SYNACKTIV

Will it run Doom?

What it takes to get arbitrary code execution on a DVB-S receiver

ESNA 2024

Whoami

SYNACKTIV

- Baptiste MOINE (**@Creased_**)
- **Security researcher** at Synacktiv (VR/RE)
- Company specialized in offensive security: **penetration testing**, **reverse engineering**, software development, trainings, etc.
- Around **170 experts** over 5 offices in France (Lille, Paris, Rennes, Toulouse and Lyon)
- **We are recruiting!**

How it started

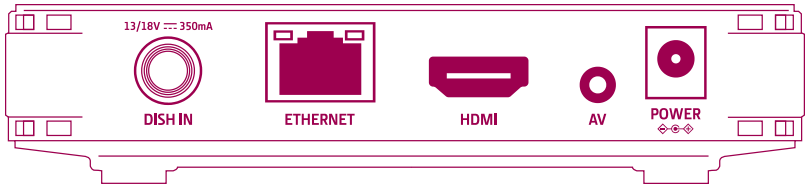
SYNACKTIV

- WEEE / e-waste: Waste from Electrical and Electronic Equipment
- Many functional electronic devices are thrown away
- It's sometimes possible to retrieve some of them by asking nicely :)

Product description

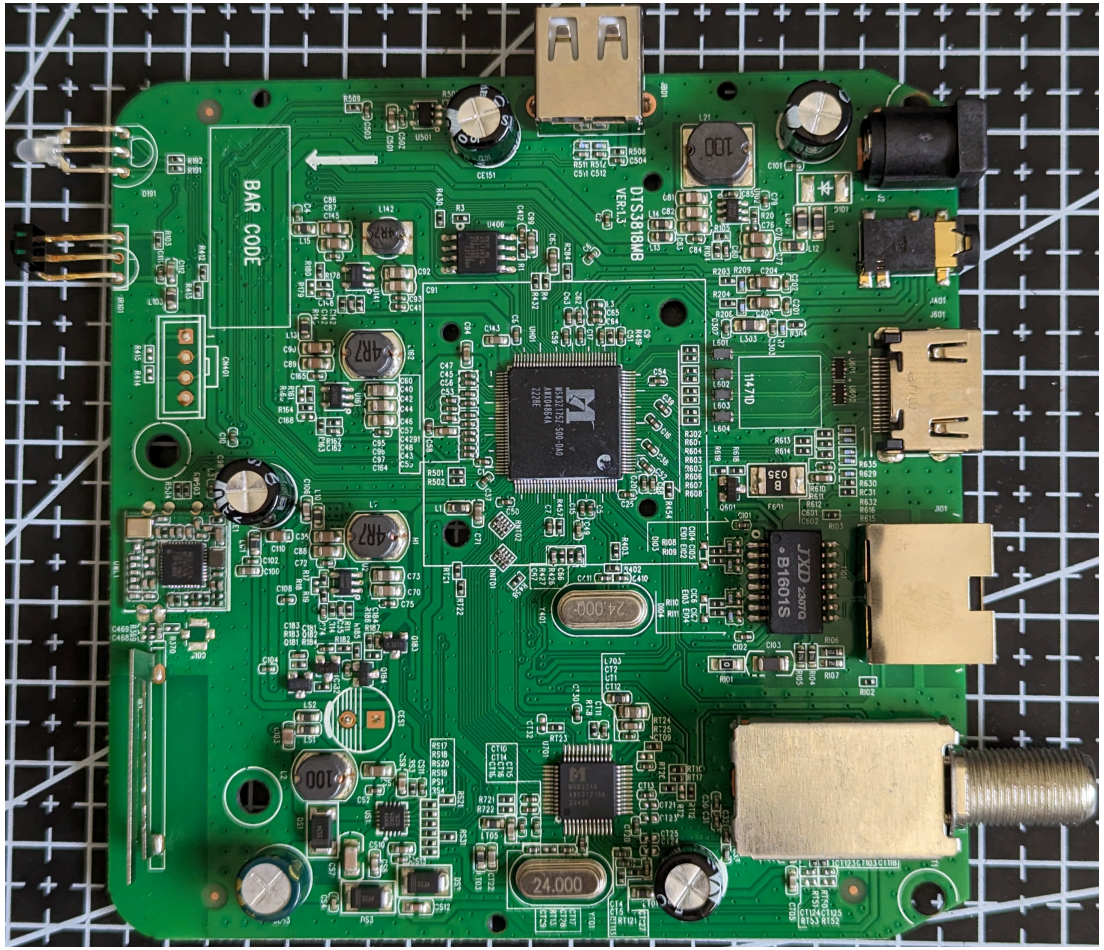
SYNACKTIV

- Manhattan SX Freesat HD Box
- Over 170 channels
- Subscription-free TV
- Low power (under 0.5W in standby)
- 802.3 10BASE-T / 100BASE-TX
- 802.11 b/g/n
- HDMI maximum 1080p
- USB 2.0



Product description

- JXD B1601S network transformer
- MediaTek MT7601U 802.11n Wi-Fi controller
- Macronix MX25L12833F 128Mbit NOR flash
- MStar MSB3245 DVB-S2/S demodulator
- MStar MSA3Z175Z-S00-DA0 SoC



MStar SoC

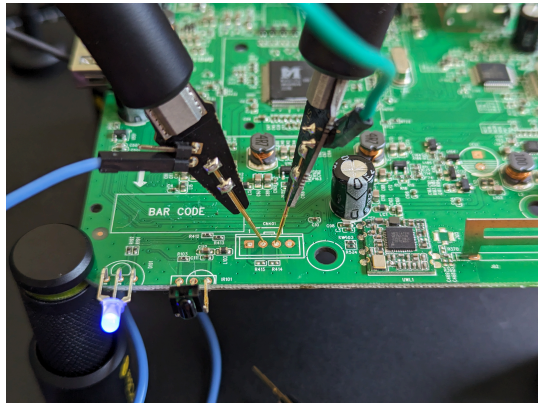
SYNACKTIV

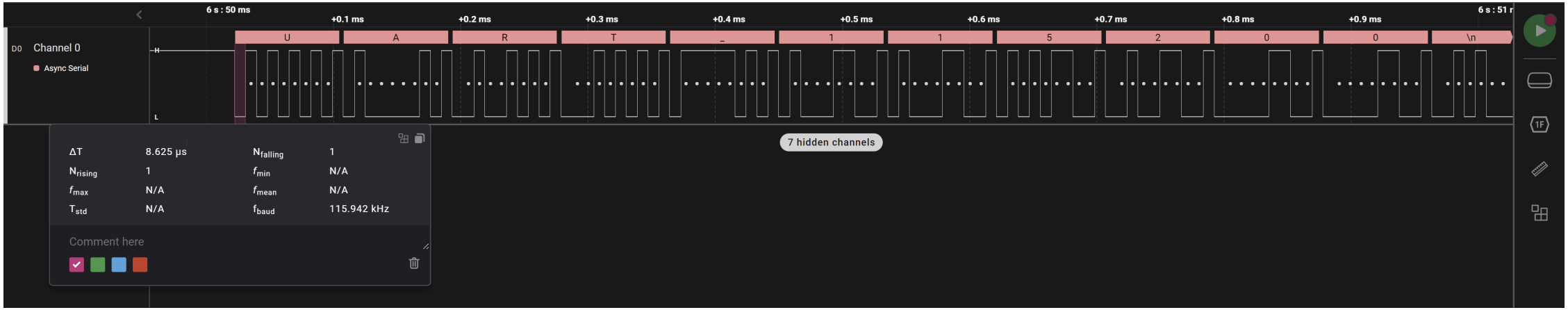
- Used in many DVB-T and DVB-S receivers
- No datasheet
- No pinout
- No knowledge
- Nothing.

UART

SYNACKTIV

- Ground probing
- Voltage probing: 3v3 pins
- Logic analyzer





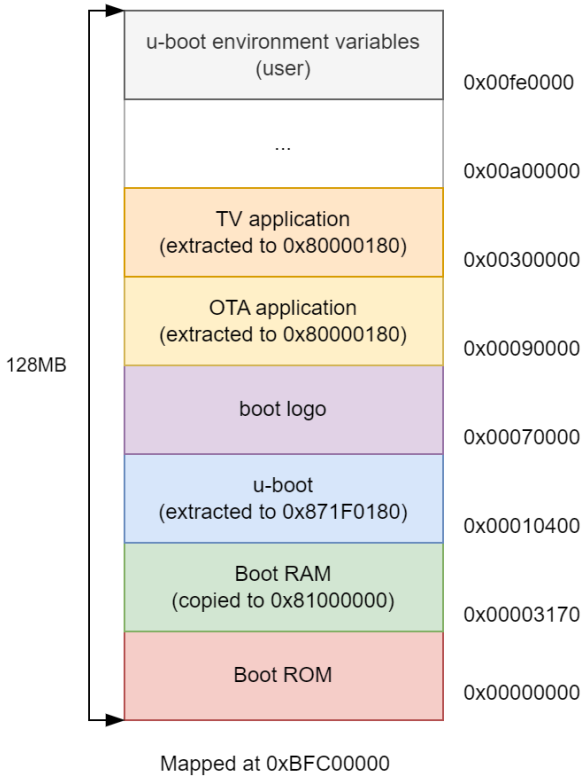
u-boot

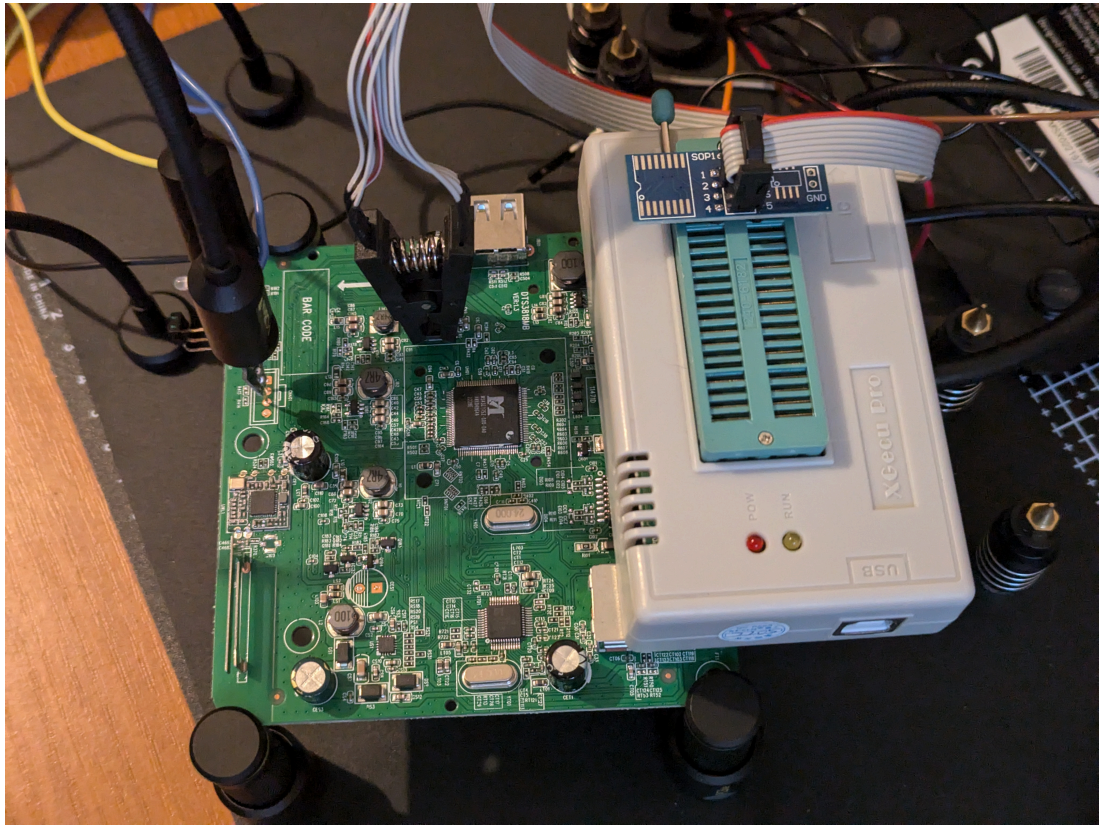
SYNACKTIV

- The autoboot interruption is enabled
- RAM: 0x80000000 - 0x90000000
- U-boot base address: 0x871F0180
- The APP is compressed and has a CRC.
- APP base address: 0x80000224

```
$ picocom -b 115200 /dev/ttyUSB0

U-Boot 2011.06-svn565 (Mar 01 2018 - 21:27:50)  MBOT-1106-0.8.KANO_TEE_NAND.a1

[...]

uboot held at [8F000000~90000000]
Now running in RAM - U-Boot at: 871F0180

[...]

Hit any key to stop autoboot:  0

Jumping to Application...
APP CRC Success...
Decompression OK!
Starting application at 0x80000224 ...


*RESET*

[...]

kiwi# bdinfo

memstart    = 0x80000000
memsize     = 0x10000000
```

Flash dump





```
kiwi# usb reset 1
kiwi# fatwrite usb 0 0x80000000 RAM.BIN 0x10000000
kiwi# spi_rdc 0x80000000 0x0 0x1000000
kiwi# fatwrite usb 0 0x80000000 FLASH.BIN 0x1000000
```

Boot flow

SYNACKTIV

1. The BootROM is executed in place (XIP) from flash (0xBFC00000)
2. The UART, DRAM, clocks, cache, hardware registers, etc. are initialized
3. The BootRAM is loaded from flash to DRAM and executed (0x81000000)
4. U-boot is loaded from flash to DRAM, extracted and executed (0x871F0180)
5. The APP image is loaded from flash to DRAM, extracted and executed (0x80000224)

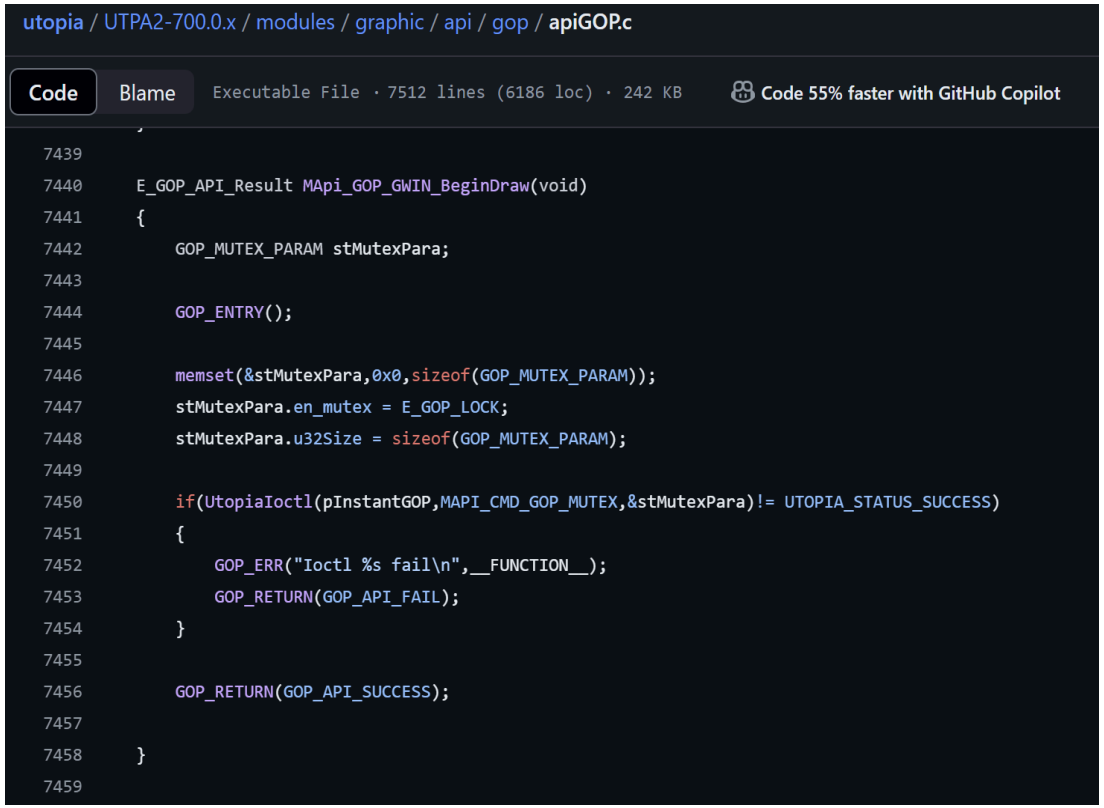
```
int jmp_to_app()
{
  printf("MsBoot.c E-%d>APP CRC Check..!!\n", 1174);
  if ( check_app_crc() )
  {
    printf("MsBoot.c E-d%>APP CRC Fail..!!\n", 1178);
  }
  else
  {
    spi_rdc(0x81100000, 0x300000, 0x700000);
    if ( mscompress7(0x81100000, 0x80000180, 0x700000) == 1 )
    {
      printf("MSBOOT.C %d-E> Decompression OK[Go]\n", 1196);
      run_command("go 0x80000224", 0);
      return 1;
    }
    printf("MSBOOT.C %d-E> Decompression NOK\n", 1215);
  }
  return 0;
}
```

```
int jmp_to_upgrader()
{
  printf("MsBoot.c I-%d>UPG CRC Check..!!\n", 1245);
  if ( check_app_crc() )
  {
    printf("MsBoot.c E-%d>UPG CRC Fail..!!\n", 1249);
  }
  else
  {
    spi_rdc(0x81100000, 0x90000, 0x250000);
    if ( mscompress7(0x81100000, 0x80000180, 0x250000) == 1 )
    {
      printf("MSBOOT.C %d-E> Decompression OK[Go]\n", 1267);
      run_command("go 0x80000224", 0);
      return 1;
    }
    printf("MSBOOT.C %d-E> Decompression NOK\n", 1286);
  }
  return 0;
}
```

Applications

SYNACKTIV

- Running on eCos kernel
- Uses the MStar API "MApi" **partially** leaked on GitHub
- We could probably just patch an existing app 🤷



utopia / UTPA2-700.0.x / modules / graphic / api / gop / **apiGOP.c**

Code | Blame | Executable File · 7512 lines (6186 loc) · 242 KB | Code 55% faster with GitHub Copilot

```
7439
7440    E_GOP_API_Result MApi_GOP_GWIN_BeginDraw(void)
7441    {
7442        GOP_MUTEX_PARAM stMutexPara;
7443
7444        GOP_ENTRY();
7445
7446        memset(&stMutexPara,0x0,sizeof(GOP_MUTEX_PARAM));
7447        stMutexPara.en_mutex = E_GOP_LOCK;
7448        stMutexPara.u32Size = sizeof(GOP_MUTEX_PARAM);
7449
7450        if(UtopiaIoctl(pInstantGOP,MAPI_CMD_GOP_MUTEX,&stMutexPara)!= UTOPIA_STATUS_SUCCESS)
7451        {
7452            GOP_ERR("Ioctl %s fail\n",__FUNCTION__);
7453            GOP_RETURN(GOP_API_FAIL);
7454        }
7455
7456        GOP_RETURN(GOP_API_SUCCESS);
7457
7458    }
7459
```

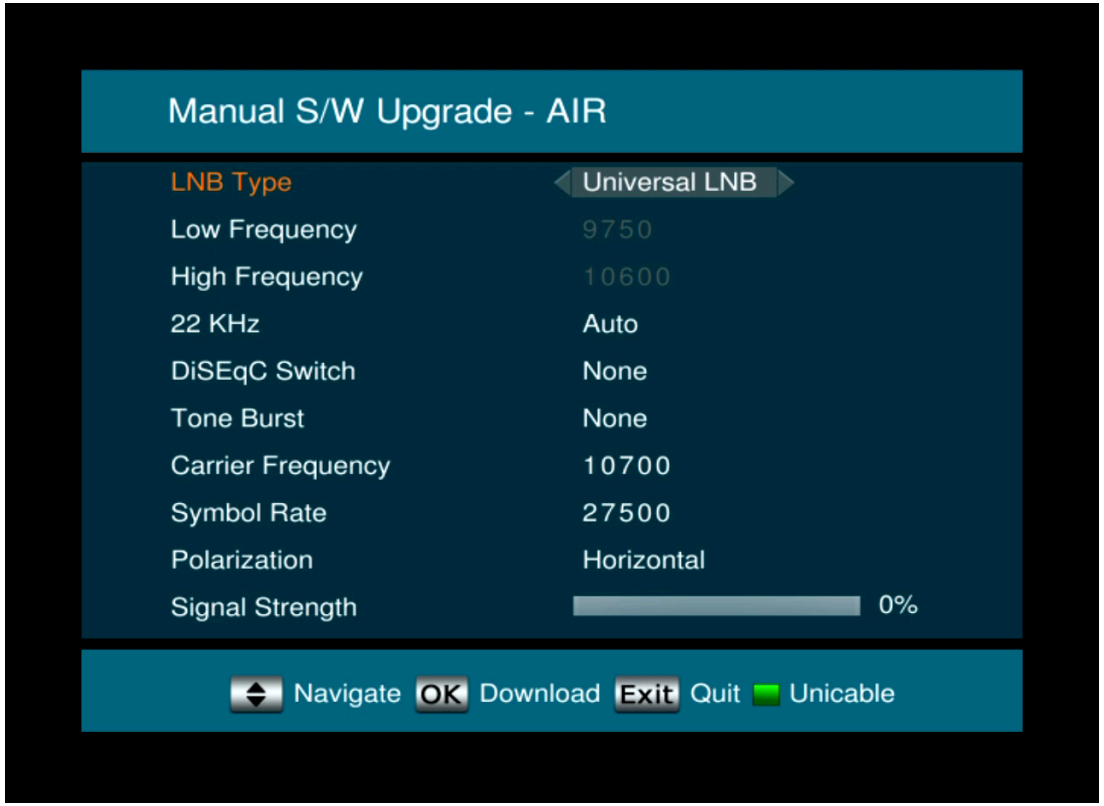
https://github.com/jockyw2001/utopia/

The upgrader

SYNACKTIV

- ~35 tasks for a minimal application (UPG)
- ~40K functions
- A LOT of broken xrefs…
- Starting the upgrader:

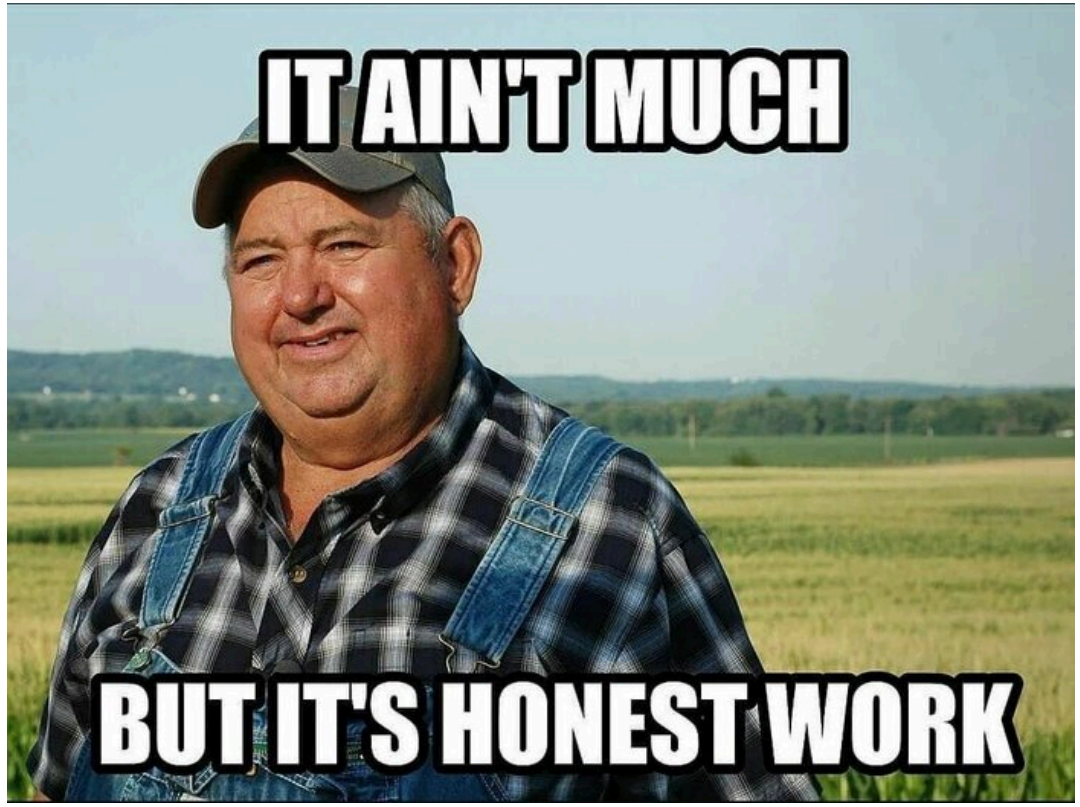
```
kiwi# spi_rdc 0x81100000 0x90000 0x250000
kiwi# mscompress7 d 0 0x81100000 0x250000 0x80000180
kiwi# go 0x80000224
```



Arbitrary code execution

SYNACKTIV

- Finding OSD-related functions and mappings
- Injecting some code in blank space
- Redirecting the main task after some initialization
- Trying to get a custom code executed
- Trying, over and over again…



Will it run Doom?

SYNACKTIV

- Not so soon…
- Needs a lot of work and effort to reverse engineer the hardware handling code
- Meanwhile, we can still run custom code on top of existing apps:

```
kiwi# usb reset 1
kiwi# fatload usb 0:1 0x80000180 SYNACKTIV.BIN
kiwi# go 0x80000224
```

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com