Glitched on Earth by Humans: A Black-Box Security Evaluation of the SpaceX Starlink User Terminal

Lennert Wouters
@LennertWo
This talk

Source: u/darkpenguin22

Space
Earth

User Terminal (UT)

Gateway

Internet

Satellite

Source: SpaceX

Lower Earth Orbit (LEO)

Laser link
Hardware revisions

Circular UT
- 59 cm (23.23"") diameter
- Residential
- rev1_pre_production
- rev1_production
- rev1_proto1/2/3
- rev2_proto0/1/3
  - rev2_proto2 (SoC cut 3)
  - rev2_proto4 (SoC cut 4)

Square UT
- 50 x 30 cm (19” x 12”)
- Residential and RV
- rev3_proto0
- rev3_proto1
- rev3_proto2

High Performance UT
- 57 x 51 cm (22” x 20”)
- Business and Maritime
- hp1_proto0
- hp1_proto1

Transceiver
- External phased array
- transceiver_rev2p0/5

This talk (but attack should apply to all UT hardware)
Accessible connectors on V2*

*V1 hardware had an extra connector, V3 does not have easily accessible connectors
UART – U-Boot

U-Boot 2020.04-gddb7af8 (Apr 16 2021 - 21:10:45 +0000)

Model: Catson
DRAM: 1004 MiB
MMC: Fast boot:eMMC: 8xbit - div2
stm-sdhci0: 0
In: nulldev
Out: serial
Err: serial
CPU ID: 0x00020100 0x87082425 0xb9ca4b91
Detected Board rev: #rev2_proto2
sdhci_set_clock: Timeout to wait cmd & data inhibit
FIP1: 3 FIP2: 3
BOOT SLOT B
Net: Net Initialization Skipped
No ethernet found.

(Newer firmware no longer uses this version)

U-Boot does not accept serial input
(on non-development/fused hardware)
Development login enabled: no

SpaceX User Terminal.
user1 login:
PCB overview

GPS receiver

59 cm (23.23”)

Clock generation

STM STA8089

POE

SoC

clock

GPS

GLLBLU
RF Components

- (A) Digital BeamFormer (DBF)
  - STM GLLBSUABBBA
  - Codename: SHIRAZ

- (B) Front-End Module (FEM)
  - Codename: PULSAR(AD)

- V2 hardware and up:
  - 1 DBF → 16 FEMs
Siliconpr0n

Thanks to John McMaster!
@johndmcmaster
• (A) System-on-Chip
  • Custom quad-core ARM Cortex-A53
  • ST Microelectronics
    • GLLCCOCA6BF (cut 3?)
    • GLLCCODA6BF (cut 4?)
  • Codename: CATSON

• (B) Secure Element
  • STM STSAFE-A110

• (C) 4GB eMMC

• (D) 2 x 4Gbit DDR3
SoC

- through substrate image
  - GLLCCOCA6BF (cut 3?)
  - Thorlabs NIR camera
  - Mitutoyo NIR objective 50x

- Can help narrow down interesting locations for some physical attacks

- Full resolution version will be available on siliconpr0n.org!
Identifying eMMC test points

*Identifying eMMC test points*
Reading eMMC in-circuit

SD card reader

TXS0202EVM

Level shifter

What I did

What I recommend

Low Voltage eMMC Adapter

by

exploitee.rs
Extracting the eMMC dump

- Split the dump into:
  - TF-A Bootstages: Firmware Image Packages
    - unpack with TF-A fiptool
  - Flattened uImage Tree (FIT)
    - unpack with U-Boot dumpimage
  - SpaceX Runtime (dm-verity, error correcting codes)
  - SpaceX Calibration (dm-verity)
  - SpaceX EDR (LUKS)
  - SpaceX dish config (LUKS)

- More details:
# This file describes the limits for thermal control.
# All temperatures are in degrees Celsius.
# All control cycle counts are for 50 Hz.

# --------- Power-cut ---------

# When any sensor exceeds these threshold for its corresponding
# persistence, the power to all DBFs and FEMSs will be cut. The User Terminal
# must reboot to recover. These temperatures are slightly above the maximum
# junction temperature of the corresponding components. MAC throttle and forced
# idle is intended to more-gracefully take care of all overtemp situations.
# This FDIR is a last-ditch response to reduce in case idling is insufficient
# or we have lost control of the beamformers.

center_power_cut.t_trip 90.0
cpu8_power_cut.t_trip 128.8
pa_power_cut.t_trip 118.0
dbf_power_cut.t_trip 118.0

# The number of cycles that the trip thresholds must be exceeded for before
# the power-cut FDIR activates.

center_power_cut.persistence_limit 2000 # 40 seconds
cpu8_power_cut.persistence_limit 2000 # 40 seconds
pa_power_cut.persistence_limit 2000 # 40 seconds
dbf_power_cut.persistence_limit 2000 # 40 seconds

# The number of cycles from when power-cut is tripped to when the UT reboots.
# Gives time to allow the UT to cool down.
power_cut_reboot_delay 30000 # 10 minutes

# --------- Forced-idle ---------

# When any sensor exceeds these threshold for its corresponding
# persistence, all DBFs and FEMSs will be commanded to Idle mode.
# Once all sensors have fallen below their clear thresholds, normal

"channel_id": 13,
"direction": "uplink",
"end": 14.1875,
"start": 14.125
},

"channel_id": 14,
"direction": "uplink",
"end": 14.25,
"start": 14.1875
},

"laser_channel_definitions": [
{
"color": "LASER_COLOR_RED",
"frequency_ghz": 192700,
"itu_channel_id": 27
},

{ "color": "LASER_COLOR_BLUE",
"frequency_ghz": 193500,
"itu_channel_id": 35
}
Development geofences
Obtaining root

Development login enabled: no

SpaceX User Terminal.
user1 login:

```bash
obtaining_root

Development login enabled: "
if [ $(is_production_hardware) -eq 0 ]; then
  echo "yes"
  sed -i -e 's/\(root:[^:]*\)/root:tSXNnW65X1Er/ /etc/shadow 2>/dev/null || true
else
  echo "no"
  if [ $(whatVehicleAmI) = "uterm" ]; then
    # Discard console output for production user terminals.
    consoletype=ttynull
  fi
fi
```
Fault injection

✓ Flip-chip packaging exposes die backside
  • Laser Fault Injection, Body Bias Injection, Electromagnetic Fault Injection

✗ PCB is too big for our automatic XYZ positioning equipment
  • Likely cumbersome to do on a roof...

✗ No development kits

• Differential clock input
  • (But PLL?)
• Reset line
• Voltage Fault Injection
Crowbar VFI

- NewAE ChipWhisperer-Lite (~ $250)
  - Glitch port is connected to the SoC core voltage
  - Momentarily shorts core voltage to GND
- Core voltage: ~1V, generated by TI TPS56C230
- All decoupling capacitors untouched at this point!
- Oscilloscope triggers on serial data
  - Trigger output is input to the ChipWhisperer-Lite
- Glitch parameters controlled from Python
  - Offset from trigger point
  - Glitch width
Development login enabled: yes

7.387682] 002: Unable to handle kernel NULL pointer dereference at virtual address 0000000000000000
7.387702] 002: Mem abort info:
sh: 0: unknown operand
7.387704] 002: ESR = 0x96000006
7.387707] 002: EC = 0x25: DABT (current EL), IL = 32 bits
7.387711] 002: SET = 0, FnV = 0
7.387714] 002: EA = 0, S1PTW = 0
7.387716] 002: Data abort info:
7.387718] 002: ISV = 0, ISS = 0x00000006
7.387721] 002: CM = 0, WnR = 0
7.387723] 002: user pgtbale: 4k, pages, 39-bit VAs, pgdp=00000000a51fd000
7.387730] 002: [0000000000000820] pgd=00000000a50d1003, pud=00000000a50d1003, pmd=0000000000000000
7.387748] 002: Modules linked in:
7.387753] 002: CPU: 2 PID: 275 Comm: syslogd Not tainted 5.4.34-rt21-gfd24730 #1
7.387760] 002: Hardware name: spacex satellite user terminal (DT)
7.387766] 002: pstate: 00000005 (nzcv daif -PAN -UAO)
7.387770] 002: pc : do_undefinedstr+0x2c/0x1d8
7.387787] 002: lr : el0 undefined+0xc/0x10
7.387793] 002: sp : ffffffff0145b3e70
7.387797] 002: x29: ffffffff0145b3e70 x28: ffffffff8025009a00
7.387803] 002: x27: 0000000000000000 x26: 0000000000000000
7.387808] 002: x25: 0000000000200000 x24: 0000000000000000
7.387814] 002: x23: 0000000000000000 x22: 0000000000403feb
7.387818] 002: x21: 00000000fffffff x20: 0000000000000000
7.387823] 002: x19: 000000000000000018 x18: 0000000000000000
7.387828] 002: x17: 0000000000000000 x16: 0000000000000000
7.387832] 002: x15: 0000000000000000 x14: 0000000000000000
Results

✓ The Proof-of-Concept works
  ✓ Was reproduced by the SpaceX PSIRT
✓ Easy to produce (undesirable) faults
  ✓ A fully booted SoC is already being pushed to its limits

x Slow: 1 attempt every 12 seconds (one per boot)
  x Low success rate: many hours for one good attempt
x Unreliable: successful glitch often also results in other errors
1. BL1 loads BL2 certificate from eMMC
2. BL1 verifies the certificate’s signature
3. BL1 loads the BL2 firmware from eMMC
4. BL1 verifies that SHA512(BL2) matches the hash contained in the certificate
BL1 Glitch setup

- Try to boot with (in)valid signature, hash and firmware
- Try to glitch a valid certificate into a signature verification failure
Normal boot

“INFO: Image id=6 loaded at address 0x30209000, size = 0x90”
→ Certificate has been loaded

UART

EM side-channel

Signature verification

x 10e6 samples
Glitched boot

“INFO: cert_nv_ctr : 1”
→ Signature verified and the rollback counter is 1

Signature verification skipped?!
**ROM Bootloader (BL1)**

- Mapped at 0x30000000 and readable from BL2!
  - BSEC eFuses mapped at 0x22400000 (shadow registers)

- Emulated the ROM bootloader using Unicorn Engine
  - Fuzzed using AFL++ in Unicorn mode

- Simulated instruction skip faults in Unicorn Engine
  - Single instruction skip faults do not result in the observed behavior!
    - Code has some control flow checks and redundant operations
  - Skipping two consecutive instructions does result in the observed behavior
    - (Actual fault model is likely to be different)
**BL1 glitch detection example**

**BL1 UART output**

INFO: BL1: Get the image descriptor
INFO: BL1: Loading BL2
INFO: Loading image id=6 at address 0x30209000
INFO: Skip reserving region [base = 0x30209000, size = 0x90]
INFO: Image id=6 loaded at address 0x30209000, size = 0x90

INFO: cert_nv_ctr : 1
INFO: plat_nv_ctr : 0
INFO: Loading image id=1 at address 0x30209000
INFO: Image id=1 loaded at address 0x30209000, size = 0xf178

NOTICE: BL1: Booting BL2
NOTICE: plat_error_handler err = -80
INFO: Authentication error !!!

Certificate has been loaded
Contains invalid signature but valid digest of BL2 firmware

Signature verification succeeded!
Loaded BL2 firmware and verified hash digest

Final control flow check detects our glitch! 😞
BL1 glitch detection example

```c
void BL1_main(void)
{
    int iVar1;
    int iVar2;
    long iVar3;
    char *format;
    undefined iVar4;
    long unaff_x21;
    long unaff_x22;
    DAT_32020080_hash_check_val = 0x5a7cbe01;
    DAT_32020004_signature_check_val = 0x5a7cbe01;
    printf(s_NOTICE__BL1__%s_3000clb8_s_v1_3(release):f26689a_3000b050);
    printf(s_NOTICE__BL1__%s_3000clb8_s_Built__i13:04:48__Jul_30_2018_3000b070);
}
```

Called right before passing control to BL2

```c
void final_error_checking(void)
{
    if ((DAT_32020080_hash_check_val == L'\xb134f725') &&
        (DAT_32020004_signature_check_val == L'\xb134f725'))
        goto LAB_3000094c;
    do {
        plat_error_handler(0xffffffffb0);
    } while (DAT_32048e4 < DAT_2240000c);
    return;
}
```

```c
bool verify_signature(long param_1, int param_2, long param_3, int param_4, long *param_5, int param_6,
                      undefined8 param_7)
{
    int iVar1;
    if (((param_2 == 0x50) && (param_3 + 0x40 == param_1)) &&
        ((param_4 == 0x48 && param_6 == 8) && param_5 == 0x39313535324445)) {
        DAT_320200084_signature_check_val = 0x5a7cbe01;
        iVar1 = ed25519_verify(param_1, 0x50, param_3, param_7);
        if (iVar1 == 1) {
            /* Signature check succeeded */
            DAT_320200084_signature_check_val = 0xb134f725;
            return iVar1 == 0;
        }
        return true;
    }
```
- Decoupling capacitors are needed for later boot stages
- Experimented with:
  - N-channel MOSFETS
  - P-channel MOSFETS
  - High/Low side switching
  - Gate voltage
  - MOSFET drivers
  - Capacitor sizes
  - Timing
Researcher access

• Demonstrated a full attack in the lab!
  • But the setup is still too bulky to be used in a practical setting (e.g., on the roof)
• SpaceX offered an easy way out: SSH access through a Yubikey
  • But I was already too far down the rabbit hole …

```bash
15 vehicle=$(whatVehicleAmI)
16 rev=$(whatRevAmI)
17 nodetype=$(whatNodeTypeAmI)
18
19 if [ "$vehicle" = "uterm" ] && [ "$rev" != "0" ]; then
20   # Create static AuthorizedPrincipalsFile for UTs and Transceivers only.
21   catson_uuid="$(printf "%08x-%08x-%08x\n" \\
22     $(cat /sys/bus/platform/devices/*/catson_fuses/devid[012])))"
23
24   # Maintain compatibility with transceiver certificate format.
25   principal=$vehicle
26   if [ "$whatVehicleVariantAmI" = "starlink_transceiver" ]; then
27     principal="transceiver"
28   fi
29   echo "spacex:$principal:researcher:$catson_uuid" > /etc/ssh/authorized_principals
```
Creating a mobile setup

- Replacing lab equipment with low-cost off-the-shelf components
- RPI Pico replaces oscilloscope and ChipWhisperer
- Works
  - But still messy…
PCB design

- Scanner @ 600 DPI
- Draw board outline at real size in Inkscape
  - Load in KiCad and use in the edgecuts layer
Modchip

Castellated holes to mount to the UT PCB

RP2040 @250MHz
PIO for triggering and glitch generation

Glitch/crowbar MOSFET

Decoupling MOSFETs

2 channel MOSFET driver

Available on GitHub!
Installed modchip

- Core voltage regulator enable pin (for power cycling)
- 12V for MOSFET drivers and standalone power
- 1V8 for level shifter
SpaceX strikes back

• I did a firmware update…
• Previously unused eFuse is now blown and disables UART output
• Modchip was designed to trigger on UART

```c
if (L'\x00000000' < BSEC_UART_EN) {
    DAT_30204160_UART_EN = L'\xde486bc3';
}
if (DAT_30204160_UART_EN == L'\xde486bc3') {
    _GLCFF_SYSCFG_PIO_A_BASE = _GLCFF_SYSCFG_PIO_A_BASE & 0xff;
    DataSynchronizationBarrier(3,3);
    _GLCFF_SYSCFG_PIO_A_BASE_A0 = _GLCFF_SYSCFG_PIO_A_BASE_A0 & 0xff;
    DataSynchronizationBarrier(3,3);
    uVar1 = 10000000;
    if ((_BOOTMODE_REGISTER_09130048 & 1) != 0) {
        uVar1 = 200000000;
    }
    set_uart_baud(0UART_BAUDRATE,uVar1,115200);
    printf(s_INFO:_AUTOSTARTUP_MODE = %d_3000b08e,(ulong)(_BOOTMODE_REGISTER_09130048 & 1));
}
Adapt

BEFORE

AFTER
Overcome

- Trigger on eMMC D0 instead of UART
- Modchip could be easily adapted
  - Disconnect UT UART TX
  - Connect to eMMC D0
  - Update glitch parameters from Python
- Alternative: new PCB revision
Network exploration

- All interesting communication uses mutually authenticated TLS (STSAFE)
- Added STSAFE support to the tlslite-ng TLS implementation
- Python script to download the latest firmware updates
- Mostly IPv6 2620:134:b000::1:0:0
- Open ports (nmap): 8001-8012, 9000, 9003, 9005, 9010, 9011

Firmware update archive
What’s next?

• You can make your own modchip and use it to:
  • Further explore the network infrastructure
    • Not accessible as a normal user
    • Integrate the STSAFE with GRPC
  • Interact with the Digital BeamFormers and update their firmware
  • Repurpose your terminal?
Conclusion

• We can bypass secure boot using voltage fault injection in BL1
  • Quad core Cortex-A53 in a black box scenario
    • no documentation, no open development kits
  • Enabling and disabling of decoupling capacitors
  • Fault injection countermeasures are only as good as the fault model that was used

• This is a well-designed product (from a security standpoint)
  • No obvious (to me) low-hanging fruit
  • In contrast to many other devices getting a root shell was challenging
  • And a root shell does not immediately lead to an attack that scales

• SpaceX PSIRT was very responsive and helpful!
  • https://bugcrowd.com/spacex vulnerabilityreporting@spacex.com
INFO - Established serial connection
INFO - Connected to modchip
Press enter to start.
INFO - Attempting to bypass secure boot...
585 [00:58, 1.00s/it]
INFO - Glitch successful!

> python3.10 utglitcher.py

U-Boot 2021.04-g84e5f81 (Feb 05 2022 - 01:17:09 +0000)
Model: Catson
DRAM: 1004 MiB
MMC: Fast boot:eMMC: 8xbit - div2
stm-sdhci0: 0
In: serial
Out: serial
Err: serial
CPU ID: 0x90020a01 0x868dc3eb 0x8332b785
sdhci_set_clock: Timeout to wait cmd & data inhibit
No SXID found
Detected Board rev: #rev2_proto4
FPI1: 3 FIP2: 3
BOOT SLOT B
Net: Net Initialization Skipped
No ethernet found.
Thanks!

- Arthur Beckers
- Gert Van Beneden
- Tim Ferrell
- John McMaster
- Dan Murray
- Colin O’Flynn