Aggregators in software supply chain
2023 03 08 KTH

Hans Thorsen Lamm
Lamm Consulting AB
hans@lammda.se

https://github.com/Nordix/bomres

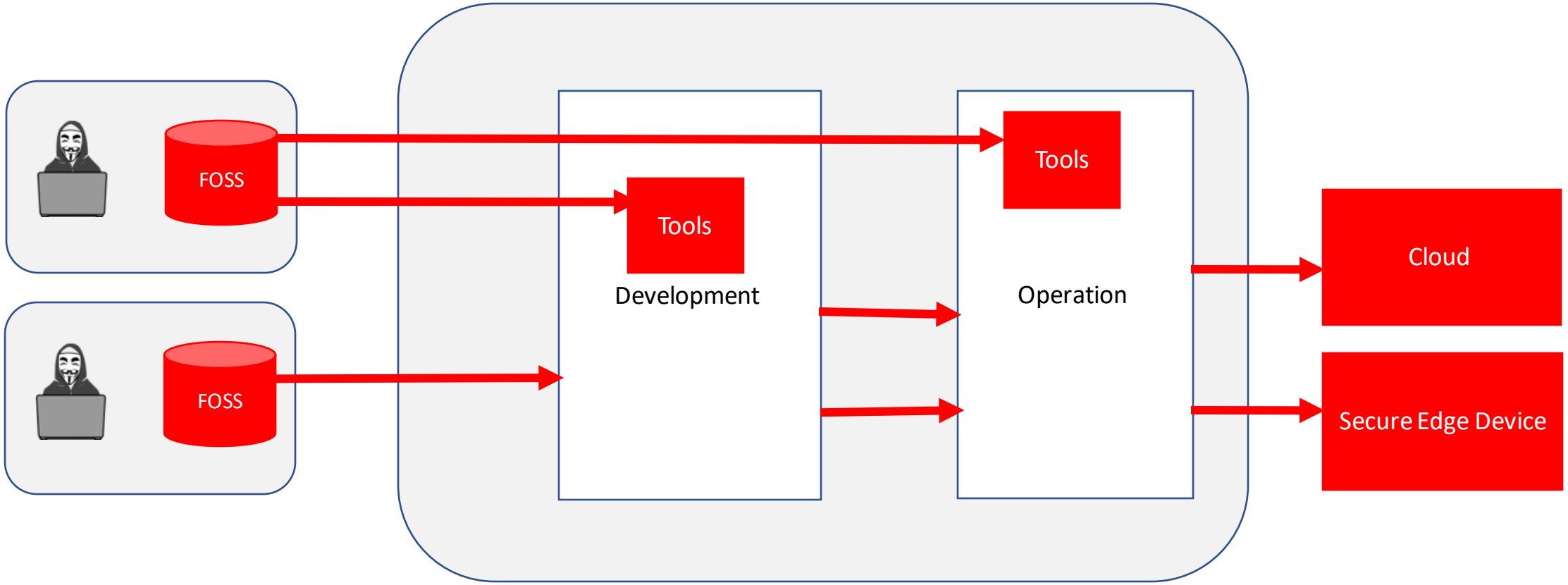
https://kth-se.zoom.us/j/64902093977

Agenda

- Problem
  - Security
  - Emerging regulations
    - EU Cybersecurity Resilience Act
- Bom Resolver
  - Build framework versus "Aggregators"
  - Backtrack the Alpine ECO system
- Demo
  - Oneline binary component in SBOM
  - Complete SBOM with product source and tools required
  - Rebuild binary component

Problem

Keep cost down -> Introduce Open Source -> Supply chain attacks



Foss in the supply chain is a reality.

How open-source software took over the world | TechCrunch

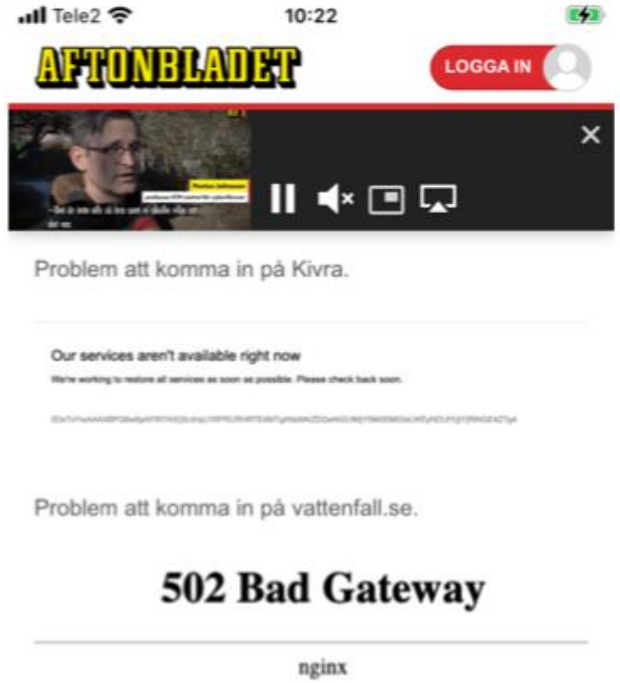
Foss may contain bugs, some impacts security

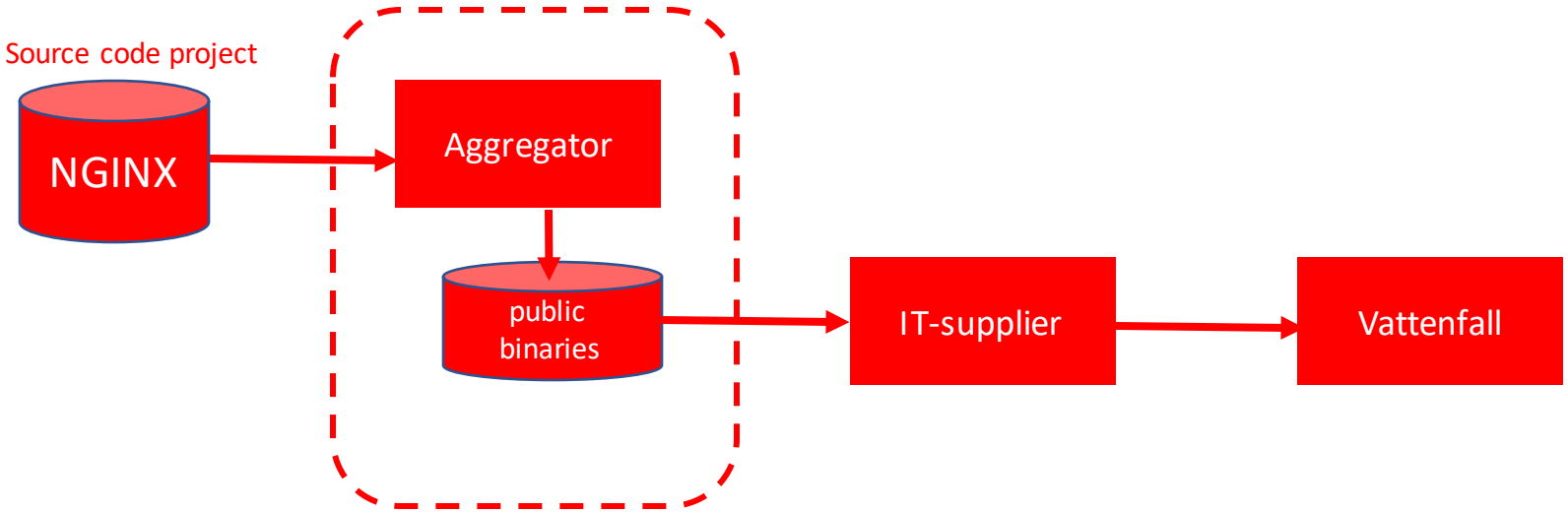
Initial Access, Tactic TA0001 - Enterprise | MITRE ATT&CK®

Nginx and Vattenfall under attack

[Open Source Software | RISE]

Open Source Software (OSS) constitutes a critical building block of our common digital infrastructure. About 90 % of today's software contains OSS. The amount of code in companies' codebases made up of OSS has increased from 36 % in 2015 to 75 % in 2020.

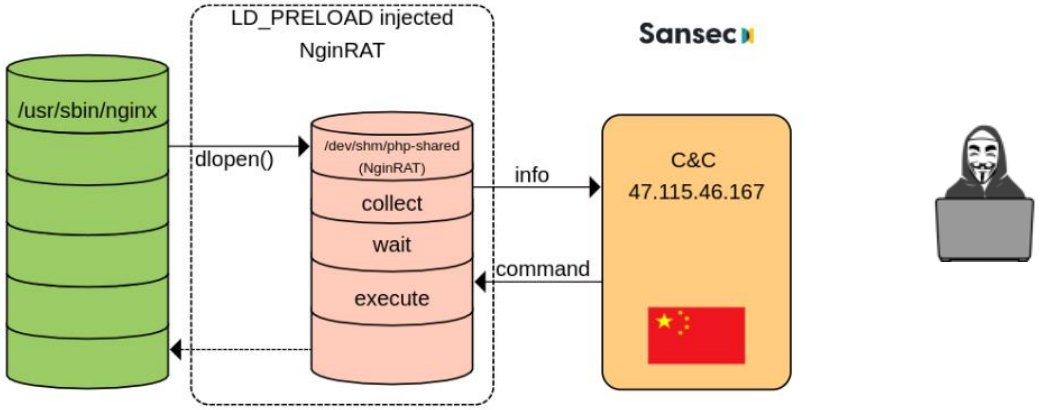




Vattenfall uses ngnix
Bad gateway indicates reverse proxy setup.
Let`s check for vulnerabilities

Nginx Vulnerabilities

1 - Top 5 Most Critical NGINX Vulnerabilities Found - Astra Security Blog (getastra.com)
Updating NGINX for Vulnerabilities in the MP4 and HLS Video-Streaming Modules - NGINX
NginRAT parasite targets Nginx – Sansec



Many security issues related to plugins loaded during runtime (dlopen)
Recompile to include module functionality in static binary

Attack by patching NGINX

```
source="https://nginx.org/download/$pkgname-$pkgver.tar.gz
        ...
        nginx-dav-ext-module~pr-56.patch::https://github.com/arut/nginx-dav-ext-module/pull/56.patch
        ...
```

Roman Arutyunyan is a member of the core development team at NGINX.

Source code patches are not listed in File, Data Source DS0022 | MITRE ATT&CK®

"Trust but verify"

```
--- foss.c.orig 2022-01-24 10:54:50.226384276 +0100
+++ foss.c      2022-01-24 11:27:51.058541111 +0100
@@ -2,6 +2,6 @@

 int main()
 {
- printf("Buggy processing\n");
+ printf(”Evil processing\n");
  return 0;
 }
```

arut github.com

```
#include <stdio.h>

int main()
{
 printf("Buggy processing\n");
 return 0;
}
```

patch foss.c < foss.patch

```
#include <stdio.h>

int main()
{
 printf(“>Evil processing\n");
 return 0;
}
```

Mitre ATT&CK

In February 2019, NGINX finally dethroned Apache HTTPD and became the most widely deployed server on the internet. According to the Netcraft December 2019 Web Server Survey, NGINX has market share of 38%.

"Russian police have raided today the Moscow offices of NGINX, Inc., a subsidiary of F5 Networks and the company behind the internet's most popular web server technology.
Equipment was seized and employees were detained for questioning."

RE: Regarding ATT&CK request 10909

HL **Hans Thorsen Lamm <hans@lammda.se>**
2023-01-28 16:27

To: ATTACK

Hi Jamie,

On a high level patches are covered in T1195/001. But there are two kind of patches

- Binary patches
- Source code patches

Ngingx is a very popular internet facing component originating from Russia.
I have analysed the Alpine build framework and it starts by unpacking the primary source and then apply patches from private GitHub accounts.

https://git.alpinelinux.org/aports
main/nginx/APKBUILD

In the source section all external dependencies are listed, and you see several patches from two private github acounts.

source="https://nginx.org/download/$pkgname-$pkgver.tar.gz

nginx-dav-ext-module~pr-56.patch::https://github.com/arut/nginx-dav-ext-module/pull/56.patch
nginx-dav-ext-module~pr-62.patch::https://github.com/arut/nginx-dav-ext-module/commit/bbf93f75ca58657fb0f8376b0898f854f13cef91.patch

nginx-module-vts~01-938c19d.patch::https://github.com/vozlt/nginx-module-vts/commit/938c19d2e49d5f3355df5375725982d15f1270c4.patch
nginx-module-vts~02-ad40022.patch::https://github.com/vozlt/nginx-module-vts/commit/ad4002262c19e81390f518a14f99bb594862c575.patch
nginx-module-vts~03-c08781c.patch::https://github.com/vozlt/nginx-module-vts/commit/c08781c5095d9e6090c47176bdea322ce983ecb6.patch
nginx-module-vts~04-1a01a87.patch::https://github.com/vozlt/nginx-module-vts/commit/1a01a87e66c9f111fb399cf40def33ab193ae393.patch
nginx-module-vts~05-ead62a0.patch::https://github.com/vozlt/nginx-module-vts/commit/ead62a0caf405731c88ac138c4c0a82bb316cc18.patch

The Data Source DS0022 is more detailed than T1195/001, but I am not sure if source code patches are included.

Maybe it is enough to just add source code patches as a data source in DS0022 and keep the T1195/001 as is.

I wish you a great weekend too.

// Hans

Russian police raid NGINX Moscow office | ZDNET

In addition to security we have emerging legal obligations

Manufacturer's obligations under the EU Cyber Resilience Act. (Image: European Commission)

Released	Security Support
3 months and 2 weeks ago (22 Nov 2022)	Ends in 1 year and 8 months (22 Nov 2024)
9 months ago (23 May 2022)	Ends in 1 year and 2 months (23 May 2024)
1 year and 3 months ago (24 Nov 2021)	Ends in 7 months and 4 weeks (01 Nov 2023)
1 year and 8 months ago (15 Jun 2021)	Ends in 1 month and 3 weeks (01 May 2023)
2 years ago (14 Jan 2021)	Ended 4 months ago (01 Nov 2022)
2 years and 9 months ago (29 May 2020)	Ended 10 months ago (01 May 2022)
3 years ago (19 Dec 2019)	Ended 1 year and 4 months ago (01 Nov 2021)
3 years and 8 months ago (19 Jun 2019)	Ended 1 year and 10 months ago (01 May 2021)
4 years ago (29 Jan 2019)	Ended 2 years ago (01 Jan 2021)
4 years and 8 months ago (26 Jun 2018)	Ended 2 years and 10 months ago (01 May 2020)
5 years ago (30 Nov 2017)	Ended 3 years and 4 months ago (01 Nov 2019)

[EU Cyber Resilience Act: the GDPR for IoT - Embedded.com]

[Alpine Linux | endoflife.date]

Software Bill Of Material

MAY 12, 2021

Executive Order on Improving the Nation's Cybersecurity

Sec 4 – Enhancing Software Supply Chain Security – mandates that the government take action to protect software – with a focus on "critical software" – against cyber-attacks.

•Within 30 days of the Order, the government will solicit input from various sources, including the private sector, regarding standards, procedures, and criteria for software security (including for a Software Bill of Materials ("SBOM")).

[Funding to secure 10000 Open source projects]

The Linux Foundation and OpenSSF project,
with backing from Microsoft and Google,
aims to improve security of **10,000** open-source projects

[My question discussed starting ~ 40 minutes into recording]

[Steve Springett]

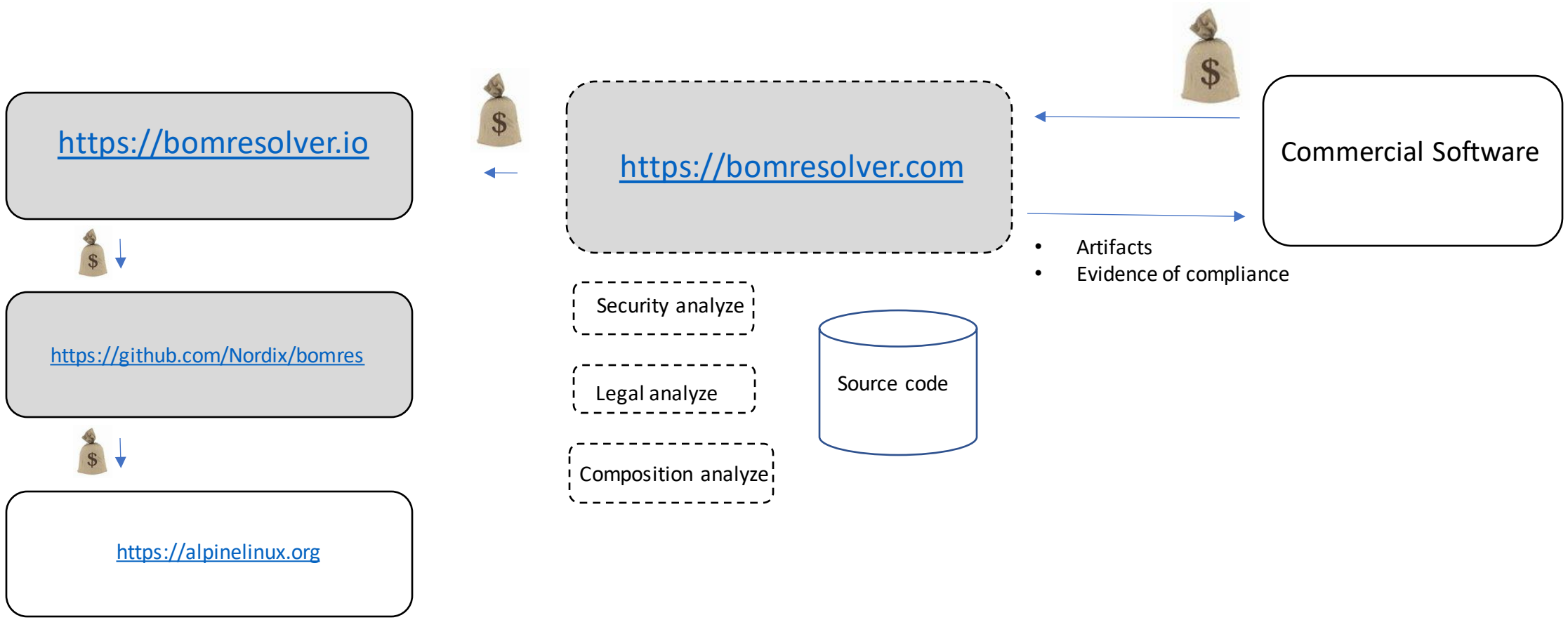
Bom Resolver

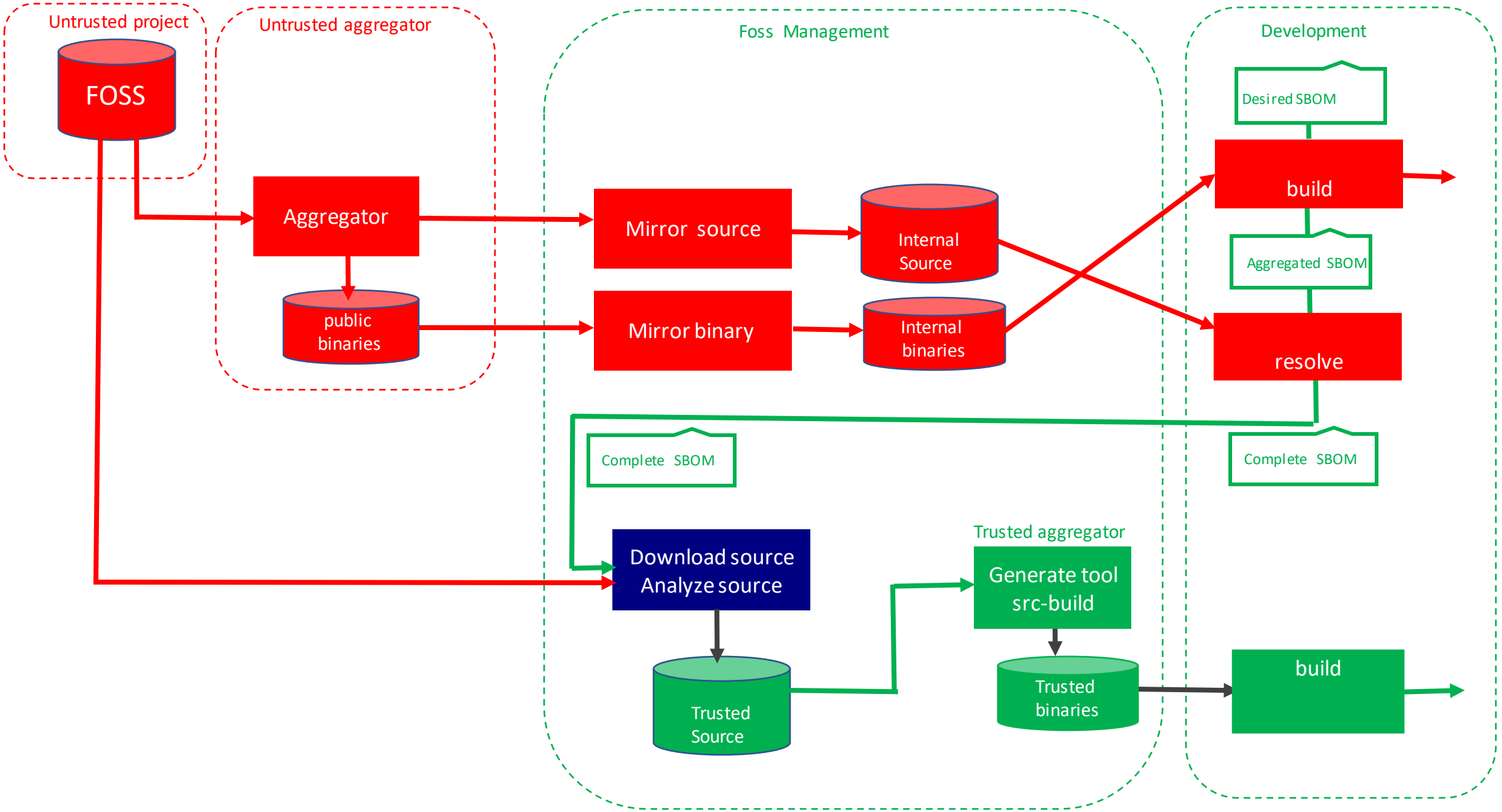
- Bridge the gap between
  - Foss culture (Cool stuff)
  - Compliance culture (Boring stuff)

- Always source code, no exemption
- Rebuild in isolation -> Correctness of SBOM
- No binaries whatsoever

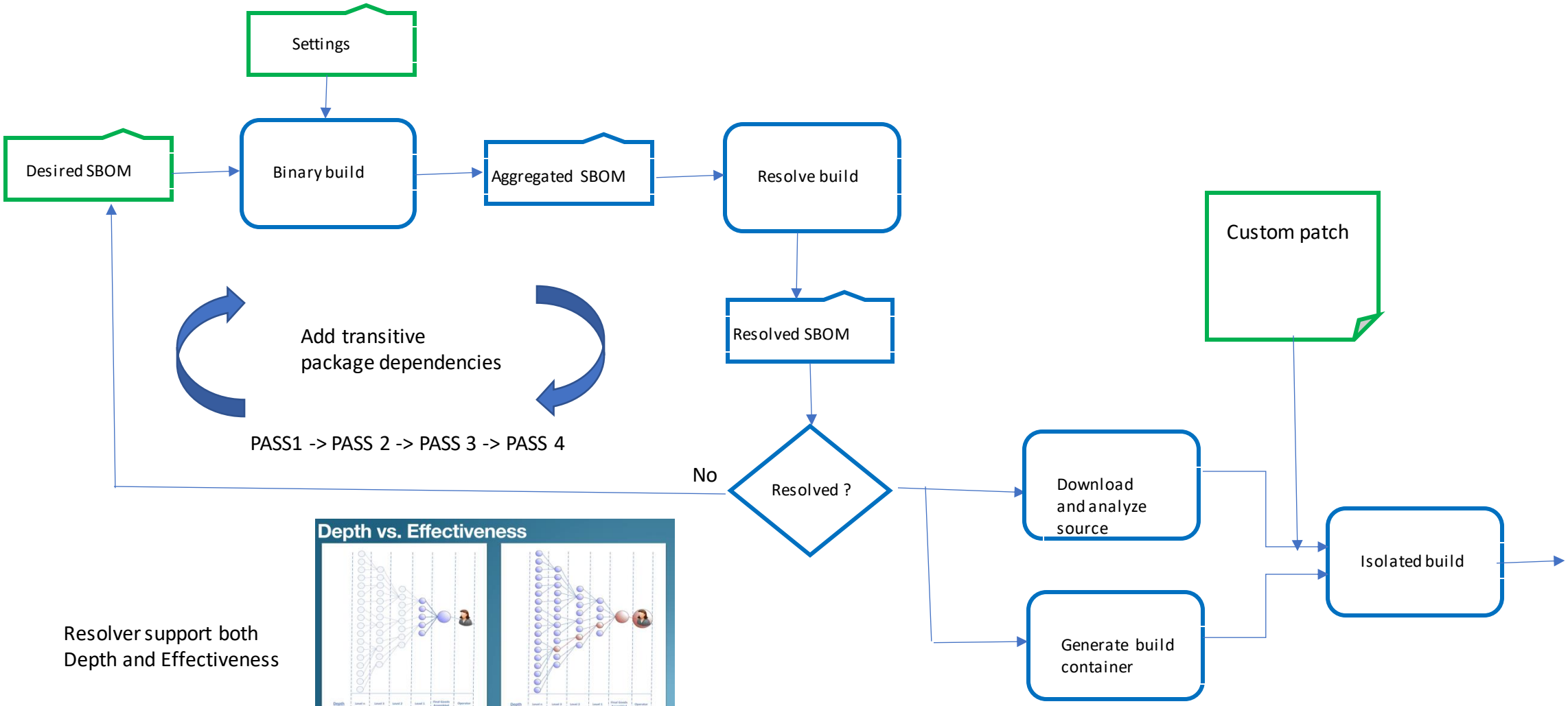
- 2018 Automate **builder** of microservice based on Alpine
- 2021 Added **resolver** as part of Internal Integrity program
- 2022 Released as Open source , presented on FOSDEM
- 2023 Received first funding 50.000 SEK from Sidiatech

BomResolver




Untrusted project
FOSS
Untrusted aggregator
Aggregator
public binaries
Foss Management
Mirror source
Mirror binary
Internal Source
Internal binaries
Development
Desired SBOM
build
Aggregated SBOM
resolve
Complete SBOM
Complete SBOM
Download source
Analyze source
Trusted aggregator
Generate tool
src-build
Trusted Source
Trusted binaries
build

Bom Resolver Flow



Resolver support both
Depth and Effectiveness

Home - Lighttpd - fly light

Alpine as a aggregator of lighttpd includes many features

Buildroot is based on rules that excludes features not needed in production.

Note: for real project the dependencies of tools for building adds requirements on the tools.

https://git.alpinelinux.org/aports

```
makedepends="
        automake
        autoconf

build() {
        ./configure \
                --with-ldap \
                --with-openssl \
                --with-zstd \
                --with-brotli \
                --with-lua
        make
```

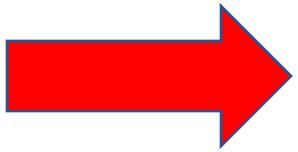
https://github.com/buildroot/buildroot.git

```
ifeq ($(BR2_PACKAGE_LIGHTTPD_LDAP),y)
  LIGHTTPD_DEPENDENCIES += openldap
  LIGHTTPD_CONF_OPTS += -Dwith_ldap=true
else
  LIGHTTPD_CONF_OPTS += -Dwith_ldap=false
endif
```

Build container required to build **product** is **generated** from **product** sbom

Product

lighttpd



Tool

Autogenerated SBOM
alpine-base
util-linux
alpine-sdk
build-base
abuild

Additional tools needed for specific packages during build
busybox
linux-headers
lighttpd
automake
lighttpd
autoconf

Additional packages needed for building the product
openssl
brotli
openldap
lua5.4
zstd

https://bomresolver.io

Combines Build System with Binary distribution

Short feedback loops
Enables us to define the
right product

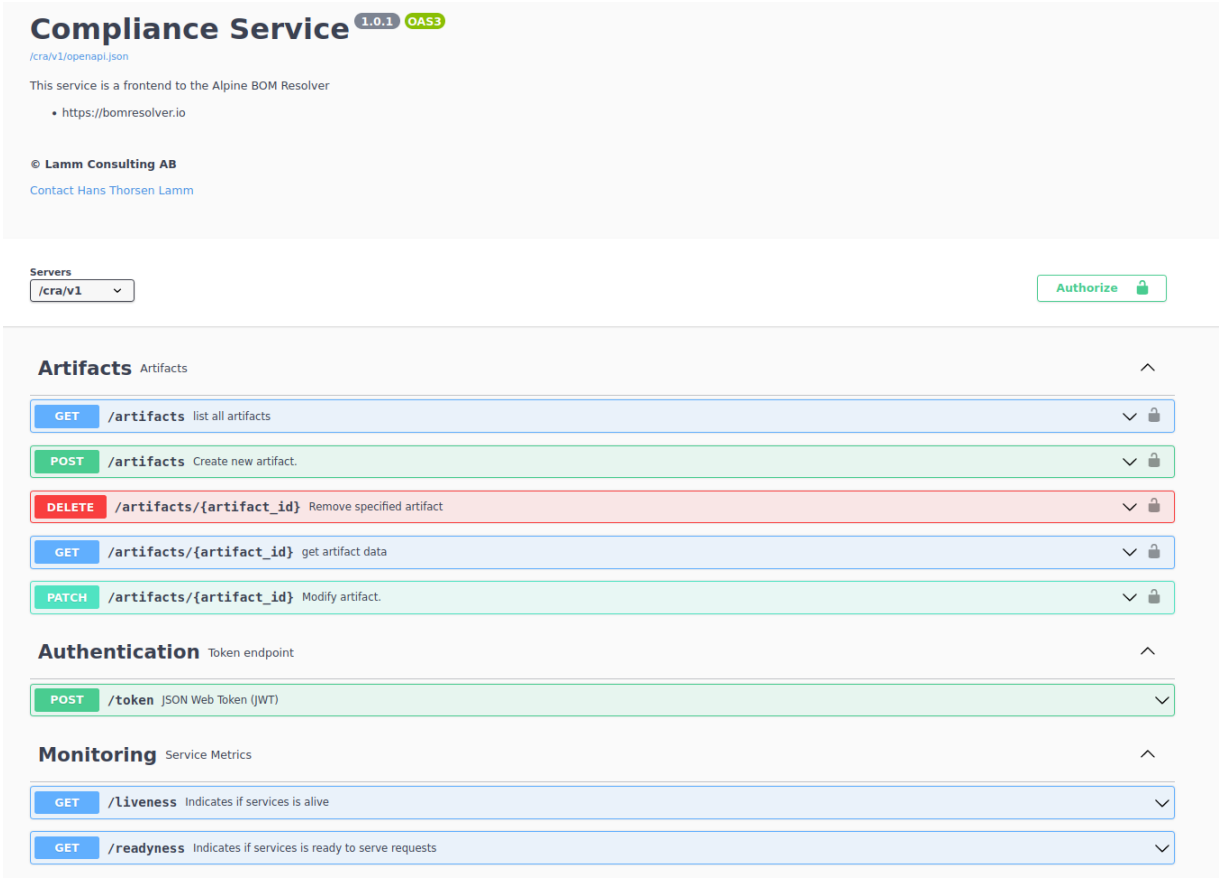
When the product is
defined, we **build it right**
(In compliance with NIS2, DORA, etc)

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

https://bootlin.com/training/buildroot/

Demo

Demo Resolver (Service)



https://bomresolver.com

Demo steps

S1: Create an access token
S2: Post a "one-line" Desired SBOM
S3: Depth / Effectiveness (CISA , slide 22)
S4: Get aggregated.json (All metadata from "binary build)
S5: Get resolved.json (contain ALL source needed for rebuild)
S8: Get standard lighttpd , check –V for module support) , ls /etc/apk/keys

S11: Patch APKBUILD for lighttpd , Applied for 1 , returns 2 (new artifact)

S12: Download custom product with build container 24 seconds)
S13: Build custom lighttpd (build 1.23 minutes)

S14: Get customized lighttpd , check –V for module support) , ls /etc/apk/keys/

CISA SBOM-A-RAMA

Takeway

- Always source code, no exemption
- No binaries whatsoever
- Bridge the gap between
  - Foss culture (Cool stuff)
  - Compliance culture (Boring stuff)
- Business model for Bom Resolver
  - Alpine resolver (Github / MIT License)
  - Bomresolver.io (Apply the concept for Go, Python etc)
  - Bomresolver.com (Compliance as a service)

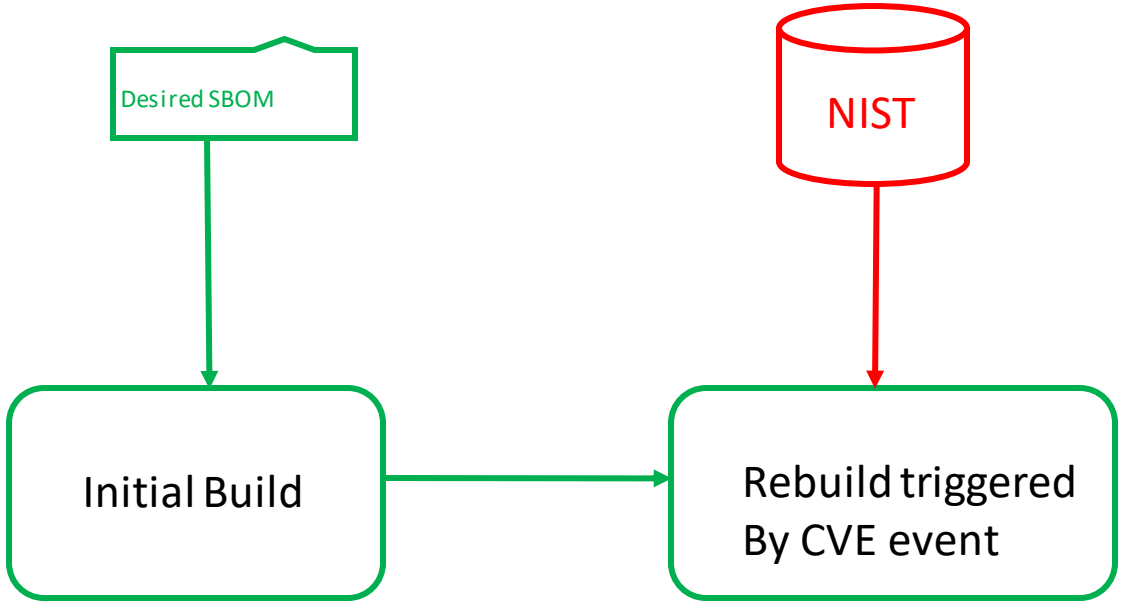
Support slides

Bom Resolver as tool

```
$ podman run  --rm docker.io/bomres/base_os_alpine make > Makefile
$ make config
$ vim product/build/base_os/config/packages
$ vim product/build/base_os/config/settings
$ make build
$ make resolve
$ make download_source
```

Bom Resolver

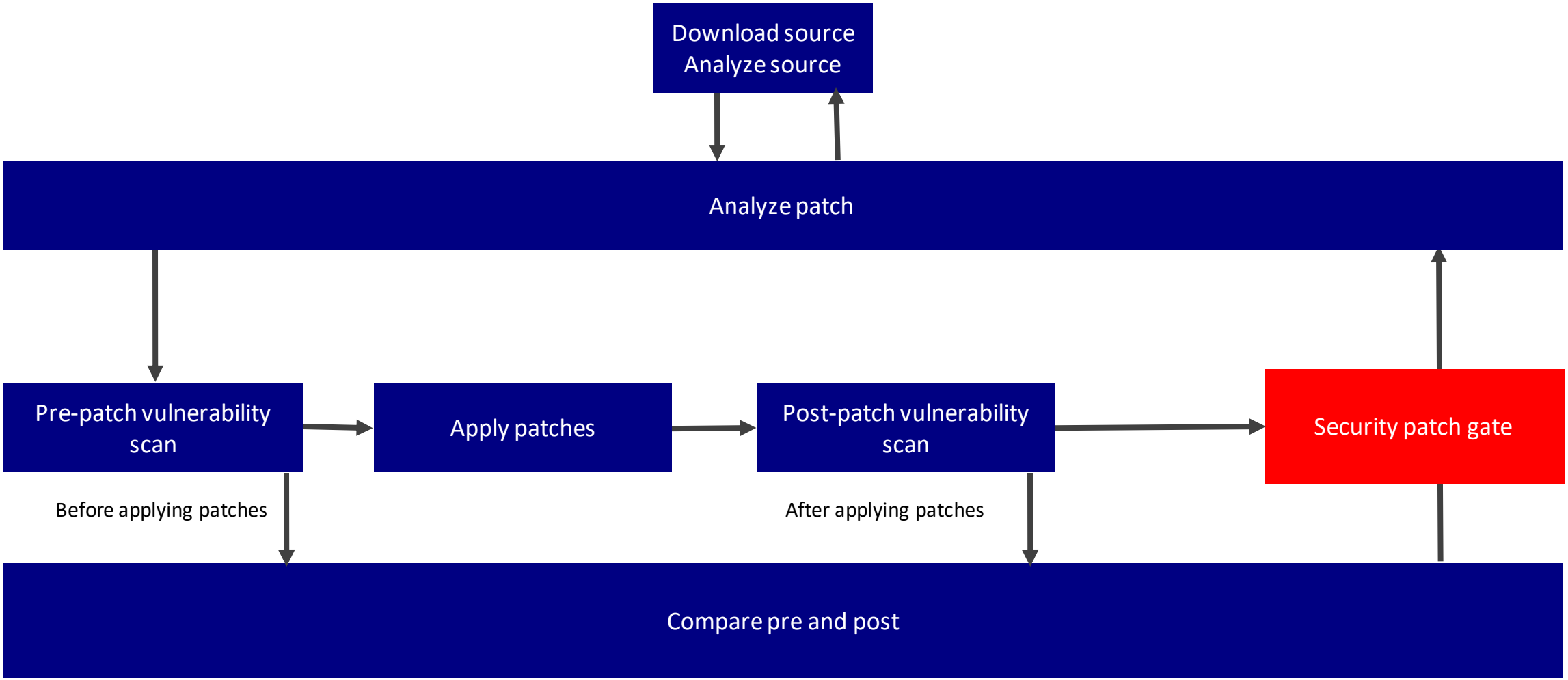
Renovate




```
"secfixes": {

        "1.2.2_pre2-r0": [
            "CVE-2020-28928"
        ],

        "1.1.23-r2": [
            "CVE-2019-14697"
        ],

        "1.1.15-r4": [
            "CVE-2016-8859"
        ]
    }
```

[Mend Renovate: Automated Dependency Updates]

Analyze patches



From mockup to market

Short feedback loops
Enables us to define the **right product**

When the product is defined, we **build it right**
(In compliance with NIS2, DORA, etc)

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well-defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

https://bootlin.com/training/buildroot/

Aggregator model

Example of aggregators are
- Redhat
- OpenSuse
- Debian
- **Alpine**

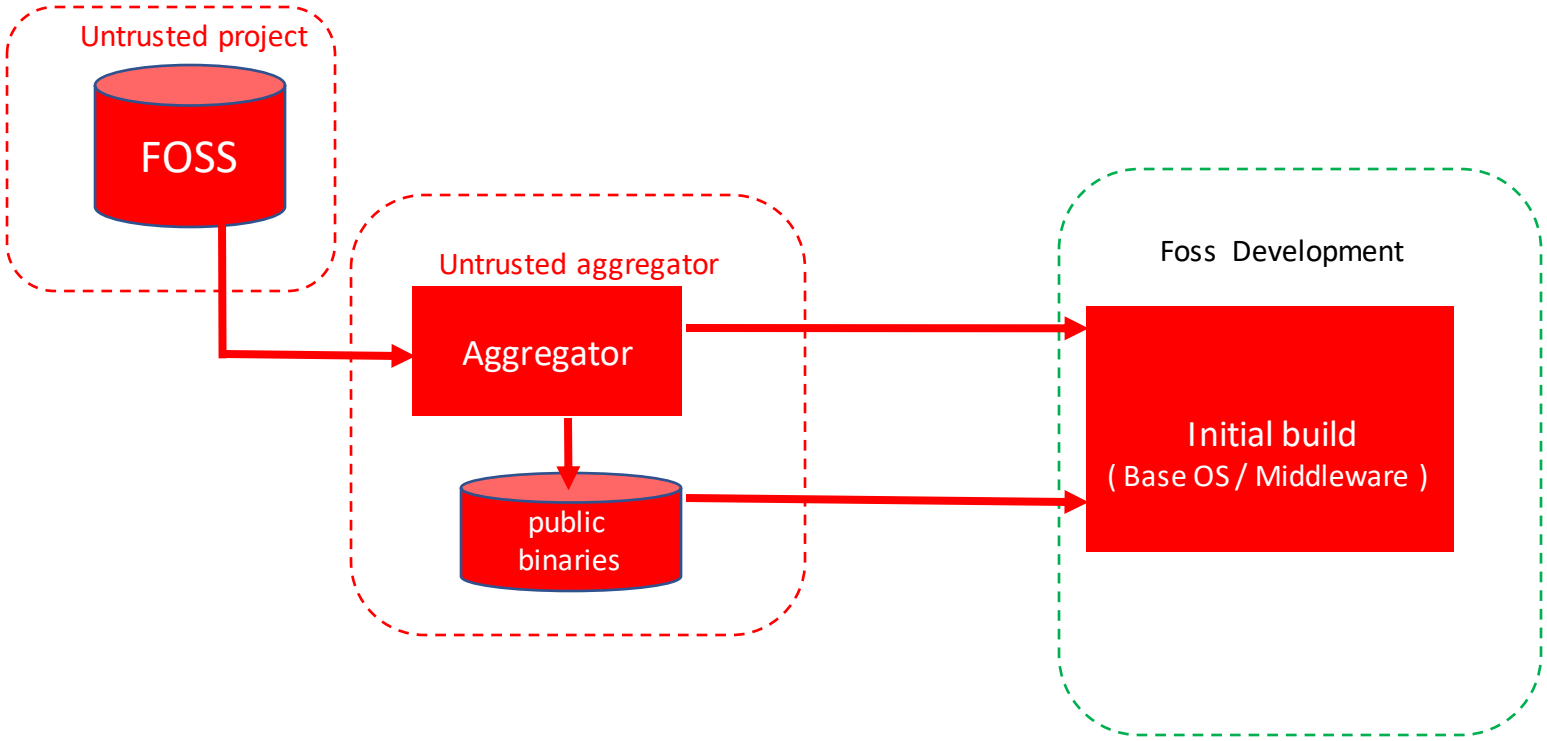
Aggregator fetch source code from Projects and creates components provided as
- Packages (rpm , apk , deb)
- ISO images
- **Containers (docker images)**

Aggregators support different architectures
- x86_64
- Arm

Aggregators improves Foss
- Add functionality
- Fix **security** issues

Short feedback loops
Enables us to define the **right product**

Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend



Take Away from Bom Resolver

Mockup phase

Product phase

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

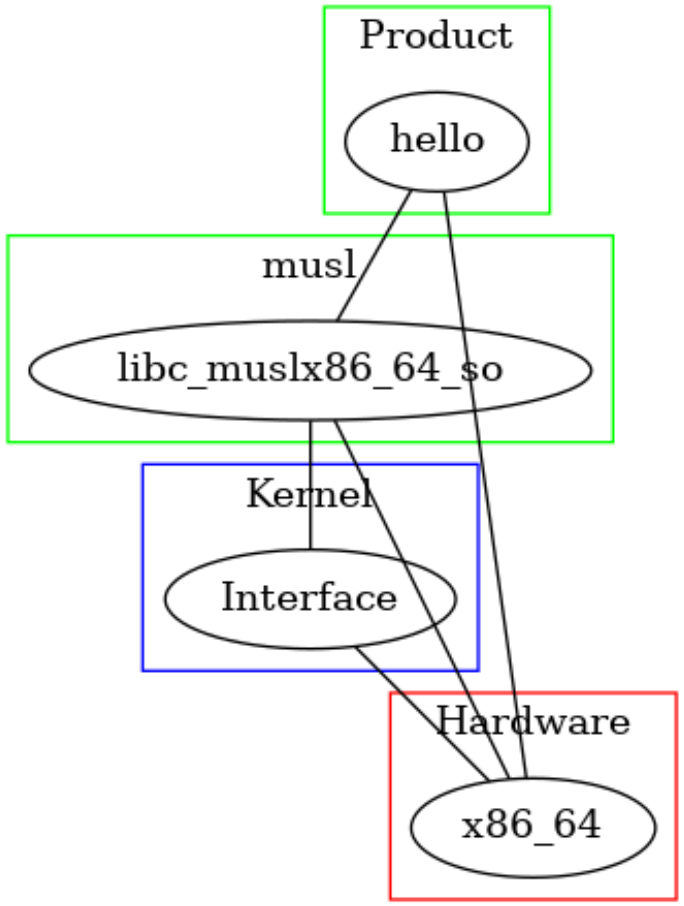
Simple product

For better understanding of the software supply chain, keep the product small and study the tools required to compile the source code

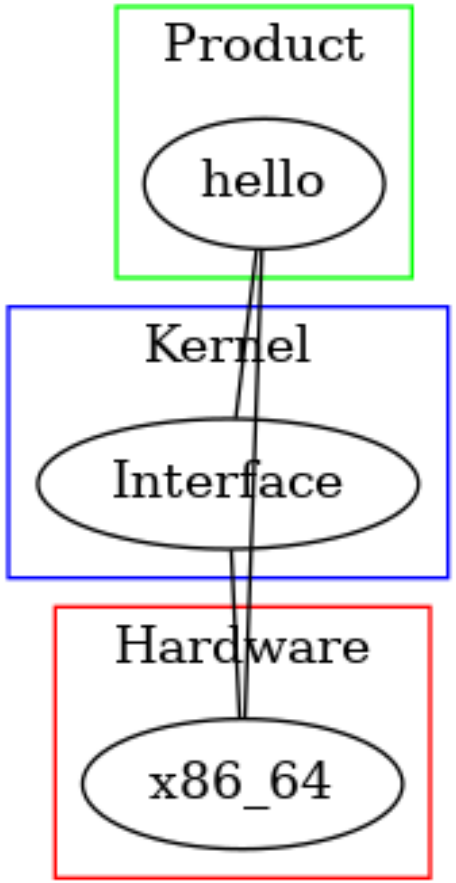
```
#include <stdio.h>

int main(void)
{
	printf("Hello World\n");
}
```

Build dependencies

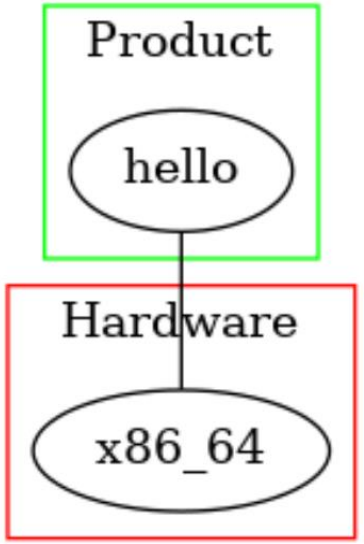
Dynamically linked build



Static linked build

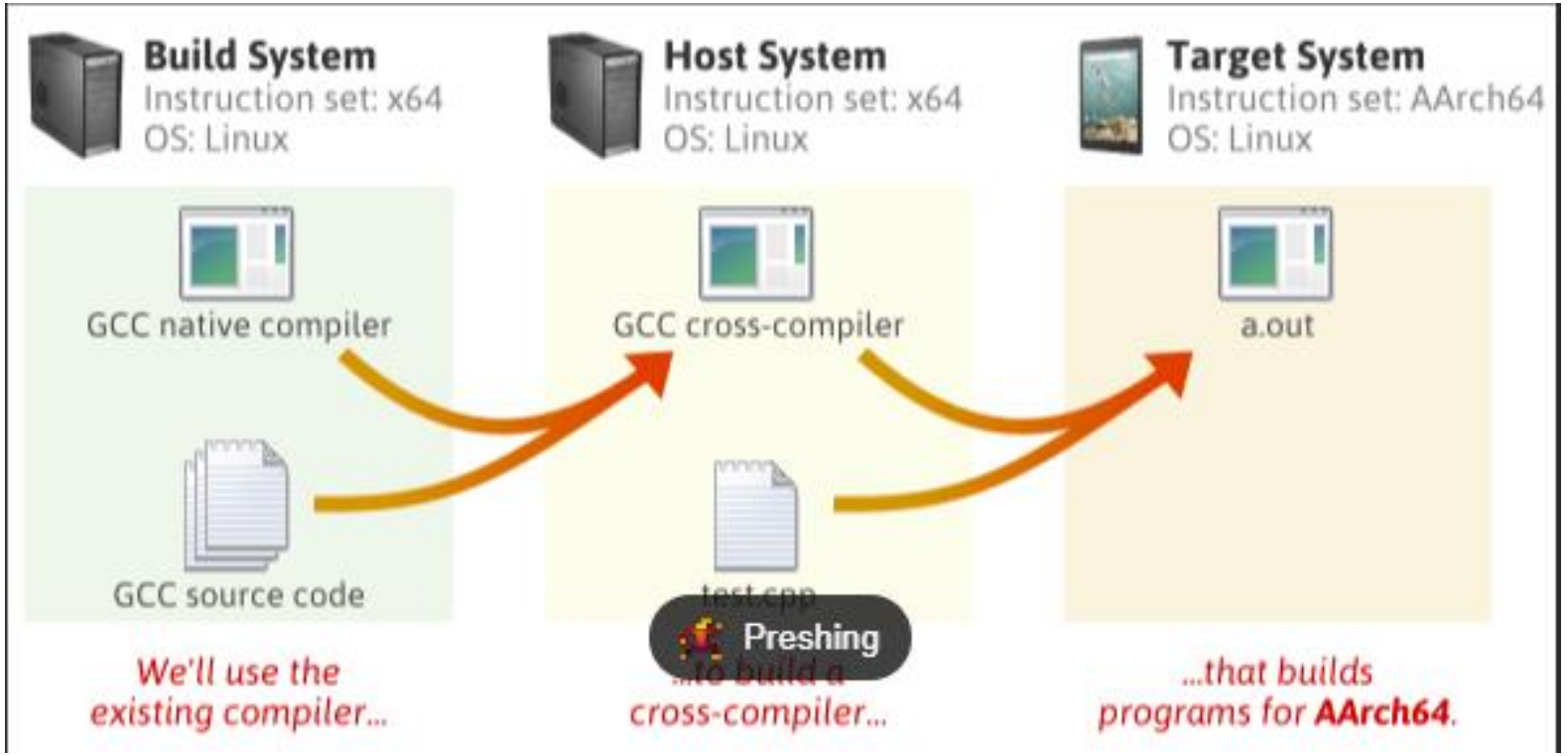


Bare metal build



(dlopen not covered)

Cross compile



[Introduction (crosstool-ng.github.io)]