CodeQL 2.20.0

https://codeql.github.com/

Outil d'analyse statique pour rechercher des vulnérabilités

Système :	Linux, Windows et macOS
Développé par :	GitHub
Logiciel libre, gratuit pour la recherche.	Personne de contact : fabien.petitcolas@smals.be

Fonctionnalités

CodeQL est un outil d'analyse statique de code basé sur plusieurs décennies de recherche menées par une équipe de l'université d'Oxford. Il peut être utilisé pour analyser automatiquement des applications à la recherche de vulnérabilités et de bogues complexes et pour aider les programmeurs à effectuer une révision manuelle du code.

CodeQL permet de rechercher facilement des faiblesses communes de la base CWE de MITRE dans différents langages de programmation comme C, C++, Go, Java, et Python. L'outil permet enfin la recherche de variantes : si une vulnérabilité est identifiée dans une base de code, il est possible de réutiliser le modèle de détection de cette vulnérabilité dans d'autres bases de code. CodeQL peut donc être utilisé pour assister les programmeurs lors de la révision manuelle du code et répondre à des questions comme : Quelle est ma surface d'attaque ? Quelles sont les entrées non sécurisées (source) fournies par l'utilisateur dans ma base de code ? Quelles fonctions dangereuses (récepteurs) utilisent ces données sources ?

CodeQL est disponible sous forme d'extension pour IntelliJ et Visual Studio Code. CodeQL utilise un langage de requêtes déclaratif similaire à SQL. Il est souvent utilisé par des chercheurs en sécurité pour identifier des vulnérabilités encore inconnues.

Voici les fonctionnalités clés de CodeQL :

- Analyse statique avancée du code : Détection des vulnérabilités connues et inconnues, recherche des failles de sécurité telles que les injections SQL, les XSS et les dépassements de mémoire tampon.
- Représentation du code sous forme de base de données : Chaque élément du code (fonctions, classes, variables, etc.) est représenté sous forme de tables et de relations.
- Bibliothèque de requêtes prêtes à l'emploi : CodeQL inclut une vaste bibliothèque de requêtes prédéfinies pour détecter des faiblesses courantes (CWE).
- CodeQL peut être intégré aux processus d'automatisation de développement (par ex. Jenkins).
- Visualisation des chemins d'exécution : Analyse les flux de données pour comprendre comment une donnée se propage à travers un programme.

Conclusions & recommandations

CodeQL est un outil puissant pour renforcer la sécurité du code, automatiser les analyses et détecter efficacement les vulnérabilités grâce à une approche basée sur les requêtes analytiques. Cependant la courbe d'apprentissage assez abrupte et la création de nouvelles requêtes CodeQL requière un temps d'adaptation non négligeable. GitHub fournit beaucoup de contenu de formation.

Tests et résultats

Nous avons testé CodeQL sur une machine virtuelle Kali Linux avec la base de code Java appelée Webgoat. Celle-ci contient volontairement des vulnérabilités. L'installation est relativement simple et l'ajout d'une extension à VSCode (gratuite) ou IntelliJ (payante après période d'essai) très facile. CodeQL est aussi intégré à GitHub.

Avant l'analyse, il faut d'abord créer une base de données à partir d'une base de code. Cela peut se faire à partir de la ligne de commande :

```
codeql database create ./my_db --language=java --source-root=.
```

CodeQL permet alors d'interroger le code comme s'il s'agissait de données. Voici un exemple permettant de trouver tous les appels à la fonction « `execute` » des objets « `java.sql.Statement` » dont l'unique argument n'est pas un chaîne littérale de charactères :

```
import java
import semmle.code.java.dataflow.DataFlow

from MethodAccess call, Expr arg
where
  call.getMethod().getName() = "execute" and
  call.getMethod().getDeclaringType().hasQualifiedName("java.sql", "Statement") and
  call.getArgument(0) = arg and
  not arg instanceof StringLiteral
select call, arg
```

Cet exemple naïf permet donc de trouver des lignes de code susceptibles d'être vulnérables à des injections de code. Il faudrait cependant l'améliorer en identifiant précisément les données entrées par l'utilisateur malveillant. Pour cela CodeQL offre la capacité à décrire les flux de code d'une source à un récepteur, même si les variables sont réattribuées ou modifiées plusieurs fois.

CodeQL couvre un grand nombre des faiblesses identifiées par le MITRE. Il n'est donc pas nécessaire d'écrire des requêtes particulières pour celle-ci. On peut appliquer les requêtes prédéfinies sur la base crée précédemment de la manière suivante :

```
codeql database analyze ./my_db java-security-and-quality.qls --format=sarif-latest
 --output=results.sarif
```

Le fichier `java-security-and-quality.qls` est un fichier de configuration incluant une suite de requêtes prédéfinies. Le résultat de l'analyse peut être un fichier au format SARIF ou CSV. Le fichier SARIF contient des liens vers les lignes de code source incriminées. Dans le cas du projet WebGoat on trouve par exemple l'ensemble des requêtes SQL dépendant de valeurs fournies par l'utilisateur et plus généralement la plupart des vulnérabilités présentes dans le projet WebGoat.

L'utilisation de CodeQL devient plus complexe lorsque l'on souhaite créer soi-même des requêtes spécifiques. Le langage de requêtes de CodeQL est très puissant et son apprentissage requiert un temps et des efforts non négligeables. Il existe cependant beaucoup de documentations détaillées et de cours en ligne.

Conditions d'utilisation & budget

L'utilisation de CodeQL est gratuite pour la recherche. En revanche l'utilisation dans le cadre d'un processus d'ingénierie normal requiert une licence « *GitHub Advanced Security*. »