

Triage des CVE applicatif avec un LLM et le contexte runtime

Fahed Dorgaa

whoami

Fahed Dorgaa
Ingénieur logiciel senior, vulnérabilités et supply chain.

- Mainteneur de nerdctl (communauté containerd)
- Maintainer de vens (Apache-2.0)



Vous connaissez ce moment.

python:3.12 : 2639 vulnérabilités, 60 critiques.

- Un score générique, aveugle à votre prod
- Des jours de tri à la main
- On livre en croisant les doigts

Je l'ai vécu. Qui, ici, aussi ?



Le moment où la CI passe au rouge

On repousse tous son backlog de vulns.
Jusqu'à ce qu'une security gate passe la CI en rouge.

Impact :
• Votre feature attend
• La MEP de vendredi aussi

Et cet impact n'est pas justifié : ce seuil vient d'un score qui ne regarde pas votre prod.

Le score générique, et le vôtre

✗ CVSS de base

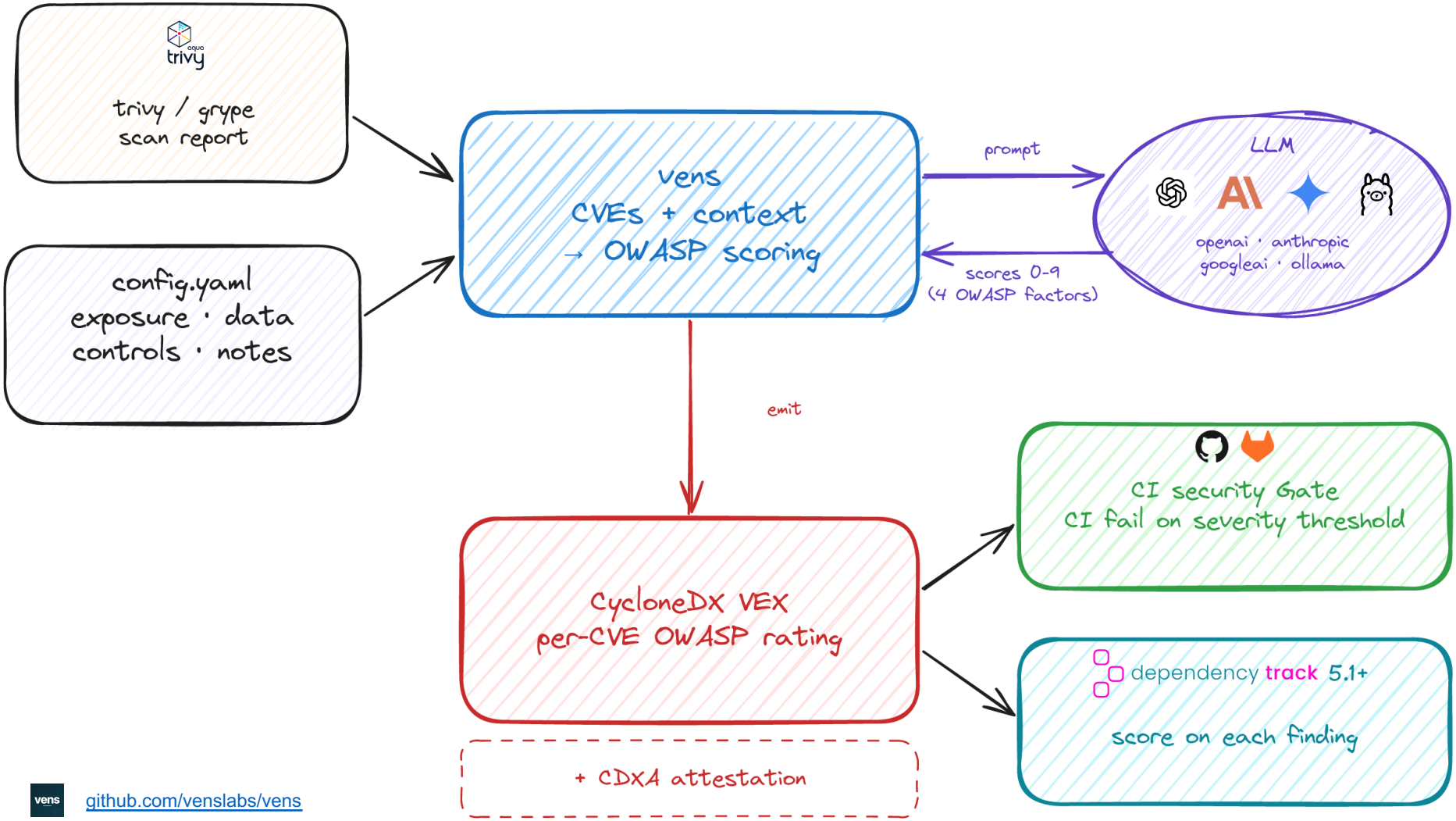
Le score que votre scanner affiche.

- Il décrit seulement la faille, pas votre application

✓ OWASP Risk Rating

Un score sur le risque, chez vous.

- Exposition, données, conformité
- La liste se réordonne par risque réel
- C'est ce que vens calcule


aqua trivy
trivy / grype
scan report
config.yaml
exposure · data
controls · notes
vens
CVEs + context
→ OWASP scoring
prompt
scores 0-9
(4 OWASP factors)
LLM
openai · anthropic
googleai · ollama
emit
CycloneDX VEX
per-CVE OWASP rating
+ CDXA attestation
CI security Gate
CI fail on severity threshold
dependency track 5.1+
score on each finding
vens
github.com/venslabs/vens

VEX et CDXA, c'est quoi ?

1 VEX

Vulnerability Exploitability eXchange.

- Un document standard (CycloneDX)
- Un nouveau score contextuel par CVE
- Consommé par votre CI, Trivy et Dependency-Track

2 CDXA

CycloneDX Attestation.

- La preuve, à côté du VEX
- Raisonnement, modèle, prompt, entrées enregistrés
- Rejouable et auditable

Le contexte se déclare

config.yaml

```yaml
context:
  # internal | private | internet
  exposure: "internet"
  data_sensitivity: "high"
  business_criticality: "high"

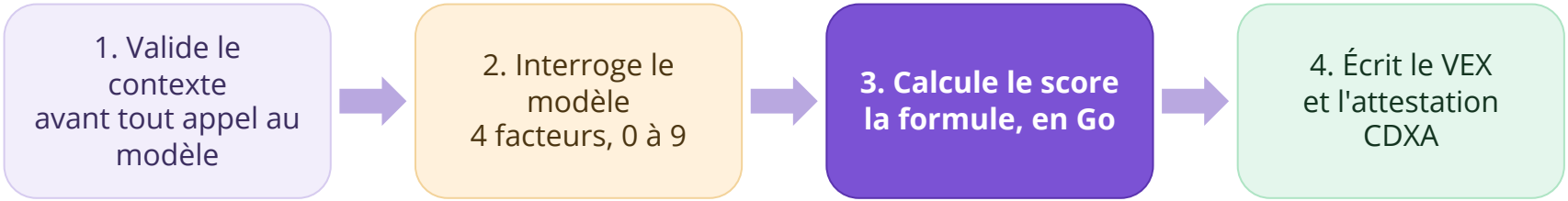
  compliance_requirements:
    - "GDPR"

  controls:
    waf: true
    segmentation: true
```

Des champs précis et validés par vens.

https://github.com/venslabs/vens/blob/main/examples/quickstart/config.yaml

Sous le capot : vens garde la main



Une seule étape sur quatre passe par le modèle.
Elle remplit des facteurs OWASP. Elle n'écrit jamais le score final.

Démo : le même rapport de scan, deux contextes

A Contexte exposé

exposure: internet,
données critiques,
service critique.

Le risque contextuel monte.

B Contexte interne

exposure: internal,
données faibles,
job de nuit isolé.

Le risque retombe.

TL;DR de la démo

Même image. Même scanner. Mêmes CVE.

Le fichier de contexte a changé : un build s'arrête sur la security gate, l'autre passe.

Le contexte fait monter ou descendre la sévérité de chaque CVE.

Le même blocage, mais justifié

On peut toujours avoir un job qui casse sur une security gate, mais pas sur le même score.

Impact :
• Votre feature attend
• La MEP de vendredi aussi

Et cette fois l'impact est justifié : ce score, lui, a regardé votre prod.

Qui écrit quoi

platform engineer

Une pipeline CI qui fait
le scan et le scoring.
Écrite une seule fois.

les développeurs

Un fichier de contexte dans
leur repo, relu en PR,
qui déclare l'exposition
du service.

security engineer

Garde le seuil et la politique.
Reçoit le VEX dans
Dependency-Track.

**La gestion des vulnérabilités quitte le monopole de la sécu
et devient une tâche partagée entre les équipes. Du DevSecOps appliqué.**

vens, un plugin Trivy

Plugin Trivy officiel, dans l'index public d'Aqua.

zsh

```
# 1 · installer le plugin
$ trivy plugin install github.com/venslabs/vens

# 2 · scorer un rapport, via trivy
$ trivy vens generate \
    --config-file config.yaml \
    --sbom-serial-number $SN \
    report.json out.vex
```

https://aquasecurity.github.io/trivy-plugin-index/

Une démo, ça ne suffit pas

Ce que vous venez de voir marche.
Avec un autre modèle, peut-être pas.

- J'ai benchmarké douze modèles dans vens, pour savoir lequel choisir et lequel éviter

Le détail et le papier, citables :
https://github.com/venslabs/vens-benchmark
https://doi.org/10.5281/zenodo.21775841

Quel LLM, et lequel éviter

✓ À utiliser

claude-sonnet-4-6
à égalité avec gpt-5.5,
et 2× moins cher que lui

gpt-5.4-mini
même précision, 0.48 $
contre 4.12 $ pour 200 CVE

✗ À éviter

gemini-2.5-flash-lite
regonfle Log4Shell, 3 fois sur 3 :
le pas cher qui plante en silence

modèles locaux
encore peu testés

Mesuré le 8 juillet 2026

Là où ça casse

- Le modèle n'est pas déterministe : deux runs, deux scores possibles
- Ce n'est pas de l'atteignabilité : le contexte est déclaré, pas analysé
- Une clé mal orthographiée dans ce contexte est ignorée en silence

Nouveau dans vens v0.6.0

- Support Windows
- Plus de température imposée : chaque modèle garde la sienne
- Une CVE sans composant arrête le run au lieu de disparaître

En cours : 16 facteurs au lieu de 4

4 Aujourd'hui

Le LLM estime tous les facteurs.

- 4 facteurs, de 0 à 9

16 En cours

Une partie se calcule, le reste au LLM.

- Plus précis
- Plus reproductible

vens

MERCI !

github.com/venslabs/vens

Votre avis