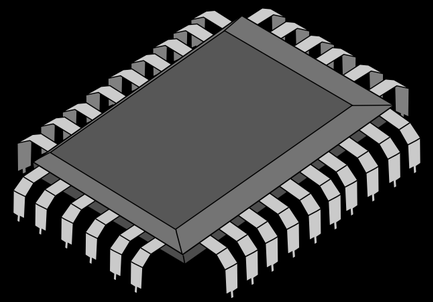
UTS:CSEC + MQCYBER

INTRO TO

HARDWARE HACKING



Acknowledgement of country

UTS:CSEC would like to acknowledge the Gadigal people of the Eora Nation upon whose ancestral lands our City campus now stands. We would also like to pay respect to the Elders both past, present, and future, acknowledging them as the traditional custodians of knowledge for this land.

University Sponsors

Platinum Sponsors

Gold Sponsors

VOLKIS™

Essential Energy

What is hardware hacking?

Hardware hacking refers to the practice of manipulating or exploiting electronic systems to gain unauthorized access, extract sensitive information, or compromise system security. Unlike software-based attacks, hardware hacking involves direct physical interaction with the target device's components.

Our target for today

TP-Link TL-WR841N

Board overview

Whats the goal?

Have a root shell on the device + a clean dump of the firmware

How do we do that?

1. Find an entrypoint to the device
    a. Serial (UART)
    b. Flash Dumping
    c. PCI-e (DMA)

^(yes seriously) (not covering today)

2. Mess with it

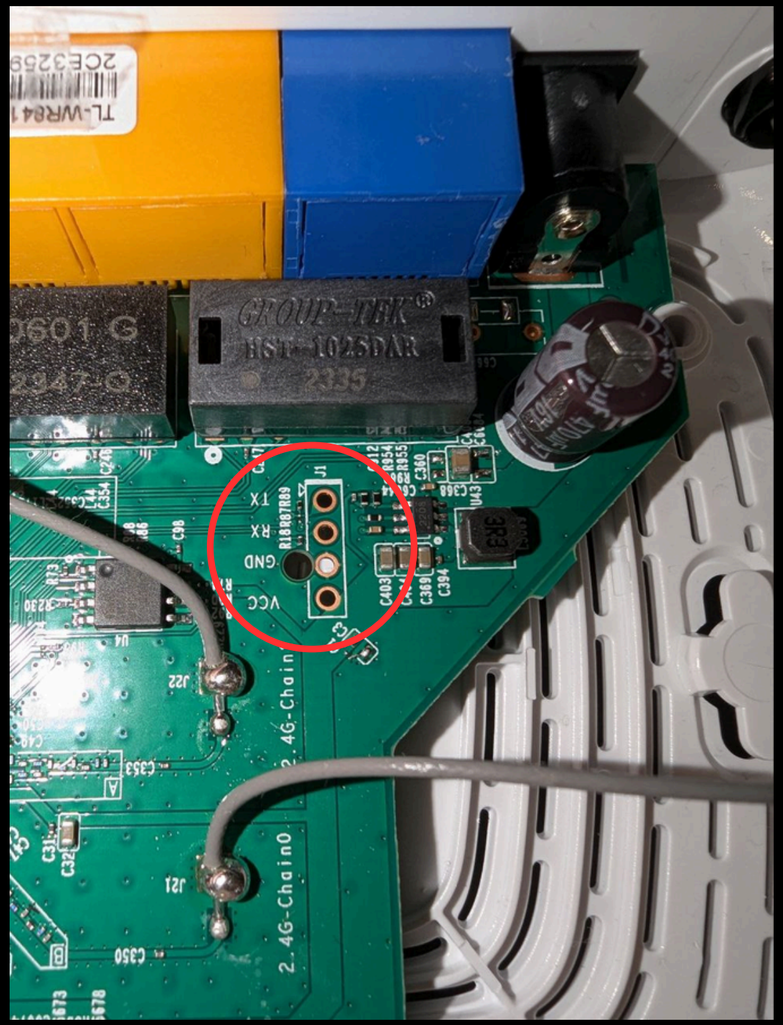
UART (Serial)

What is UART?

UART is a hardware communication protocol that enables serial data transmission between devices. It uses two wires for communication—one for transmitting (TX) and one for receiving (RX). UART is widely used in embedded systems and microcontrollers.

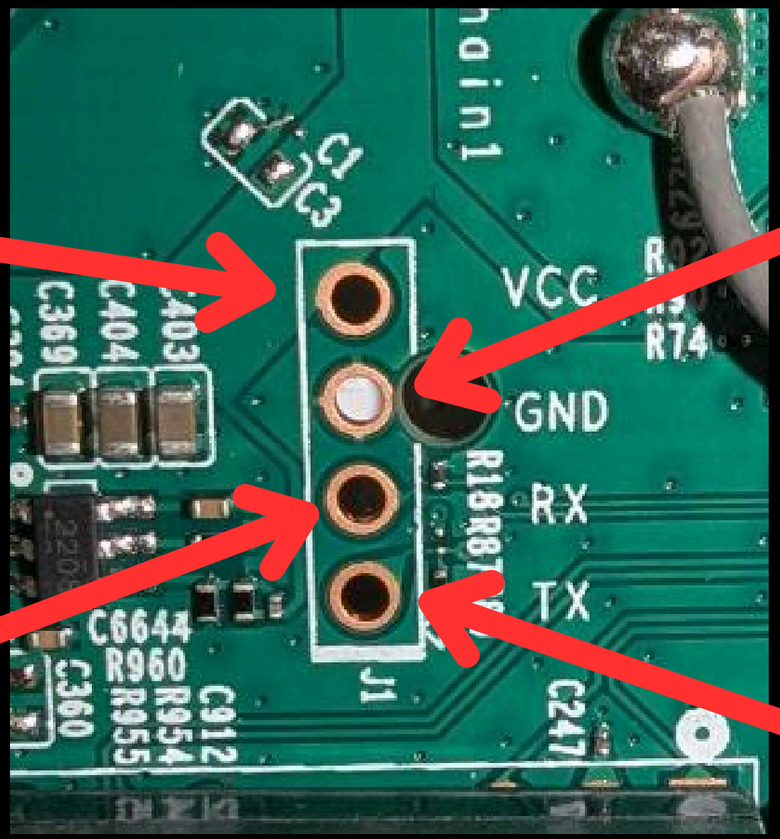
Board overview

Finding exposed UART headers



Finding exposed UART headers

VCC = Positive (+)
Note: This will only power the CPU, NOT THE WHOLE DEVICE.



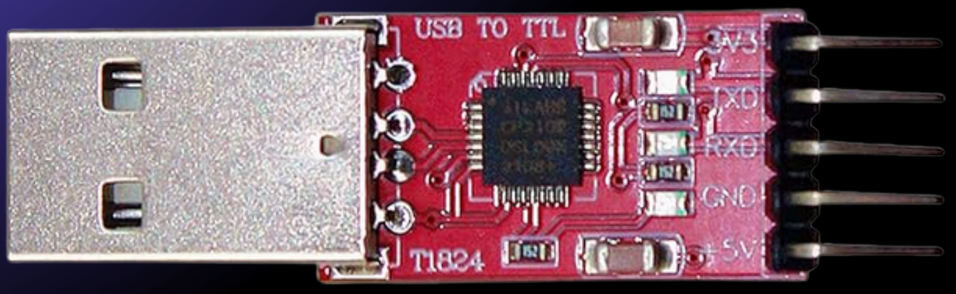
GND = Ground (-)

Board RX (receive)
Note: This is the **CPU receiving** serial

Board TX (transmit)
Note: This is the **CPU transmitting** serial

How do I connect to the UART headers?

Use a serial to USB device



These are extremely cheap (~$10) and give you a way to interface with serial easily

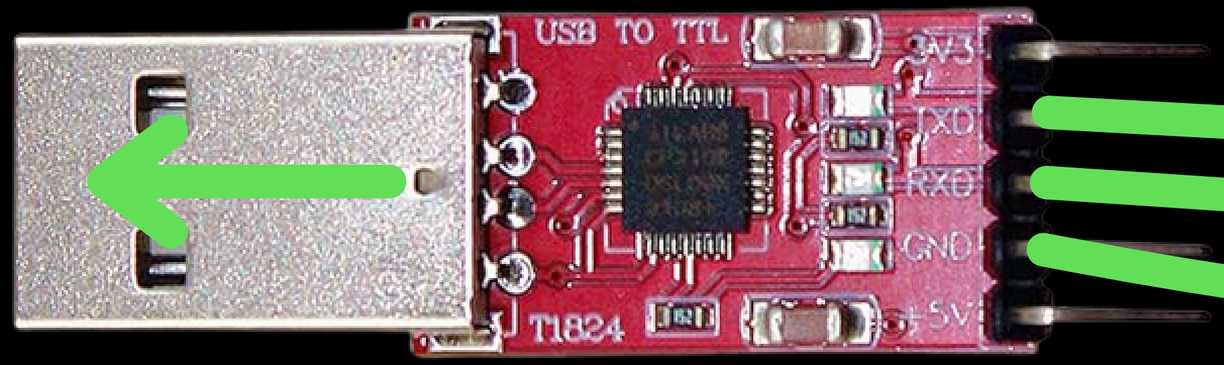
(yes, the flipper zero does serial to USB very well but is not ~$10)

How do I connect to the UART headers? - Example

Usually its best to NOT connect the VCC (+) pin, you risk frying the device

You MUST share a ground between the device and converter

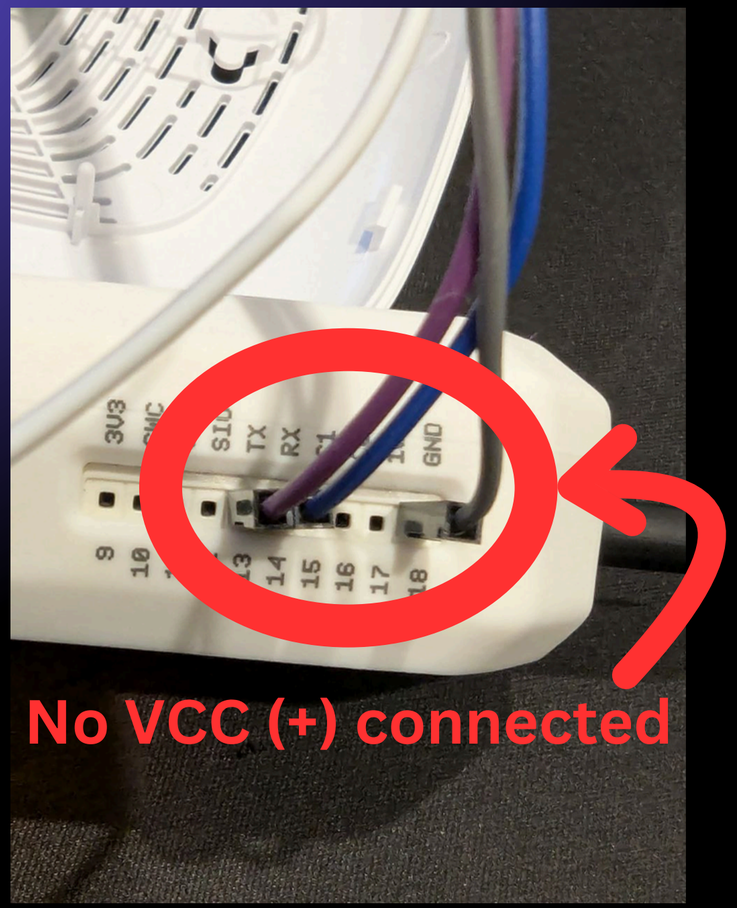
Serial RX + TX pins must be swapped from board to converter

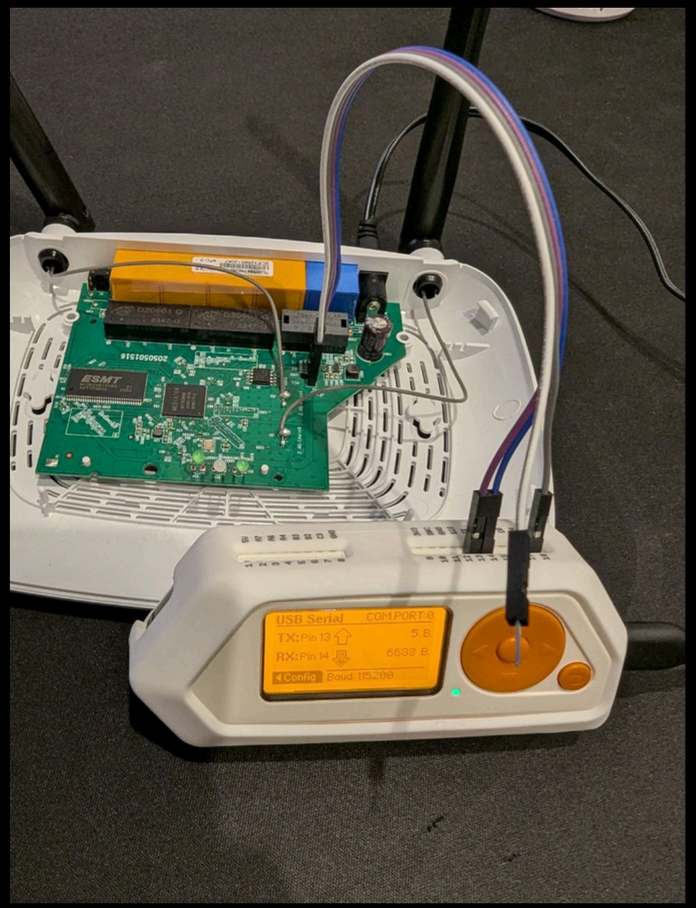


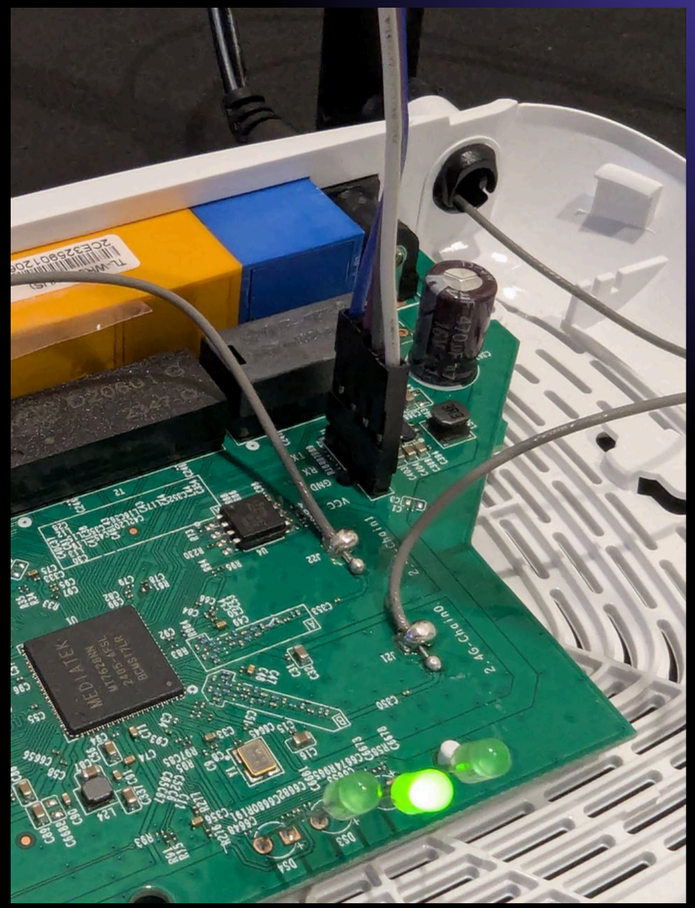
to computer

Back to the router..

After soldering headers to the pins (for ease of connection), we connect to the flipper (my serial to USB)







Interacting with serial on the computer

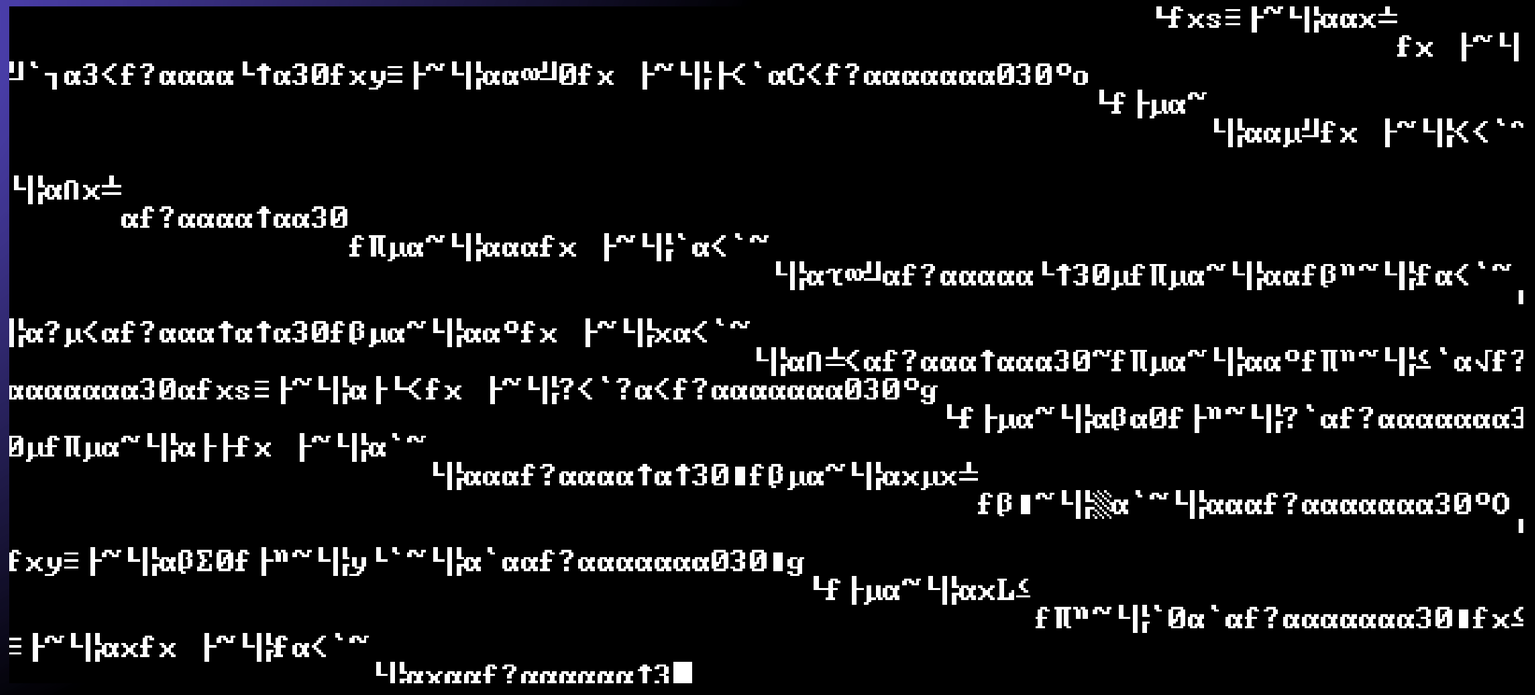
There are a million programs to do this, easiest to use is “minicom”

sudo minicom -D <serial device path>

For example - sudo minicom -D /dev/ttyACM0

(if you dont know your usb serial path, you can look for devices starting with “tty” in “/dev/”
(ls -al /dev/)

Baud Rates



After opening the serial connection, we get jumbled characters. This is because the “baud rate” (the speed commands are sent between devices) is wrong, they must be matching on each side

Baud Rates

Most Common Baud Rates

- 9600
- 19200
- 38400
- 57600
- 115200

Minicom will default to 9600. Lets try 115200, the **other** most common baud rate for embedded devices

Baud Rates

```
[04080B0E][04080B0E][7F7F0000][18182F2F][00181838]
DU Setting Cal Done


U-Boot 1.1.3 (Aug 16 2022 - 12:01:12)

Board: Ralink APSoC DRAM:  32 MB
relocate_code Pointer at: 81fc0000
flash manufacture id: 1c, device id 70 16
Warning: un-recognized chip ID, please update bootloader!
=========================================
Ralink UBoot Version: 4.3.0.0
-----------------------------------------
ASIC 7628_MP (Port5<->None)
DRAM component: 256 Mbits DDR, width 16
DRAM bus: 16 bit
Total memory: 32 MBytes
Flash component: SPI Flash
Date:Aug 16 2022  Time:12:01:12
=========================================
icache: sets:512, ways:4, linesz:32 ,total:65536
dcache: sets:256, ways:4, linesz:32 ,total:32768

 ##### The CPU freq = 580 MHZ ####
 estimate memory size =32 Mbytes
RESET MT7628 PHY!!!!!!
continue to starting system.
 0
disable switch phyport...

3: System Boot system code via Flash.(0xbc010000)
do_bootm:argc=2, addr=0xbc010000
## Booting image at bc010000 ...
   Uncompressing Kernel Image ... OK
No initrd
## Transferring control to Linux (at address 8000c150) ...
## Giving linux memsize in MB, 32

Starting kernel ...


LINUX started...

 THIS IS ASIC
Linux version 2.6.36 (jenkins@sohoici) (gcc version 4.6.3 (Buildroot 2012.11.1) ) #1 Tue Aug 16 12:04:29 CST 2022

 The CPU feqenuce set to 575 MHz

 MIPS CPU sleep mode enabled.
CPU revision is: 00019655 (MIPS 24Kc)
Software DMA cache coherency
Determined physical RAM map:
 memory: 02000000 @ 00000000 (usable)
Initrd not found or empty - disabling initrd
Zone PFN ranges:
  Normal   0x00000000 -> 0x00002000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0: 0x00000000 -> 0x00002000
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 8128
Kernel command line: console=ttyS1,115200 root=/dev/mtdblock2 rootfstype=squashfs init=/sbin/init
PID hash table entries: 128 (order: -3, 512 bytes)
```

Success! We get proper output!

What does the router reply?

```
[04080B0E][04080B0E][7F7F0000][18182F2F][00181838]
DU Setting Cal Done


U-Boot 1.1.3 (Aug 16 2022 - 12:01:12)

Board: Ralink APSoC DRAM:  32 MB
relocate_code Pointer at: 81fc0000
flash manufacture id: 1c, device id 70 16
Warning: un-recognized chip ID, please update bootloader!
=========================================
Ralink UBoot Version: 4.3.0.0
-----------------------------------------
ASIC 7628_MP (Port5<->None)
DRAM component: 256 Mbits DDR, width 16
DRAM bus: 16 bit
Total memory: 32 MBytes
Flash component: SPI Flash
Date:Aug 16 2022  Time:12:01:12
=========================================
icache: sets:512, ways:4, linesz:32 ,total:65536
dcache: sets:256, ways:4, linesz:32 ,total:32768

 ##### The CPU freq = 580 MHZ ####
 estimate memory size =32 Mbytes
RESET MT7628 PHY!!!!!!
continue to starting system.
 0
disable switch phyport...

3: System Boot system code via Flash.(0xbc010000)
do_bootm:argc=2, addr=0xbc010000
## Booting image at bc010000 ...
   Uncompressing Kernel Image ... OK
No initrd
## Transferring control to Linux (at address 8000c150) ...
## Giving linux memsize in MB, 32

Starting kernel ...


LINUX started...

 THIS IS ASIC
Linux version 2.6.36 (jenkins@sohoici) (gcc version 4.6.3 (Buildroot 2012.11.1) ) #1 Tue Aug 16 12:04:29 CST 2022

 The CPU feqenuce set to 575 MHz

 MIPS CPU sleep mode enabled.
CPU revision is: 00019655 (MIPS 24Kc)
Software DMA cache coherency
Determined physical RAM map:
 memory: 02000000 @ 00000000 (usable)
Initrd not found or empty - disabling initrd
Zone PFN ranges:
  Normal   0x00000000 -> 0x00002000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0: 0x00000000 -> 0x00002000
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 8128
Kernel command line: console=ttyS1,115200 root=/dev/mtdblock2 rootfstype=squashfs init=/sbin/init
PID hash table entries: 128 (order: -3, 512 bytes)
```

Just boring bootloader logs, although we do see that its running U-Boot, an open source bootloader thats super common in embedded devices.

We can use this as another attack vector to spawn a root shell (wont be covering today)

What does the router reply?

```
[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/netfilter/nf_conntrack_h323.ko"

[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/ipv4/netfilter/nf_nat_h323.ko"

[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/netfilter/nf_conntrack_sip.ko"

[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/ipv4/netfilter/nf_nat_sip.ko"

[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/netfilter/nf_conntrack_rtsp.ko"

[ util_execSystem ] 185:  setupModules cmd is "insmod /lib/modules/kmdir/kernel/net/ipv4/netfilter/nf_nat_rtsp.ko"

nf_nat_rtsp v0.6.21 loading
enable switch phyport...
Set: phy[0].reg[0] = 3900
Set: phy[1].reg[0] = 3900
Set: phy[2].reg[0] = 3900
Set: phy[3].reg[0] = 3900
Set: phy[4].reg[0] = 3900
Set: phy[0].reg[0] = 3300
Set: phy[1].reg[0] = 3300
Set: phy[2].reg[0] = 3300
Set: phy[3].reg[0] = 3300
Set: phy[4].reg[0] = 3300
resetMiiPortV over.
Set: phy[0].reg[4] = 01e1
Set: phy[0].reg[0] = 3300
Set: phy[1].reg[4] = 01e1
[cmd_dutInit():1094] init shm
[tddp_taskEntry():151] tddp task start
Set: phy[1].reg[0] = 3300
Set: phy[2].reg[4] = 01e1
Set: phy[2].reg[0] = 3300
Set: phy[3].reg[4] = 01e1
Set: phy[3].reg[0] = 3300
Set: phy[4].reg[4] = 01e1
Set: phy[4].reg[0] = 3300
turn off flow control over.
[ util_execSystem ] 185:  prepareDropbear cmd is "dropbearkey -t rsa -f /var/tmp/dropbear/dropbear_rsa_host_key"

Will output 1024 bit rsa secret key to '/var/tmp/dropbear/dropbear_rsa_host_key'
Generating key, this may take a while...
[ util_execSystem ] 185:  prepareDropbear cmd is "dropbearkey -t dss -f /var/tmp/dropbear/dropbear_dss_host_key"

Will output 1024 bit dss secret key to '/var/tmp/dropbear/dropbear_dss_host_key'
Generating key, this may take a while...

~ #
~ # ls
web      usr      sbin     mnt      lib      dev
var      sys      proc     linuxrc  etc      bin
~ # ls /usr/bin/
xtables-multi  tdpd           killall        igmpd          dhcpd
wscd           tddp           iwpriv         httpd          dhcpc
wlNetlinkTool  tc             iwconfig       free           cos
wanType        taskset        iptables       ebtables       cmxdns
upnpd          scp            ipping         dyndns         ated_tp
traceroute     rt2860apd      ipcs           dropbearmulti  arping
top            pwdog          ipcrm          dropbearkey    afcd
tmpd           ntpc           ip6tables      dropbear
tftp           noipdns        ip             dnsProxy
~ #
```

Unfortunately, this router is extremely insecure and has root shell open without authentication over serial. Most devices **WILL NOT.** For the sake of this talk, we are going to assume that there is a password prompt here of which we need to find the password.

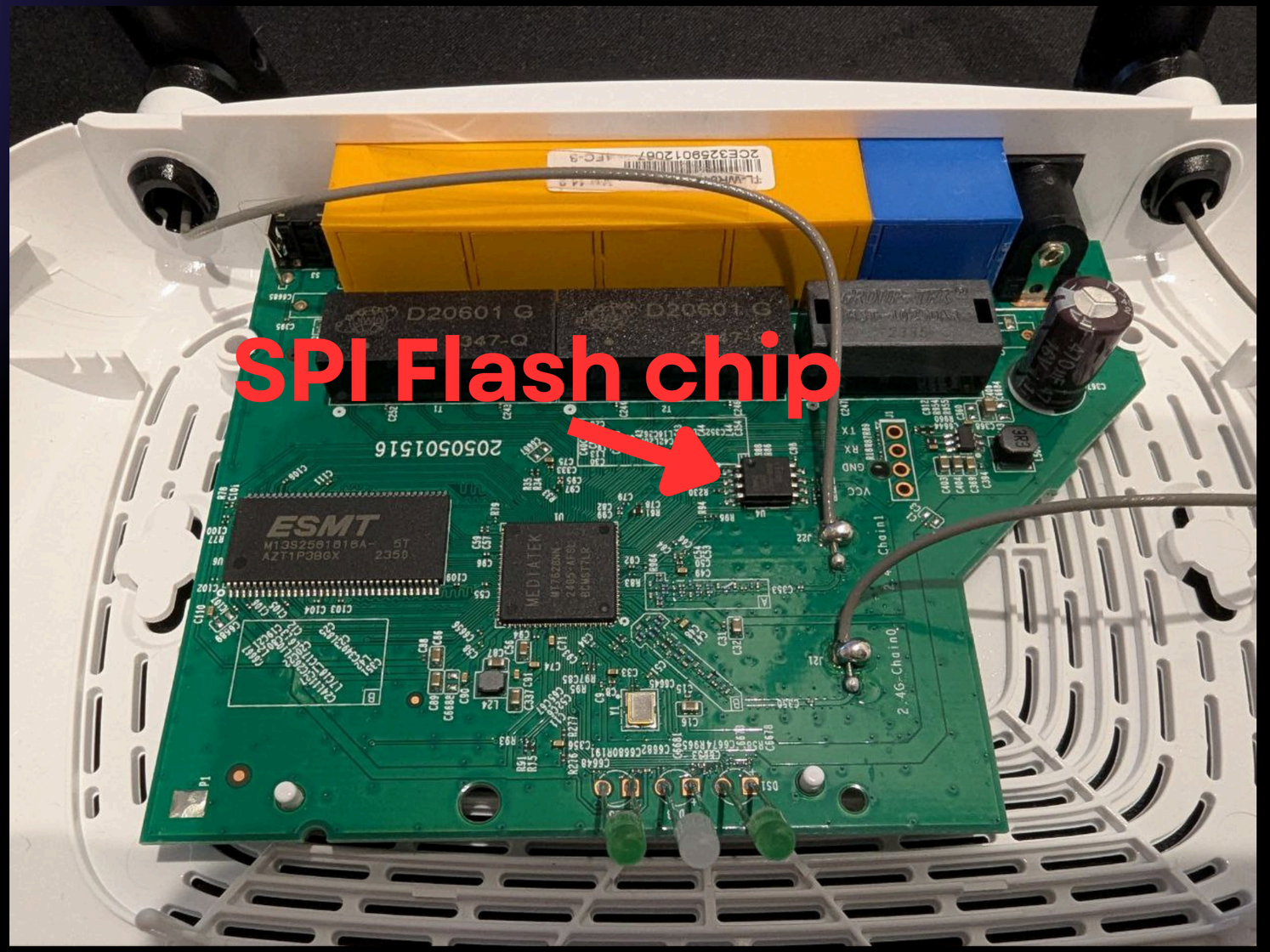
How would we get a login?

By dumping the firmware and having a poke around

Flash Dumping

Where’s Waldo (firmware edition)

Where’s Waldo (firmware edition)



What is flash dumping?

Simply: Making a copy of the system that's stored on the flash chip

There are usually a few partitions on the chip:

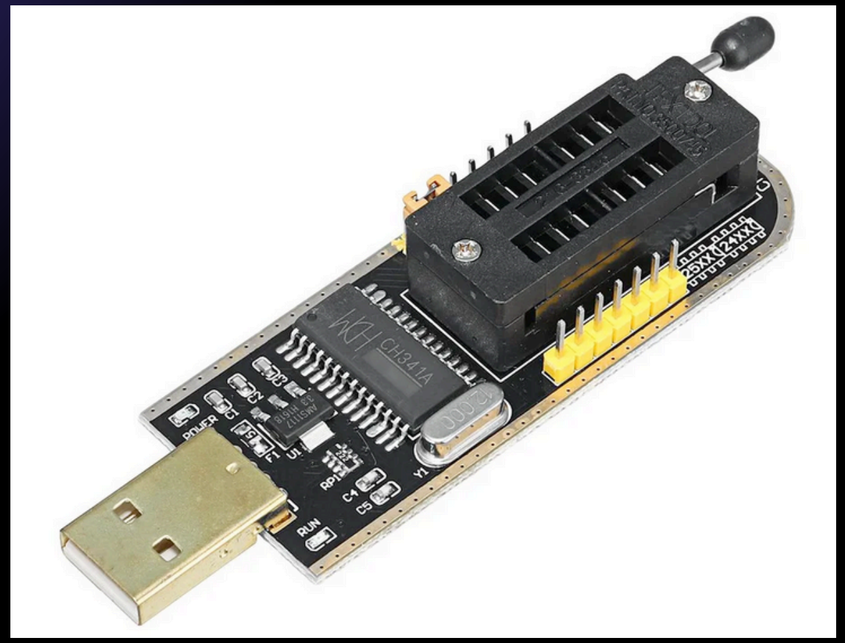
1. The actual OS, mostly in a read only file system like SquashFS
2. A preferences partition that is r/w to store data
3. An update partition to store files while the main fs is updating

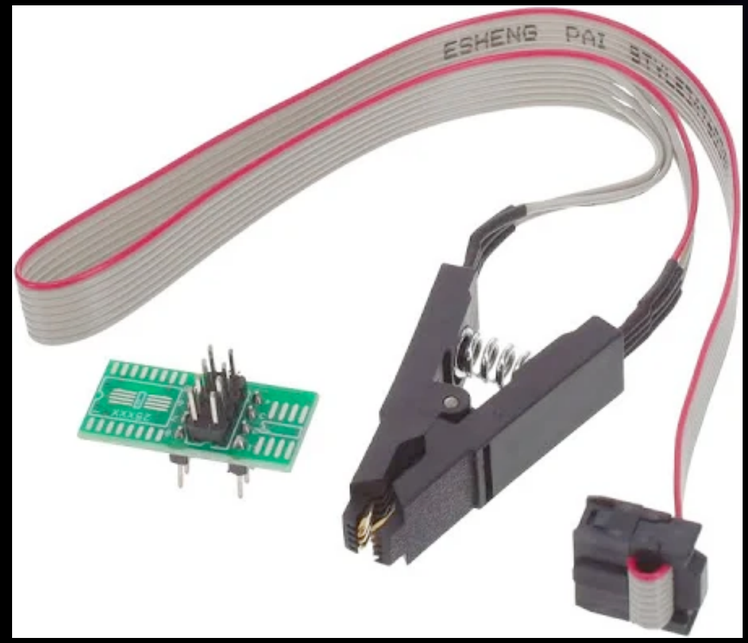
Why is flash dumping useful?

It allows us (the leet hackerman) to have a copy of every binary, library, and any other file that the product has on it. This includes hidden / developer binaries, sensitive files (e.g. /etc/passwd), source code (e.g. router webpage HTML), and more!

How to dump all the firmwares

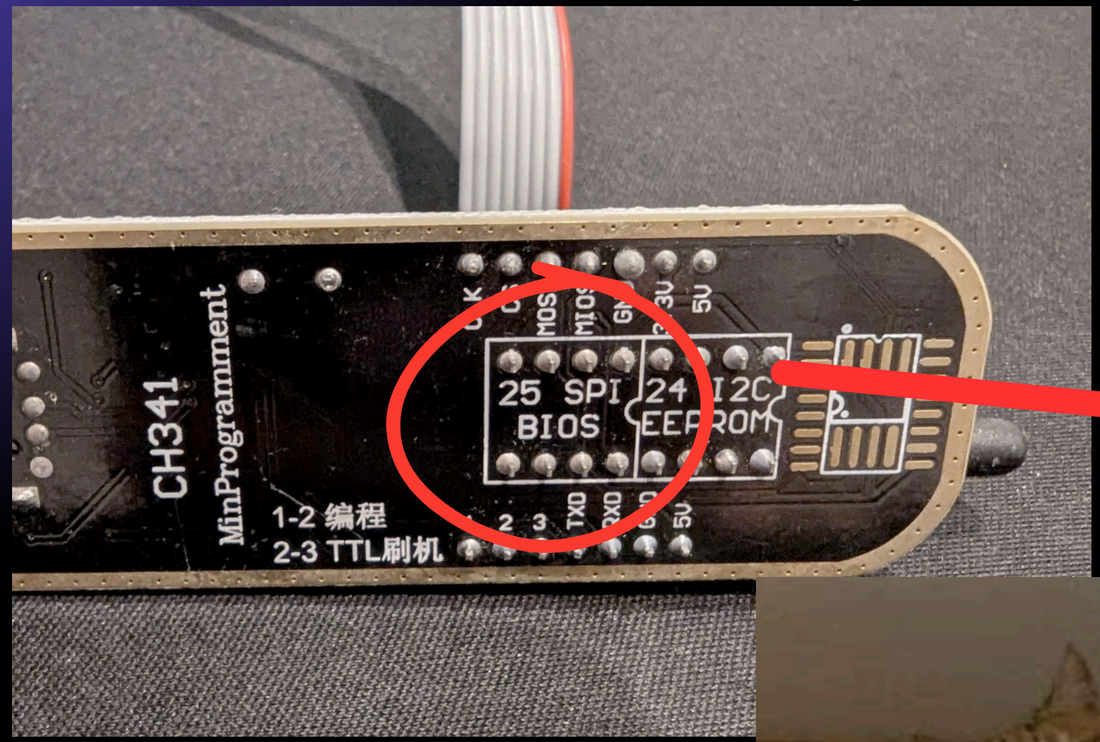
The most commonly used (and what we will use for this workshop) is the CH134A and a SOP8 clamp!

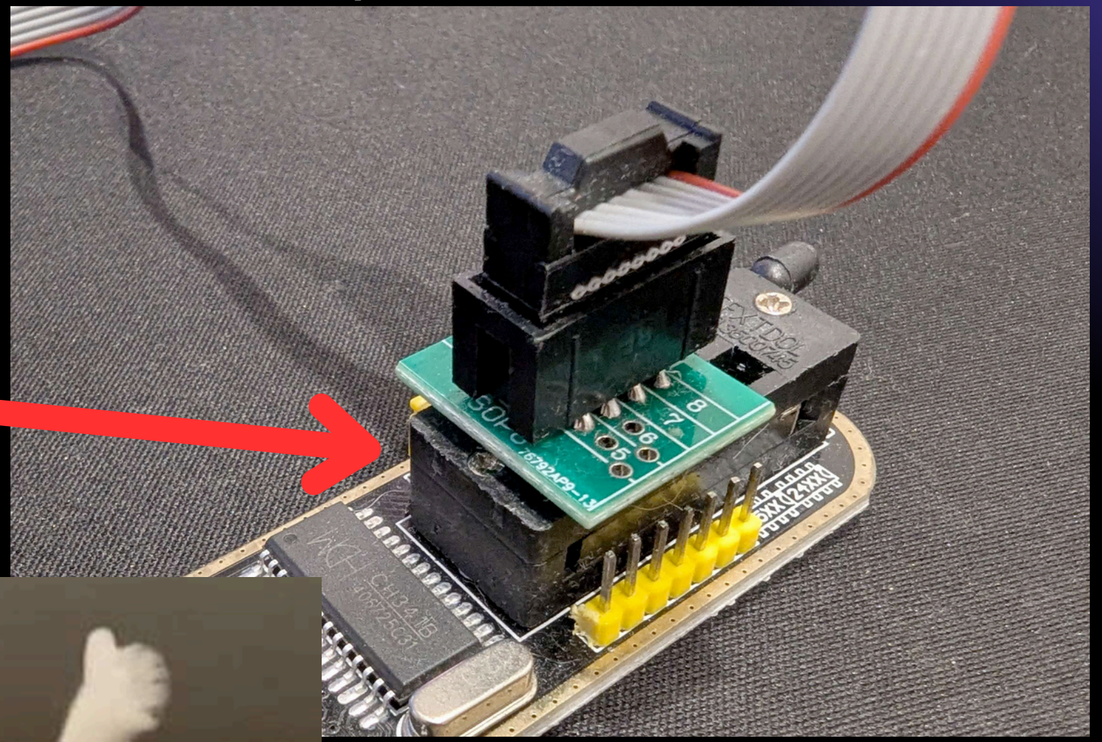




Building the damn thing

We are dumping an SPI chip, so we need to make sure to place the adapter on the correct pins

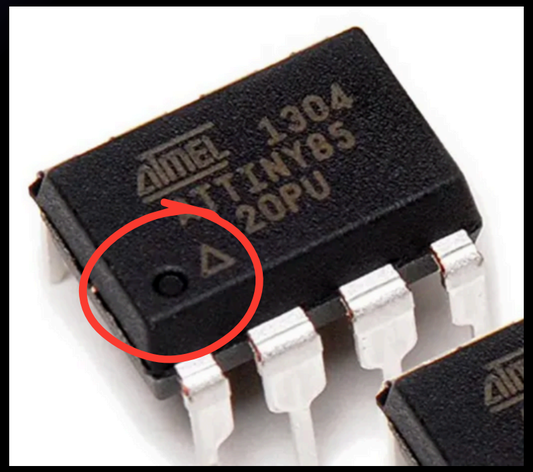


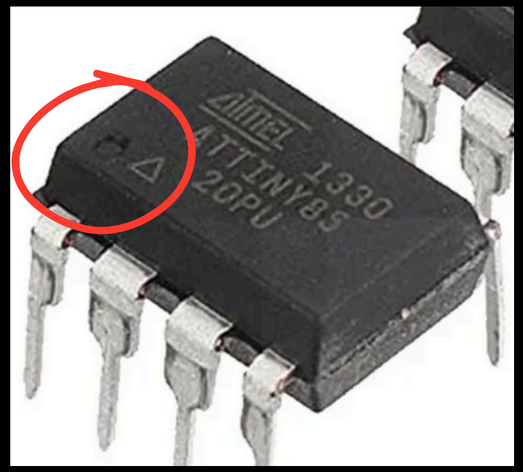


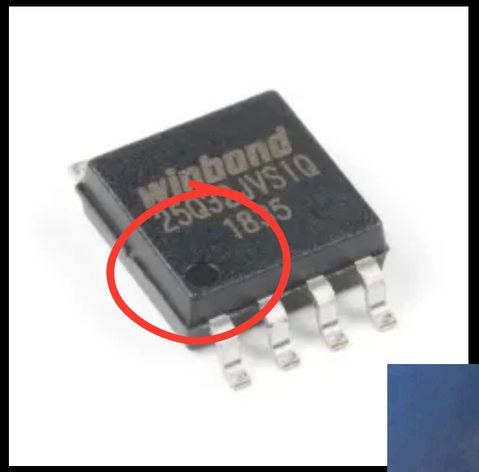
Where do I put the clamp?

Generally, the chips will have:

- 4 pins on either side (for SOP firmware chips)
- a small dot/hole above 1 of the pins, this is pin 1 (generally)

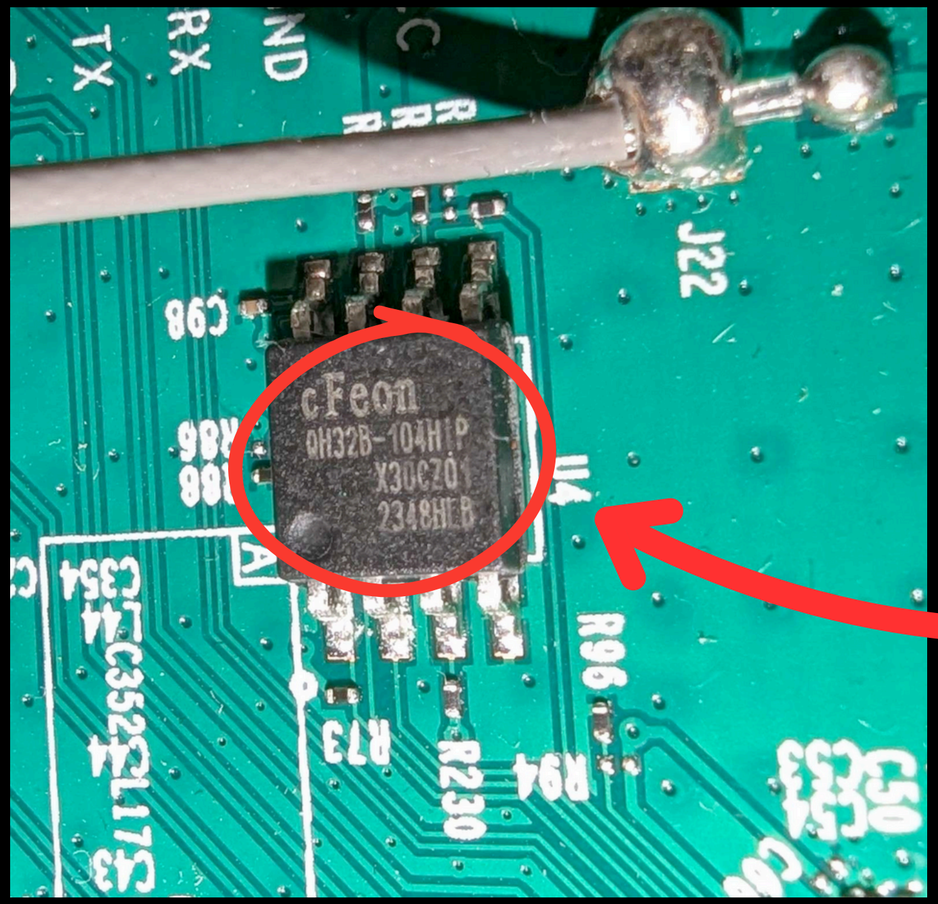




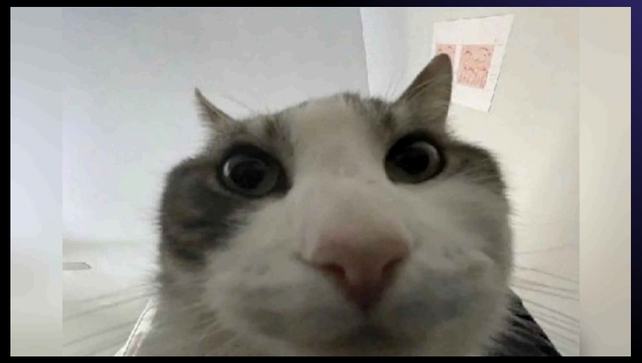


trying to spot the tiny dot

Identifying the chip

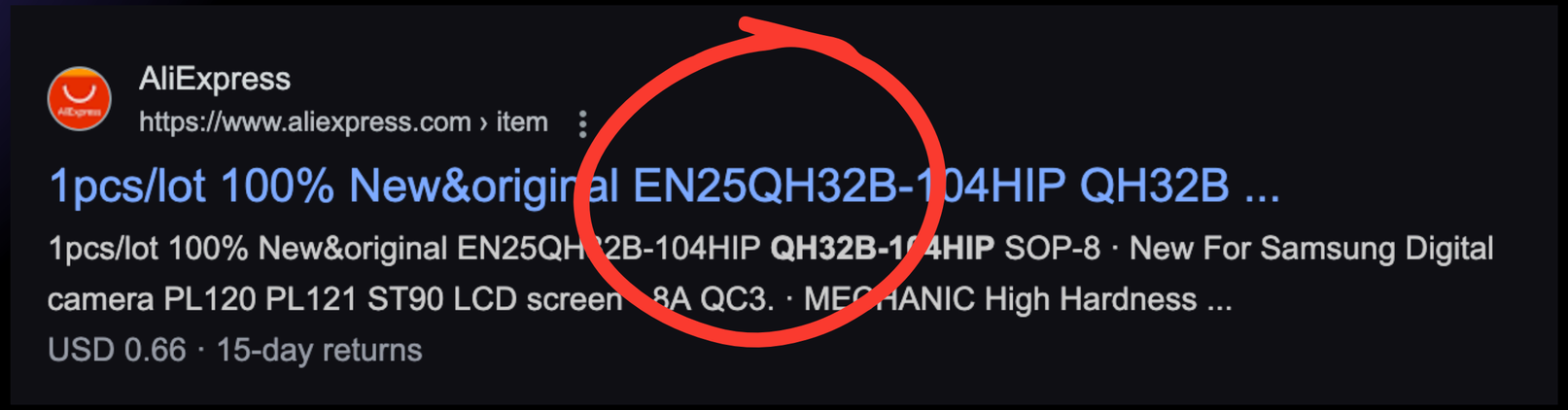


that text is tiny



Identifying the chip

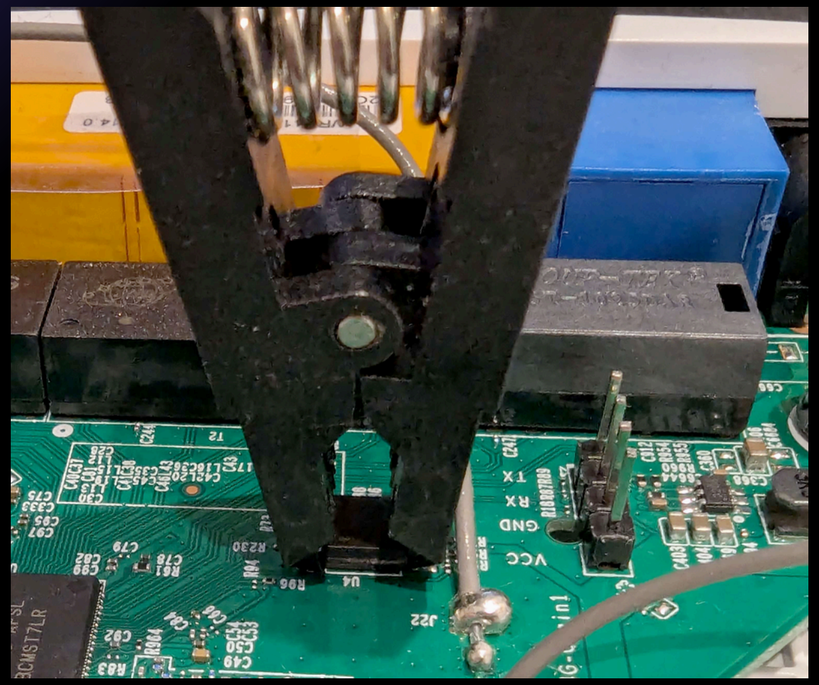
With a little bit of interwebs
on ‘QH32B-104HIP’...

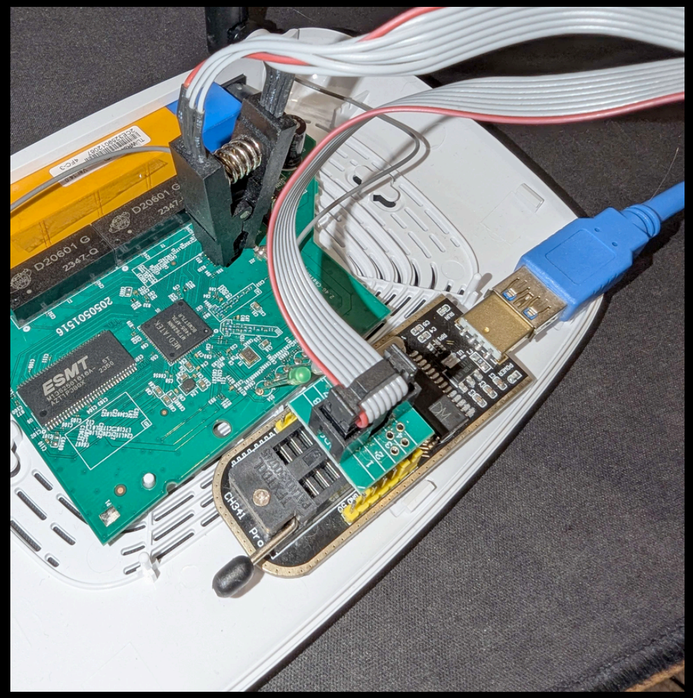


Seems to be called the
‘EN25QH32B’!

Clamp time

Note: make sure that the red wire on the clamp lines up with the pin 1 on the chip, same goes for the CH134a!





Dumping the firmware

There are Windows GUI options for this (AsProgrammer) but we will be using the flashrom utility!

$ sudo apt-get update
$ sudo apt-get install flashrom -y

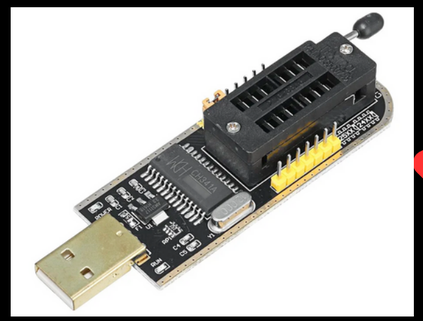
```
noahcooper@noahs-macbook-pro ~
$ flashrom
flashrom v1.5.1 on Darwin 23.6.0 (arm64)
flashrom is free software, get the source code at https://flashrom.org

Please select a programmer with the --programmer parameter.
Valid choices are:
dummy, raiden_debug_spi, ft2232_spi, serprog, buspirate_spi, dediprog,
developerbox, pony_spi, usbblaster_spi, pickit2_spi, ch341a_spi, ch347_spi,
digilent_spi, stlinkv3_spi, dirtyjtag_spi.
```

It’s having an error about not knowing what programmer to use!

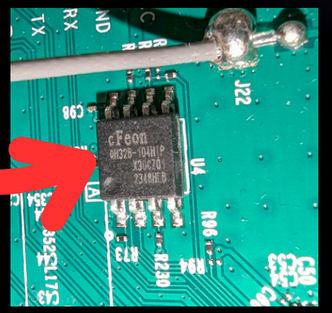
Dumping the firmware

We can specify the programmer ‘ch341a_spi’, and we can specify the chip as we identified it earlier: ‘EN25QH32B’

sudo flashrom -p ch341a_spi -r router.bin -c EN25QH32B



output file



Let’s try it!

Dumping the firmware

LIVE DEMO TIME

Dumping the firmware (if the live demo failed)

We now have a router.bin, which contains the firmware on the chip!

```
root@laptop:/home/l/work# flashrom -p ch341a_spi -r router.bin -c EN25QH32B
flashrom 1.4.0 on Linux 6.12.17-amd64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Found Eon flash chip "EN25QH32B" (4096 kB, SPI) on ch341a_spi.
===
This flash part has status UNTESTED for operations: WP
The test status of this chip may have been updated in the latest development
version of flashrom. If you are running the latest development version,
please email a report to flashrom@flashrom.org if any of the above operation
work correctly for you with this flash chip. Please include the flashrom log
file for all operations you tested (see the man page for details), and menti
which mainboard or programmer you tested in the subject line.
You can also try to follow the instructions here:
https://www.flashrom.org/contrib_howtos/how_to_mark_chip_tested.html
Thanks for your help!
Reading flash... done.
```

What do I do with this?

```
noahcooper@noahs-macbook-pro ~/Downloads/router
$ file router.bin
router.bin: data
```

What do I do with this?

We can use binwalk to look for and extract filesystems inside the firmware dump.

```
noahcooper@noahs-macbook-pro ~/Downloads/router
$ binwalk -e router.bin

                                                                                /Users/noahcooper/Downlo
--------------------------------------------------------------------------------------------------------
DECIMAL                         HEXADECIMAL                     DESCRIPTION
--------------------------------------------------------------------------------------------------------
66048                           0x10200                         LZMA compressed data, properties:
1048576                         0x100000                        SqushFS file system, little endi
                                                                2022-08-16 04:14:58
--------------------------------------------------------------------------------------------------------
[+] Extraction of lzma data at offset 0x10200 completed successfully
[+] Extraction of squashfs data at offset 0x100000 completed successfully
--------------------------------------------------------------------------------------------------------

Analyzed 1 file for 85 file signatures (187 magic patterns) in 296.0 milliseconds
```

Yay! SquashFS file system!

What can I find in this?

```
noahcooper@noahs-macbook-pro ~/Downloads/router/extractions/router.bin.extracted/100000/squashfs-root
$ ls
bin     dev     etc     lib     linuxrc mnt     proc    sbin    sys     usr     var     web
```

Typical linux filesystem, let's check /etc!

```
noahcooper@noahs-macbook-pro ~/Downloads/router/extractions/router.bin.extracted/100000/squashfs-root/etc
$ ls
MT7628_AP_2T2R-4L_V15.BIN  SingleSKU_FCC.dat  fstab       inittab       passwd.bak          resolv.conf
MT7628_EEPROM_20140317.bin TZ                 group       iptables-stop ppp                 samba
RT2860AP.dat               default_config.xml init.d      passwd        reduced_data_model.xml services
```

passwd.bak could be of interest...

```
noahcooper@noahs-macbook-pro ~/Downloads/router/extractions/router.bin.extracted/100000/squashfs-root/etc
$ cat passwd.bak
admin:$1$$iC.dUsGpxNNJGeOm1dFio/:0:0:root:/:/bin/sh
dropbear:x:500:500:dropbear:/var/dropbear:/bin/sh
nobody:*:0:0:nobody:/:/bin/sh
```

A password hash for admin, let's crack it!

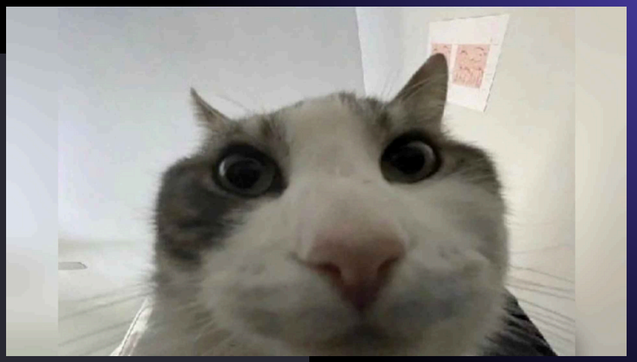
Hash Cracking for fun & profit

Let’s crack that hash with hashcat, hashcat autodetects the hash type for us!

```
noahcooper@noahs-macbook-pro ~/Downloads/router
$ hashcat hash ~/Documents/Wordlists/SecLists/Passwords/Leaked-Databases/rockyou.txt
hashcat (v6.2.6) starting in autodetect mode

* Device #2: Apple's OpenCL drivers (GPU) are known to be unreliable.
             You have been warned.

METAL API (Metal 343.21)
========================
* Device #1: Apple M1 Max, 24512/49152 MB, 32MCU

OpenCL API (OpenCL 1.2 (Jul 19 2024 22:07:05)) - Platform #1 [Apple]
===================================================================
* Device #2: Apple M1 Max, skipped

Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

500 | md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5) | Operating System
```

Hash Cracking for fun & profit

After some time, it’s cracked!



```
$1$$iC.dUsGpxNNJGeOm1dFio/:1234

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: $1$$iC.dUsGpxNNJGeOm1dFio/
Time.Started.....: Tue Mar 18 23:12:15 2025 (1 sec)
Time.Estimated...: Tue Mar 18 23:12:16 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/Users/noahcooper/Documents/Wordlists/SecLists/Passwords/Leaked-Databases/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1415.5 kH/s (9.74ms) @ Accel:128 Loops:62 Thr:64 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 262144/14344384 (1.83%)
Rejected.........: 0/262144 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:992-1000
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → rebel91
Hardware.Mon.SMC.: Fan0: 0%, Fan1: 0%
Hardware.Mon.#1..: Util: 91%
```

We can see the password for ‘admin’ is ‘1234’ (very secure)

Hash Cracking for fun & profit

If the serial console **was** password protected, this would allow us to login and get a shell.

What Next?

You now have a clean dump of the firmware, as well as a root shell on the device!

Have a poke around the filesystem for secrets leftover (api keys, passwords, etc..), reverse engineer system service binaries, manipulate the environment to test exploit ideas, etc..

Conclusion

Go buy the cheapest and shittiest IOT e-waste you can find and get hacking!

Questions + Kahoot quiz time >>>>

Kahoot Quiz

Prizes:

#1 Your choice of either a CH341a flash dumper (same model used in the talk) OR a UTS:CSEC bucket hat

#2 Box of chocolates

#3 Bag of lollies