SBOM et OBOM :

comment générer des inventaires logiciels pertinents pour ses clients dans le cadre du CRA ?

Maxime ALAY-EDDINE

10/03/2026

Faisons connaissance

Maxime ALAY-EDDINE

>10 ans dans la Gestion des Vulnérabilités et Correctifs

Co-fondateur de cyberwatch (exit 2022)

Co-fondateur de CoreUpdate

VP cybersécurité HEXATRUST CLOUD CONFIDENCE & CYBERSECURITY

Président du label FRANCE CYBER SECURITY

Travaux Open Source :

- Créateur de Richelieu (dictionnaire de mots de passe français)
- Contributeur Wapiti (scanner de vulnérabilités web)
- Créateur de CVE2CAPEC (générateur de MITRE ATT&CK / D3FEND)

Plan

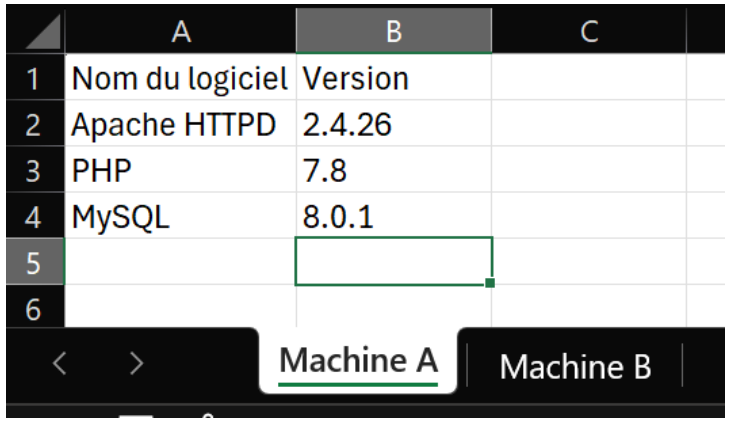
1. Présentation rapide
2. Pourquoi réaliser un inventaire logiciel ?
3. SBOM, OBOM, SaaSBOM,… XBOM : revue des différents types d’inventaires
4. Quel format d’inventaire logiciel utiliser en 2026 ?
5. Limites des inventaires logiciels
6. Cas pratique : comment générer un SBOM pour un logiciel, et pour une appliance ?

Conclusion / Q&A

2. Pourquoi faire un inventaire logiciel ?

1ère raison : prérequis de base du MCS

- La base de tout projet de **Maintien en Condition de Sécurité**, c'est l'**inventaire**.
- L'inventaire permet d'**identifier** facilement les **vulnérabilités CVE** du système d'information.
- Un bon inventaire :
  - Une **liste d'actifs** (noms des hôtes, système d'exploitation, etc)…
  - Une **liste de technologies** présentes sur ces actifs



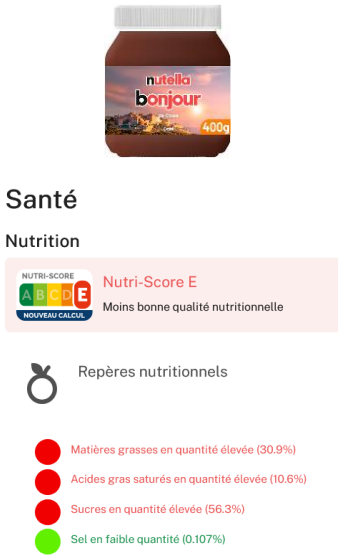
Remarque : un excellent inventaire ajoutera aussi l'OS, le rôle métier, les responsables, la documentation…

2ème raison : répondre à des exigences réglementaires, dont le CRA

- Règlement nº2024/2847 dit « Cyber Resilience Act » ou « CRA »
- Objectif : renforcer la sécurité des utilisateurs des produits numériques, avec plus de transparence
- Concerne les fabricants de « ***produits logiciels ou matériels*** *et leurs solutions de traitement de données à distance, y compris les composants logiciels ou matériels sur le marché séparément* »
- Principales obligations :
  1. Notifier aux CSIRT les vulnérabilités exploitées et incidents graves (signalement initial >24h !)
  2. Permettre le téléchargement de chaque mise à jour de sécurité pendant 10 ans
  3. Beaucoup de documentation, dont la **création d'une nomenclature des logiciels** : « *un* ***document officiel*** *contenant les* ***détails*** *et les* ***relations*** *avec la chaîne d'approvisionnement des* ***différents composants utilisés dans la fabrication d'un produit*** *comportant des éléments numériques* »
- Sanction : 15M€ ou 2.5% du C.A. mondial
- Agenda :
  - **11 septembre 2026** : début des notifications des vulnérabilités pour les fabricants
  - 11 décembre 2027 : application complète du CRA, les obligations deviennent contraignantes

Le CRA expliqué en plus simple

Le Cyber Resilience Act introduit, pour les fabricants, des obligations semblables au Nutri-score : il s'agit de fournir la liste des ingrédients, et les dangers associés.

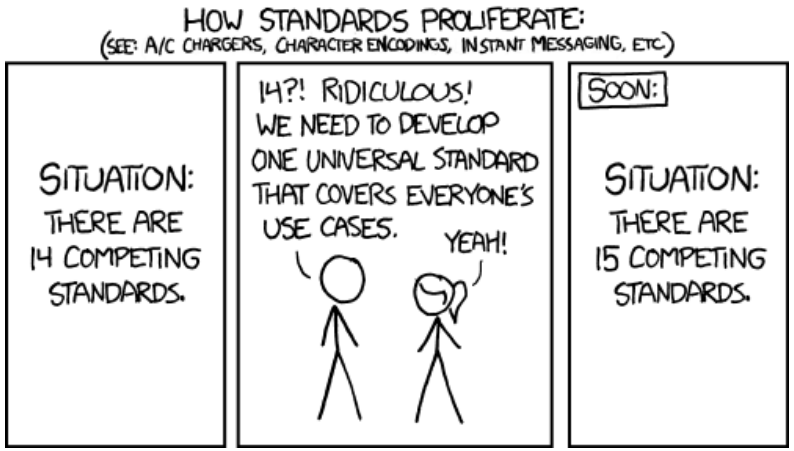


elasticsearch

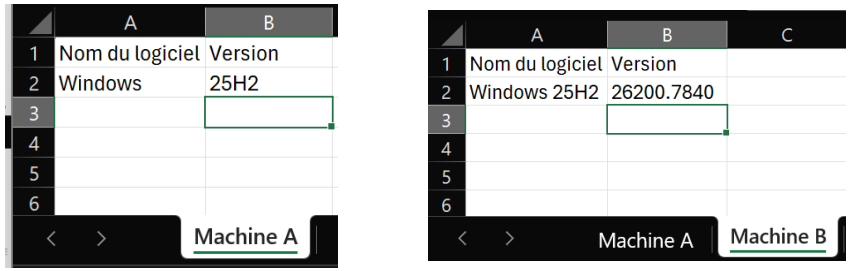
version 5.6.12
JDK 8 + Log4J 2.11.1

Elasticsearch	JDK	CVE IDs	Information Leak	Remote Code Execution
6.0.0 - 6.3.2	< 9	CVE-2021-44228, CVE-2021-45046	Yes	No
5.6.11 - 5.6.16	8	CVE-2021-44228, CVE-2021-45046	Yes	Yes

Limite des nomenclatures logicielles : à chacun son standard



- Principal outil de nomenclature logicielle :

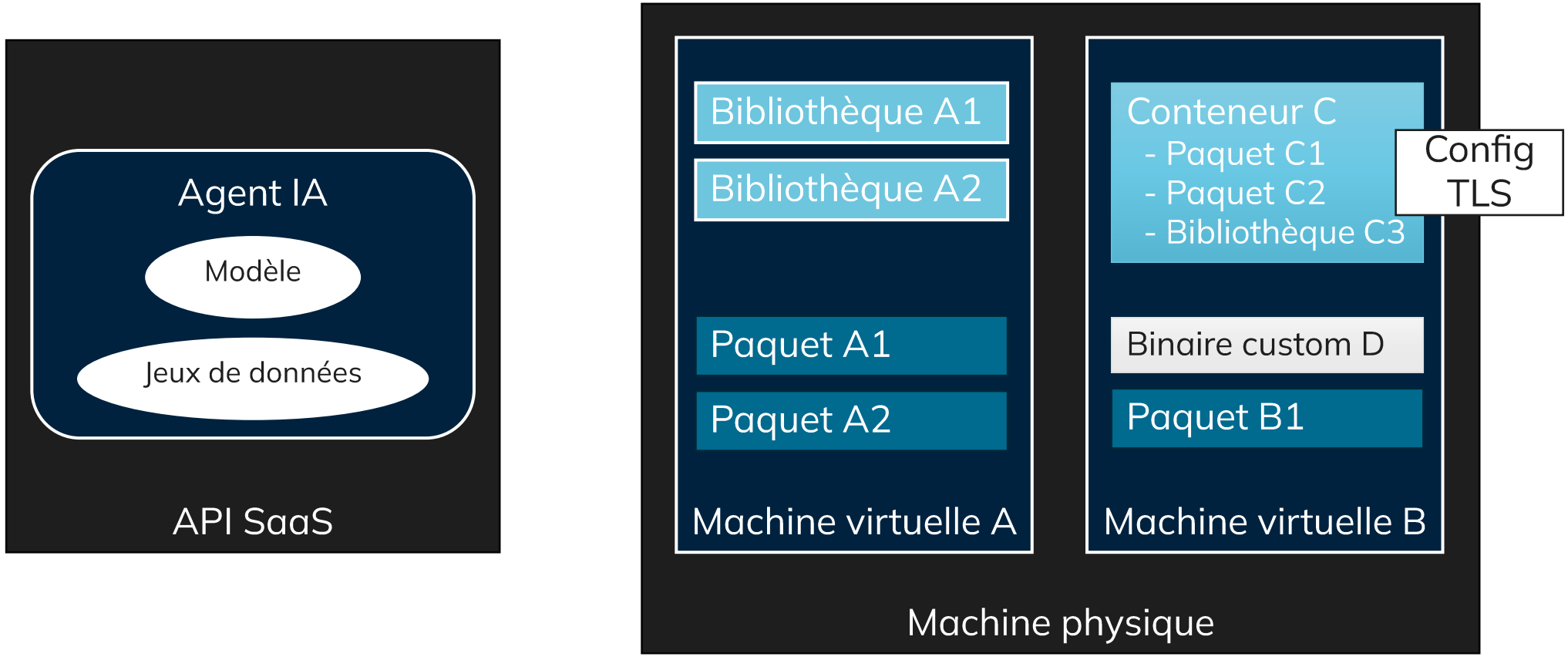


Le problème n'est pas simple : comment gérer le SaaS ? Les logiciels conteneurisés ? Les appliances virtuelles ? Faut-il aussi inclure le matériel lorsqu'il s'agit d'une appliance physique ? Faut-il inclure l'OS du conteneur ? Quid de l'OS de l'hôte ?

3. SBOM, OBOM, SaaSBOM,... XBOM : revue des différents types d'inventaires

XBOM – Les « X Bills of Materials »

Les « Bills of Materials », ou nomenclatures, décrivent des systèmes plus ou moins complexes.
Illustrons cela avec une application ayant de nombreux composants.



SBOM – Software BOM – Nomenclature logicielle

Le **SBOM** décrit un ensemble de **bibliothèques logicielles**, ou une **image Docker**.
Le SBOM est en général **statique**, et **lié à une version d'un produit**.

Bibliothèque A1

Bibliothèque A2

Liste de modules Go, PIP, Gem, NPM…

Conteneur C
- Paquet C1
- Paquet C2
- Bibliothèque C3

Liste de modules Go, PIP, Gem, NPM…

Liste de paquets DEB, RPM…

Exemples de SBOM CycloneDX et SPDX


```
"components": [
  {
    "type": "library",
    "bom-ref": "pkg:npm/body-parser@1.19.0",
    "name": "body-parser",
    "version": "1.19.0",
    "description": "Node.js body parsing middleware",
    "hashes": [
      {
        "alg": "SHA-1",
        "content": "96b2709e57c9c4e09a6fd66a8fd979844f69f08a"
      }
    ],
    "licenses": [
      {
        "license": {
          "id": "MIT"
        }
      }
    ],
    "purl": "pkg:npm/body-parser@1.19.0",
    "externalReferences": [
      {
        "type": "website",
        "url": "https://github.com/expressjs/body-parser#readme"
      },
      {
        "type": "issue-tracker",
        "url": "https://github.com/expressjs/body-parser/issues"
      },
      {
        "type": "vcs",
        "url": "git+https://github.com/expressjs/body-parser.git"
      }
    ]
  }
```

```
    {
        "type": "software_Package",
        "spdxId": "urn:npm-elliptic-6.5.2-4fe40e24-20e3-11ee-be56-0242ac120002",
        "creationInfo": "_:creationinfo",
        "name": "npm-elliptic",
        "software_packageVersion": "6.5.2",
        "suppliedBy": "urn:github.com-indutny-c4fe40e24-20e3-11ee-be56-0242ac120002",
        "software_primaryPurpose": "library",
        "externalIdentifier": [
            {
                "type": "ExternalIdentifier",
                "externalIdentifierType": "other",
                "identifier": "https://github.com/indutny/elliptic/releases/tag/v6.5.2"
                        }
                    ]

    },
    {
        "type": "software_Package",
        "spdxId": "urn:container-alpine-latest-sha256:69665d02cb32192e52e07644d76bc6f25abeb5410edc1c7a81a10ba3f0efb90a-4fe40e24-20e3-11ee-be56-0242ac120002",
        "creationInfo": "_:creationinfo",
        "name": "alpine:latest",
        "software_packageVersion": "69665d02cb32192e52e07644d76bc6f25abeb5410edc1c7a81a10ba3f0efb90a",
        "suppliedBy": "urn:github.com-alpinelinux-4fe40e24-20e3-11ee-be56-0242ac120002",
        "software_primaryPurpose": "container"
    },
    {
        "type": "software_Package",
        "spdxId": "urn:openssl-3.0.4-4fe40e24-20e3-11ee-be56-0242ac120002",
        "creationInfo": "_:creationinfo",
        "name": "openssl",
        "software_packageVersion": "3.0.4",
        "software_primaryPurpose": "library"
    },
```

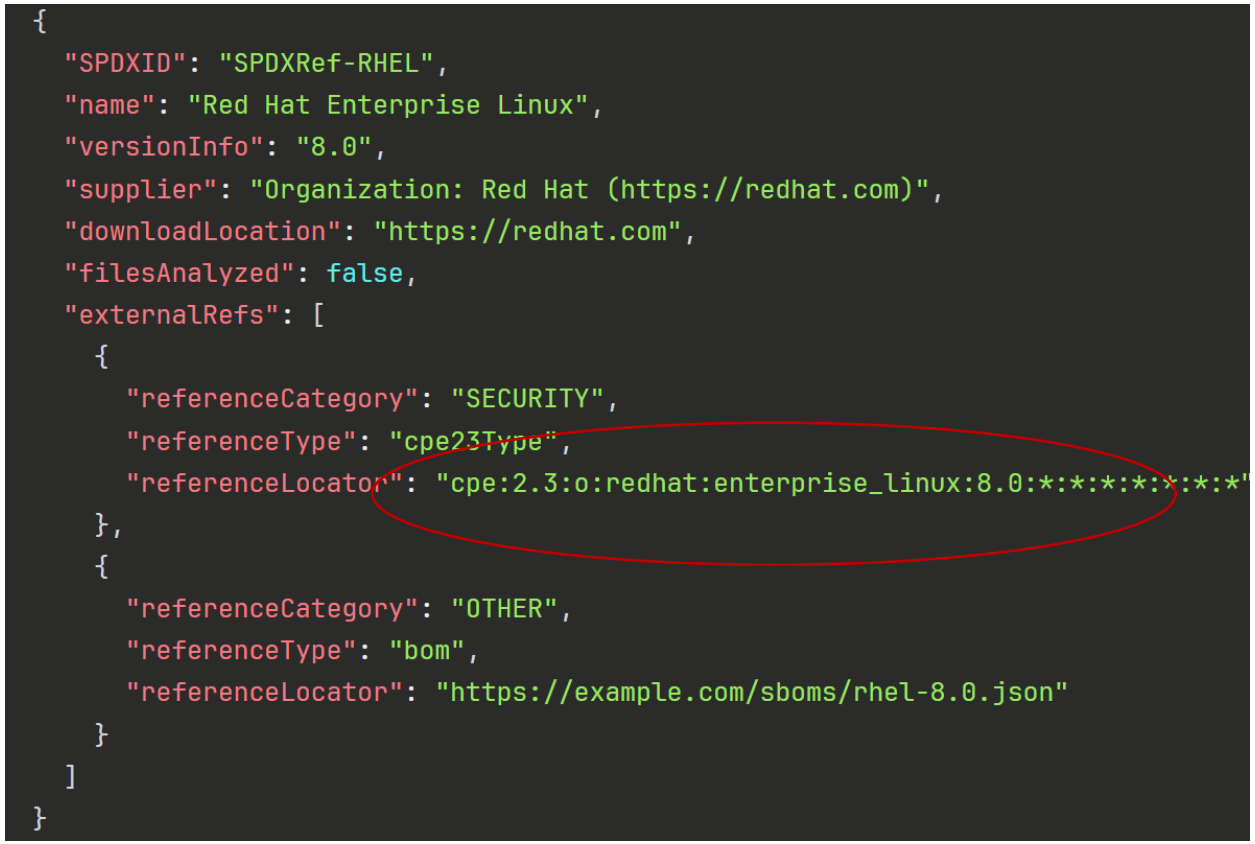
OBOM – Operations BOM – Nomenclature OS

Le **OBOM** décrit un **actif opérationnel, dynamique**, qui va recevoir régulièrement des mises à jour.

OS avec paquets déployés

Exemples de OBOM CycloneDX et SPDX

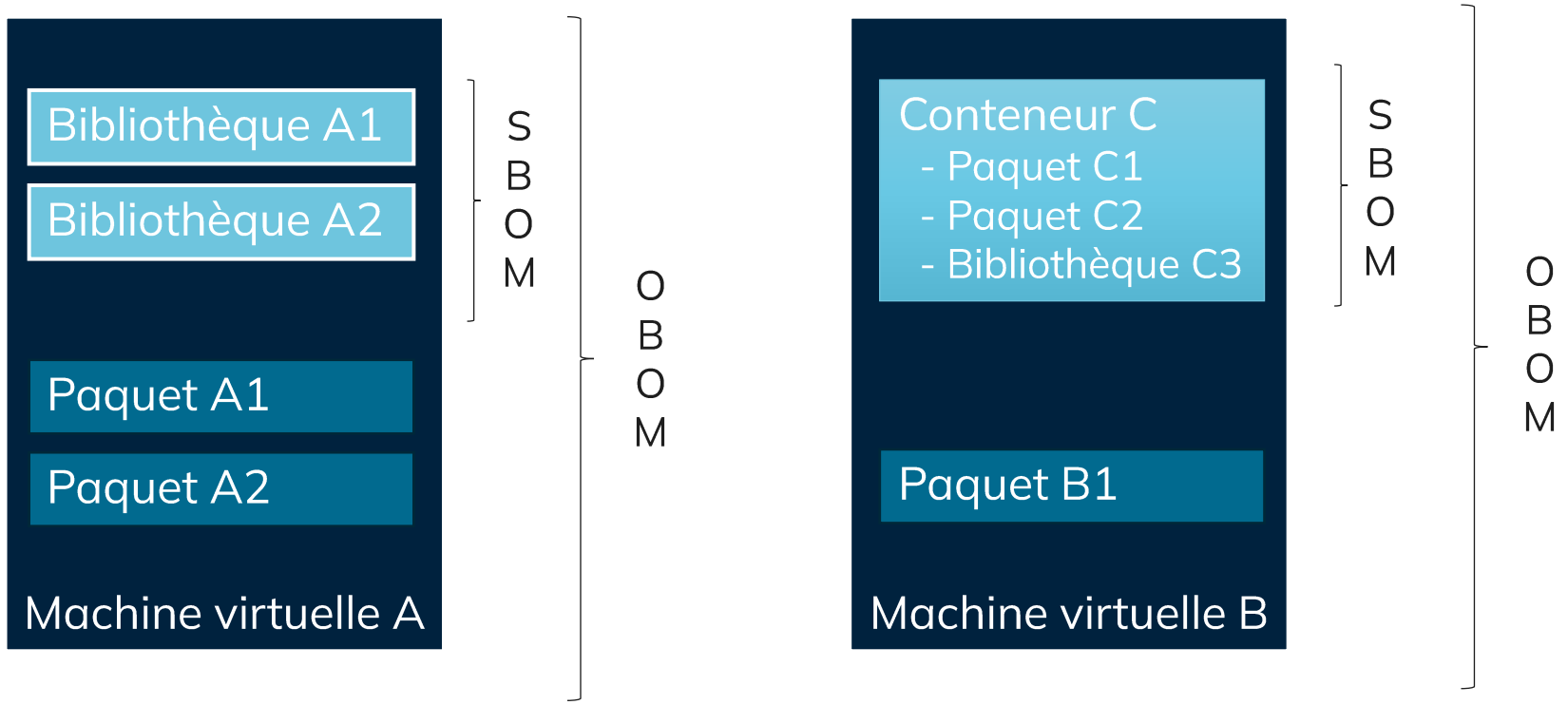
```
          "name": "internal.deployment.cloud.region",
          "value": "us-west-1"
        },
        {
          "name": "internal.deployment.cloud.region",
          "value": "eu-west-2"
        },
        {
          "name": "internal.deployment.cloud.region",
          "value": "eu-north-1"
        }
      ]
    }
  },
  "components": [
    {
      "type": "operating-system",
      "bom-ref": "rhel-8.0",
      "supplier": {
        "name": "Red Hat",
        "url": [
          "https://redhat.com"
        ]
      },
      "group": "com.redhat",
      "name": "Red Hat Enterprise Linux",
      "version": "8.0",
      "cpe": "cpe:2.3:o:redhat:enterprise_linux:8.0:*:*:*:*:*:*:*",
      "externalReferences": [
        {
          "type": "bom",
          "url": "https://example.com/sboms/rhel-8.0.json"
        }
      ]
    }
  ]
```



```
{
  "SPDXID": "SPDXRef-RHEL",
  "name": "Red Hat Enterprise Linux",
  "versionInfo": "8.0",
  "supplier": "Organization: Red Hat (https://redhat.com)",
  "downloadLocation": "https://redhat.com",
  "filesAnalyzed": false,
  "externalRefs": [
    {
      "referenceCategory": "SECURITY",
      "referenceType": "cpe23Type",
      "referenceLocator": "cpe:2.3:o:redhat:enterprise_linux:8.0:*:*:*:*:*:*:*"
    },
    {
      "referenceCategory": "OTHER",
      "referenceType": "bom",
      "referenceLocator": "https://example.com/sboms/rhel-8.0.json"
    }
  ]
}
```

OBOM + SBOM sont très complémentaires

Le **OBOM** décrit **l'hôte**, le **SBOM** décrit l'**ensemble logiciel** exécuté sur cet hôte.



OBOM + SBOM = environnement complet de la plupart des environnements IT modernes.
Le SBOM étant « statique », et le OBOM « dynamique », **il est recommandé de les dissocier**.

CBOM – Cryptography BOM – Config Cryptographique

Le CBOM décrit la configuration cryptographique du système, avec par exemples les certificats, les ciphers SSL/TLS, ou encore les algorithmes utilisés par une application pour stocker ses secrets.


```
"components": [
  {
    "name": "google.com",
    "type": "cryptographic-asset",
    "bom-ref": "crypto/certificate/google.com@sha256:1e15e0fbd3ce95bde5945633ae96add551341b11e5bae7bba12e98ad84a5beb4",
    "cryptoProperties": {
      "assetType": "certificate",
      "certificateProperties": {
        "subjectName": "CN = www.google.com",
        "issuerName": "C = US, O = Google Trust Services LLC, CN = GTS CA 1C3",
        "notValidBefore": "2016-11-21T08:00:00Z",
        "notValidAfter": "2017-11-22T07:59:59Z",
        "signatureAlgorithmRef": "crypto/algorithm/sha-512-rsa@1.2.840.113549.1.1.13",
        "subjectPublicKeyRef": "crypto/key/rsa-2048@1.2.840.113549.1.1.1",
        "certificateFormat": "X.509",
        "certificateExtension": "crt"
      }
    }
  },
  {
    "name": "SHA512withRSA",
```

```
"components": [
  {
    "type": "cryptographic-asset",
    "name": "AES-128-GCM-128-12",
    "cryptoProperties": {
      "assetType": "algorithm",
      "algorithmProperties": {
        "algorithmFamily": "AES",
        "primitive": "ae",
        "parameterSetIdentifier": "128",
        "mode": "gcm",
        "executionEnvironment": "software-plain-ram",
        "implementationPlatform": "x86_64",
        "certificationLevel": [ "none" ],
        "cryptoFunctions": [ "keygen", "encrypt", "decrypt", "tag" ],
        "classicalSecurityLevel": 128,
        "nistQuantumSecurityLevel": 1
      },
      "oid": "2.16.840.1.101.3.4.1.6"
    }
  },
```


Exemples : description d'un Certificat, et d'un chiffrement AES-256-GCM, avec CycloneDX

ML-BOM (ou AI-BOM) – Décrit un modèle, une IA

Le ML-BOM, ou AI-BOM, décrit un modèle d'IA avec les jeux de données associés.

```
@ai_system{gpt4_openai,
    title = {GPT-4 by OpenAI},

    % Hardware Information
    hardware = {
        training = {TPU_v4, Nvidia_A100}, % List of hardware used for training
        inference = {TPU_v3, Nvidia_T4}   % List of hardware used for inference
    },

    % Programming Language Information
    languages = {
        training = {Python},
        inference = {Python, C++, Rust}
    },

    % Developers involved
    developers = {
        jasebell = {link = {https://github.com/jasebell}}
    }

    % Source Data
    data_source = {
        training = {WebText, Books, Journals, SyntheticDataset}, % List or description of datasets
        validation = {CustomTestSet},                            % Validation datasets
        synthetic_data = {                                       % Details on synthetic data
```

Exemples : description de GPT-4, entraîné sur des puces NVIDIA A100, en CycloneDX

SaaSBOM – SaaS BOM – Décrit des services tiers

Le SaaSBOM décrit les services appelés (API Mail, buckets S3…).

```
"type": "service",
"bom-ref": "sendgrid-email",
"provider": {
  "name": "Twilio SendGrid",
  "url": ["https://sendgrid.com"]
},
"name": "SendGrid Email Service",
"version": "v3",
"description": "Transactional and notification emails",
"endpoints": [
  "https://api.sendgrid.com/v3/mail/send",
  "https://api.sendgrid.com/v3/suppression/bounces",
  "https://api.sendgrid.com/v3/stats"
],
"authenticated": true,
```

```
  "name": "Stripe Inc.",
  "url": ["https://stripe.com"]
},
"name": "Stripe Payment Processing",
"version": "v1",
"description": "Payment processing for subscription billing",
"endpoints": [
  "https://api.stripe.com/v1/charges",
  "https://api.stripe.com/v1/customers",
  "https://api.stripe.com/v1/subscriptions",
  "https://api.stripe.com/v1/invoices"
],
"authenticated": true,
```

Exemples : description d'appels vers du SendGrid et du paiement Stripe

MBOM – Manufacturing BOM – Décrit la compilation

Le MBOM décrit la méthode de compilation d'un binaire.


```
"components": [
  {
    "bom-ref": "file:///CycloneDX/MBOM-examples/simple-application-makefile/helloworld.c",
    "type": "file",
    "name": "helloworld.c",
    "version": "1.0",
    "mime-type": "text/x-csrc"
  },
  {
    "bom-ref": "file:///CycloneDX/MBOM-examples/simple-application-makefile/Makefile",
    "type": "file",
    "name": "Makefile",
    "version": "1.0"
  },
  {
    "bom-ref": "file:///Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/gcc",
    "type": "application",
    "name": "gcc",
    "version": "16.0.0 (clang-1600.0.26.4)"
  },
  {
    "bom-ref": "file:///Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/make",
    "type": "application",
    "name": "GNU Make",
    "version": "3.81"
  }
],
```

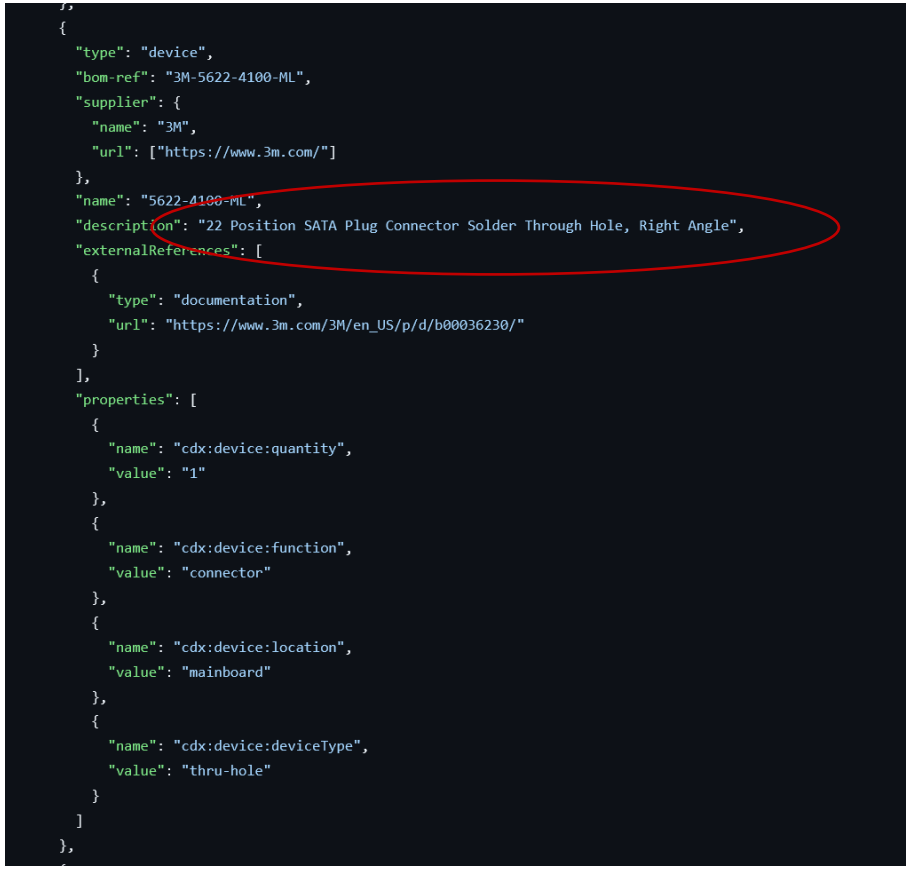


```
"workflows": [
  {
    "bom-ref": "workflow-1",
    "uid": "uuid:8a2c3bf8-77fe-4c1d-849b-c534abc73aee",
    "taskTypes": [ "clean", "build" ],
    "tasks": [
      {
        "bom-ref": "task-1",
        "uid": "uuid:dbb6c5c0-6958-4a18-ac67-d897dbee76b6",
        "taskTypes": ["clean", "build"],
        "name": "make build task",
        "description": "A task that captures 'make build' step.",
        "steps": [
          {
            "name": "run make build",
            "commands": [
              {
                "executed": "make build"
```


Exemples : description d'une compilation de helloworld.c avec un Makefile, make et gcc, en CycloneDX

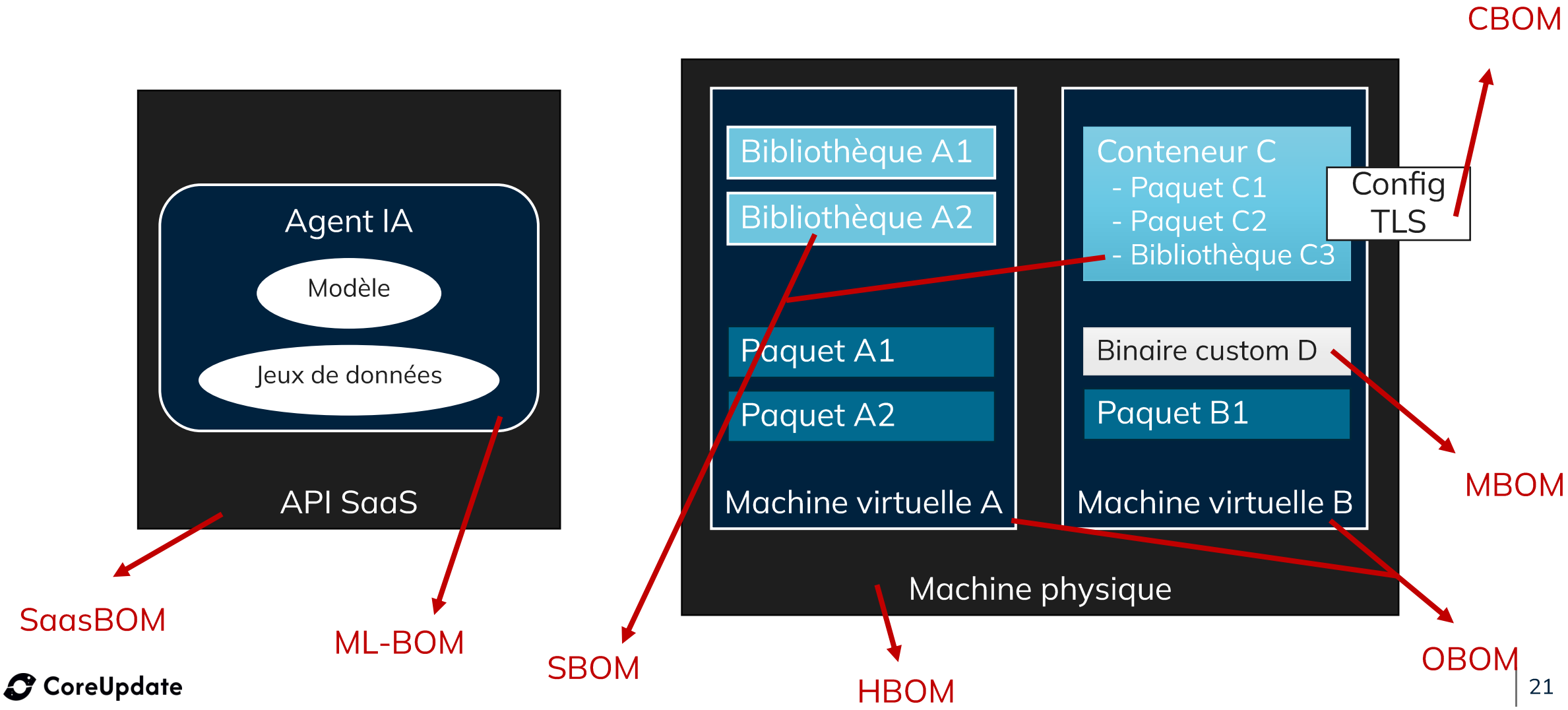
HBOM – Hardware BOM – Décrit le matériel

Le HBOM décrit un ensemble de composants matériels (configuration PC, configuration PCB…).



Exemples : description d’un connecteur SATA en CycloneDX

XBOM – Résumé


CBOM
Agent IA
Modèle
Jeux de données
API SaaS
Bibliothèque A1
Bibliothèque A2
Paquet A1
Paquet A2
Machine virtuelle A
Conteneur C
- Paquet C1
- Paquet C2
- Bibliothèque C3
Config TLS
Binaire custom D
Paquet B1
Machine virtuelle B
Machine physique
MBOM
SaasBOM
ML-BOM
SBOM
HBOM
OBOM

Revue des différents types d'inventaires (BOM ou XBOM)

Je veux documenter *(du plus important au moins important à connaître pour les éditeurs de logiciels soumis au CRA)* :

- L'ensemble des **bibliothèques tierces**, ou une **image Docker** de mon logiciel => **SBOM**
- L'ensemble des **paquets**, et l'**OS** => **OBOM**
- Les **suites cryptographiques** => **CBOM**
- Les **modèles d'IA** utilisés => **ML-BOM**
- Les API **Cloud** tierces appelées => **SaaSBOM**
- La configuration de **compilation** => **MBOM**
- Le **matériel** => **HBOM**

Ces inventaires permettent également de générer des listes de **vulnérabilités** (**BOV**), des **résultats d'analyse** d'exploitabilité (**VEX**), rapports de **licences utilisées**, etc.

Pourquoi autant de nomenclatures ?

- Sujet difficile car une nomenclature unique demanderait de traiter différents constructeurs et cas d'usages
  - Exemples :
    - Editeurs d'Appliances physiques (équipements réseaux) vs Images Docker
    - Editeurs de logiciels qui ne sont que des surcouches d'IAs

- Difficultés aussi liées aux besoins d'exploitation qui résultent de ces nomenclatures
  - Exemples en matière de gestion des vulnérabilités:
    - Différences Windows / Linux
      - Sur **Windows**, une vulnérabilité est liée à l'**absence d'une KB** (*commence à changer depuis Windows 10*)
      - Sur **Linux**, une vulnérabilité est liée à la **présence d'une version vulnérable**
    - Différences entre les distributions Linux
      - Sur **Linux**, **un même produit** peut être **modifié par chaque distribution** (*CVE-2024-3094 affectait XZ-Utils sur Debian, pas Ubuntu)*
  - Les nomenclatures logicielles doivent permettre de gérer ces différences

4. Quel format d’inventaire utiliser en 2026 ?

Quelques formats à connaître

Standard	Date	Catégorie	Cas d'usage principal
CPE	2007	Identification	Nommer un logiciel, notamment dans une CVE (cf NVD)
SWID	2009	Identification	Inventaire de logiciels installés
PURL	2017	Identification	Nommer une bibliothèque tierce (NPM, Maven…) ou une image Docker
SPDX®	2010	SBOM	Historiquement : licences logicielles Maintenant : sécurité Supply Chain
CycloneDX	2017	SBOM	Sécurité Suppy Chain
CSAF	2012	Avis de sécurité	Publier des informations sur les mises à jour de sécurité
OSV	2021	Vulnérabilités	Aider à scanner les CVEs des produits open source

Pour la sécurité de la Supply Chain, 4 sont à connaître

CPE

cpe:2.3:a:apache:http_server:2.4.51:*:*:*:*:*:*:*
cpe:2.3:o:redhat:enterprise_linux:9.0:*:*:*:*:*:*:*

PURL

pkg:npm/lodash@4.17.21
pkg:docker/ubuntu@22.04

SPDX®

Le plus ancien, pratique pour décrire les bibliothèques d'un logiciel, notamment les dépendances Open Source et leurs licences

CycloneDX

Le plus moderne, permet de décrire une Appliance complète

Ma recommandation pour les éditeurs : priorisez CycloneDX
Ecosystème fort et orienté sécurité
Permet de décrire l'ensemble des composants de vos produits

5. Limites des inventaires logiciels

1. L’écosystème Windows est mal couvert

Build 22621.2428 (KB5031354)

Côté CPE, il y a plusieurs approches :
cpe:2.3:o:microsoft:windows_11:22h2:*:*:*:*:*:*:*:*
⇒ Pas assez complet

cpe:2.3:o:microsoft:windows_11_22h2:10.0.22621.2428:*:*:*:*:*:*:*:*
⇒ Considère que W11 22H2 est un OS, pas de consensus

Côté PURL :
pkg:generic/microsoft/kb5034441
⇒ Mot-clé « générique » utilisé faute de consensus.

La phase Identification est mal couverte, et cela impacte la faisabilité réelle des SBOMs.

2. Un même produit peut être décrit de plusieurs façons

Sur Linux, CPE et PURL n'ont pas d'équivalence directe :

cpe:2.3:a:apache:http_server:2.4.7:*:*:*:*:*:*:*
vs
pkg:deb/ubuntu/apache2@2.4.7-1ubuntu4.22

Le standard CPE, largement utilisé dans les bases de vulnérabilités, n'est pas assez précis. PURL corrige cela, mais n'est pas encore suffisamment adopté.

De même, pour réduire les efforts, je peux mettre un « generic » partout :
pkg:deb/ubuntu/apache2@2.4.7-1ubuntu4.22
vs
pkg:generic/apache2@2.4.7-1ubuntu4.22

Le standard permet au producteur de l'inventaire de prendre des raccourcis, qui rendent les inventaires difficiles à exploiter ensuite.

3. Quid des rachats ?

Cas de Vmware, racheté par Broadcom :

cpe:2.3:a:vmware:vcenter_server:7.0:*:*:*:*:*:*:*
vs
cpe:2.3:a:broadcom:vcenter_server:7.0:*:*:*:*:*:*:*

Idem pour HPE qui a racheté Aruba.
Pire encore pour McAfee et FireEye, qui ont **fusionné** en Trellix.

Les backports se sont jamais faits, tant pour CPE que pour PURL.

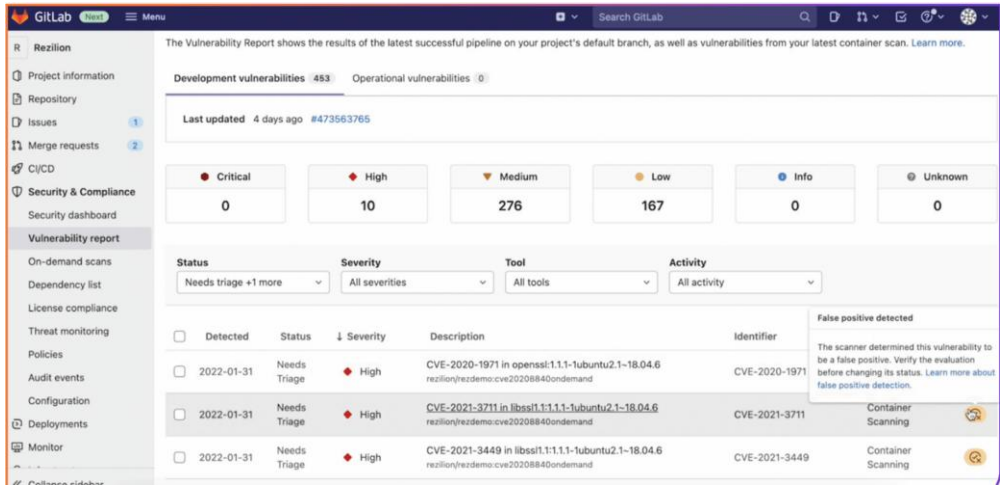
La limite vient ainsi principalement de la partie Identification

Mon souhait : à quand une base communautaire capable de traduire et maintenir l'ensemble des formats disponibles ? (chercher des CPE, traduire en PURL, etc)

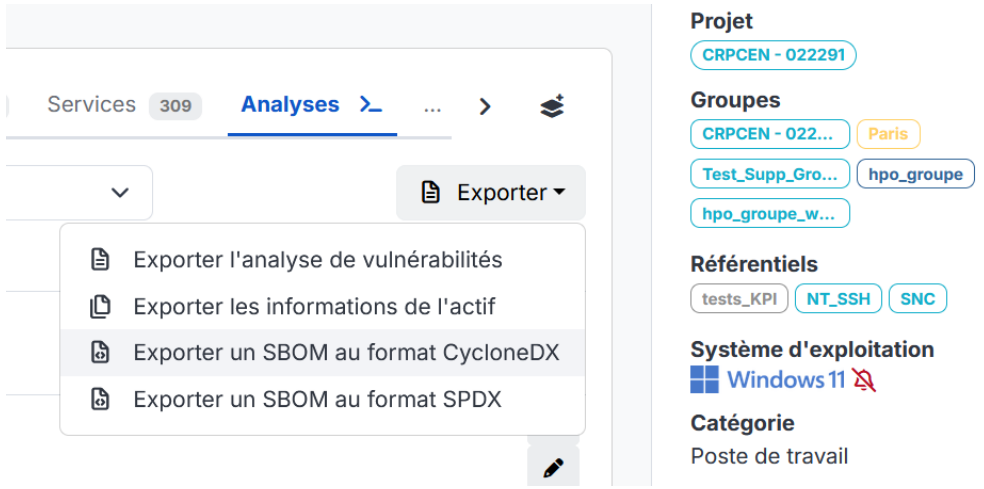
6. Cas pratique : comment générer un SBOM pour un logiciel, et pour une appliance ?

Plusieurs logiciels le proposent par défaut

- Forges logicielles : GitLab, GitHub



- Scanners de vulnérabilités ou de dépendances



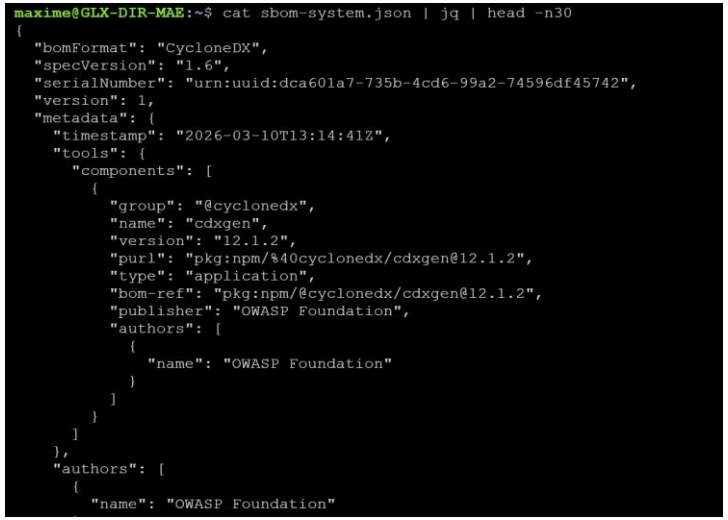
Pour générer du CycloneDX à la main : cdxgen

- Installation :
  - npm install -g @cyclonedx/cdxgen
  - winget install cdxgen
  - ou docker pull ghcr.io/cyclonedx/cdxgen

- Générer le **SBOM** d'une image Docker :
  - cdxgen -t docker -o sbom.json nginx:latest

- Générer le **OBOM** du système actuel (**TRES LONG**) :
  - Linux : cdxgen -t os -o sbom-system.json
  - Windows : cdxgen-windows-amd64.exe -t os -o sbom-system.json

- **Les autres formats (MLBOM, HBOM…) sont difficiles à générer !**



CONCLUSION

1. Pas d’excuse pour ne pas générer votre SBOM / OBOM.
2. Attention : le CRA arrive, préparez-vous maintenant.
3. Les standards ne sont pas parfaits, mais s’améliorent régulièrement
4. Difficulté restante : comment transmettre les XBOM à ses clients ?

Q&A

Merci pour votre attention !