CodeQL介绍

1. 简介

CodeQL 是一个**语义代码分析引擎**，它的核心思想是：**将代码转换为一个可以查询的数据库。**

1. **代码 -> 数据库**：首先，CodeQL 会将你的源代码（如 Java、C/C++、C#、Python、JavaScript 等）解析并转换成一个关系型数据库。这个数据库并不存储代码本身，而是存储代码元素（如变量、函数、类、表达式等）以及它们之间的逻辑关系（如“A 函数调用了 B 函数”、“变量 X 是类 Y 的实例”）。
2. **编写查询 -> 执行搜索**：然后，开发者使用一种名为 **QL** 的声明式、面向对象的查询语言，来编写“查询规则”。这些规则用于描述他们想要在代码中寻找的特定模式，例如安全漏洞、代码错误或不遵守的编码规范。
3. **分析结果**：最后，CodeQL 引擎在生成的代码数据库上执行这些查询，并列出所有匹配该模式的代码位置，即潜在的缺陷或漏洞。

2. 环境配置

2.1 CodeQL-cli

首先配置命令行的cli。

直接到官方仓库（https://github.com/github/codeql-cli-binaries/releases）下载对应版本的.zip包，解压后添加环境变量，得到如下效果即成功：

```
C:\Users\liyanqi>codeql --version
CodeQL command-line toolchain release 2.23.2.
Copyright (C) 2019-2025 GitHub, Inc.
Unpacked in: F:\CodeQL\codeql
   Analysis results depend critically on separately distributed query and
   extractor modules. To list modules that are visible to the toolchain,
   use 'codeql resolve packs' and 'codeql resolve languages'.
```

2.2 CodeQL-SDK

可以理解为一个demo仓库，有很多现成的ql规则。

下载链接：https://github.com/github/codeql

在SDK根路径下打开cmd命令行窗口，执行命令`codeql pack ls`可以看到当前SDK中默认支持的规则集

如果发现一个包存在于多个路径，则需要删除或指定搜索路径，否则会引发冲突

2.3 运行demo（cmd）

推荐单个目录结构如下：

```
-[rootpath]
    -db           # 存放数据库
    -sourcecode   # 待检测源码
    -query        # 针对db的查询
        -qlpack.yml  # 必要的配置文件
    -results      # 生成的结果
```

`qlpack.yml`参考内容如下：

```
name: python-dataflow-queries  # 自己定（删除#注释）
version: 0.0.1
dependencies:
  codeql/python-all: "*"   # 还有codeql/java-all: "*"等
```

`sourcecode` to `db` demo:

```
codeql database create F:\codeql3\1\db\python-sql_injection --language=python --
source-root F:\codeql3\1\sql_injection
```

指令解释：将source-root下的源码以db形式命名为python-sql_injection保存在`F:\codeql3\1\db\`路径下

第三方资料补充：

执行命令，将项目代码生成数据库形式：

> codeql database create db-ql --language=java --command= “mvn clean install”

主要参数：

- –command 参数如果不指定，会使用默认的编译命令和参数
- –source-root 源码路径
- –overwrite 表示 create 的目标 database 对已有的 database 做覆盖
- –language 要根据具体项目的编译语言指定

Language对应关系如下：

Language	Identity
C/C++	cpp
C#	csharp
Go	go
Java	java
javascript/Typescript	javascript
Python	python

添加`--command`，需要本机有对应环境。

use `.ql` to analyze `db` demo:

```
codeql database analyze F:\codeql3\1\db\python-sql_injection --format=csv --
output=F:\codeql3\1\results\sql_injection_results.csv
F:\codeql3\1\queries\sql_injection.ql --rerun --search-path
C:\Users\liyanqi\.codeql\packages\
```

注意：如果之前生成过一次，需要添加`--overwrite`或`--rerun`之类的命令，否则新命令不能运行。

如果要运行demo，找到SDK或者其他仓库的security目录下，例如：`F:\CodeQL\codeql-main\cpp\ql\src\Security\CWE\CWE-327`，然后把源码和ql分别cp到对应位置。

还有一种用vscode插件，参考链接：https://www.freebuf.com/articles/web/402726.html

3. CodeQL规范

ql基本长这样：

```
/**
 *
 * Query metadata
 *
 */

import /* ... CodeQL libraries or modules ... */

/* ... Optional, define CodeQL classes and predicates ... */

from /* ... variable declarations ... */
where /* ... logical formula ... */
select /* ... expressions ... */
```

相关语法介绍参考链接：

CodeQL library for Python — CodeQL

Analyzing data flow in Python — CodeQL

根据上述官方文档来看，是能够直接进行污点分析，得到source->path->sink全路径的。

其他Reference

【1】ASTTeam/CodeQL: 《深入理解CodeQL》Finding vulnerabilities with CodeQL.

【2】Python Demo: codeql/python/ql/src/Security at codeql-cli/latest · github/codeql

【3】Java Demo: codeql/java/ql/src/Security/CWE at codeql-cli/latest · github/codeql

【4】C&Cpp Demo: codeql/cpp/ql/src/Security/CWE at codeql-cli/latest · github/codeql