Error Correction built into the radio

while the toolset was created to make communicating with <ghz much easier, you will find the cc1111 manual from ti a great value. the better you understand the radio, the better your experience will be. play with the radio settings, but i recommend playing in small amounts and watch for the effects. several things in the radio configuration settings are mandatory to get right in order to receive or transmit anything (one of those odd requirements is the TEST2/1/0 registers!)

If you watched any of my talks on rfcat, you will likely remember that you need to put the radio in **IDLE state** before configuring. (I said it three times, in a row, in different inflections).

However, you will find that I've done that for you in the client for most things. The only time you need to do this yourself are: * If you are doing the changes in firmware * If you are using the "d.poke()" functionality * if you use "d.setRFRegister()", this is handled for you * `use d.setRFRegister()`

Epilogue

Other than that, hack fun, and feel free to share any details you can about successes and questions about failures you are able!

@ and the rest of the development team.