Hacking IoT with Emulator

Reverse Engineering IoT devices with Qiling Framework

Global Security Conference
Feb 2020

KaiJern LAU, kj -at- qiling.io
NGUYEN Anh Quynh, aquynh -at- gmail.com
huitao, CHEN null -at- qiling.io
TianZe DING, dliv3 -at- gmail.com
BoWen SUN, w1tcher.bupt -at- gmail.com
Tong YU, spikeinhouse -at- gmail.com

About xwings

SECURITY.JD.COM

Hoping making the world a better place

- Lab Director / Founder
- Blockchain Research
- IoT Research

hackersbadge.com

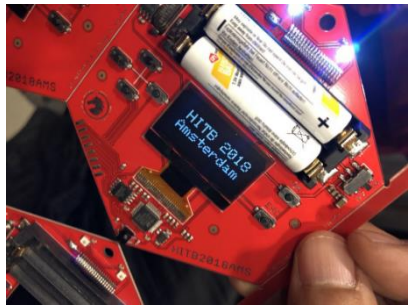
Electronic fan boy, making toys from hacker to hacker

- Reversing Binary
- Reversing IoT Devices
- Part Time CtF player

Qiling Framework

Cross platform and multi architecture advanced binary emulation framework

- https://qiling.io
- Lead Developer
- Founder



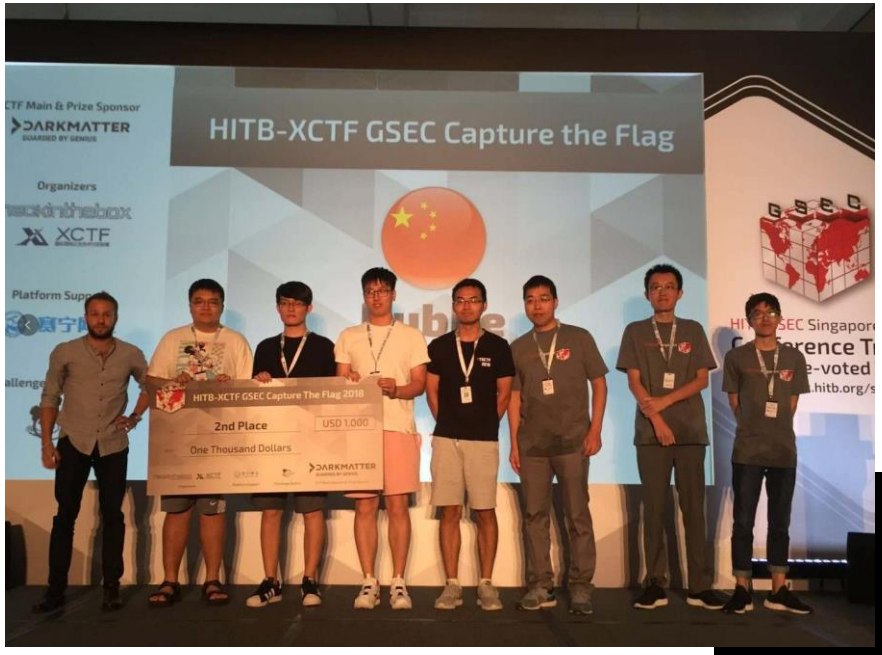
- 2005, HITB CTF, Malaysia, First Place /w 20+ Intl. Team
- 2010, Hack In The Box, Malaysia, Speaker
- 2012, Codegate, Korean, Speaker
- 2015, VXRL, Hong Kong, Speaker
- 2015, HITCON Pre Qual, Taiwan, Top 10 /w 4K+ Intl. Team
- 2016, Codegate PreQual, Korean, Top 5 /w 3K+ Intl. Team
- 2016, Qcon, Beijing, Speaker
- 2016, Kcon, Beijing, Speaker
- 2017, Kcon, Beijing, Trainer
- 2018, KCON, Beijing, Trainer
- 2018, Brucon, Brussel, Speaker
- 2018, H2HC, San Paolo, Brazil, Speaker
- 2018, HITB, Beijing/Dubai, Speaker
- 2018, beVX, Hong Kong, Speaker
- 2019, Defcon 27, Las Vegas, Speaker
- 2019, HITCON, Taiwan, Speaker
- 2019, Zeronight, Russia, Speaker

- MacOS SMC, Buffer Overflow, suid
- GDB, PE File Parser Buffer Overflow
- Metasploit Module, Snort Back Oriffice
- Linux ASLR bypass, Return to EDX

About NGUYEN Anh Quynh

- Nanyang Technological University, Singapore
- PhD in Computer Science
- Operating System, Virtual Machine, Binary analysis, etc
- Usenix, ACM, IEEE, LNCS, etc
- Blackhat USA/EU/Asia, DEFCON, Recon, HackInTheBox, Syscan, etc
- Capstone disassembler: http://capstone-engine.org
- Unicorn emulator: http://unicorn-engine.org
- Keystone assembler: http://keystone-engine.org

About Dliv3/w1tcher/Null/Sp1ke



THE SHEPHERD LAB 京东牧者安全实验室

DUBHE HACKING SECURITY

Rest of the team members are from JD.COM theshepherdlab and Dubhe CTF team

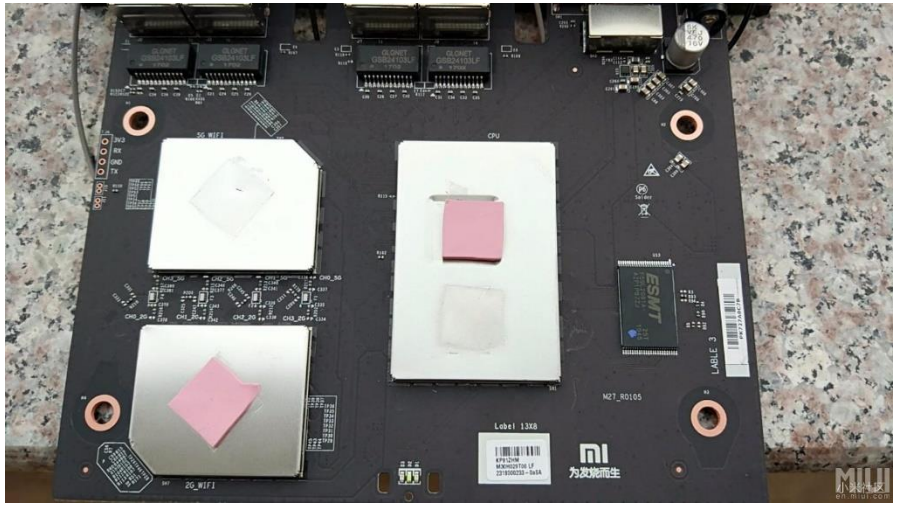
Internet of Things

What is IoT

- IoT
  - Camera
  - Air-con
  - TV
  - FAN
  - Heater
  - Fridge
  - Watch
  - Lock
  - Security
  - Kitchen
  - Phone

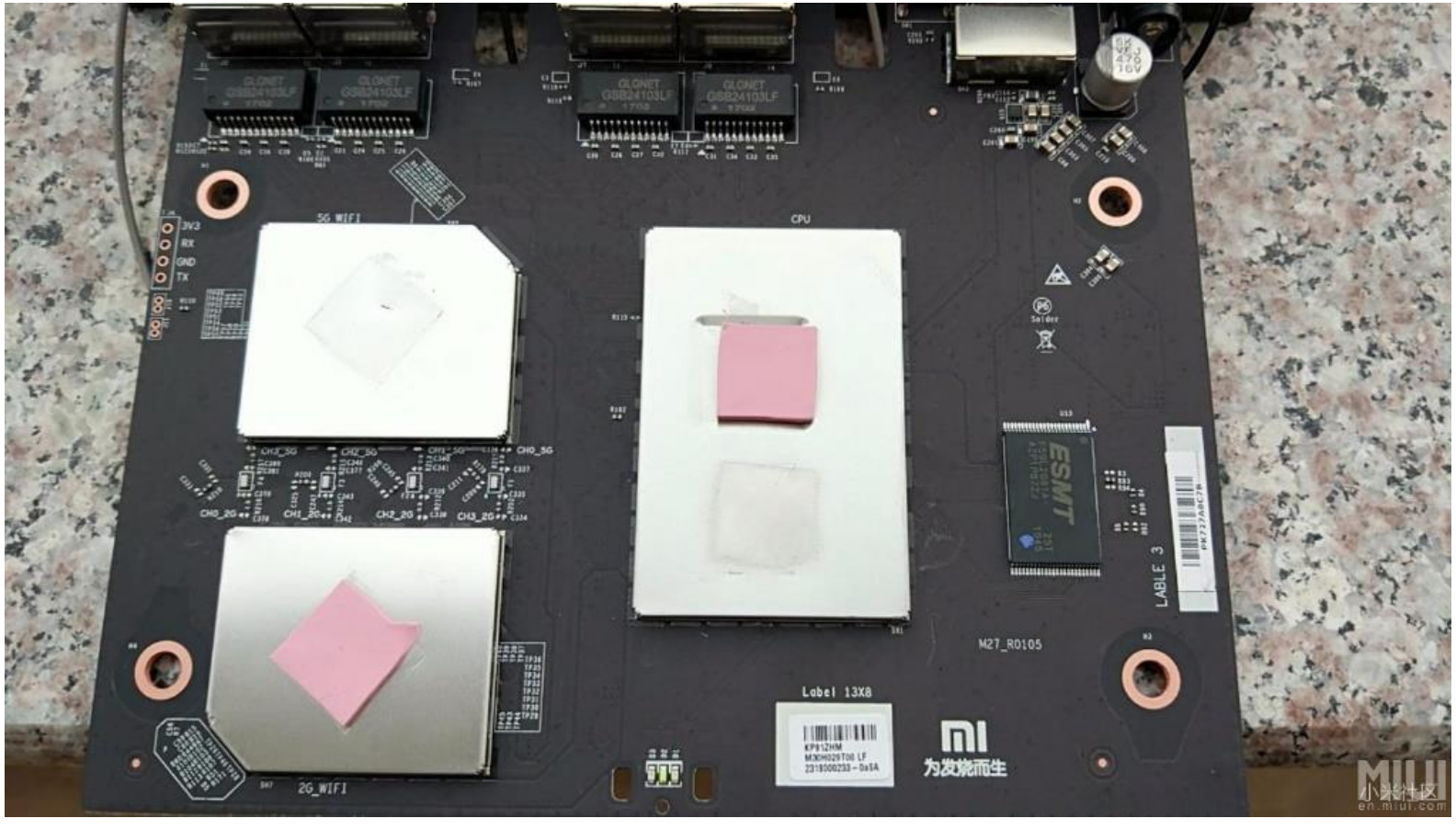
Deep In The Heart

Hacker’s OpenBox



What In IoT

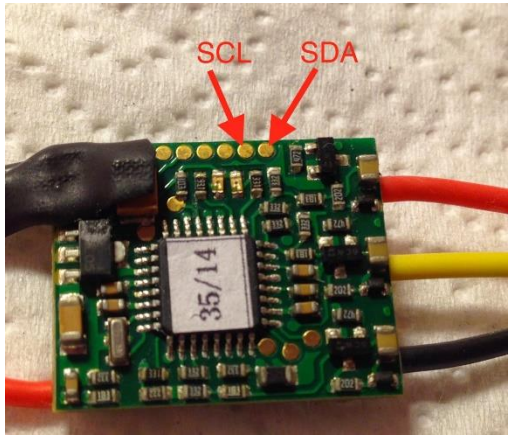
Hardware Identification

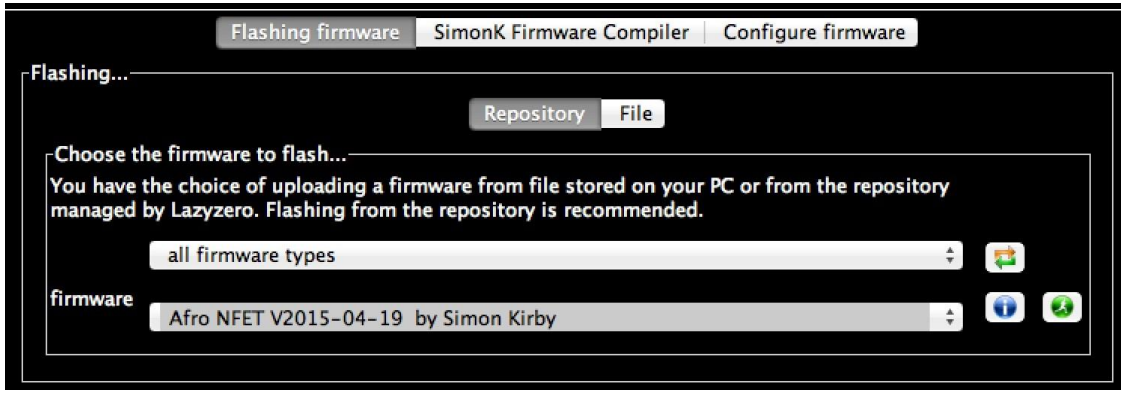


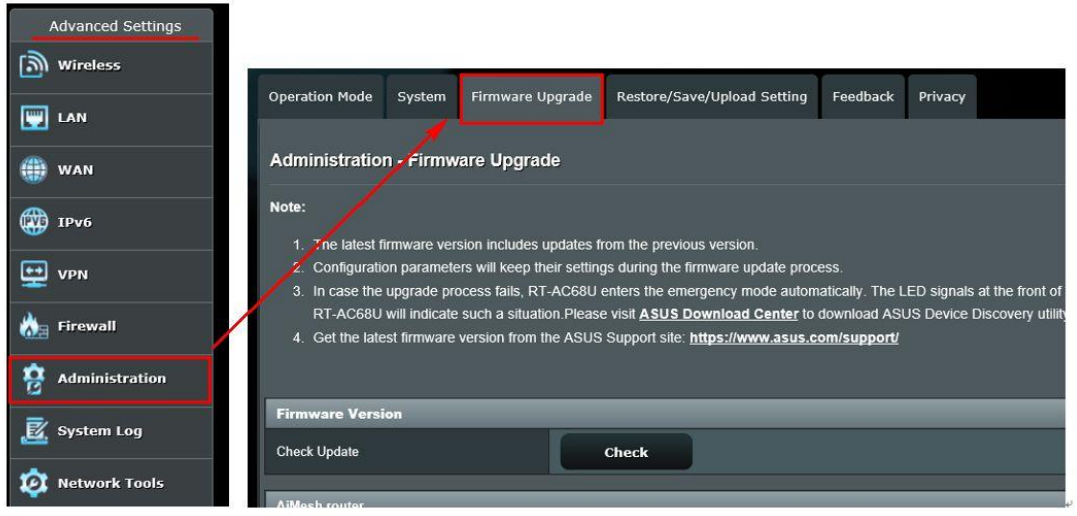
What Do You See

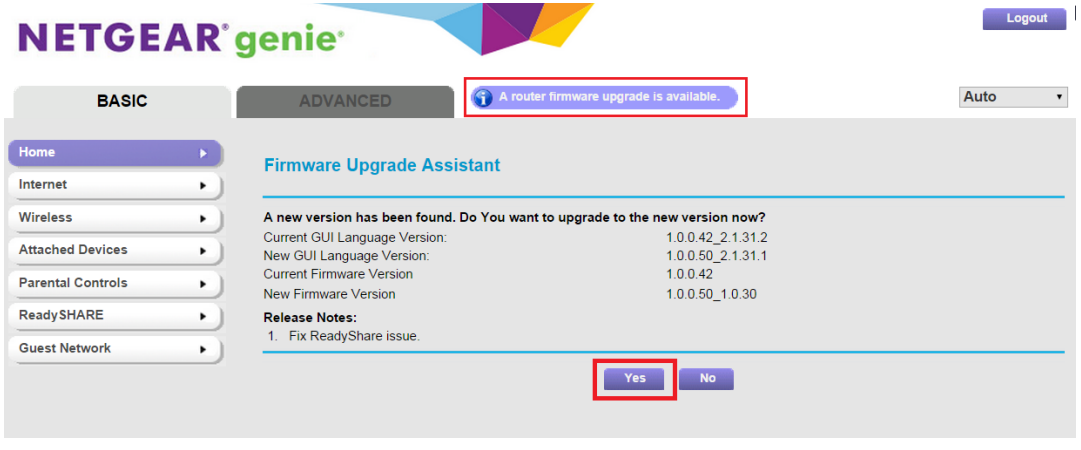
How I (We) Start

Third Party Firmware







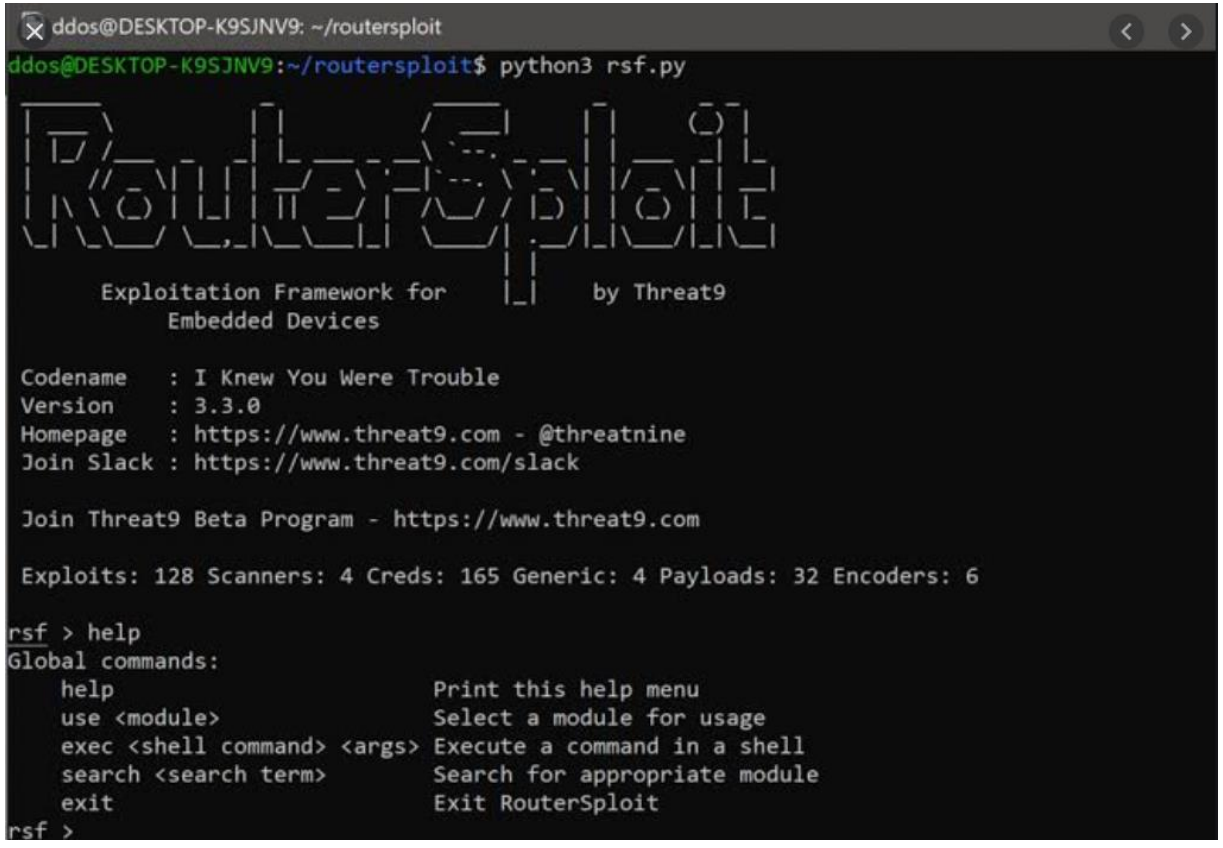


Started With This

Having Fun With This

Traditional IoT Hacking

The Web Hacker



Exploits found on the INTERNET

This is live excerpt from our database. Available also using API

Edit	Date	Name	Status
	2019-08-25	D-Link DIR-600M Authentication Bypass Metasploit	Published
	2019-08-01	D-Link 6600-AP XSS / DoS / Information Disclosure	Published
	2019-05-07	D-Link DWL-2600AP Authenticated OS Command Injection	Published
	2019-04-11	D-Link DI-524 2.06RU Cross Site Scripting	Published
	2019-03-03	Xoops 1.0.2 PD-Links Modules 1.0 Krobi Database Disclosure	Published
	2018-12-23	D-Link DSL-2770L / DIR-140L / DIR-640L Credential Disclosure	Published
	2018-12-23	D-Link DSL-2770L Credential Disclosure	Published
	2018-11-09	D-LINK Central WifiManager CWM 100 1.03 r0098 Man-In-The-Middle	Published
	2018-11-09	D-LINK Central WifiManager CWM 100 1.03 r0098 DLL Hijacking	Published
	2018-11-09	D-LINK Central WifiManager CWM 100 1.03 r0098 Server-Side Request Forgery	Published
	2018-10-19	D-Link Plain-Text Password Storage / Code Execution / Directory Traversal	Published
	2018-10-13	D-Link DSL-2640T Cross Site Scripting	Published
	2018-09-06	D-Link Dir-600M N150 Cross-Site Scripting	Published
	2018-09-03	D-Link DIR-615 - Denial of Service	Published
	2018-08-28	D-Link DSL-2750U Setup Wizard Page Authentication Bypass	Published
	2018-08-24	D-Link EyeOn Baby Monitor DCS-825L Remote Code Execution	Published
	2018-08-24	D-Link EyeOn Baby Monitor DCS-825L Command Injection	Published
	2018-07-25	D-link DAP-1360 Path Traversal / Cross-Site Scripting	Published
	2018-07-03	D-Link DIR-890L A2 Improper Access Control	Published
	2018-05-26	D-Link DSL-2750B OS Command Injection Metasploit	Published
	2018-05-25	D-Link DSL-2750B OS Command Injection	Published
	2018-05-09	D-Link DIR-868L 1.12 Cross Site Request Forgery	Published
	2018-04-17	D-Link DIR-615 Persistent Cross Site Scripting	Published
	2018-03-31	D-Link DIR-850L Wireless AC1200 Dual Band Gigabit Cloud Router Authentication Bypass	Published
	2018-03-01	D-Link DGS-3000-10TC Cross Site Request Forgery	Published
	2018-01-15	D-Link DNS-343 ShareCenter 1.05 Command Injection	Published
	2018-01-15	D-Link DNS-325 ShareCenter 1.05B03 Shell Upload / Command Injection	Published

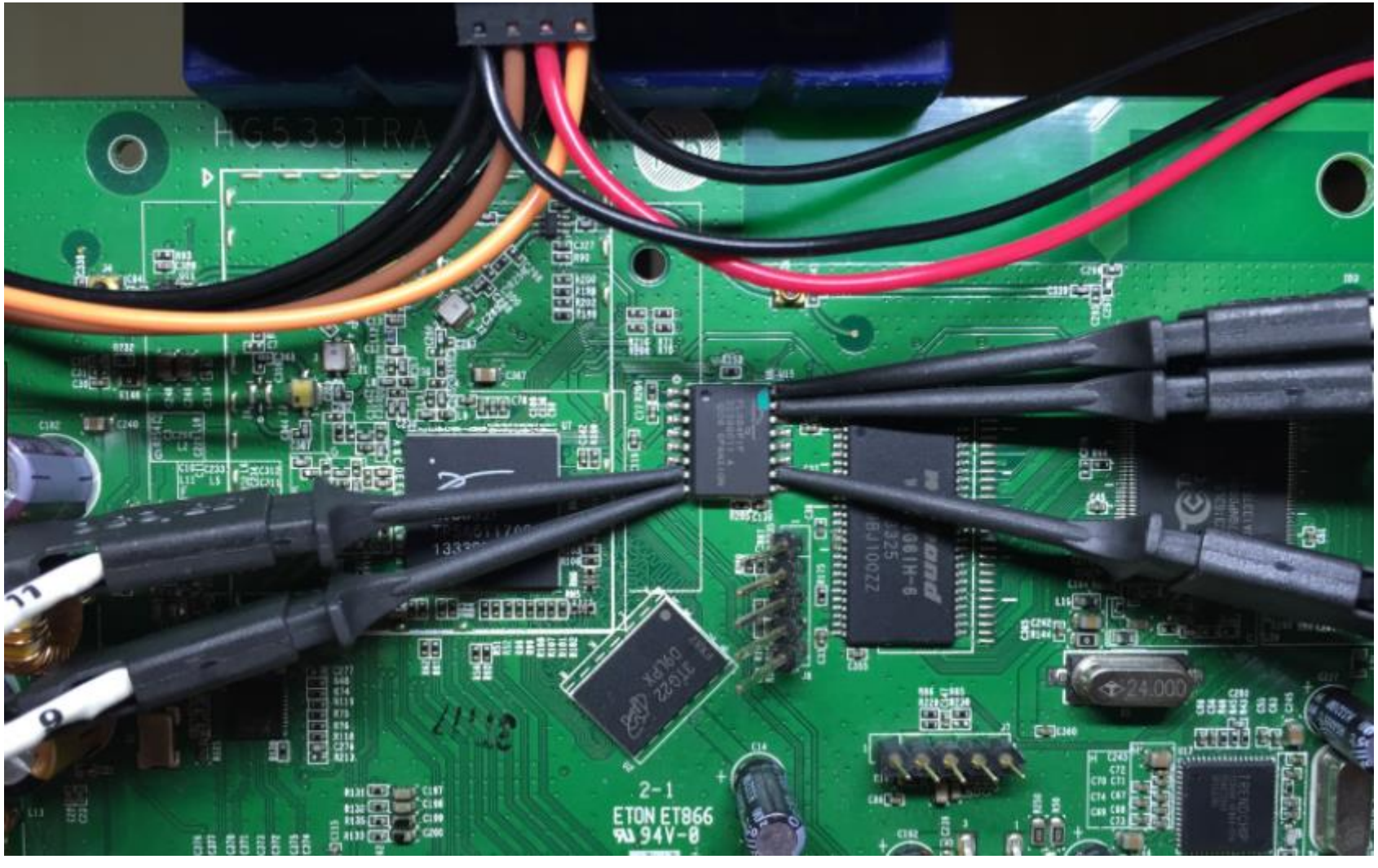
Firmware Hacking

```
→ tools binwalk -e test.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
218040        0x353B8         CRC32 polynomial table, little endian
524288        0x80000         uImage header, header size: 64 bytes, header CRC:
0x4687D1AC, created: 2007-06-15 10:36:26, image size: 2217656 bytes, Data Addres
s: 0x2000000, Entry Point: 0x2000040, data CRC: 0xA54D09E1, OS: Linux, CPU: ARM,
 image type: OS Kernel Image, compression type: none, image name: "gm8136"
524352        0x80040         Linux kernel ARM boot executable zImage (little-en
dian)
542452        0x846F4         gzip compressed data, maximum compression, from Un
ix, last modified: 1970-01-01 00:00:00 (null date)
3670112       0x380060        xz compressed data
3800908       0x39FF4C        xz compressed data
3931872       0x3BFEE0        xz compressed data
4979008       0x4BF940        xz compressed data
```

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
217628        0x3521C         CRC32 polynomial table, little endian
524288        0x80000         uImage header, header size: 64 bytes, header CRC: 0x68F55153, created: 2006-09-23 11:52:56, image size: 2
217456 bytes, Data Address: 0x2000000, Entry Point: 0x2000040, data CRC: 0xD41DD892, OS: Linux, CPU: ARM, image type: OS Kernel Image,
compression type: none, image name: "gm8136"
524352        0x80040         Linux kernel ARM boot executable zImage (little-endian)
542452        0x846F4         gzip compressed data, maximum compression, from Unix, last modified: 1970-01-01 00:00:00 (null date)
3670016       0x380000        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 6963644 bytes, 183 inodes, blocksi
ze: 131072 bytes, created: 2006-09-24 03:01:35
11534336      0xB00000        JFFS2 filesystem, little endian
^C
→       dd if=./MX25L12805_20170912_140739.BIN bs=3670016 count=1 of=part1.bin ; \
> dd if=./MX25L12805_20170912_140739.BIN bs=11534336 skip=1 of=part2.bin ; \
> mksquashfs  squashfs-root squashfs-customize.bin -comp xz ; \
>
1+0 records in
1+0 records out
3670016 bytes (3.7 MB, 3.5 MiB) copied, 0.0050487 s, 727 MB/s
0+1 records in
0+1 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.00694756 s, 755 MB/s
```

extract the front and back parts of the file system and repackage the file system

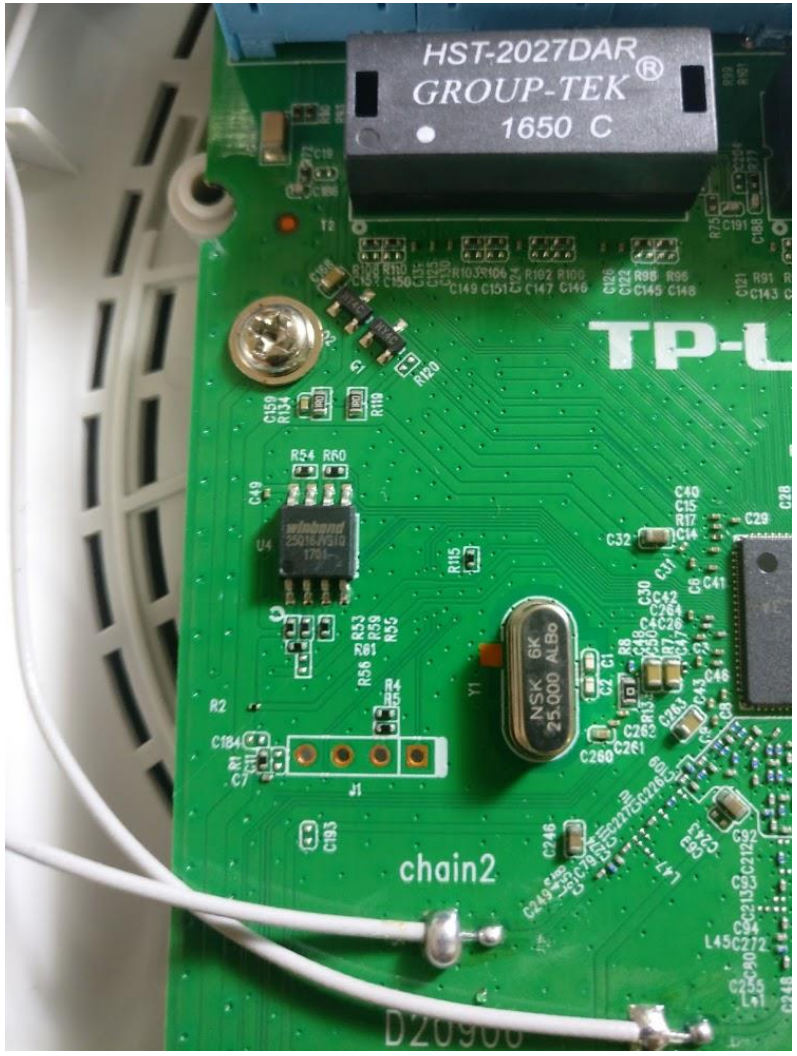
Hardware Hacking

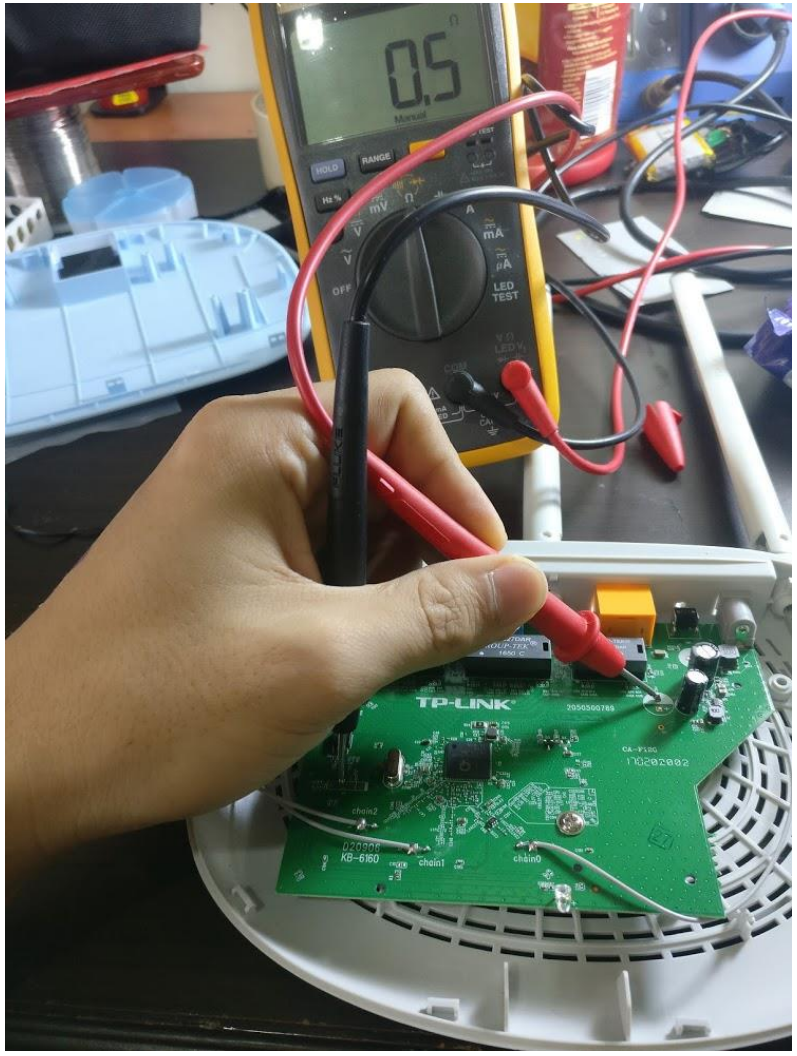


Getting Started

IoT Attack Case Study

IoT Attack Case Study





BUY

Must Have

买1送5 胜利正品数字万用表VC890C+ 全保护万能表数显多用表电表

胜利经典款 欧洲安全标准 测试快稳定

天猫 购物券 全天猫实物商品通用 去领券

价格 ~~¥176.00-612.00~~

促销价 ¥88.00-306.00

本店活动 满2件9.8折；满5件9.6折 更多优惠

运费 湖南长沙 至 杭州 快递: 0.00 EMS: 25.00 平邮: 30.00

月销量 4691 累计评价 35418 送天猫积分 44起

颜色分类

- VC990C+标配【送鳄鱼夹和仪表包】
- VC890C+标配+仪表包【送鳄鱼夹】
- VC890C+标配【送鳄鱼夹】
- VC890C+标配+20A原装表笔+充电套装【送鳄鱼夹】
- VC890C+标配+充电套装【送鳄鱼夹】
- VC890C+标配+仪表包+充电套装【送鳄鱼夹】
- VC890C+标配+仪表包+20A原装表笔【送鳄鱼夹】
- VC890C+标配+20A特尖+充电套装【送鳄鱼夹】
- VC890C+标配+20A原装表笔【送鳄鱼夹】

分享 收藏商品（20733人气） 举报

单排针 圆排针2.54MM 2.0MM间距1*40P双排针2*40P 弯针单双排针

品质保证 长久耐用 7天包退换

天猫 购物券 全天猫实物商品通用 去领券

价格 ¥2.50

运费 广东深圳 至 广州 快递: 0.00

月销量 808 累计评价 1100

颜色分类

- 2.54单排针 铁【普通长度】 10条
- 2.54单排针 铜【长15MM】10条
- 2.54单排针 铜【长17MM】10条
- 2.54单排针 铜【长19MM】 10条
- 2.54单排针 铜【长25MM】10条
- 2.54双排针 铜【长15MM】10条
- 2.54单排弯针 铜【长11MM】 10条
- 2.54双排弯针 铜 【长11MM】 10条
- 2.54双排针 铜【长11MM】10条
- 2.54双排针 铜【长19MM】10条
- 2.54双排针 铜【长25MM】10条
- 2.0单排针 铜 10条
- 2.0单排弯针 铜 10条
- 2.54镀金单排针 10条

Risym 六合一多功能USB转UART串口模块CP2102 usb/TTL485/232互转

RISYM 出口 质优产品

天猫 购物券 全天猫实物商品通用 去领券

价格 ~~¥19.90~~

促销价 ¥16.80 首件体验价

本店活动 满68元,包邮；满688元,包邮,赠满688送充电宝 更多优惠

运费 广东深圳 至 广州 快递: 3.00 EMS: 23.00

月销量 120 累计评价 147 送天猫积分 1

数量 1 件 库存8954件

立即购买 加入购物车

服务承诺 正品保证 极速退款 七天无理由退换 支付方式

分享 收藏商品（376人气） 举报

QUICK快克203H/203D数显无铅高频恒温焊台90W大功率烙铁204电焊台

控温准确 回温讯速 大功率90W

天猫 购物券 全天猫实物商品通用 去领券

价格 ~~¥888.00~~

促销价 ¥850.00 品牌钜惠

运费 广东深圳 至 广州 快递: 12.00 EMS: 70.00 平邮: 39.00

月销量 49 累计评价 179 送天猫积分 425

颜色分类

- 203H(数显90W)
- 204H(机械90W)
- 203(数显60W)
- 204(机械60W)
- 203D(双数显90W)

数量 1 件 库存586件

立即购买 加入购物车

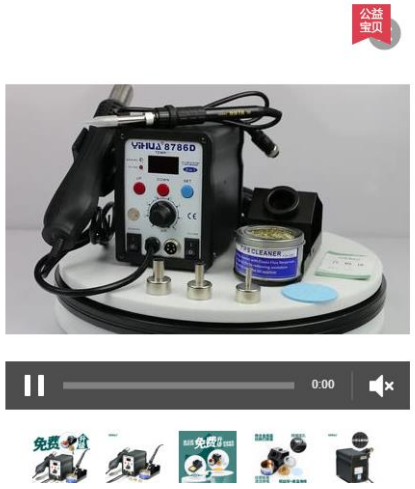
服务承诺 正品保证 七天无理由退换 支付方式

What To Buy

Hot Air Gun


QUICK 快克锡焊
官方授权 正品保证
1年保修
防静电设计
数字化温度校准
203H
回温迅速
分享
收藏商品（376人气）
举报


QUICK快克203H/203D数显无铅高频恒温焊台90W大功率烙铁204电焊台
控温准确 回温讯速 大功率90W
天猫 购物券 全天猫实物商品通用
去领券
价格 ¥888.00
促销价 ¥850.00 品牌钜惠
运费 广东深圳 至 广州 快递: 12.00 EMS: 70.00 平邮: 39.00
月销量 49
累计评价 179
送天猫积分 425
颜色分类
203H(数显90W)
204H(机械90W)
203(数显60W)
204(机械60W)
203D(双数显90W)
数量 1 件 库存586件
立即购买
加入购物车
服务承诺 正品保证 七天无理由退换
支付方式


公益宝贝
YIHUA 8786D
0:00


YIHUA-8786D数显热风枪焊台二合一恒温电烙铁焊台维修必备包邮
谊华正品 恒温稳定 升温迅速 部分包邮
天猫 购物券 全天猫实物商品通用
去领券
价格 ¥499.00
促销价 ¥196.00 夏季促销
本店活动 满100元减3元；满300元减10元
更多优惠
运费 广东广州 至 广州 快递 0.00
17:00前付款，预计8月13日(明天)送达
月销量 599
累计评价 3807
送天猫积分 98
颜色分类
数量 1 件 库存26件
立即购买
加入购物车

Multi Meter

分享 ★ 收藏商品（20733人气） 举报

买1送5 胜利正品数字万用表VC890C+ 全保护万能表数显多用表电表

胜利经典款 欧洲安全标准 测试快稳定

天猫 购物券 全天猫实物商品通用 去领券

价格 ¥ 176.00-612.00

促销价 ¥88.00-306.00

本店活动 满2件9.8折；满5件9.6折 更多优惠

运费 湖南长沙 至 杭州 快递: 0.00 EMS: 25.00 平邮: 30.00

月销量 4691 累计评价 35418 送天猫积分 44起

颜色分类

- VC990C+标配【送鳄鱼夹和仪表包】
- VC890C+标配+仪表包【送鳄鱼夹】
- VC890C+标配【送鳄鱼夹】
- VC890C+标配+20A原装表笔+充电套装【送鳄鱼夹】
- VC890C+标配+充电套装【送鳄鱼夹】
- VC890C+标配+仪表包+充电套装【送鳄鱼夹】
- VC890C+标配+仪表包+20A原装表笔【送鳄鱼夹】
- VC890C+标配+20A特尖+充电套装【送鳄鱼夹】
- VC890C+标配+20A原装表笔【送鳄鱼夹】

分享 ★ 收藏商品（11人气） 举报

福禄克万用表F15B+数字万用表FLUKE17B+/18B+高精度数字万能表

原装正品 新升级

天猫 购物券 全天猫实物商品通用 去领券

价格 ¥ 499.00-699.00

促销价 ¥468.00-684.00

运费 广东东莞 至 杭州 快递: 0.00

月销量 11 累计评价 10 送天猫积分 234起

颜色分类

数量 1 件 库存256件

立即购买 加入购物车

服务承诺 正品保证 赠运费险 七天无理由退换 支付方式

USB-TTL & Rainbow Wire

Risym 六合一多功能USB转UART串口模块CP2102 usb/TTL485/232互转

RISYM 出口 质优产品

天猫 购物券 全天猫实物商品通用 去刮券

价格 ¥ 19.90

促销价 ¥16.80 首件体验价

本店活动 满68元包邮；满688元包邮,赠满688送充电宝 更多优惠

运费 广东深圳 至 广州 快递: 3.00 EMS: 23.00

月销量 120 累计评价 147 送天猫积分 1

数量 1 件 库存8954件

立即购买 加入购物车

服务承诺 正品保证 极速退款 七天无理由退换 支付方式

微雪 FT232模块 FT232RL USB转TTL USB转UART 串口模块 micro接口

FT232 模块 刷机神器

天猫 购物券 全天猫实物商品通用 去刮券

价格 ¥28.00

本店活动 满508元减10元 更多优惠

运费 广东深圳 至 广州 快递: 8.00 EMS: 24.00

月销量 103 累计评价 50 送天猫积分 2

数量 1 件 库存7350件

立即购买 加入购物车

服务承诺 正品保证 极速退款 赠运费险 七天无理由退换 支付方式

分享 收藏商品（25人气） 举报

¥3.10

40P彩色杜邦线 母对母 公对公 公对母

荣域华府旗舰店

月成交 2474笔 评价 9561

¥6.19

域能 40P彩色杜邦线母对母 公对公 公

域能旗舰店

月成交 360笔 评价 2251

¥3.28

40P彩色杜邦彩虹排线电子延长线 公转

艾速旗舰店

月成交 500笔 评价 1418

Pin and The Gang





单排针 圆排针2.54MM 2.0MM间距1*40P双排针2*40P 弯针单双排针

品质保证 长久耐用 7天包退换

天猫 购物券 全天猫实物商品通用 去割券

价格 ¥2.50

运费 广东深圳 至 广州 快递: 0.00

月销量 808 | 累计评价 1100

颜色分类

- 2.54单排针 铁【普通长度】 10条
- 2.54单排针 铜【长15MM】10条
- 2.54单排针 铜【长17MM】10条
- 2.54单排针 铜【长19MM】 10条
- 2.54单排针 铜【长25MM】10条
- 2.54双排针 铜【长15MM】10条
- 2.54单排弯针 铜【长11MM】 10条
- 2.54双排弯针 铜 【长11MM】 10条
- 2.54双排针 铜【长11MM】10条
- 2.54双排针 铜【长19MM】10条
- 2.54双排针 铜【长25MM】10条
- 2.0单排针 铜 10条
- 2.0单排弯针 铜 10条
- 2.54镀金单排针 10条


ARTHYLY 间距2.00MM 双排弯针 双排针 弯 插针 2*40PIN

天猫 购物券 全天猫实物商品通用 去割券

价格 ¥2.00

促销价 ¥1.98 促销

运费 广东深圳 至 广州 快递: 0.00 EMS: 0.00 平邮: 0.00

月销量 10 | 累计评价 5

数量 1 件 库存189599件

立即购买 加入购物车

服务承诺 正品保证 极速退款 七天无理由退换 支付方式

Other

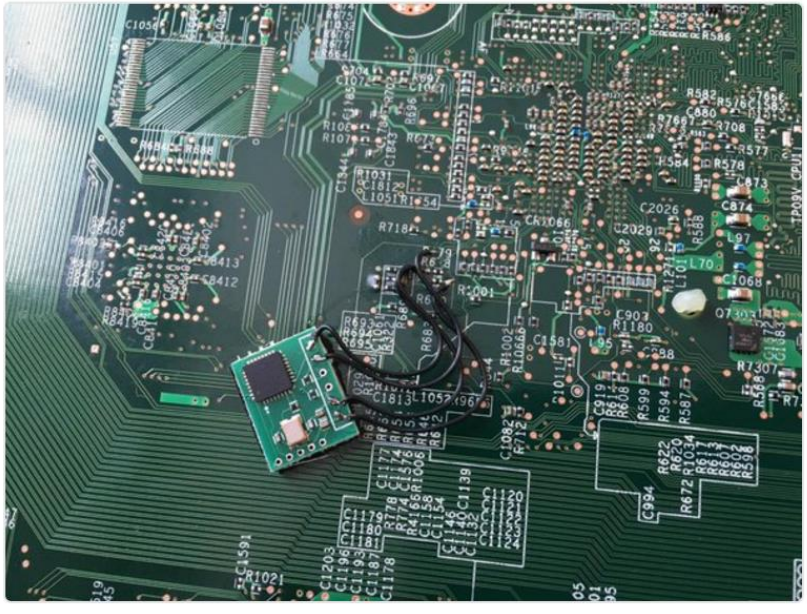
Why Hardware Hacking Is Important

Best Way To Understand The Device

You Retweeted

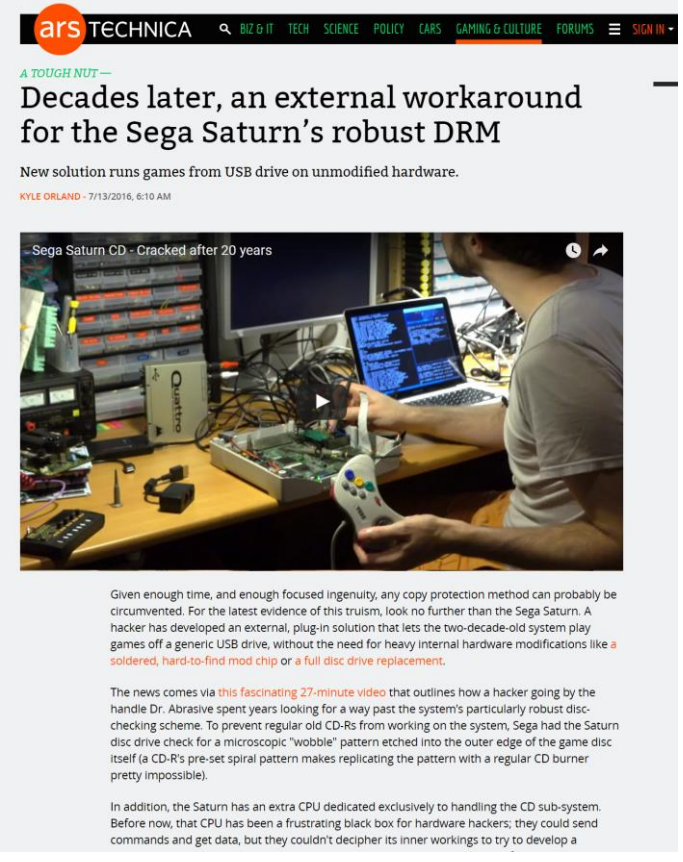
Hacker Fantastic @hackerfantastic · 13 May 2016

Suspicious daughterboard inside Chinese sold cisco gear via @osxreverser ...
soooo dodgy lol



34 459 369

Refurbish



Hacking Game Boy For Fun

Hacking Sega for Real

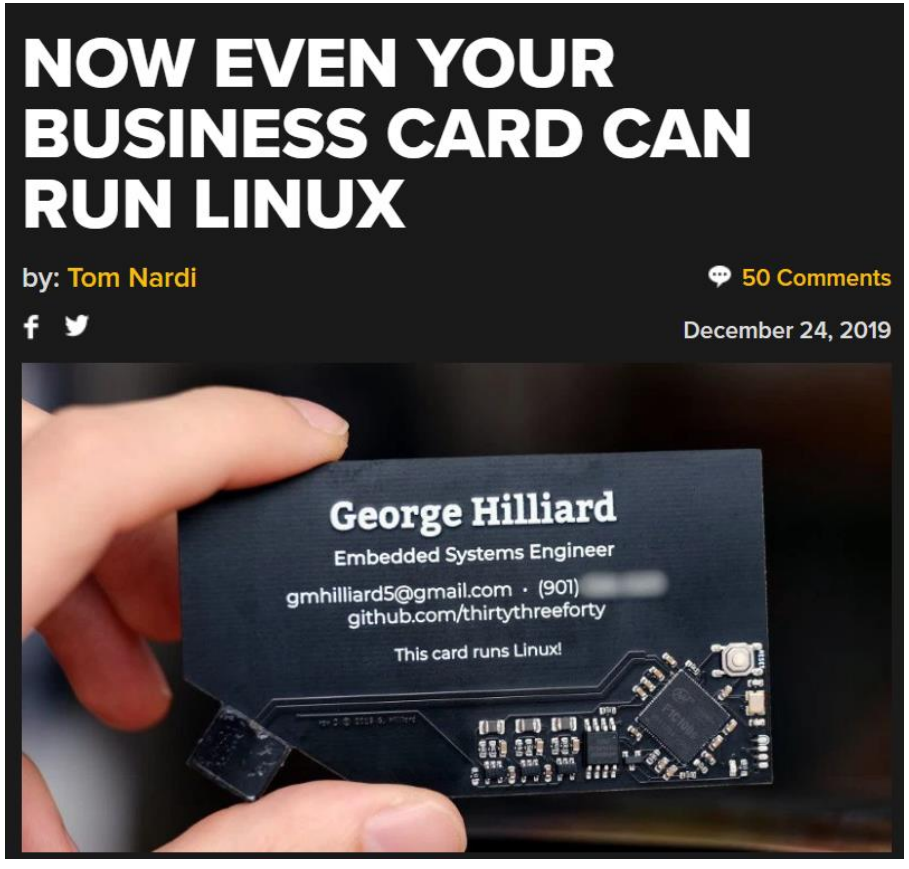
Car Hacking

Hackers crack Tesla Model 3 in competition, Tesla gives them the car

Fred Lambert - Mar. 23rd 2019 4:32 pm ET @FredericLambert

Car is a bigger target now

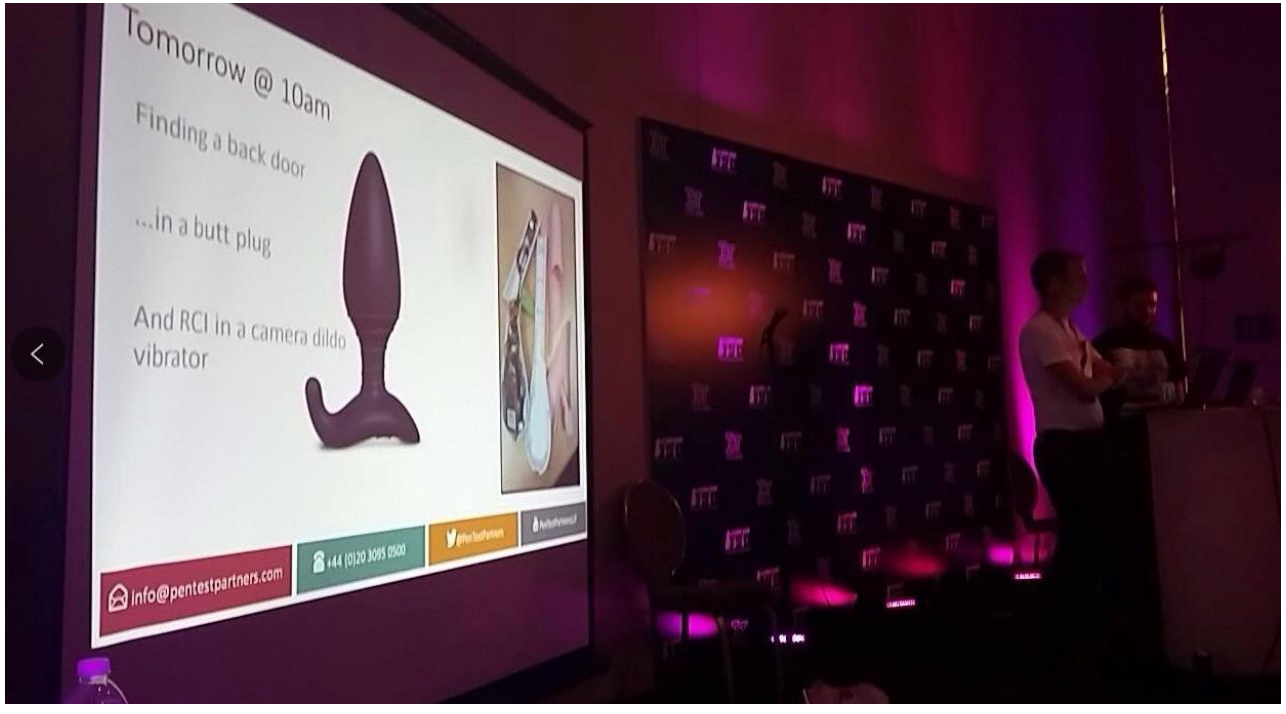
Hacking Little Tools



Hacking Widely Use Chip

One Bug PWN it All

Discussion



1. Any Other IoT
2. Any other thing to hack

Hacking Without Understanding
- Case Study -

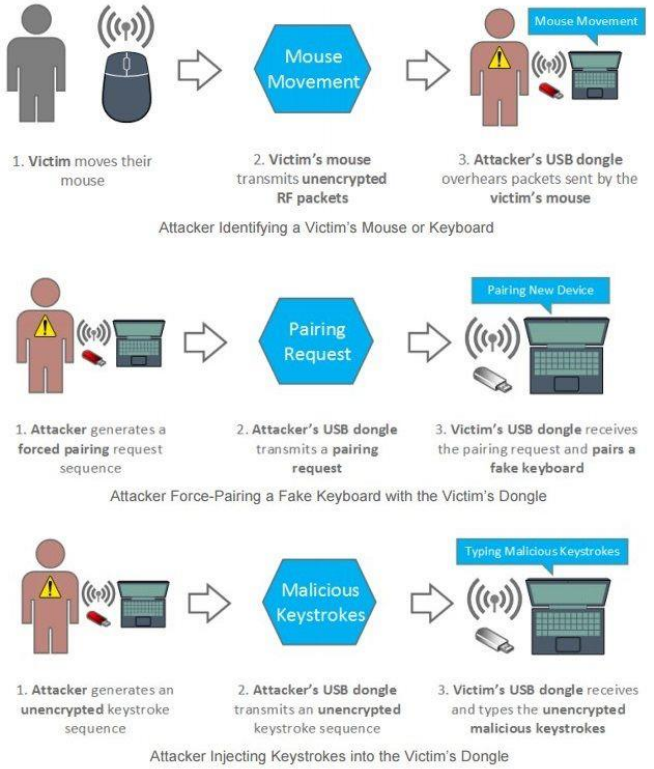
Research Credits

黑客与极客

- Partner In Crime: klks84, https://twitter.com/klks84
- All my missing Logitech keyboard and mouse

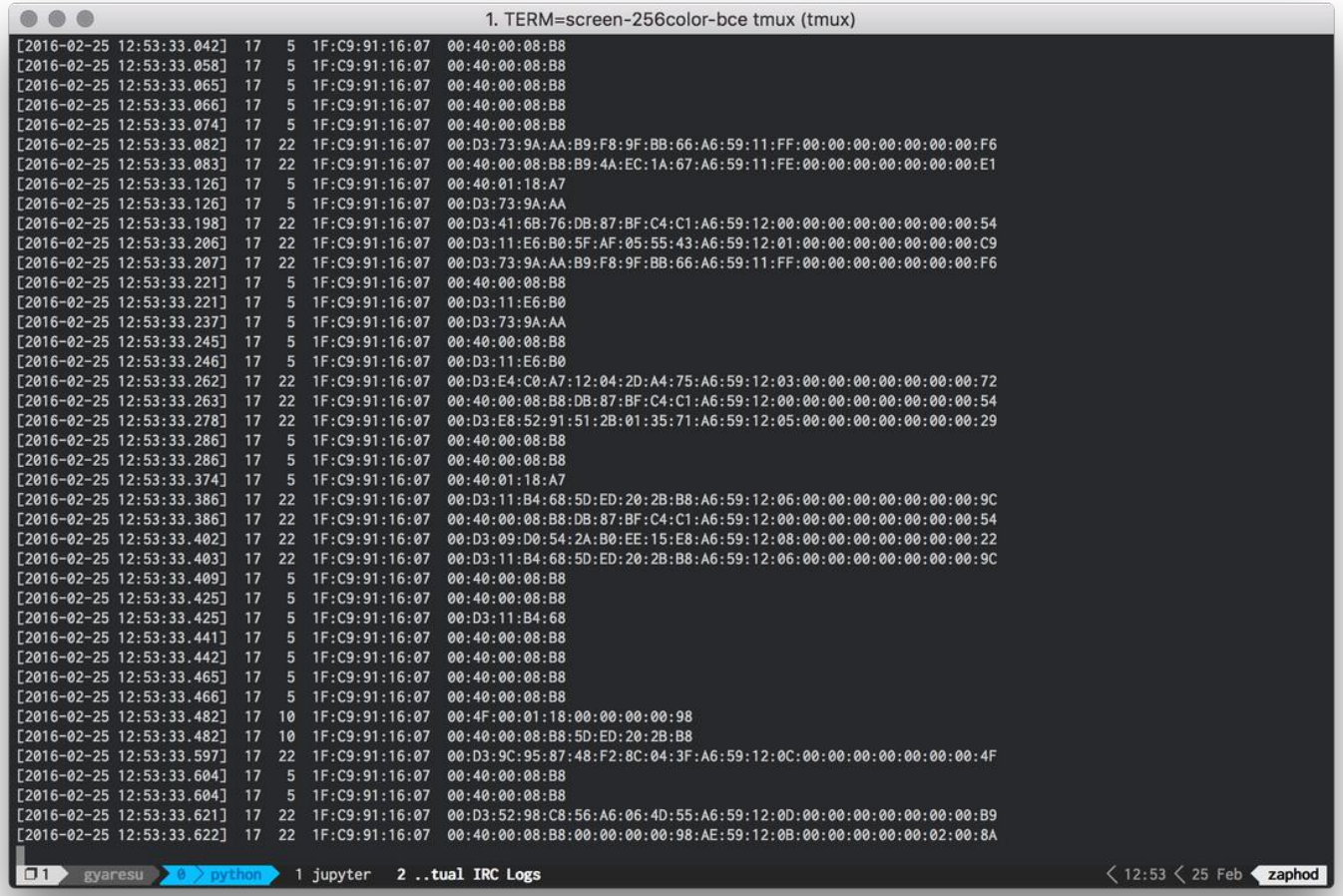
What is This All About

Mousejack



- Targeting non-Bluetooth keyboard and mice
- Sniff and transmit special crated radio packet towards victims
- Keyboards normally sends encrypted packets
- Affected Product ? Most of the non-Bluetooth keyboard and mouse

How It Works



- Scan all the nearby wireless mouse
- Sniff targeted victim
- Dump “keystroke”
- Replay, Hijack, 0w3d

What is Our Target

Motivatios

- Most complete MouseJack implementation guide, in chinese
- Both guide based on Crazyradio. “PA” and non “PA”
- Objective 1: Can it be cheaper?
- Objective 2: Smaller? (Not too obvious)
- Objective 3: Easier to purchase? Just tabao it?

What Is Not In This Research

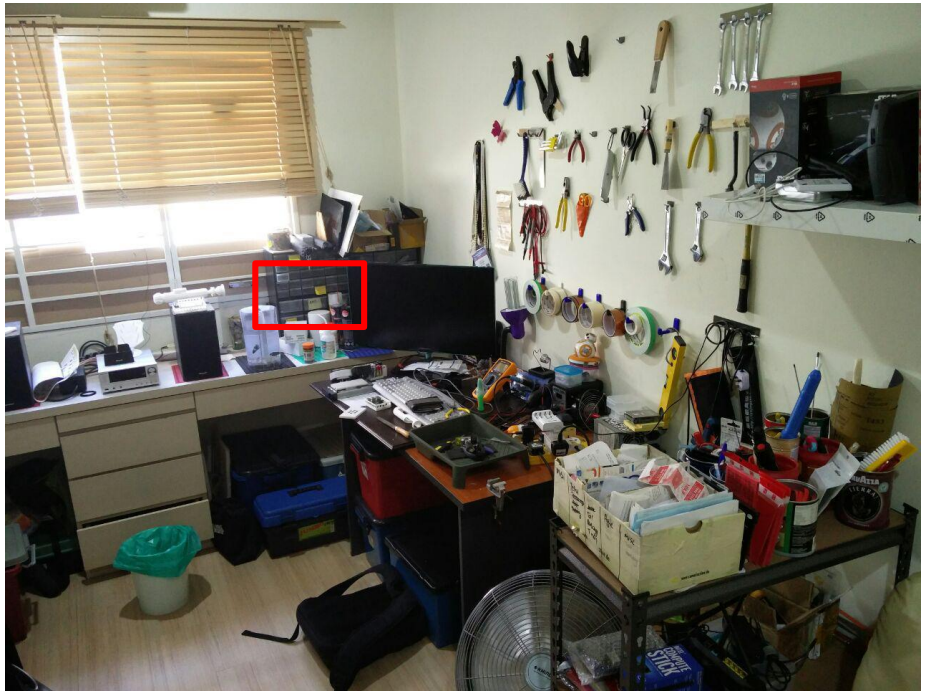
- Nothing to do with keyboard direct injection
- Nothing to do with breaking keyboard encryption
- Nothing to do with mouse injection
- Nothing to do with super long distance sniffing
- BUT

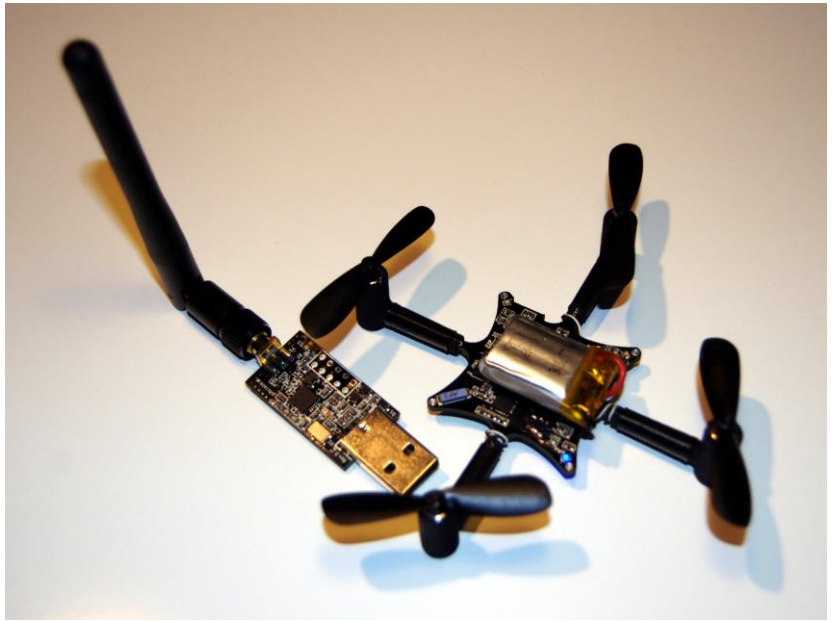
In The Beginning

The Crazy Radio

0xBC

- Based on nRF24LU1+
- 125 radio channels
- Send and receive packets up to 32 bytes
- Design by bitcraze.io

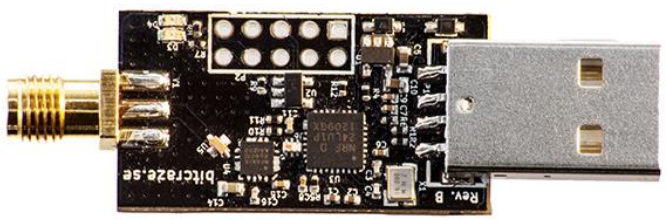


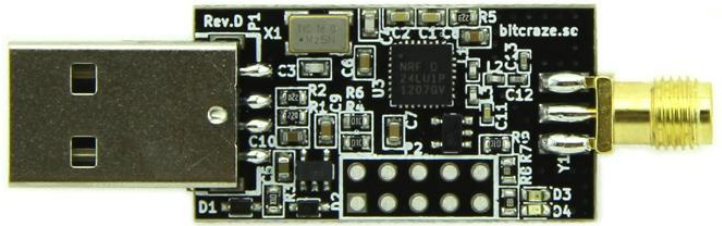


- Got it few years back
- Always hide in small little corner
- After two times flying

Before I Start, I Kill It

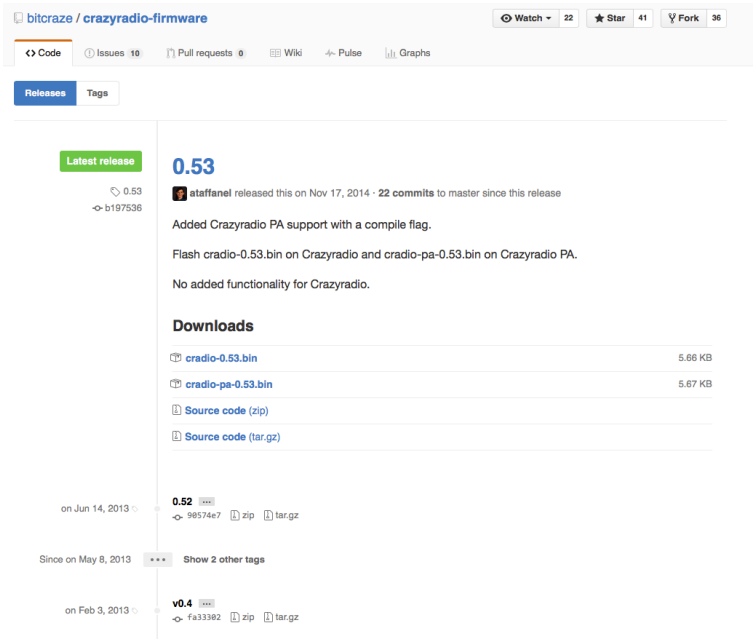
Old and New Crazy Radio





- In the beginning, there are two crazyradio
- Item 1. Crazyradio PA
- Item 2, Crazyradio (Obsolete)
- Crazyradio PA comes with extended range. 1KM
- Bitcraze no longer selling crazyradio

Firmware For The New Crazyradio



USB bootloader (command line instructions)

Please note that you might have to exchange *python* with *python2* if you distro uses python3.

First Crazyradio has to be rebooted in USB bootloader mode. To do so insert the dongle in the pc, open a terminal window and run the bootloader launcher:

```
> cd crazyradio-firmware
> python usbtools/launchBootloader.py
Launch bootloader .
Bootloader started
```

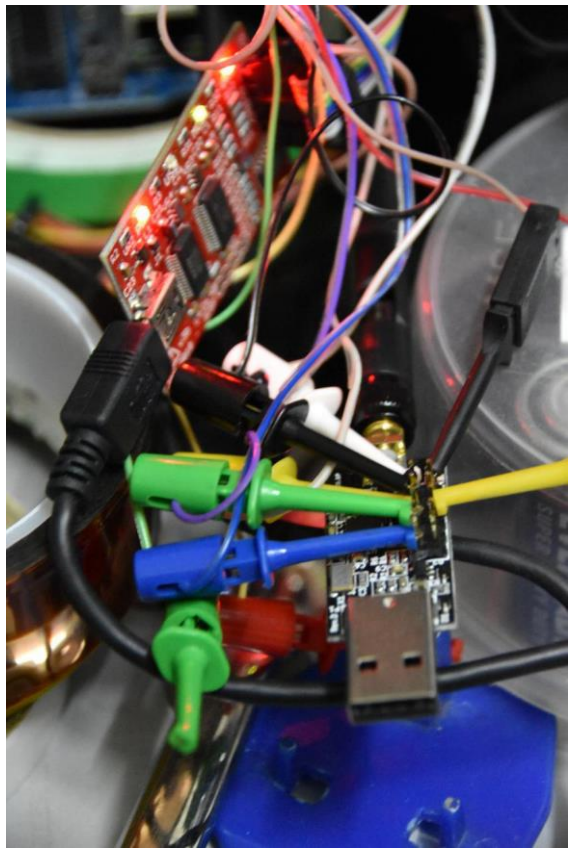
After running this tool the Crazyradio dongle should have disappeared and a new device named **nRF24LU1P-F32 BOOT LDR** should appear.

To flash the firmware use the nrfbootload.py script:

```
> cd crazyradio-firmware
> python usbtools/nrfbootload.py flash cradio-0.53.bin
Found nRF24LU1 bootloader version 18.0
Flashing:
  Flashing 5810 bytes...
Flashing done!
Verifying:
  Reading cradio-pa-0.53.bin...
  Reading 5810 bytes from the flash...
Verification succeded!
```

- At the beginning, there are two crazyradio
- Flashing the “PA” in to the the NON “PA” is a bad idea
- Bootloader is designed for “PA” , but firmware is the same
- The End

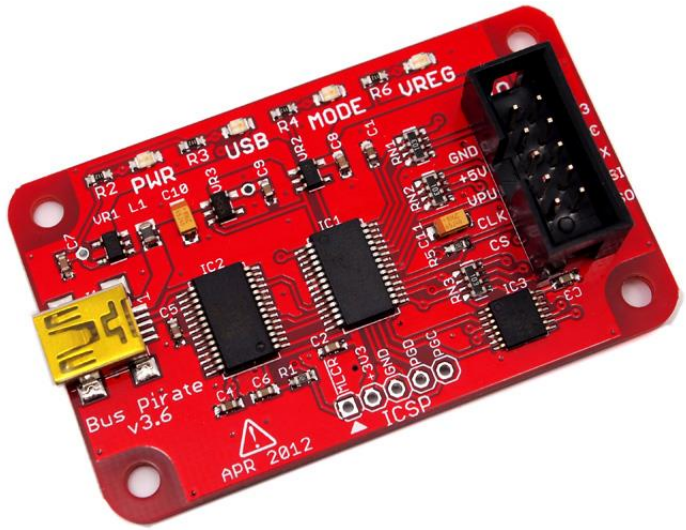
Saving The Crazyradio

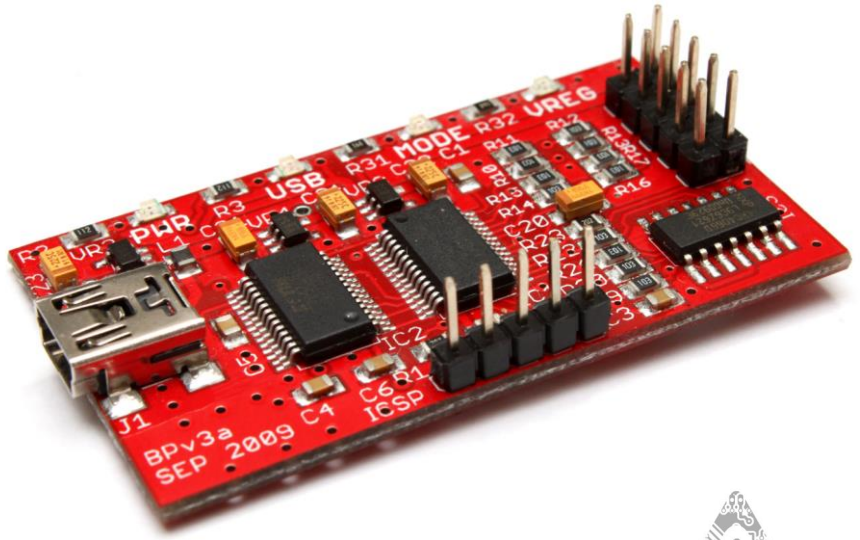


- Need to do a complete flash
- 1 Unit Bus Pirate
- 1 Unit 2 x 5 Pins
- Almost stable hand

Having The Actual Tool

What Is Bus Pirates





Dangerous Prototypes

- Item 1, version 3.6
- Item 2, version 4
- Support 1-Wire, I2C, SPI, JTAG, Asynchronous Serial, MIDI and etc
- SPI is what we need
- Sells by seeeds studio and not in taobao

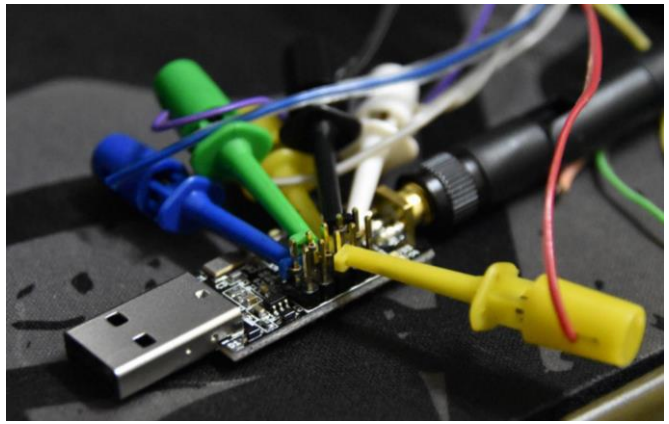
Flash Perl Script

```
#!/usr/bin/perl -w

# Simple perl script to drive the Bus Pirate and unbrick your CrazyRadio dongle.
# Adapted (sorta) from the Bus Pirate example script and mbed NRF24LU1+ flasher projects:
#  http://code.google.com/p/the-bus-pirate/source/browse/trunk/scripts/SPIeeprom.pl
#  http://mbed.org/users/mux/code/nrflash
#
# This script uses the aux output on the Bus Pirate as the PROG pin on the CrazyRadio's NRF24LU1+ chip.
#
# Electrical connections are as follows:
#
#  Bus Pirate       CrazyRadio
#  ==========================
#  MOSI ()      ->  MOSI  (6)
#  MISO ()      ->  MISO  (8)
#  SCK  ()      ->  SCK   (4)
#  CS   ()      ->  CS    (10)
#  AUX  ()      ->  PROG  (2)
#  3V3  ()      ->  3V3   (5)
#  GND  ()      ->  GND   (9)

use strict;
use feature 'say';
use Getopt::Long;
use Device::SerialPort;
use Time::HiRes qw/usleep/;

use constant {
    WREN        => "\x06",
    WRDIS       => "\x04",
    RDSR        => "\x05",
    WRSR        => "\x01",
    READ        => "\x03",
    PROGRAM     => "\x02",
    ERASE_PAGE  => "\x52",
    ERASE_ALL   => "\x62",
    RDFPCR      => "\x89",
    RDISMB      => "\x85",
    ENDEBUG     => "\x86",
    RDYN        => "\x10",
    FLASH_LEN   => 32768,

    BP_CS       => "\x01",
    BP_AUX      => "\x02",
    BP_PULLUP   => "\x04",
    BP_POWER    => "\x08",
};

my %opts;
my $port;
my $time = 500;
my $status_byte;
my $return;
```

- https://raw.githubusercontent.com/koolatron/buspirate_nrf24lu1p/master/flasher.pl
- The defector standard SPI Flasing script for crazyradio

Rescue

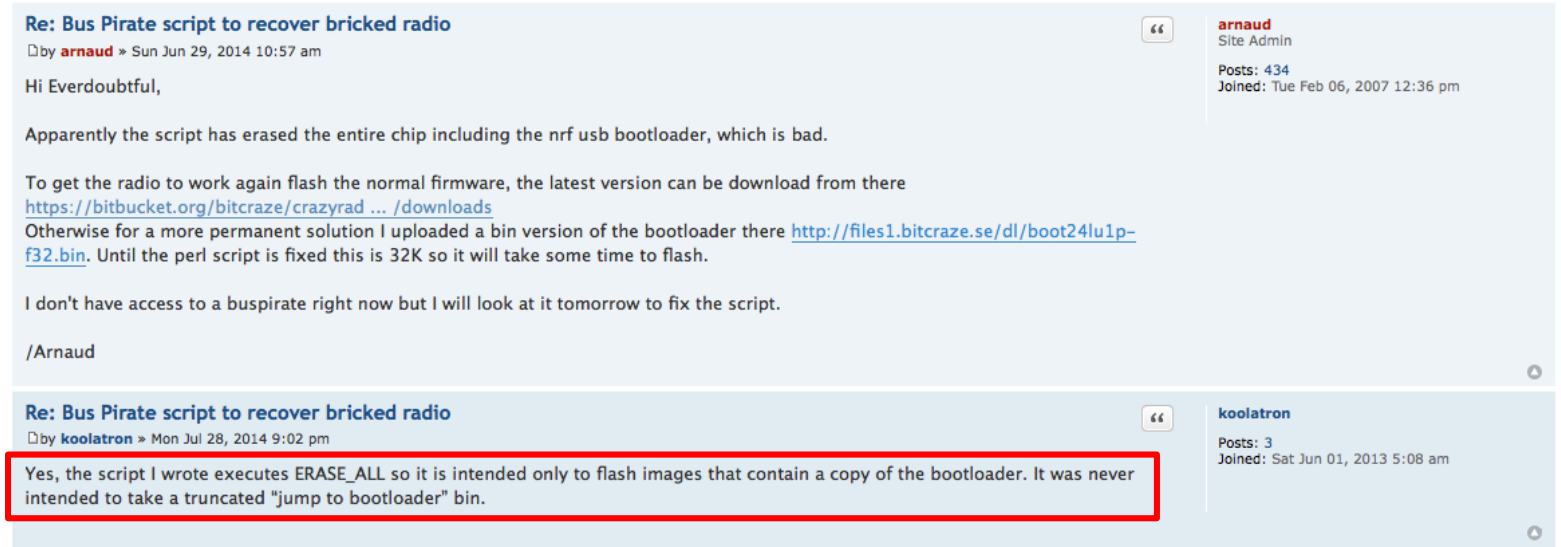
Soldering



Pin Lineup

- Crazyradio comes with breakout pin
- Solder a 2 x 5 Pin into crazy radio
- “Clipped” in Bus Pirate accordingly
- Beware of crazyradio breakout pin sequance

Problem 1: Bootloder Missing



Re: Bus Pirate script to recover bricked radio

by **arnaud** » Sun Jun 29, 2014 10:57 am

Hi Everdoubtful,

Apparently the script has erased the entire chip including the nrf usb bootloader, which is bad.

To get the radio to work again flash the normal firmware, the latest version can be download from there
https://bitbucket.org/bitcraze/crazyrad ... /downloads
Otherwise for a more permanent solution I uploaded a bin version of the bootloader there http://files1.bitcraze.se/dl/boot24lu1p-f32.bin. Until the perl script is fixed this is 32K so it will take some time to flash.

I don't have access to a buspirate right now but I will look at it tomorrow to fix the script.

/Arnaud

arnaud
Site Admin
Posts: 434
Joined: Tue Feb 06, 2007 12:36 pm

Re: Bus Pirate script to recover bricked radio

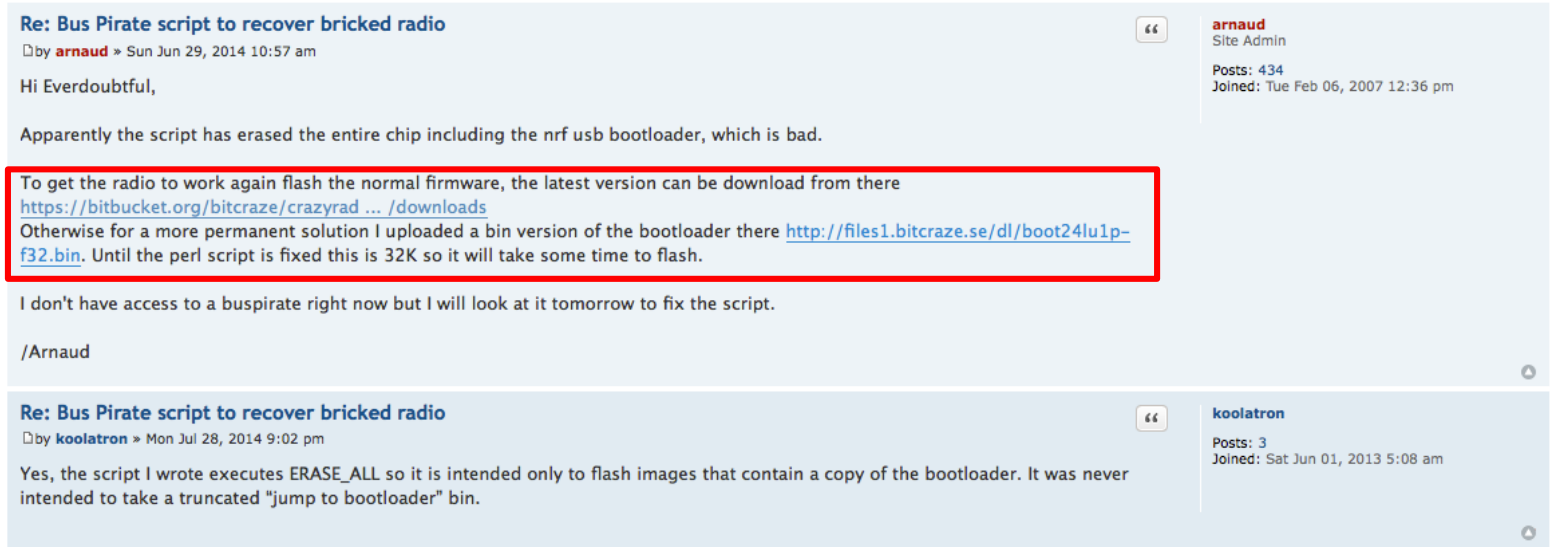
by **koolatron** » Mon Jul 28, 2014 9:02 pm

Yes, the script I wrote executes ERASE_ALL so it is intended only to flash images that contain a copy of the bootloader. It was never intended to take a truncated "jump to bootloader" bin.

koolatron
Posts: 3
Joined: Sat Jun 01, 2013 5:08 am

- Due to the “PA” flashed in to the NON “PA”, it overwrites the bootloader
- Almost broken perl script not able to execute completely
- ERASE_ALL makes it all worse
- Info: https://forum.bitcraze.io/viewtopic.php?t=323

The bootloader



Re: Bus Pirate script to recover bricked radio

by **arnaud** » Sun Jun 29, 2014 10:57 am

Hi Everdoubtful,

Apparently the script has erased the entire chip including the nrf usb bootloader, which is bad.

To get the radio to work again flash the normal firmware, the latest version can be download from there
https://bitbucket.org/bitcraze/crazyrad ... /downloads
Otherwise for a more permanent solution I uploaded a bin version of the bootloader there http://files1.bitcraze.se/dl/boot24lu1p-f32.bin. Until the perl script is fixed this is 32K so it will take some time to flash.

I don't have access to a buspirate right now but I will look at it tomorrow to fix the script.

/Arnaud

arnaud
Site Admin
Posts: 434
Joined: Tue Feb 06, 2007 12:36 pm

Re: Bus Pirate script to recover bricked radio

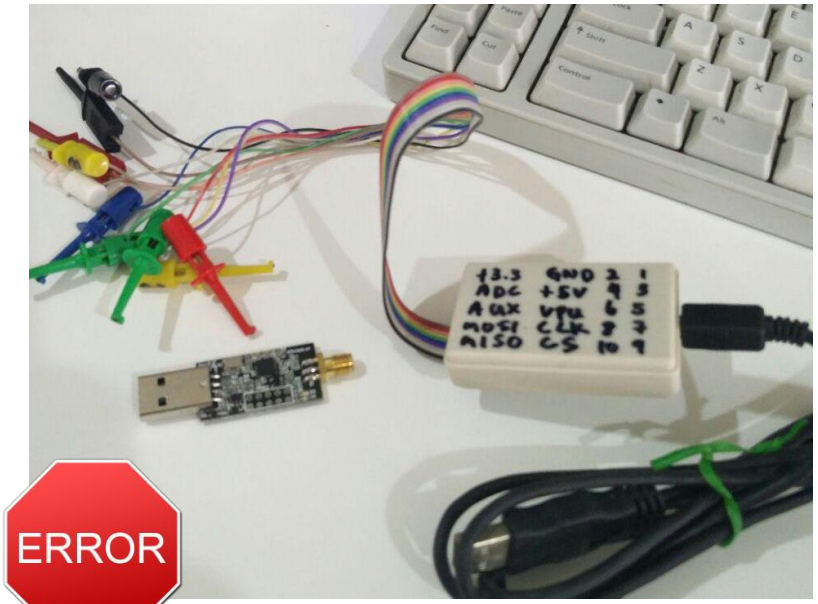
by **koolatron** » Mon Jul 28, 2014 9:02 pm

Yes, the script I wrote executes ERASE_ALL so it is intended only to flash images that contain a copy of the bootloader. It was never intended to take a truncated "jump to bootloader" bin.

koolatron
Posts: 3
Joined: Sat Jun 01, 2013 5:08 am

- The possible way is, flash the boot loader
- Once completed, flash the crazyradio firmware
- Bootloder: https://github.com/xwings/tuya

The Final Error



- # git clone https://github.com/RFStorm/mousejack.git
- # cd mousejack
- # make
- Flash the firmware into crazyradio
- Almost working perl script not working

The “Broken” Perl Script

```
use strict;
use feature 'say';
use Getopt::Long;
use Device::SerialPort;
use Time::HiRes qw/usleep/;

use constant {
    WREN        => "\x06",
    WRDIS       => "\x04",
    RDSR        => "\x05",
    WRSR        => "\x01",
    READ        => "\x03",
    PROGRAM     => "\x02",
    ERASE_PAGE  => "\x52",
    ERASE_ALL   => "\x62",
    RDFPCR      => "\x89",
    RDISMB      => "\x85",
    ENDEBUG     => "\x86",
    RDYN        => "\x10",
    FLASH_LEN   => 32768,

    BP_CS       => "\x01",
    BP_AUX      => "\x02",
    BP_PULLUP   => "\x04",
    BP_POWER    => "\x08",
};

my %opts;
my $port;
my $time = 500;
my $status_byte;
my $return;

if (!&GetOptions(\%opts,
    'input=s',
    'device=s',
) || ( !$opts{input} && !$opts{device} ) ) {
    die "Please specify both -input <input_file.bin> and -device <Bus Pirate devnode>";
}

$port = new Device::SerialPort( $opts{device} );

# Setup serial

$port->baudrate(115200);
$port->parity("none");
$port->databits(8);
$port->stopbits(1);
$port->buffers(1,1);
$port->write_settings || undef $port;

die "Unable to write settings to serial port." unless $port;

# Setup BP
say "Entering raw bitbang mode...";
    while ( ( $port->read( 5 ) ne "BBIO1" ) && --$time ) {
    $port->write( "\x00" );
    usleep( 20000 );
}
die "Unable to enter raw bitbang mode!" unless $time;
```

```
use strict;
use feature 'say';
use Getopt::Long;
use Device::SerialPort;
use Time::HiRes qw/usleep/;

use constant {
    WREN        => "\x06",
    WRDIS       => "\x04",
    RDSR        => "\x05",
    WRSR        => "\x01",
    READ        => "\x03",
    PROGRAM     => "\x02",
    ERASE_PAGE  => "\x52",
    ERASE_ALL   => "\x62",
    RDFPCR      => "\x89",
    RDISMB      => "\x85",
    ENDEBUG     => "\x86",
    RDYN        => "\x10",
    FLASH_LEN   => 32768,

    BP_CS       => "\x01",
    BP_AUX      => "\x02",
    BP_PULLUP   => "\x04",
    BP_POWER    => "\x08",
};

my %opts;
my $port;
my $time = 500;
my $status_byte;
my $return;

if (!&GetOptions(\%opts,
    'input=s',
    'device=s',
) || ( !$opts{input} && !$opts{device} ) ) {
    die "Please specify both -input <input_file.bin> and -device <Bus Pirate devnode>";
}

$port = new Device::SerialPort( $opts{device} );

# Setup serial

$port->baudrate(115200);
$port->parity("none");
$port->databits(8);
$port->stopbits(1);
$port->buffers(1,1);
$port->write_settings || undef $port;

die "Unable to write settings to serial port." unless $port;

# Setup BP
say "Entering raw bitbang mode...";
    while ( ( $port->read( 5 ) ne "BBIO1" ) && --$time ) {
    usleep( 40000 );
}
die "Unable to enter raw bitbang mode!" unless $time;
```

- https://raw.githubusercontent.com/koolatron/buspirate_nrf24lu1p/master/flasher.pl
- Broken by default under VM
- Replace all usleep(20000) to usleep(40000)
- The Fix: https://github.com/xwings/tuya

Re-Flash

```
root@kali:~/buspirate_nrf24lu1p# perl flasher.pl -i boot24lu1p-f32.bin -device /
dev/ttyUSB0
Entering raw bitbang mode...
Entering binary SPI mode...
Configuring peripherals...
Configuring SPI...
Enabling programming...
Reading status byte...
Status: 20
Erasing chip...
Enabling programming...
Reading status byte...
Status: 20
Programming device...
0000 : 0278
0002 : 0000
```

bootloader

```
root@kali:~/buspirate_nrf24lu1p# perl flasher.pl -input cradio-0.51.bin -device /dev/ttyUSB0
Entering raw bitbang mode...
Entering binary SPI mode...
Configuring peripherals...
Configuring SPI...
Enabling programming...
Reading status byte...
Status: 20
Erasing chip...
Enabling programming...
Reading status byte...
Status: 20
Programming device...
0000 : 0200
0002 : 6b32
0004 : 0000
```

firmware

- Two hours for the bootloader
- Two hours for the crazyradio firmware
- Two hours for the mousejack firmware

It Works

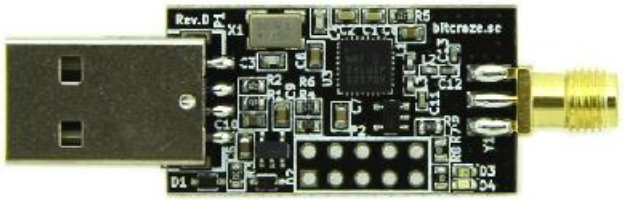
```
[  416.993066] usb 1-2.2: new full-speed USB device number 7 using uhci_hcd
[  417.089596] usb 1-2.2: New USB device found, idVendor=1915, idProduct=0102
[  417.089599] usb 1-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[  417.089600] usb 1-2.2: Product: Research Firmware
[  417.089601] usb 1-2.2: Manufacturer: RFStorm
```

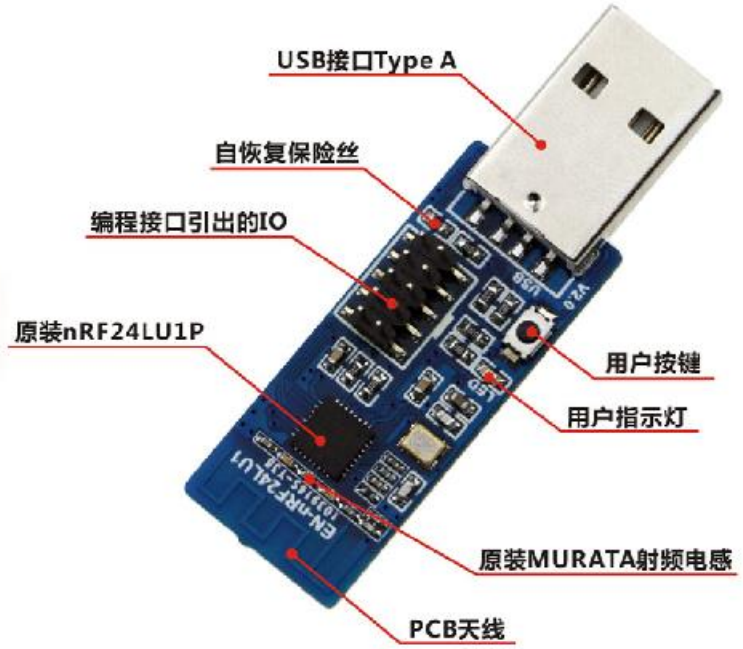
```
(15)# python ./nrf24-scanner.py
[2016-03-24 21:20:07.388]  32   0  72:E4:
[2016-03-24 21:20:07.425]  32   0  72:E4:
[2016-03-24 21:20:07.458]  32  10  72:E4:          00:C2:00:00:02:D0:FF:00:00:6D
[2016-03-24 21:20:32.988]  32   5  72:E4:          00:40:00:6E:52
```

```
(21)# python ./nrf24-sniffer.py -a 72:E4
[2016-03-24 21:23:08.242]  32   5  72:E4          00:40:00:6E:52
[2016-03-24 21:23:08.335]  32   5  72:E4          00:40:00:6E:52
[2016-03-24 21:23:08.427]  32   5  72:E4          00:40:00:6E:52
[2016-03-24 21:23:08.521]  32  10  72:E4          00:C2:00:00:FA:0F:00:00:00:35
[2016-03-24 21:23:08.529]  32  10  72:E4          00:C2:00:00:F4:0F:00:00:00:3B
[2016-03-24 21:23:08.537]  32  10  72:E4          00:C2:00:00:F0:0F:00:00:00:3F
[2016-03-24 21:23:08.544]  32  10  72:E4          00:C2:00:00:F4:FF:FF:00:00:4C
[2016-03-24 21:23:08.552]  32  10  72:E4          00:C2:00:00:F5:DF:FF:00:00:6B
[2016-03-24 21:23:08.559]  32  10  72:E4          00:C2:00:00:FA:EF:FF:00:00:56
[2016-03-24 21:23:08.569]  32  10  72:E4          00:C2:00:00:FE:FF:FF:00:00:42
[2016-03-24 21:23:08.580]  32  10  72:E4          00:C2:00:00:FE:FF:FF:00:00:42
[2016-03-24 21:23:08.593]  32  10  72:E4          00:4F:00:00:6E:00:00:00:00:43
[2016-03-24 21:23:08.600]  32   5  72:E4          00:40:00:6E:52
[2016-03-24 21:23:08.693]  32   5  72:E4          00:40:00:6E:52
[2016-03-24 21:23:08.732]  32  10  72:E4          00:C2:00:00:00:10:00:00:00:2E
[2016-03-24 21:23:08.739]  32  10  72:E4          00:4F:00:00:6E:00:00:00:00:43
[2016-03-24 21:23:08.756]  32  10  72:E4          00:C2:00:00:01:20:00:00:00:1D
[2016-03-24 21:23:08.763]  32  10  72:E4          00:4F:00:00:6E:00:00:00:00:43
```

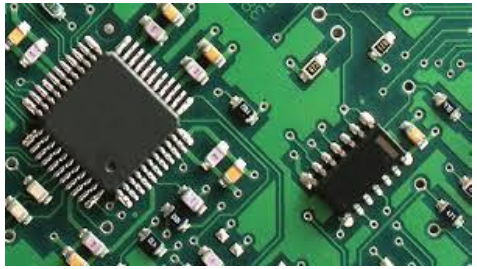
Can We Go Further

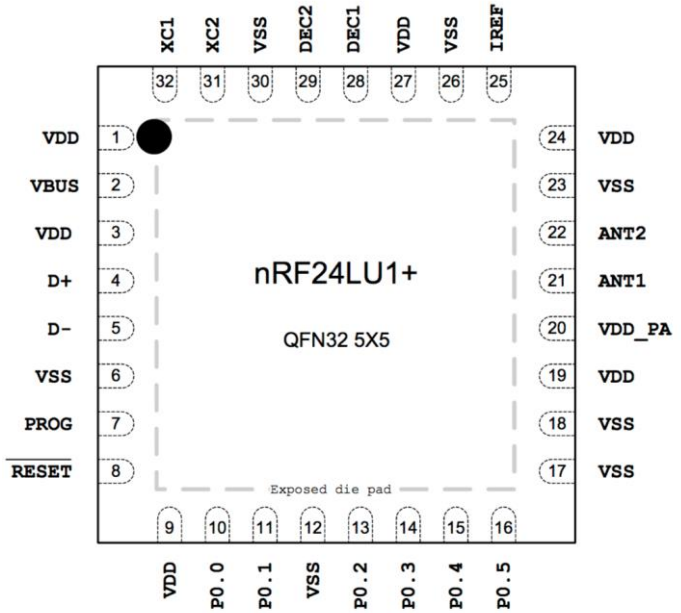
Cheaper Alternative to Crazyradio





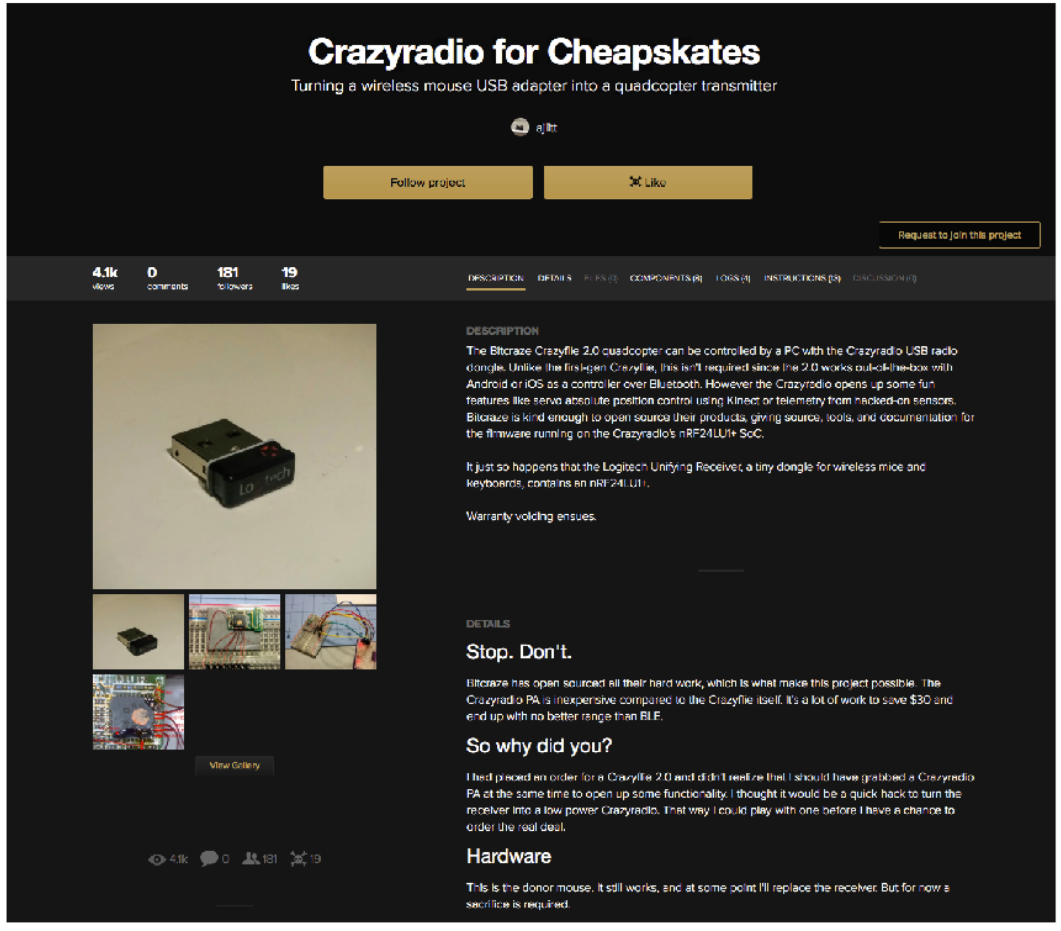
nRF24LU1+ Specification





- nRF24L01 + 2.4 GHz RF transceiver
- Full speed USB 2.0 compliant device controller
- 8-bit microcontroller
- 16 or 32 kilobytes of flash memory
- Up to 12 Mbps air data rate
- Full Spec document in: https://github.com/xwings/tuya

Going Smaller



Crazyradio for Cheapskates

Turning a wireless mouse USB adapter into a quadcopter transmitter

ajlt

Follow project

Like

Request to join this project

4.1k views · 0 comments · 181 followers · 19 likes

DESCRIPTION · DETAILS · FILES (2) · COMPONENTS (6) · LOGS (3) · INSTRUCTIONS (3) · DISCUSSIONS (0)

View Gallery

4.1k · 0 · 181 · 19

DESCRIPTION

The Bitcraze Crazyflie 2.0 quadcopter can be controlled by a PC with the Crazyradio USB radio dongle. Unlike the first-gen Crazyflie, this isn't required since the 2.0 works out-of-the-box with Android or iOS as a controller over Bluetooth. However the Crazyradio opens up some fun features like servo absolute position control using Kinect or telemetry from hacked-on sensors. Bitcraze is kind enough to open source their products, giving source, tools, and documentation for the firmware running on the Crazyradio's nRF24LU1+ SoC.

It just so happens that the Logitech Unifying Receiver, a tiny dongle for wireless mice and keyboards, contains an nRF24LU1+.

Warranty voiding ensues.

DETAILS

Stop. Don't.

Bitcraze has open sourced all their hard work, which is what make this project possible. The Crazyradio PA is inexpensive compared to the Crazyflie itself. It's a lot of work to save $30 and end up with no better range than BLE.

So why did you?

I had placed an order for a Crazyflie 2.0 and didn't realize that I should have grabbed a Crazyradio PA at the same time to open up some functionality. I thought it would be a quick hack to turn the receiver into a low power Crazyradio. That way I could play with one before I have a chance to order the real deal.

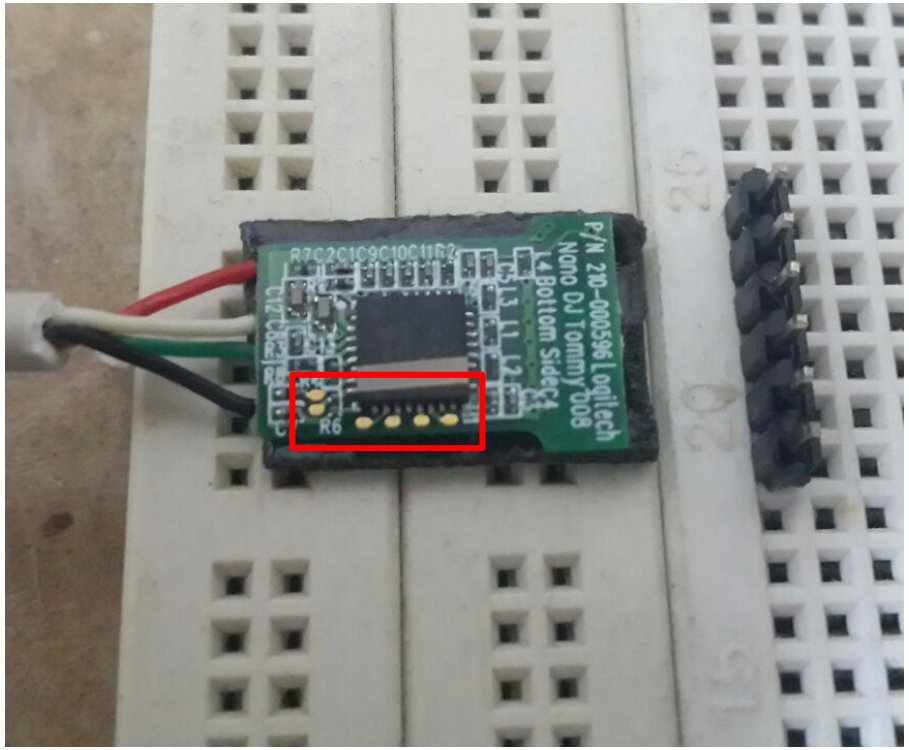
Hardware

This is the donor mouse. It still works, and at some point I'll replace the receiver. But for now a sacrifice is required.

What Are Those

- One for all, all for one
- 25 RMB at taobao

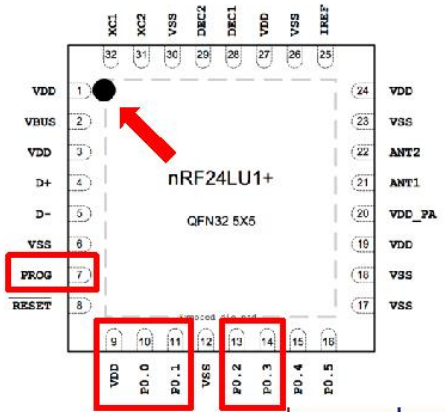
What Is in Logitech Unify Dongle

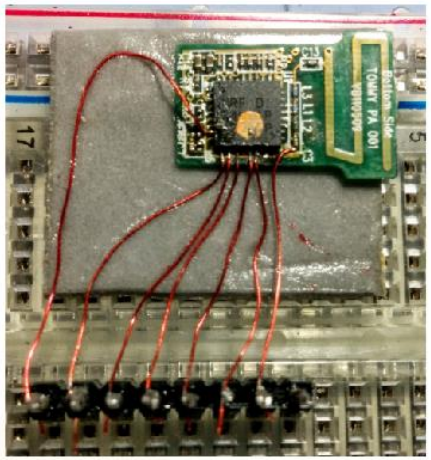


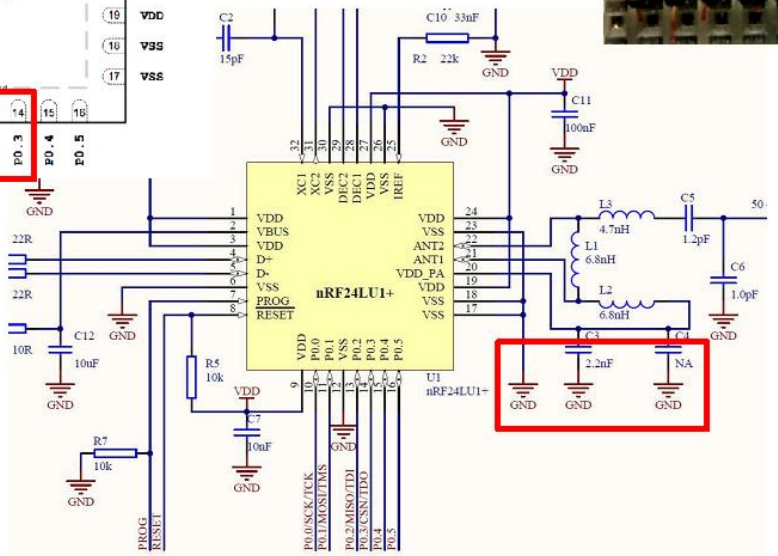
- Open up the casing
- It comes with breakout PINS !
- Find the GRD
- ULTRA STABLE HAND

Identifying The Pin

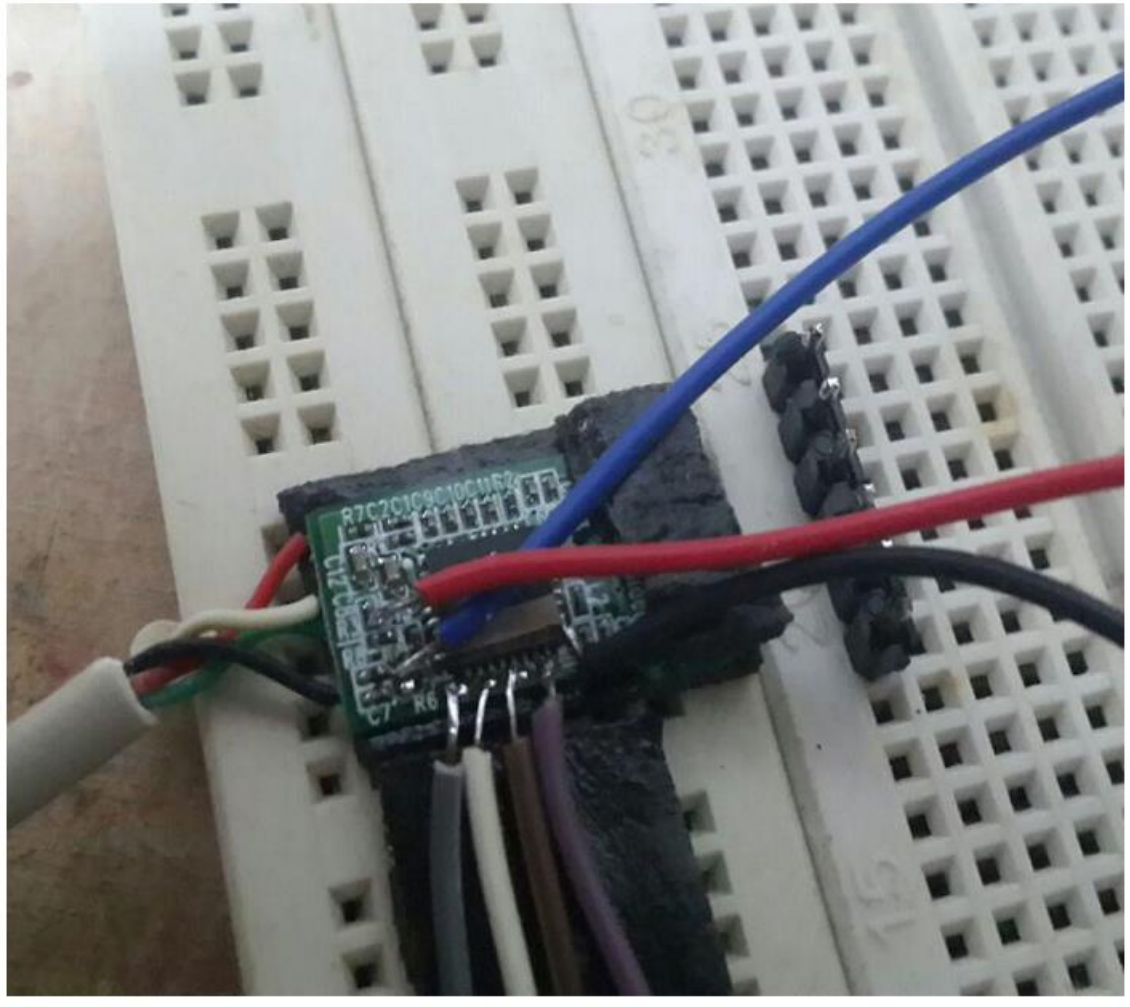
```
#!/usr/bin/perl -w

# Simple perl script to drive the Bus Pirate and unbrick your CrazyRadio dongle.
# Adapted (sorta) from the Bus Pirate example script and mbed NRF24LU1+ flasher projects:
#  http://code.google.com/p/the-bus-pirate/source/browse/trunk/scripts/SPIeeprom.pl
#  http://mbed.org/users/mux/code/nrflash
#
# This script uses the aux output on the Bus Pirate as the PROG pin on the CrazyRadio's NRF24LU1+ chip.
#
# Electrical connections are as follows:
#
#  Bus Pirate          CrazyRadio
#  ==========================
#  MOSI ()       ->  MOSI   (6)
#  MISO ()       ->  MISO   (8)
#  SCK  ()       ->  SCK    (4)
#  CS   ()       ->  CS     (10)
#  AUX  ()       ->  PROG   (2)
#  3V3  ()       ->  3V3    (5)
#  GND  ()       ->  GND    (9)

use strict;
use feature 'say';
use Getopt::Long;
use Device::SerialPort;
use Time::HiRes qw/usleep/;

use constant {
    WREN          => "\x06",
    WRDIS         => "\x04",
    RDSR          => "\x05",
    WRSR          => "\x01",
    READ          => "\x03",
    PROGRAM       => "\x02",
    ERASE_PAGE    => "\x52",
    ERASE_ALL     => "\x62",
    RDFPCR        => "\x89",
    RDISMB        => "\x85",
    ENDEBUG       => "\x86",
    RDYN          => "\x10",
    FLASH_LEN     => 32768,

    BP_CS         => "\x01",
    BP_AUX        => "\x02",
    BP_PULLUP     => "\x04",
    BP_POWER      => "\x08",
};

my %opts;
my $port;
my $time = 500;
my $status_byte;
my $return;
```

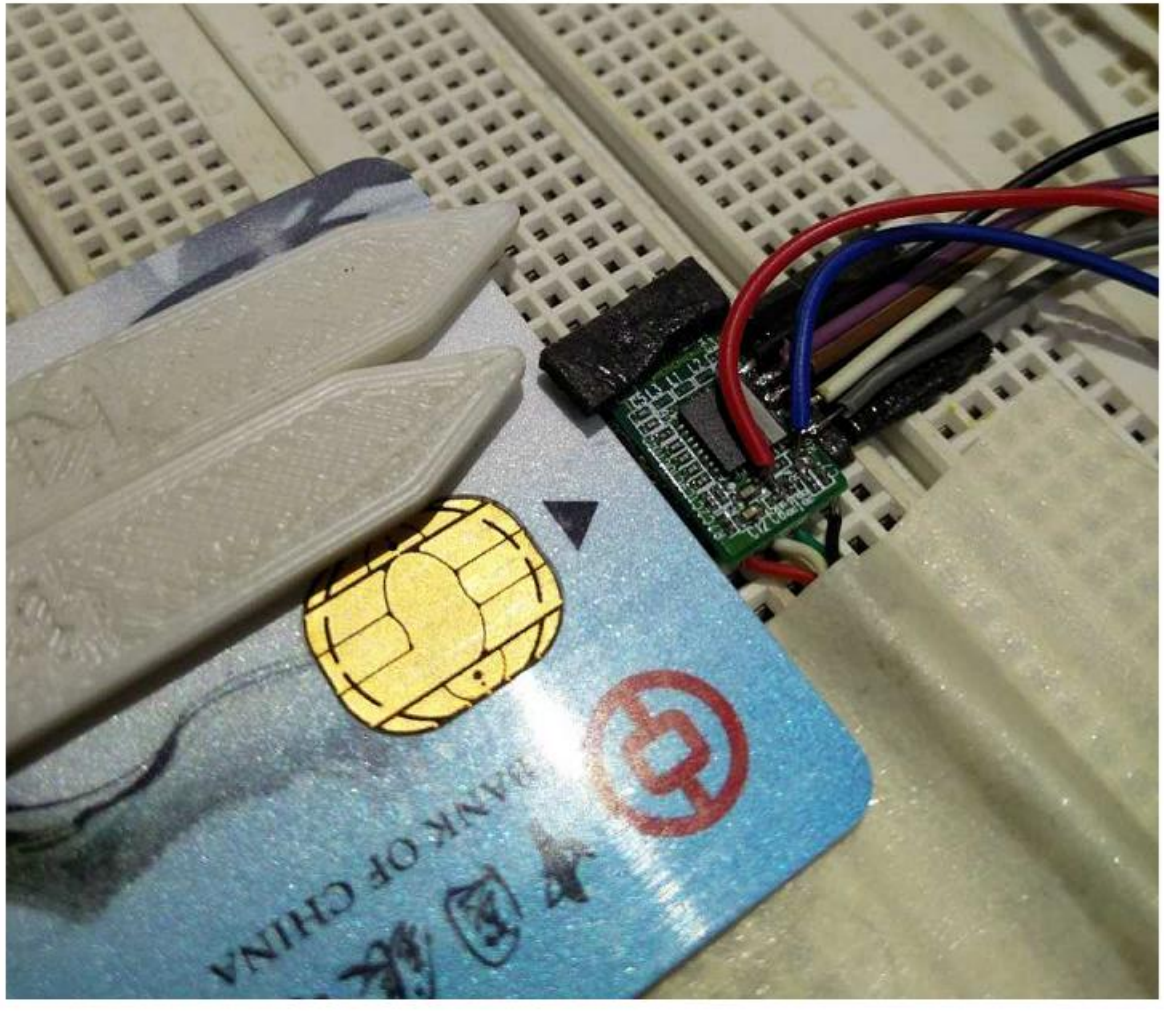




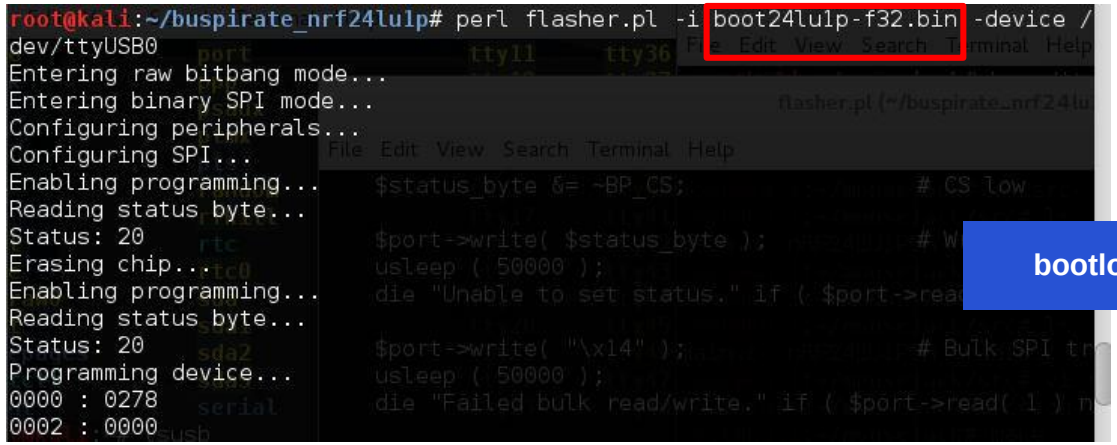


Soldering Onto The Pin

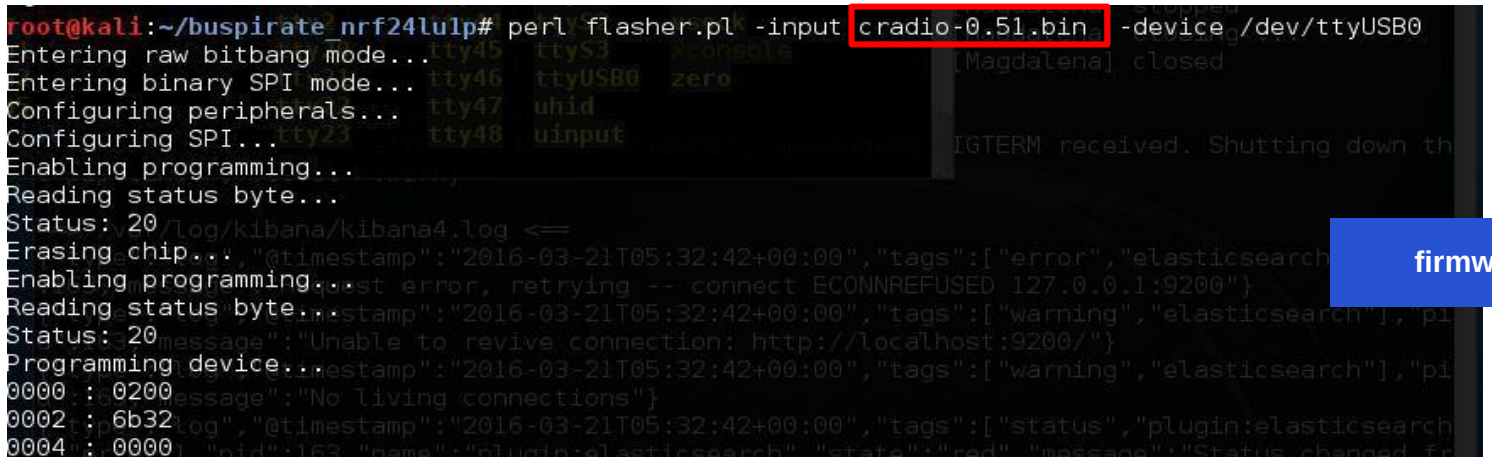




Flashing Into Logitech



bootloader



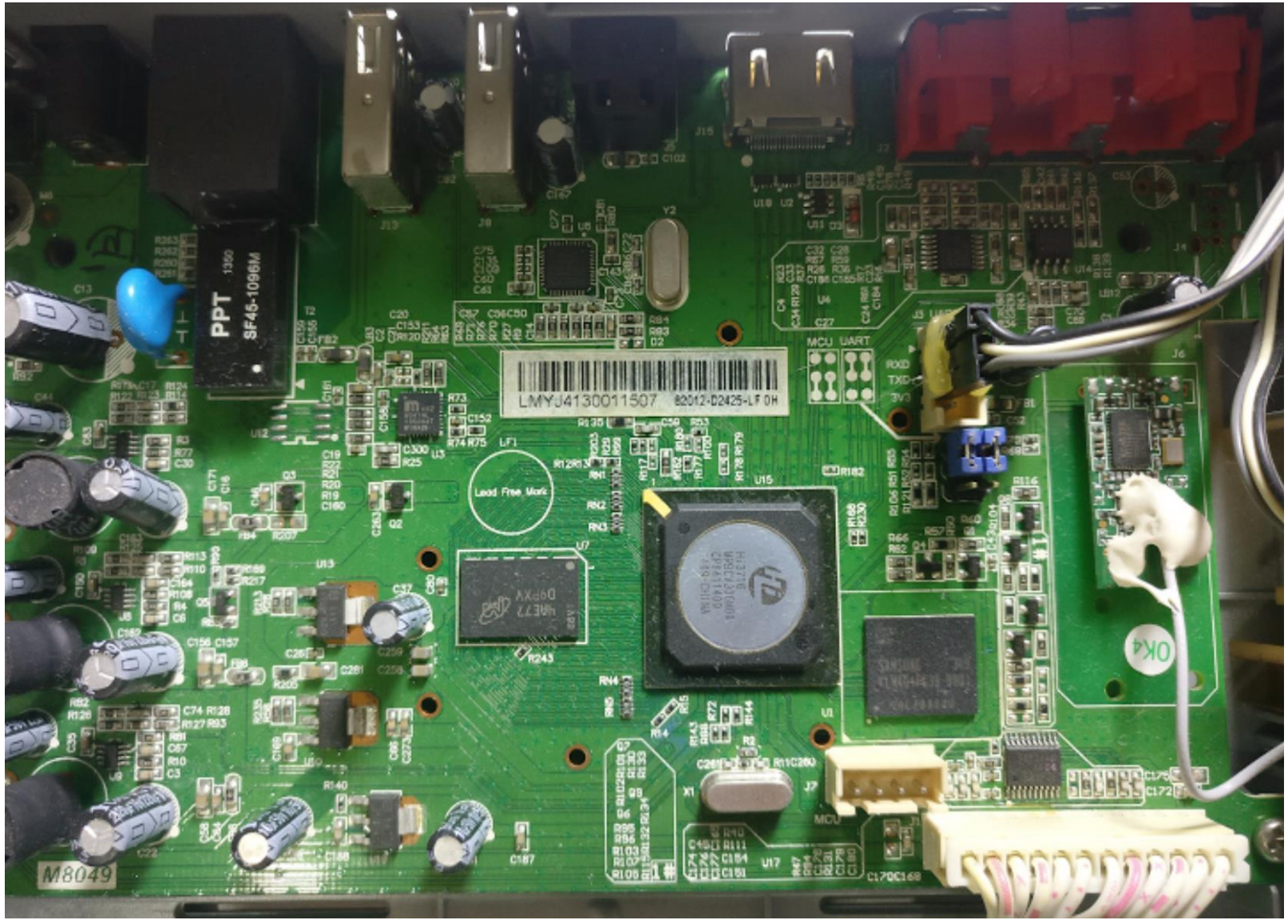
firmware

- Two hours for the bootloader
- Two hours for the crazyradio firmware
- Two hours for the mousejack firmware

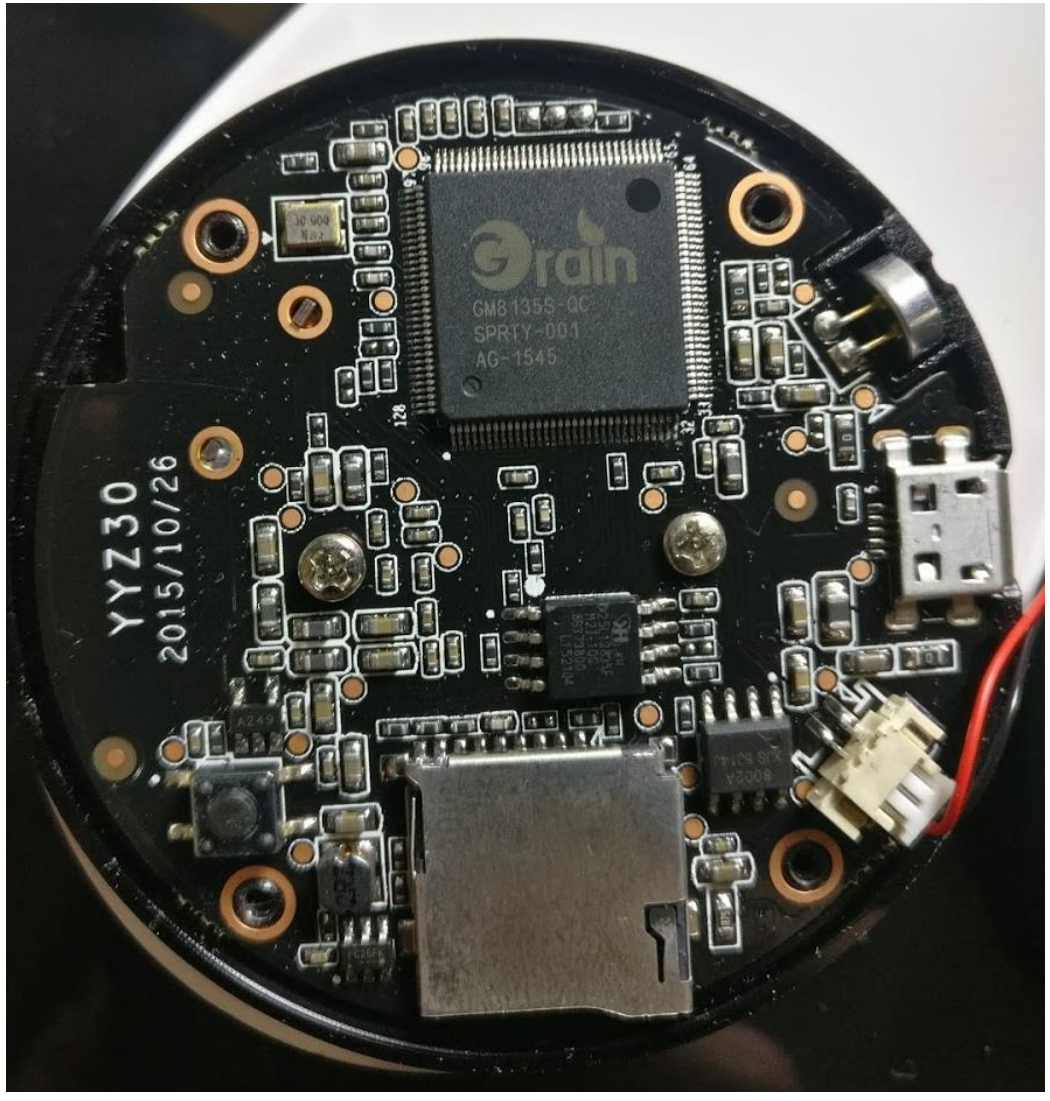
Questions and Break

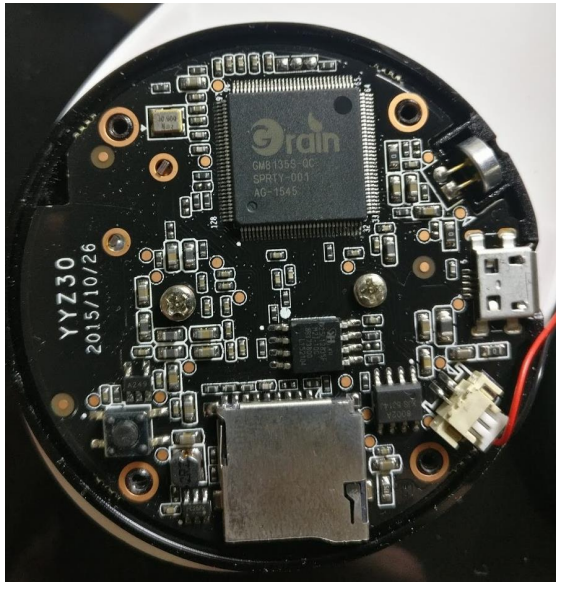
Revision

UART or Not


PPT
SF45-1096M
LMYJ4130011507
MCU UART
RXD
TXD
3V3
Lead Free Mark
OK4
M8049

Spot No UART





2.2 Pin Assignments

Figure 2-1through Figure 2-4 show the pin assignments of GM8136S/GM8135S.

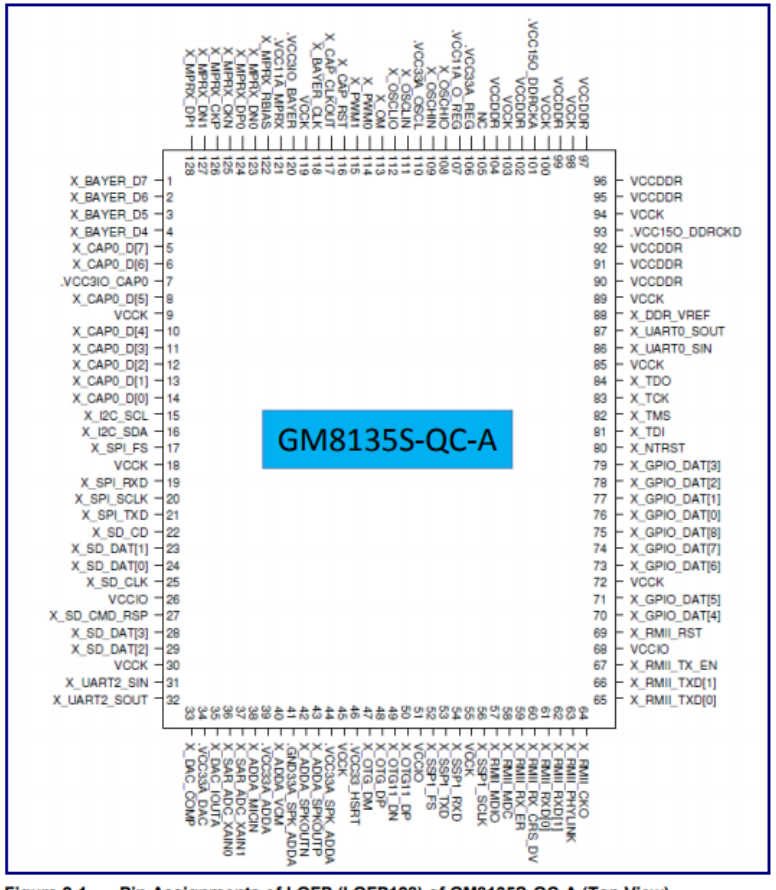
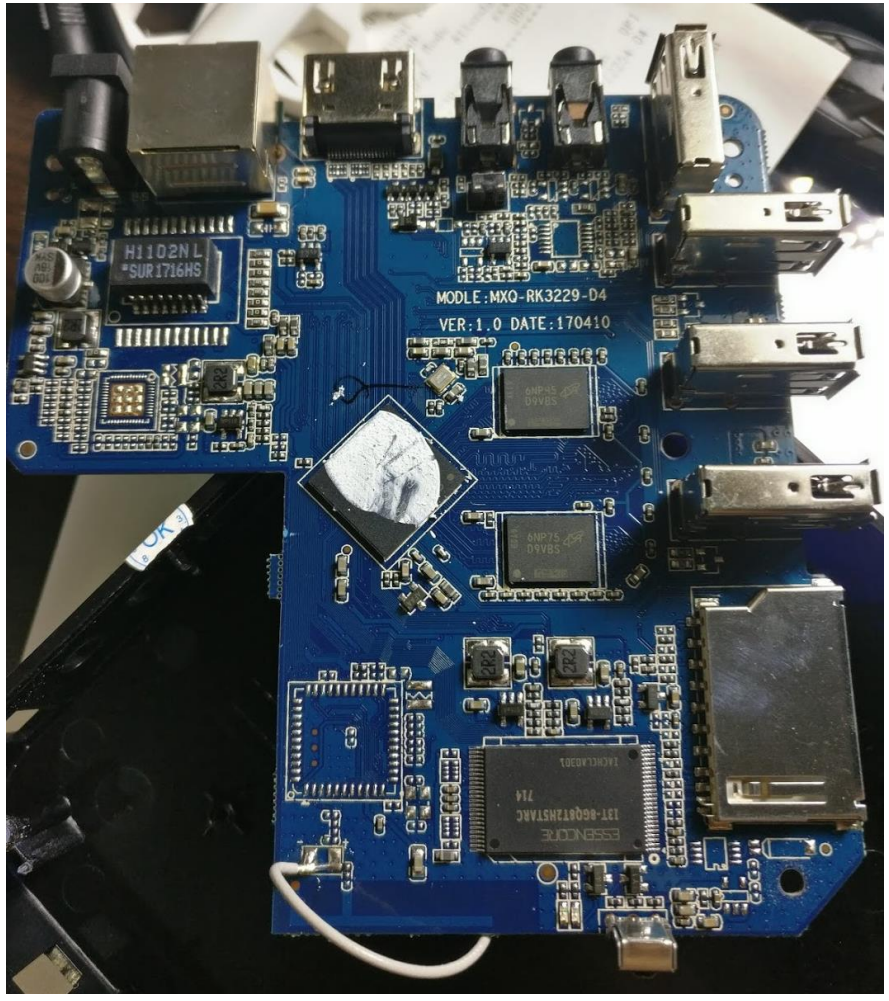
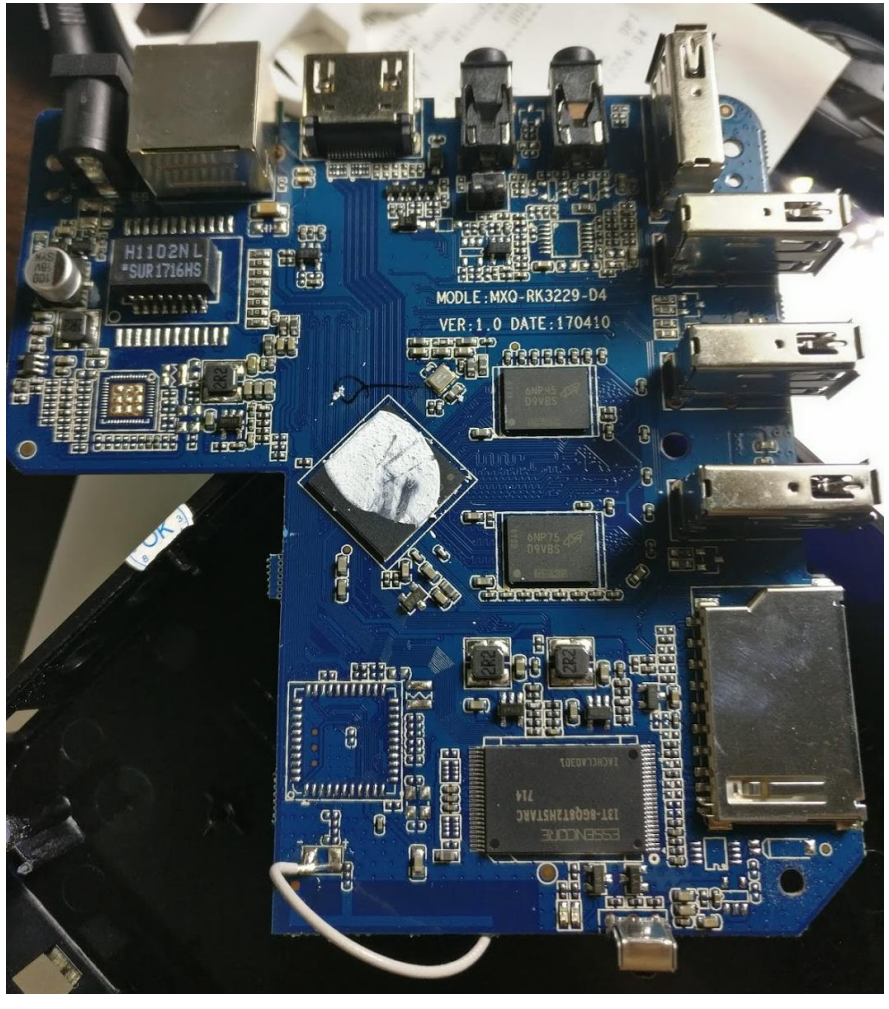


Figure 2-1. Pin Assignments of LQFP (LQFP128) of GM8135S-QC-A (Top View)

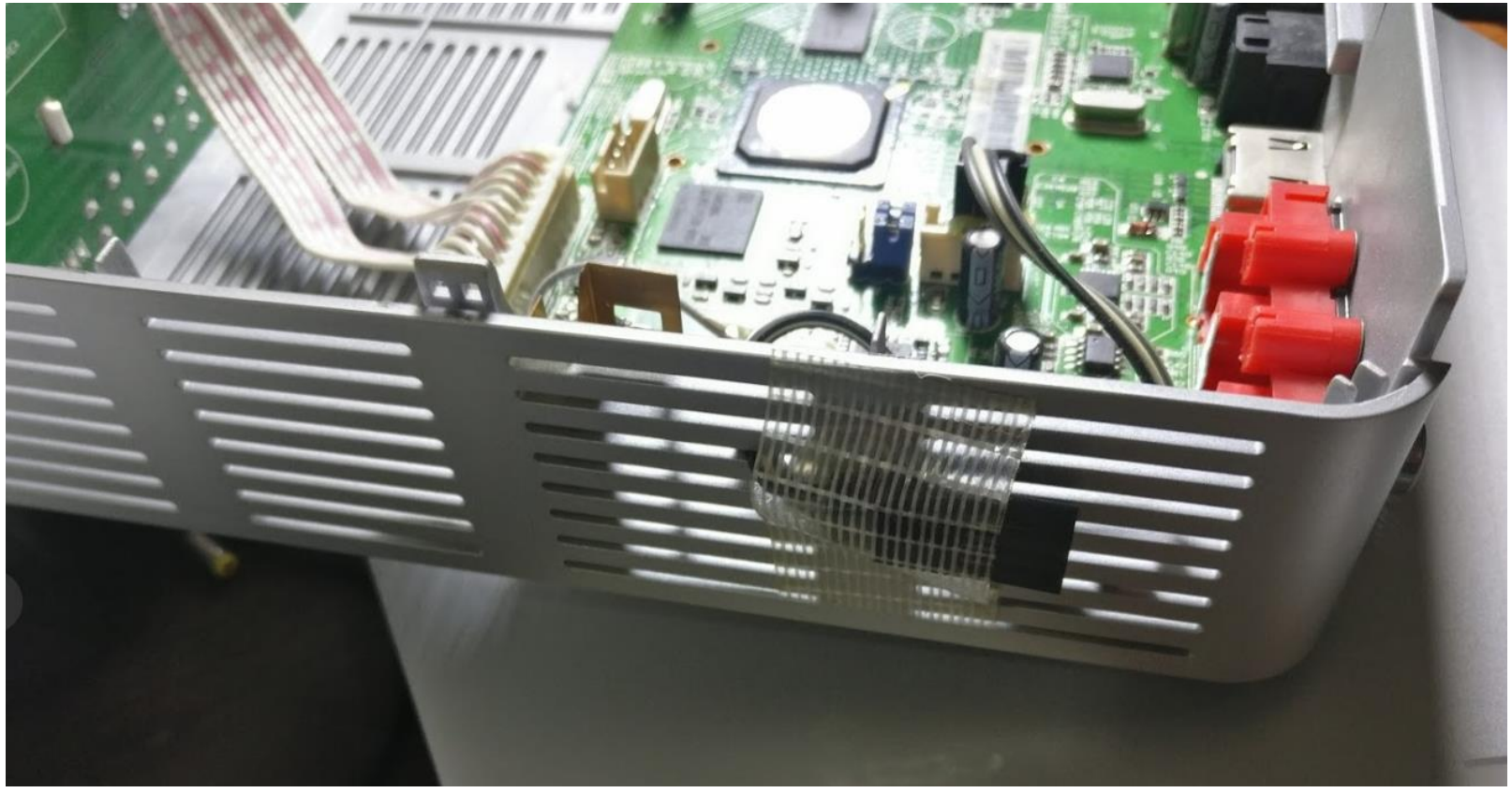
Finding The UART



Finding The UART



Always Make a Backdoor For Yourself



Case Study

Buying a China Only Cam

小蚁智能摄像机1080p一代升级版高清夜视手机网络监控摄像头无线

菜鸟发货 只换不修

天猫电器城 正 快 省 新 闪电到家 超值包邮

全球3C家电狂欢周 此商品8月14日开卖，请提前加入购物车

天猫 购物券 全天猫实物商品通用 去刮券

专柜价 ~~¥ 169.01 219.01~~

价格 ¥169.00-219.00

运费 浙江嘉兴 至 杭州 上城区 清波街道 快递 0.00

次日达·菜鸟联盟 24:00前付款，承诺8月13日送达

月销量 8567 | 累计评价 25315 | 送天猫积分 16起

颜色分类 1080p智能摄像机一代升级版 | 1080p智能摄像机一代升级版+16G内存卡 | 1080p智能摄像机一代升级版+30天云存储充值卡

数量 1 件 库存49件

立即购买 加入购物车

分享 收藏商品（39319人气） 举报

服务承诺 超值包邮 闪电到家 正品保证 只换不修 极速退款 赠运费险 七天无理由退换 支付方式

Talking Cam's Warming

Yi2 1080p camera doesn't work anymore outside of China :(· Issue #9 ...
https://github.com/niclet/yi-hack-v2/issues/9
Dec 17, 2016 - Xiaomi **Yi** Ants **Camera** 2 hack. Contribute to **yi**-hack-v2 development by creating an account on GitHub.

How to use Yi Home Camera 2 (1080p) outside of China | Mientras ...
tomascrespo.sofiytommy.com/how-to-use-yi-home-camera-2-1080p-outside-of-china/
Jun 19, 2016 - How to use **Yi** home **camera** 2 outside of **China**. I've been using a **Yi** Homme **Camera** for a long (aka Xiaomi/Xiaoyi Small Ants **Camera**).

Images for yi cam 1080p China

More images for yi cam 1080p China Report images

Xiaomi Yi Action Camera - Chinese vs International Version (black ...

https://www.youtube.com/watch?v=EA0l0jxU21I
Mar 14, 2016 - Uploaded by el Producente
Review & Unboxing of both versions of the Xiaomi **Yi** Action **Camera**. International Version (black): http ...

yi home camera 2 1080p problem (outside china?) - YouTube

https://www.youtube.com/watch?v=nvAeWJ-q9D4
May 23, 2016 - Uploaded by Ricardo Molina
i got several of this **cameras** like a month ago and i can not make them to work and also it gets super hot when ...

How to fix Xiaoyi "This Camera can only be used within China" English ...

https://www.youtube.com/watch?v=SsAMklqUZLQ
Apr 26, 2016 - Uploaded by Momo
Turn on the **camera** and hold down the reset button for 6-8 seconds 2. ... I am trying to Downgrade the ...

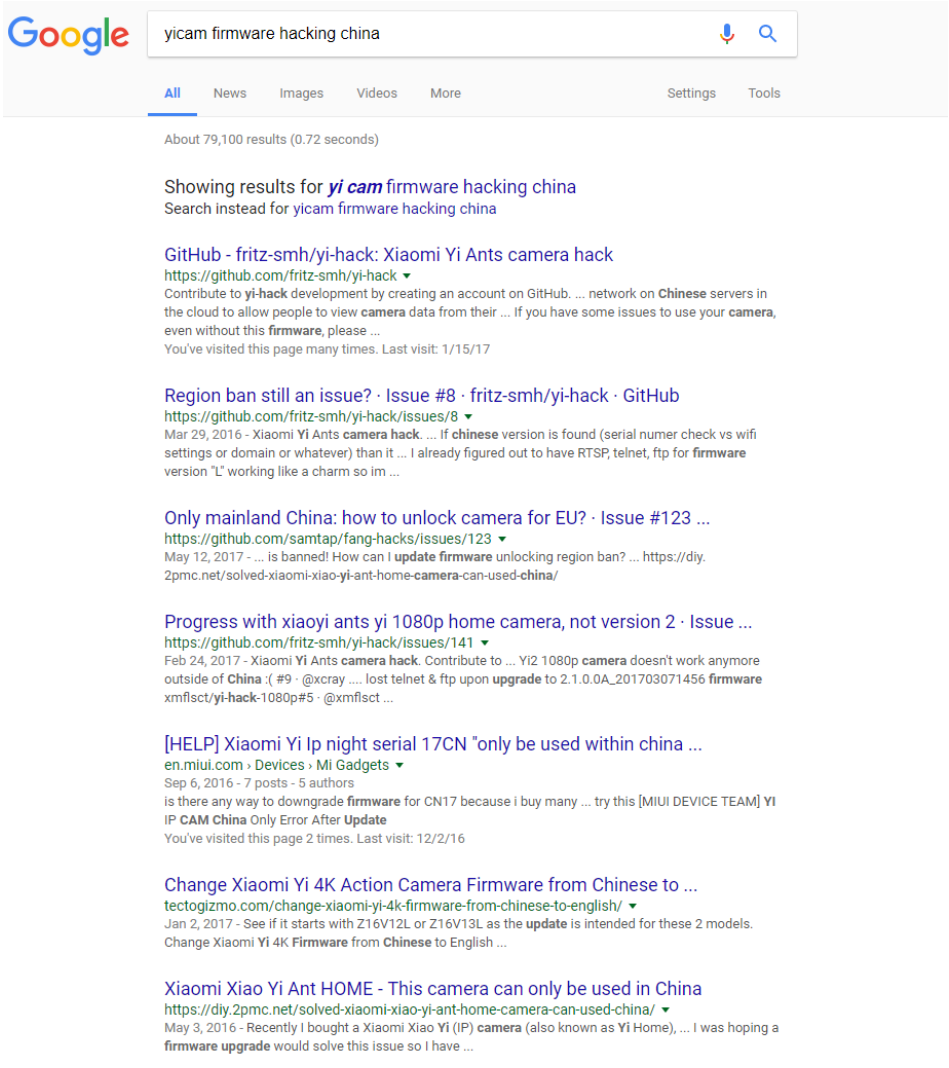
Xiaomi Yi Action Camera Chinese vs International Version - 1080p 60fps

https://www.youtube.com/watch?v=anwr_8JIB1g
Mar 14, 2016 - Uploaded by el Producente
Demo Footage, Comparison and side-by-side video in **1080p** with 60fps. International Version: http://bit.ly ...

YI | See Everything
https://www.yitechnology.com/
See everything with **YI** - VR **camera**, 360 **camera**, mirrorless **camera**, action **camera**, drone, home **camera**, and dash **camera**. Shop now!

"Not Allow To Use Outside China"

Answer from Google and Baidu



Hacking Started

[SOLVED] Xiaomi Xiao Yi Ant HOME — This camera can only be used in China (1.8.6.1)

In IT DIY Tags firmware, hack, pentesting May 3, 2016 Csaba Peter

Recently I bought a Xiaomi Xiao Yi (IP) camera (also known as Yi Home), Chinese version. The camera looks nice, the picture quality is ok, and worked fine on my local Wifi.

However, I was unfortunate enough to receive and test the camera when Xiaomi decided to deny access from the iOS app to the camera outside of China (error 5400). I was hoping a firmware upgrade would solve this issue so I have upgraded from 1.8.5.1L to 1.8.6.1B. Now my camera was useless. The camera would say "This camera can only be used in China" and would shut down.

This was the tipping point when I have decided I will investigate what's happening with this camera and what can be done to make it functional again. At the time of writing the remote access (error 5400) has been solved by the provider so no additional action is required. (I tried to convert a CN camera to international one by changing the serial of the device, but couldn't test from a European or US IP and probably I would have needed access to the system files of a functional international camera to compare)

So the remaining issue was the camera shut down with the latest firmware (tested with 1.8.6.1A and 1.8.6.1B).

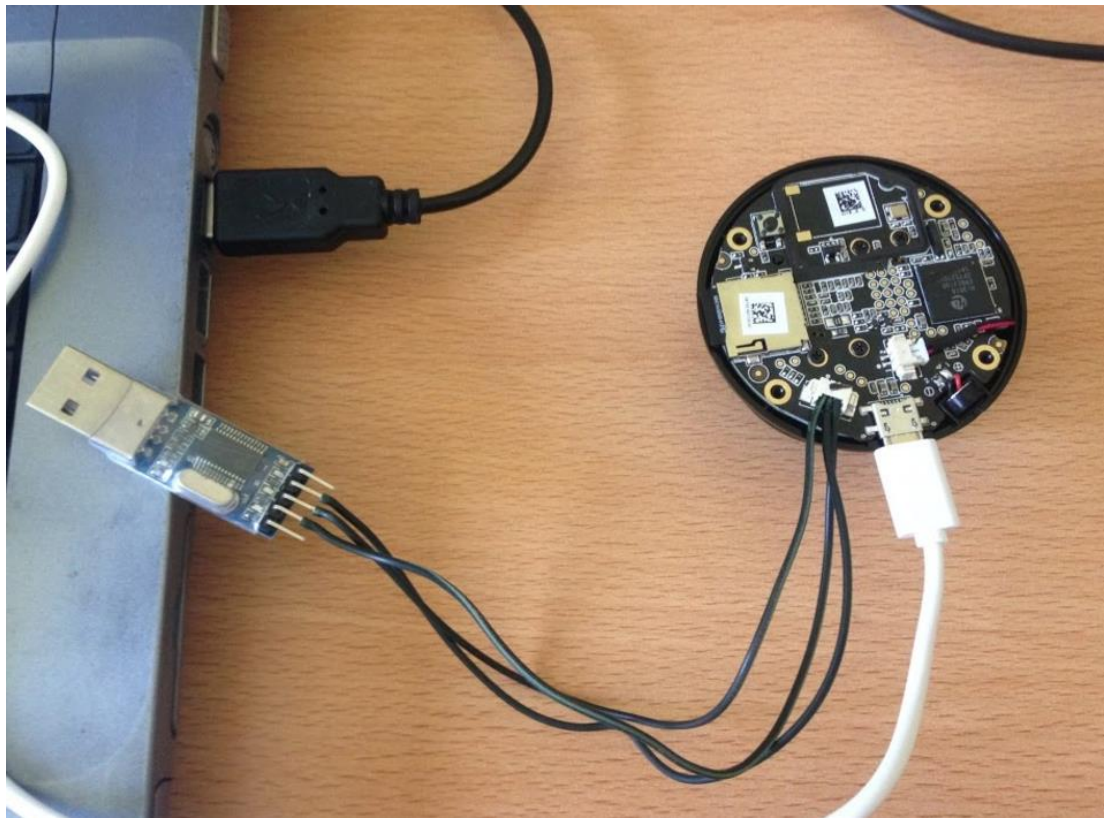
If you do a search there are heaps of websites describing how you can gain access to the camera and ultimately enable remote access via telnet. I won't get into those details, you can check some of the websites I listed below.

Once you logged into the camera via telnet the fun part begins. The camera is running a Linux version.

```
# uname -a
Linux (none) 3.0.8 #1 Wed Apr 30 16:56:49 CST 2014 armv5tejl GNU/Linux
```

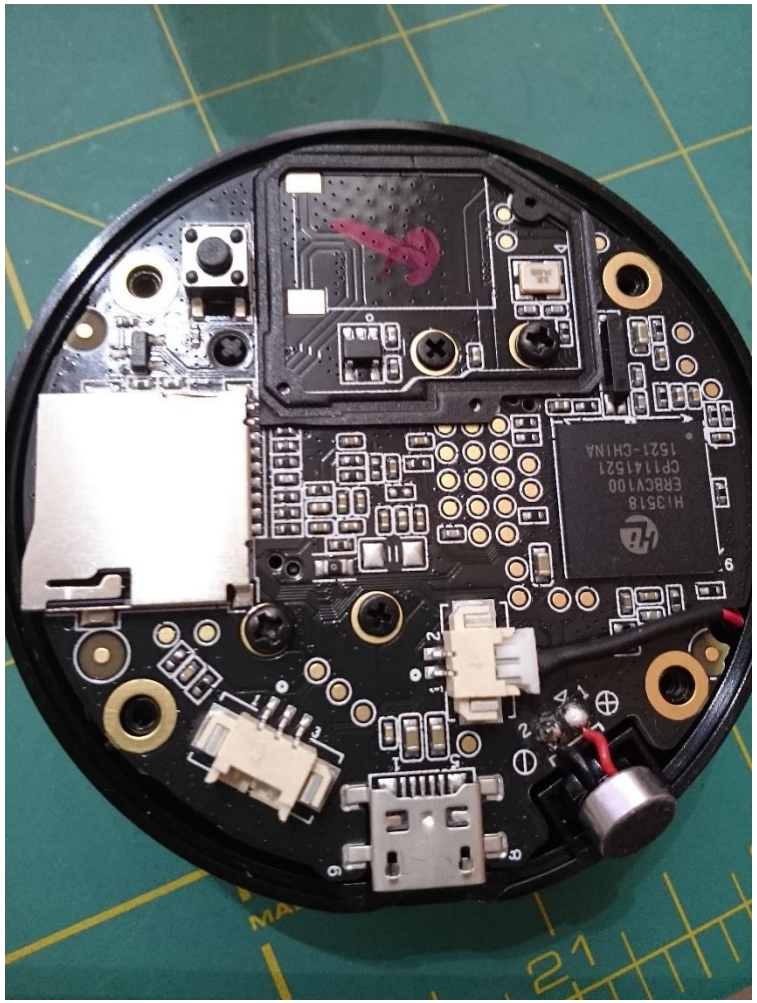
- 17CN 1.8.6.1R_201611191201
- Not downgrade able
- Not down gradable could be a bug

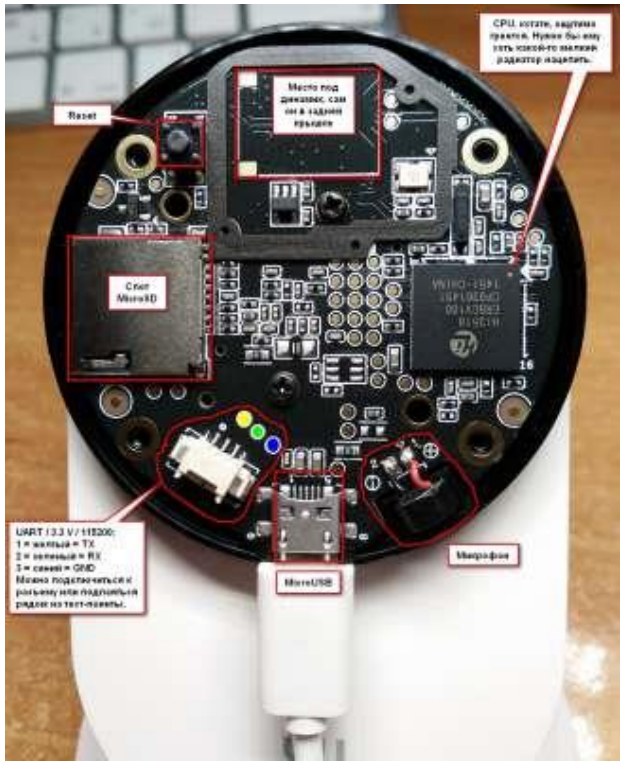
Try to Connect to USB TTL



- Power
- USB TTL
- No Way To Get Near USB TTL

Solving Puzzle





- Finding GND
- Guessing RX TX
- Multi meter

What To We Want To Archive

Network settings

> /etc/init.d # cat /home/conf/wpa_supplicant.conf

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
network={
ssid="MY_WIFI_L4H"
scan_ssid=1
proto=WPA RSN
key_mgmt=WPA-PSK
pairwise=CCMP TKIP
group=CCMP TKIP
psk="my_PASSWORD_l4h"
}
```

- Work without Xiaomi app
- Turn on WiFi while Boot
- Turn on telnet while boot
- Turn on ftp while boot
- Turn RTSP whole boot

Enabling Services

Bring up some services

> /etc/init.d # cat S88telnet

```
#!/bin/sh
/home/app/telnetd &
(sleep 10; /home/base/tools/wpa_supplicant -iwlan0 -c/home/conf/wpa_supplicant.conf) &
(sleep 20; /sbin/ifconfig wlan0 192.168.0.100 netmask 255.255.255.0) &
```

> /etc/init.d # cat S89ftp

```
#!/bin/sh
/home/app/tcpsvd -vE 0.0.0.0 21 ftpd -w / &
```

RTSP returns segmentation fault

Fire up IDA pro and look at the RTSP Binary, we found few files requred before it can run, so this is how we fix it.

```
ln -s /tmp/hd1 /home/hd1
ln -s /tmp/hd2 /home/hd2
ln -s /tmp /home/mmap_tmpfs
mkdir /home/jrview
ln -s /home/app/busybox /bin/renice
ln -s /home/lib/libcrypt-0.9.32.1.so libcrypt.so.0
ln -s /home/lib/libstdc\+\+.so.6.0.12 libstdc++.so.6
```

Forgotten to mount FS after boot

```
i2c /dev entries driver
hisi_i2c hisi_i2c.0: Hisilicon [i2c-0] probed!
hisi_i2c hisi_i2c.1: Hisilicon [i2c-1] probed!
hisi_i2c hisi_i2c.2: Hisilicon [i2c-2] probed!
TCP: cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 17
NET: Registered protocol family 15
lib80211: common routines for IEEE802.11 drivers
Registering the dns_resolver key type
VFS: Mounted root (jffs2 filesystem) on device 31:4.
Freeing init memory: 112K
Kernel panic - not syncing: No init found.  Try passing init= option to kernel. See Linux Documentation/init.txt for
```

Back To Data Sheet

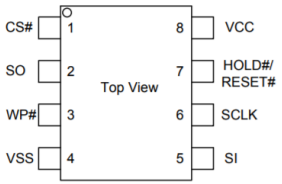
GD25Q128CxIGx 3.3V Uniform Sector Dual and Quad Serial Flash

http://www.elm-tech.com

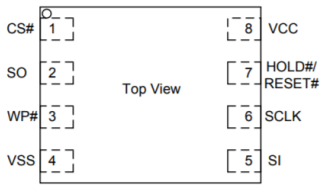
. GENERAL DESCRIPTION

The GD25Q128C(128M-bit) Serial flash supports the standard Serial Peripheral Interface (SPI), and supports ıe Dual/Quad SPI: Serial Clock, Chip Select, Serial Data I/O0 (SI), I/O1 (SO), I/O2 (WP#) and I/O3 (HOLD#/ :ESET#). The Dual I/O data is transferred with speed of 208Mbits/s and the Quad I/O & Quad Output data is ransferred with speed of 320Mbits/s.

Connection Diagram



8-LEAD SOP

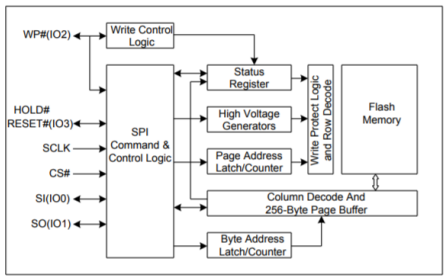


8-LEAD WSON

in Description

Pin Name	I / O	Description
CS#	I	Chip Select Input
SO (IO1)	I/O	Data Output (Data Input Output 1)
WP# (IO2)	I/O	Write Protect Input (Data Input Output 2)
VSS		Ground
SI (IO0)	I/O	Data Input (Data Input Output 0)
SCLK	I	Serial Clock Input
HOLD#/RESET (IO3)	I/O	Hold or Reset Input (Data Input Output 3)
VCC		Power Supply

lock Diagram



- sdcard Is not readable while boot

Analyzing The Actual Firmware

XiaoYI Ants unofficial info page

HOME INSTRUCTIONS FIRMWARES BUY A YI

Firmwares

Hardware version v2.1 needs a firmware version 1.8.5.1K or higher!
You can find the how to on the firmware flash instruction page.
Note: flash firmware is at your own risk!

Original for CN hardware

- 1.8.5.1B_201513211614
- 1.8.5.1H_201505211709
- 1.8.5.1J_201507201424
- 1.8.5.1K_201508311131
- 1.8.5.1L_201506291725
- 1.8.5.1M_201512011815
- 1.8.5.1N_201512212009
- 1.8.6.1A_201602241619
- 1.8.6.1B_201603181307

Original for international hardware

- 1.8.5.1N_201601071352

Modified for CN hardware

Additional features are added to this firmwares (RTSP, FTP, telnet, timezone, ...)
How to use the different additional features is described on the instruction page.

- 1.8.5.1B_rtsp
- 1.8.5.1J_easy_boot
- 1.8.5.1K_rtspfix-v3
- 1.8.5.1L_rtspfix-v3
- 1.8.5.1M_rtspfix-v4
- 1.8.6.1B_rtspfix

Branch: master ▾ yi-hack-v3 / src / Create new file

shadow-1 Fixed errors in startup scripts.

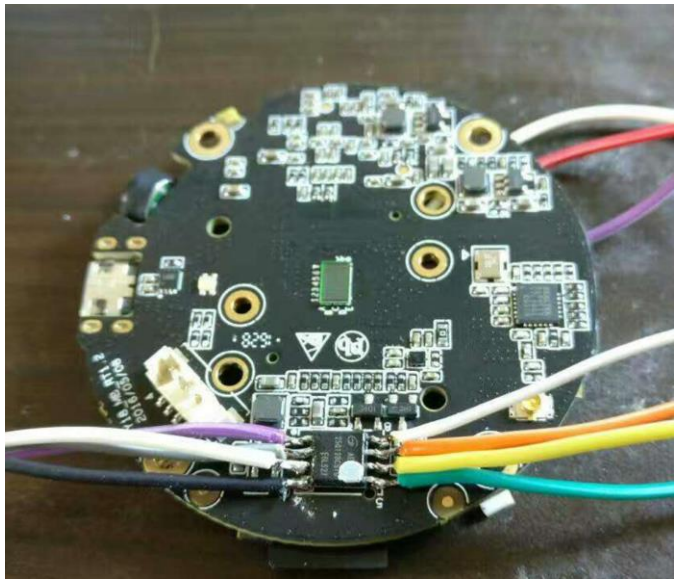
..

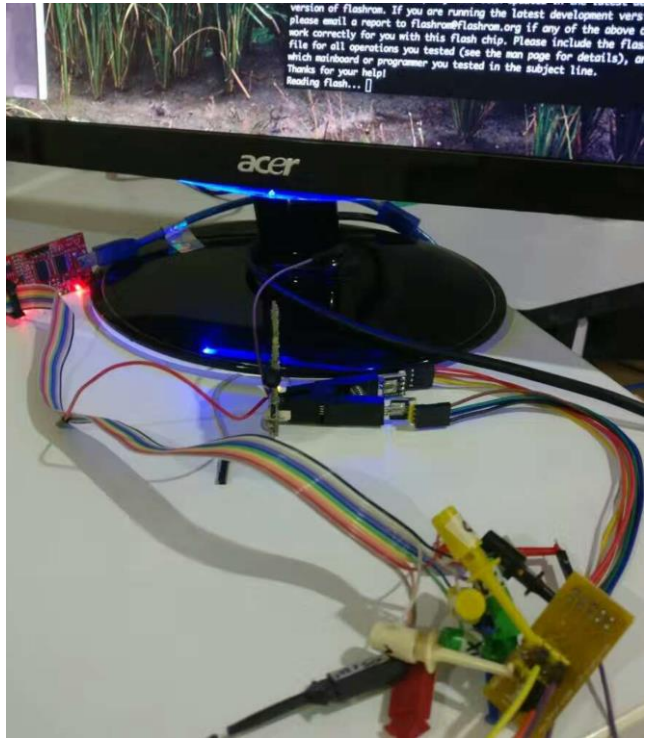
busybox	Added ability to randomly select the number of proxy servers to downl...
home/yi-hack-v3	Fixed errors in startup scripts.
libwebsockets-plugins	Firmware no longer affected by Xiaomi updates.
libwebsockets	Firmware no longer affected by Xiaomi updates.
proxychains-ng	Firmware no longer affected by Xiaomi updates.
rootfs/etc	Fixed errors in startup scripts.
uClibc	Initial tested version of the firmware for Yi 1080p Dome camera.

Understanding dmesg

```
brd: module loaded
Check Flash Memory Controller v100 ...  Found.
SPI Nor(cs 0) ID: 0xc8 0x40 0x18
Block:64KB Chip:16MB Name:"GD25Q128"
SPI Nor total size: 16MB
8 cmdlinepart partitions found on MTD device hi_sfc
8 cmdlinepart partitions found on MTD device hi_sfc
Creating 8 MTD partitions on "hi_sfc":
0x000000000000-0x000000040000 : "boot"
0x000000040000-0x000000050000 : "env"
0x000000050000-0x000000060000 : "conf"
0x000000060000-0x0000001f0000 : "os"
0x0000001f0000-0x000000330000 : "rootfs"
0x000000330000-0x000000fe0000 : "home"
0x000000fe0000-0x000000ff0000 : "vd1"
0x000000ff0000-0x000001000000 : "ver"
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
hiusb-ehci hiusb-ehci.0: HIUSB EHCI
hiusb-ehci hiusb-ehci.0: new USB bus registered, assigned bus number 1
hiusb-ehci hiusb-ehci.0: irq 15, io mem 0x100b0000
hiusb-ehci hiusb-ehci.0: USB 0.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
i2c /dev entries driver
hisi_i2c hisi_i2c.0: Hisilicon [i2c-0] probed!
hisi_i2c hisi_i2c.1: Hisilicon [i2c-1] probed!
hisi_i2c hisi_i2c.2: Hisilicon [i2c-2] probed!
```

Dumping The Firmware





- Making sure the firmware is the same with the one on the internet

Debug and Patch

Extract !

Taking Partition Notes

Partition by size, take from the boot log

```
0x000000000000-0x000000040000 : "boot"
0x000000040000-0x000000050000 : "env"
0x000000050000-0x000000060000 : "conf"
0x000000060000-0x0000001f0000 : "os"
0x0000001f0000-0x000000330000 : "rootfs"
0x000000330000-0x000000fe0000 : "home"
0x000000fe0000-0x000000ff0000 : "vd1"
0x000000ff0000-0x000001000000 : "ver"
```

Dump using bus pirate

```
flashrom -p buspirate_spi:dev=/dev/ttyUSB0 -c GD25Q128C -r yicam_night_GD25Q128C.bin -V -f
```

Spliting the image

This is how you split the file according to partition size

```
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_bootloader.bin bs=1 count=$((0x040000))
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_env.bin bs=1 count=$((0x050000-0x040000)) skip=$((
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_conf.bin bs=1 count=$((0x060000-0x050000)) skip=$(
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_os.bin bs=1 count=$((0x1f0000-0x060000)) skip=$((0
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_rootfs.bin bs=1 count=$((0x330000-0x1f0000)) skip=
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_home.bin bs=1 count=$((0xfe0000-0x330000)) skip=$(
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_vd1.bin bs=1 count=$((0xff0000-0xfe0000)) skip=$((
dd if=yicam_night_test_GD25Q128C.bin of=yicam_night_test_GD25Q128C_ver.bin bs=1 count=$((0x1000000-0xff0000)) skip=$(
```

Extract JFFS

TL;DR

Here's a quick overview of the entire mounting process:

1. Extract the JFFS2 file system image from the U-Boot image:
   `uImage.py -x home`
2. Pad the JFFS2 image to make it work with `block2mtd`:
   `./jffs2.py --pad=0 7518-hi3518-home`
3. Load the kernel modules:
   `modprobe block2mtd mtdblock`
4. Setup the loopback device:
   `losetup /dev/loop0 7518-hi3518-home`
5. Associate loopback device with MTD device
6. Mount the MTD device (finally)

If all this seems tedious, I wrote a `mount-jffs2` shell script that performs steps 3 to 6. You just need to specify the (padded) image file, mount point and block size:

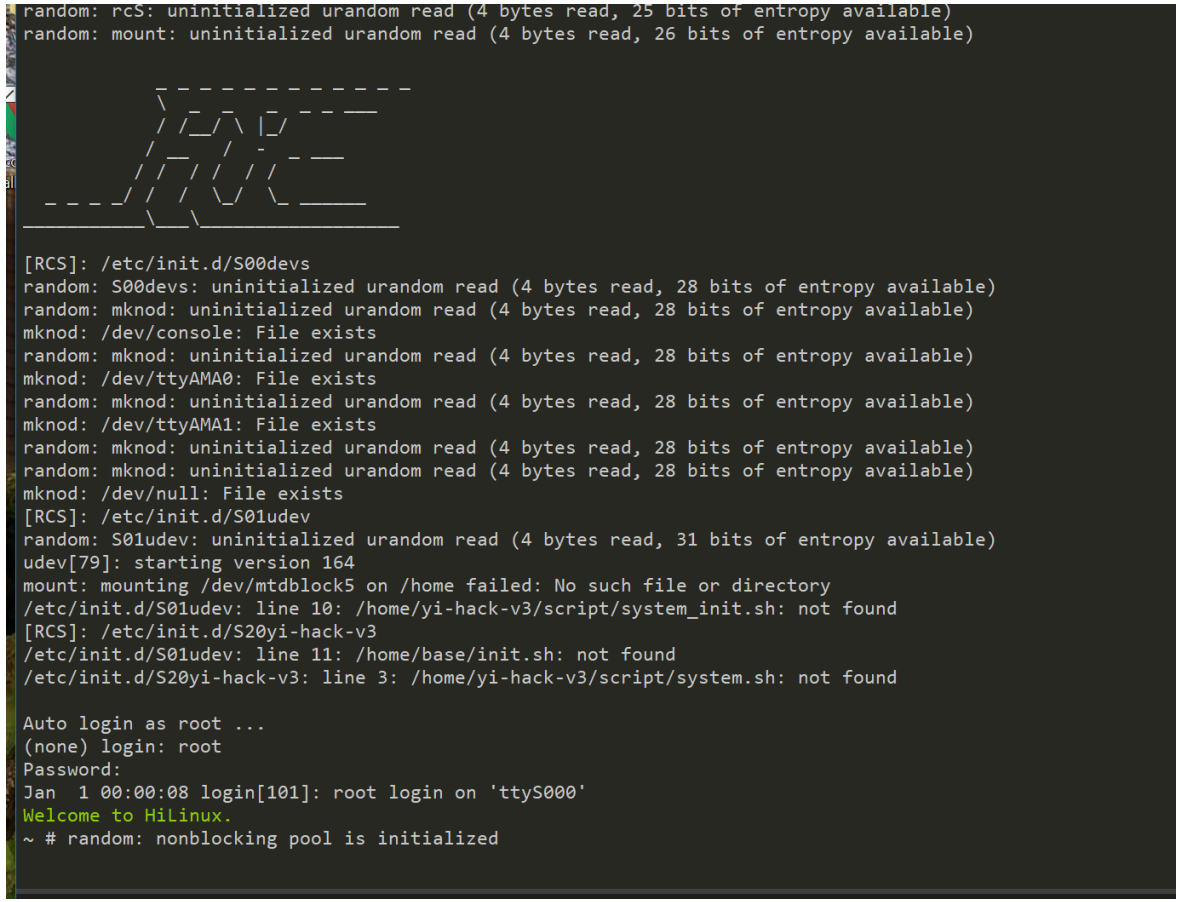
```
./mount-jffs2 7518-hi3518-home /mnt/image 64KiB
```

Making The Firmware

```
(23:52:06):xwings@kali32:<~/yicam_home_720p/yi-hack-v3/rootfs_mount>
(117)$ ls -alF
total 60
drwxr-xr-x 15 root   root   4096 Jan  1  1970 ./
drwxr-xr-x  5 xwings xwings 4096 Aug 15 23:11 ../
drwxr-xr-x  2 root   root   4096 Jul  2 22:34 bin/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 dev/
drwxr-xr-x  4 root   root   4096 Jul  2 22:24 etc/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 home/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 lib/
lrwxrwxrwx  1 root   root     11 Jul  2 22:34 linuxrc -> bin/busybox*
drwxr-xr-x  3 root   root   4096 Jul  2 22:24 mnt/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 proc/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 root/
drwxr-xr-x  2 root   root   4096 Jul  2 22:34 sbin/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 sys/
drwxr-xr-x  2 root   root   4096 Jul  2 22:24 tmp/
drwxr-xr-x  4 root   root   4096 Jul  2 22:34 usr/
drwxr-xr-x  3 root   root   4096 Jul  2 22:24 var/
(23:52:08):xwings@kali32:<~/yicam_home_720p/yi-hack-v3/rootfs_mount>
```

- # qemu-img create test.img 1024M
- # mkfs.ext2 –F test.img
- # mount –t ext2 –o loop,rw test.img /mnt/test
- Copy all files
- umount

Test Booting with QEMU



- /home/xwings/qemu-2.9.0/arm-softmmu/qemu-system-arm -cpu arm1176 -M versatilepb -kernel /home/xwings/yicam_home_720p/testrun/kernel-qemu-4.4.34-jessie -append "console=ttyAMA0 root=/dev/sda rootfstype=ext2 rw" -hda /home/xwings/yicam_home_720p/yi-hack-v3/rootrootfs.img -nographic

Firmware Repacking

Mount, Edit and Pad

Look for JFFS mounting tutorial, make all the changes you need Just In case you need padding before mergeing the ROM

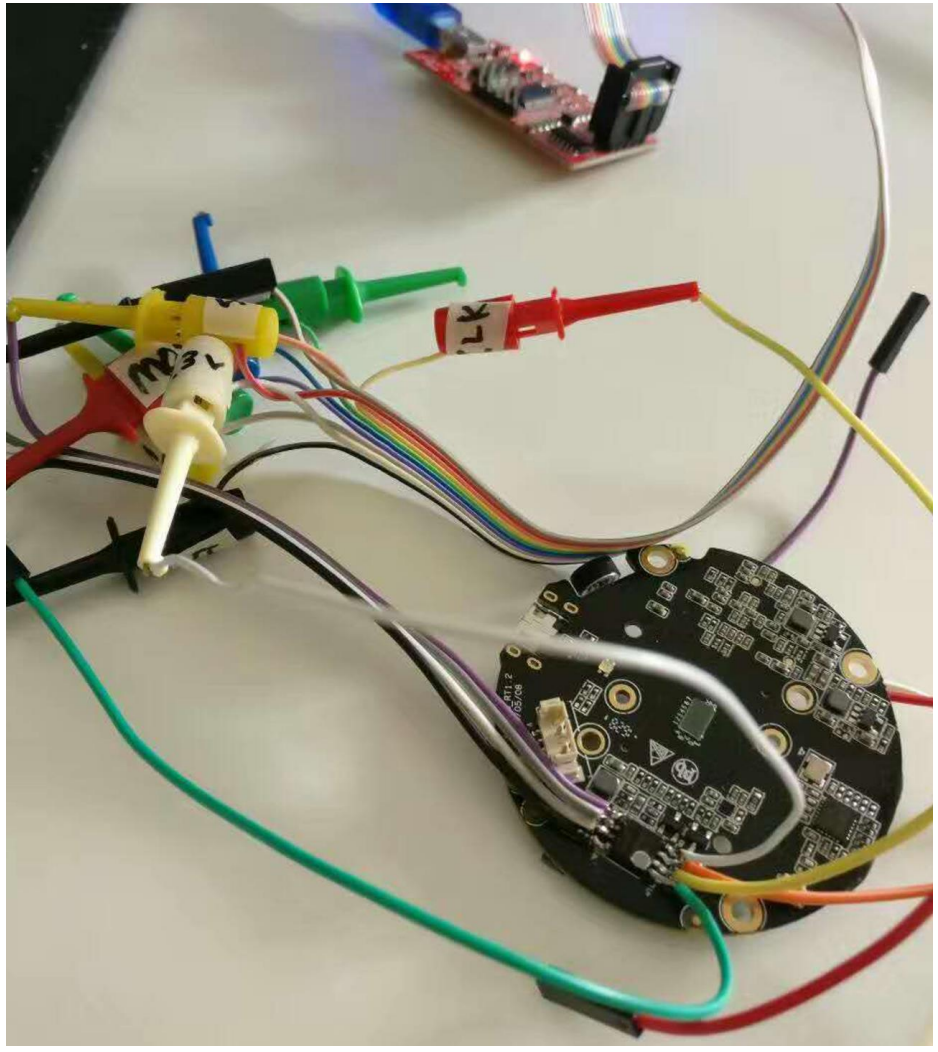
```
ruby -e 'print "\xFF" * 393216' >> rootfs_e.jjfs
```

Merging the ROM

```
(dd if=yicam_night_test_GD25Q128C_bootloader.bin ) > yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_env.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_conf.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_os.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_rootfs_e.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_home.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_vd1.bin ) >> yicam_full_e.bin
(dd if=yicam_night_test_GD25Q128C_ver.bin ) >> yicam_full_e.bin
```

Seal

Flashing Back The Firmware

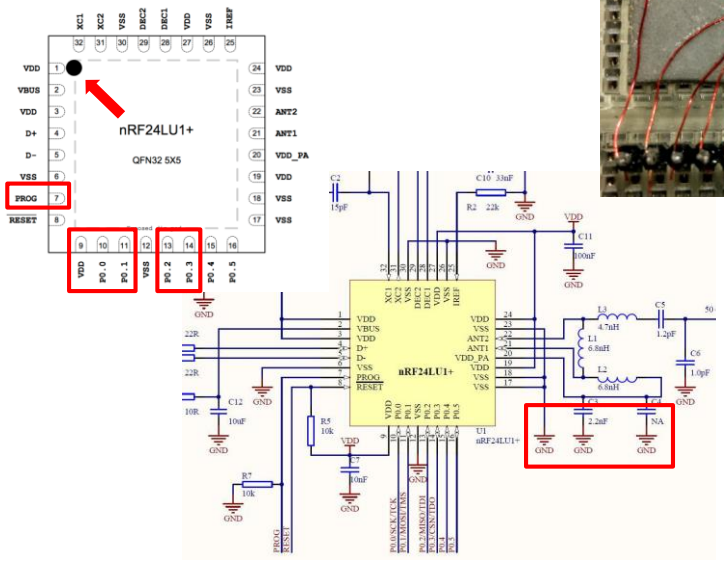


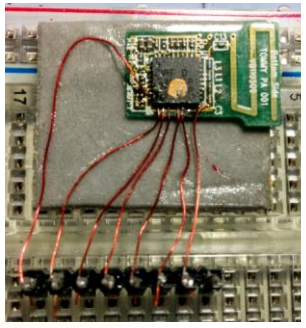
Note and Never Forget

Always Read The Data Sheet

Identifying the Pins

```
#!/usr/bin/perl -w

# Simple perl script to drive the Bus Pirate and unbrick your CrazyRadio dongle.
# Adapted (sorta) from the Bus Pirate example script and mbed NRF24LU1+ flasher projects:
#  http://code.google.com/p/the-bus-pirate/source/browse/trunk/scripts/SPIeeprom.pl
#  http://mbed.org/users/mux/code/nrflash
#
# This script uses the aux output on the Bus Pirate as the PROG pin on the CrazyRadio's NRF24LU1+ chip.
#
# Electrical connections are as follows:
#
#  Bus Pirate          CrazyRadio
#  ==========================
#  MOSI ()       ->  MOSI   (6)
#  MISO ()       ->  MISO   (8)
#  SCK  ()       ->  SCK    (4)
#  CS   ()       ->  CS     (10)
#  AUX  ()       ->  PROG   (2)
#  3V3  ()       ->  3V3    (5)
#  GND  ()       ->  GND    (9)

use strict;
use feature 'say';
use Getopt::Long;
use Device::SerialPort;
use Time::HiRes qw/usleep/;

use constant {
    WREN        => "\x06",
    WRDIS       => "\x04",
    RDSR        => "\x05",
    WRSR        => "\x01",
    READ        => "\x03",
    PROGRAM     => "\x02",
    ERASE_PAGE  => "\x52",
    ERASE_ALL   => "\x62",
    RDFPCR      => "\x89",
    RDISMB      => "\x85",
    ENDEBUG     => "\x86",
    RDYN        => "\x10",
    FLASH_LEN   => 32768,

    BP_CS       => "\x01",
    BP_AUX      => "\x02",
    BP_PULLUP   => "\x04",
    BP_POWER    => "\x08",
};

my %opts;
my $port;
my $time = 500;
my $status_byte;
my $return;
```





- MOSI - Pin 11
- MISO - Pin 13
- SCK - Pin 10
- CS - PIN 14
- AUX – Pin 7
- 3V3 - Pin 1
- GND – Any GND

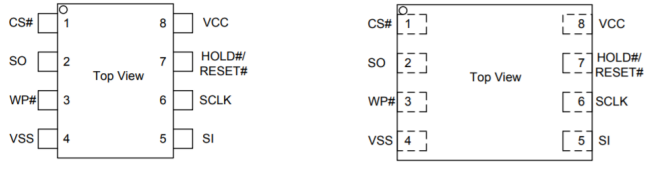
GD25Q128CxIGx 3.3V Uniform Sector Dual and Quad Serial Flash

http://www.elm-tech.com

. GENERAL DESCRIPTION

The GD25Q128C(128M-bit) Serial flash supports the standard Serial Peripheral Interface (SPI), and supports ie Dual/Quad SPI: Serial Clock, Chip Select, Serial Data I/O0 (SI), I/O1 (SO), I/O2 (WP#) and I/O3 (HOLD#/ ESET#). The Dual I/O data is transferred with speed of 208Mbits/s and the Quad I/O & Quad Output data is ansferred with speed of 320Mbits/s.

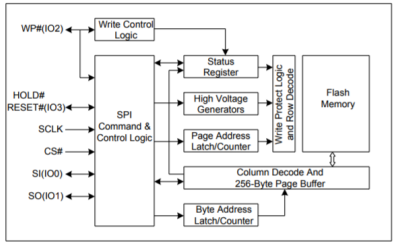
Connection Diagram



in Description

Pin Name	I / O	Description
CS#	I	Chip Select Input
SO (IO1)	I/O	Data Output (Data Input Output 1)
WP# (IO2)	I/O	Write Protect Input (Data Input Output 2)
VSS		Ground
SI (IO0)	I/O	Data Input (Data Input Output 0)
SCLK	I	Serial Clock Input
HOLD#/RESET (IO3)	I/O	Hold or Reset Input (Data Input Output 3)
VCC		Power Supply

lock Diagram



Buying Hardware Is a Must?

Obtain Firmware and Unpacking

```
→ tools binwalk -e test.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
218040        0x353B8         CRC32 polynomial table, little endian
524288        0x80000         uImage header, header size: 64 bytes, header CRC:
0x4687D1AC, created: 2007-06-15 10:36:26, image size: 2217656 bytes, Data Addres
s: 0x2000000, Entry Point: 0x2000040, data CRC: 0xA54D09E1, OS: Linux, CPU: ARM,
 image type: OS Kernel Image, compression type: none, image name: "gm8136"
524352        0x80040         Linux kernel ARM boot executable zImage (little-en
dian)
542452        0x846F4         gzip compressed data, maximum compression, from Un
ix, last modified: 1970-01-01 00:00:00 (null date)
3670112       0x380060        xz compressed data
3800908       0x39FF4C        xz compressed data
3931872       0x3BFEE0        xz compressed data
4979008       0x4BF940        xz compressed data
```

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
217628        0x3521C         CRC32 polynomial table, little endian
524288        0x80000         uImage header, header size: 64 bytes, header CRC: 0x68F55153, created: 2006-09-23 11:52:56, image size: 2
217456 bytes, Data Address: 0x2000000, Entry Point: 0x2000040, data CRC: 0xD41DD892, OS: Linux, CPU: ARM, image type: OS Kernel Image,
compression type: none, image name: "gm8136"
524352        0x80040         Linux kernel ARM boot executable zImage (little-endian)
542452        0x846F4         gzip compressed data, maximum compression, from Unix, last modified: 1970-01-01 00:00:00 (null date)
3670016       0x380000        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 6963644 bytes, 183 inodes, blocksi
ze: 131072 bytes, created: 2006-09-24 03:01:35
11534336      0xB00000        JFFS2 filesystem, little endian
^C
→       dd if=./MX25L12805_20170912_140739.BIN bs=3670016 count=1 of=part1.bin ; \
> dd if=./MX25L12805_20170912_140739.BIN bs=11534336 skip=1 of=part2.bin ; \
> mksquashfs  squashfs-root squashfs-customize.bin -comp xz ; \
>
1+0 records in
1+0 records out
3670016 bytes (3.7 MB, 3.5 MiB) copied, 0.0050487 s, 727 MB/s
0+1 records in
0+1 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.00694756 s, 755 MB/s
```

extract the front and back parts of the file system and repackage the file system

How To Locate Target

```
root@kali:~/iot/firmware-mod-kit/fmk/rootfs/usr/bin# ls -al
total 28
drwxr-xr-x 2 root root 4096 Jan 21  2015 .
drwxr-xr-x 6 root root 4096 Mar 27  2014 ..
lrwxrwxrwx 1 root root   17 Jun 30 05:48 [ -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 [[ -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 arping -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 awk -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 basename -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 bunzip2 -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 bzcat -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 clear -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 cmp -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 crontab -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 cut -> ../../bin/busybox
lrwxrwxrwx 1 root root   16 Jun 30 05:48 dbclient -> ../sbin/dropbear
lrwxrwxrwx 1 root root   17 Jun 30 05:48 dirname -> ../../bin/busybox
lrwxrwxrwx 1 root root   16 Jun 30 05:48 dropbearkey -> ../sbin/dropbear
lrwxrwxrwx 1 root root   17 Jun 30 05:48 du -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 env -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 expr -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 find -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 free -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 head -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 hexdump -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 hostid -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 id -> ../../bin/busybox
-rwxr-xr-x 1 root root 7144 Mar 27  2014 jshn
lrwxrwxrwx 1 root root   17 Jun 30 05:48 killall -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 less -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 logger -> ../../bin/busybox
-rwxr-xr-x 1 root root 9284 Mar 27  2014 lua
lrwxrwxrwx 1 root root   17 Jun 30 05:48 md5sum -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 mkfifo -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 nc -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 nslookup -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 passwd -> ../../bin/busybox
lrwxrwxrwx 1 root root   17 Jun 30 05:48 pgrep -> ../../bin/busybox
```

软件调试

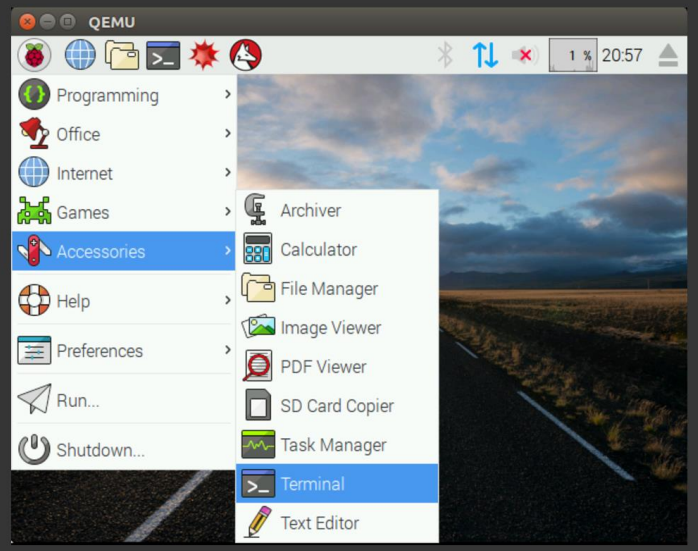
- LINUX
- Web
- Perl
- Python
- ASM
- C

What Do You See

QEMU

```
$ qemu-system-arm -kernel ~/qemu_vms/<your-kernel-qemu> -cpu arm1176 -m 256 -M versatilepb -serial stdio -append
```

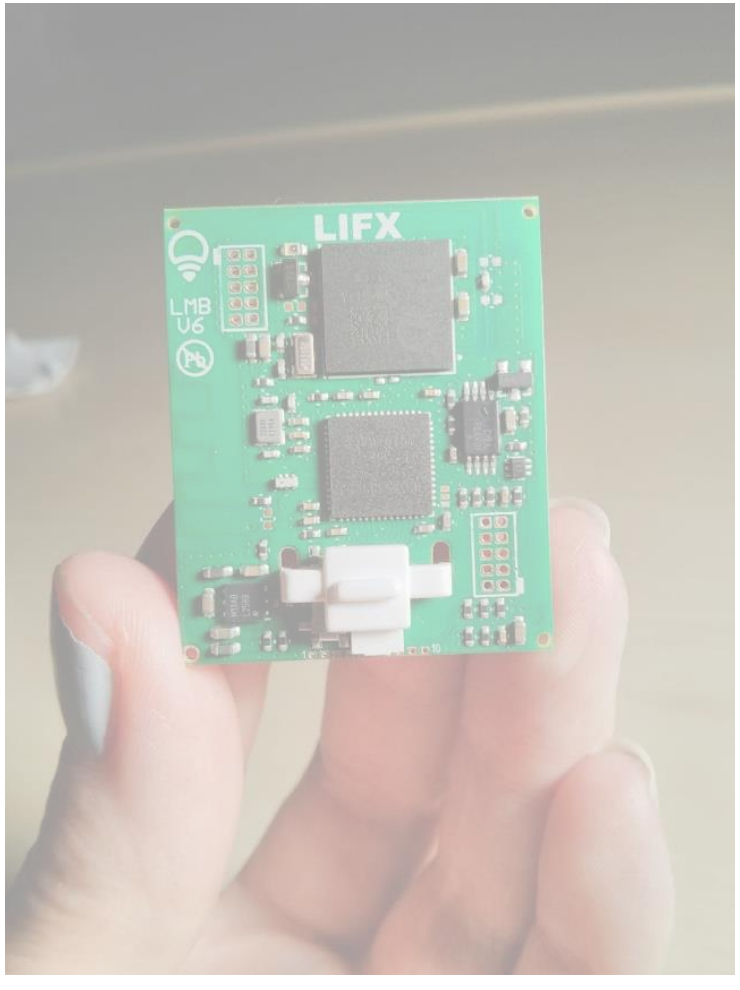
If you see GUI of the Raspbian OS, you need to get into the terminal. Use Win key to get the menu, then navigate with arrow keys until you find Terminal application as shown below.



From the terminal, you need to start the SSH service so that you can access it from your host system (the one from which you launched the qemu).

- Can QEMU Emulate All
- What Could We Possibility Face

Hacking Started

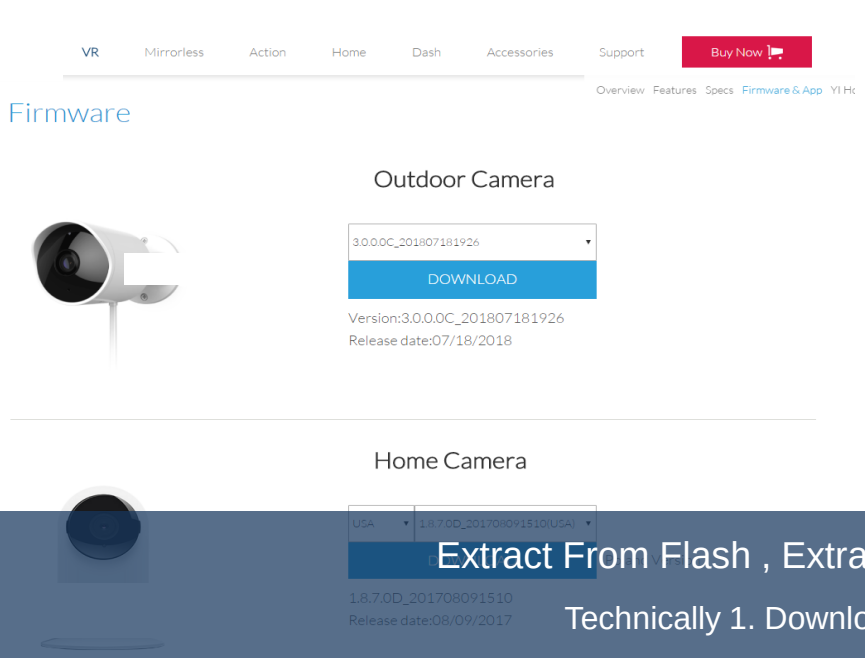


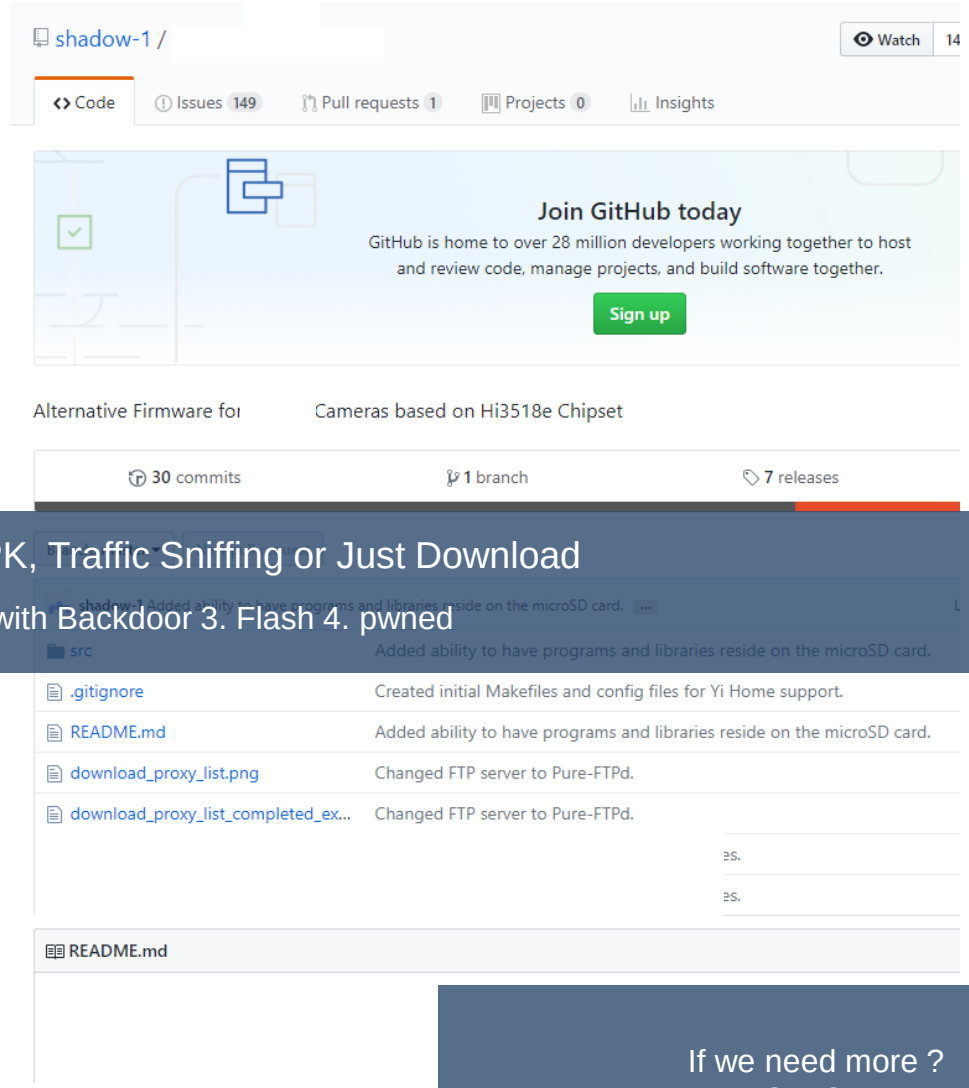

```
MOVS     R2, #0x68          ; Rd = Op2
MOV      R0, R4             ; Rd = Op2
LDR      R1, =0x20003AA0    ; Load from Memory
BL       sub_8035060        ; Branch with Link
ADD      R0, SP, #0x330+var_128 ; Rd = Op1 + Op2
MOVS     R2, #0x80          ; Rd = Op2
LDR      R1, =AES_Key       ;
BL       Ref_sbox_1         ; Branch with Link
ADD      R0, SP, #0x330+var_128 ; Rd = Op1 + Op2
MOVS     R1, #1             ; Rd = Op2
MOVS     R2, #0x70          ; Rd = Op2
LDR      R3, =AES_IV        ; Load from Memory
STMEA.W  SP, {R4,R5}        ; Store Block to Memory
BL       Ref_Ref_sbox       ; Branch with Link
MOVS     R0, #0x70          ; Rd = Op2
ADD.W    SP, SP, #0x324     ; Rd = Op1 + Op2
POP      {R4,R5,PC}         ; Pop registers
; End of function AES
```

Virtualized

Getting Firmware

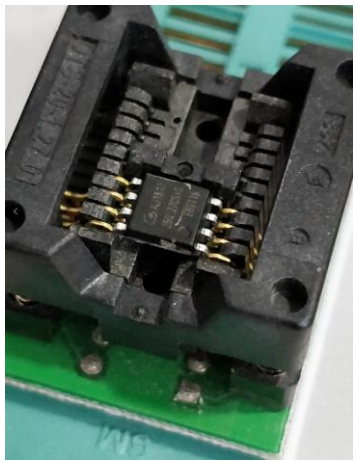
Firmware and Hardware





Extract From Flash , Extract From APK, Traffic Sniffing or Just Download

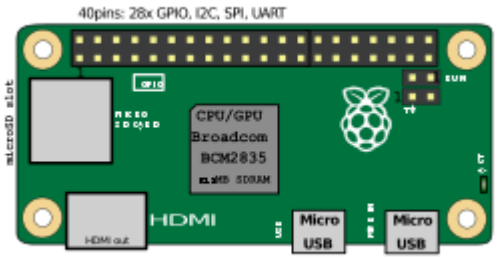
Technically 1. Download 2. Patch with Backdoor 3. Flash 4. pwned

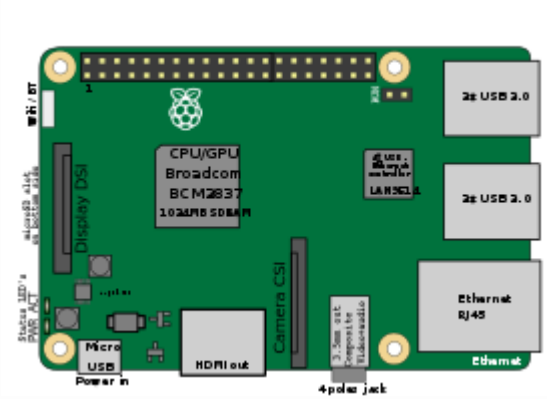


If we need more ?
1. RCE 2. Fuzz

Work Around

Complete Kit to Success





MIPS

ARM

AARCH64

How Many Dev Board

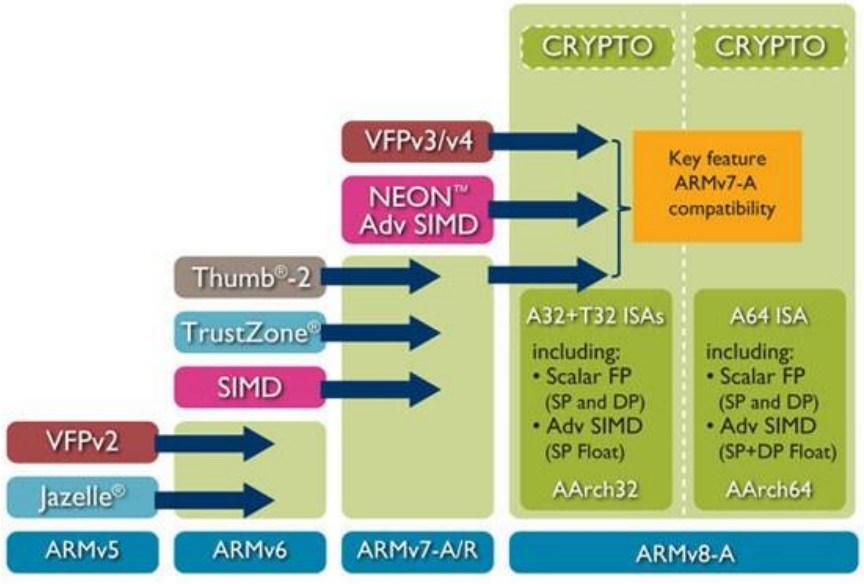
Classic LIBC Issue

```
fstat(3, {st_mode=S_IFREG|0644, st_size=35112, ...}) = 0
mmap(NULL, 99840, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xffff8b54d000
mprotect(0xffff8b554000, 65536, PROT_NONE) = 0
mmap(0xffff8b564000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0xffff8b564000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b54e000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b543000
mprotect(0xffff8bee3000, 16384, PROT_READ) = 0
mprotect(0xffff8b564000, 4096, PROT_READ) = 0
mprotect(0xffff8b585000, 4096, PROT_READ) = 0
mprotect(0xffff8b708000, 16384, PROT_READ) = 0
mprotect(0xffff8b738000, 4096, PROT_READ) = 0
mprotect(0xffff8c2fb000, 4096, PROT_READ) = 0
mprotect(0xffff8bbf9000, 4096, PROT_READ) = 0
mprotect(0xffff8b839000, 45056, PROT_READ) = 0
mprotect(0xffff8bcea000, 4096, PROT_READ) = 0
mprotect(0xffff8b8c0000, 4096, PROT_READ) = 0
mprotect(0xffff8b91d000, 4096, PROT_READ) = 0
mprotect(0xffff8b9c7000, 4096, PROT_READ) = 0
mprotect(0xffff8b985000, 4096, PROT_READ) = 0
mprotect(0xffff8ba0b000, 4096, PROT_READ) = 0
mprotect(0xffff8b068000, 53248, PROT_READ) = 0
mprotect(0xffff8bb8c000, 4096, PROT_READ) = 0
mprotect(0xffff8bbaf000, 4096, PROT_READ) = 0
mprotect(0xffff8bf0f000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8b547000
mprotect(0xffff8c08e000, 40960, PROT_READ) = 0
mprotect(0xffff8bd51000, 155648, PROT_READ|PROT_WRITE) = 0
mprotect(0xffff8bd51000, 155648, PROT_READ|PROT_EXEC) = 0
mprotect(0xffff8c1af000, 32768, PROT_READ) = 0
mprotect(0xffff8c59f000, 4096, PROT_READ) = 0
munmap(0xffff8c596000, 19536)           = 0
set_tid_address(0xffff8b549580)         = 3637
set_robust_list(0xffff8b549510, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0xffff8c2da768, sa_mask=[], sa_flags=SA_SIGINFO}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0xffff8c2da838, sa_mask=[], sa_flags=SA_RESTART|SA_SIGINFO}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
--- SIGILL {si_signo=SIGILL, si_code=ILL_ILLOPC, si_addr=0xffff8c574338} ---
+++ killed by SIGILL +++
Illegal instruction
```

Hardware is not “down gradable”

Assembly Instruction Compatibility

```
gef> gef config context.layout "code stack"
gef> break *0x0001043c
Breakpoint 1 at 0x1043c
gef> run
Starting program: /home/azeria/exp/stack
AAAAAAA                                                    user's input
-------------------------------------------------------------------------------[ code:arm ]----
      0x10424 <main+8>         sub    sp,  sp,  #16
      0x10428 <main+12>        str    r0,  [r11,  #-16]
      0x1042c <main+16>        str    r1,  [r11,  #-20] ; 0xffffffec
      0x10430 <main+20>        sub    r3,  r11,  #12
      0x10434 <main+24>        mov    r0,  r3
      0x10438 <main+28>        bl     0x102c4 <gets@plt>
->    0x1043c <main+32>        mov    r0,  r3
      0x10440 <main+36>        sub    sp,  r11,  #4
      0x10444 <main+40>        pop    {r11,  pc}
      0x10448 <__libc_csu_init+0> push   {r3,  r4,  r5,  r6,  r7,  r8,  r9,  lr}
      0x1044c <__libc_csu_init+4> mov    r7,  r0
      0x10450 <__libc_csu_init+8> ldr    r6,  [pc,  #76]        ; 0x104a4 <__libc_csu_init+92>
-------------------------------------------------------------------------------[ stack ]----
0xbefff238|+0x00: 0xbefff3a4 -> 0xbefff503 -> "/home/azeria/exp/stack"  <-$sp
0xbefff23c|+0x04: 0x00000001
0xbefff240|+0x08: "AAAAAAA"     <-$r0                                          Stack Frame
0xbefff244|+0x0c: 0x00414141 ("AAA"?)          "buffer"
0xbefff248|+0x10: 0x00000000                   prev. R11/FP
0xbefff24c|+0x14: 0xb6e8c294 -> <__libc_start_main+276> bl 0xb6ea4b28 <__GI_exit>   prev. LR
0xbefff250|+0x18: 0xb6fb1000 -> 0x0013cf20
0xbefff254|+0x1c: 0xbefff3a4 -> 0xbefff503 -> "/home/azeria/exp/stack"
-------------------------------------------------------------------------------------------
```



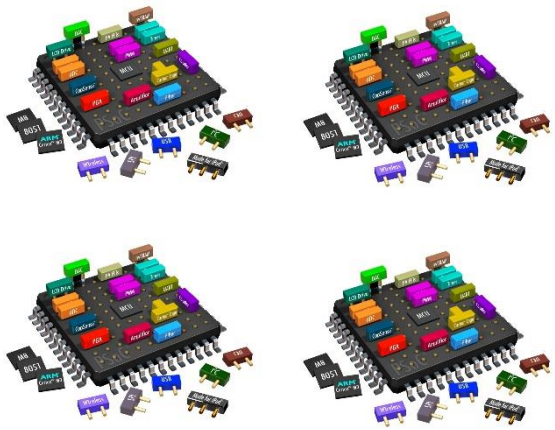
ARM

AARCH64

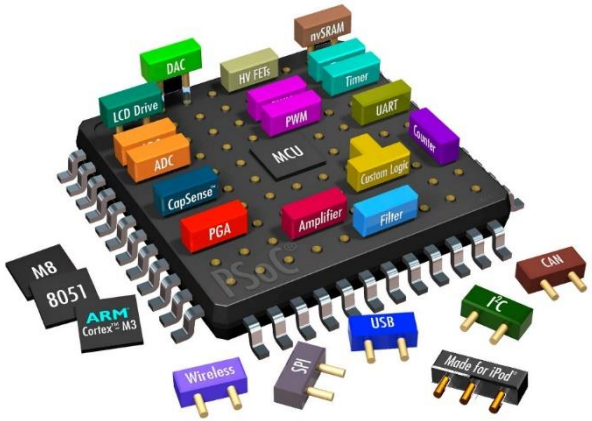
Why Firmware Emulation

More Resources = More Power

Multicore



MAX RAM



MAX Space

Processor

Normally 1-2 Core

RAM

Normally
256MB/512MB

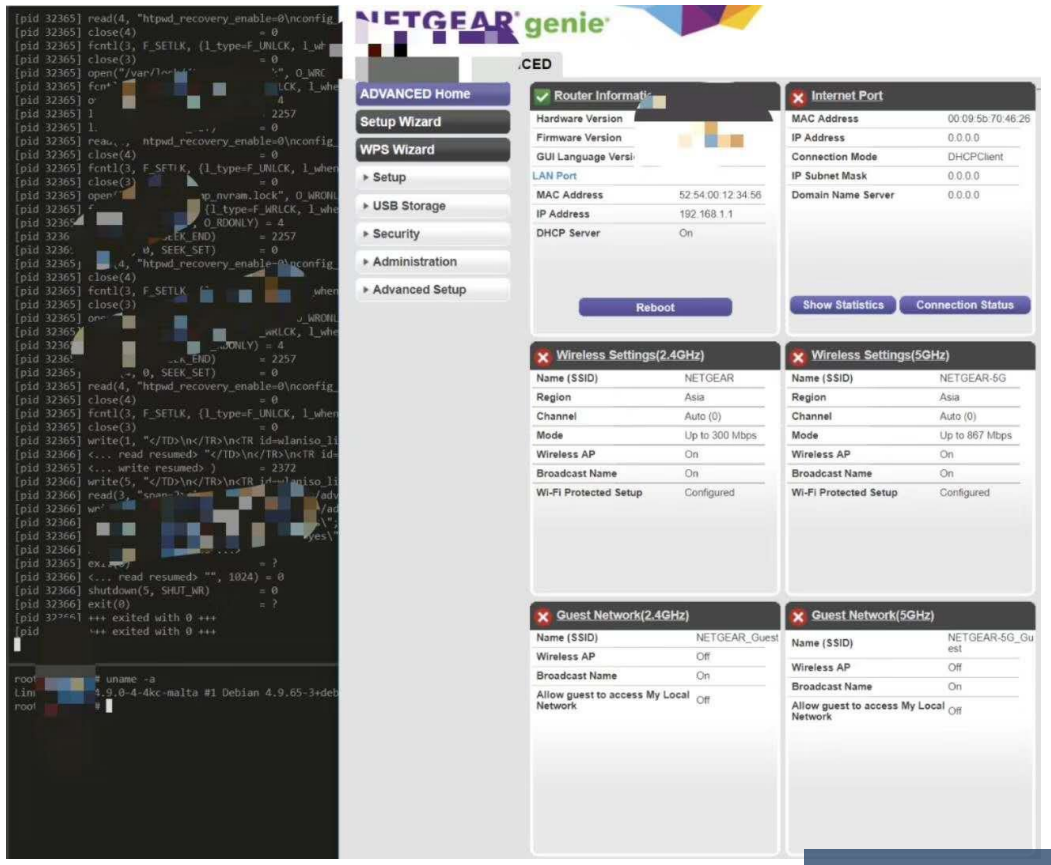
FLASH

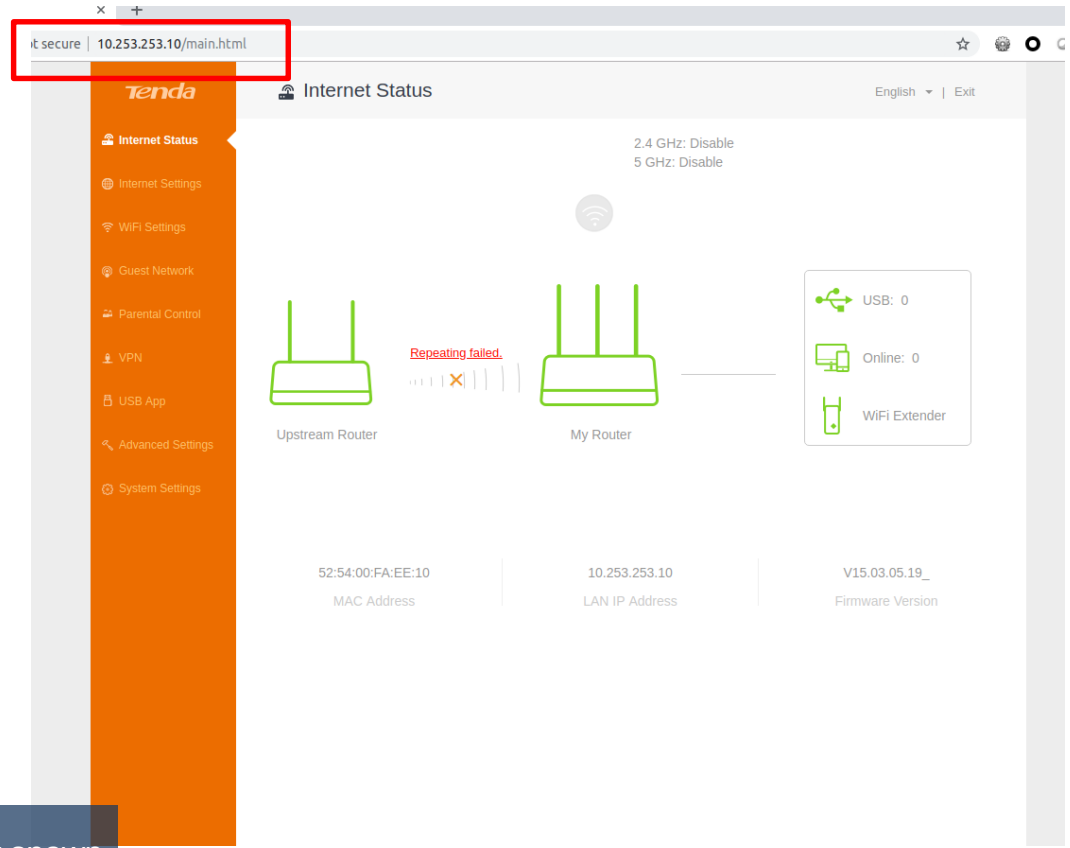
Normally
8MB/16MB/32MB/256MB

Most Important, we got apt-get

Objectives

Only One Process with Interaction





Hunt for the one that spawn listener port

most of the devices comes with one big binary

Boot

Distro and Kernel Mix and Match

script to boot arm

```
#!/bin/bash

sudo tunctl -d tap0

sudo screen -dm /opt/qemu/bin/qemu-system-arm -m 2048 -M virt -cpu cortex-a15 -smp cpus=
4,maxcpus=4 -kernel boot.stretch.armhf.virt/vmlinuz-4.9.0-6-armmp-lpae -initrd boot.stre
tch.armhf.virt/initrd.img-4.9.0-6-armmp-lpae -append "root=/dev/vda2" -drive file=debian
-stretch.armhf_virt.qcow2,if=none,format=qcow2,id=hd0 -device virtio-blk-device,drive=hd
0 -netdev type=tap,id=net0 -device virtio-net-device,netdev=net0,mac=52:54:00:fa:ee:10 -
nographic

sudo sysctl -w net.ipv4.ip_forward=1

echo "Stopping firewall and allowing everyone..."
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT

sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
sudo iptables -I FORWARD 1 -i tap0 -j ACCEPT
sudo iptables -I FORWARD 1 -o tap0 -m state --state RELATED,ESTABLISHED -j ACCEPT


sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1022 -j DNAT --to-destination
 10.253.253.10:22
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1080 -j DNAT --to-destination
 10.253.253.10:80
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 10443 -j DNAT --to-destinatio
n 10.253.253.10:443

echo "Booting VM, eta 10 seconds"
sleep 10
sudo ifconfig tap0 10.253.253.254 netmask 255.255.255.0
```

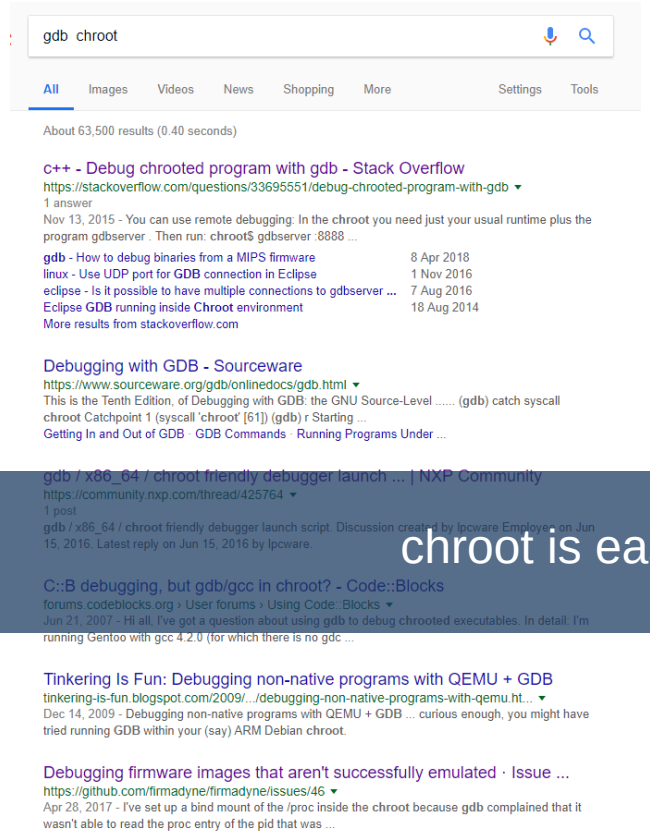
script to boot mips

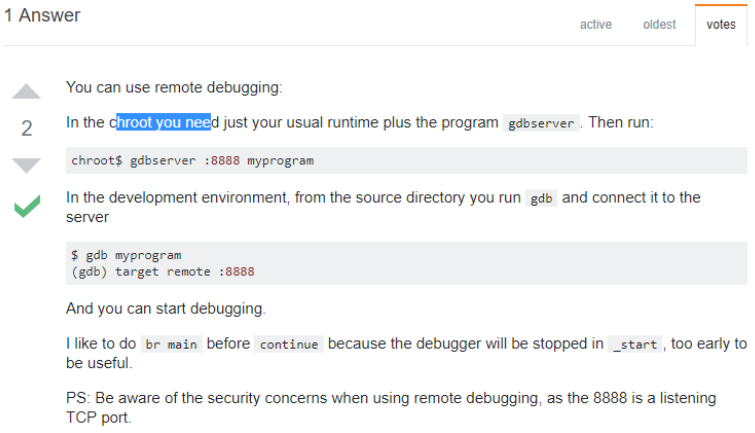
```
#!/bin/bash

sudo screen -dm /opt/qemu/bin/qemu-system-mipsel -m 512 -M malta -kernel boot.stretch.mi
psel/vmlinux-4.9.0-4-4kc-malta -initrd boot.stretch.mipsel/initrd.img-4.9.0-4-4kc-malta
-append "root=/dev/sda1  net.ifnames=0 biosdevname=0 nokaslr" -hda debian-stretch.mipsel
.qcow2 -net nic -net tap,ifname=tap0,script=no,downscript=no -net nic -net tap,ifname=ta
p1,script=no,downscript=no -nographic

sudo tunctl -t tap0 -u xwings
sudo ifconfig tap0 10.253.253.254 netmask 255.255.255.0

sudo sysctl -w net.ipv4.ip_forward=1

echo "Stopping firewall and allowing everyone..."
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT

sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
sudo iptables -I FORWARD 1 -i tap0 -j ACCEPT
sudo iptables -I FORWARD 1 -o tap0 -m state --state RELATED,ESTABLISHED -j ACCEPT


sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1122 -j DNAT --to-destination
 10.253.253.11:22
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 1180 -j DNAT --to-destination
 10.253.253.11:80
sudo iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 11443 -j DNAT --to-destinatio
n 10.253.253.11:443
```

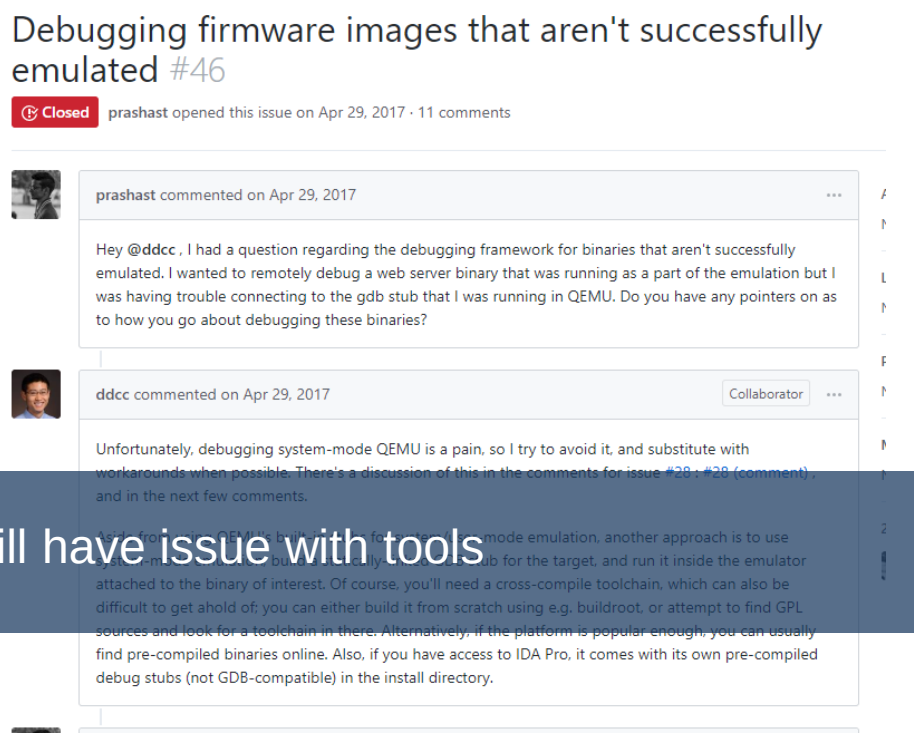
argument: running new or old distro + kernel

chroot

Easy Way Out, chroot







chroot is easy (still hardware dependent), but we will have issue with tools

Running without chroot

Stage 0 Issue: File Not Found

The File Missing Trick

We Missed You

```
chdir("/")                                      = 0
execve("/bin/bash", ["/bin/bash", "-i"], 0xffffca14f650 /* 18 vars */) = -1 ENOENT (No such file or d
irectory)
openat(AT_FDCWD, "/usr/lib/aarch64-linux-gnu/charset.alias", O_RDONLY|O_NOFOLLOW) = -1 ENOENT (No suc
h file or directory)
write(2, "chroot: ", 8chroot: )                 = 8
write(2, "failed to run command '/bin/bash'", 33failed to run command '/bin/bash') = 33
write(2, ": No such file or directory", 27: No such file or directory) = 27
write(2, "\n", 1
)                       = 1
close(1)                                        = 0
close(2)                                        = 0
exit_group(127)                                 = ?
```

We found you

```
root@rpi3:/opt/                 /lib64# file ../bin/bash
../bin/bash: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interprete
r /lib64/ld-linux-aarch64.so.1, for GNU/Linux 3.14.0, BuildID[sha1]=22e2854c58b1814825b95cba103ac658d
371f5b0, stripped
```

Stage 1 Issue: .SO Not Found

Out from chroot, we need feeeding

```
[pid  2680] close(4)                    = 0
[pid  2680] write(1, "<dhcpc script>no udhcpc pid can be killed, but udhcpc id is ", 60) = 60
[pid  2680] newfstatat(AT_FDCWD, "/usr/local/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file
r directory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/local/bin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file o
 directory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or dir
ctory)
[pid  2680] newfstatat(AT_FDCWD, "/usr/bin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or dire
tory)
[pid  2680] newfstatat(AT_FDCWD, "/sbin/ps", 0xfffffe081a30, 0) = -1 ENOENT (No such file or directo
y)
[pid  2680] newfstatat(AT_FDCWD, "/bin/ps", {st_mode=S_IFREG|0755, st_size=535832, ...}, 0) = 0
[pid  2680] pipe2([4, 7], 0)            = 0
[pid  2680] clone(strace: Process 2681 attached
```

```
Usage: unzip [-lnopq] FILE[.zip] [FILE]... [-x FILE...] [-d DIR]
root@aarch64:/opt/                i2/bin# ln -s busybox.nosuid unzip
root@aarch64:/opt/                i2/bin# ./busybox.nosuid sync
root@aarch64:/opt/                i2/bin# ./busybox.nosuid syn
syn: applet not found
root@aarch64:/opt/                i2/bin# ln -s busybox.nosuid sync
root@aarch64:/opt/                i2/bin#
```

```
root@                2/usr/lib64# ln -s libgnutls.so.30.9.0 libgnutls.so.30
root@                2/usr/lib64# ln -s libidn.so.11.6.16 libidn.so.11
root@                2/usr/lib64# ln -s libnettle.so.6.2 libnettle.so.6
root@                2/usr/lib64# ln -s libhogweed.so.4.2 libhogweed.so.4
root@                2/usr/lib64# ln -s libgmp.so.10.3.1 libgmp.so.10
root@                2/usr/lib64# ln -s libpcre.so.1.2.7 libpcre.so.1
root@                2/usr/lib64# ln -s libexpat.so.1.6.2 libexpat.so.1
root@                2/usr/lib64#
```

Feeding all the required so and binary with “ln –s”

Out from chroot, we need feeding

```
bash-3.2# /usr/bin/appmainprog
<appmain>*******************************************
<appmain>child process id is 3931
<appmain>Appcliation Init Begin
<appmain>Audio Mas process Init
[Aud][PPC] AudioPPCControl constructor
[Aud][PPC] AudioPPCControl getInstance
[Aud][PPC] AudioPPCControl freeInstance
[Aud][PPC] AudioPPCControl destructor
[Aud][PPC][deInit] PPC deinit begin.
[Aud][PPC][ppcStructUnalloc] ppc_destroy_info begin.
Segmentation fault
bash-3.2#
```

```
close(3)                                = 0
write(1, "<appmain>Appcliation Init Begin\n", 32<appmain>Appcliation Init Begin
) = 32
write(1, "<appmain>Audio Mas process Init\n", 32<appmain>Audio Mas process Init
) = 32
umask(000)                              = 022
faccessat(AT_FDCWD, "/data/log_all", F_OK) = -1 ENOENT (No such file or directory)
socket(AF_UNIX, SOCK_DGRAM|SOCK_CLOEXEC, 0) = 3
connect(3, {sa_family=AF_UNIX, sun_path="/dev/log"}, 110) = -1 ENOENT (No such file or directory)
close(3)                                = 0
write(1, "[Aud][PPC] AudioPPCControl constructor\n", 39[Aud][PPC] AudioPPCControl constructor
) = 39
write(1, "[Aud][PPC] AudioPPCControl getInstance\n", 39[Aud][PPC] AudioPPCControl getInstance
) = 39
faccessat(AT_FDCWD, "/tmp/ppcfifo", F_OK) = -1 ENOENT (No such file or directory)
FDCWD, "/tmp/ppcfifo", S_IFIFO|0777) = -1 ENOENT (No such file or directory)
```

Classical file not found error

“segfault” without clear error. strace come to rescue

NVram

Dark side of NVRAM

```
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (N
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)          = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

main process

ask for nvram info

Relationship between main binary is so intimate, but in actual fact. Is just a hit and run

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted]/ /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

Dark Side of NVRAM

main process

```
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (No
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)    = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

ask for nvram info

Relationship between main binary is so intimate, but in actual fact. Is just a hit and run

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

Dark Side of the main process, we ignore and con't to next step

```
efused)
[pid  3088] close(5)                    = 0
[pid  3088] write(1, "[08-28 20:45:32][utils/SNManager.cpp:26][D] : Read NVRAM Failed\n", 64[08-28 20
:45:32][utils/SNManager.cpp:26][D] : Read NVRAM Failed
) = 64
[pid  3088] write(1, "<AST>[RegisterCmdHandler:113]:Cmd [22] Registered Handler!\n", 59<AST>[Register
```

A fake NVRAM

```
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (N
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)          = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

main process

ask for nvram info

IF interactor is the medium,

can we fake it ?

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted]/ /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

A fake NVRAM

main process

```
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/BT_Addr", O_RDONLY) = 5
2750] flock(5, LOCK_SH)                 = 0
2750] read(5, "\0\0F\201g\1`\0#\20\0\0\7\200\0\6\5\7\3@\37@\37\0\4\200\0\377\
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 64) = 64
2750] close(5)                          = 0
2750] openat(AT_FDCWD, "/dev/disk/by-partlabel/NVRAM", O_RDWR) = -1 ENOENT (N
)
2750] openat(AT_FDCWD, "/dev/mtd1", O_RDWR) = -1 ENOENT (No such file or dire
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDEB/PRODUCT_INFO", O_RDONLY) = 5
2750] close(5)                          = 0
2750] newfstatat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", {st_mode=S_IF
..}, 0) = 0
2750] openat(AT_FDCWD, "/data/nvram/APCFG/APRDCL/FILE_VER", O_RDONLY) = 5
2750] read(5, "NVRAM_VER_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 70) = 70
2750] lseek(5, 3626, SEEK_SET)     = 3626
2750] read(5, "PRODUCT_INFO\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

ask for nvram info

IF interactor is the medium,

can we fake it ?

reply with nvram info

```
root@rpi3:/opt/[redacted]# strace -f -s 256 chroot /opt/[redacted] /usr/bin/appmainprog
/abc 2>&1
^Croot@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# ^C
root@rpi3:/opt/[redacted]# cat /tmp/abc | grep nvram
openat(AT_FDCWD, "/lib64/libnvram.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libnvram_custom.so", O_RDONLY|O_CLOEXEC) = 3
root@rpi3:/opt/dingdongmini2#
```

interactor

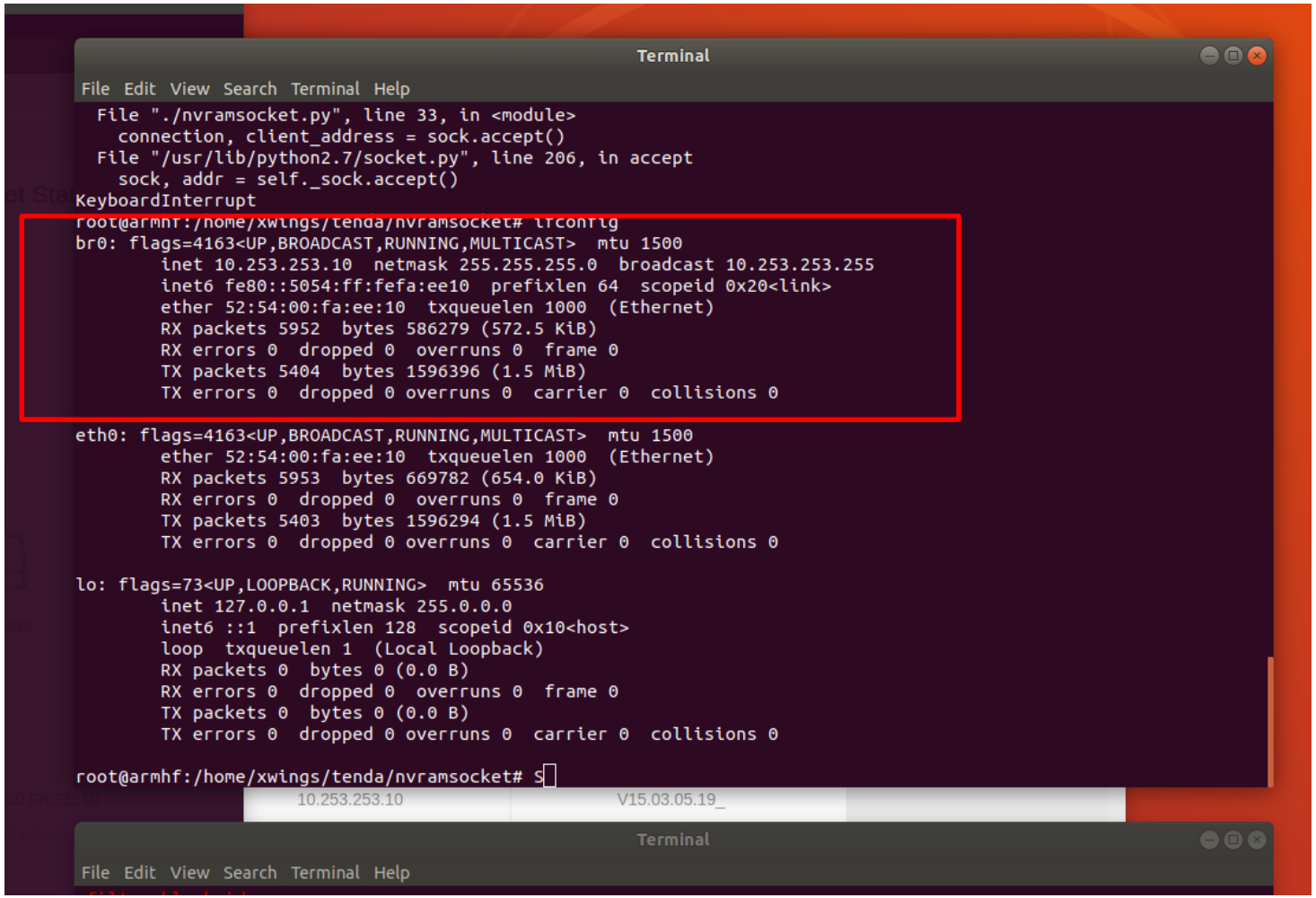
Custom Interactor

nvramsocket.py 2.4 KB

```
#!/usr/bin/python

# For l            ulation
# This code suppose to replace cfmd
# cfmd suppose to be the bridge between nvram and httpd and othe
# so far only httpd works will find out more`

import socket
import sys
import os

server_address = '/opt/            .socket'
data = ''

# Make sure the socket does not already exist
try:
        os.unlink(server_address)
except OSError:
        if os.path.exists(server_address):
                raise
# Create a UDS socket
sock = socket.socket(socket.AF_UNIX,socket.SOCK_STREAM)
# Bind the socket to the port
print >>sys.stderr, 'starting up on %s' % server_address
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

while True:
    # Wait for a connection
    #print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept()
    try:
        #print >>sys.stderr, 'connection from', client_address
        while True:
            data += connection.recv(1024)
            data = str(data)
            #data = data.decode('utf-8')
```

br0

The bridge trick



The switch looking device

Wireless Devices

Faking wpa_supplicant

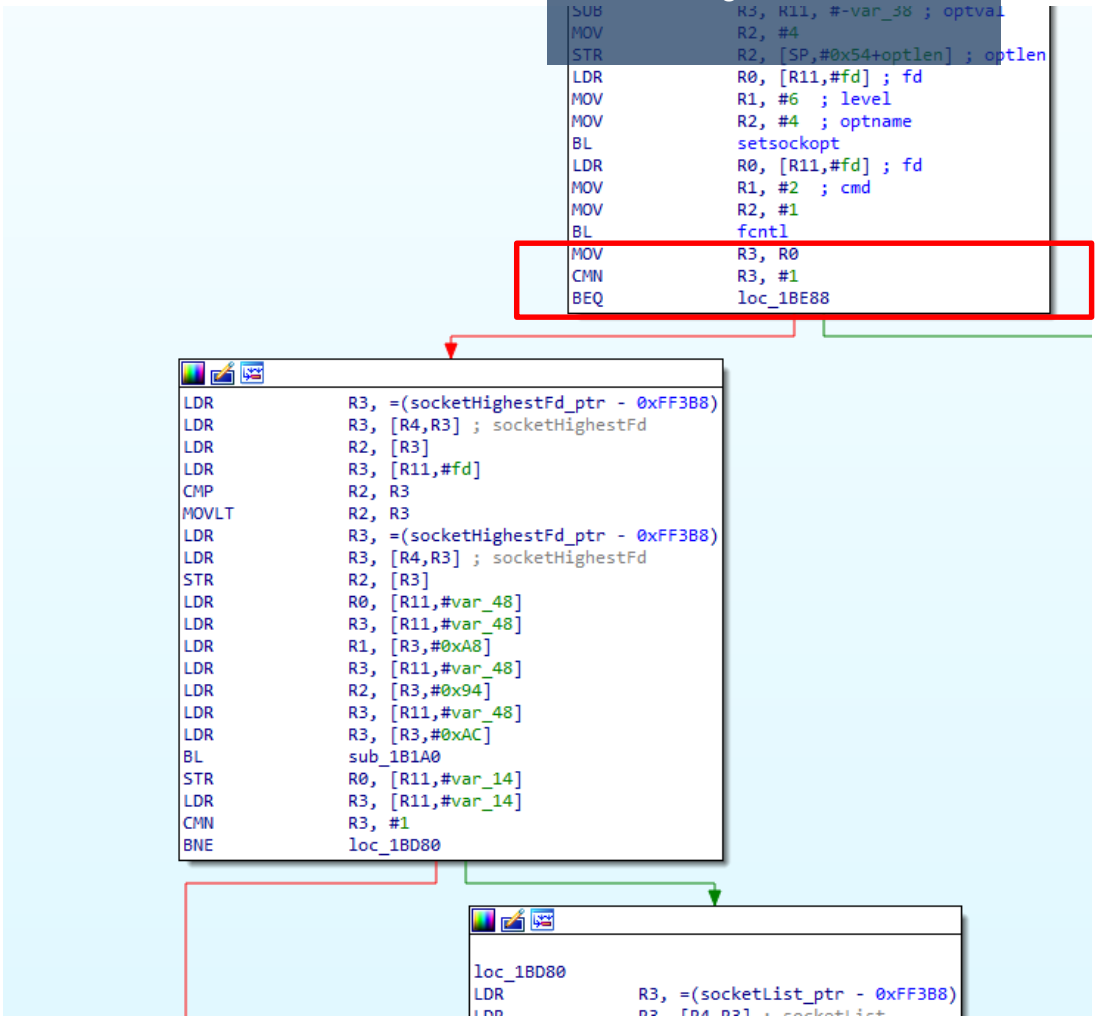
```
[WIFI_MW] Current PID=808

[WIFI_MW]
 control interface dir: /tmp/wpa_supplicant/
 wpa control client path: /tmp/wpa_supplicant/wpa_ctrl_808
 wpa monitor client path: /tmp/wpa_supplicant/wpa_moni_808
 p2p control client path: /tmp/wpa_supplicant/p2p_ctrl_808
 p2p monitor client path: /tmp/wpa_supplicant/p2p_moni_808

[WIFI_MW] [WPA_CTRL] Enter wpaCtrlOpen: ctrl_path = /tmp/wpa_supplicant/wlan0.
[WIFI_MW] wpaCtrlOpen: unlink(), ctrl->s: 11, ctrl->mLocal.sun_path: /tmp/wpa_supplicant/wpa_ct
[WIFI_MW] wpaCtrlOpen: bind(), bindRet = 0.
[WIFI_MW] wpaCtrlOpen: connect(), ctrl->s: 11, ctrl->dest.sun_path: /tmp/wpa_supplicant/wlan0
[WIFI_MW] [WPA_CTRL] Leave wpaCtrlOpen(), conn = 0.
[WIFI_MW] [WPA_CTRL] Enter wpaCtrlOpen: ctrl_path = /tmp/wpa_supplicant/wlan0.
[WIFI_MW] wpaCtrlOpen: unlink(), ctrl->s: 12, ctrl->mLocal.sun_path: /tmp/wpa_supplicant/wpa_mo
[WIFI_MW] wpaCtrlOpen: bind(), bindRet = 0.
```

making eth0 looks like wlan0 works too

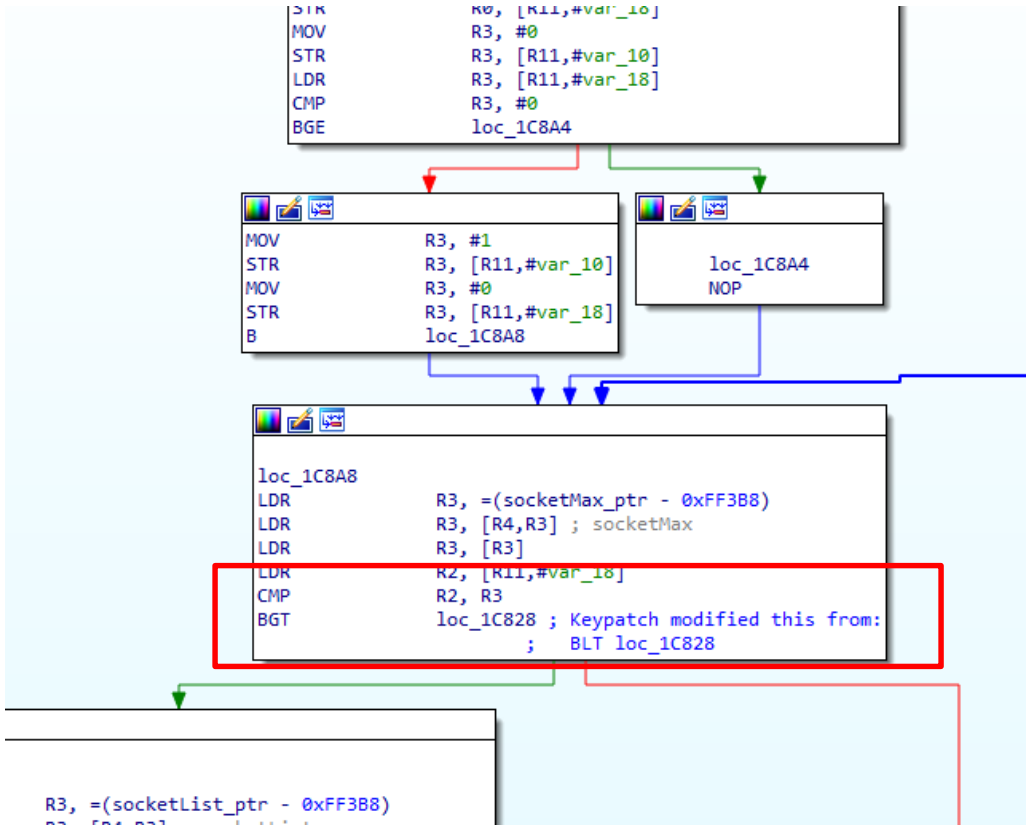
Every Thing Else Fail

BL, BNE, BEQ and friends

Original BIN

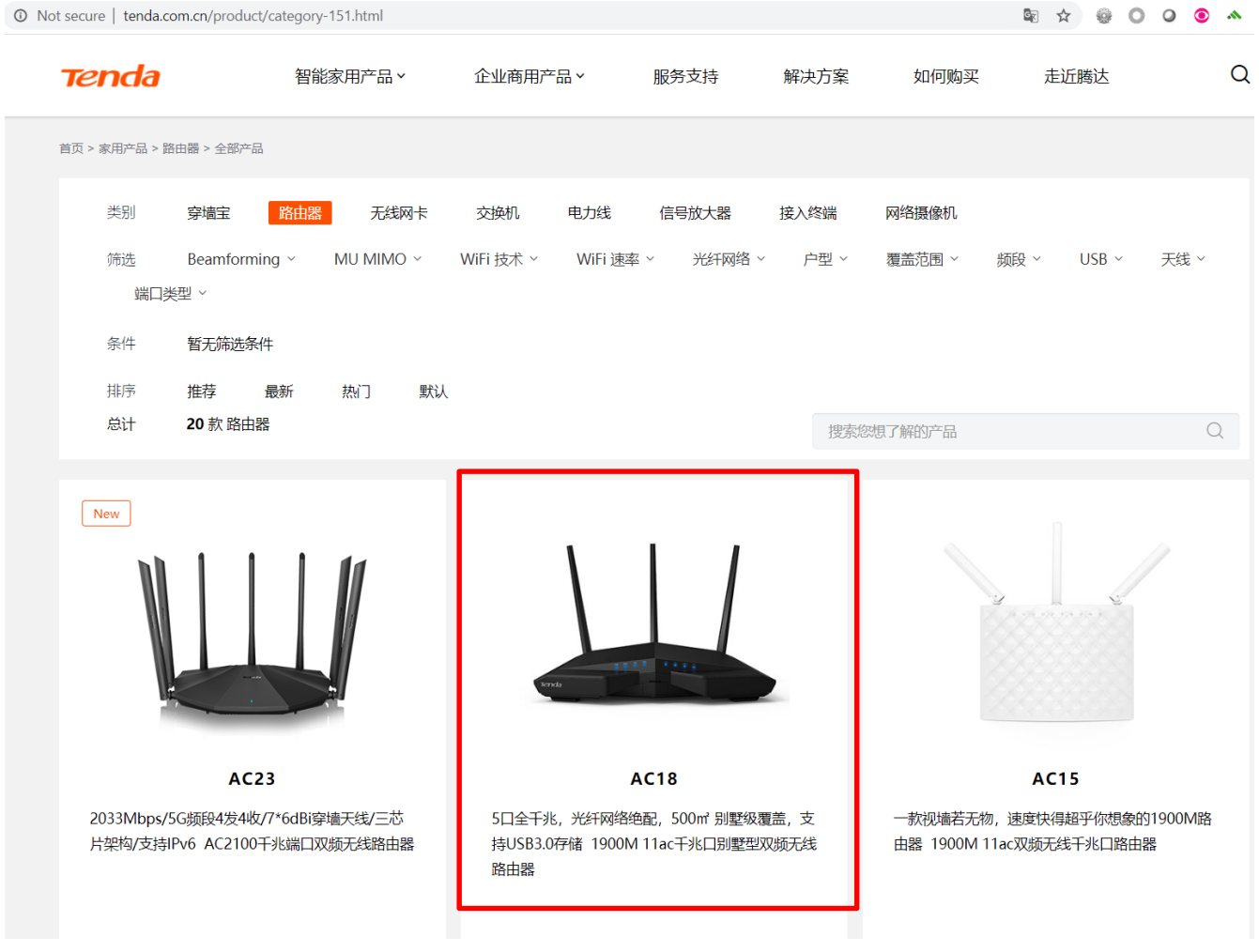


Patched BIN



QEMU: Hands On Time

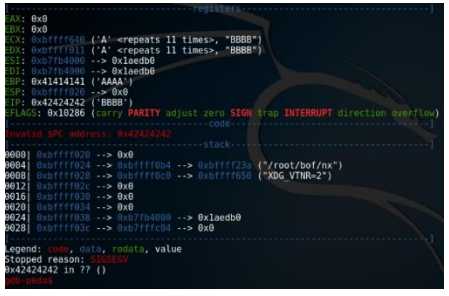
Emulate a Router

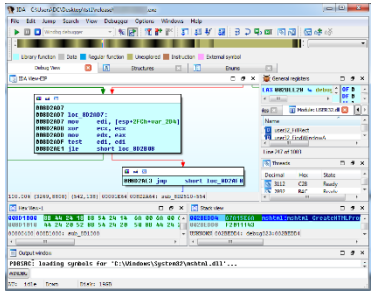


https://www.tenda.com.cn/product/category-151.html

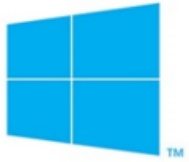
More Flexible Emulation

What is Required

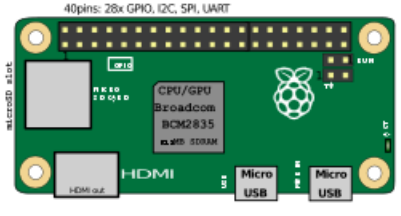


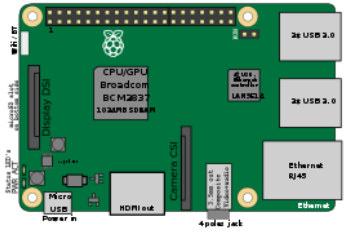


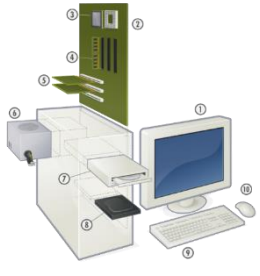
Debugger or Disassembler



*BSD Linux MacOS Windows







MIPS ARM AARCH64 X86

Full Scale Emulator

1960s - IBM released hypervisors on CP-40 and CP-67 operating systems [2]

1999 - VMware introduced virtualization to the x86 platform with VMware Workstation 1.0 [3][4]

2001 - IBM launched a hypervisor for midrange UNIX systems [2]

2001 - VMware released ESX Server 1.0 [5]

2003 - Public release of Xen [6]

2004 - Microsoft releases Virtual Server 2005 [7]

Many more vendors and enhanced virtualization solutions are released.

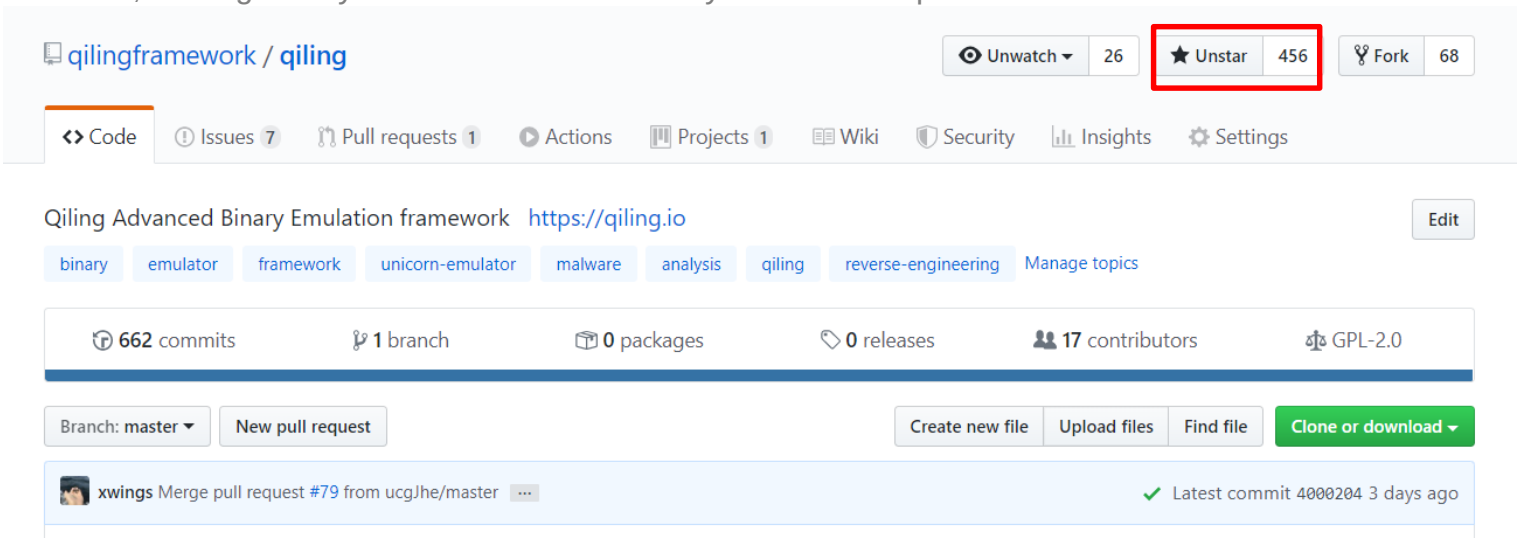
Over Emulate

More Emulate = Higher Chances Being Detected

What Is Qiling Framework

Features and Functionality

- Cross platform: Windows, MacOS, Linux, BSD
- Cross architecture: X86, X86_64, Arm, Arm64, Mips
- Multiple file formats: PE, MachO, ELF
- Emulate & sandbox machine code in a isolated environment
- Provide high level API to setup & configure the sandbox
- Fine-grain instrumentation: allow hooks at various levels (instruction/basic-block/memory-access/exception/syscall/IO/etc)
- Allow dynamic hotpatch on-the-fly running code, including the loaded library
- True Python framework, making it easy to build customized analysis tools on top



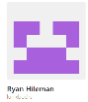
Please Follow and STAR Our Project

** https://github.com/qilingframework/qiling **

User Mode Emulation

qemu-usermode

- The TOOL
- Limited OS Support, Very Limited
- No Multi OS Support
- No Instrumentation
- **Syscall Forwarding**



usercorn

- Very good project !
- It's a Framework !
- Mostly *nix based only
- Limited OS Support (No Windows)
- Go and Lua is not hacker's friendly
- **Syscall Forwarding**

Binee

- Very good project too
- Only X86 (32 and 64)
- Limited OS Support (No *NIX)
- Just a tool, we don't need a tool
- Again, is GO

WINE

- Limited ARCH Support
- Limited OS Support, only Windows
- Not Sandbox Designed
- No Instrumentation

WSL/2

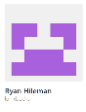
- Limited ARCH Support
- Only Linux and run in Windows
- Not Sandboxed, It linked to /mnt/c
- No Instrumentation (maybe)

Syscall Forwarding

User Mode Emulation

qemu-usermode

- Over Emulate
- The TOOL
- Limited OS Support, Very Limited
- No Multi OS Support
- No Instrumentation
- **Syscall Forwarding**



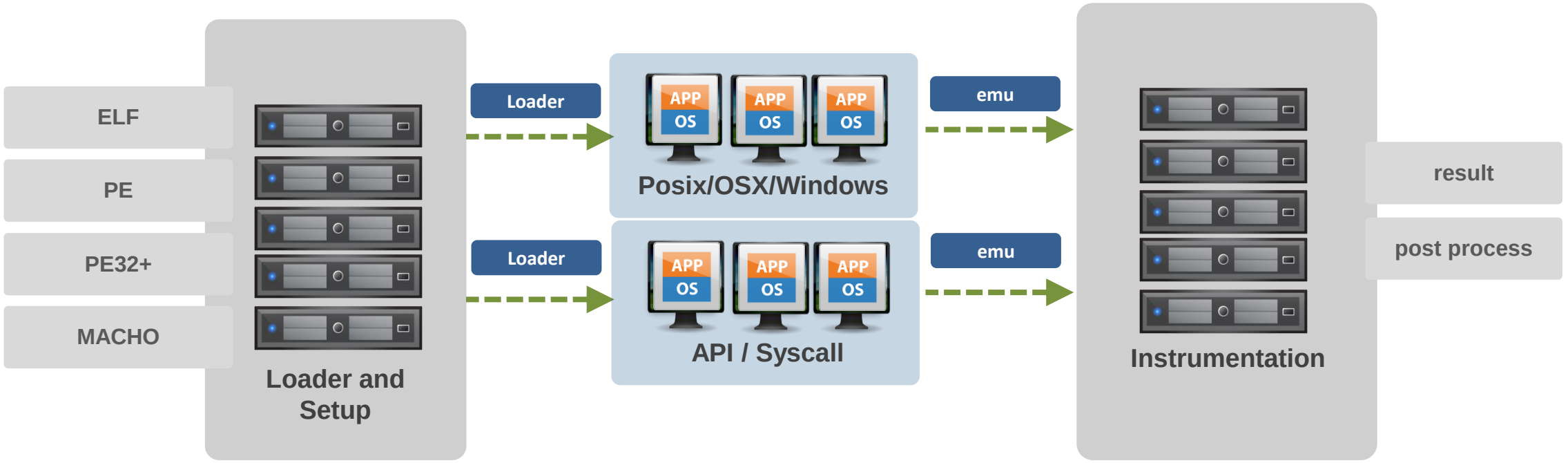
usercorn

- Very good project !
- It's a Framework !
- Mostly *nix based only
- Limited OS Support (No Windows)
- Go and Lua is not hacker's friendly
- **Syscall Forwarding**

```
$ pwd
/home/xwings/qemu-3.1.0
$ uname -a
FreeBSD freebsd 12.0-RELEASE FreeBSD 12.0-RELEASE r341666 GENERIC  amd64
$ ./configure --help

Usage: configure [options]
Options: [defaults in brackets after descriptions]

Standard options:
  --help                   print this message
  --prefix=PREFIX          install in PREFIX [/usr/local]
  --interp-prefix=PREFIX   where to find shared libraries, etc.
                           use %M for cpu name [/usr/gnemul/qemu-%M]
  --target-list=LIST       set target list (default: build everything)
                           Available targets: aarch64-softmmu alpha-softmmu
                           arm-softmmu cris-softmmu hppa-softmmu i386-softmmu
                           lm32-softmmu m68k-softmmu microblaze-softmmu
                           microblazeel-softmmu mips-softmmu mips64-softmmu
                           mips64el-softmmu mipsel-softmmu moxie-softmmu
                           nios2-softmmu or1k-softmmu ppc-softmmu ppc64-softmmu
                           riscv32-softmmu riscv64-softmmu s390x-softmmu
                           sh4-softmmu sh4eb-softmmu sparc-softmmu
                           sparc64-softmmu tricore-softmmu unicore32-softmmu
                           x86_64-softmmu xtensa-softmmu xtensaeb-softmmu
                           i386-bsd-user sparc-bsd-user sparc64-bsd-user
                           x86_64-bsd-user
```

How It Works

How Does It Work



Base OS can be Windows/Linux/BSD or OSX

And not limited to ARCH

OS Adventure

Loader

```python
class ELFParse:
    def __init__(self, path, ql):
        self.path = path
        self.ql = ql

        with open(path, "rb") as f:
            self.elfdata = f.read()

        self.ident = self.getident()

        if self.ident[ : 4] != b'\x7fELF':
            ql.nprint(">>> ERROR: NOT a ELF")
            exit(1)

        if self.ident[0x4] == 1: # 32 bit
            self.is32bit = True
        else:
            self.is32bit = False

        if self.ident[0x4] == 2: # 64 bit
            self.is64bit = True
        else:
            self.is64bit = False

        if self.ident[0x5] == 1: # little endian
            self.endian = 1
        elif self.ident[0x5] == 2: # big endian
            self.endian = 2
```

```python
class PE32:
    def __init__(self, ql, path=""):
        self.ql = ql
        self.uc = ql.uc
        self.path = path
        self.PE_IMAGE_BASE = 0
        self.PE_IMAGE_SIZE = 0
        self.PE_ENTRY_POINT = 0
        self.sizeOfStackReserve = 0
        self.dlls = {}
        self.import_symbols = {}
        self.import_address_table = {}
        self.cmdline = ''
        self.filepath = ''

    def loadx86Shellcode(self, dlls):
        self.initTEB()
        self.initPEB()
        self.initLdrData()
        for each in dlls:
            self.loadDll(each)

    def loadPE32(self):
        self.pe = pefile.PE(self.path, fast_load=True)

        # for simplicity, no image base relocation
        self.ql.PE_IMAGE_BASE = self.PE_IMAGE_BASE = self.pe.OPTIONAL_HEADER.ImageBase
        self.ql.PE_IMAGE_SIZE = sel PE_ENTRY_POINT: int f.pe.OPTIONAL_HEADER.SizeOfImage
        self.ql.entry_point = self.PE_ENTRY_POINT = self.PE_IMAGE_BASE + self.pe.OPTIONAL_HEADER.AddressOfEntryPoint
        self.sizeOfStackReserve = self.pe.OPTIONAL_HEADER.SizeOfStackReserve
        self.ql.nprint(">>> Loading %s to 0x%x" % (self.path, self.PE_IMAGE_BASE))
```

- ELF Loader
- PE Loader
- MACHO Loader

Parse != Loader

Posix Series - Syscall Emulator

```python
def ql_syscall_read(ql, uc, read_fd, read_buf, read_len, null0, null1, null2):
    path = (ql_read_string(ql, uc, read_buf))

    if read_fd < 256 and ql.file_des[read_fd] != 0:
        try:
            if isinstance(ql.file_des[read_fd], socket.socket):
                data = ql.file_des[read_fd].recv(read_len)
            else:
                data = ql.file_des[read_fd].read(read_len)
            uc.mem_write(read_buf, data)
            ql.nprint("|--->>> Read Completed %s" % path)
            regreturn = len(data)
        except:
            regreturn = -1
    else:
        regreturn = -1
    ql.nprint("read(%d, 0x%x, 0x%x) = %d" % (read_fd, read_buf, read_len, regreturn))
    ql_definesyscall_return(ql, uc, regreturn)


def ql_syscall_lseek(ql, uc, lseek_fd, lseek_ofset, lseek_origin, null0, null1, null2):
    ql.file_des[lseek_fd].seek(lseek_ofset, lseek_origin)
    regreturn = (ql.file_des[lseek_fd].tell())
    ql.nprint("lseek(%d, 0x%x, 0x%x) = %d" % (lseek_fd, lseek_ofset, lseek_origin, regreturn))
    ql_definesyscall_return(ql, uc, regreturn)


def ql_syscall_brk(ql, uc, brk_input, null0, null1, null2, null3, null4):
    ql.nprint("|--->>> brk(0x%x)" % brk_input)
    if brk_input != 0:
        if brk_input > ql.brk_address:
            uc.mem_map(ql.brk_address, (int(((brk_input + 0xfff) // 0x1000) * 0x1000 - ql.brk_address)))
            ql.brk_address = int(((brk_input + 0xfff) // 0x1000) * 0x1000)
    else:
        brk_input = ql.brk_address
    ql_definesyscall_return(ql, uc, brk_input)
    ql.nprint("|--->>> brk return(0x%x)" % ql.brk_address)


def ql_syscall_mprotect(ql, uc, mprotect_start, mprotect_len, mprotect_prot, null0, null1, null2):
    regreturn = 0
    ql.nprint("mprotect(0x%x, 0x%x, 0x%x) = %d" % (mprotect_start, mprotect_len, mprotect_prot, regreturn))
    ql_definesyscall_return(ql, uc, regreturn)
```

- Syscall almost the same for OSX/Linux/*BSD
- Kernel Programming 101
- Emulate Syscall
- Skip/Forward or Emulate Code
- Prepare Execution Report

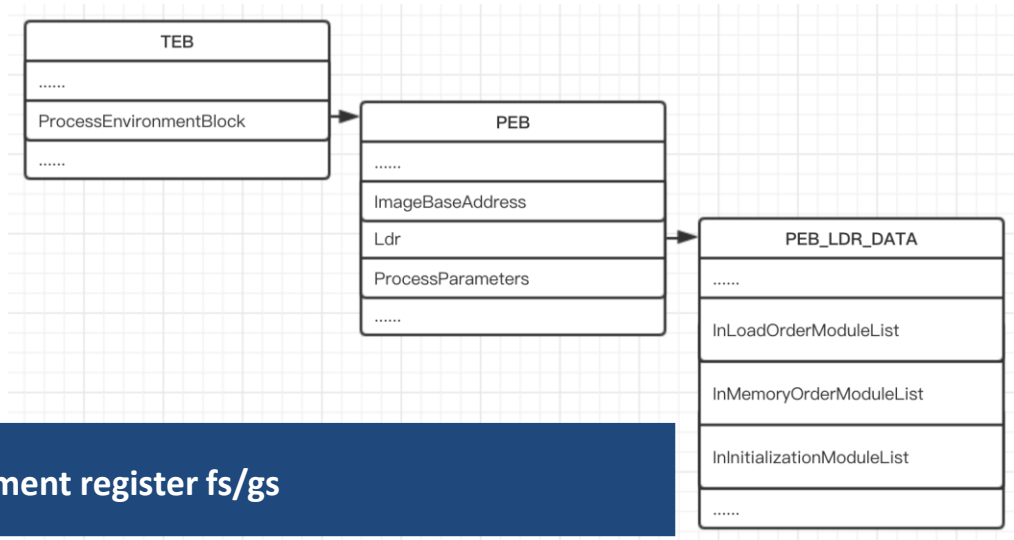
Syscall Implementation

Windows Emulator 0x1

```python
def setup_gdt_segment(uc, GDT_ADDR, GDT_LIMIT, seg_reg, index, SEGMENT_ADDR, SEGMENT_SIZE, init = True):

    # map GDT table
    if init:
        uc.mem_map(GDT_ADDR, GDT_LIMIT)

    # map this segment in
    uc.mem_map(SEGMENT_ADDR, SEGMENT_SIZE)

    # create GDT entry
    gdt_entry = create_gdt_entry(SEGMENT_ADDR, SEGMENT_SIZE, A_PRESENT | A_DATA | A_DATA_WRITABLE | A_PRIV_3 |

    # then write GDT entry into GDT table
    uc.mem_write(GDT_ADDR + (index << 3), gdt_entry)

    # setup GDT by writing to GDTR
    uc.reg_write(UC_X86_REG_GDTR, (0, GDT_ADDR, GDT_LIMIT, 0x0))

    # create segment index
    selector = create_selector(index, S_GDT | S_PRIV_3)
    # point segment register to this selector
    uc.reg_write(seg_reg, selector)
```

```python
def set_gs_msr(uc, SEGMENT_ADDR, SEGMENT_SIZE):

    uc.mem_map(SEGMENT_ADDR, SEGMENT_SIZE)
    uc.msr_write(GSMSR, SEGMENT_ADDR)
```

```python
def init_TEB_PEB(uc):
    print(">> TEB addr is " + hex(config64.GS_LAST_BASE))
    TEB_SIZE = len(TEB(0).tobytes())
    teb_data = TEB(base = config64.GS_LAST_BASE, PEB_Address = config64.GS_LAST_BASE + TEB_SIZE)
    uc.mem_write(config64.GS_LAST_BASE, teb_data.tobytes())
    config64.GS_LAST_BASE += TEB_SIZE
    data = teb_data.tobytes()

    print(">> PEB addr is " + hex(config64.GS_LAST_BASE))
    PEB_SIZE = len(PEB(0).tobytes())
    peb_data = PEB(base = config64.GS_LAST_BASE, LdrAddress = config64.GS_LAST_BASE + PEB_SIZE)
    uc.mem_write(config64.GS_LAST_BASE, peb_data.tobytes())
    config64.GS_LAST_BASE += PEB_SIZE

    LDR_SIZE = len(LDR(0).tobytes())
    ldr_data = LDR(base = config64.GS_LAST_BASE,
                InLoadOrderModuleList = {'Flink' : config64.GS_LAST_BASE + 0x10, 'Blink' : config64.GS_LAST_BASE + 0x10
                InMemoryOrderModuleList = {'Flink' : config64.GS_LAST_BASE + 0x20, 'Blink' : config64.GS_LAST_BASE + 0x
                InInitializationOrderModuleList = {'Flink' : config64.GS_LAST_BASE + 0x30, 'Blink' : config64.GS_LAST_B
    uc.mem_write(config64.GS_LAST_BASE, ldr_data.tobytes())
```



Setup segment register fs/gs

x86_32 : Setup GDT/GDTR

x86_64 : Use wrmsr to setup gs

Setup TEB Structure

Setup PEB Structure

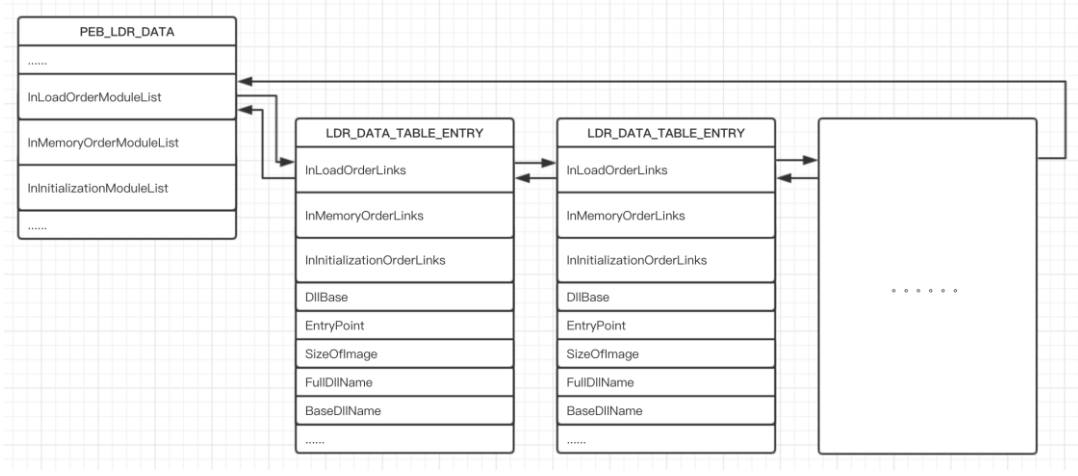
Setup PEB_LDR_DATA Structure

Windows Emulator 0x2

```
ldr_table = LDR_TABLE(LDR_base = config64.GS_LAST_BASE,
                      InLoadOrderLinks = {'Flink' : config64.LDR_TABLE_LIST[-1].InLoadOrderLinks['Flink'], 'Blink
                      InMemoryOrderLinks = {'Flink' : config64.LDR_TABLE_LIST[-1].InMemoryOrderLinks['Flink'], 'B
                      InInitializationOrderLinks = {'Flink' : config64.LDR_TABLE_LIST[-1].InInitializationOrderLi
                      DllBase = dll_base,
                      EntryPoint = 0,
                      FullDllName = path,
                      BaseDllName = fname,)

config64.LDR_TABLE_LIST[-1].InLoadOrderLinks['Flink'] = ldr_table.LDR_base
config64.LDR.InLoadOrderModuleList['Blink'] = ldr_table.LDR_base

config64.LDR_TABLE_LIST[-1].InMemoryOrderLinks['Flink'] = ldr_table.LDR_base + 0x10
config64.LDR.InMemoryOrderModuleList['Blink'] = ldr_table.LDR_base + 0x10

config64.LDR_TABLE_LIST[-1].InInitializationOrderLinks['Flink'] = ldr_table.LDR_base + 0x20
config64.LDR.InInitializationOrderModuleList['Blink'] = ldr_table.LDR_base + 0x20

uc.mem_write(config64.LDR.base, config64.LDR.tobytes())
uc.mem_write(config64.LDR_TABLE_LIST[-1].LDR_base, config64.LDR_TABLE_LIST[-1].tobytes())
uc.mem_write(ldr_table.LDR_base, ldr_table.tobytes())
```

```
if address in utils64.import_symbols:
    try:
        globals()['hook_' + utils64.import_symbols[address].decode()](uc, address, esp)
    except KeyError as e:
        print("[!]", e, "\t is not implemented")
```

```
def hook_LoadLibraryA(uc, rip, rsp):
    rip_saved = pop64(uc)
    (lpLibFileNameAddr,) = tuple(parse_arg(uc, 1))
    lpLibFileName = string_pack(uc.mem_read(lpLibFileNameAddr, 0x100))

    print('0x%0.2x:\tcall LoadLibraryA(\'%s\')' % (rip_saved, lpLibFileName))

    dll_base = dll_loader(uc, lpLibFileName)

    push64(uc, rip_saved)
    uc.reg_write(UC_X86_REG_RAX, dll_base)
```



InMemoryOrderModuleList

InLoadOrderModuleList

InInitializationOrderList

Setup LDR_DATA_TABLE_ENTRY for Loaded Modules

Setup Three Double Linked Lists

Parse DLL & Get All Export Functions

Hook Windows API

Sample Code on How To Execute X86_32/64bit Windows PE

CPU Adventure

X86 32/64 Series

```
QL_X86_F_GRANULARITY = 0x8
QL_X86_F_PROT_32 = 0x4
QL_X86_F_LONG = 0x2
QL_X86_F_AVAILABLE = 0x1

QL_X86_A_PRESENT = 0x80

QL_X86_A_PRIV_3 = 0x60
QL_X86_A_PRIV_2 = 0x40
QL_X86_A_PRIV_1 = 0x20
QL_X86_A_PRIV_0 = 0x0

QL_X86_A_CODE = 0x10
QL_X86_A_DATA = 0x10
QL_X86_A_TSS = 0x0
QL_X86_A_GATE = 0x0
QL_X86_A_EXEC = 0x8

QL_X86_A_DATA_WRITABLE = 0x2
QL_X86_A_CODE_READABLE = 0x2
QL_X86_A_DIR_CON_BIT = 0x4

QL_X86_S_GDT = 0x0
QL_X86_S_LDT = 0x4
QL_X86_S_PRIV_3 = 0x3
QL_X86_S_PRIV_2 = 0x2
QL_X86_S_PRIV_1 = 0x1
QL_X86_S_PRIV_0 = 0x0

QL_X86_GDT_ADDR = 0x3000
QL_X86_GDT_LIMIT = 0x1000
QL_X86_GDT_ENTRY_SIZE = 0x8
```

X86 32/64bit GDT For Linux

```
ql_x86_setup_gdt_segment_ds(ql, ql.uc)
ql_x86_setup_gdt_segment_cs(ql, ql.uc)
ql_x86_setup_gdt_segment_ss(ql, ql.uc)
```

X86 32bit GDT For Windows

```
# New set GDT Share with Linux
ql_x86_setup_gdt_segment_fs(ql, ql.uc, ql.FS_SEGMENT_ADDR, ql.FS_SEGMENT_SIZE)
ql_x86_setup_gdt_segment_gs(ql, ql.uc, ql.GS_SEGMENT_ADDR, ql.GS_SEGMENT_SIZE)
ql_x86_setup_gdt_segment_ds(ql, ql.uc)
ql_x86_setup_gdt_segment_cs(ql, ql.uc)
ql_x86_setup_gdt_segment_ss(ql, ql.uc)
```

X86 64bit GDT For Windows

```
def set_pe64_gdt(ql):
    # uc.mem_map(GS_SEGMENT_ADDR, GS_SEGMENT_SIZE)
    # setup_gdt_segment(uc, GDT_ADDR, GDT_LIMIT, UC_X86_REG_
    GSMSR = 0xC0000101
    ql.uc.mem_map(ql.GS_SEGMENT_ADDR, ql.GS_SEGMENT_SIZE)
    ql.uc.msr_write(GSMSR, ql.GS_SEGMENT_ADDR)
```

It took us sometime to fix the GDT and Set Thread Area

ARM/64 Series

```
main mcr: str
    mcr p15, 0, r0, c13, c0, 3
    adr r1, ret_to
    add r1, r1, #1
    bx r1
.THUMB
```

```
def ql_arm_init_kernel_get_tls(uc):
    uc.mem_map(0xFFFF0000, 0x1000)
    sc = 'adr r0, data; ldr r0, [r0]; mov pc, lr; data:.ascii "\x00\x00"'
```

```
def ql_arm64_enable_vfp(uc):
    ARM64FP = uc.reg_read(UC_ARM64_REG_CPACR_EL1)
    ARM64FP |= 0x300000
    uc.reg_write(UC_ARM64_REG_CPACR_EL1, ARM64FP)
```

- ARM/Thumb and ARM64
- Making Sure Loader is compatible
- ARM MCR instruction for Set TLS
- ARM Kernel Initialization
- ARM and ARM64 Enable VFP

MIPS32EL Series

unicorn-engine / unicorn

Code | Issues 262 | Pull requests 32 | Projects 0 | Wiki

Removed hardcoded CP0C3_ULRI (#1098)

```
* activate CP0C3_ULRI for CONFIG3, mips

* updated with mips patches

* updated with mips patches

* remove hardcoded config3

* git ignore vscode

* fix spacing issue and turn on floating point
```

master (#1098)

xwings authored and aquynh committed on Jul 6 1

Showing 12 changed files with 45 additions and 10 deletions.

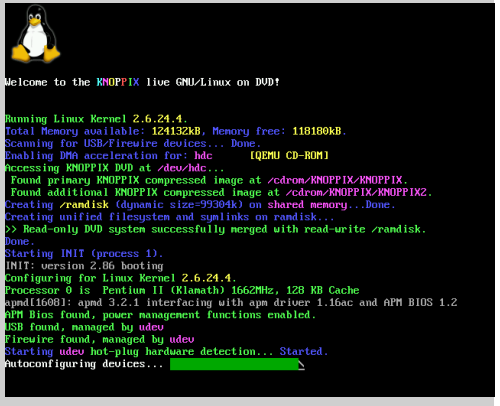
```
    sw $ra, -8($sp)
    sw $a0, -12($sp)
    sw $a1, -16($sp)
    sw $a2, -20($sp)
    sw $a3, -24($sp)
    sw $v0, -28($sp)
    sw $v1, -32($sp)
    sw $t0, -36($sp)

    slti $a2, $zero, -1
lab1:
    bltzal $a2, lab1

    addu $a1, $ra, 140
    addu $t0, $ra, 60
    lw $a0, -4($sp)
    li $a2, 8
    jal $t0
    nop

    lw $ra, -8($sp)
    lw $a0, -12($sp)
    lw $a1, -16($sp)
    lw $a2, -20($sp)
    lw $a3, -24($sp)
    lw $v0, -28($sp)
    lw $v1, -32($sp)
    lw $t0, -36($sp)
    j 0
    nop


my_mem_cpy:
    move    $a3, $zero
    move    $a3, $zero
    b       loc_400804
    nop
```

- MIPS Comes with CO Processor
- Configuration needed for CO Processor
- Unicorn does not support Floating Point
- Patch Unicorn to Support CO Processors
- Custom Binary Injected for Set Thread Area

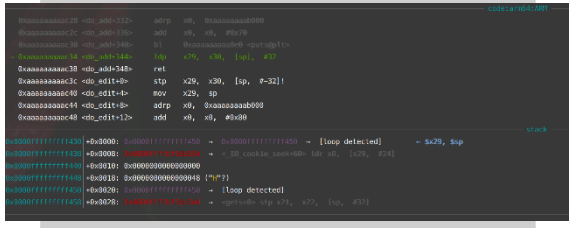
Demo Setup

VirtualBox or VMware

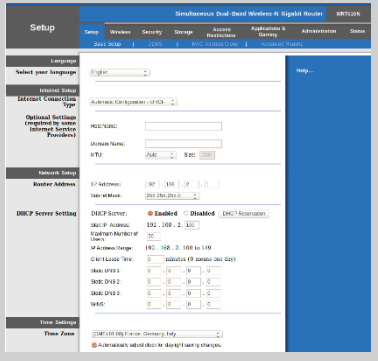
Linux Demo



X86
Reversing.kr Challenge



ARM64 Debug Mode



ARM/MIPS
Wi-Fi Router Firmware

VMware with Ubuntu 64Bit on XPS

Simple Crackme Challenge

```
run_one_round: run_one_round
def run_one_round(payload):
    stdin = MyPipe()
    ql = Qiling(["rootfs/x86_linux/bin/crackme_linux"], "rootfs/x86_linux", output = "off", stdin = stdin, stdout = sys.stdout
    ins_count = [0]
    ql.hook_code(instruction_count, ins_count)
    stdin.write(payload)
    ql.run()
    del stdin
    del ql
    return ins_count[0]


def solve():
    idx_list = [1, 4, 2, 0, 3]

    flag = b'\x00\x00\x00\x00\x00\n'

    old_count = run_one_round(flag)
    for idx in idx_list:
        for i in b'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~ ':
            flag = flag[ : idx] + chr(i).encode() + flag[idx + 1 : ]
            tmp = run_one_round(flag)
            if tmp > old_count:
                old_count = tmp
                break
        # if idx == 2:
        #     break

    print(flag)


if __name__ == "__main__":
    solve()
```

Brute Forcer

ARM HelloWorld

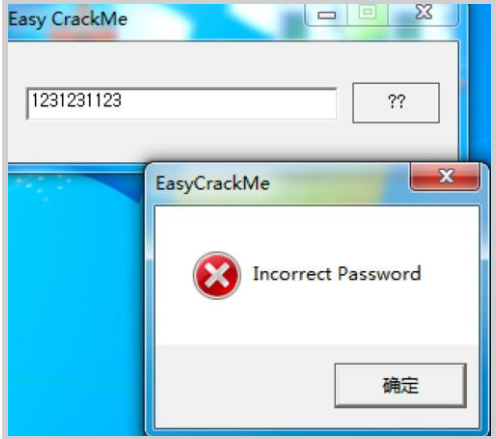
```python
from qiling import *

def run_sandbox(path, rootfs, ostype, output):
    ql = Qiling(path, rootfs, ostype = ostype, output = output)
    ql.run()


if __name__ == "__main__":
    run_sandbox(["rootfs/arm_linux/bin/arm32-hello-static"], "rootfs/arm_linux", "linux", "debug")
```

Debug Mode

Windows Demo

Real CTF Challenge

Half “Cooked” Wannacry



Patching a
CrackMe Challenge

Emulating Windows DialogBox within Qiling

Real World CTF Challenge

```python
from qiling import *

class StringBuffer:
    def __init__(self):
        self.buffer = ''

    def read(self, n):
        ret = self.buffer[:n]
        self.buffer = self.buffer[n:]
        return ret

    def write(self, string):
        self.buffer += string
        return len(string)


def instruction_count(uc, address, size, user_data):
    user_data[0] += 1

def get_count(flag):
    ql = Qiling(["rootfs/x86_windows/bin/crackme.exe"], "rootfs/x86_windows", output = "off")
    ql.stdin = StringBuffer()
    ql.stdin.write("".join(flag) + "\n")
    count = [0]
    ql.hook_code(instruction_count, count)
    ql.run()
    print(" ============ count: %d ============ " % count[0])
    return count[0]

def solve():
    # BJWXB_CTF{C5307D46-E70E-4038-B6F9-8C3F698B7C53}
    prefix = list("BJWXB_CTF{")
    flag = list("\x00"*100)
    base = get_count(prefix + flag)
    i = 0
    for i in range(len(flag)):
        for j in "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-{}":
            flag[i] = j
            data = get_count(prefix + flag)
            if data > base:
                base = data
                print("\n\n\n>>> FLAG: " + "".join(prefix + flag) + "\n\n\n")
                break
        if flag[i] == "}":
            break
    print("SOLVED!!!")

if __name__ == "__main__":
    solve()
```

Brute Force

Windows PE, Wannacry Execution

```python
from qiling import *

def stopatkillerswtich(ql):
    print("killerswtch found")
    ql.uc.emu_stop()

if __name__ == "__main__":
    ql = Qiling(["rootfs/x86_windows/bin/wannacry.bin"], "rootfs/x86_windows", output = "debug")
    ql.hook_address(stopatkillerswtich, 0x40819a)
    ql.run()
```

Catch Killer Switch

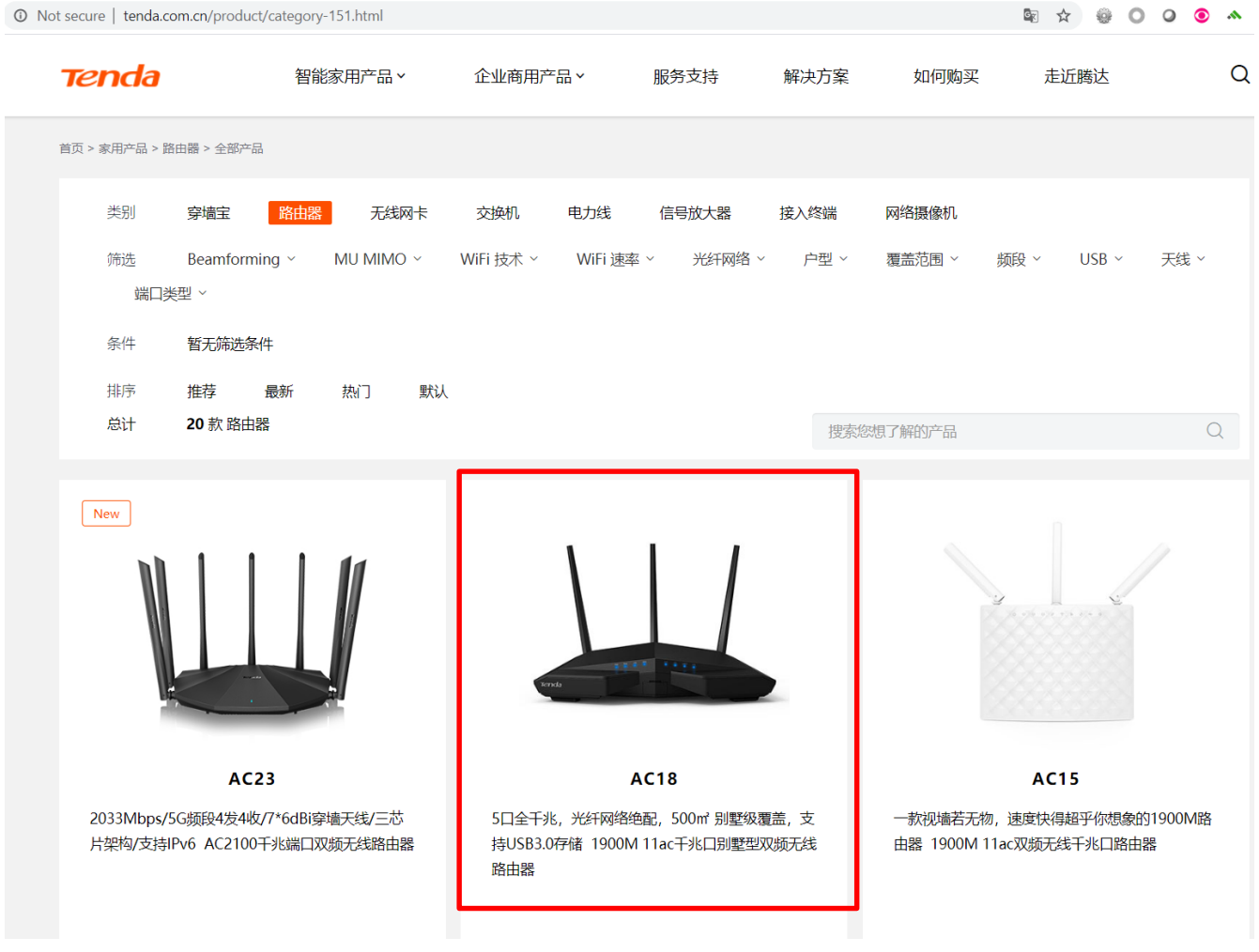
Windows Crack Me

```python
from qiling import *

def force_call_dialog_func(ql):
    # get DialogFunc address
    lpDialogFunc = ql.unpack32(ql.mem_read(ql.sp - 0x8, 4))
    # setup stack for DialogFunc
    ql.stack_push(0)
    ql.stack_push(1001)
    ql.stack_push(273)
    ql.stack_push(0)
    ql.stack_push(0x0401018)
    # force EIP to DialogFunc
    ql.pc = lpDialogFunc


def hook_memread(ql):
    print("demo for ql.hook_mem_read")


def my_sandbox(path, rootfs):
    ql = Qiling(path, rootfs)
    ql.patch(0x004010B5, b'\x90\x90')
    ql.patch(0x004010CD, b'\x90\x90')
    ql.patch(0x0040110B, b'\x90\x90')
    ql.patch(0x00401112, b'\x90\x90')
    ql.hook_mem_read(hook_memread, 0xffffdef4)
    ql.hook_address(force_call_dialog_func, 0x00401016)
    ql.run()


if __name__ == "__main__":
    my_sandbox(["rootfs/x86_windows/bin/Easy_CrackMe.exe"], "rootfs/x86_windows")
```

Patch without Changing the File Content

Qiling: Hands On Time

Emulate a Router



https://www.tenda.com.cn/product/category-151.html