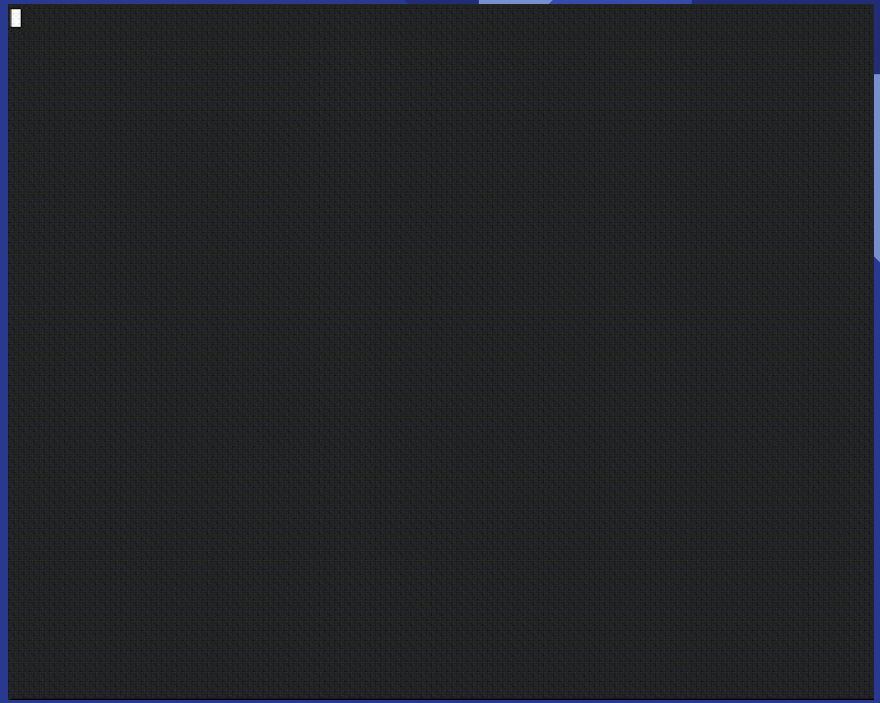
Securing devices or profits?

Examining the device security of a network appliance vendor



Hal Martin

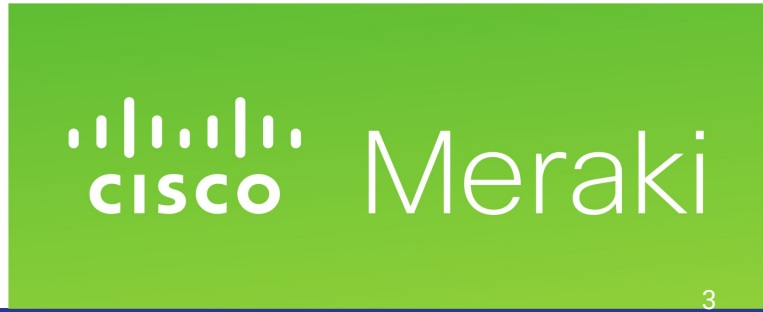
OrangeCon 2024

Agenda

- Who are Meraki?
- Motivation
- Bootloaders
- Target devices
- ???
- Profit

Who are Meraki?

- Founded in 2006
- Acquired by Cisco in 2012
- Sell cloud managed networking devices (routers, switches, APs, cameras, sensors)



Meraki business model

You: Managing & monitoring network assets is hard 😫

Meraki: We'll punch through your NAT with a tunnel to manage them from "the cloud"

You: buy the hardware 💰💰

You: buy a **license** for the hardware 💰

Meraki: provide a dashboard to manage your network

You, after solving your network asset management problem

“Cisco Meraki may find it necessary to discontinue products for a number of reasons, including product line enhancements, market demand, technology innovation, or if the product simply matures over time and needs to be replaced by something functionally richer.”

https://documentation.meraki.com/General_Administration/Other_Topics/Meraki_End-of-Life_(EOL)_Products_and_Dates

Forget the cloud?

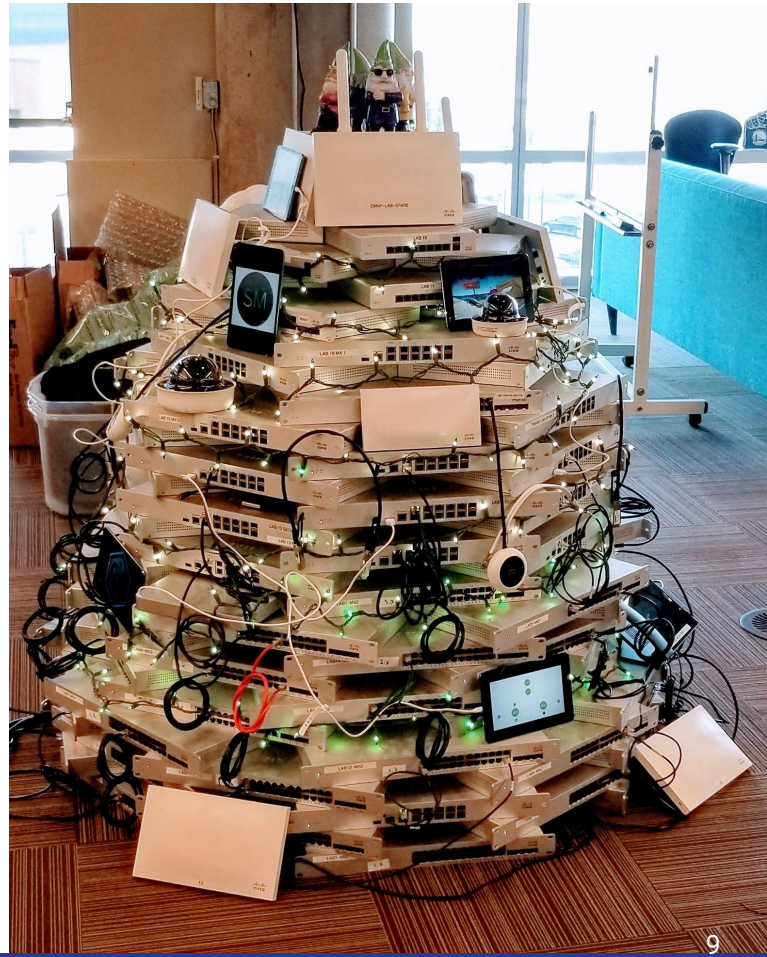
- Manage it yourself?
  - You can only manage devices from the Meraki dashboard (nice SPOF!)

    “Meraki switches are only manageable through the dashboard and you need a license for that.”

https://community.meraki.com/t5/Switching/Managing-Switch-without-License/m-p/108768/highlight/true#M7927

tl;dr

- Late-stage capitalism: rent your devices
  - An expensive lease
- Meraki sales: deprecating existing products 🤤🤤
  - e-waste? 
- Secondary market for resale? lol. lmao.
  - “Claimed” devices cannot be reset and managed by a new owner:
    “If you can claim it into a dashboard (which is only possible after the previous owner unclaimed it)”



https://community.meraki.com/t5/Switching/Managing-Switch-without-License/m-p/44556/highlight/true#M3719
https://community.meraki.com/t5/Off-the-Stack/It-s-beginning-to-look-a-lot-like-Switchmas/m-p/34263

Motivation

- Bought my first claimed Meraki switch in 2019
  - Wanted 10G networking on the cheap
- Learned more about their practices and business model
  - e-waste? Yes, please!
- Develop and maintain a FLOSS firmware for older switch models
- Started researching the security of their products

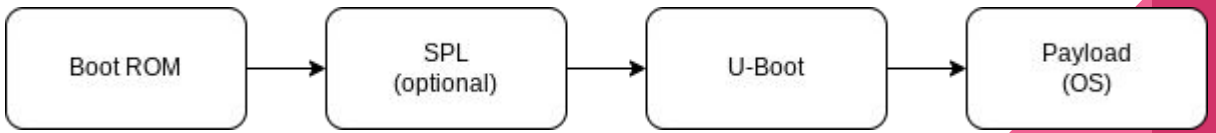
Motivation: GPL Infringement

“Unfortunately, GPL source code for the MS42 is no longer available.
We provide GPL source code for up to 3 years after a product's end-of-sale announcement.
The announced end-of-sale date for the MS42 was Apr 2014”

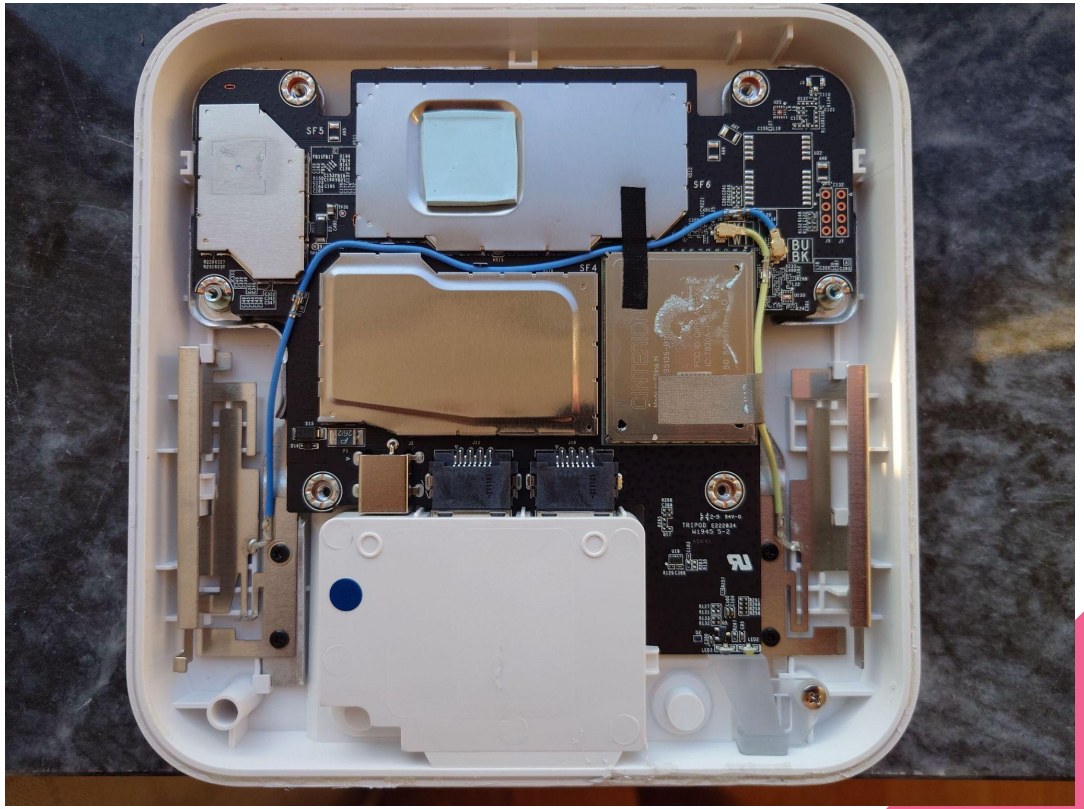
This was their reply to my first GPL request. It is prima facie GPL infringement as they **still** released firmware updates for the MS42 at the time of this request

Bootloaders: U-Boot?

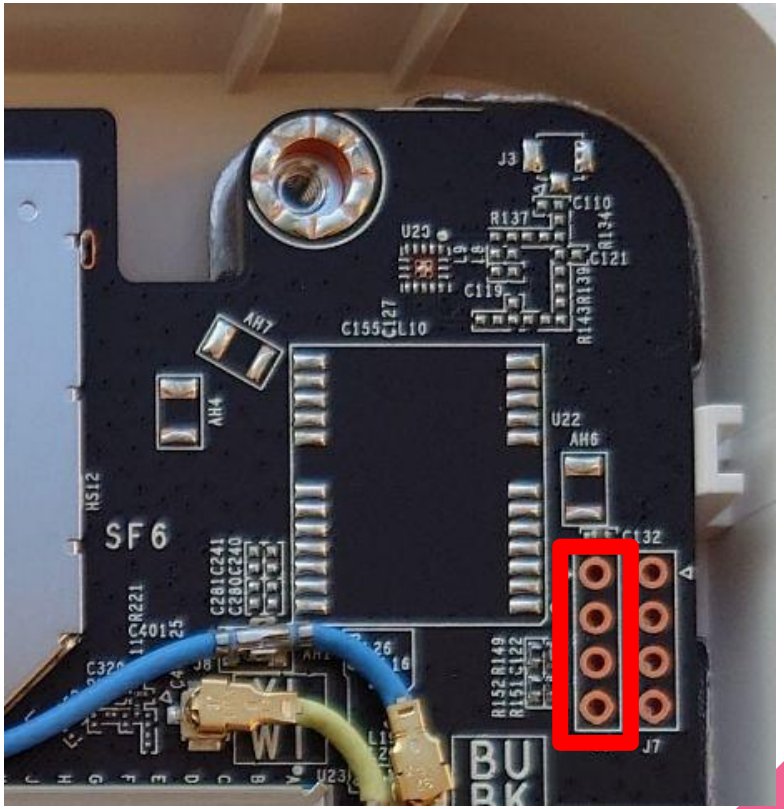
- Bootloader targeting embedded devices
  - First released in 1999
  - Supports many architectures
- (Usually) loaded by SoC Boot ROM
- May initialise hardware
  - DRAM training, IO (e.g. networking, UART), storage (NOR/NAND/MMC/SATA/USB)
- (Usually) includes command line with simple scripting
- Conceptually similar to BIOS/UEFI for embedded systems



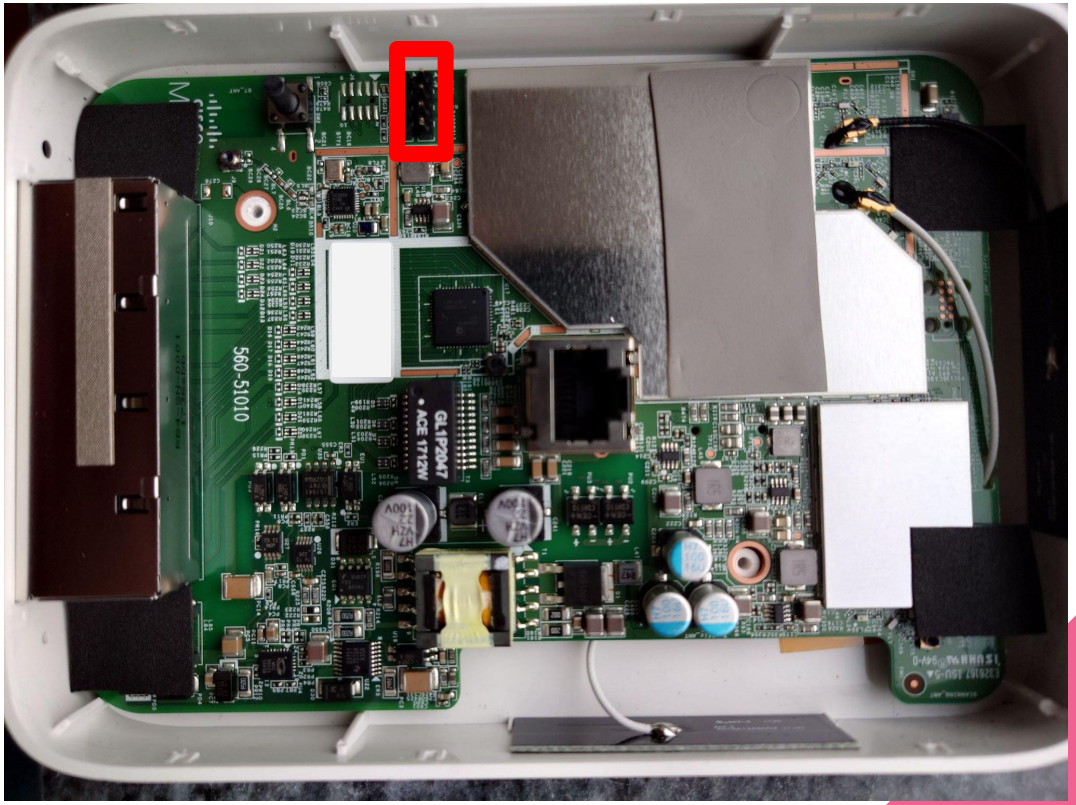
Where is UART?



Where is UART?



Where is UART?



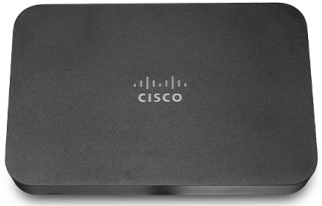
Reverse engineering resources

- U-Boot is GPL licensed
- Meraki take 12+ months on average to provide source code

But!

- NAND is unencrypted 😉

Target devices



Wired routers

- Z3
- Z3C

Cellular gateways

- MG21
- MG41

Common bootloader weaknesses

- Interrupt boot and get a prompt
- Environment stored on or sourced from NVRAM
- Compile/flash the bootloader from source

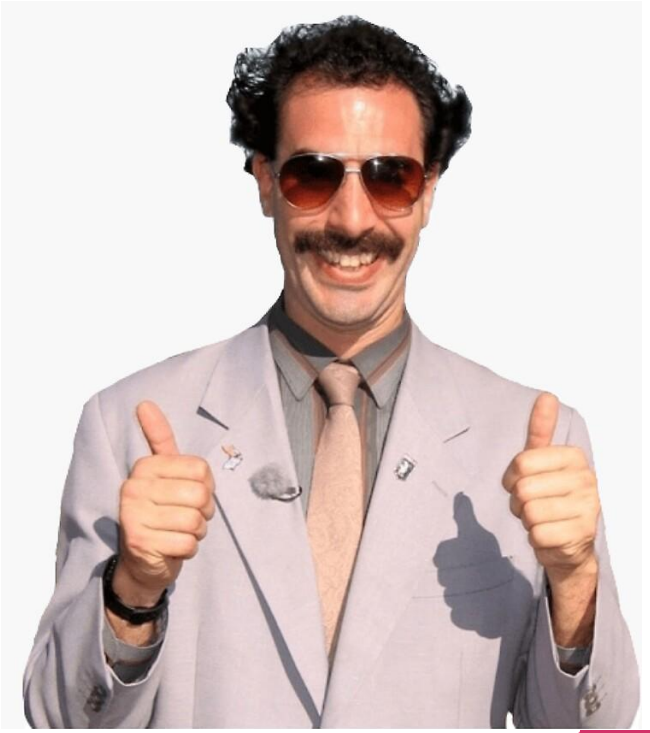
Method	
Boot delay	
Modify the environment	
Recompile and overwrite the bootloader	

Bootloader command line

CONFIG_CMDLINE=y

CONFIG_HUSH_PARSER=y

CONFIG_SYS_PROMPT="=> "



Autoboot

CONFIG_AUTOBOOT=y

CONFIG_AUTOBOOT_KEYED=y

CONFIG_AUTOBOOT_PROMPT="Autoboot in %d secon

CONFIG_AUTOBOOT_ENCRYPTION is not set

CONFIG_AUTOBOOT_DELAY_STR=""

CONFIG_AUTOBOOT_STOP_STR="xyzzy"

CONFIG_AUTOBOOT_KEYED_CTRLC is not set

Boot delay

#

Environment

#

CONFIG_BOOTDELAY=0

u-boot evolution

pre-2017	2017+
#define CONFIG_AUTOBOOT_STOP_STR "xyzzy"	CONFIG_BOOTDELAY=0
Allows interrupting autoboot with the string “xyzzy” on UART	Can no longer interrupt boot

```
8733823b 62 6f 6f       ds          "bootstopkey"
         74 73 74
         6f 70 6b ...
87338247 78             ??          78h    x
87338248 79             ??          79h    y
87338249 7a             ??          7Ah    z
8733824a 7a             ??          7Ah    z
8733824b 79             ??          79h    y
8733824c 00             ??          00h
```

CONFIG_ENV_IS_NOWHERE

- The environment is compiled into u-boot
- No possibility to modify, or persist changes

U-Boot situation

Method	**Target device**
Boot delay	❌ removed
Modify the environment	❌ compiled into u-boot
Recompile and overwrite the bootloader	🤔

Secure boot?

```
ubi0: max/mean erase counter: 78/30, WL threshold: 4096, image sequence number: 578697608
ubi0: available PEBs: 397, total reserved PEBs: 499, PEBs reserved for bad PEB handling: 20


Secure boot enabled.

Read 0 bytes from volume part.safe to 84000000
No size specified -> Using max size (25628672)
Valid image
## Loading kernel from FIT Image at 84000028 ...
```

U-Boot situation

Method	**Target device**
Boot delay	❌ removed
Modify the environment	❌ compiled into u-boot
Recompile and overwrite the bootloader	❌ secure boot

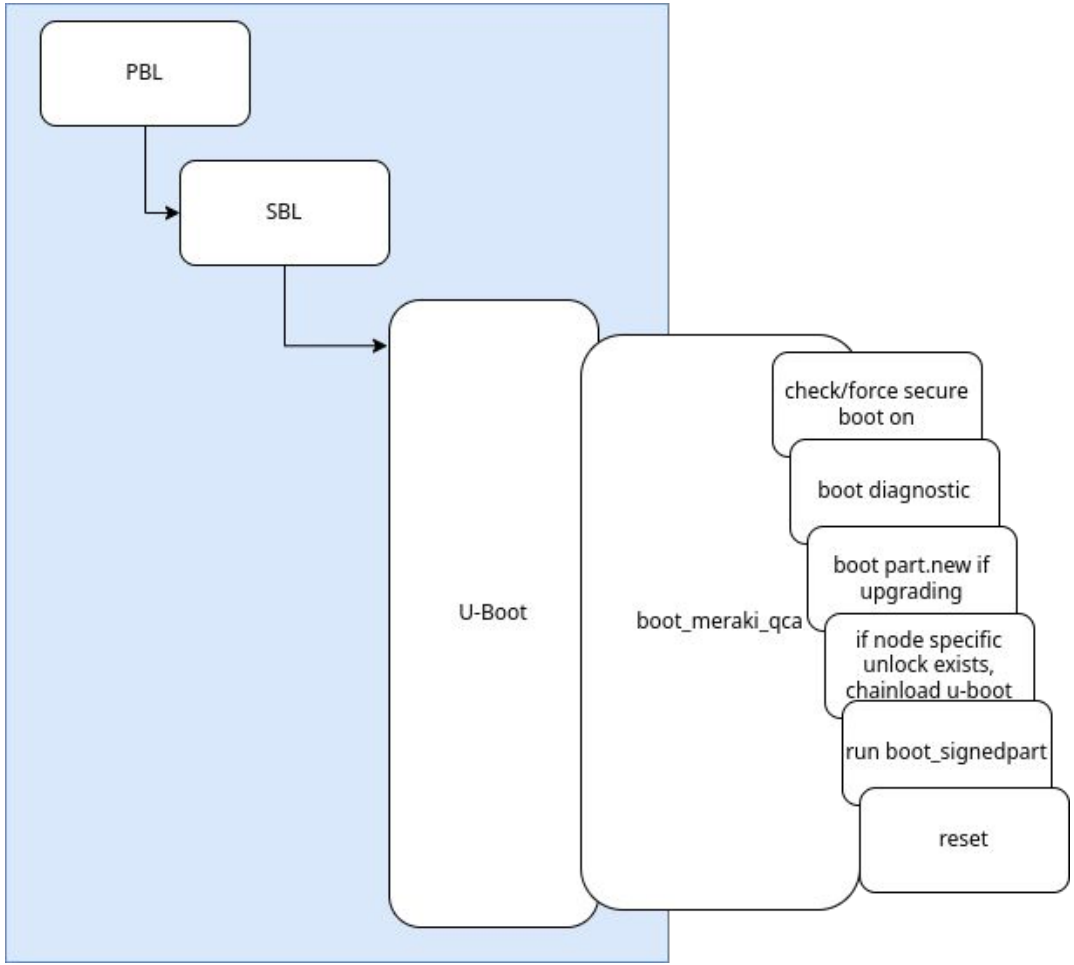
U-Boot verified boot

- U-Boot supports verifying signatures in FIT images
- Tampering with signed FIT images is not possible
  - Configurations are signed
  - Configuration contains hash of kernel/initrd/fdt
  - Not possible to swap out images in the configuration or add more configurations
- U-Boot defaults to checking signatures if `verify` is unset in environment
- Vendors use U-Boot's verified boot

Vendors use U-Boot verified boot, right?

```
=> printenv
baudrate=115200
boot_part=ubi read $loadaddr $part; bootm $loadaddr#$itb_config
boot_signedpart=ubi read $loadaddr $part; validate $loadaddr && bootm $imgaddr#$itb_config
bootcmd=run meraki_boot
bootdelay=5
bootkernel1=setenv part part.safe; run boot_part
bootkernel2=setenv part part.old; run boot_part
```

```
itb_config=config@1
loadaddr=0x84000000
machid=8010001
meraki_boot=run set_bootargs; run set_ubi; boot_meraki_qca; run bootkernel1; run bootkernel2
part=part.safe
scrloadaddr=0x81000000
scrname=boot.scr
set_bootargs=setenv bootargs loader=u-boot maxcpus=1
set_ubi=setenv mtdids nand0=nand0; setenv mtdparts mtdparts=nand0:0x7000000@0xc00000(ubi); ubi part ubi
```



No obvious security flaws here

How about now?

```
25  product_number = get_meraki_product_id();
26  if (product_number < 0x11) {
27    argc = 1;
28    argv = DAT_87301a34 & 1 << (product_number & 0xff);
29    if (argv != 0) {
30      return 0;
31    }
32  }
33  FUN_87331fcc(DAT_87301a38,extraout_r1,argc,argv);
34  product_number = 0;
35  uVar5 = 0;
36  piVar2 = (int *)FUN_8730089c(8,7,(uint *)0x0,0,(uint *)((int)&uStack_c + 3),1);
37  if (piVar2 == (int *)0x0) {
38    product_number = uStack_c >> 0x18;
39    piVar2 = extraout_r1_00;
40    if (product_number != 1) {
41      FUN_87331fcc(DAT_87301a3c,extraout_r1_00,uVar5,product_number);
42      piVar2 = extraout_r1_01;
43      uVar6 = uVar5;
44      goto LAB_873018b0;
45    }
46  }
47  else {
48    FUN_87331fcc(DAT_87301a40,piVar2,uVar5,product_number);
```

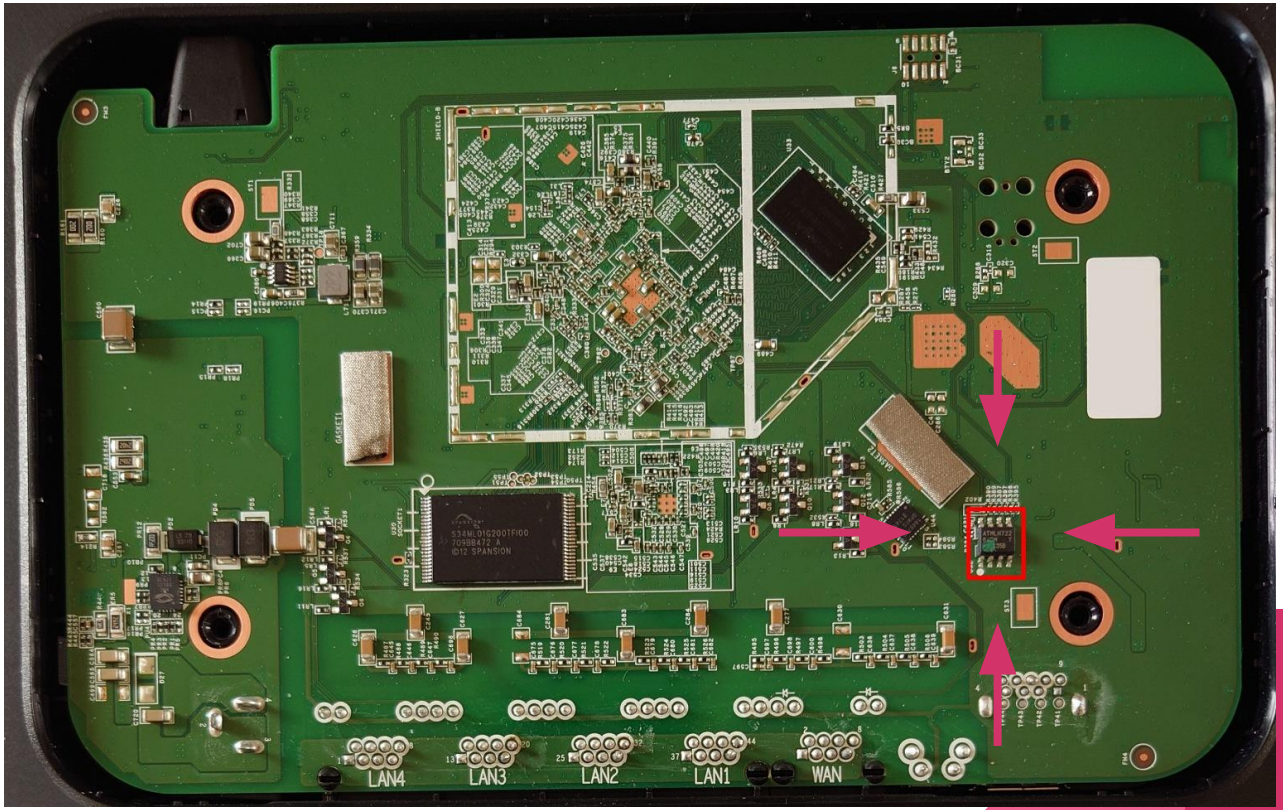
```
static int do_meraki_qca_boot(cmd_tbl_t * cmdtp, int flag,
int argc, char * const argv[])
{
  /* unsupported boards */
  switch(get_meraki_product_id()) {
    case MERAKI_BOARD_STINKBUG:
    case MERAKI_BOARD_LADYBUG:
    case MERAKI_BOARD_NOISY_CRICKET:
    case MERAKI_BOARD_YOWIE:
    case MERAKI_BOARD_BIGFOOT:
    case MERAKI_BOARD_SASQUATCH:
    case MERAKI_BOARD_WOOKIE:
      return 0;
    default:
      break;
  }

  /* Check 0: check/force secure boot on */
  force_secboot();
```

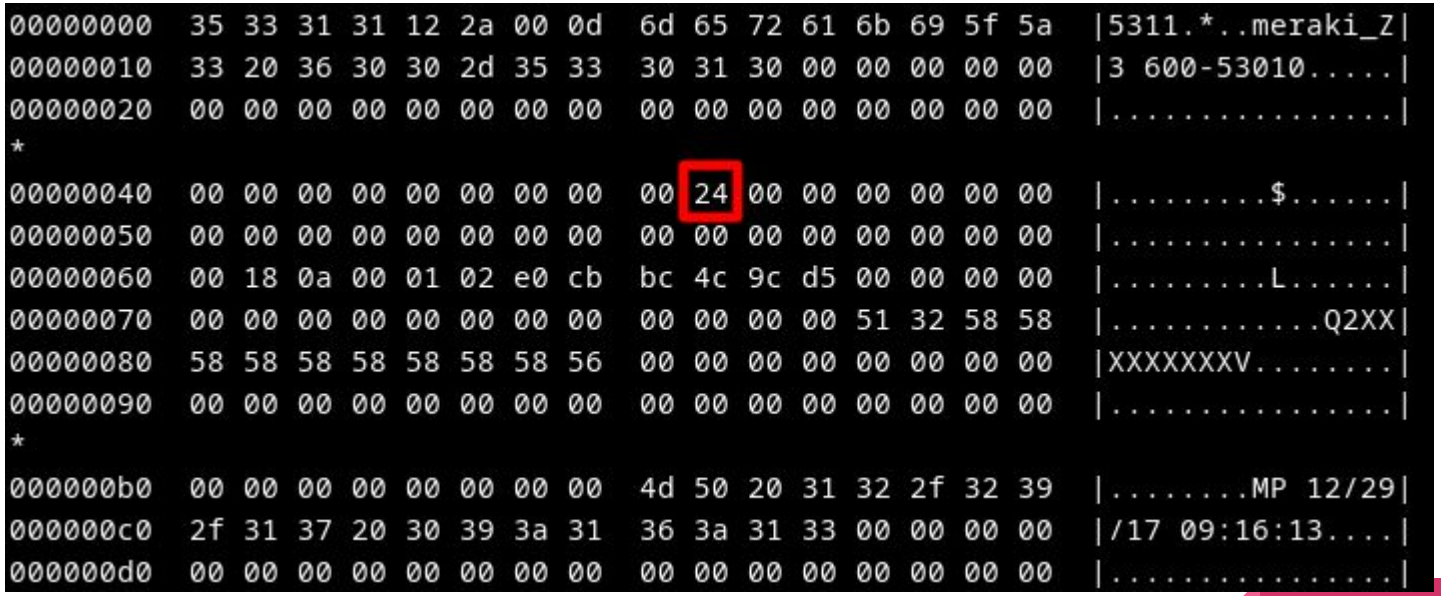
Meraki provided the Z3 U-Boot source code after a delay of more than 6 months

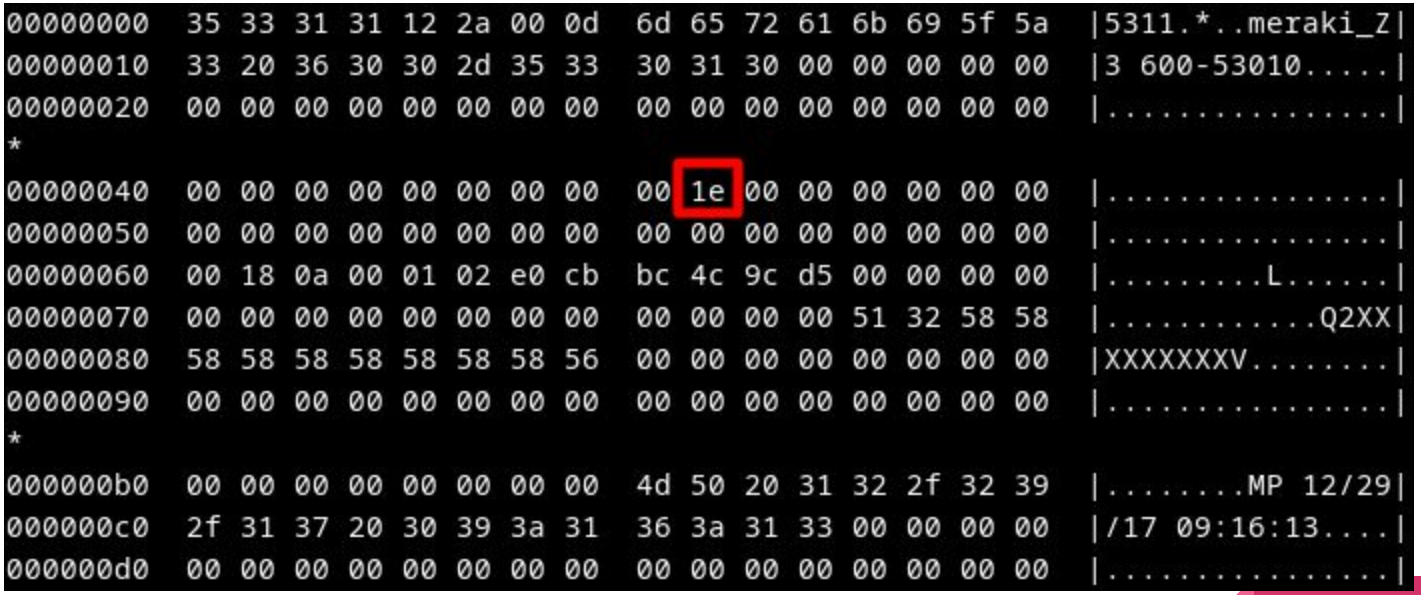
Product ID

```
static const struct product_map_entry product_map[] = {
    /* BOARD=insect */
    { "meraki_Stinkbug", 30, "STINKBUG # ", MERAKI_BOARD_STINKBUG, "config@1" },
    { "meraki_Ladybug", 31, "LADYBUG # ", MERAKI_BOARD_LADYBUG,   "config@3" },
    { "meraki_Noisy_Cricket", 32, "NOISY CRICKET # ", MERAKI_BOARD_NOISY_CRICKET, "config@2" },
    { "meraki_Maggot", 37, "MAGGOT # ", MERAKI_BOARD_MAGGOT, "config@4" },
    { "meraki_Dungbeetle_Omni", 38, "DUNGBEETLE OMNI # ", MERAKI_BOARD_DUNGBEETLE_OMNI, "config@5" },
    { "meraki_Dungbeetle_Patch", 39, "DUNGBEETLE PATCH # ", MERAKI_BOARD_DUNGBEETLE_PATCH, "config@6" },
    { "meraki_Grub", 44, "GRUB # ", MERAKI_BOARD_GRUB, "config@4" },
    { "meraki_Toe_biter_Omni", 45, "TOE_BITER OMNI # ", MERAKI_BOARD_TOE_BITER_OMNI, "config@5" },
    { "meraki_Toe_biter_Patch", 46, "TOE_BITER PATCH # ", MERAKI_BOARD_TOE_BITER_PATCH, "config@6" },

    /* BOARD=wired-arm-qca */
    { "meraki_Fuzzy_Cricket", 36, "FUZZY CRICKET # ", MERAKI_BOARD_FUZZY_CRICKET, "config@1" },
    { "meraki_Fairyfly", 43, "FAIRYFLY # ", MERAKI_BOARD_FAIRYFLY, "config@2" },
    { "meraki_Heart_of_Gold", 42, "HOG # ", MERAKI_BOARD_HEART_OF_GOLD, "config@3" },

    { NULL, MERAKI_BOARD_UNKNOWN },
};
```

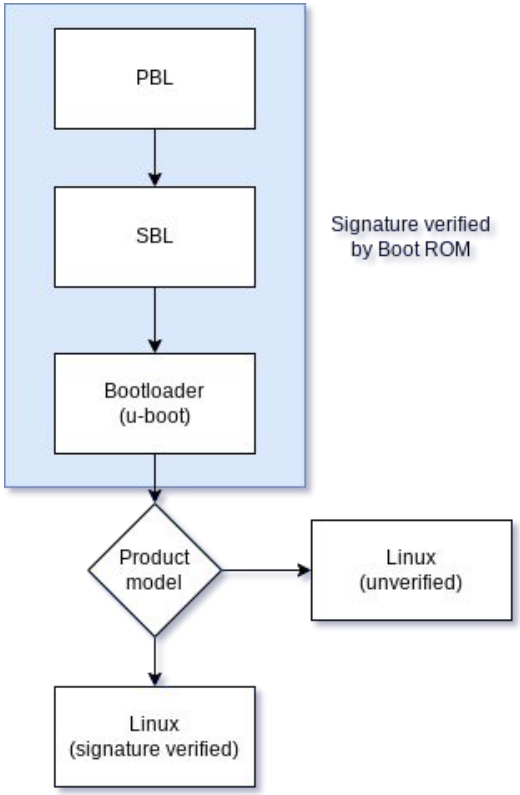
Where is the product ID?




00000000 35 33 31 31 12 2a 00 0d 6d 65 72 61 6b 69 5f 5a |5311.*..meraki_Z|
00000010 33 20 36 30 30 2d 35 33 30 31 30 00 00 00 00 00 |3 600-53010.....|
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
00000040 00 00 00 00 00 00 00 00 00 24 00 00 00 00 00 00 |.........$......|
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000060 00 18 0a 00 01 02 e0 cb bc 4c 9c d5 00 00 00 00 |.........L......|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 51 32 58 58 |............Q2XX|
00000080 58 58 58 58 58 58 58 56 00 00 00 00 00 00 00 00 |XXXXXXXV........|
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
000000b0 00 00 00 00 00 00 00 00 4d 50 20 31 32 2f 32 39 |........MP 12/29|
000000c0 2f 31 37 20 30 39 3a 31 36 3a 31 33 00 00 00 00 |/17 09:16:13....|
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|


00000000 35 33 31 31 12 2a 00 0d 6d 65 72 61 6b 69 5f 5a |5311.*..meraki_Z|
00000010 33 20 36 30 30 2d 35 33 30 31 30 00 00 00 00 00 |3 600-53010.....|
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
00000040 00 00 00 00 00 00 00 00 00 1e 00 00 00 00 00 00 |................|
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000060 00 18 0a 00 01 02 e0 cb bc 4c 9c d5 00 00 00 00 |.........L......|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 51 32 58 58 |............Q2XX|
00000080 58 58 58 58 58 58 58 56 00 00 00 00 00 00 00 00 |XXXXXXXV........|
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
000000b0 00 00 00 00 00 00 00 00 4d 50 20 31 32 2f 32 39 |........MP 12/29|
000000c0 2f 31 37 20 30 39 3a 31 36 3a 31 33 00 00 00 00 |/17 09:16:13....|
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|

Z3 (IPQ4029)




```
U-Boot 2017.07-RELEASE-g39cabb9bf3 (May 24 2018
- 14:07:32 -0700)

DRAM:  242 MiB
machid : 0x8010001
Product: meraki Fuzzy Cricket
NAND:  ONFI device found
ID = 1d80f101
Vendor = 1
Device = f1
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)


Secure boot enabled.

Read 0 bytes from volume part.safe to 84000000
No size specified -> Using max size (16584704)
Valid image
## Loading kernel from FIT Image at 84000028
...
```

```
U-Boot 2017.07-RELEASE-g39cabb9bf3 (May 24 2018
- 14:07:32 -0700)

DRAM:  242 MiB
machid : 0x8010001
Product: meraki Stinkbug
NAND:  ONFI device found
ID = 1d80f101
Vendor = 1
Device = f1
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 84000000
No size specified -> Using max size (16584704)
Wrong Image Format for bootm command
ERROR: can't get kernel image!
Read 0 bytes from volume part.old to 84000000
No size specified -> Using max size (16547840)
Wrong Image Format for bootm command
ERROR: can't get kernel image!
resetting ...
```

Situation

1. Secure boot is enabled
2. Bootloader chain is signed
3. U-Boot supports booting unsigned images
4. Product ID is stored on external EEPROM with no encryption or signature
    a. Devices manufactured after 2017 have two copies in EEPROM
    b. Neither is signed; U-Boot doesn't even compare them
5. Product ID -> change to device without secure boot
6. U-Boot stops verifying signature of payload

Next steps

1. Compile U-Boot from GPL source code
2. Package into flattened device tree (FIT) image
3. Put into ubivol “part.safe”
4. U-Boot will now boot this without verification
5. Unlocked bootloader!

Can Meraki patch it?

- Meraki EoL'd 802.11ac APs without secure boot
- U-Boot builds after 2019 don't support these devices, vulnerable code path removed

…

- Multiple devices use the same signing certificate

```
Version Rollback Feature Disabled
```

- There is no anti-rollback protection!

IPQ40xx devices affected

Device	Vulnerable	Device	Vulnerable
Z3	✅	MG21	✅
Z3C	✅	MG41	❌

Meraki Go

- Meraki, but cheaper
- Devices managed through an app
- No Dashboard
- Device still connects to the cloud, Meraki push a config

“As we look to the future, we will no longer be developing new Meraki Go products. To better serve your networking needs, we will continue innovating our Cisco Meraki portfolio to make it even more accessible to small businesses.”

https://community.meraki.com/t5/Announcements/Meraki-Go-End-of-Sale-April-2025/ba-p/233755

Target devices - Meraki Go

802.11ac APs

- GR10

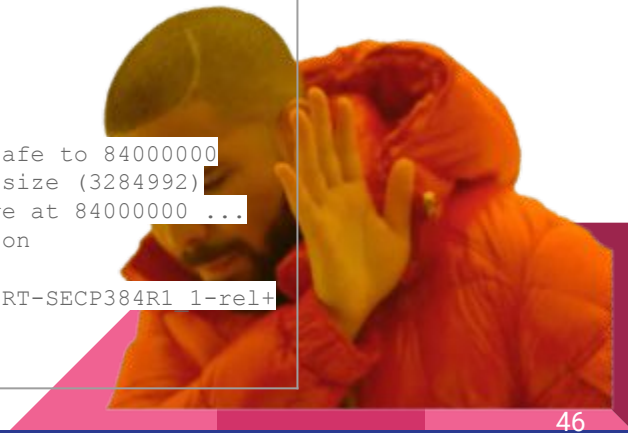
Wired routers

- GX20

EEPROM Product ID

```
U-Boot 2012.07-g03cdfe19e00f [local,local] (Aug 29
2017 - 11:59:45)

DRAM:  498 MiB
machid : 0x8010001
Product: meraki_Fairyfly
NAND:  ONFI device found
ID = 1d80f101
Vendor = 1
Device = f1
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 84000000
```

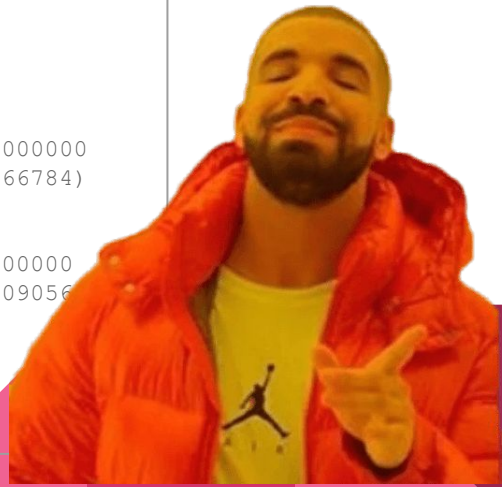
```
U-Boot 2012.07-g03cdfe19e00f [local,local] (Aug 29
2017 - 11:59:45)

DRAM:  498 MiB
machid : 0x8010001
ERROR: Unknown board
NAND:  ONFI device found
ID = 1d80f101
Vendor = 1
Device = f1
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 84000000
No size specified -> Using max size (3284992)
## Booting kernel from FIT Image at 84000000 ...
   Using 'config@1' configuration
   Verifying Hash Integrity ...
sha384,secp384r1:wired-arm-qca-RT-SECP384R1 1-rel+
OK
```



Meraki Go: replace U-Boot?

```
U-Boot 2012.07-g03cdfe19e00f [local,local] (Aug 29
2017 - 11:59:45)

DRAM:  498 MiB
machid : 0x8010001
Product: meraki_Fairyfly
NAND:  ONFI device found
ID = 1d80f101
Vendor = 1
Device = f1
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 84000000
```

```
U-Boot 2017.07-RELEASE-g78ed34f31579 (Sep 29 2017
- 07:43:44 -0700)

DRAM:  242 MiB
machid : 0x8010001
Product: meraki_Stinkbug
NAND:  ONFI device found
128 MiB
Using default environment

In:    serial
Out:   serial
Err:   serial
machid: 8010001
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 84000000
No size specified -> Using max size (14966784)
Wrong Image Format for bootm command
ERROR: can't get kernel image!
Read 0 bytes from volume part.old to 84000000
No size specified -> Using max size (11309056
Wrong Image Format for bootm command
ERROR: can't get kernel image!
resetting ...
```



Meraki Go devices affected

Device	Vulnerable	Device	Vulnerable
GX20 (Z3)	✅	GR10 (MR20)	✅
GX50 (MX67)	❌	GR60 (MR70)	(probably)

Target devices

802.11ax APs

- MR36

WiFi 6 u-boot

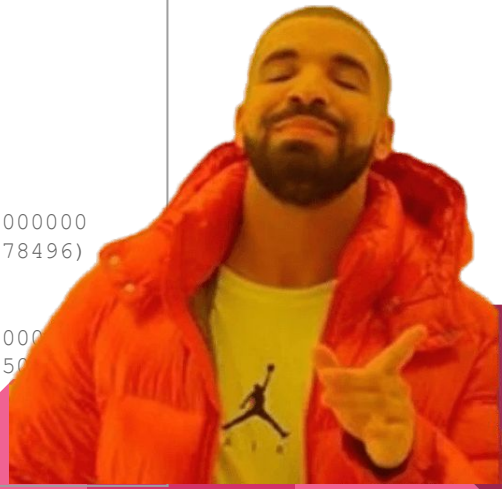
```
uStack_c = param_2;
product_number = FUN_4a902c0c();
if (product_number < 0x10) {
  param_3 = 1;
  param_4 = 1 << (product_number & 0xff) & 0xb40e;
  if (param_4 != 0) {
    return 0;
  }
}
FUN_4a90b980(DAT_4a902520,DAT_4a90251c);
FUN_4a93a868(DAT_4a902524,extraout_r1,param_3,param_4);
ppuVar8 = (uint **)0x1;
puVar7 = (uint *)((int)&uStack_c + 3);
uVar9 = FUN_4a900e18(8,7,puVar7,1);
piVar3 = (int *)((ulonglong)uVar9 >> 0x20);
if ((int *)uVar9 != (int *)0x0) {
  FUN_4a93a868(DAT_4a90256c,(int *)uVar9,puVar7,ppuVar8);
  goto LAB_4a902448;
}
```

No source code, Meraki have not provided it

MR36 (IPQ807x)

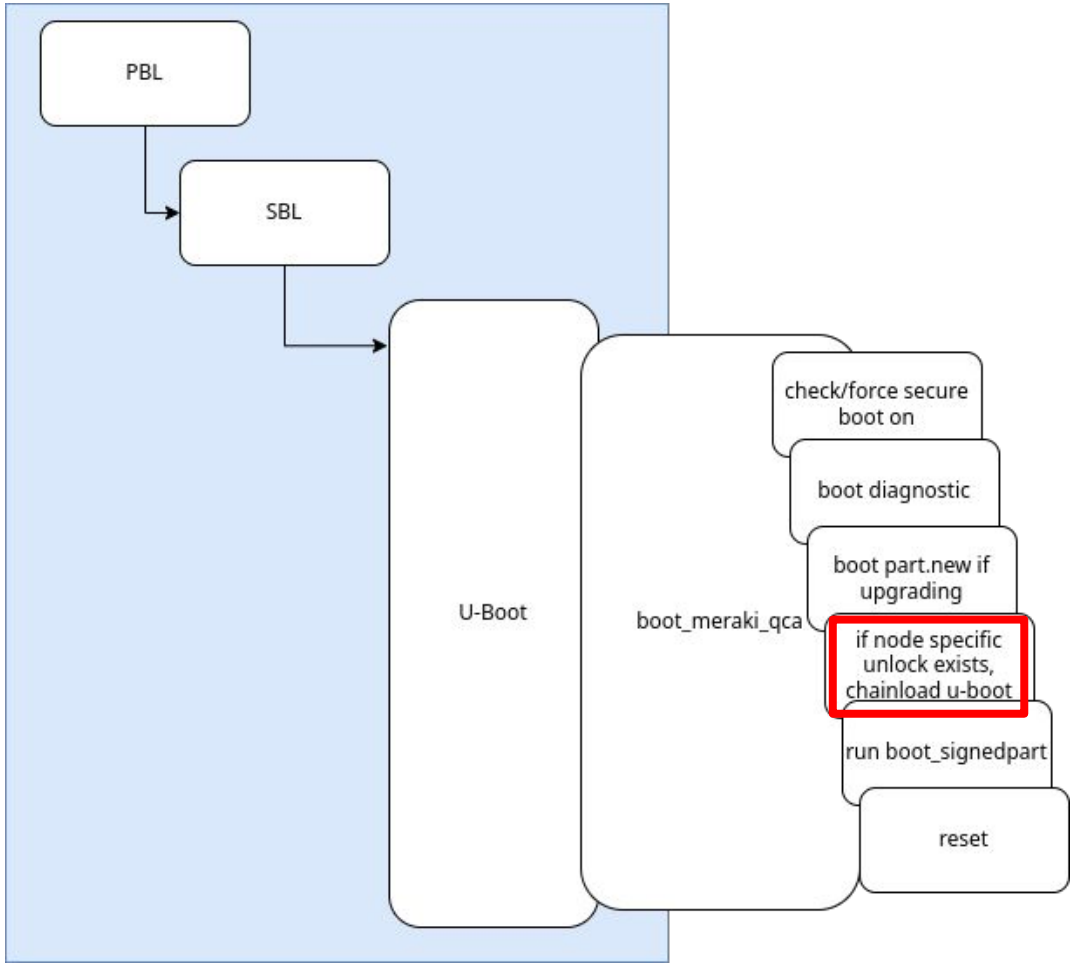
```
U-Boot 2018.01-RELEASE-gb0bd058b3f (Nov 25 2019 -
16:41:18 -0800)

DRAM:  1020 MiB
Setting bus to 0
Valid chip addresses: 56
Meraki Product (major num): 56
NAND:  ONFI device found
256 MiB
Using default environment

In:    serial@78B3000
Out:   serial@78B3000
Err:   serial@78B3000
Device Tree: QCA, IPQ807x-AC01
machid: 8010009
ubi0: (removed for space)


Secure boot enabled.

Read 0 bytes from volume part.safe to 50000000
No size specified -> Using max size (28778496)
Validating 1b71900 bytes @ addr 50000000
Valid image
```

```
U-Boot 2018.01-RELEASE-gb0bd058b3f (Nov 25 2019 -
16:41:18 -0800)

DRAM:  1020 MiB
Setting bus to 0
Valid chip addresses: 56
Meraki Product (major num): 1
NAND:  ONFI device found
256 MiB
Using default environment

In:    serial@78B3000
Out:   serial@78B3000
Err:   serial@78B3000
Device Tree: QCA, IPQ807x-AC01
machid: 8010009
ubi0: (removed for space)
Read 0 bytes from volume part.safe to 50000000
No size specified -> Using max size (28778496)
Wrong Image Format for bootm command
ERROR: can't get kernel image!
Read 0 bytes from volume part.old to 500000
No size specified -> Using max size (26050
```



Meraki 802.11ax devices affected

Device	Vulnerable
MR36	✅
MR44	(unverified; probably)
MR46	(unverified; probably)
MR56	(unverified; probably)



Devices support bootloader unlocking…

But Meraki have zero incentive to support this

Regulatory solution wen?

Shipping a secure device checklist

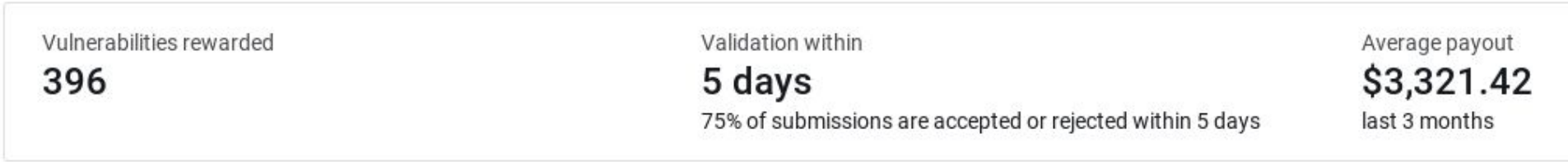
- Disable u-boot command line
  - CONFIG_CMDLINE=n
- Use encrypted autoboot option
  - CONFIG_AUTOBOOT_ENCRYPTION
- Compile the environment into u-boot
  - CONFIG_ENV_IS_NOWHERE=y
- Use anti-rollback features!
  - Don’t allow the user to flash an older (vulnerable) version of your bootloader
- Don’t ship encryption keys in your firmware
  - Use secure storage (enclave/TZ) in the SoC

B2B due diligence

- Ask your vendor for their GPL source code
- Source code is easier to scan for vulnerabilities than binaries
- Your vendor might not even have it themselves (subcontracting)
  - Gives you insight into how your vendor manages updates and patching (or doesn't)
- If your vendor won't comply with GPL license, what other things are they infringing?

Responsible disclosure?

- Meraki have a bug bounty operated by BugCrowd
- Both Meraki and BugCrowd have had issues with their disclosure programs
  - Ignored/invalid reports, fixing the issue without a pay-out
- Avoiding e-waste is more valuable than their pay-out (max $10k)



Responsible disclosure: NDA

By participating in the Program, investigating a potential vulnerability, or submitting a vulnerability, you affirm that you have not disclosed and agree that you will not disclose the vulnerability to anyone other than Cisco Meraki.

Absent Cisco Meraki's prior written consent, any disclosure outside of this process would violate this Agreement. You agree that money damages may not be a sufficient remedy for a breach of this paragraph by you and that Cisco Meraki will be entitled to specific performance as a remedy for any such breach.

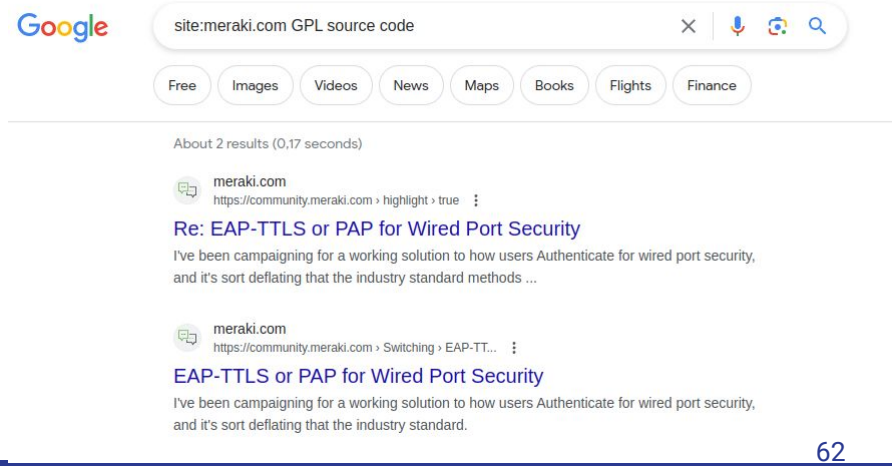
https://bugcrowd.com/engagements/ciscomeraki

Responsible disclosure: Not in-scope

- Deficiencies in a security feature in an on-prem product.
  - Secure boot is a security feature, the devices are on-prem
- Any attack which renders the device permanently inoperable
  - Devices will not boot their signed firmware release after modification
- Any vulnerability not present in the most recent beta firmware of a product
  - The most recent U-Boot release **does** remove the exploit, but you can always **roll-back**
- Any hardware bugs which require a debugger to recreate
  - Hardware programming tools are required to reprogram the device

Meraki GPL compliance

- It’s not good:
  - No written offer for source code with their products
  - No documented way to request GPL source code
  - Requests take years to “process”
- If you contributed to
  - Linux
  - U-Boot
  - Busybox
- Please get in touch!

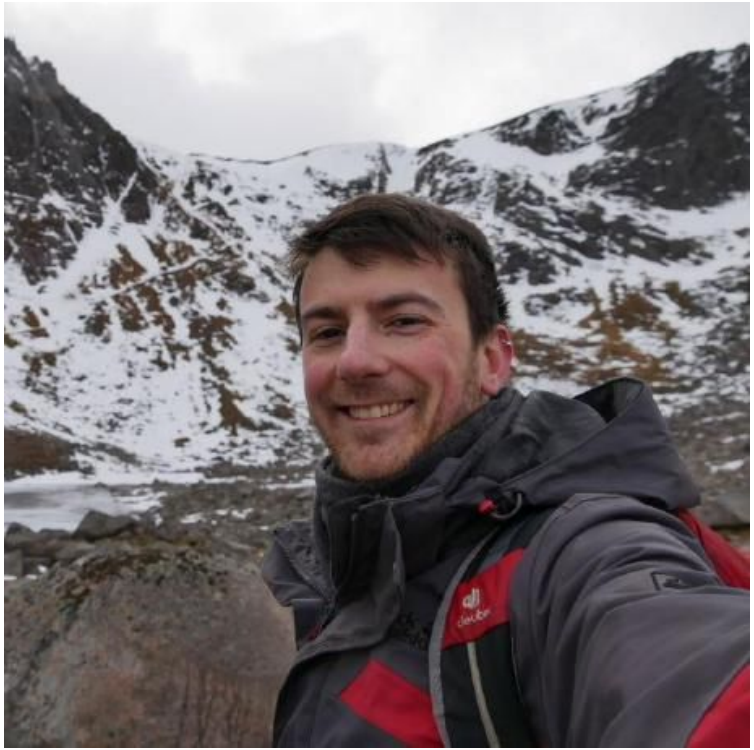


Defeating Planned Obsolescence for Cisco Meraki Switches

Thank you!

- Thanks to OrangeCon for providing me a platform
- Holding vendors accountable for software licenses they use
- Considering user freedom when you ship a device with secure boot

Questions?



GitHub: halmartin

Email: halmartin@gmail.com

Website: watchmysys.com

Fediverse: @hal9000@infosec.exchange