

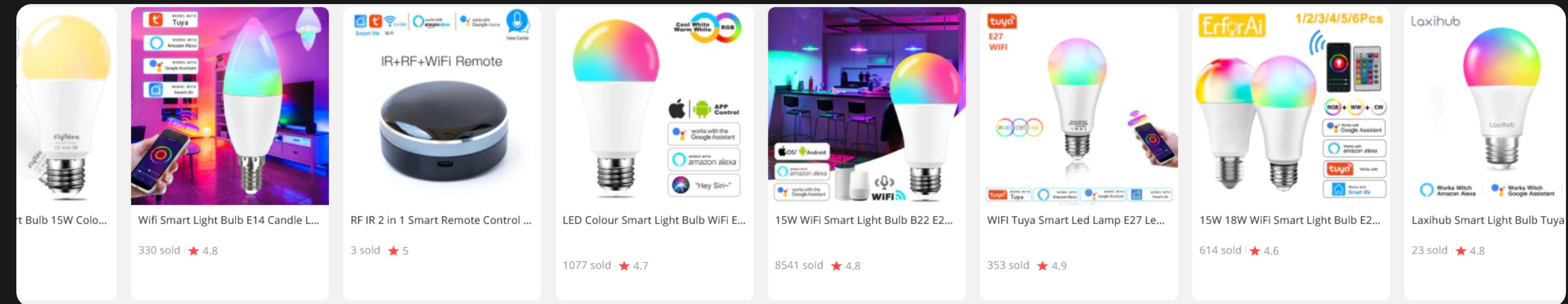
“Smart” Vacuum Cleaners

An Audit Into The Security and Integrity of IoT Systems

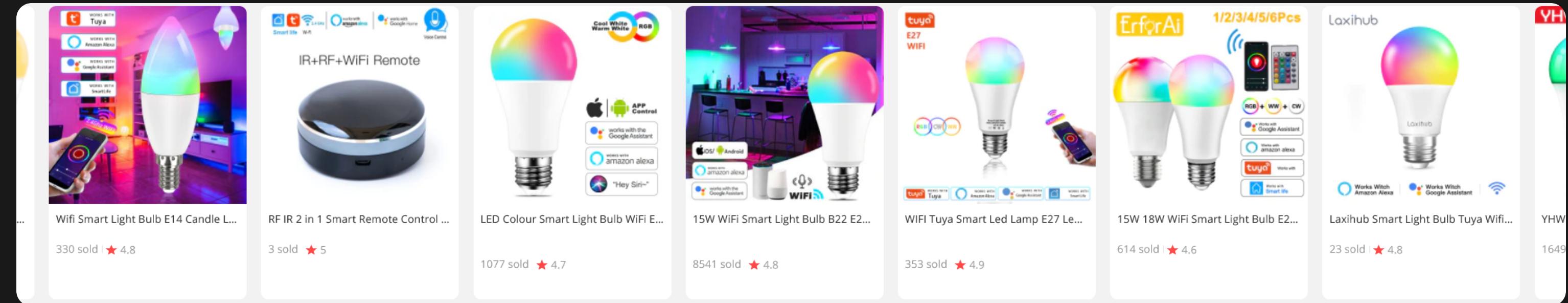
Andrew Wong | UNSW Sydney

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...



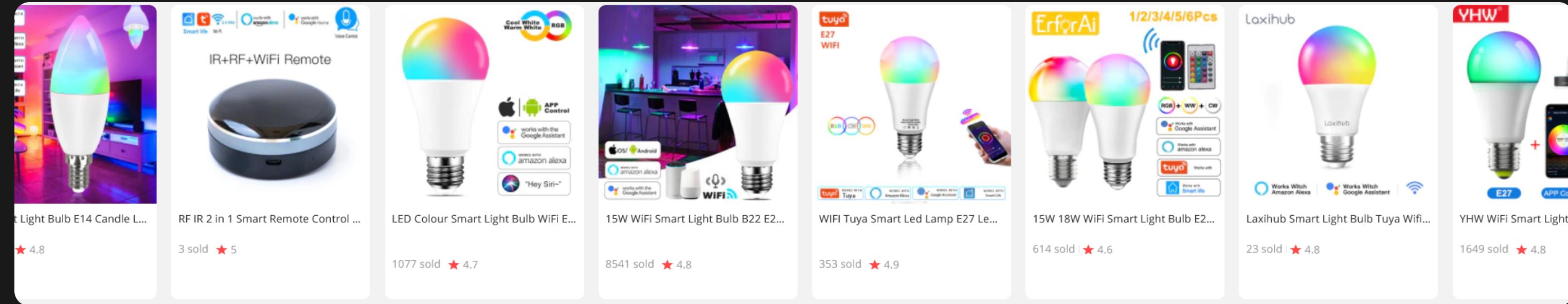
Are competing products looking suspiciously similar to you?
 Most are white-labelled products, the biggest ecosystem vendor being **tuya**

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)

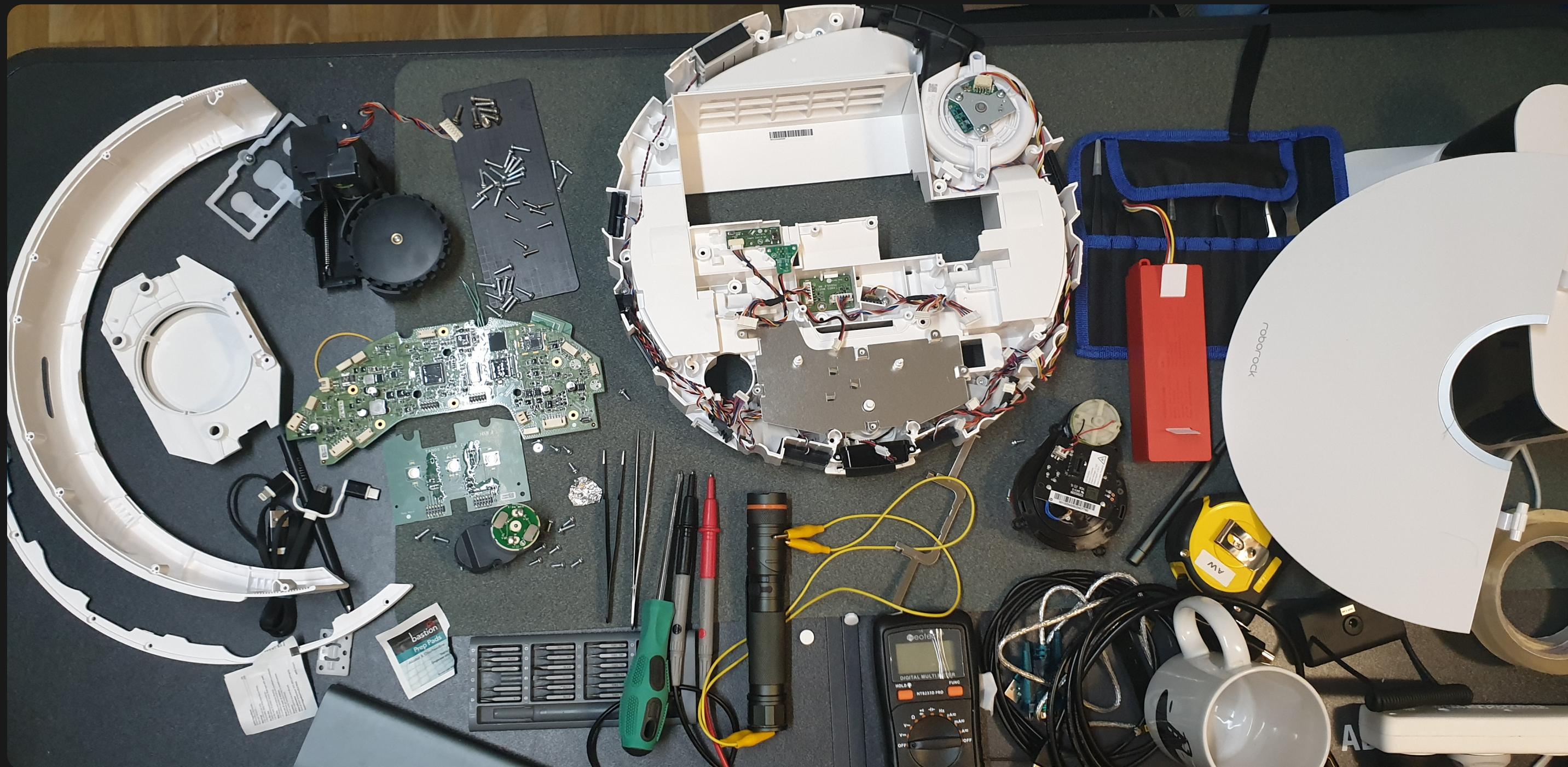


*It works pretty well, according to reviews.
But is it safe to connect to your home?*

Where we left off

Thesis A | Results

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

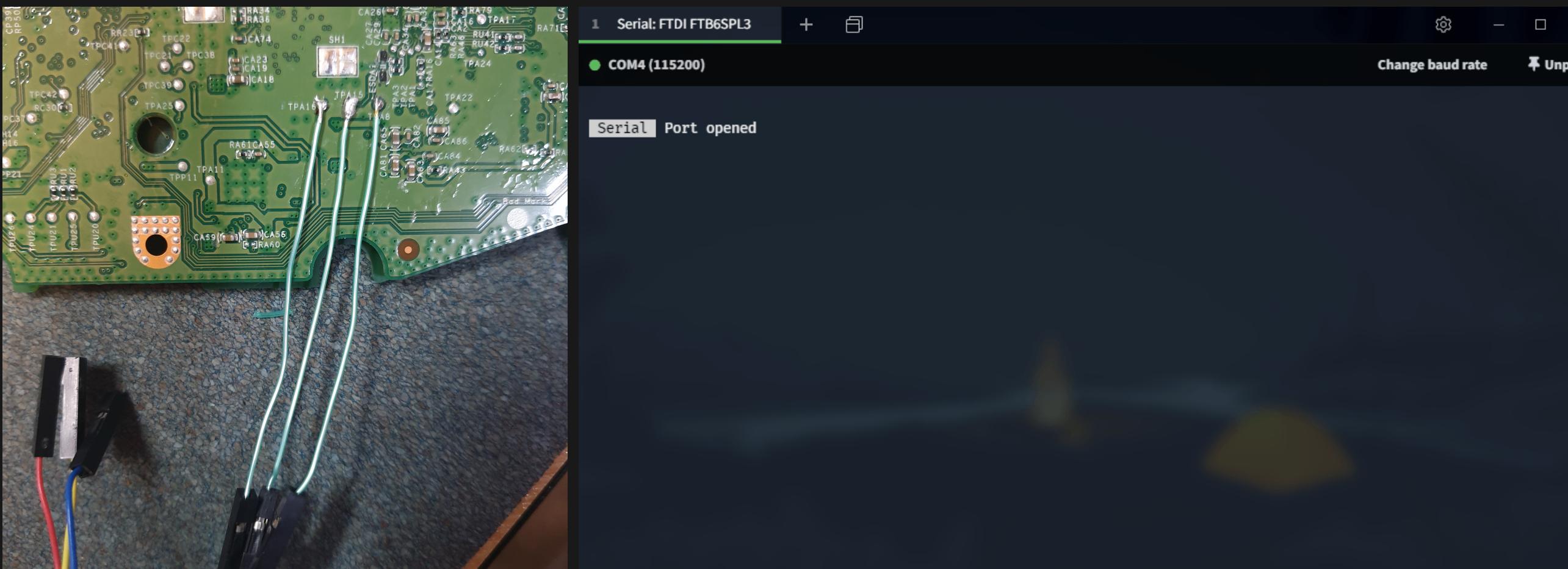


Thesis A - Found the UART pins and got some terminal

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

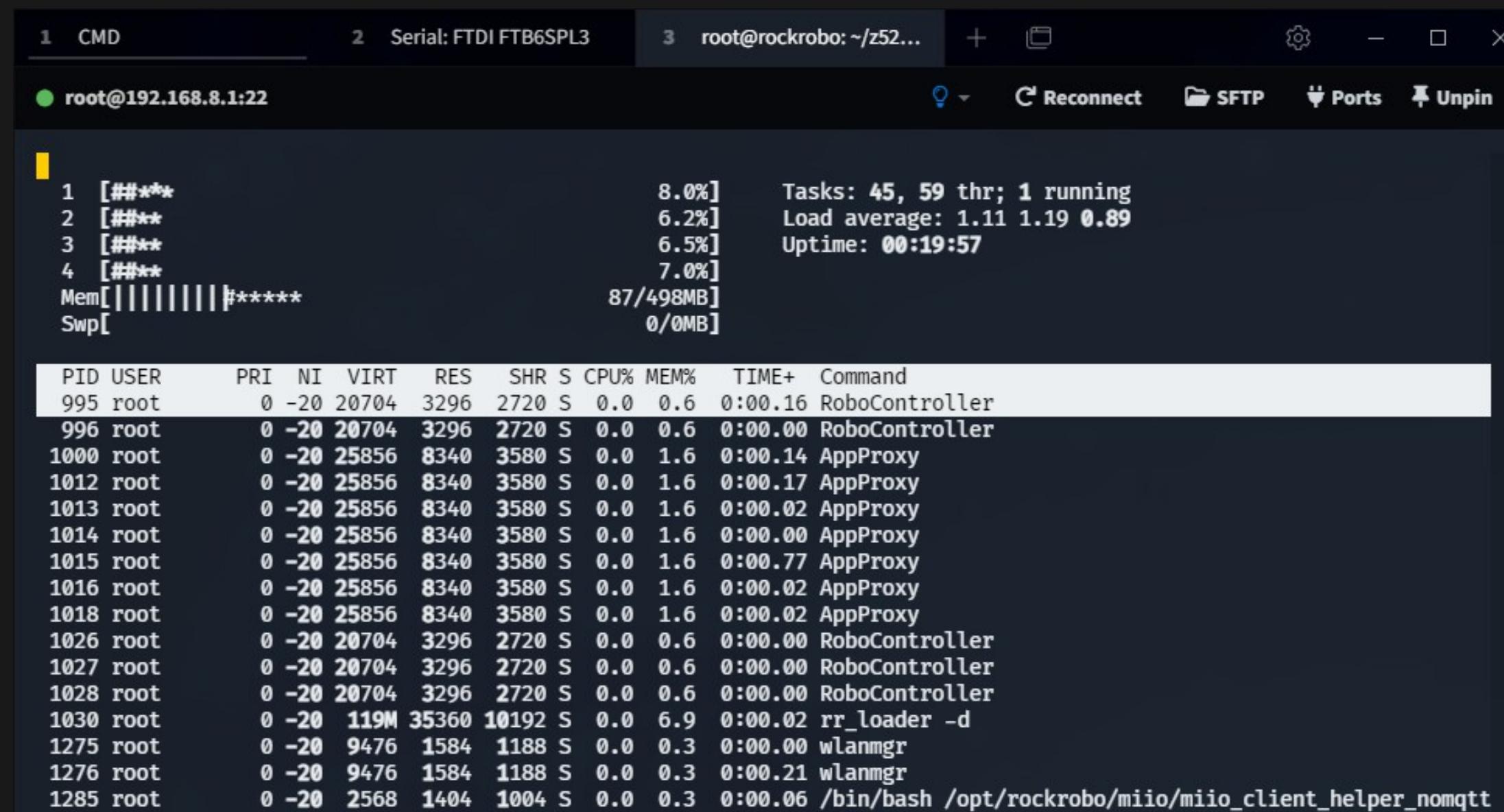
Thesis B - Firmware dump (dd) for offline/static analysis

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?

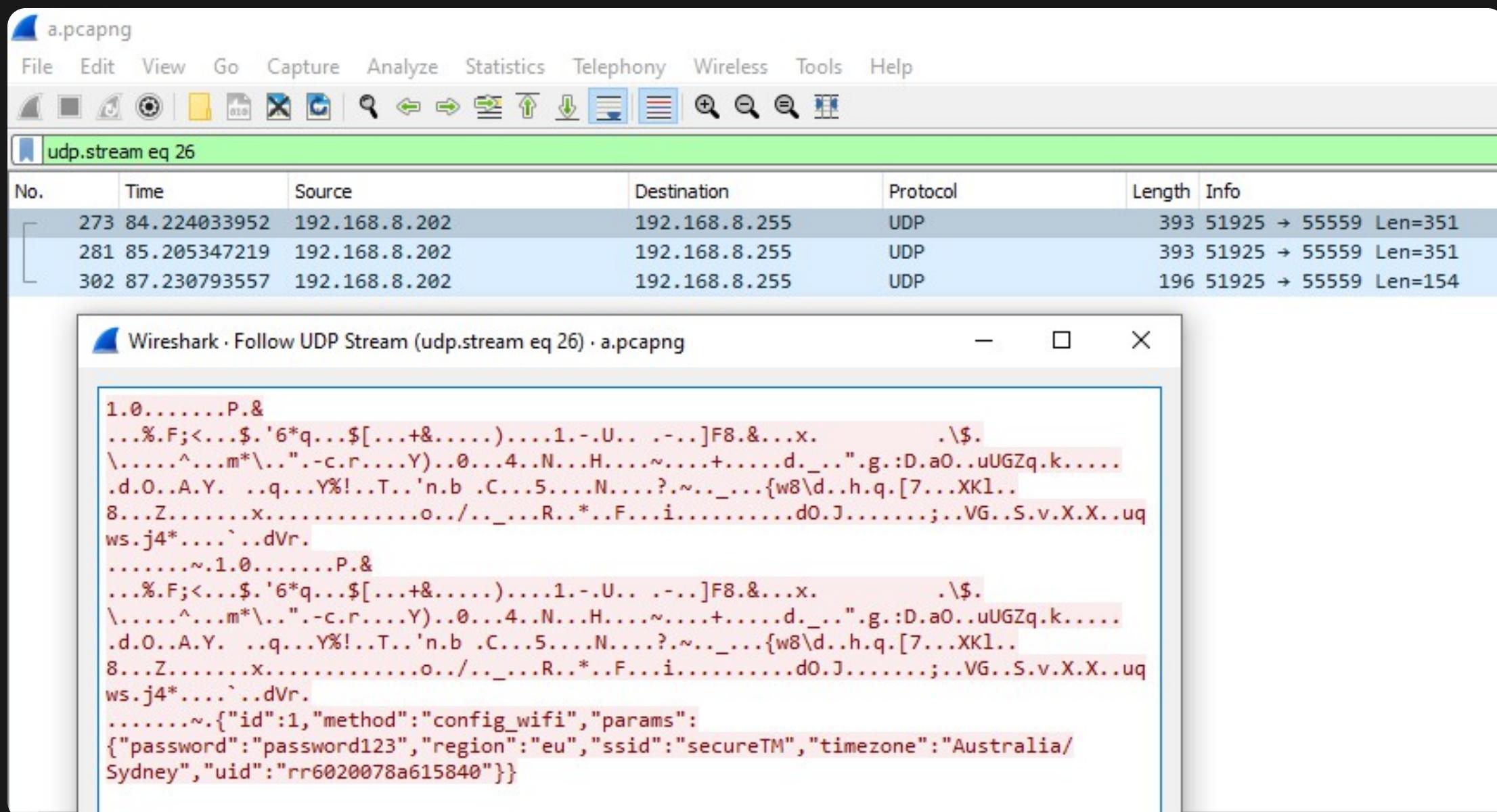
Recovery partition is modifiable

- 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/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

The screenshot shows a web application with a header "CSE Thesis Musings" and navigation links "About", "Posts", "Docs", "Progress Log". The main content is titled "SSH Access". It displays a terminal session with the following command and output:

```
root@rockrobo:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p udp -m udp --dport 6665 -j DROP
-A INPUT -p tcp -m tcp --dport 6665 -j DROP
-A INPUT -p tcp -m tcp --dport 22 -j DROP
```

Below the terminal, there is explanatory text:

So first we'll need to enable access, by deleting the drop rule.
(You can find the rules by doing `iptables -S`, and then replacing `-A` with `-D`)

```
| iptables -D INPUT -p tcp -m tcp --dport 22 -j DROP
```

Note that this rule gets added back by some scripts running on the system, so you'll need to patch those files

The screenshot shows a ZeroTier configuration interface. A device named "abit_suspicious_dont_you_think" is selected. The interface displays the following details:

ID	abfd	Managed IPs
Name	abit_suspicious_dont_you_think	10.147.20.87/24
Type	PRIVATE	
Status	OK	
Ethernet MAC	42:53:██████████	
Virtual NIC Device	ethernet_32784	
Virtual NIC MTU	2800	10.147.20.0/24 via (lan)
Ethernet Broadcast	enabled	
Ethernet Bridging	prohibited	
DNS Domain	(not configured)	
DNS Servers	(none)	
Allow Managed IPs	<input checked="" type="checkbox"/>	
Allow Global Internet IPs	<input type="checkbox"/>	
Allow Default Route Override	<input type="checkbox"/>	
All... DNS Configuration	<input type="checkbox"/>	

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

The screenshot shows a terminal window with two tabs. Tab 1 shows a serial connection to an FTDI FTB65PL3. Tab 2 shows a root shell on the host "root@10.10.10:822". The terminal output shows the following commands and progress:

```
root@rockrobo:/mnt/data/z5206677# curl-armv7 https://download.zerotier.com/RELEASES/1.8.1/dist/debian/trusty/zerotier-one_1.8.1_armhf.deb -o zerotier-one_1.8.1_armhf.deb
% Total    % Received  % Xferd  Average Speed   Time   Time  Current
          0     0      0     0      0      0      0      0
100 775k  100 775k  0     0      1282k  0  --:--:-- 0:00:00 1283k
root@rockrobo:/mnt/data/z5206677# dpkg -i zerotier-one_1.8.1_armhf.deb
Selecting previously unselected package zerotier-one.
(Reading database ... 14969 files and directories currently installed.)
Preparing to unpack zerotier-one_1.8.1_armhf.deb ...
Unpacking zerotier-one (1.8.1) ...
Setting up zerotier-one (1.8.1) ...
zerotier-one start/running, process 11052
Processing triggers for ureadahead (0.100.0-16) ...
root@rockrobo:/mnt/data/z5206677# zerotier-cli join abfd31bd47e18e42
200 join OK
root@rockrobo:/mnt/data/z5206677# zerotier-cli info
200 info 9ae0ff5bec 1.8.1 ONLINE
root@rockrobo:/mnt/data/z5206677# ifconfig
lo      Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:231 errors:0 dropped:0 overruns:0 frame:0
            TX packets:231 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:20829 (20.8 kB)  TX bytes:20829 (20.8 kB)
wlan0   Link encap:Ethernet HWaddr 64:90:c1:1d:24:c6
          inet addr:10.10.10.8  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::6690:c1ff:fe1d:24c6/64 Scope:Link
            UP BROADCAST RUNNING MTU:1500 Metric:1
            RX packets:10464 errors:0 dropped:0 overruns:0 frame:0
            TX packets:8603 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:4968643 (4.9 MB)  TX bytes:3058663 (3.0 MB)
```

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 b1 r0, #0x1533e. Below the assembly, a complex control flow graph (CFG) is displayed, showing various branches and loops between different sections of the code.

Right Window: Shows a specific function: sub_1b2d4(int32_t arg1, int32_t arg2, int32_t arg3). The assembly code includes fopen(arg1, 0x24ef4, arg3, __stack_chk_guard), if (r0 == 0), and a loop starting at 6 @ 0001b318. The CFG below shows the flow of control through this function, including calls to fgets and zx.d(var_41c).

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 got locked out. hahah....
 - Compare file changes
- Factory reset device, check for remnant files

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...

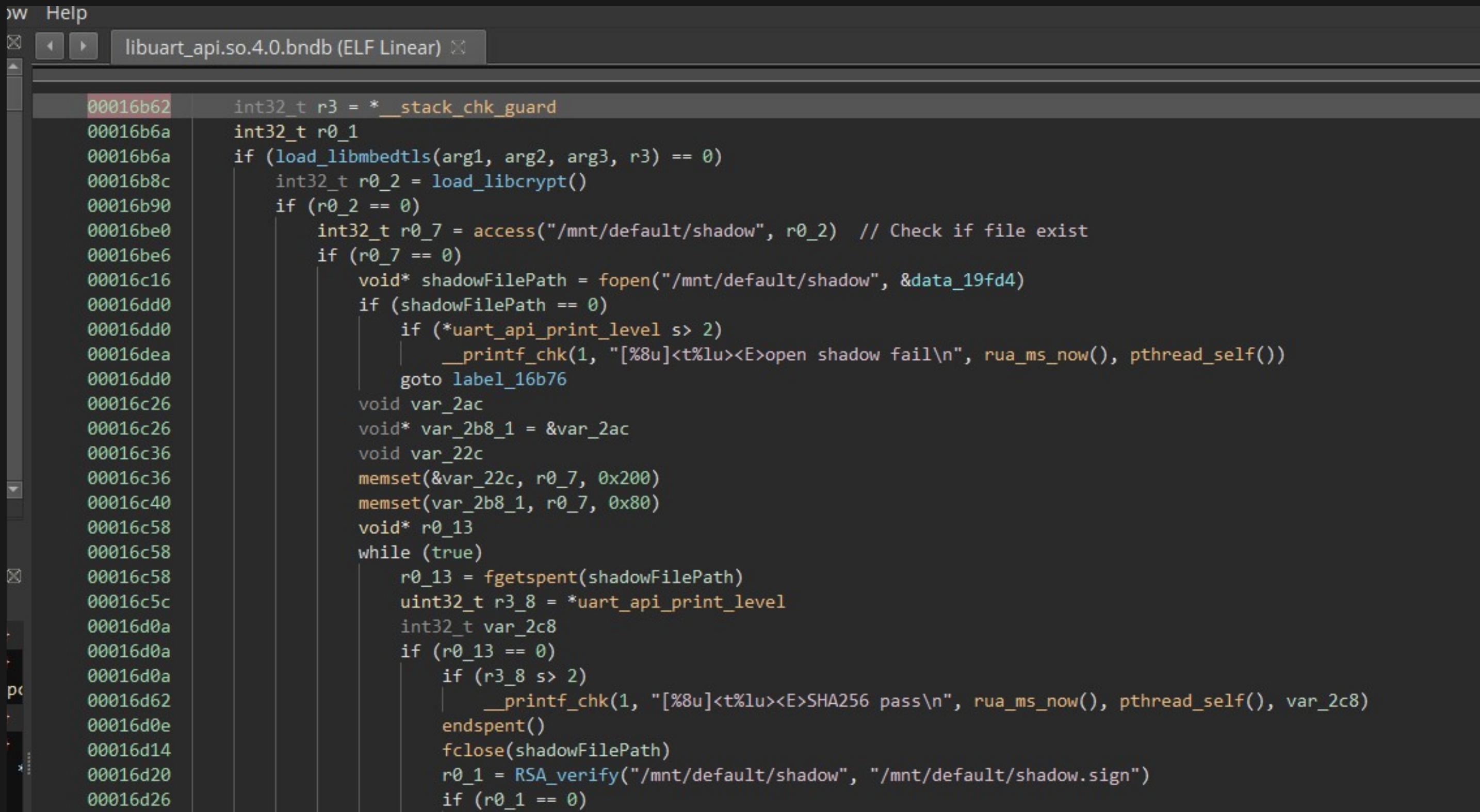
Affected: `rr_login` (serial), `dropbear` (SSH), `adbd` (USB)

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  
> █
```

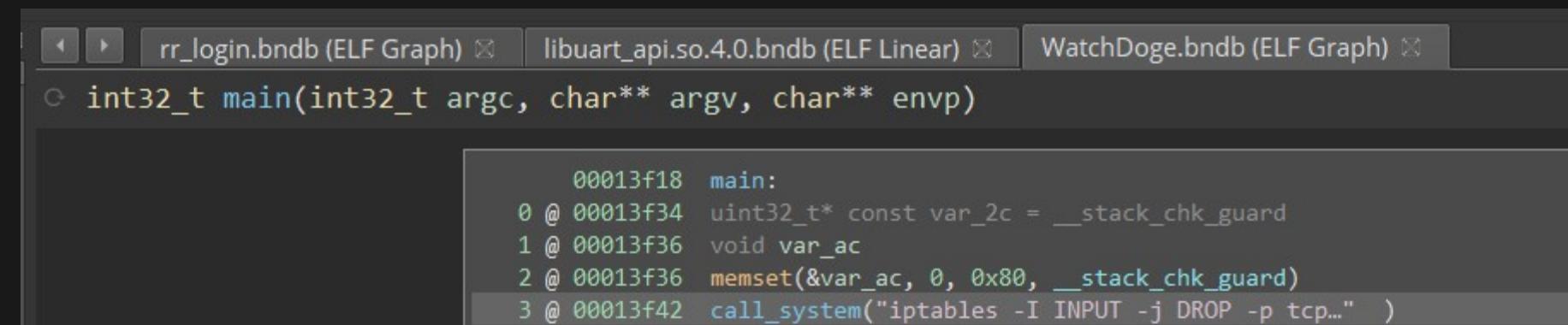
- Effectively now running embedded Linux

[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

File Persistence (Upgrade and Reset)

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

Reset Persistent

- [mmcblk0p11] /mnt/reserve

Upgrade Persistent

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

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

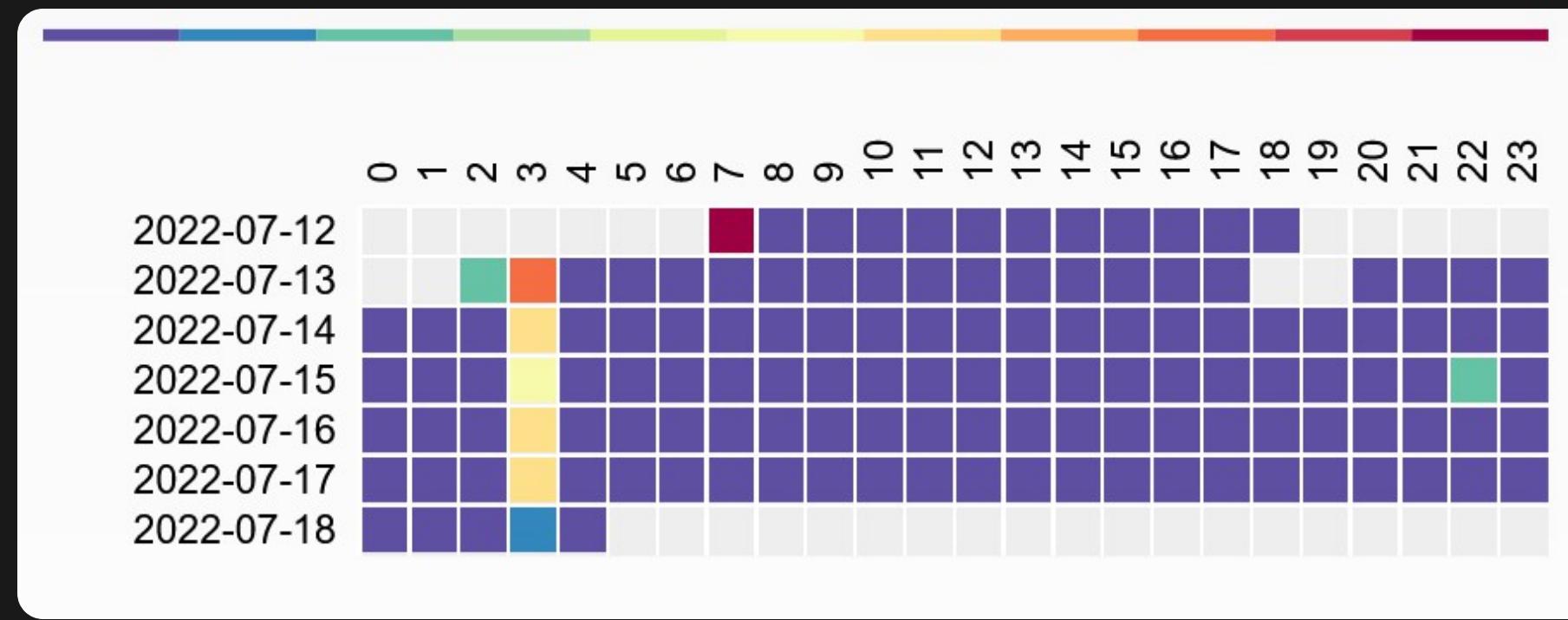
Network Inspection

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

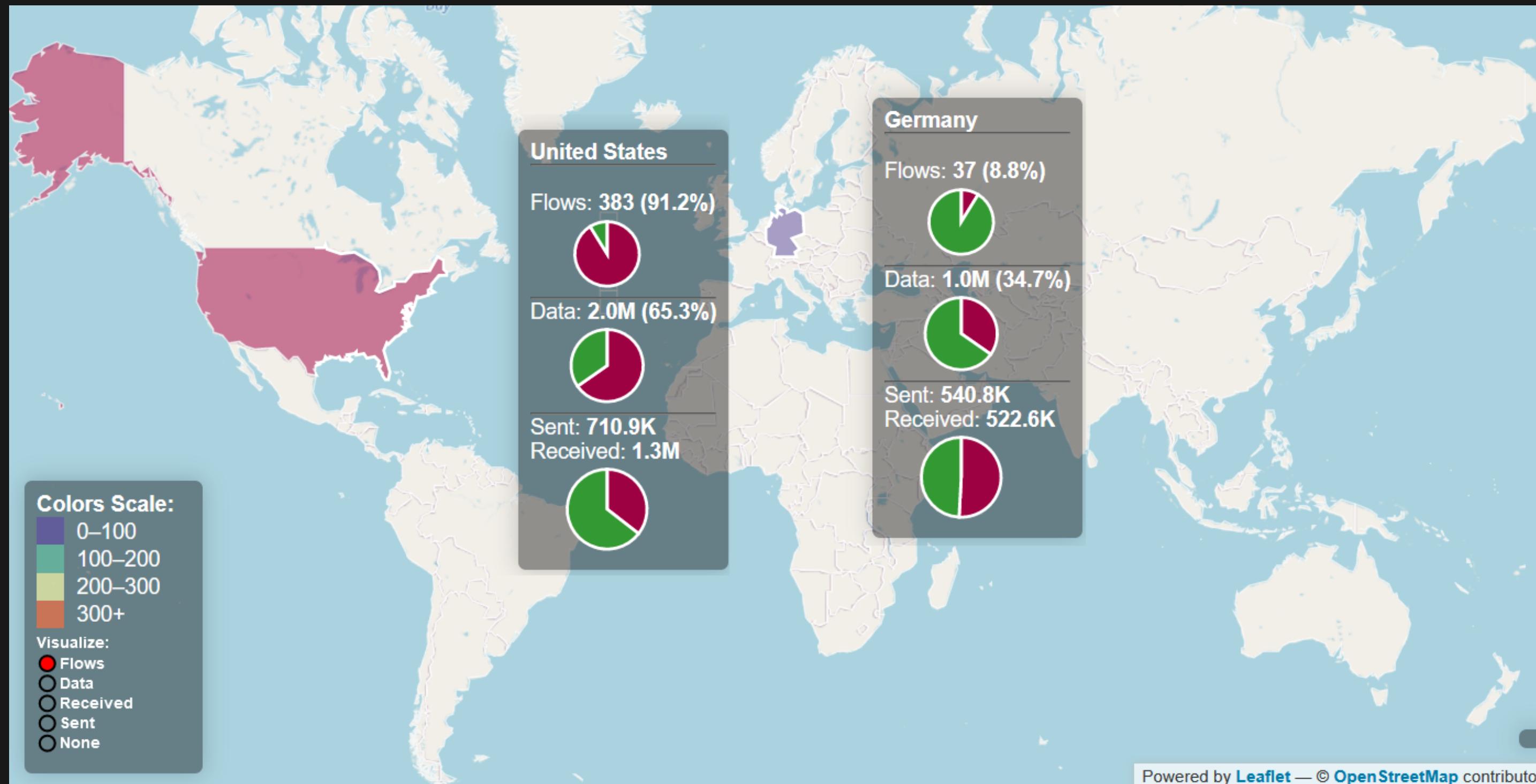
Network Behaviour (FW v02.29.02) (exc FDS) (1 week)



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  
  }]  
}
```

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

Network Inspection

Network Map

Let's Talk Threats

Talk about threat models

-
-
-
-
-

Reset Persistence

Patch the recovery partition `mmcblk0p7`

Upgrade Persistence

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

Remote Access

ZeroTier, etc

OTA Rooting

```
{  
    "method": "milo.ota",
```

Reset Persistence

Patch the recovery partition `mmcblk0p7`

Upgrade Persistence

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

Remote Access

ZeroTier, etc

OTA Rooting

Security Response

<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 paper

- UNSW Elec - MUD - they could do this to heighten security
 - note - doesn't prevent them from being malicious from within their own C2 server, but will mitigate modified devices

Takeaway Summary (answer main point)

tldr Who What When Why Where How

Address the statement

Thesis in a Year

The screenshot shows 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

e: andrew.j.wong@student.unsw.edu.au