Présentation des outils d'analyse de code

Ordre du jour

01 Statuts d'avancement

02 Choix de l'outil selon les besoins

03 Présentation Snyk

04 Présentation SonarQube

05 Présentation des Linters

06 Présentation checkmarX

07 Comparaison des outils d'analyse de code

08 Points forts de chaque outil

09 Conclusion

État d'avancement des outils d'analyse de code

01

SonarQube

Installation locale réussie, déploiement sur R7, premier scan projet, rapports initiaux générés

02

Échanges Snyk

2 courriels, pas d'option 'On Premise', stockage en Allemagne, recommandation version gratuite

03

CheckmarX

5 relances effectuées, attente de réponse

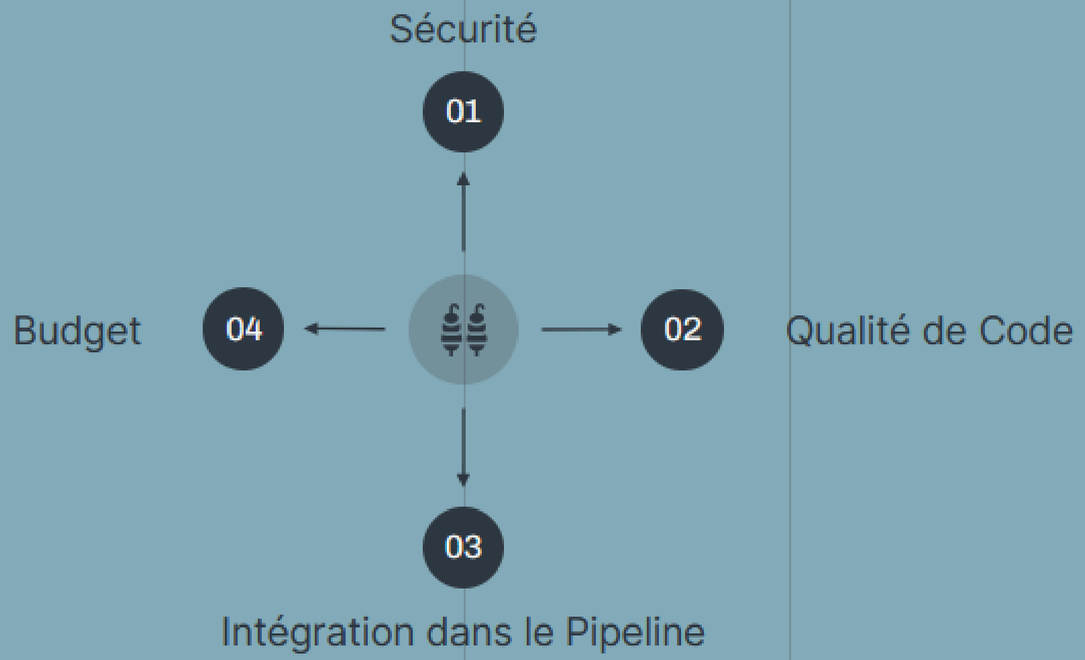
04

Linter

Déjà mis en place

Choix de l'Outil selon les Besoins

Détermination des Priorités pour la Sélection des Outils



PRÉSENTATION SNYK

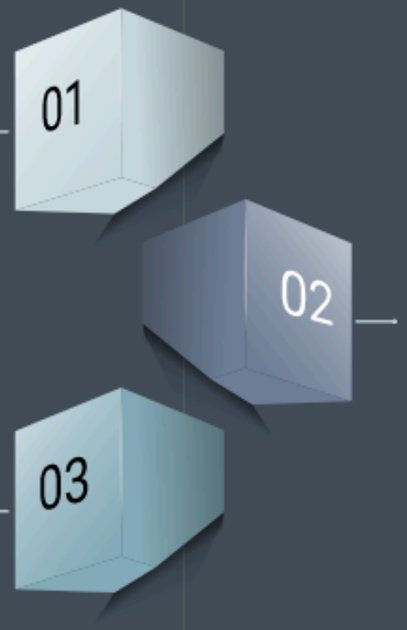
Outil d'analyse de code

Présentation de Snyk

Outil spécialisé en détection des vulnérabilités de sécurité

Détection en temps réel

Il offre une détection en temps réel des failles de sécurité, renforçant la sécurité des applications.



Analyse des dépendances

Il excelle dans l'analyse des dépendances, permettant une gestion efficace des relations entre les composants logiciels.

Intégration CI/CD

Facile à intégrer dans les pipelines de CI/CD, s'adapte parfaitement aux processus DevOps.

Outils d'analyse de code

Caractéristiques de Snyk

Support de plusieurs langages et mise à jour constante des vulnérabilités

01

Langages de programmation variés

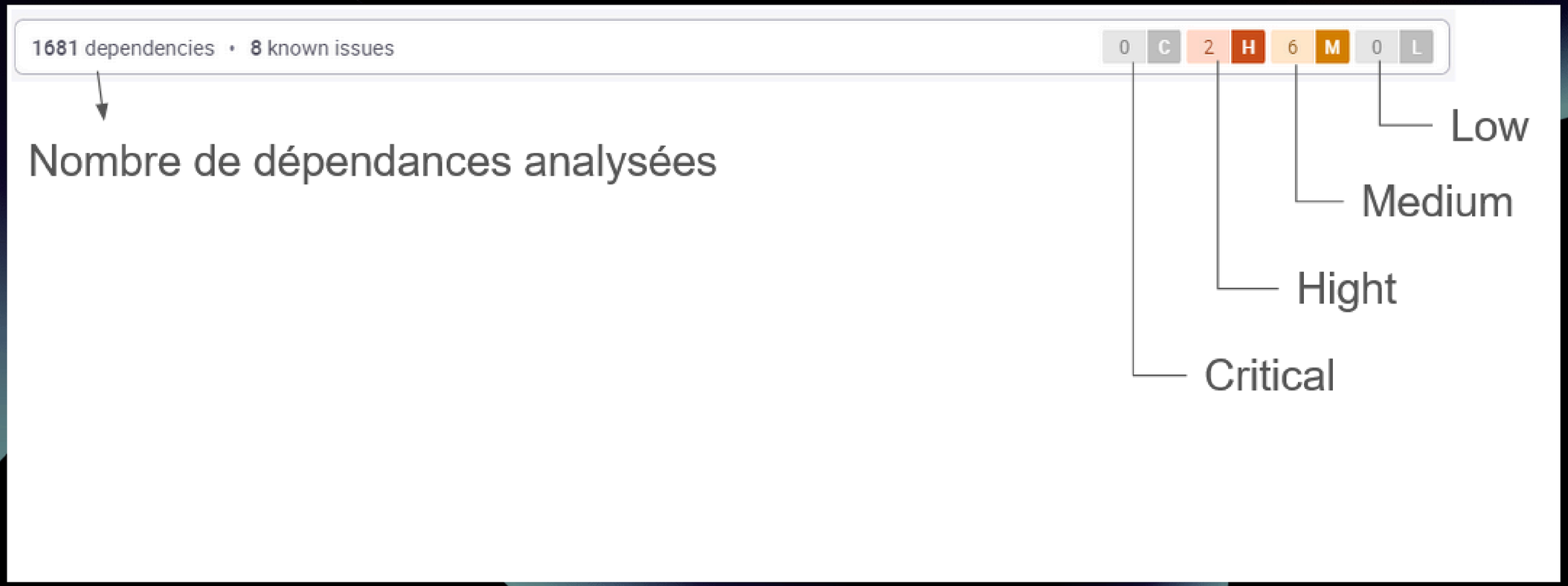
Snyk prend en charge divers langages tels que JavaScript, Python, Java et Go.

02

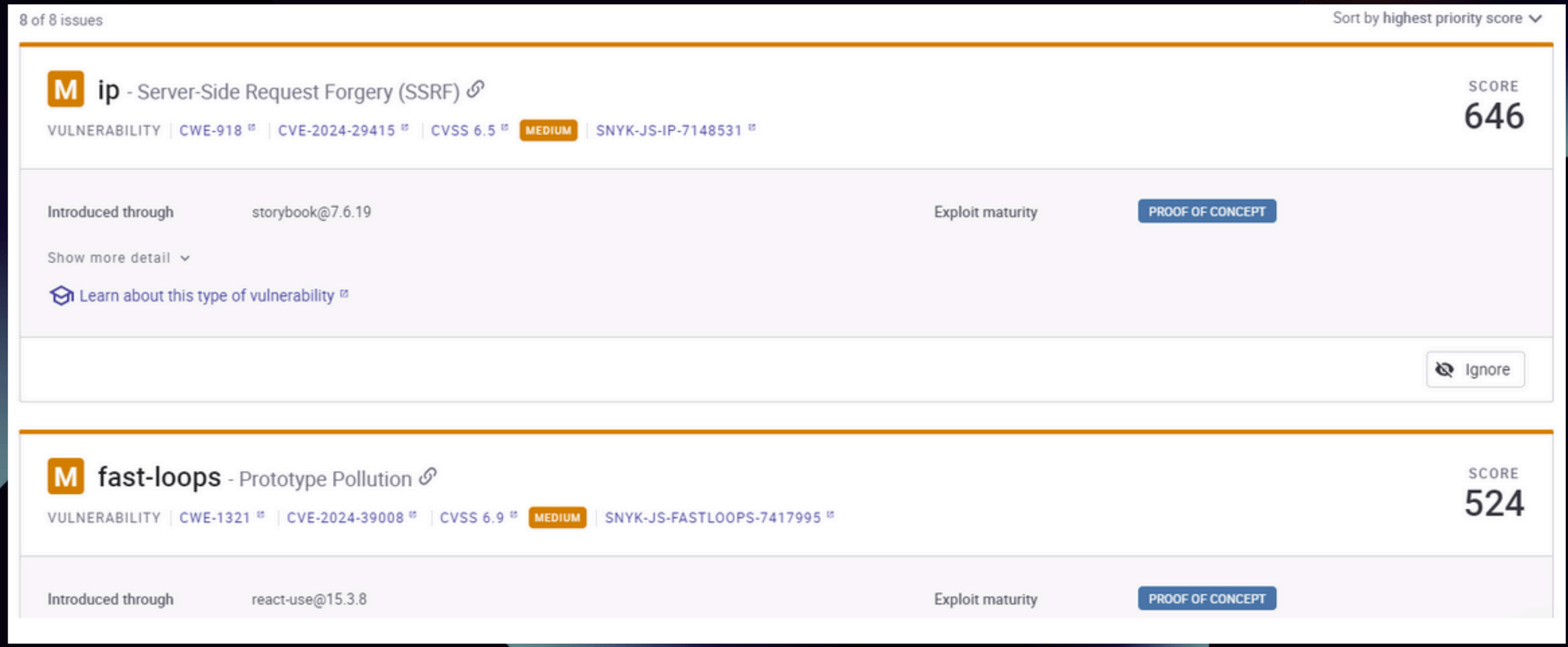
Modèle premium

Propose une version gratuite avec des fonctionnalités avancées payantes.

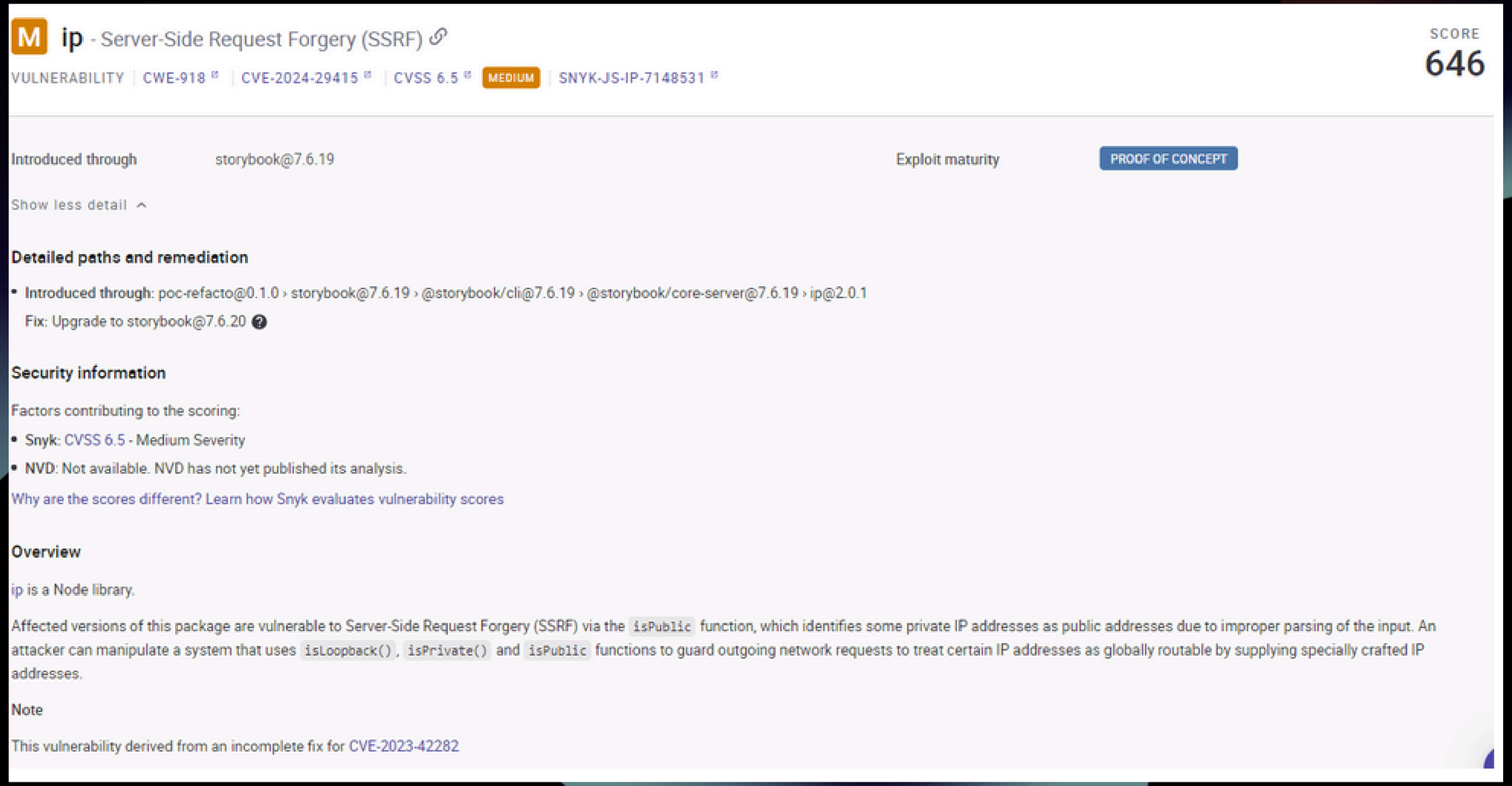
SNYK -COMMENT SE PRÉSENTE UN RAPPORT - VUE SYNTHÉTIQUE



SNYK -COMMENT SE PRÉSENTE UN RAPPORT - VUE DÉTAILLÉE



SNYK -COMMENT SE PRÉSENTE UN RAPPORT - VUE DÉTAILLÉE ++



M ip - Server-Side Request Forgery (SSRF)

VULNERABILITY | CWE-918 | CVE-2024-29415 | CVSS 6.5 | MEDIUM | SNYK-JS-IP-7148531

SCORE 646

Introduced through storybook@7.6.19

Exploit maturity PROOF OF CONCEPT

Show less detail

Detailed paths and remediation

- **Introduced through**: poc-refacto@0.1.0 › storybook@7.6.19 › @storybook/cli@7.6.19 › @storybook/core-server@7.6.19 › ip@2.0.1
  Fix: Upgrade to storybook@7.6.20

Security information

Factors contributing to the scoring:

- **Snyk**: CVSS 6.5 - Medium Severity
- **NVD**: Not available. NVD has not yet published its analysis.

Why are the scores different? Learn how Snyk evaluates vulnerability scores

Overview

ip is a Node library.

Affected versions of this package are vulnerable to Server-Side Request Forgery (SSRF) via the `isPublic` function, which identifies some private IP addresses as public addresses due to improper parsing of the input. An attacker can manipulate a system that uses `isLoopback()`, `isPrivate()` and `isPublic` functions to guard outgoing network requests to treat certain IP addresses as globally routable by supplying specially crafted IP addresses.

Note

This vulnerability derived from an incomplete fix for CVE-2023-42282

1. Intégration avec les plateformes de gestion de code pour détecter les failles.
2. Analyse des vulnérabilités en CI/CD avec correctifs automatisés pour les dépendances.
3. Analyse approfondie du code avec DeepCode AI et recommandations de correctifs.
4. Détection des vulnérabilités 0Day et analyse active des logiciels malveillants.
5. Solutions pour sécuriser les conteneurs et les applications Kubernetes.
6. Base de données OWASP étendue, incluant des failles non encore publiées.

PÉSENTATION SONARQUBE

Outil d'analyse de code

Présentation de SonarQube

01

Détection de bugs

SonarQube se distingue par sa capacité à détecter les bugs dans le code source, facilitant ainsi leur résolution rapide.

02

Analyse des vulnérabilités

Il offre une analyse approfondie des vulnérabilités potentielles, renforçant la sécurité des applications développées.

03

Gestion de la dette technique

Il propose une approche proactive pour analyser et gérer la dette technique, favorisant un développement durable et de qualité.

04

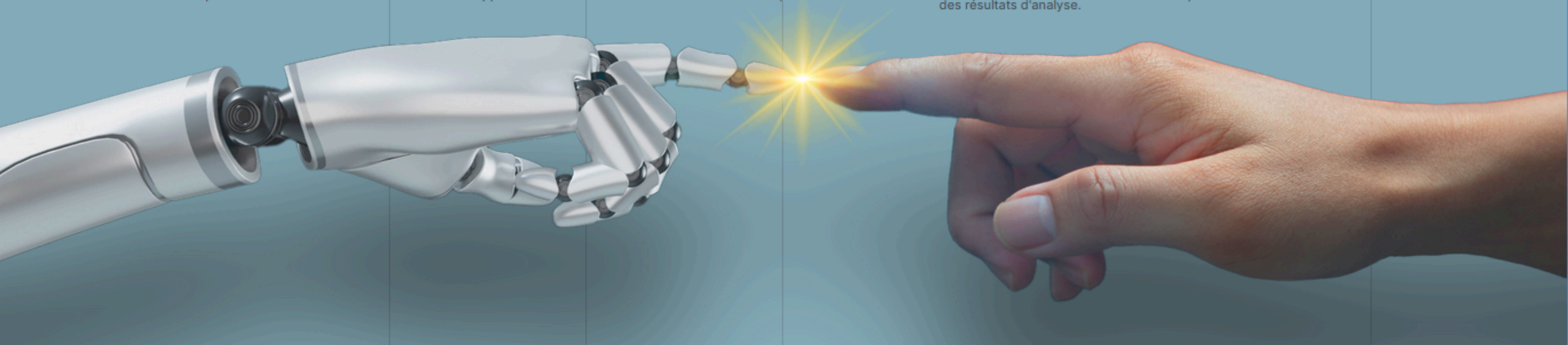
Interface utilisateur intuitive

Il se distingue par son interface conviviale, facilitant la navigation et l'interprétation des résultats d'analyse.

05

Tableaux de bord clairs

Les tableaux de bord clairs et concis de SonarQube offrent une vue d'ensemble précise des analyses, facilitant la prise de décisions informées.



Outils d'Analyse de Code

Caractéristiques de SonarQube

Personnalisable et Polyvalent

Polyvalence Linguistique

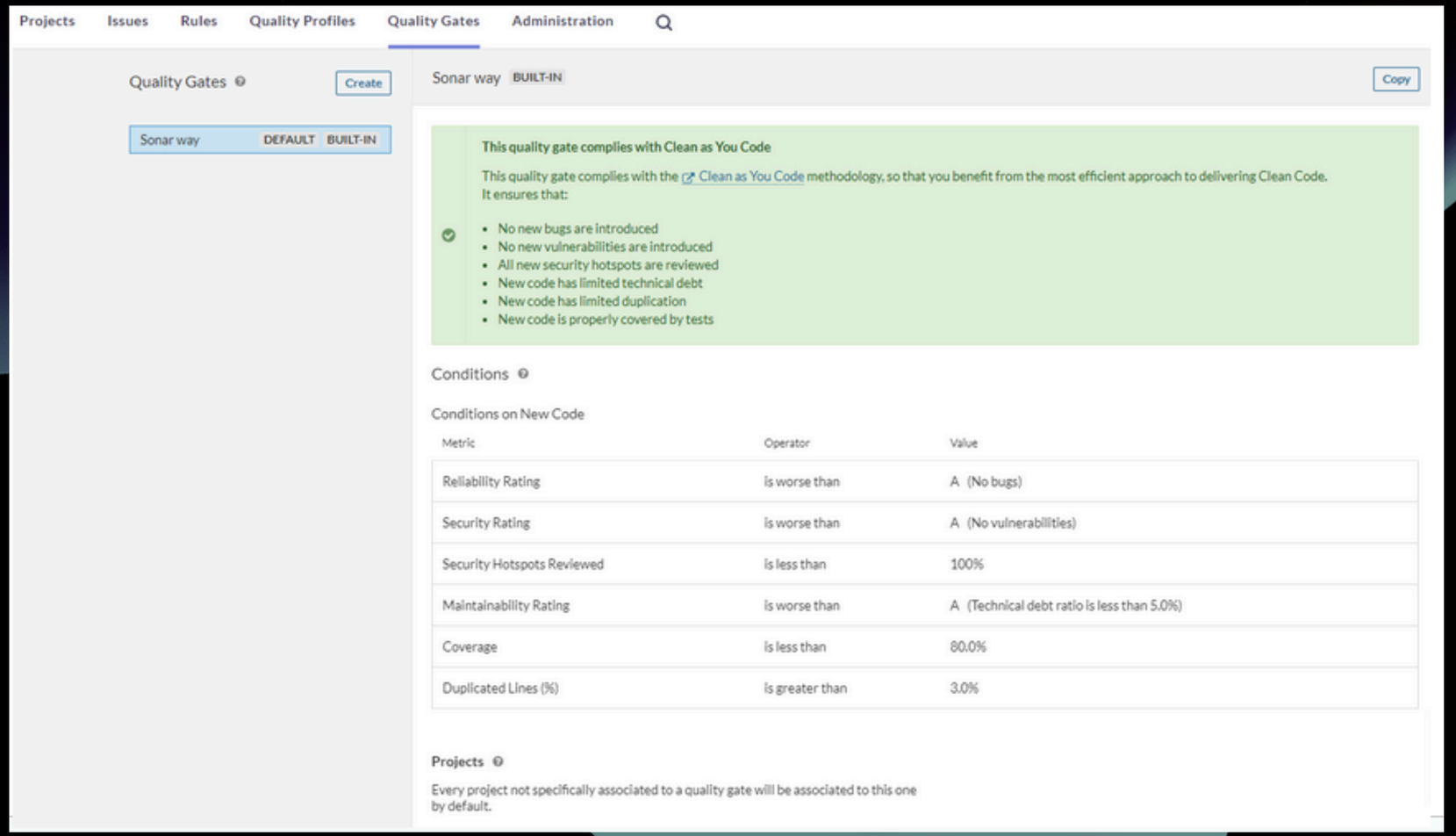
SonarQube supporte un large éventail de langages de programmation, offrant une couverture exhaustive pour divers environnements de développement.

Version Gratuite et Open-Source

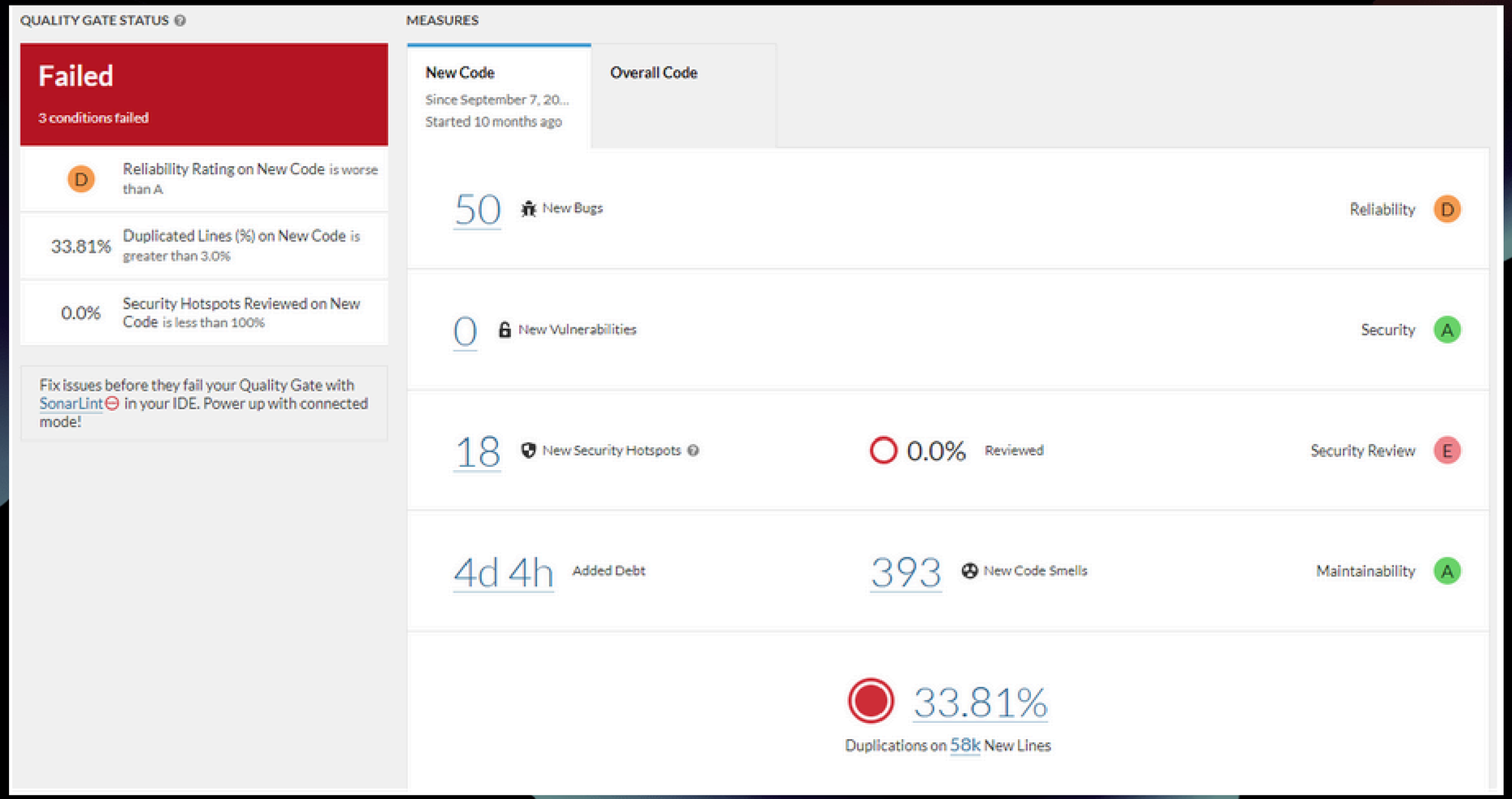
La version de SonarQube est gratuite et open-source, offrant ainsi un accès libre à ses fonctionnalités de base pour les utilisateurs.

QUALITY GATE AU MOMENT DE L'ENVOI EN R7

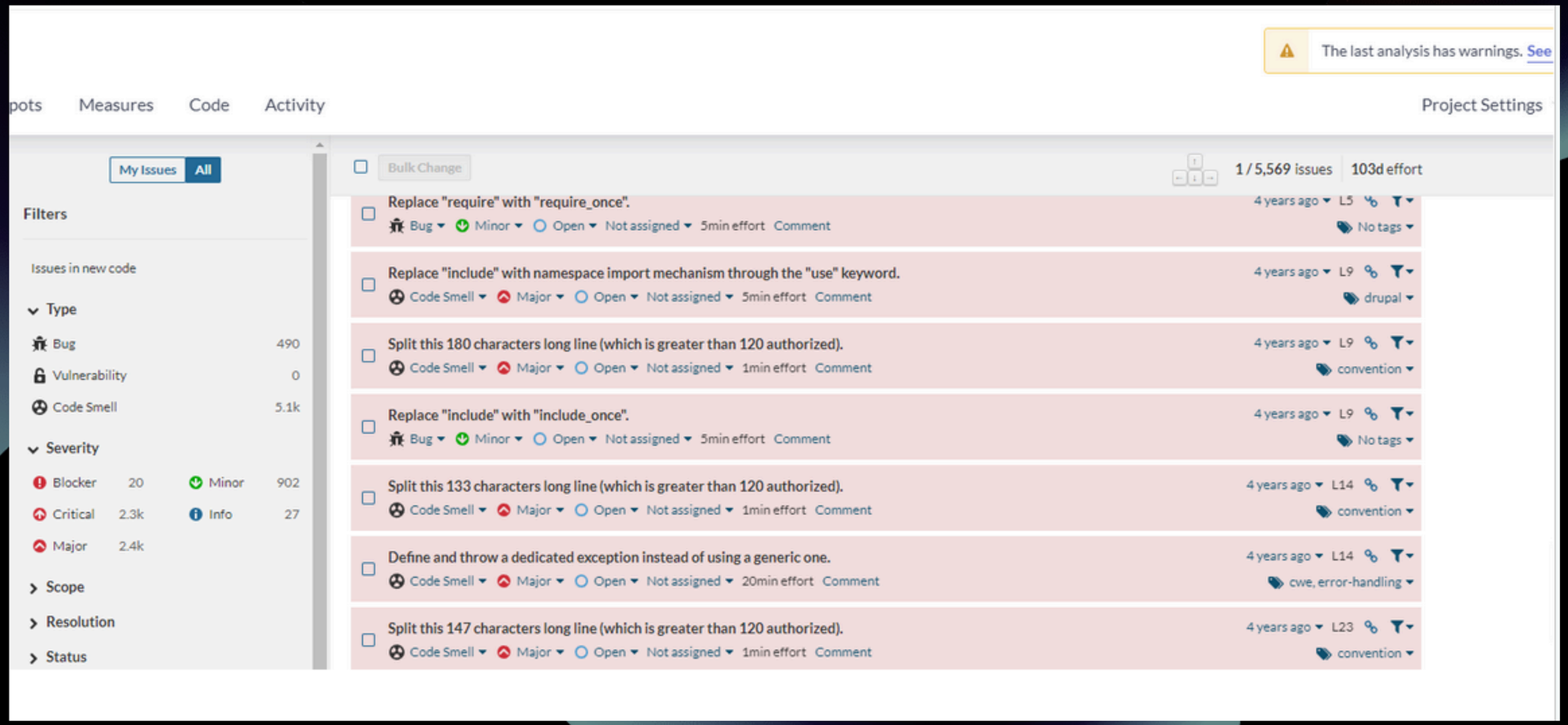
Permet de définir en commun, et en amont, les normes acceptables pour tout nouveau code envoyé. Plusieurs normes sont réglables (sécurité, dette technique, tests unitaires...)



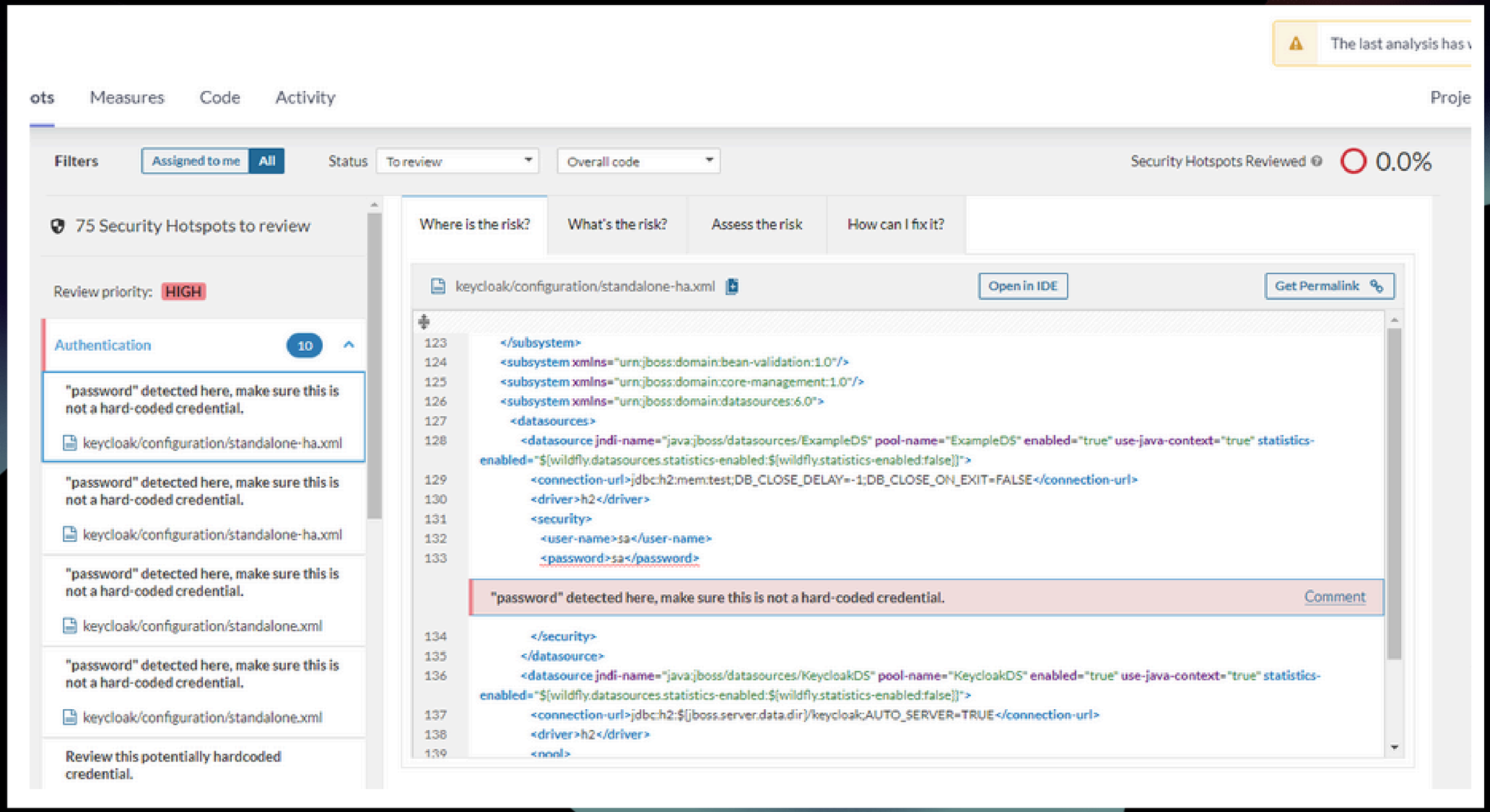
SQ - COMMENT SE PRÉSENTE UN RAPPORT - VUE SYNTHÉTIQUE



SQ -COMMENT SE PRÉSENTE UN RAPPORT - VUE DÉTAILLÉE



SQ -COMMENT SE PRÉSENTE UN RAPPORT - DÉTAIL D'UNE ANOMALIE



1. Analyse les vulnérabilités dans divers langages et propose des corrections basées sur les normes OWASP Top 10 et CWE Top 25.
2. Détecte les problèmes dès le début du développement et alerte les développeurs sur le code suspect.
3. Identifie les duplications de code et vérifie la conformité aux normes de codage.
4. Repère les problèmes exploitables tels que les XSS, les injections SQL et les "code smells".
5. Suggère des vérifications manuelles pour les parties sensibles du code.
6. S'intègre facilement aux outils et processus DevOps.

Remarque : L'édition Community (gratuite) de SonarQube ne propose pas la détection des failles de sécurité.

PRÉSENTATION DES LINTERS

Outils d'Analyse

Présentation des Linters

Analyse Statique du Code

Ils sont principalement utilisés pour l'analyse statique du code et la vérification du style.

Outils Légers et Rapides

Les Linters, tels qu'ESLint pour JavaScript ou Pylint pour Python, sont des outils légers et rapides.

Intégration Facile

Les Linters sont facilement intégrables dans les éditeurs de code et les pipelines CI/CD.

TYPESCRIPT ESLINT

S'intègre facilement dans les pipelines de développement grâce à son architecture flexible.

Aide à maintenir des normes de codage cohérentes.

Identifie et signale les erreurs de syntaxe ainsi que les problèmes de style.

Effectue une analyse statique du code TypeScript.

Respect des normes d’accessibilité de base

Respect des normes d’un code de qualité

Respect des normes de nommage

Respect des normes de complexité par composant

```
need the element to resemble a link, use a button and change it with appropriate styles. Learn more: https://github.com/jsx-eslint/eslint-plugin-jsx-a11y/
r-is-valid.md  jsx-a11y/anchor-is-valid
 /builds/hipe/hipe/frontend-v2/src/pages/Errors/ServiceError.tsx
  49:15  warning  The href attribute requires a valid value to be accessible. Provide a valid, navigable address as the href value. If you cannot provid
need the element to resemble a link, use a button and change it with appropriate styles. Learn more: https://github.com/jsx-eslint/eslint-plugin-jsx-a11y/
r-is-valid.md  jsx-a11y/anchor-is-valid
 /builds/hipe/hipe/frontend-v2/src/pages/EspaceAsile/EspaceAsile.tsx
  175:21  warning  img elements must have an alt prop, either with meaningful text, or an empty string for decorative images  jsx-a11y/alt-text
  228:23  warning  img elements must have an alt prop, either with meaningful text, or an empty string for decorative images  jsx-a11y/alt-text
  301:1   warning  File has too many lines (304). Maximum allowed is 300                                                       max-lines
/builds/hipe/hipe/frontend-v2/src/pages/EspaceRF/components/Beneficiaires.tsx
  78:1  warning  Block is redundant  no-lone-blocks
/builds/hipe/hipe/frontend-v2/src/pages/EspaceRF/components/DetailsInstruction.tsx
  11:20  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
/builds/hipe/hipe/frontend-v2/src/pages/EspaceRF/components/MiseEnAvant.tsx
  12:20  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
  13:15  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
/builds/hipe/hipe/frontend-v2/src/pages/EspaceRF/helpers/cmsContent.ts
  20:30  warning  Arrow function has a complexity of 40. Maximum allowed is 12  complexity
/builds/hipe/hipe/frontend-v2/src/pages/Location/Location.helpers.ts
  12:87  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
/builds/hipe/hipe/frontend-v2/src/pages/MonAllocation/DownloadButton.tsx
  8:26  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
/builds/hipe/hipe/frontend-v2/src/pages/MonAllocation/MonAllocation.tsx
   27:25  warning  Unexpected any. Specify a different type                                                                   @typescript-eslint/no-expl
  27:32  warning  Arrow function has a complexity of 31. Maximum allowed is 12                                              complexity
```

PRÉSENTATION CHECKMARX

Sécurité des applications

Présentation de Checkmarx

Solution de sécurité des applications

Analyses complètes du code

Il offre des analyses statiques (SAST), dynamiques (DAST) et interactives (IAST) du code.

Adapté aux grandes entreprises

Particulièrement adapté aux grandes entreprises ayant des besoins de sécurité complexes.

Analyse des outils de sécurité

Caractéristiques de Checkmarx

Analyse avancée du code pour les équipes de développement

01

Analyse de code compilé et source

Checkmarx prend en charge l'analyse de code compilé et source dans de nombreux langages, offrant une approche complète pour identifier les vulnérabilités.

02

Coût élevé

Le coût élevé de Checkmarx peut constituer un obstacle pour les petites équipes de développement cherchant des solutions plus abordables.

1. Identifie et corrige les vulnérabilités de sécurité avec un faible taux de faux positifs.
2. Analyse le code non compilé et supporte les principaux langages de programmation.
3. S'intègre de manière transparente dans le cycle de vie du développement logiciel (SDLC) pour une détection précoce des failles.
4. Fournit des recommandations pour corriger les vulnérabilités identifiées via CxSAST.
5. Propose des actions de sécurité comme le blocage des intrusions et des adresses IP suspectes.

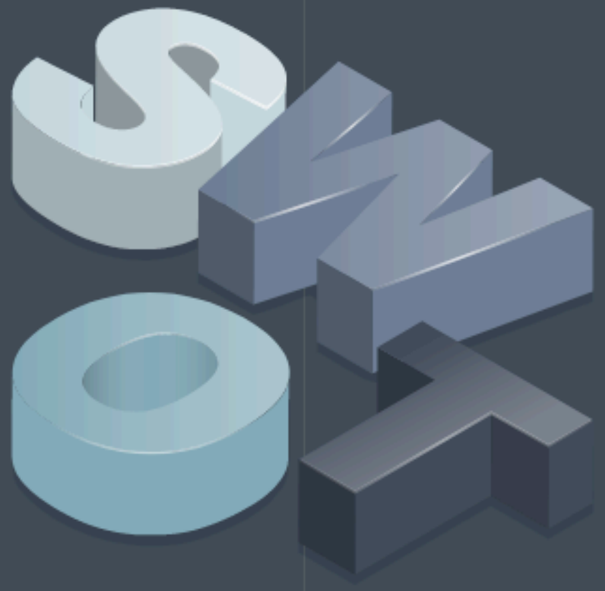
Analyse Comparative

Comparaison des Outils d'Analyse de Code

Points Forts de Chaque Outil

Sécurité des dépendances
Snyk assure la sécurité des dépendances utilisées dans le code.

Vérifications rapides et légères
Les Linters effectuent des vérifications rapides et légères pour identifier les erreurs potentielle.



Qualité globale du code
SonarQube permet d'évaluer et d'améliorer la qualité globale du code source.

Solution de sécurité complète
Checkmarx propose une solution de sécurité complète pour garantir la protection du code contre les vulnérabilités.

Caractéristique	SonarQube	CheckMarx	Snyk
Analyse	Code + vulnérabilité	Sécurité	Vulnérabilité (conteneurs, IaC..)
Besoin équipe	Dev. + produit	Dev. + produit	Dev + infra
Cible	Code & dépendances (plugin)	Code & dépendances	Dépendance, conteneurs
Rapports effectués	Qualité de code, aperçu complet et synthétique du projet & sécurité de code basique	Sécurité du code (détaillée)	Vulnérabilité des dépendances & conteneurs

SonarQube pourrait-il couvrir tous nos besoins ?

- Je n'ai pas pu faire de test concernant le plugin de dépendances SonarQube

Mais nous savons que :

- SonarQube et équivalents s'appuient principalement sur : CWE (Common Weakness Enumeration)
- Snyk et équivalents s'appuient principalement sur : CVE (Common Vulnerabilities and Exposures)
- Mais chaque service possède des avantages/inconvénients différents, mais le cœur d'analyse reste similaire.

CONCLUSION

L'outil d'analyse de code à retenir :

- SonarQube est un outil déjà mis en place sur le Portail, il reste à déployer.
- SonarQube (version gratuite utilisée) + Complément SonarQube (détection dépendances) est l'équivalent de Checkmarx
- SonarQube reste à exploiter : partir sur cette solution

1 Les prochaines étapes :

- Temps de mise en place par l'Infra (à estimer)
- Atelier technique entre développeurs pour définir les règles et configuration à mettre en place (3 heures)
- Temps fixe de mise en place pour régler la dette technique initiale (à estimer)
- Les corrections à apporter dans le code peuvent ajouter +10% du temps en moyenne
- Flexibilité et juste milieu à trouver pour "contourner" le code

2 Questions annexes :

- Des outils sont-ils déjà mis en place sur les applications de l'OFII ?
- Le Portail peut-il être un POC pour la solution retenue ?
- Doit-on continuer sur ces outils de qualité de code ou doit-on poursuivre une veille ? ou cela se fait du côté de Yamina ?

MERCI !
AVEZ-VOUS DES QUESTIONS ?