


# SBOMs

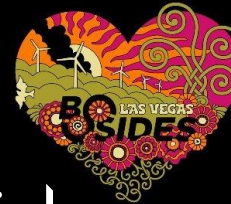
## The right way

---

Michael Messner, Benedikt Kühne

 <https://github.com/e-m-b-a>





RC3  
2021  
NOW  
HERE

 **black hat**<sup>®</sup>

**D3FC0N**

# id

---





Benedikt and Mike

Penetration tester  
Focused on products  
Firmware analysis  
Hardware analysis  
Siemens Energy



 **black hat**<sup>®</sup>  
MIDDLE EAST AND AFRICA



 <https://github.com/e-m-b-a>  
 @securefirmware  
 @securefirmware.bsky.social  
 <https://infosec.exchange/>  
@securefirmware



## EMBArk User groups

---

- # Hobbyist, Kiddies, ...
- # The pentester
- # The security researcher
- # The research team
- # The vendor
- # The developer
- # The SBOM only guy
- # The product security guys



## EMBArk User groups

---

- # Hobbyist, Kiddies, ...
- # The pentester
- # The security researcher
- # The research team
- # The vendor
- # The developer
- # The SBOM only guy
- # The product security guys



## The externals

---

- # Hobbyist, Kiddies, ...
- # The pentester
- # The security researcher
- # The research team
- # The vendor
- # The developer
- # The SBOM only guy
- # The product security guys

# Understand the pentester needs

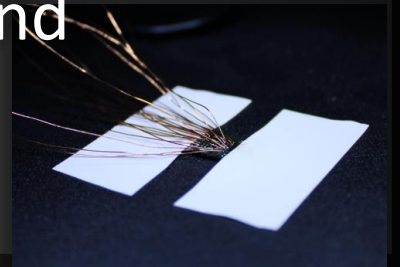
# Target: 3rd party product

# Approach:

- Black-box analysis
- Manual testing the device
- Automated tests wherever possible
- Firmware analysis will give insights of the target
- Firmware analysis extends the black-box approach to be more grey-box
- The SBOM is the source for further identification of known vulnerabilities

# Goal: Identify and report unknown vulnerabilities or known and exploitable vulnerabilities

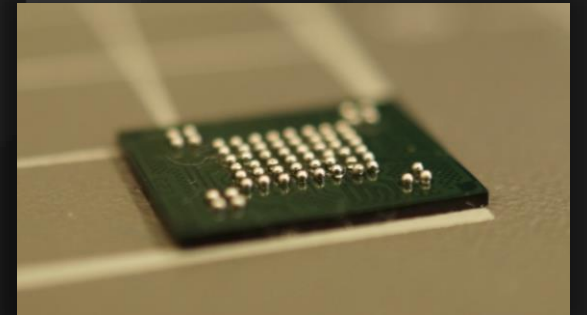
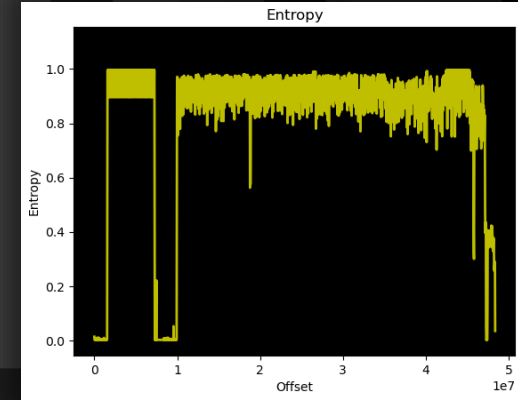
```
(mik3@emba)~$ sudo unblob ./firmware-stuff/Firmware_images/DSP-W110_REVA_FIRMWARE_v1.22B03.zip -e temp
2023-06-29 07:15:59 [info] Start processing file file=/home/mik3/firmware-stuff/Firmware_images/DSP-W110_REVA_FIRMWARE_v1.22B03.zip pid=2924787
2023-06-29 07:16:00 [warning] Found unknown Chunks chunks=[0x0-0x88, 0xcb687-0xe0048] pid=2924806
2023-06-29 07:16:00 [info] Extracting unknown chunk chunk=0x0-0x88 path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_extract/
2023-06-29 07:16:00 [info] Extracting unknown chunk chunk=0xcb687-0xe0048 path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_
2023-06-29 07:16:00 [info] Extracting valid chunk chunk=0xe0048-0x421048 path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_
2023-06-29 07:16:00 [info] Extracting valid chunk chunk=0x88-0xcb687 path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_exti
2023-06-29 07:16:00 [warning] Found unknown Chunks chunks=[0x0-0x32ef] pid=2924806
2023-06-29 07:16:00 [info] Extracting unknown chunk chunk=0x0-0x32ef path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_extrai
2023-06-29 07:16:00 [info] Extracting valid chunk chunk=0x32ef-0x8bf7 path=DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_ex
t/13039-35831.elf32 pid=2924806
(mik3@emba)~$ ls temp/DSP-W110_REVA_FIRMWARE_v1.22B03.zip_extract/DSP-W110_A2_FW122B03.bin_extract/917576-4329544.squashfs_v4_le_extract
bin dch dev etc etc-ro lib linuxrc lost+found mnt proc root sbin sys tmp usr var version wwwd www-ro
```



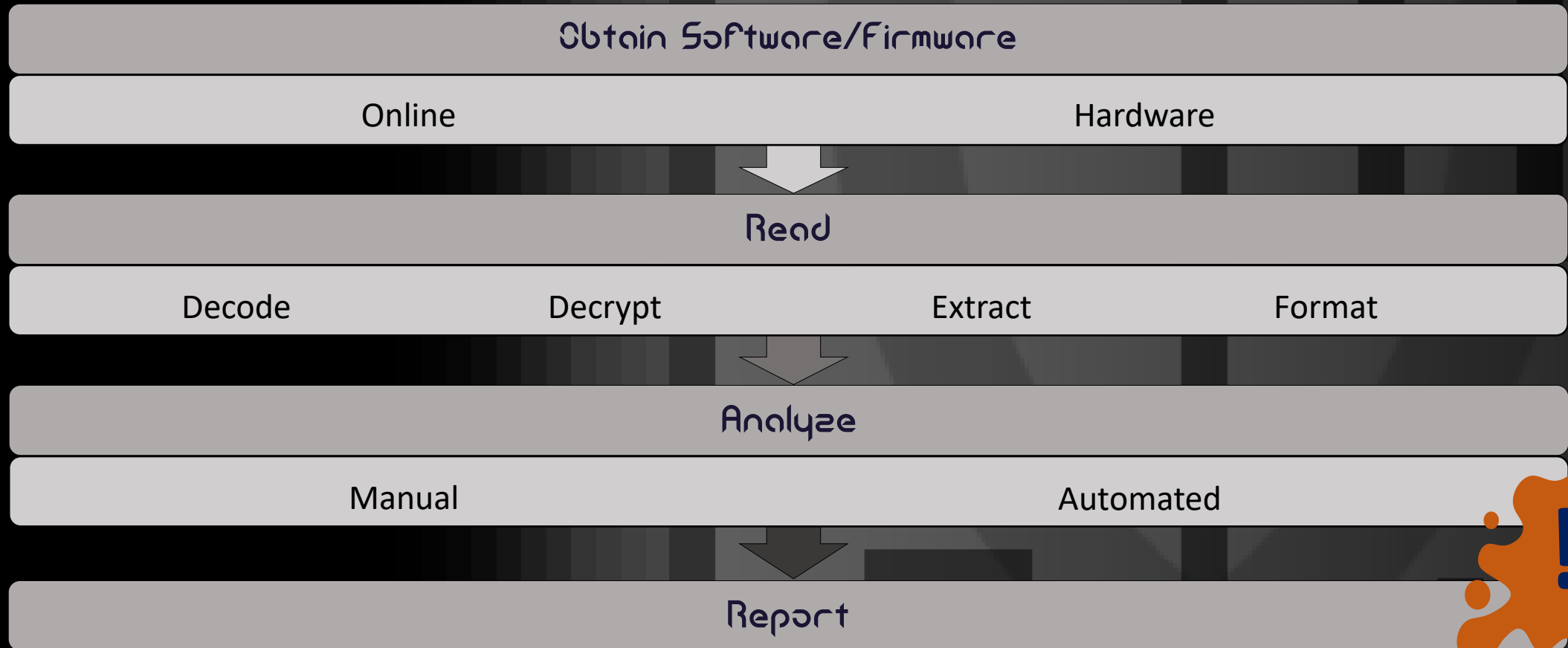
# Understand the pentester needs

## # Penetration testers needs:

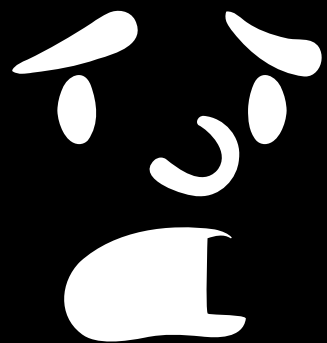
- Tooling to improve the manual testing workflow
- Automated Firmware extraction
- Automated Configuration analysis
- Automated SBOM building (including CVE and PoC/exploit detection)
- Non package manager SBOM needed
- Binary analysis should identify the juicy stuff
- Central management and team support



# External – Process







Got another  
box to test





Can you give me the  
firmware?

Get it from the  
vendors website

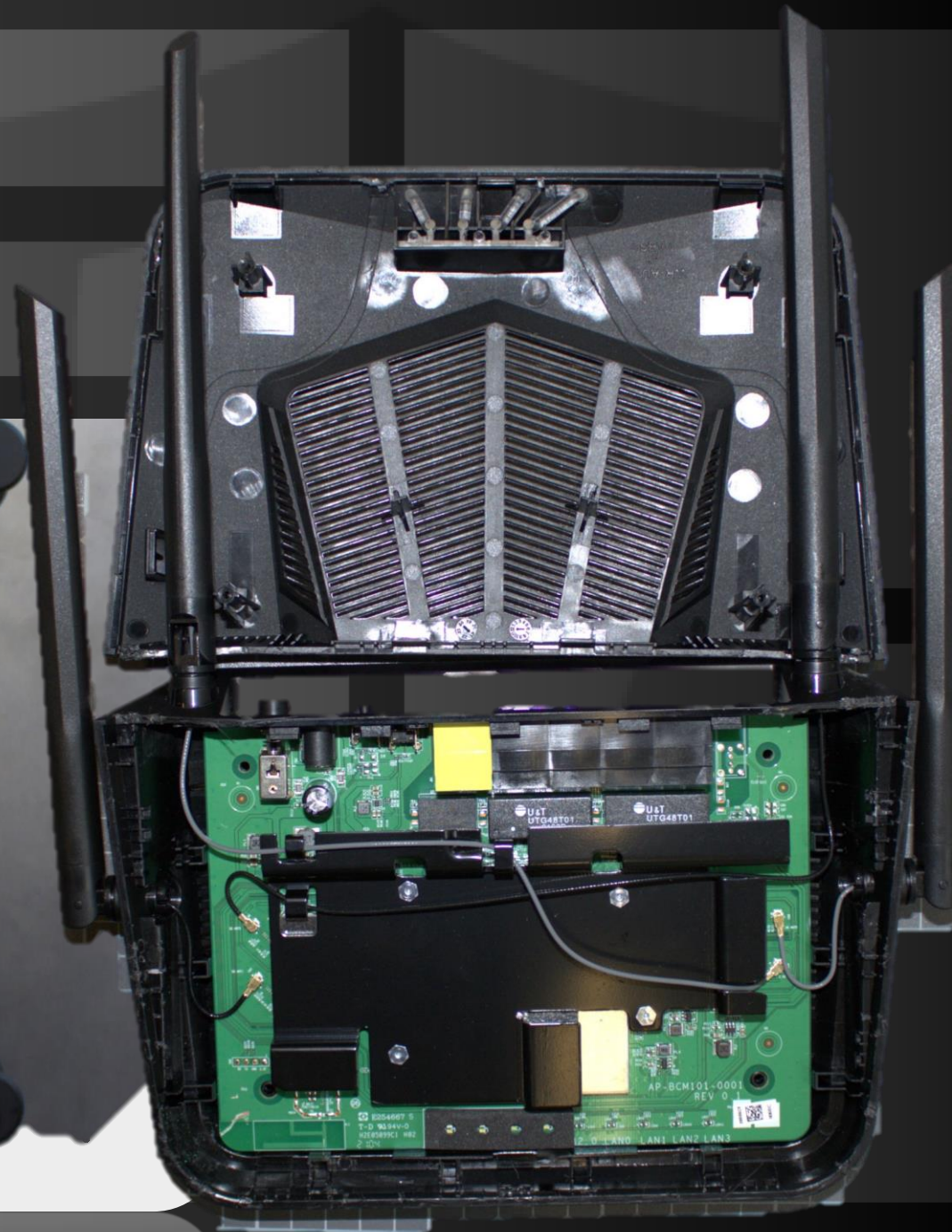
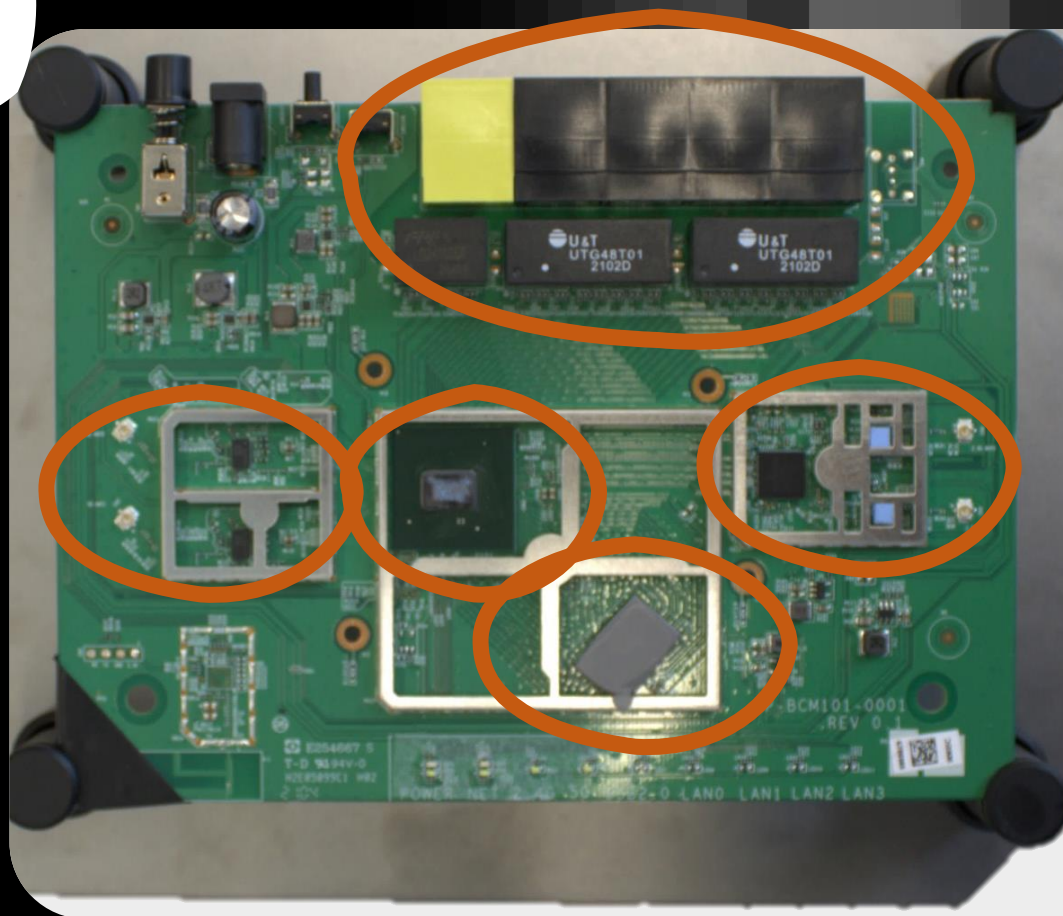
No firmware  
updates from the  
vendor

Let's look at the  
hardware than





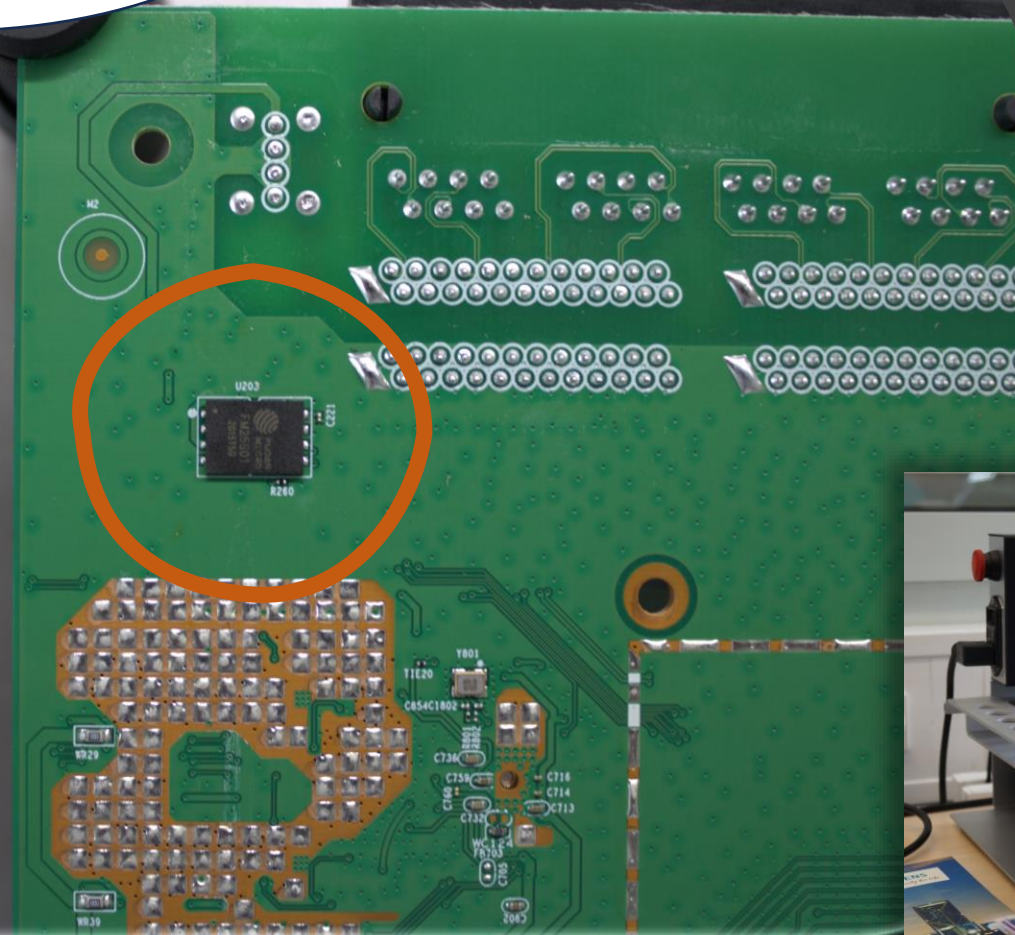
Found the  
Flash yet?







How do I  
get it out of  
there?



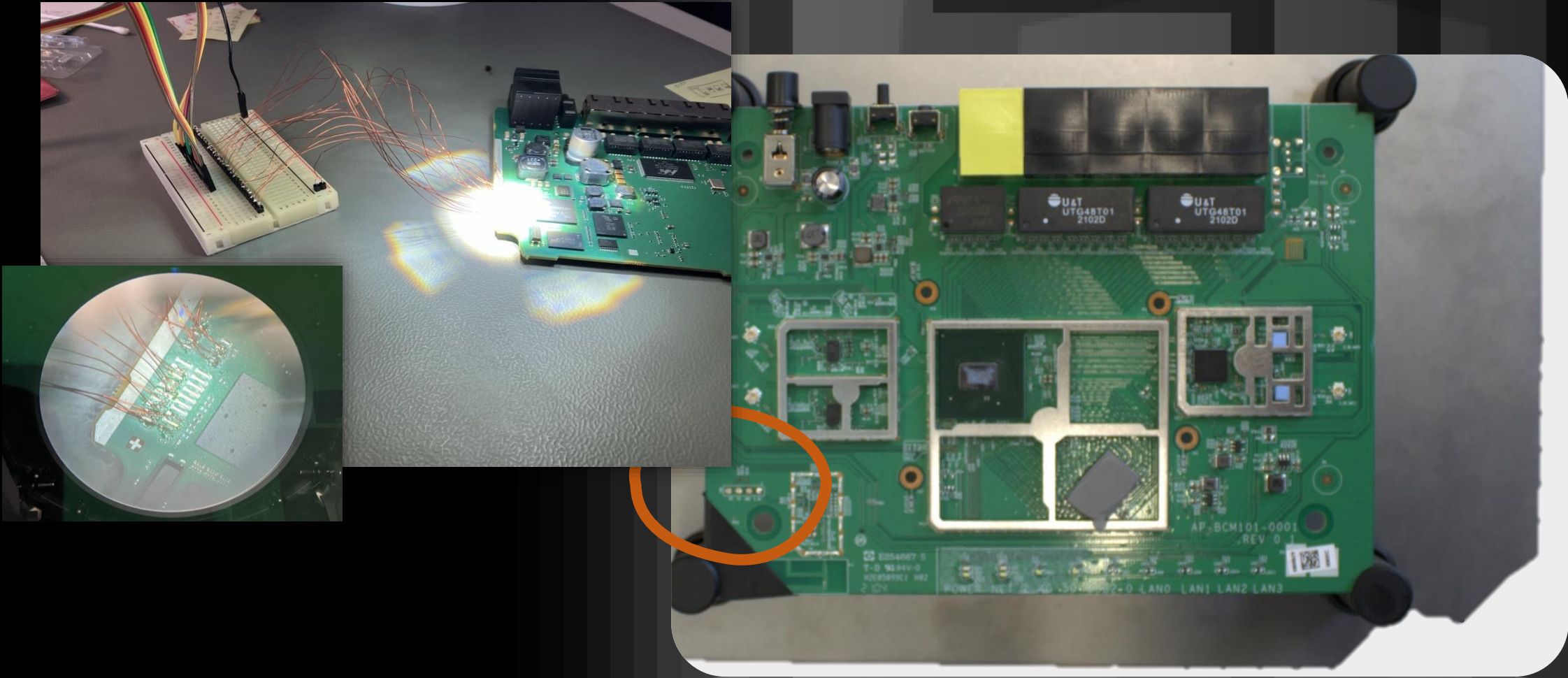


No firmware updates from the vendor

Let's look at the hardware than



# Debug Interface





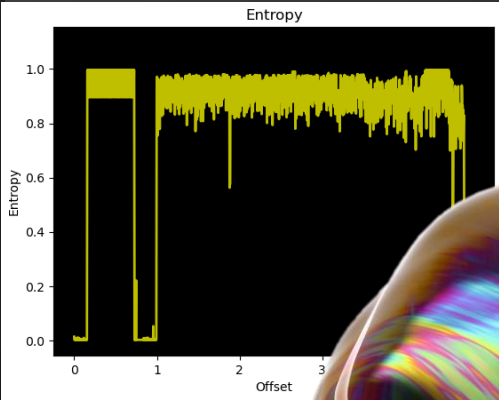
Let's dump  
the flash

```
(benedikt@WorkKaliVM)-[~/git-repos/cfedump]  
$ python -m bcm_cfedump -D /dev/ttyUSB0 -O nand.img -t 0.01 nand
```

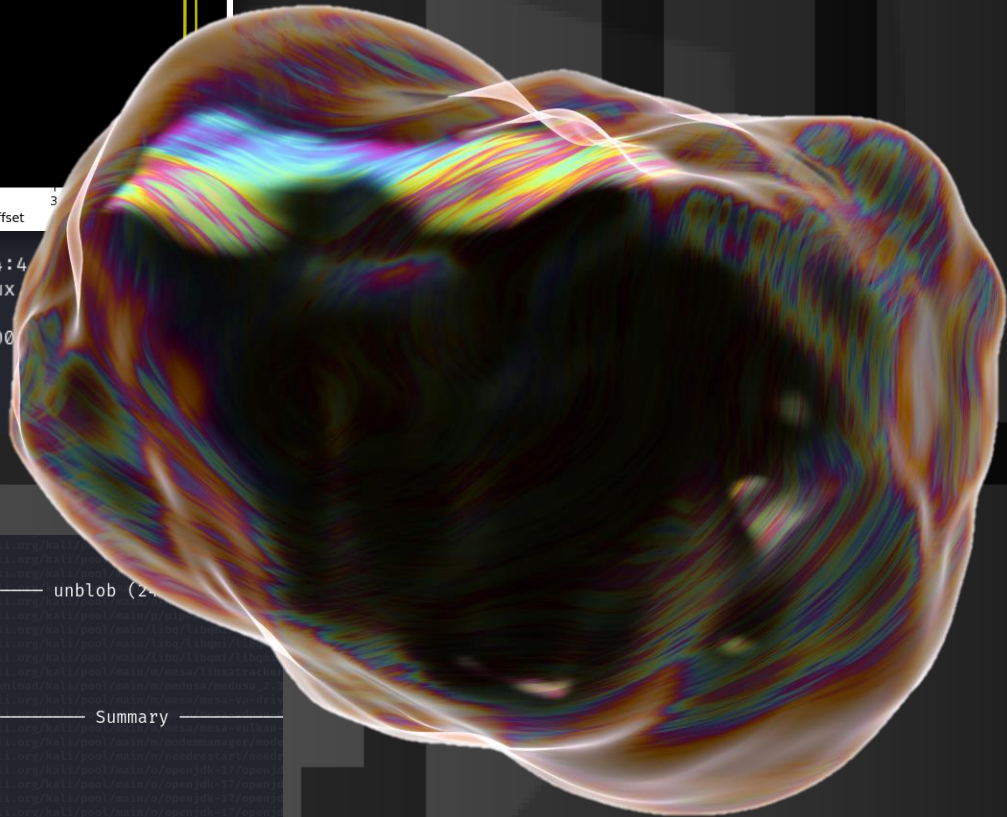
Waiting for a prompt...

```
:: [195/262144 pages] [390.0KB/512.0MB] [1.2B/s] [ETA: 2d 13h 35m 32s]
```

# BLOB



DECIMAL	HEXADECIMAL	DESCRIPTION
64	0x40	uImage header, header size: 64 bytes, header CRC: 0x74D4CA3C, created: 2020-09-04 14:40:34, image size: 1345726 bytes, Data Address: 0x80100000, Entry Point: 0x80361B30, data CRC: 0xF23A1762, OS: Linux CPU: MIPS, image type: OS Kernel Image, compression type: gzip, image name: "OpenStage-uImage-3.5.21.0000"
128	0x80	gzip compressed data, maximum compression, from Unix, last modified: 1970-01-01 00:00:00 (null date)
2097152	0x200000	gzip compressed data, from Unix, last modified: 2020-09-04 15:34:45



\$ unblob Downloads/OS15\_SIP\_V3\_R5\_21\_0.img

Extracted files: 985  
Extracted directories: 94  
Extracted links: 161  
Extraction directory size: 104.18 MB  
Chunks identification ratio: 96.31%

Chunks distribution

Chunk type	Size	Ratio
TAR	43.77 MB	43.93%
ELF32	37.50 MB	37.64%
GZIP	14.68 MB	14.73%
UNKNOWN	3.68 MB	3.69%

(kali@kali)~[~/git-repos/emba]  
\$ sudo ./emba.sh -f ~/DIR300B5 FW214WWB01.bin -l ~/emba-logs





Do you have the  
Firmware Analysis  
Report for me

Yes, on a USB.  
Can you wait until  
tomorrow?

There has to be a  
better way...

...





Do you have the  
Firmware Analysis  
Report for me

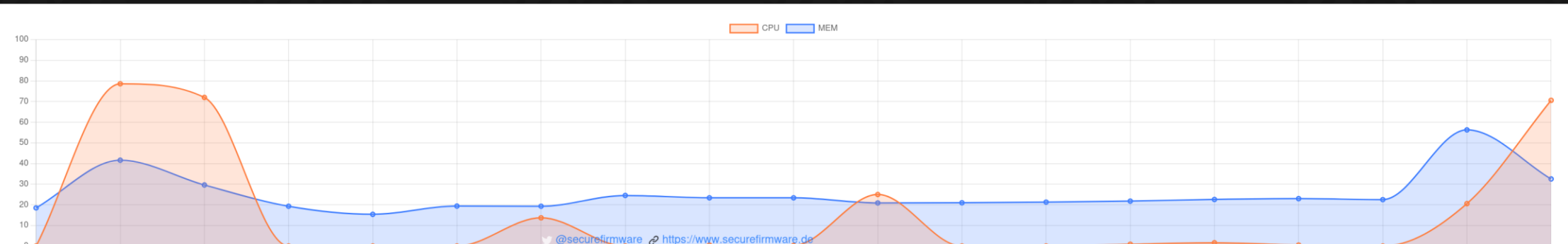
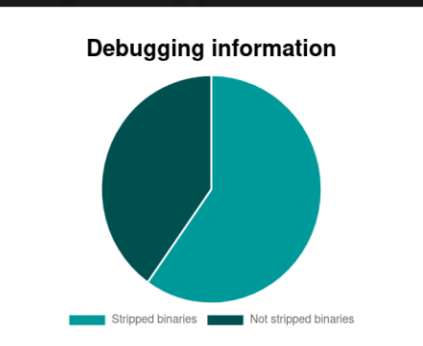
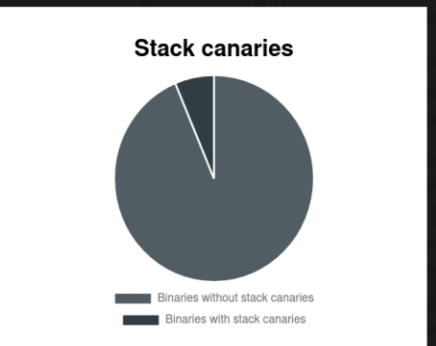
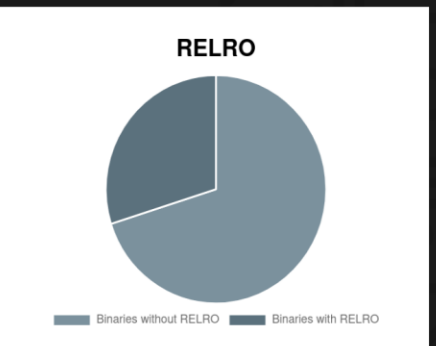
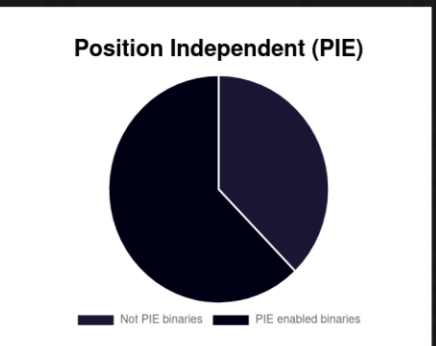
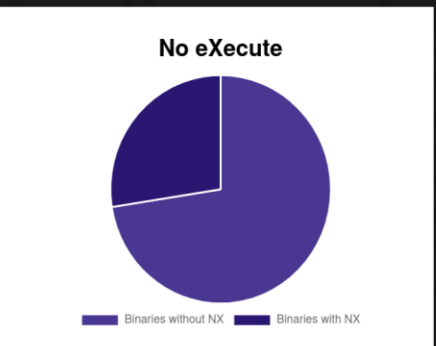
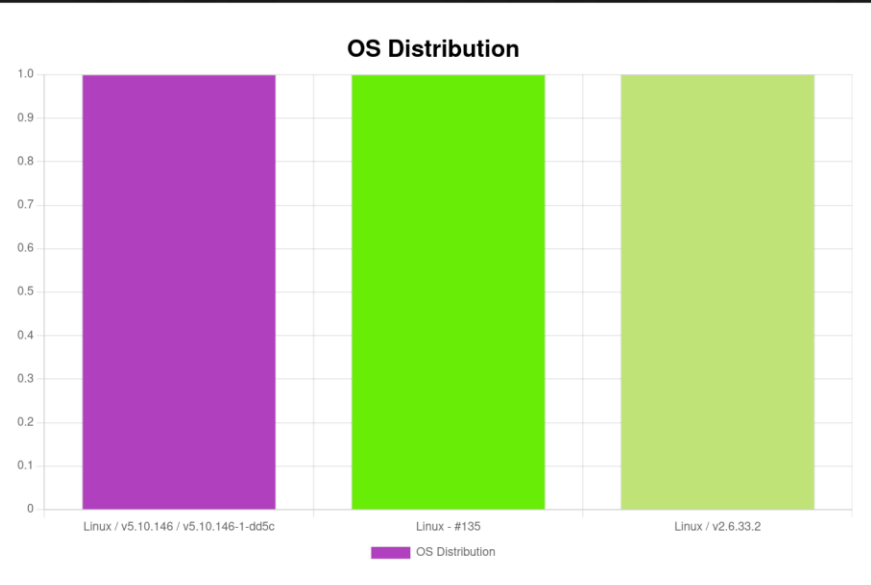
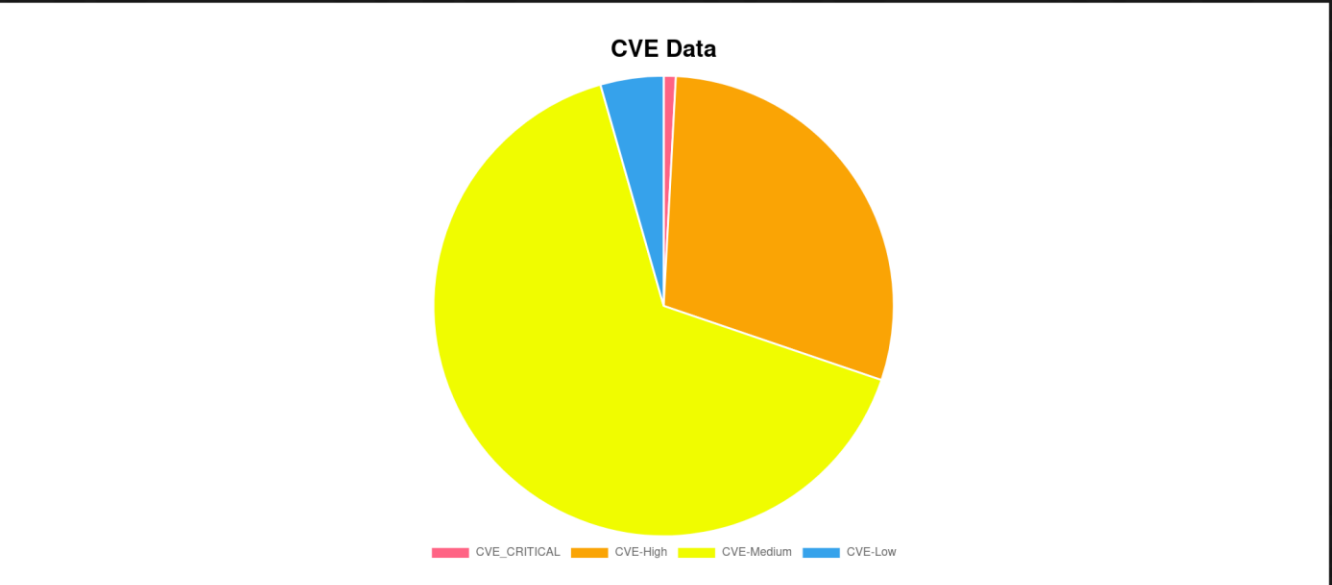
Yes, on a USB.  
Can you wait until  
tomorrow?

There has to be a  
better way...

Here:  
<https://embark.local>



Firmware tested	Files analysed	Directories found	Binaries analysed	Vulnerabilities (CVEs)	Exploits
3	2993	322	331	4673	197



# EMBArk

Current Version:

[Release-Notes](#)

## EMBA Configuration

EMBA version 1.5.2

Container version 11e1a5cdd129a1f1b6adf65dbc3122465b025c448639b164caa76de42e81471a

Github hash e2b11c5c8aa7a2645616c61e2f61d40762643e75

## EMBA Log

```
[*] Necessary utils on system:
    docker - ok
    inotifywait - ok
    notify-send - ok

[*] EMBA update starting ...
From https://github.com/e-m-b-a/emba
* branch      master       -> FETCH_HEAD
Already up to date.
Using default tag: latest
latest: Pulling from embeddedanalyzer/emba
Digest: sha256:8c2a6c8cbe73024ef01f649089fbc500da95c012f296056fa45b3cab6d8b245
Status: Image is up to date for embeddedanalyzer/emba:latest
docker.io/embeddedanalyzer/emba:latest
[*] EMBA update - EPSS database update
    Already up to date.
[*] EMBA update - CVE database update
    Already up to date.
[*] EMBA update - docker image
Using default tag: latest
latest: Pulling from embeddedanalyzer/emba
Digest: sha256:8c2a6c8cbe73024ef01f649089fbc500da95c012f296056fa45b3cab6d8b245
Status: Image is up to date for embeddedanalyzer/emba:latest
docker.io/embeddedanalyzer/emba:latest
[*] Please note that this was no update of installed system packages.
[*] Please restart your EMBA scan to apply the updates ...
```

## Controls

↑

↑x5

↑

increase view size

↓

↓x5

↓

decrease view size

[Raw EMBA log file](#)

Check EMBA

Update EMBA

Option

Host and container

Check EMBA

Option

- ☐ Git Update  
☐ Docker Update  
☐ CVE Update

Update EMBA

Check

Update

1. Upload

2. Add new Vendor (optional)

3. Add new Label (optional)

4. Define the device-under-test (optional)

5. Analyze

Upload firmware

Supported archive types : firmware binary and most archive types


Selected file:  
None





Drag & Drop Firmware files


Select file

Upload


 Dashboard


 Tracker


 Uploader

 Im-/Exporter

 Progress

 Reports

 Updater

 My Account



1. Upload

2. Add new Vendor (optional)

3. Add new Label (optional)

4. Define the device-under-test (optional)

5. Analyze

Analyze a firmware image/archive

SBOM mode

Expert mode

Firmware

0f71a079-105a-4ddc-9ebd-48d27b1c312f - DIR600B6\_FW215WWb02.bin

Firmware File object

Firmware version

2.15

Firmware version

Device

☐ Dir600(DLink)

☐

☐

☐ octean-edge-router(openwrt)

☐

device/platform

Testing notes

latest version

Testing notes

Cancel

Analyze

Dashboard

Tracker

Uploader

Im-/Exporter

Progress

Reports

Updater

My Account





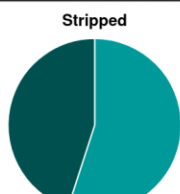
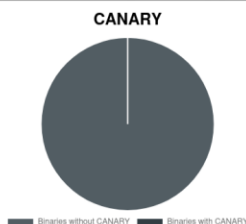
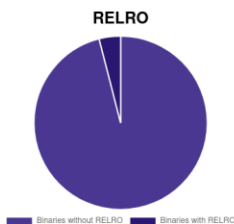
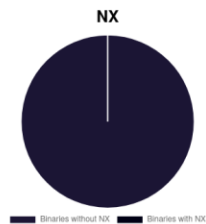
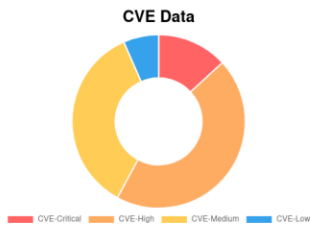


e42cb60c-bf3a-4a38-943f-21950e6e4492	dir600b_v2.14_d1mg.bin	ICS demo	Dir600(DLink)	2.14	March 21, 2025, 11:12 a.m.	March 21, 2025, 12:34 p.m.	<a href="#">router</a>	<a href="#">+</a>
--------------------------------------	------------------------	----------	---------------	------	----------------------------	----------------------------	------------------------	-------------------

## Action-Menue

Delete Analysis





## Collect CVE and exploit details.

[\*] Vulnerability details for **ddnsd** / version **0.1** / source **UEMU**:

[+] Found **NO** CVEs and **NO** exploits (including POC's) in **ddnsd** with version **0.1** (source **UEMU**).

[\*] Vulnerability details for **ecmh** / version **2005.02.09** / source **UEMU**:

[+] Found **NO** CVEs and **NO** exploits (including POC's) in **ecmh** with version **2005.02.09** (source **UEMU**).

[\*] Vulnerability details for **busybox** / version **1.14.1** / source **STAT/UEMU**:

```

busybox      : 1.14.1      : CVE-2016-6301 : 7.8 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2018-100051: 7.5 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2016-2148 : 7.5 : STAT/UEMU      : Exploit (Github: jamesz
irctl (G))
busybox      : 1.14.1      : CVE-2013-1813 : 7.2 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2018-100050: 6.8 : STAT/UEMU      : Exploit (Github: alphaS
busybox      : 1.14.1      : CVE-2011-2716 : 6.8 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2017-16544 : 6.5 : STAT/UEMU      : Exploit (Github: qazbnr
6 (G))
busybox      : 1.14.1      : CVE-2019-5747 : 5.0 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2018-20679 : 5.0 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2016-2147 : 5.0 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2011-5325 : 5.0 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2015-9261 : 4.3 : STAT/UEMU      : No exploit available
busybox      : 1.14.1      : CVE-2014-9645 : 2.1 : STAT/UEMU      : No exploit available

```

Shell scripts detected	114
Shell script issues	509
Binaries checked	100
Versions identified	23
Exploits identified	13
Metasploit modules	0
Critical CVE	["10", "4"]
High CVE	["34", "60"]
Medium CVE	["27", "72"]
Low CVE	["5", "9"]
NX disabled binaries	100
RELRO disabled binaries	96
PIE disabled binaries	50
Stack canaries disabled binaries	100
Stripped binaries	55
STRCPY binary: dnsmasq	28
STRCPY binary: rgin	7
STRCPY binary: iptables-multi	7
STRCPY binary: tc	5
STRCPY binary: rdisc6	5
STRCPY binary: nsbbox	3
STRCPY binary: updateleases	2
STRCPY binary: ubcom	2
STRCPY binary: neaps	2



## The internals

---

- # Hobbyist, Kiddies, ...
- # The pentester
- # The security researcher
- # The research team
- # The vendor
- # The developer
- # The SBOM only guy
- # The product security guys

# Understand the needs

## # Target:

- 3rd party product
- Internally developed products

## # Approach:

- All what we have seen so far
- More white box analysis possible
- **Massive need for SBOMs**

## # Goal: Build SBOMs automatically

SBOMs ...



# What the BOM?!?

## Bill of materials

🌐 14 languages ▾

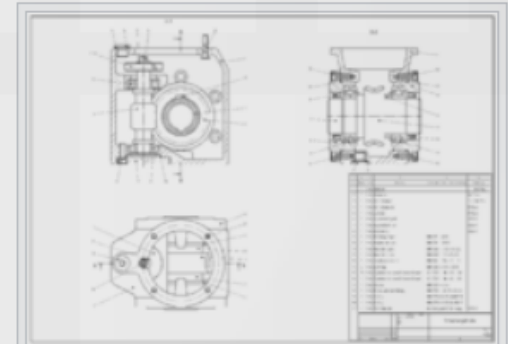
Article [Talk](#)


[Read](#) [Edit](#) [View history](#) [Tools](#) ▾

From Wikipedia, the free encyclopedia

A **bill of materials** or **product structure** (sometimes **bill of material**, **BOM** or **associated list**) is a [list](#) of the raw materials, sub-assemblies, intermediate assemblies, sub-components, parts, and the quantities of each needed to manufacture an [end product](#). A BOM may be used for communication between manufacturing partners or confined to a single manufacturing plant. A bill of materials is often tied to a [production order](#) whose issuance may generate reservations for components in the bill of materials that are in stock and requisitions for components that are not in stock.

The first hierarchical databases were developed for automating bills of materials for manufacturing organizations in the early 1960s. At present, this BOM is used as a data base to identify the many parts and their codes in automobile manufacturing companies.



An example of a BOM for a mechanical assembly (in German) 

# What the $S_{\text{software}}$ SBOM?!?

A **Software Bill of Materials (SBOM)** is a formal record containing the **details and supply chain relationships of various components** used in building software.

- ➔ SBOM does not include our source code
- ➔ SBOM does not include our configuration
- ➔ SBOM does not include our build parameters
- ➔ SBOM should not include vulnerability information (but it can – VEX ahead)

**SBOMs only define the structure, content, and interpretation between parties but it does not guarantee the quality of the data**

# Example of a CycloneDX SBOM

```
{
  "$schema": "http://cyclonedx.org/schema/bom-1.5.schema.json",
  "bomFormat": "CycloneDX",
  "specVersion": "1.5",
  "serialNumber": "urn:uuid:e17430e1-7cc0-4f9a-b8d2-3779ffab7af7",
  "version": 1,
  "metadata": {
    "timestamp": "2025-02-15T00:54:55+01:00",
    "tools": {
      "components": [
        {
          "type": "application",
          "author": "EMBA community",
          "name": "EMBA binary analysis environment",
          "version": "1.5.1-1afe761dc1179b33373ef88a0139ac8c6f89db40",
          "description": "EMBA firmware analyzer - https://github.com/e-m-b-a/emba"
        }
      ]
    },
    "component": {
      "name": "/home/m1k3/firmware-analysis/emba_logs_sbom_directory/firmware/testimage_linux_sbom_small",
      "type": "data",
      "bom-ref": "e4273b36-e5e1-477e-87c4-5dc63eb20358"
    },
    "supplier": {
      "name": "EMBA binary analyzer",
      "url": [
        "https://github.com/e-m-b-a/emba"
      ]
    },
    "lifecycles": [
      {
        "phase": "operations"
      }
    ]
  }
}
```

A few required elements

## CycloneDX v1.5 JSON Reference

Type: object

No Additional Properties

> bomFormat Required

> specVersion Required

> serialNumber

> version

> metadata

> components

> services

> externalReferences

> dependencies

> compositions

> vulnerabilities

> annotations

> formulation

# Example of a CycloneDX SBOM

- A few required elements
- Warning: BSI requirements are stricter

▼ components

Type: array

A list of software and hardware components.

All items must be unique

No Additional Items

Each item of this array must be:

Type: object

No Additional Properties

> type Required

> mime-type

> bom-ref

> supplier

> author

```

"components": [
{
  "type": "library",
  "name": "avahi-autoipd",
  "version": "0.6.28-1",
  "supplier": {
    "name": "openwrt_developers_team_openwrt-devel@openwrt.org"
  },
  "group": "OpenWRT",
  "bom-ref": "f1a13c39-07fd-4810-8dea-8fac44c39d6d",
  "scope": "required",
  "cpe": "cpe:2.3:a:avahi-autoipd:avahi-autoipd:0.6.28-1:*:*:*:*:*:*:*",
  "purl": "pkg:opkg/debian/avahi-autoipd@0.6.28-1&distro=debian-12",
  "properties": [
    {
      "name": "EMBA:sbom:source_location:1:source_path",
      "value": "'/logs/firmware/testimage_linux_sbom_small/usr/lib/opkg/info/avahi-autoipd.control'"
    },
    {
      "name": "EMBA:sbom:2:minimal_identifier",
      "value": "'\\:\\:avahi-autoipd:0.6.28-1'"
    },
    {
      "name": "EMBA:sbom:3:confidence",
      "value": "'high'"
    },
    {
      "name": "EMBA:sbom:4:dependency",
      "value": "'libdaemon'"
    },
    {
      "name": "EMBA:sbom:location:5:path",
      "value": "'/etc/avahi/avahi-autoipd.action'"
    },
    {
      "name": "EMBA:sbom:location:6:path",
      "value": "'/usr/sbin/avahi-autoipd'"
    }
  ],
  "hashes": [
    {
      "alg": "MD5",
      "content": "af2d2b2b3e7b9ad806d3b3d58e8ba1fc"
    }
  ]
}
]

```



# Why is a SBOM useful?

With a SBOM infrastructure we always know which component or library is used in which product.


Product A has multiple components listed in the SBOM:

- ➡ BusyBox v1.2.3
- ➡ Linux Kernel v9.11.2
- ➡ Component 3 **which is vulnerable to some critical vulnerability**
- ➡ ...
- ➡ Component x

With these details we can perform targeted Vulnerability Management and Software supply chain risk management

# Why is SBOM essential?

- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- Building and maintaining more secure products is really important for us



Official Journal  
of the European Union

EN  
L series

2024/2847 20.11.2024

**REGULATION (EU) 2024/2847 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL**  
**of 23 October 2024**  
**on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) No 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act)**

(Text with EEA relevance)

individuals or entities receive recognition and compensation for their efforts. This refers to so-called ‘bug bounty programmes’.

(77) In order to facilitate vulnerability analysis, manufacturers should identify and document components contained in the products with digital elements, including by drawing up an **SBOM**. An **SBOM** can provide those who manufacture, purchase, and operate software with information that enhances their understanding of the supply chain, which has multiple benefits, in particular it helps manufacturers and users to track known newly emerged vulnerabilities and cybersecurity risks. It is of particular importance that manufacturers ensure that their products with digital elements do not contain vulnerable components developed by third parties. Manufacturers should not be obliged to make the **SBOM** public.

(78) Under the new complex business models linked to online sales, a business operating online can provide a variety of services. Depending on the nature of the services provided in relation to a given product with digital elements, the same entity may fall within different categories of business models or economic operators. Where an entity provides only online intermediation services for a given product with digital

Source: <https://eur-lex.europa.eu/eli/reg/2024/2847/oj>

# Why is SBOM essential?



CRA proposed  
(2022)

EU Parliament  
approved (2024)

Vulnerability and  
incident reporting  
takes effect (2026)

EU Parliament  
provisional  
agreement (2023)

EU Journal  
publication (Nov.  
2024)

Most requirements  
(including SBOM)  
take effect (Dec.  
2027)

# Why is SBOM essential?

- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- We need to sell products with digital elements also after 2027
- Building and maintaining more secure products is really important for us



€15 million

Failure to comply with the CRA could lead to significant penalties of up to **€15 million** or 2.5% of a company's global turnover (revenue), whichever is higher. Alongside the potential loss of the requisite CE mark, the CRA effectively bans non-compliant products from being sold in the EU.

[Understanding the EU Cyber Resilience Act \(CRA\): Why it matters a...](https://mender.io/blog/understanding-the-eu-cyber-resilience-act-cra-why-it-matters-an...)  
[mender.io/blog/understanding-the-eu-cyber-resilience-act-cra-why-it-matters-an...](https://mender.io/blog/understanding-the-eu-cyber-resilience-act-cra-why-it-matters-an...)

War dies hilfreich?

# SBOM challenges

- **New developed software in modern frameworks**

- The used SBOM is out of the box available
  - Source SBOM via Metadata
- DotNet, Maven, ...
- Not covered in this talk



Source: <https://wileyeducation.com.au/blog/brownfield-vs-greenfield-projects-in-food-processing-a-comprehensive-guide-for-australian-business-leaders/>



# SBCM challenges

- **The real world:**
  - Metadata only partly available
  - Binary blobs (like firmware)
  - 3<sup>rd</sup> party components/libraries
  - Brownfield environments
  - Hybrid environments



Source: <https://wileyeducation.com.au/blog/brownfield-vs-greenfield-projects-in-food-processing-a-comprehensive-guide-for-australian-business-leaders/>

# SBCM challenges

- **Installed software components in OS**

- In theory it is as simple as “**The package manager is the master**”
- The reality ... dpkg (+deb), python (pip, requirements, poetry, wheel), rust, rpm (sqlite, packages, berkley), opkg (+ipk), conanfiles, Alpine apk, java, npm, ...

- **Used components and libraries in binary blobs**

- Packed blobs, file systems, firmware update, extracted from hardware
- Detection if something is already handled via package manager
- \*.exe, \*.elf, \*.dll, \*.so, kernel and modules, libraries

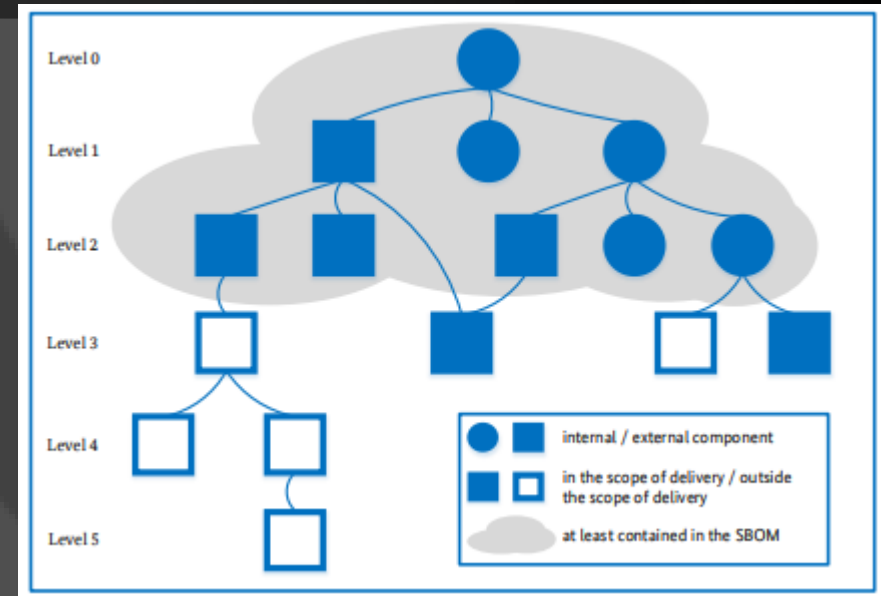
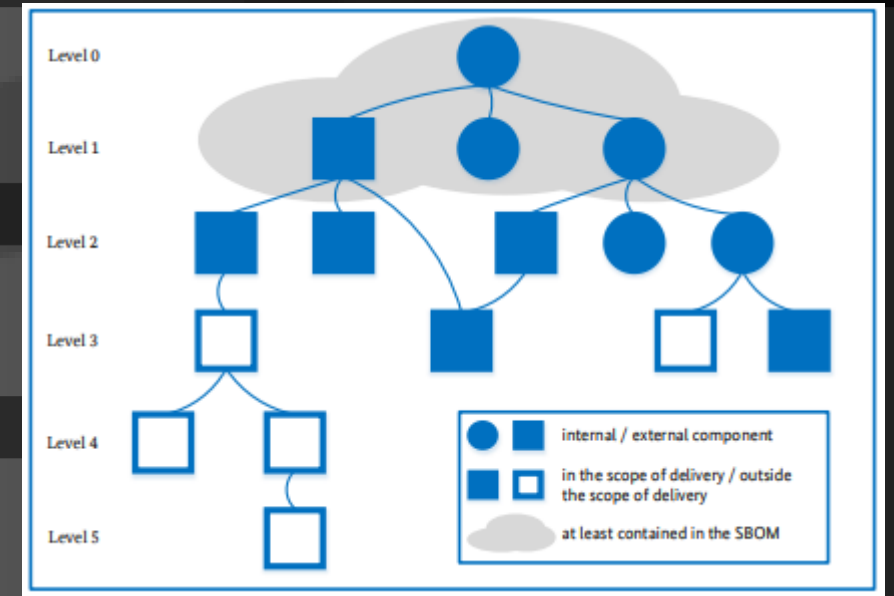
# SBOM challenges

- **Building dependency tree**

- Every package manager is doing it individually
- Cross package manager dependencies
- Binary dependencies without package managers

- **Recursive analysis**

- Extraction, binary and package manager analysis
- Firmware filesystem with package manager and additional binary blobs





# SBCM Demo

Admin-page

The servers timezone is: UTC

Time zone:

UTC

Set

API Key:

Copy

Generate



Admin-page

The servers timezone is: UTC

Time zone:

UTC

Set

API Key:

Z6NjUyujp0GnVyeta534Rp4

Copied!

Generate

←

→

Home

Workspaces

API Network

Search Postman

CtrlK

Invite

Upgrade

GET http://embark.local/user

POST http://embark.local/ap

GET http://embark.local/stat

GET http://embark.local/api/

+

No environment

Collections

Environments

Flows

History

HTTP

http://embark.local/api/uploader/

Save

Share

</>

POST

http://embark.local/api/uploader/

Send

Params

Authorization

Headers (10)

Body

Scripts

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key		Value	Description	Bulk Edit
✓	file	File	DIR600B6_FW215WWb02.bin		
	Key	Text	Value	Description	

Body

Cookies (1)

Headers (13)

Test Results

201 Created

1.09 s

641 B

{ } JSON

Preview

Visualize

```
1 {
2   "status": "success",
3   "id": "be68326b-5f62-40b5-b08a-5d75dd2435d4"
4 }
```

```
cylox@EMBark25:~$ curl -L 'http://embark.local/api/uploader/' -H 'Authorization: bs7xXgI70gxDB05p-hG5JO_Yr0D5cOP4o5Jf0ESL058' -H 'Cookie: messages=W1siX19qc29uX21lc3NhZ2UiLDAsMjAsInVwbG9hZCBzdWNjZXRnZnVsLiIsIiJdXQ:1uNSwL:KkRKYoQqSQVsYxMVHrQh8iMzCFQZYvkC_ULoKEwNvE8' -F 'file=@"/home/cylox/Downloads/DIR600B6_FW215WWb02.bin"' | jq
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 3528k 100 64 100 3528k 194 10.4M --:--:-- --:--:-- --:--:-- 10.5M
{
  "status": "success",
  "id": "597ee3fe-28f0-4477-92f2-2eef1e90c093"
}
```

←

→

Home

Workspaces

API Network

Search Postman

CtrlK

Invite

Upgrade

GET http://embark.local/user

POST http://embark.local/ap

GET http://embark.local/stat

GET http://embark.local/api/

+

No environment

HTTP

http://embark.local/status\_report/be68326b-5f62-40b5-b08a-5d75dd2435d4

Save

Share

</>

GET

http://embark.local/status\_report/be68326b-5f62-40b5-b08a-5d75dd2435d4

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

Body

Cookies (1)

Headers (10)

Test Results

202 Accepted • 432 ms • 447 B •

{ } JSON

Preview

Visualize

1 {

2 "status": "running",

3 "message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",

4 "completion": "1.25% finished"

5 }

cylox@EMBArk25:~\$ curl -L 'http://embark.local/status\_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5JO\_Yr0D5cOP4o5Jf0ESL058' -H 'Cookie: messages=.eyJLjlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUsguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUmpmXrlCVWVCQmoJfCw4rYgFC\_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 132 100 132 0 0 1488 0 ---:--:-- ---:--:-- ---:--:-- 1500

{

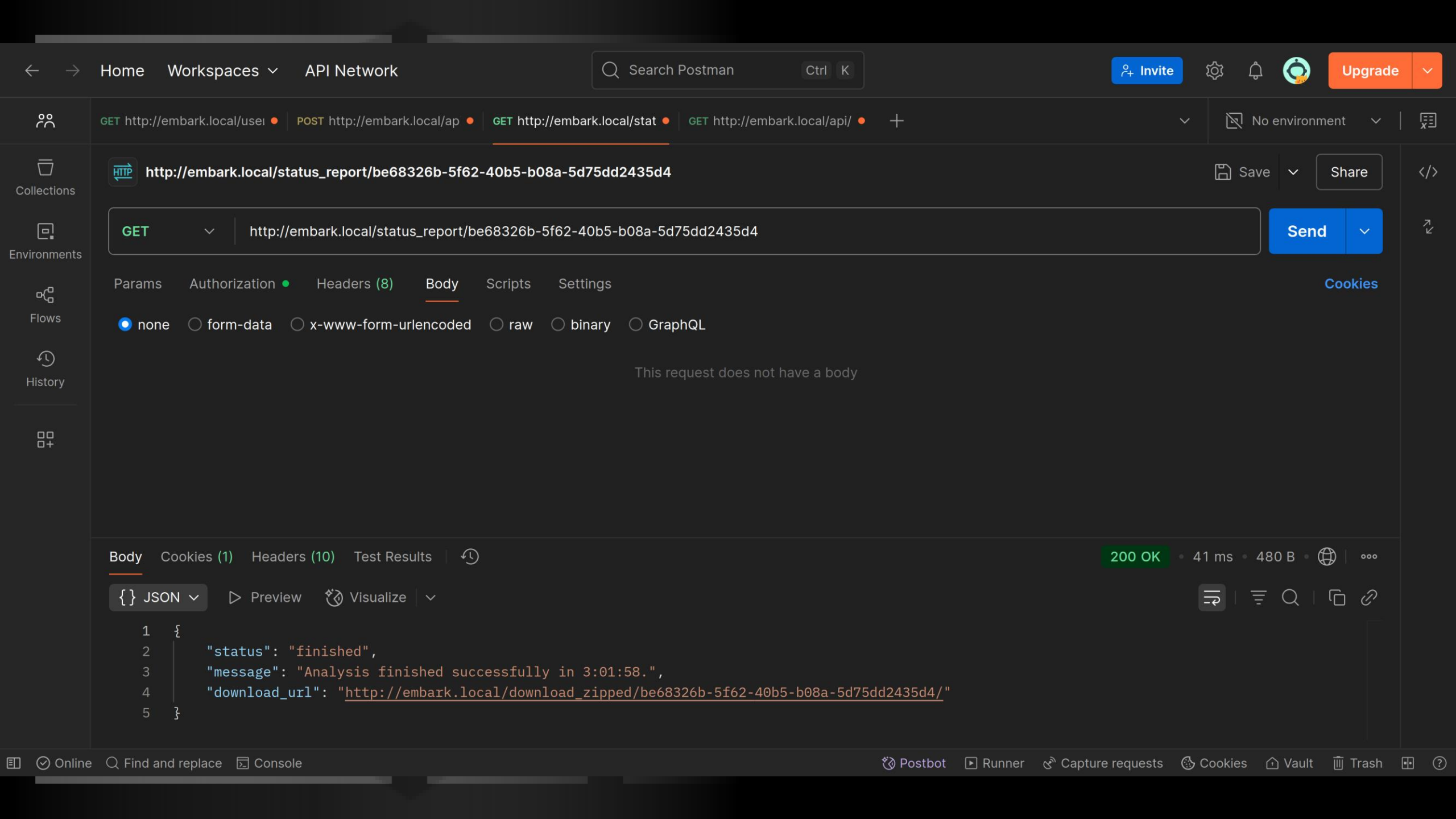
"status": "running",

"message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",

"completion": "7.5% finished"

}

```
cylox@EMBArk25:~$ curl -L 'http://embark.local/status_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5J0_Yr0D5c0P4o5Jf0ESL058' -H 'Cookie: messages=.eyJlJlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUSguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUMpmXrLCVWVCQmoJfCw4rYgFC_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    147    100    147     0     0    4112       0 --:--:-- --:--:-- --:--:--  4200
{
  "status": "running",
  "message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",
  "completion": "26.041666666666668% finished"
}
cylox@EMBArk25:~$ curl -L 'http://embark.local/status_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5J0_Yr0D5c0P4o5Jf0ESL058' -H 'Cookie: messages=.eyJlJlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUSguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUMpmXrLCVWVCQmoJfCw4rYgFC_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    147    100    147     0     0    2758       0 --:--:-- --:--:-- --:--:--  2826
{
  "status": "running",
  "message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",
  "completion": "30.729166666666668% finished"
}
cylox@EMBArk25:~$ curl -L 'http://embark.local/status_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5J0_Yr0D5c0P4o5Jf0ESL058' -H 'Cookie: messages=.eyJlJlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUSguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUMpmXrLCVWVCQmoJfCw4rYgFC_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    135    100    135     0     0    2039       0 --:--:-- --:--:-- --:--:--  2076
{
  "status": "running",
  "message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",
  "completion": "40.625% finished"
}
cylox@EMBArk25:~$ curl -L 'http://embark.local/status_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5J0_Yr0D5c0P4o5Jf0ESL058' -H 'Cookie: messages=.eyJlJlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUSguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUMpmXrLCVWVCQmoJfCw4rYgFC_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    146    100    146     0     0    3956       0 --:--:~ --:~:~ --:~:~  4055
{
  "status": "running",
  "message": "Analysis has been running since 2025-06-18 12:01:26.491146+00:00.",
  "completion": "91.66666666666669% finished"
}
cylox@EMBArk25:~$ curl -L 'http://embark.local/status_report/be68326b-5f62-40b5-b08a-5d75dd2435d4' -H 'Authorization: bs7xXgI70gxDB05p-hG5J0_Yr0D5c0P4o5Jf0ESL058' -H 'Cookie: messages=.eyJlJlaKj88qzs-Lz00tLk5MT1XSMDAxMtbRKi3IyU9MUSguTU4GSqSV5ugp6SgpxergUu-Tn16skFiUqpCUMpmXrLCVWVCQmoJfCw4rYgFC_DB7:1uRrTq:6PCFxZDSQhAhAyYduxA1mPi7sXi4etNk5E5CIt-DVzc' | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100    131    100    131     0     0    2173       0 --:~:~ --:~:~ --:~:~  2183
{
  "status": "finished",
  "message": "Analysis finished successfully. The logs are being zipped and will soon be ready for download."
}
```



GET http://embark.local/user ● POST http://embark.local/ap ● GET http://embark.local/stat ● GET http://embark.local/api/ ● +

▾ No environment ▾ 📄



Collections



Environments



Flows



History



http://embark.local/status\_report/be68326b-5f62-40b5-b08a-5d75dd2435d4

💾 Save Share

GET ▾ http://embark.local/status\_report/be68326b-5f62-40b5-b08a-5d75dd2435d4

Send ▾

Params Authorization ● Headers (8) Body Scripts Settings

Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies (1) Headers (10) Test Results 🔄

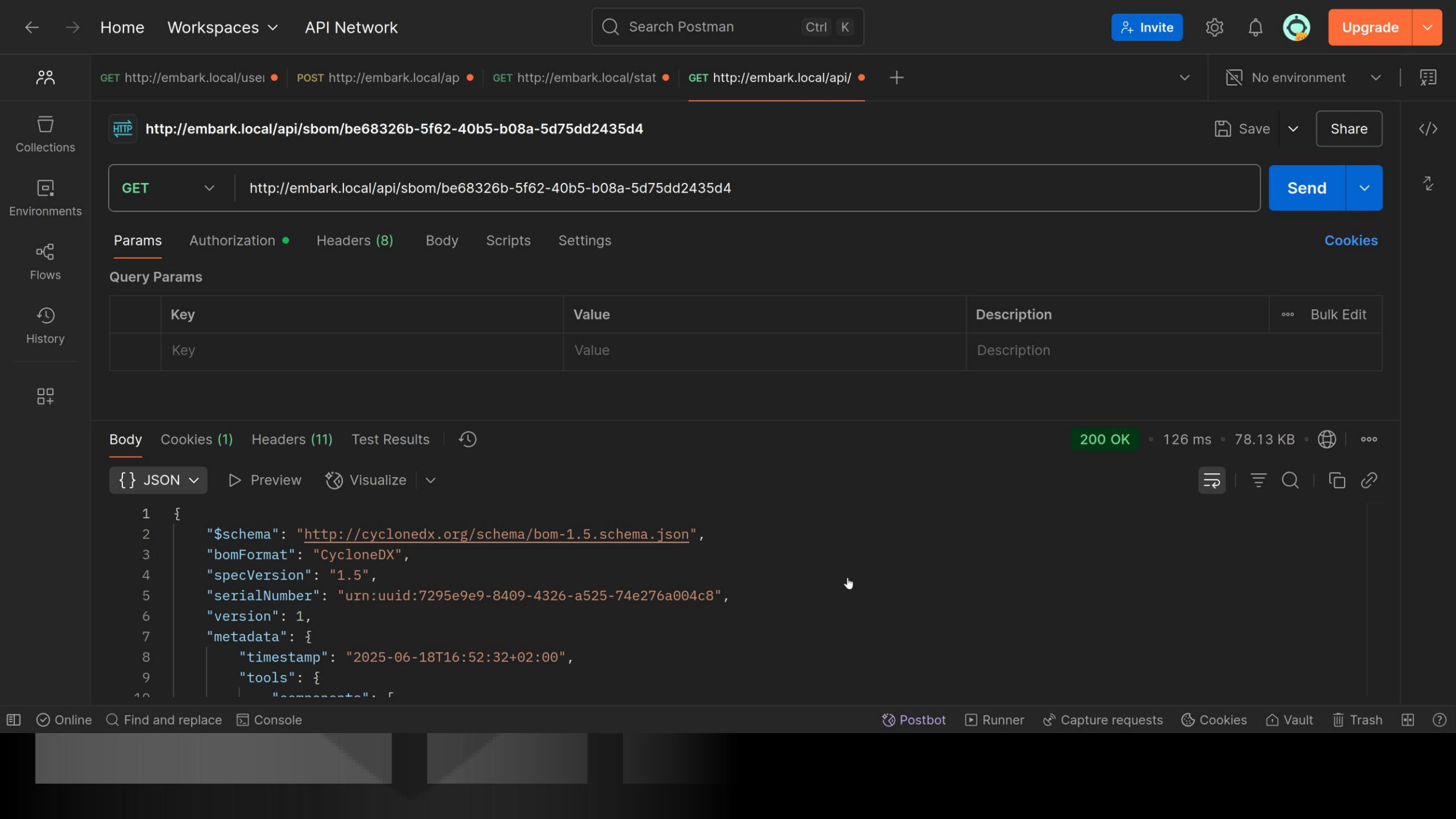
200 OK • 41 ms • 480 B • 🌐 ⋮

{} JSON ▾ ▶ Preview 🔗 Visualize ▾

🔄 ≡ 🔍 📄 🔗

```
1 {
2   "status": "finished",
3   "message": "Analysis finished successfully in 3:01:58.",
4   "download_url": "http://embark.local/download_zipped/be68326b-5f62-40b5-b08a-5d75dd2435d4/"
5 }
```





GET http://embark.local/user ● POST http://embark.local/ap ● GET http://embark.local/stat ● GET http://embark.local/api/ ● +

🗑️ No environment ▾ 📄



Collections



Environments



Flows



History



Grid

HTTP http://embark.local/api/sbom/be68326b-5f62-40b5-b08a-5d75dd2435d4

💾 Save ▾ ➦ Share

</>

GET http://embark.local/api/sbom/be68326b-5f62-40b5-b08a-5d75dd2435d4

Send ▾ ↻

Params Authorization ● Headers (8) Body Scripts Settings

Cookies

### Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

Body Cookies (1) Headers (11) Test Results ↻

200 OK • 126 ms • 78.13 KB • 🌐 ⋮

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

🔄 ⋮ 🔍 📄 🔗

```
1 {
2   "$schema": "http://cyclonedx.org/schema/bom-1.5.schema.json",
3   "bomFormat": "CycloneDX",
4   "specVersion": "1.5",
5   "serialNumber": "urn:uuid:7295e9e9-8409-4326-a525-74e276a004c8",
6   "version": 1,
7   "metadata": {
8     "timestamp": "2025-06-18T16:52:32+02:00",
9     "tools": {
10      "components": [
```



```
"$schema": "http://cyclonedx.org/schema/bom-1.5.schema.json",
"bomFormat": "CycloneDX",
"specVersion": "1.5",
"serialNumber": "urn:uuid:7295e9e9-8409-4326-a525-74e276a004c8",
"version": 1,
"metadata": {
  "timestamp": "2025-06-18T16:52:32+02:00",
  "tools": {
    "components": [
      {
        "type": "application",
        "author": "EMBA community",
        "name": "EMBA binary analysis environment",
        "version": "1.5.2-8c059144ab6f440b8b81ec433efaf55425ab10f5",
        "description": "EMBA firmware analyzer - https://github.com/e-m-b-a/emba"
      }
    ]
  },
},
"component": {
  "name": "/var/www/active/be68326b-5f62-40b5-b08a-5d75dd2435d4/DIR600B6_FW215WWb02.bin",
  "type": "data",
  "bom-ref": "34ac94c5-df3e-4cae-99f1-689fb3484d7b",
  "hashes": [
    {
      "alg": "MD5",
      "content": "c0d364001d5b1e66e98a697596c5aa6b"
    },
    {
      "alg": "SHA-256",
      "content": "0fc68a60bd7206be34c7cd4f21b2b51e3da32f69ddd6d08941ecdc72626f471d"
    },
    {
      "alg": "SHA-512",
      "content": "f60d87b246bf523ec971c01cdcb7d573f6e40e44ba6b08b2f971da6d07b9b00ea3b9ce7991d8cdf9eafbbbee18589f2a75eec13fddc76c6bc8da06e1492fc59ad"
    }
  ],
},
"supplier": {
  "name": "",
  "url": [
    "https://github.com/e-m-b-a/emba"
  ],
},
"lifecycles": [
  {
    "phase": "operations"
  }
],
}
```

# Fightclub ... Some observations

## Syft – the top dog



CLI tool only



Easy to install



Very fast (fastest I've seen)



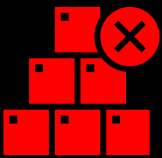
Lightweight (but SBOM only)



Many package managers



Very good SBOM



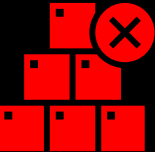
No VEX (post-processing needed)

No extraction (pre-processing)

No binary SBOM

# Fightclub ... Some observations

## Syft – the top dog

- ✓ CLI tool only
- ✓ Easy to install
- ✓ Very fast (fastest I've seen)
- ✓ Lightweight (but SBOM only)
- ✓ Many package managers
- ✓ Very good SBOM
-  No VEX (post-processing needed)
- No extraction (pre-processing)
- No binary SBOM

## EMBArk – the challenger

- ✓ CLI tool / Web interface / API
- ✓ Easy to install (but a big beast)
- ✓ Not so „Very fast“
- ✓ Includes firmware analysis
- ✓ Many package managers
- ✓ Very good SBOM
- ✓ Includes VEX (via scanning profile)
- ✓ Binary blob extraction included
- ✓ Enhanced binary analysis/SBOM

# Fightclub ... Further observations

- **False positives in the SBOM will hit you hard**
  - We have seen solutions with more than 20% of false (non existing) components
  - You are going to hunt for vulnerabilities your product is not affected
- **Missed components result in blind spots in your triage process**
- **Common Identifiers for further processing stay hard (component identifier/CPE/PURL/...)**



# Don't trust your SBOM

Challenge it

<https://github.com/e-m-b-a>