Journal

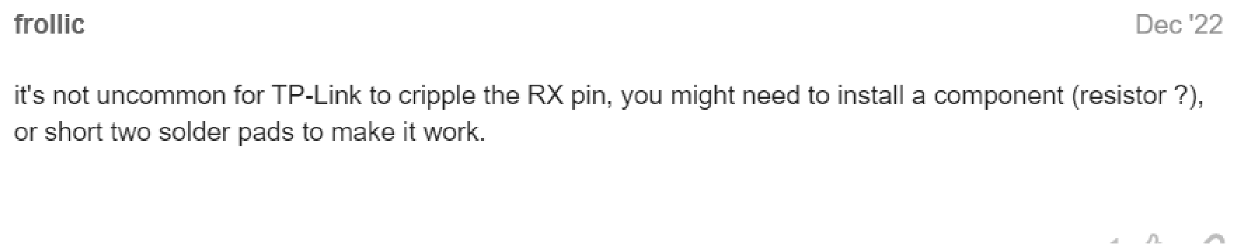
300M Wireless N Router
Model No. TL-WR845N

31.10.2023

Firstly, I opened up the router, not to lose time with the web interface I wanted to get the filesystem as fast as possible to have a larger surface to explore.

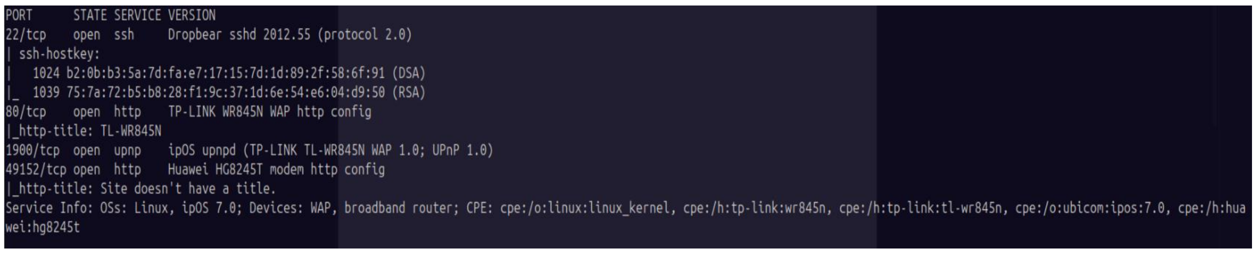
Opening the router revealed a PCB of course. After a bit of looking around I found what I was looking for, the UART port. Usually, this UART port lets you connect to the U-Boot shell or gain a shell straight as a user (admin) on the device.

It seemed easy as first, but after configuring my Bus Pirate with the baud rate of 115200 and default configuration for the rest of the options, data started coming so router's TX was working but I could not send any data, RX was somehow crippled ☹. After reading some posts online it seemed that especially TP Link likes to cripple serial interfaces on its routers.


frollic Dec '22

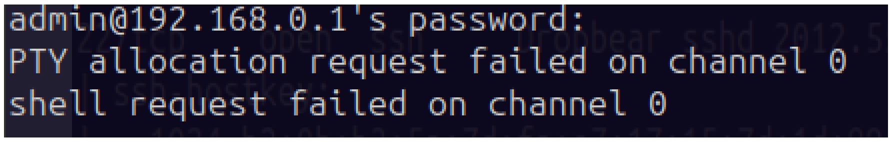
it's not uncommon for TP-Link to cripple the RX pin, you might need to install a component (resistor ?), or short two solder pads to make it work.

Ok so taking a break on hardware stuff would seem a great idea, a breadth first approach is better in initial information gathering and only after something looks promising a depth first approach is suitable. Let's then perform a port scan on the router.


```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     Dropbear sshd 2012.55 (protocol 2.0)
| ssh-hostkey:
|   1024 b2:0b:b3:5a:7d:fa:e7:17:15:7d:1d:89:2f:58:6f:91 (DSA)
|_  1039 75:7a:72:b5:b8:28:f1:9c:37:1d:6e:54:e6:04:d9:50 (RSA)
80/tcp    open  http    TP-LINK WR845N WAP http config
|_http-title: TL-WR845N
1900/tcp  open  upnp    ipOS upnpd (TP-LINK TL-WR845N WAP 1.0; UPnP 1.0)
49152/tcp open  http    Huawei HG8245T modem http config
|_http-title: Site doesn't have a title.
Service Info: OSs: Linux, ipOS 7.0; Devices: WAP, broadband router; CPE: cpe:/o:linux:linux_kernel, cpe:/h:tp-link:wr845n, cpe:/h:tp-link:tl-wr845n, cpe:/o:ubicom:ipos:7.0, cpe:/h:hua
wei:hg8245t
```

Of course first thing that I have tried was to SSH as root:root. Didn't work.

After searching a bit on google for default user and password found that it is admin:admin. Tried and….



```
admin@192.168.0.1's password:
PTY allocation request failed on channel 0
shell request failed on channel 0
```

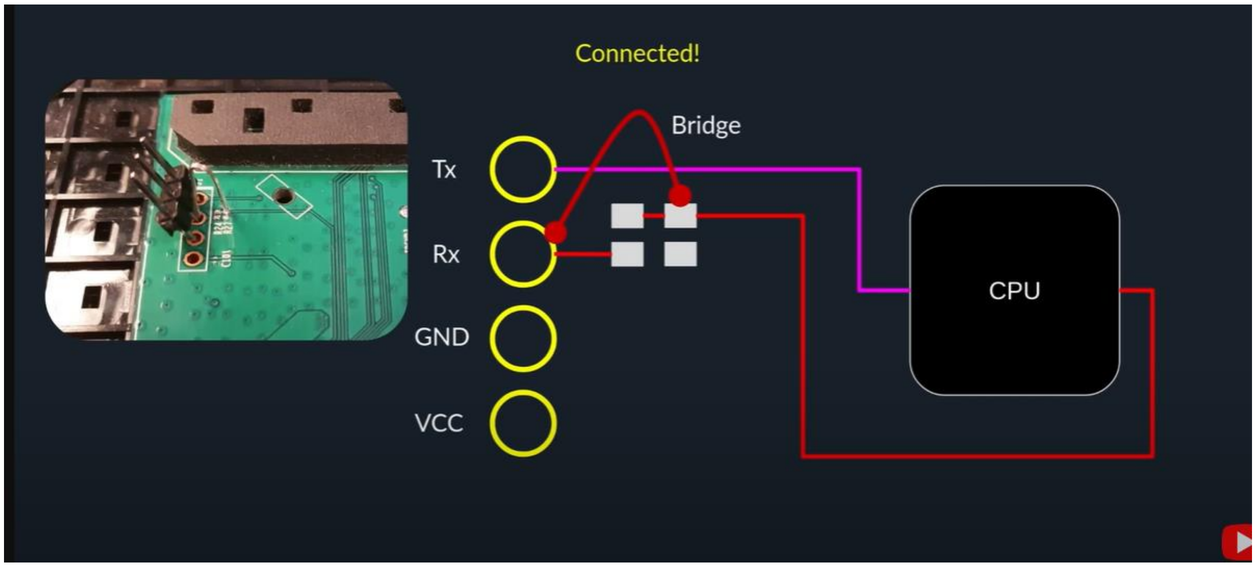
Did not succeeded to make it work, putty on windows returns the same so I guess it is something to do with a configuration.

01.11.2023

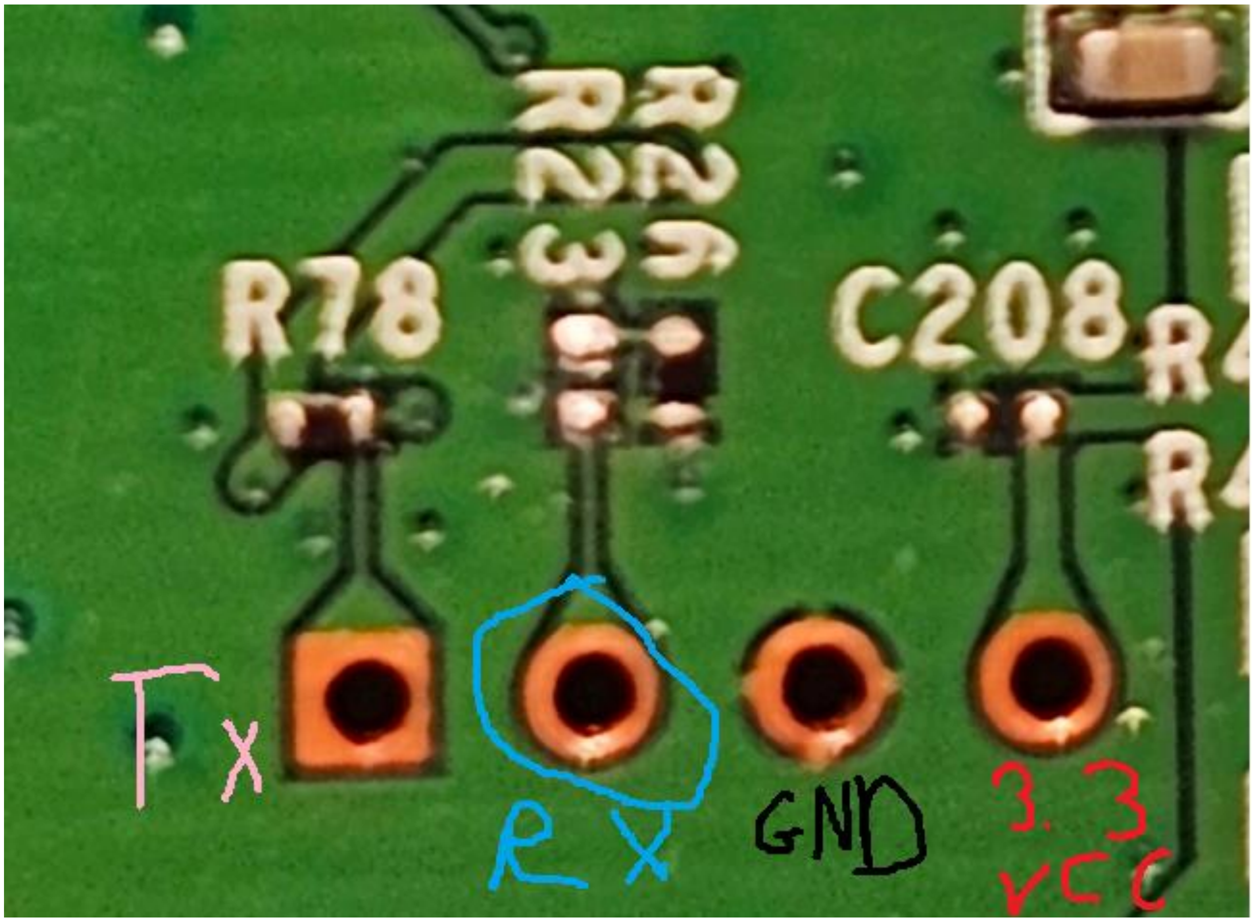
Decided this to be a hardware day, wanted to get the UART working. Found a nice youtube video with someone having the same problem but different router.

https://www.youtube.com/watch?v=01mw0oTHwxg&ab_channel=FlashbackTeam from 14:40, he has too a disabled RX pin.

This was his solution



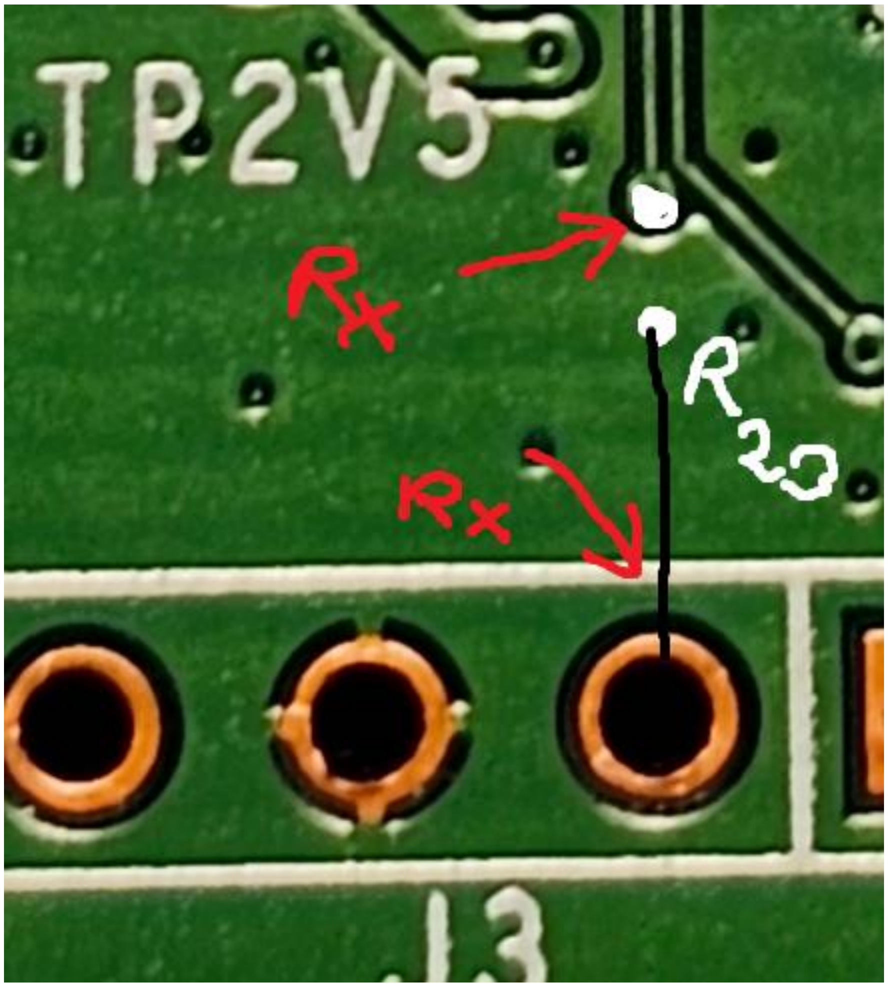
At first glance I have something that looks the same



I also have two Rxx ports and checking with a multimeter reveals that RX is connected to the lower part of R23 but it is not connected to the upper part of R23, which when traced with the eyesight seems to go straight to the micro controller.

At this point there are a lot of options, either connect R23 pins, or if I see correctly upper R23 is connected with upper R26 and maybe lower R23 should be connected with lower R26 so in this way the current will pass R26 resistor. But that is a speculation.

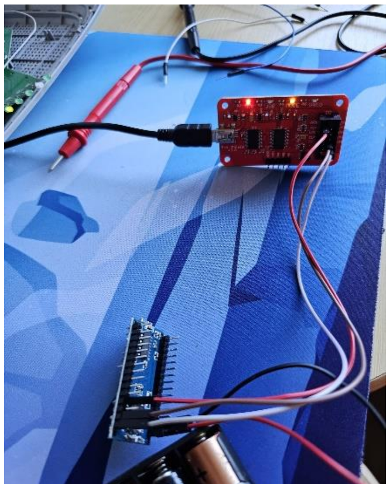
Looking at the other side and measuring with a piece of metal reveals that the connection going to the microcontroller would fit upper part of R23.

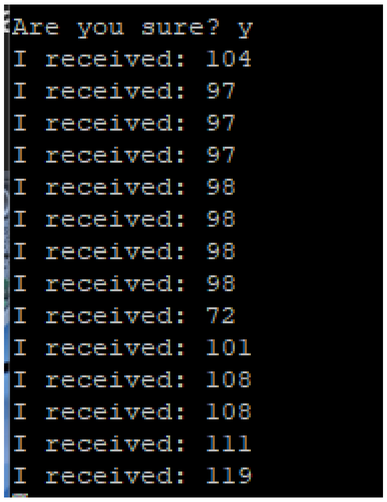


It should be easy right? Just hold my TX wire to that connection on the PCB, well I tried and it did not work I could not see my characters echoed on the screen. Tomorrow is another day!

02.11.2023

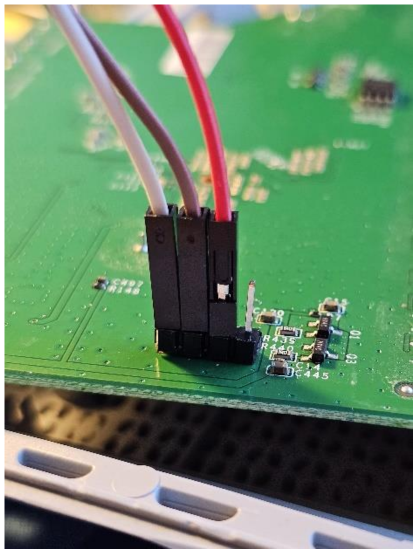
Started to get paranoic that my Bus Pirate does not work so I tested it with Arduino and I have seen that it works fine. A simple echo program.



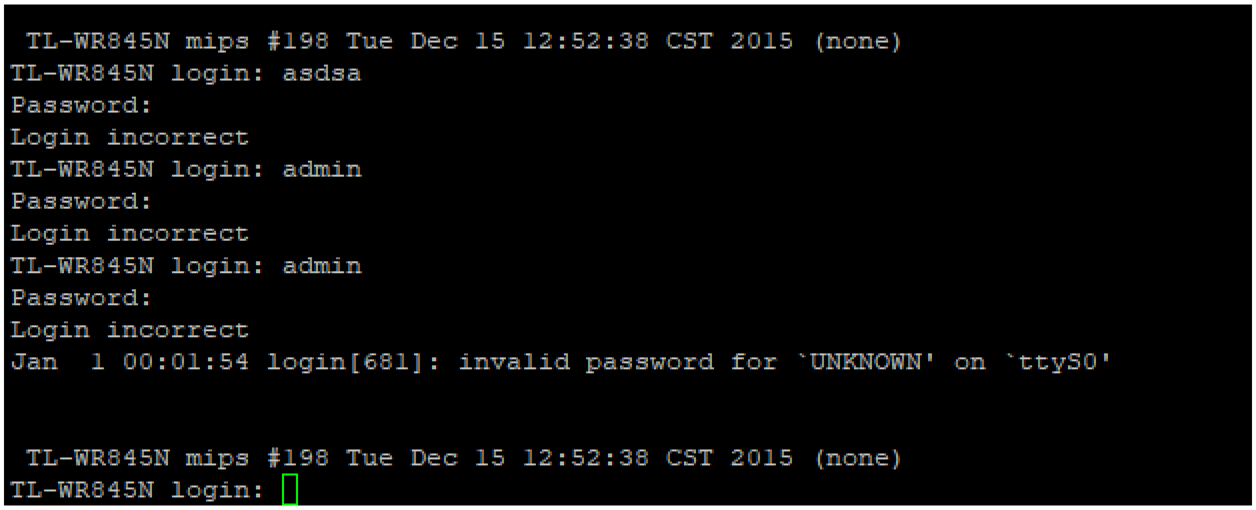


```
Are you sure? y
I received: 104
I received: 97
I received: 97
I received: 97
I received: 98
I received: 98
I received: 98
I received: 98
I received: 72
I received: 101
I received: 108
I received: 108
I received: 111
I received: 119
```

Firstly, I soldered pins for a better stability during the connection.



After that I tried holding a wire to the upper part of R23 for input and after a lot of reboots and tries, I could see my output being displayed 😉.



```
 TL-WR845N mips #198 Tue Dec 15 12:52:38 CST 2015 (none)
TL-WR845N login: asdsa
Password:
Login incorrect
TL-WR845N login: admin
Password:
Login incorrect
TL-WR845N login: admin
Password:
Login incorrect
Jan  1 00:01:54 login[681]: invalid password for `UNKNOWN' on `ttyS0'


 TL-WR845N mips #198 Tue Dec 15 12:52:38 CST 2015 (none)
TL-WR845N login: 
```

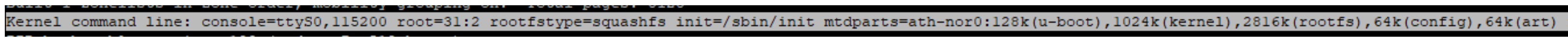
Unfortunately, no combination of trivial usernames or password was correct.

The input was unstable because I needed to fix a wire with my hand so I decided so solder R23 ports together resulting in a stable output. I am sure there was supposed to be a resistor but I did not want to try random values. 0 Ohm resistor it is.

I know it does not look very professional but the contact patches are really small.

Now I wanted to perform a naive jailbreaking by letting the bootloader load from the flash but do not let the OS run so I can change the boot arguments.

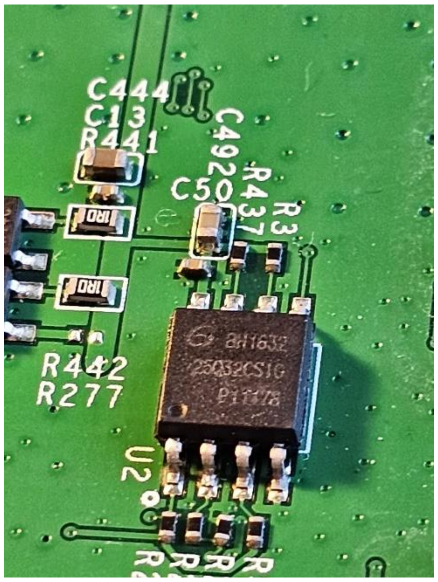
Especially I was interested in the **init** argument.



```
Kernel command line: console=ttyS0,115200 root=31:2 rootfstype=squashfs init=/sbin/init mtdparts=ath-nor0:128k(u-boot),1024k(kernel),2816k(rootfs),64k(config),64k(art)
```

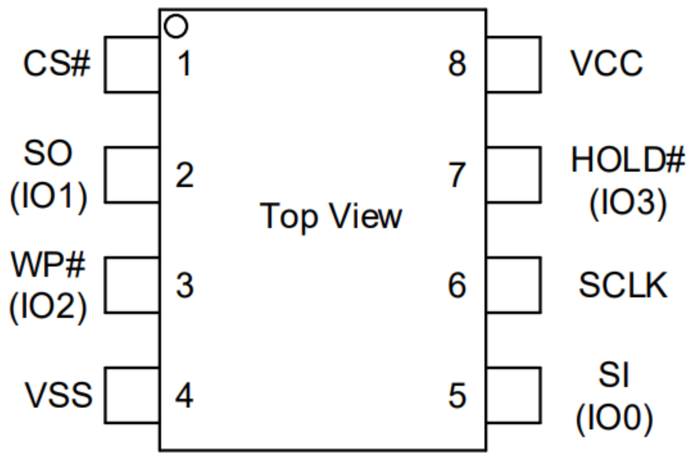
Would be nice to change it from */sbin/init* to */bin/sh* in order to bypass the login.

Firstly, I needed to find the flash memory. It is easy to find since usually it is a little squared circuit.



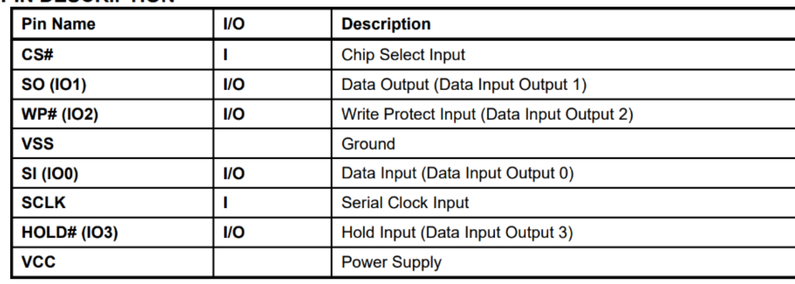
After searching on the internet the code in the second row ***25Q32CSIG*** we could easily found its datasheet: https://eu.mouser.com/datasheet/2/870/gd25q32c_v4.1_20200319-1668129.pdf

CONNECTION DIAGRAM



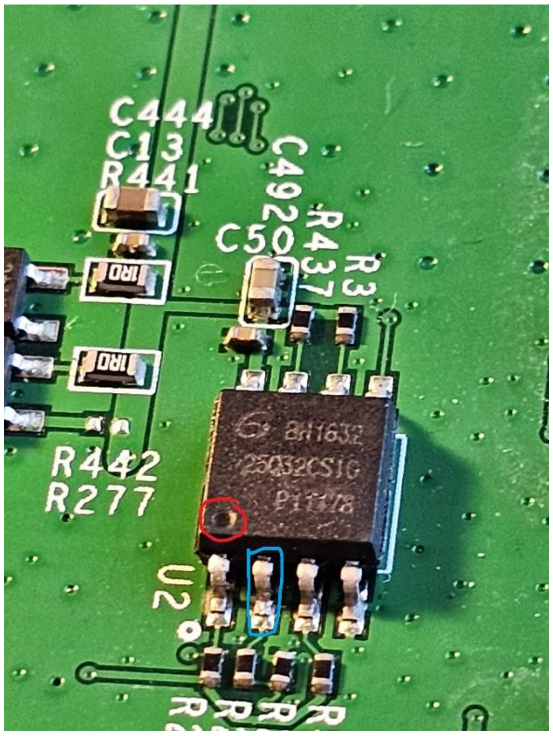
8 – LEAD SOP/VSOP/DIP

PIN DESCRIPTION



Pin Name	I/O	Description
CS#	I	Chip Select Input
SO (IO1)	I/O	Data Output (Data Input Output 1)
WP# (IO2)	I/O	Write Protect Input (Data Input Output 2)
VSS		Ground
SI (IO0)	I/O	Data Input (Data Input Output 0)
SCLK	I	Serial Clock Input
HOLD# (IO3)	I/O	Hold Input (Data Input Output 3)
VCC		Power Supply

We are interested in the SO port, to redirect the data output to GND.

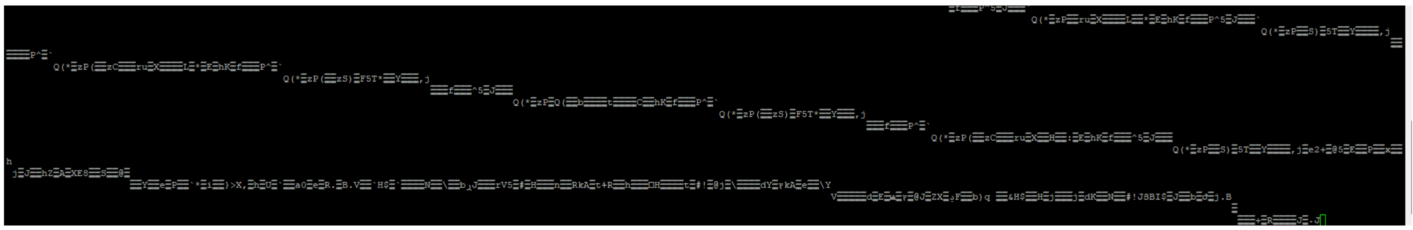


Red circle shows the dot on the chip so we can follow the schematic, and blue circle shows the SO (Data Out) pin.

Now we have to time when we make the SO -> GND connection in order to interrupt the normal boot process and be able to modify those boot arguments.

After a lot of tries so far, I ended up in 2 situations:

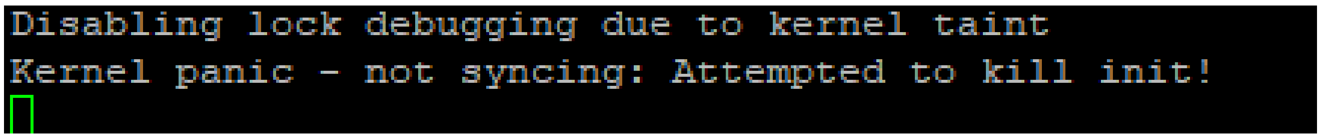
- To fast



```
[illegible]
```

Ended up in this region, I do not know what it is here, I suppose it is the bootloader.

- To late



```
Disabling lock debugging due to kernel taint
Kernel panic - not syncing: Attempted to kill init!
```

I only suppose it is to late, because I cannot input anything from the keyboard.

06.11.2023

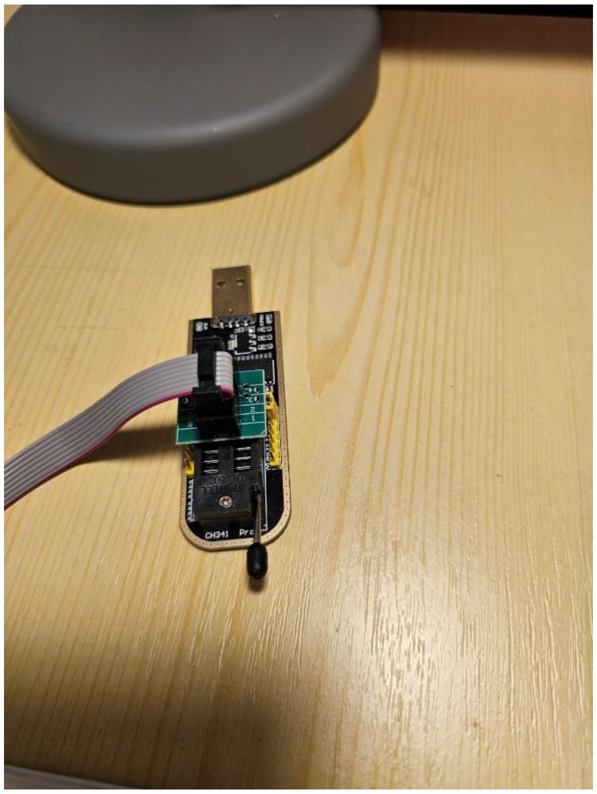
Today I tried to dump the contents of the flash chip. I have tried using the Bus-Pirate with the **Linux** command **flashrom** which supports bus pirate as the intermediate device. Tried a lot of firmware versions of the Bus-Pirate firmware but none work.

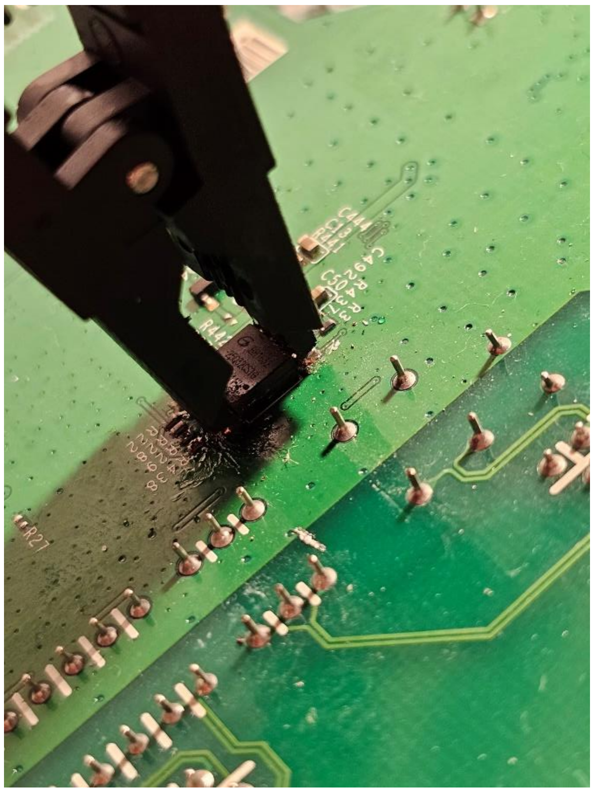
I also tried to read the contents by using the SPI from my Raspberry Pi 3B+ but it did not work again. In both cases I get no chip found so the **flashrom** command does not recognize my chip from its list but it is in there.

Long day 8h+ of trying and soldering with no result. Next time will try to read the flash from another router.

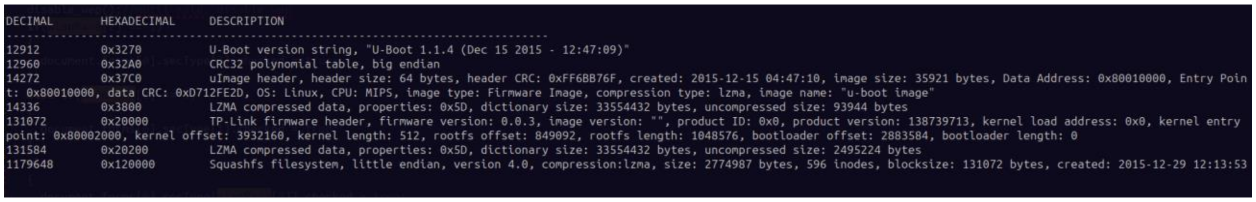
25.11.2023

After a long break, my **CH340** programmer arrived and I was able to read the flash on the router using **AsProgrammer V2.1.0.13**.



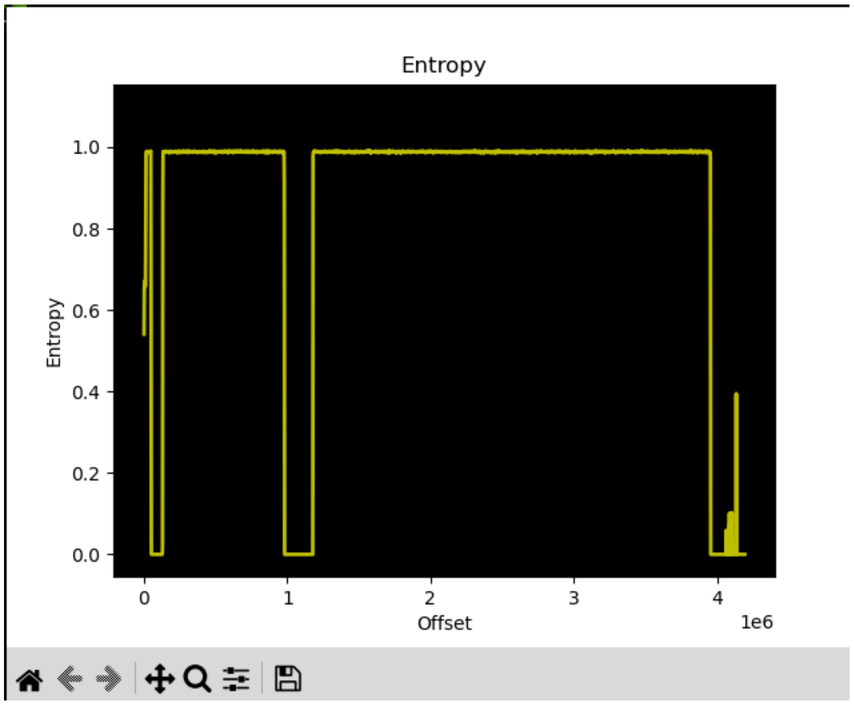


Quick **binwalk** command reveals this:



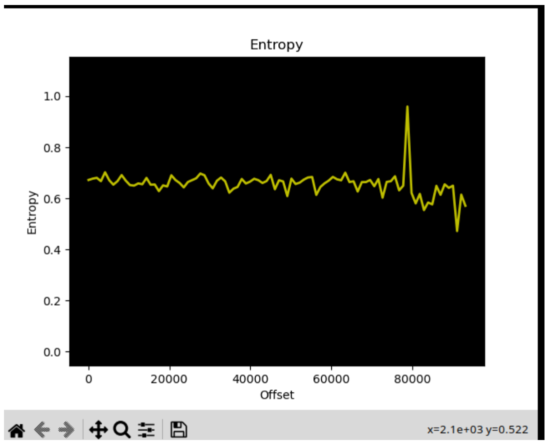
```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
12912         0x3270          U-Boot version string, "U-Boot 1.1.4 (Dec 15 2015 - 12:47:09)"
12960         0x32A0          CRC32 polynomial table, big endian
14272         0x37C0          uImage header, header size: 64 bytes, header CRC: 0xFF6BB76F, created: 2015-12-15 04:47:10, image size: 35921 bytes, Data Address: 0x80010000, Entry Poin
t: 0x80010000, data CRC: 0xD712FE2D, OS: Linux, CPU: MIPS, image type: Firmware Image, compression type: lzma, image name: "u-boot image"
14336         0x3800          LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 93944 bytes
131072        0x20000         TP-Link firmware header, firmware version: 0.0.3, image version: "", product ID: 0x0, product version: 138739713, kernel load address: 0x0, kernel entry
point: 0x80002000, kernel offset: 3932160, kernel length: 512, rootfs offset: 849092, rootfs length: 1048576, bootloader offset: 2883584, bootloader length: 0
131584        0x20200         LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 2495224 bytes
1179648       0x120000        Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 2774987 bytes, 596 inodes, blocksize: 131072 bytes, created: 2015-12-29 12:13:53
```

We can see that most of it is compressed.

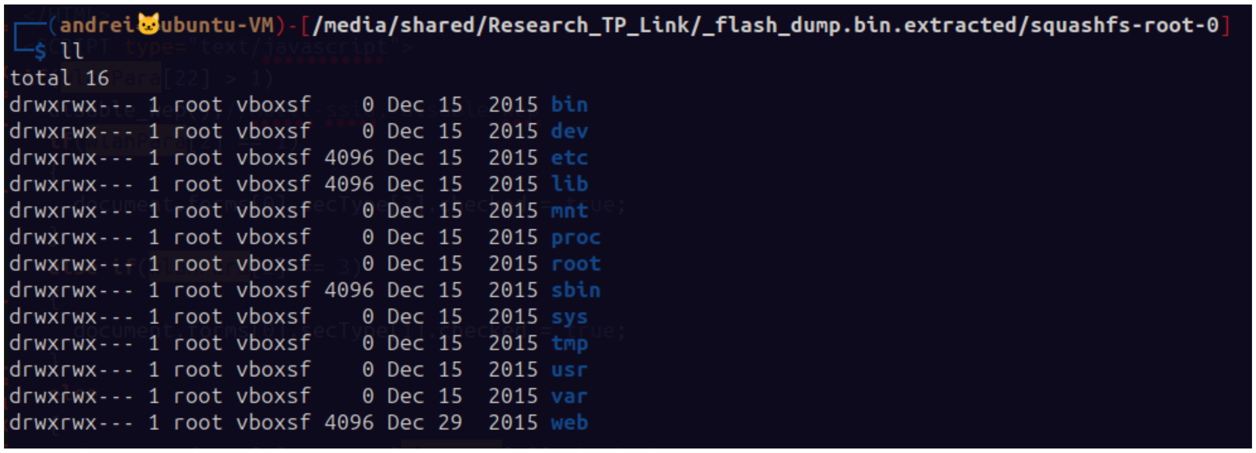


In the rest of the files not much information is useful, either U-Boot, certificates or kernel.

Decompressed lzma files and after running `binwalk -E` again on them it appears their entropy is somewhat bellow 0.7 which is the threshold for compressed/encrypted data.

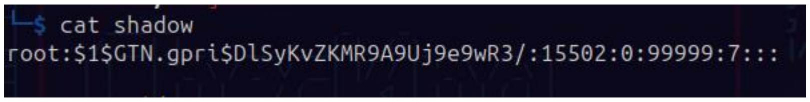


In the squashfs a valid Linux file system is present.



```
┌──(andrei🐱ubuntu-VM)-[/media/shared/Research_TP_Link/_flash_dump.bin.extracted/squashfs-root-0]
└─$ ll
total 16
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 bin
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 dev
drwxrwx--- 1 root vboxsf 4096 Dec 15  2015 etc
drwxrwx--- 1 root vboxsf 4096 Dec 15  2015 lib
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 mnt
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 proc
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 root
drwxrwx--- 1 root vboxsf 4096 Dec 15  2015 sbin
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 sys
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 tmp
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 usr
drwxrwx--- 1 root vboxsf    0 Dec 15  2015 var
drwxrwx--- 1 root vboxsf 4096 Dec 29  2015 web
```

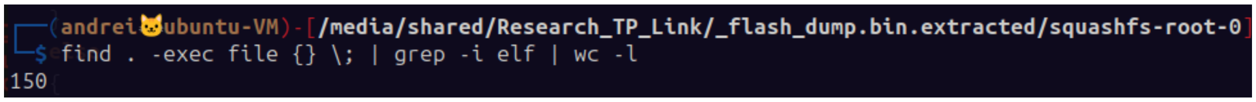
In /etc of the squashfs is the **shadow** file.



```
└─$ cat shadow
root:$1$GTN.gpri$DlSyKvZKMR9A9Uj9e9wR3/:15502:0:99999:7:::
```

While trying to bruteforce the hash with **rockyou.txt** as wordlist. I just googled the hash online and found a lot of results, saying that the root password is **sohoadmin**. Rushed to login as root via ssh with this password but did not work ☹.

After running a find command to look for ELF binaries we see that there are 150, all of them are either default networking binaries or libraries and one other is busybox.



```
┌──(andrei🐱ubuntu-VM)-[/media/shared/Research_TP_Link/_flash_dump.bin.extracted/squashfs-root-0]
└─$ find . -exec file {} \; | grep -i elf | wc -l
150
```

I also have another router that I was able to extract its firmware, if I am stuck here maybe I will give that one a try.