

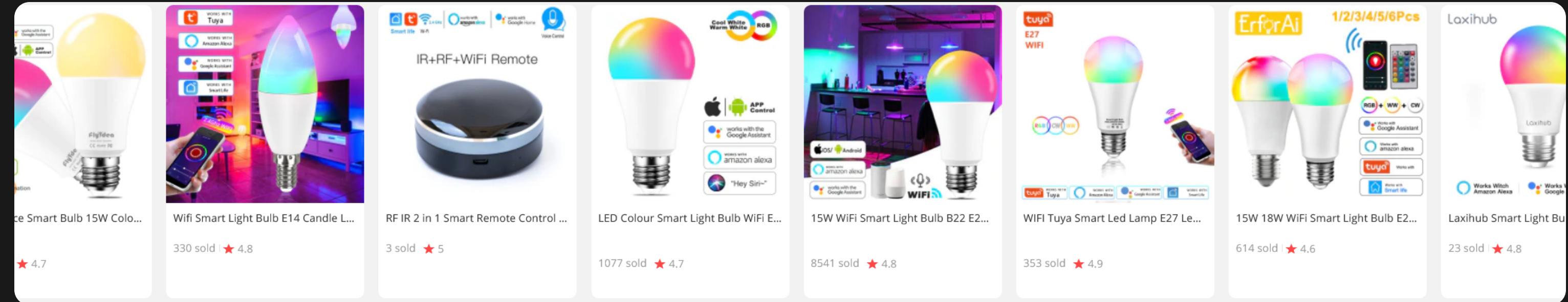
# “Smart” Vacuum Cleaners

An Audit Into The Security and Integrity of IoT Systems

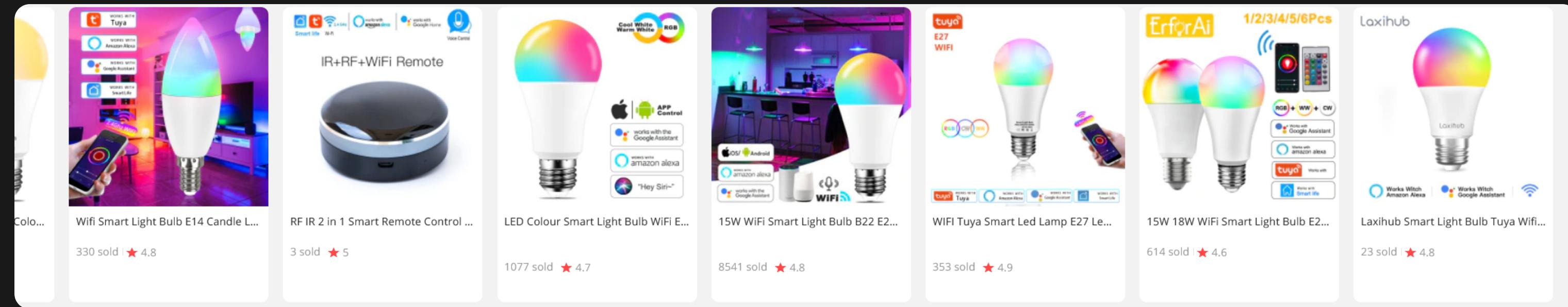
Andrew Wong | UNSW Sydney

# 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...*



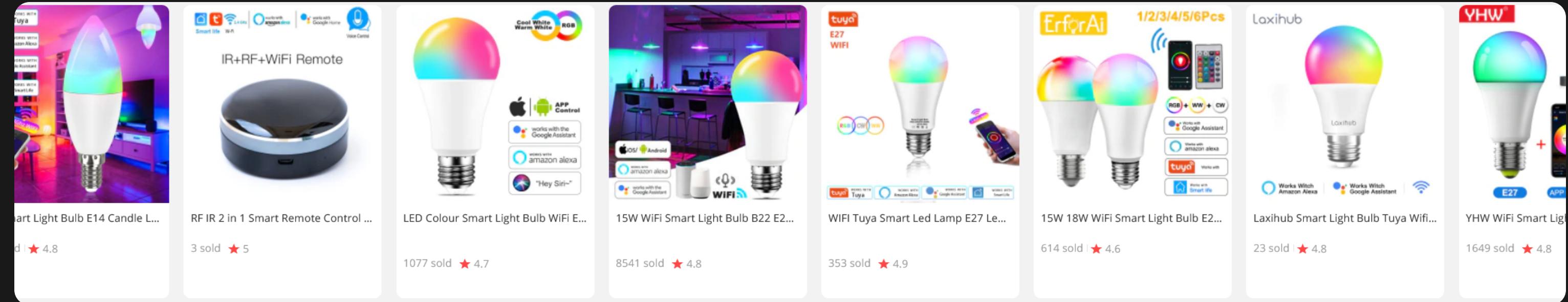
Are competing products looking suspiciously similar to you?  
Most are white-labelled products, the biggest ecosystem vendor being **tuya**

## 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 / Xiaomi / Tuya)

- 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)

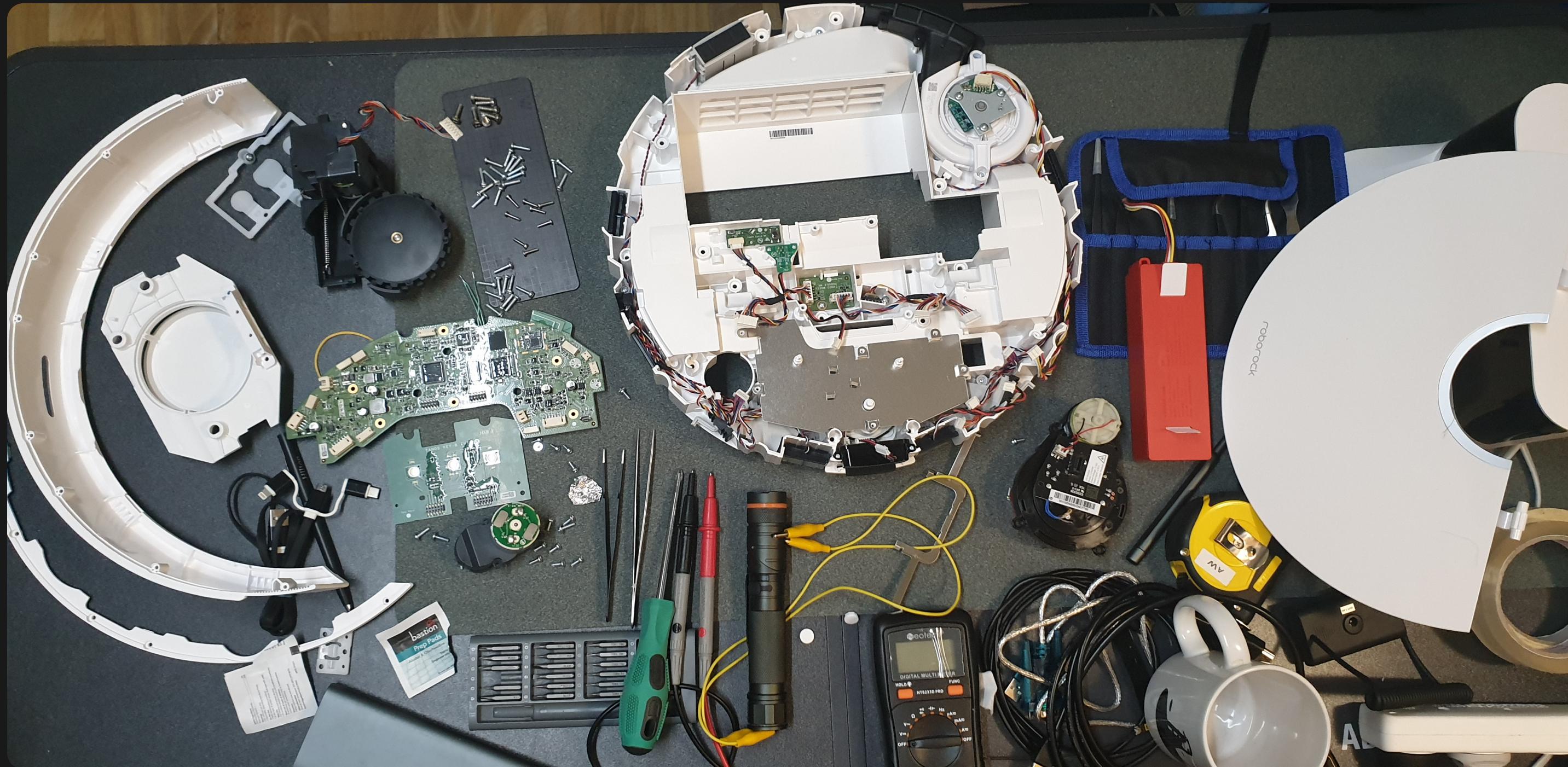


*It works pretty well, according to reviews.  
But is it safe to connect to your home?*

Where we left off

# Thesis A | Results

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

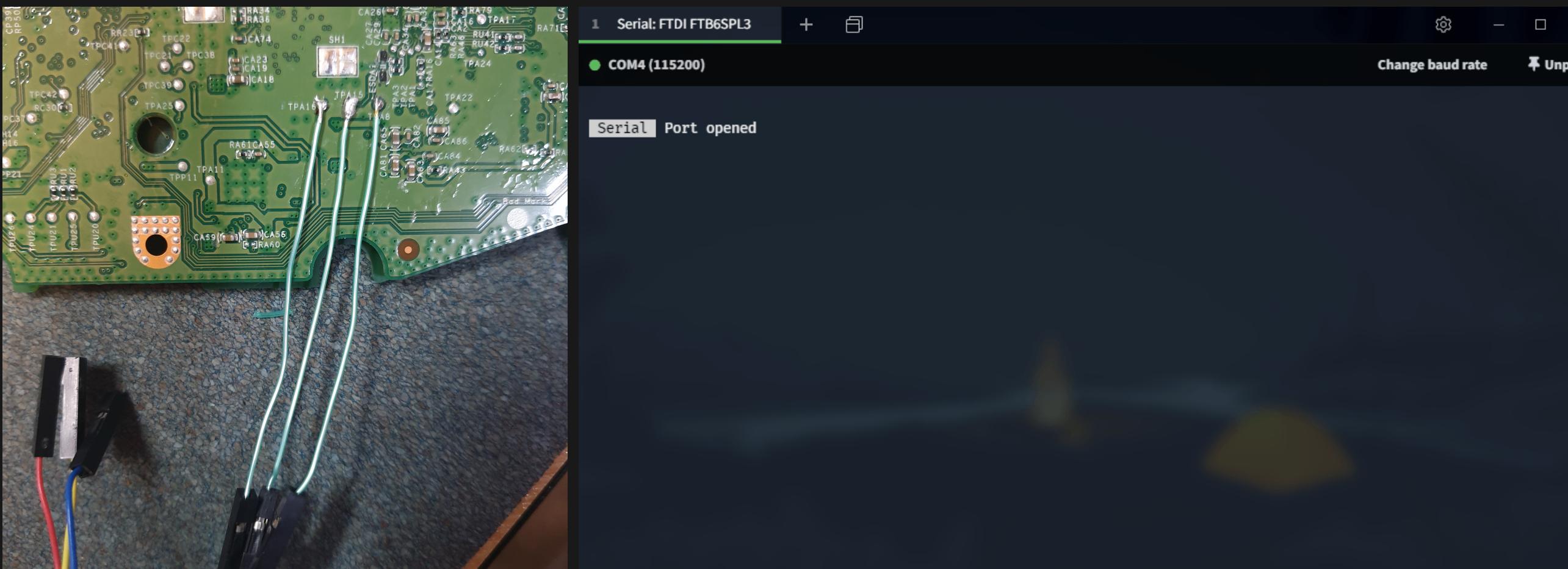


Thesis A - Found the UART pins and got some terminal

## 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

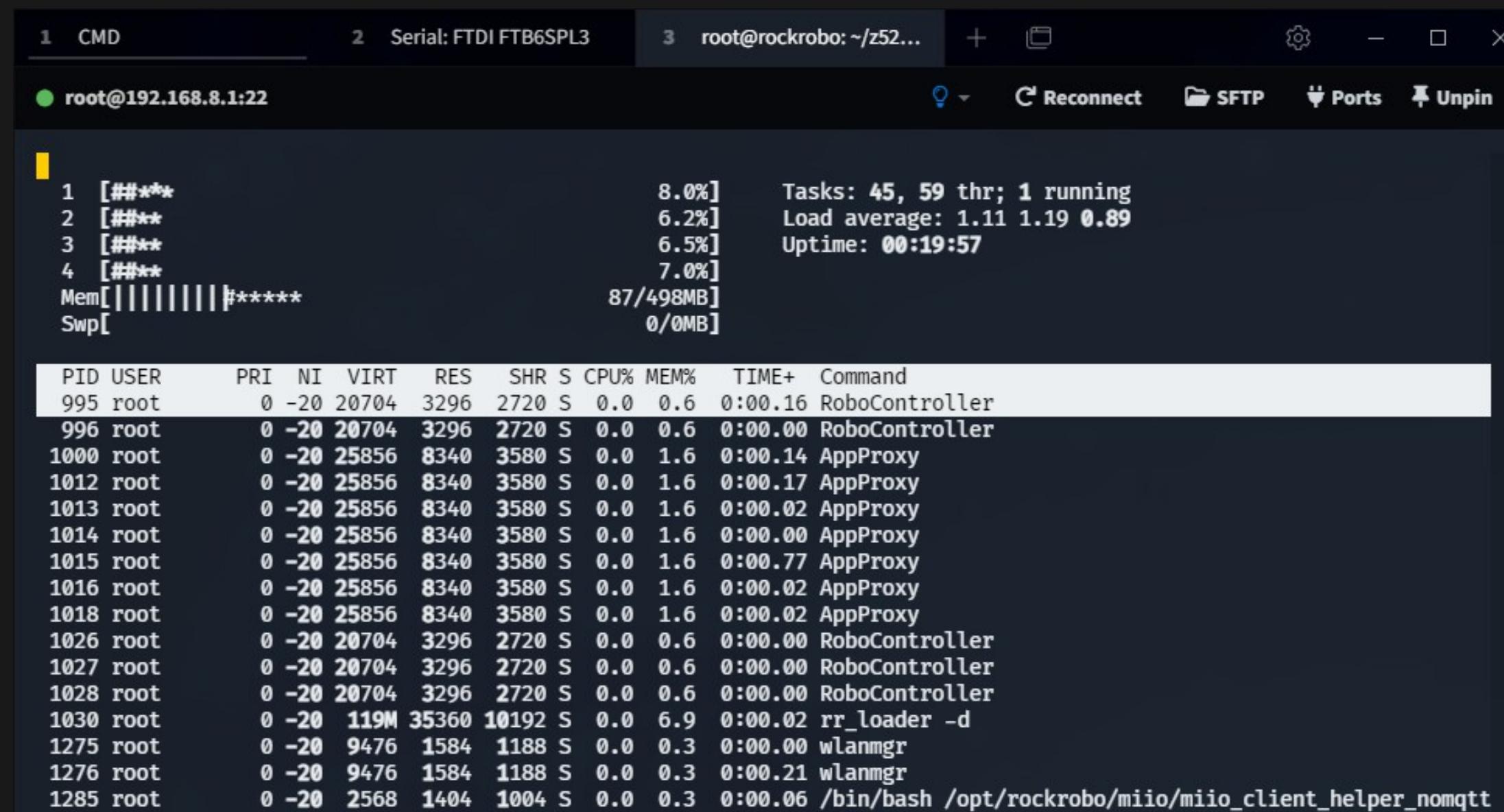
# Thesis B - Firmware dump (dd) for offline/static analysis

# 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?

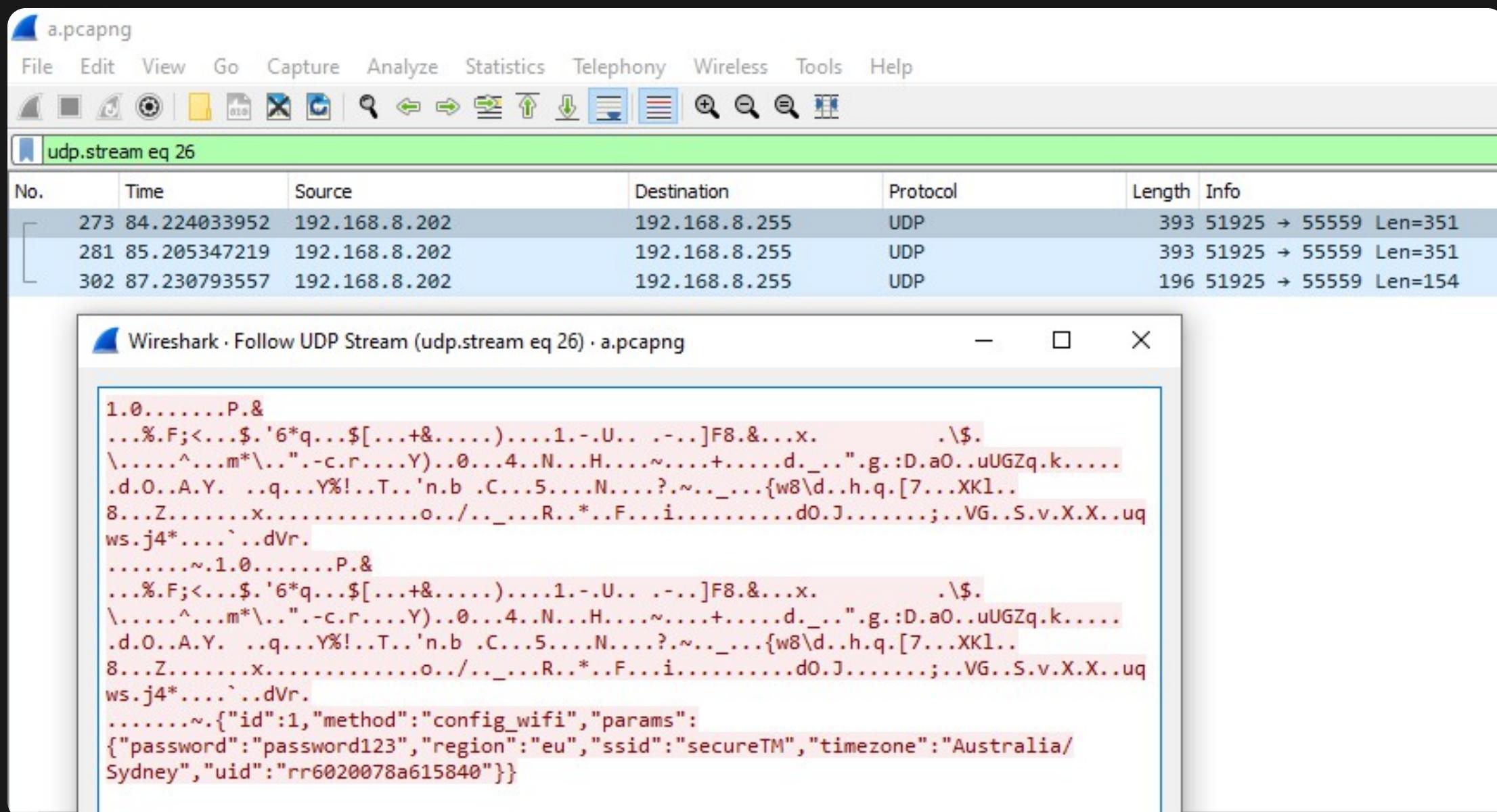
### *Recovery partition is modifiable*

- 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

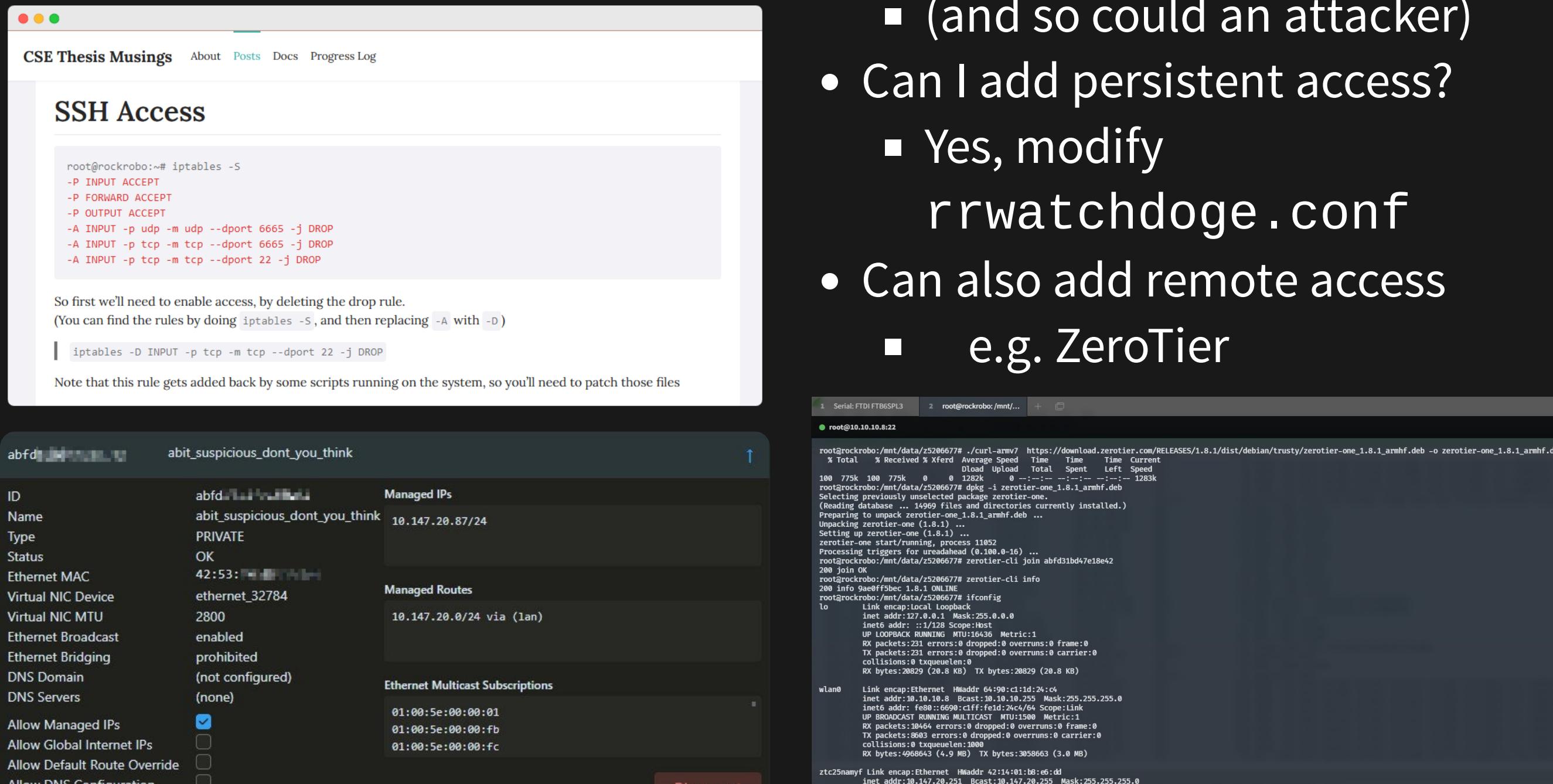
```
root@rockrobo:~# netstat -nltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 127.0.0.1:54322          0.0.0.0:*            LISTEN     991/mio_c
tcp      0      0 127.0.0.1:54323          0.0.0.0:*            LISTEN     991/mio_c
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_
tcp      0      0 0.0.0.0:6668           0.0.0.0:*            LISTEN     998/rriot_
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.

# Thesis B - Investigating adbd

## (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 did. hahah....
  - Compare file changes
- Factory reset device, check for remnant files

# Let's Talk Threats

*Talk about threat models*

- 
- 
- 
- 
-

# Firmware Comparison

/etc/OS\_VERSION

ro.product.device=MI1558\_TANOS\_MP\_S2020032500REL\_M3.3.0\_RELEASE\_20200325-204847

## Notes

- The Roborock S6 was released in June 2019
- The device I was working on was manufactured June 2020
- There has since been firmware upgrades since 2019
  - As a result my device has a firmware from 25th March 2020
- Note the presence of “MI” at the start of the product string

# Stock Ubuntu 14.04.3 LTS

Performed an md5 + diff between the firmware and the stock [Ubuntu 14.04.3 Core LTS \(armhf\)](#) firmware.

All binaries that were present on the device matched the stock firmware, except for `ntpdate` (synchronise computer time via NTP)

Whilst somewhat dissimilar in signature, still functionally equivalent.  
Just has less function exports than the stock version.

# Firmware Upgrade

```
-ro.product.device=MI1558_TANOS_MP_S2020032500REL_M3.3.0_RELEASE_20200325-204847
-ro.build.display.id=TANOS_MP_R16_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_20220428-202811
    ro.sys.cputype=R16.STM32.A3.G1
-ro.build.version.release=1558
-ro.build.date.utc=1585140527
+ro.build.version.release=V2902
+ro.build.date.utc=1651148891
```

Remember this?

*Update to latest version (and hope we don't get locked out)*

no longer use vinda

<https://featherbear.cc/UNSW-CSE-Thesis/posts/upgrade-notes/>

We got locked out ☺ <https://featherbear.cc/UNSW-CSE-Thesis/posts/upgraded-accessing-the-shell/>

# Lockdown: Admin

[mmcblk0p6]/vinda is no longer used!

Programs now request a verification from [mmcblk0p6]/shadow and [mmcblk0p6]/shadow.sign.

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

So they fail

Affected: rr\_login (serial), dropbear (SSH), adbd (USB)

## System Changes

```
-aa  
+Linux rockrobo 3.4.39 #1 SMP PREEMPT Thu Apr 28 20:27:25 CST 2022 armv7l GNU/Linux
```

\*: Also an optimisation to reduce space

# Lockdown: Shell

- Serial handler no longer uses getty - now a modified version called rr\_login
- OpenSSH replaced with modified dropbear - root access only
- rr\_login - root only [https://featherbear.cc/UNSW-CSE-Thesis/posts/sbin-rr\\_login/](https://featherbear.cc/UNSW-CSE-Thesis/posts/sbin-rr_login/)
  - used to be getty

# Lockdown: Firewall

- ipv6 blocked!
  - apps also no longer ping for AAAA records (fds)
- several processes re-run iptables block
  - rrlogd, watchdog

# Analysis of binaries Hooking into pre-encryption / post-decryption routines

# The IPv6 World

No IPv6 lease was assigned Regardless, there are now `ip6tables` rules to drop all Firewall

# OTA Rooting

OTA Update After 11/2019 (remember the product was release 06/2019, ours was manufactured 06/2020) - OTA was blocked Beforehand - we could remotely root it

<https://featherbear.cc/UNSW-CSE-Thesis/posts/ota-updates-blocked-as-of-late-2019/>

rrlogd (RSA + AES)

rriot\_rr /dev/shm/.migrate\_to\_rriot

encrypts the files before sending. Instead of decrypting this we can just look at the log files that it extracts

rriot\_tuya - used to have tuya\_iot\_impl\_upload\_file function but no longer has

Threat models

talk about miIO vuln miIO ota - patched

<https://featherbear.cc/UNSW-CSE-Thesis/posts/disclosures/> Response to disclosures tuya and roborock only have one disclosure listed on their site?

Hard to find xiaomi has a

Takeaway Summary (answer main point)

tldr Who What When Why Where How

# Address the statement

# Thesis in a Year

The screenshot displays a digital dashboard with a teal header bar containing the title "CSE Thesis Devlog TL;DR". Below the header, there are three columns of tasks:

- Research** (Left Column):
  - Wednesday 29/09/2021
  - Tuesday 5/10/2021
  - Tuesday 12/10/2021
  - Monday 18/10/2021
  - Monday 1/11/2021
  - Wednesday
- Hardware Hacking** (Middle Column):
  - Monday 25/10/2021
  - Tuesday 26/10/2021
  - Friday 29/10/2021
  - Sunday 01/05/2022
  - Friday 24/06/2022
- Software Hacking** (Right Column):
  - Monday 25/10/2021
  - Friday 29/10/2021
  - Saturday 30/10/2021
  - Wednesday 02/03/2022
  - Filesystem inspection 19/03/2022
  - 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)