Java代码审计

一、什么是代码审计

顾名思义就是检查源代码中的安全缺陷，检查程序源代码是否存在安全隐患，或者有编码不规范的地方，通过自动化工具或者人工审查的方式，对程序源代码逐条进行检查和分析，发现这些源代码缺陷引发的安全漏洞，并提供代码修订措施和建议。

代码审计的主要目的是在软件发布前尽可能减少错误，提高软件的安全性和稳定性。这一过程可能涉及使用自动化工具或手动审查源代码，以检查安全缺陷、性能瓶颈、代码结构问题等。例如，在安全方面，代码审计可能涉及检测跨站脚本攻击(XSS)、SQL注入、代码注入、拒绝服务攻击(DoS)等安全漏洞，以及评估密码安全、会话管理□和权限控制等。

此外，代码审计还包括对第三方组件和开源库的审计，以确保它们的安全性和可靠性，并符合相关的法律法规和行业标准。通过代码审计，开发团队可以及时发现和解决代码中的问题，从而提升软件的整体质量。

二、CodeQL

1.简介

CodeQL是一个源代码白盒审计工具，以一种非常新颖的方式组织代码与元数据，使研究人员能够“像查询数据库一样检索代码”，并发现其中的安全问题。CodeQL的核心思想是使用数据流分析技术去发现代码中的潜在安全问题和漏洞，它可以在编译时或者运行时对代码进行分析，找出潜在的程序缺陷和安全隐患。CodeQL不仅支持多种编程语言，而且提供了大量的现成代码库和规则，使开发人员可以快速地构建和扩展自己的代码分析平台。

CodeQL是github的一个开源项目，是Github为了解决其托管的海量项目的安全性问题，收购了CodeQL的创业公司，并宣布开源了CodeQL的规则部分，这样全世界的安全工程师就可以贡献高效的QL审计规则给Github，帮助它解决托管项目的安全问题。

对于安全工程师，也就多了一个非商业的开源代码自动化审计工具。CodeQL支持非常多的语言，在官网有如下支持的语言和框架列表。

Language	Variants	Compilers	Extensions
C/C++	C89, C99, C11, C17, C++98, C++03, C++11, C++14, C++17, C++20 [1] [2]	Clang (including clang-cl [3] and armclang) extensions (up to Clang 12.0), GNU extensions (up to GCC 11.1), Microsoft extensions (up to VS 2019), Arm Compiler 5 [4]	`.cpp`, `.c++`, `.cxx`, `.hpp`, `.hh`, `.h++`, `.hxx`, `.c`, `.cc`, `.h`
C#	C# up to 12	Microsoft Visual Studio up to 2019 with .NET up to 4.8, .NET Core up to 3.1 .NET 5, .NET 6, .NET 7, .NET 8	`.sln`, `.csproj`, `.cs`, `.cshtml`, `.xaml`
Go (aka Golang)	Go up to 1.22	Go 1.11 or more recent	`.go`
Java	Java 7 to 22 [5]	javac (OpenJDK and Oracle JDK), Eclipse compiler for Java (ECJ) [6]	`.java`
Kotlin [7]	Kotlin 1.5.0 to 1.9.2x	kotlinc	`.kt`
JavaScript	ECMAScript 2022 or lower	Not applicable	`.js`, `.jsx`, `.mjs`, `.es`, `.es6`, `.htm`, `.html`, `.xhtm`, `.xhtml`, `.vue`, `.hbs`, `.ejs`, `.njk`, `.json`, `.yaml`, `.yml`, `.raml`, `.xml` [8]
Python [9]	2.7, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12	Not applicable	`.py`
Ruby [10]	up to 3.3	Not applicable	`.rb`, `.erb`, `.gemspec`, `Gemfile`
Swift [11]	Swift 5.4–5.9.1	Swift compiler	`.swift`
TypeScript [12]	2.6–5.4	Standard TypeScript compiler	`.ts`, `.tsx`, `.mts`, `.cts`

2.原理

CodeQL是将代码转化成类似数据库的形式，并基于该database进行分析。

在 CodeQL 中，代码被视为数据。安全漏洞、Bug 和其他错误被建模为可以从代码中提取的数据库执行的查询。

> CodeQL 的整体思路是把源代码转化成一个可查询的数据库，通过 Extractor 模块对源代码工程进行关键信息分析提取，构成一个关系型数据库。CodeQL 的数据库并没有使用现有的数据库技术，而是一套基于文件的自己的实现。
> 对于编译型语言，Extractor 会监控编译过程，编译器每处理一个源代码文件，它都会收集源代码的相关信息，如：语法信息（AST 抽象语法树）、语意信息（名称绑定、类型信息、运算操作等），控制流、数据流等，同时也会复制一份源代码文件。
> 而对于解释性语言，Extractor 则直接分析源代码，得到类似的相关信息。
> 关键信息提取完成后，所有分析所需的数据都会导入一个文件夹，这个就是 CodeQL database, 其中包括了源代码文件、关

> 系数据、语言相关的 database schema（schema 定义了数据之间的相互关系）。

3.安装

CodeQL本身包含两部分：解析引擎 + SDK 。

- 解析引擎：不开源，解析我们编写的规则，但是可以直接在官网下载二进制文件直接使用。
- SDK(规则库)：完全开源，里面包含大部分现成的漏洞规则，我们也可以利用其编写自定义规则。

3.1安装解析引擎CodeQL CLI

CodeQL CLI用来创建和分析数据库。

下载地址：https://github.com/github/codeql-cli-binaries/releases/download/v2.17.0/codeql-win64.zip

在Java配套教学工具中有下载好的安装包`codeql-win64.zip`，将`codeql-win64.zip`解压之后的文件夹放在C盘根目录，并重命名为`codeql`，然后将`C:\codeql`加入系统环境变量中，方便灵活调用。

执行codeql命令，如下图所示。

```
C:\Users\guo>codeql
Usage: codeql <command> <argument>...
Create and query CodeQL databases, or work with the QL language.

GitHub makes this program freely available for the analysis of open-source
software and certain other uses, but it is not itself free software. Type
codeql --license to see the license terms.

      --license             Show the license terms for the CodeQL toolchain.
Common options:
  -h, --help                Show this help text.
  -v, --verbose             Incrementally increase the number of progress
                              messages printed.
  -q, --quiet               Incrementally decrease the number of progress
                              messages printed.
Some advanced options have been hidden; try --help -v for a fuller view.
Commands:
  query       Compile and execute QL code.
  bqrs        Get information from .bqrs files.
  database    Create, analyze and process CodeQL databases.
  dataset     [Plumbing] Work with raw QL datasets.
  test        Execute QL unit tests.
  resolve     [Deep plumbing] Helper commands to resolve disk locations etc.
  execute     [Deep plumbing] Low-level commands that need special JVM options.
  version     Show the version of the CodeQL toolchain.
  generate    Commands that generate useful output.
  github      Commands useful for interacting with the GitHub API through
                CodeQL.
  pack        Commands to manage QL packages.
  diagnostic  [Experimental] Create, process, and export diagnostic information.
```

3.2下载规则库

规则库中包含了必须的一些标准库（内置库）和一些查询样例。

CodeQL的标准库不像其他编程语言一样会与解释器同时安装，而是需要我们手动下载。

下载地址：https://github.com/github/codeql/archive/refs/tags/codeql-cli/latest.zip

在Java配套教学工具中有下载好的压缩包`codeql-codeql-cli-latest.zip`，将解压后得到`codeql-codeql-cli-latest`文件夹放到`C:\codeql`目录下，并重命名为`codeql-sdk`。

三、命令行项目审计

1.确定项目源码

虽然CodeQl的优点非常多，但也存在一个比较明显的缺点：不能直接通过打包好的程序进行代码审计。

当你得到的只是一个jar包，那么会比较棘手，因为现在市面上的反编译程序的反编译结构都不太满意，在我个人的使用体验中感觉idea算是最完美的了，但是还可能存在一些反编译问题，比如语法错误，最终可能导致编译过程中CodeQl解析的不准确。所以这里提供两个思路：

- 直接审计源代码
- 将编译好的文件反编译后，再执行审计

在本节课中，我们将通过直接审计源代码去发现其中的漏洞。源代码文件是Java配套教学工具包中的`java-sec-code-master.zip`，将解压之后的文件夹放在`C盘`下并重命名为`C:\java-sec-code`。

2.安装Maven

Maven是Java源代码的编译工具，CodeQL在创建数据库时，会自动探测或使用指定的编译方式，对源代码进行分析。在编译的过程中构建一个抽象的语法树，在抽象的语法树的基础上，CodeQL进行语义标记，比如变量的定义和使用、函数调用等。当前的项目源码我们需要自己指定CodeQL使用Maven去执行编译。

下载地址：https://archive.apache.org/dist/maven/maven-3/3.9.6/binaries/apache-maven-3.9.6-bin.zip

在Java配套教学工具中有下载好的压缩包`apache-maven-3.9.6-bin.zip`，将解压之后的文件夹放到`C盘`的根目录中，并将`C:\apache-maven-3.9.6\bin`路径加入到系统环境变量中。

执行命令测试，如下图所示：

```
C:\Users\guo>mvn -v
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)
Maven home: C:\apache-maven-3.9.6
Java version: 1.8.0_391, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-1.8\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

3.生成数据库

CodeQL只可以对解析引擎编译产生的数据库进行扫描，所以需要先生成目标数据库。

语法结构：

```
codeql database create 数据库名 --language=java --source-root=源码路径 --command="编译命令" --overwrite
```

参数说明：

- --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

对应关系如下：

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

打开CMD，执行下面命令生成审计数据库。

```
codeql database create java_code_db --language=java --source-root=C:\java-sec-code --command="mvn clean install" --overwrite
```

4.漏洞扫描

CodeQL会根据现有的代码数据库，进行对应的漏洞扫描。如果源代码有更新，还需要再次生成审计数据库。

语法结构：

```
codeql database analyze 审计数据库路径 codeql规则库漏洞路径 --format=csv --output=result.csv
```

参数说明：

- analyze：分析数据库，在源代码的上下文中产生有意义的结果
- --format：结果输出格式
- --output：结果文件输出路径
- --threads：多线程运行

打开CMD，执行下面命令扫描数据库并生成结果。

```
codeql database analyze ./java_code_db C:\codeql\codeql-sdk\java\ql\src\Security\CWE\CWE-022 --threads=10 --format=csv --output=result.csv
```

扫描的时候，可以扫描指定的漏洞，也可以扫描全部漏洞。

5.分析结果

代码问题	漏洞	警报级别	触发警报的代码行	源代码文件路径	代码行号
Uncontrolled data used in path expression	Accessing paths influenced by users can allow an attacker to access unexpected resources.	error	This path depends on a [["user-provided value" "relative:///src/main/java/org/joychou/controller/FileUpload.java:62:53:62:78"]].	/src/main/java/org/joychou/controller/FileUpload.java	62

> 解释：
>
> Uncontrolled data used in path expression

> 路径表达式中使用了不受控制的数据
>
> Accessing paths influenced by users can allow an attacker to access unexpected resources.
>
> 访问受用户影响的路径可以使攻击者访问到意外的资源
>
> error
>
> 代码扫描警报级别，警报严重性级别可能为 `Error`、`Warning` 和 `Note`
>
> 其余的每列都会告诉使用者是哪个文件的哪一行触发了漏洞规则。

6.总结

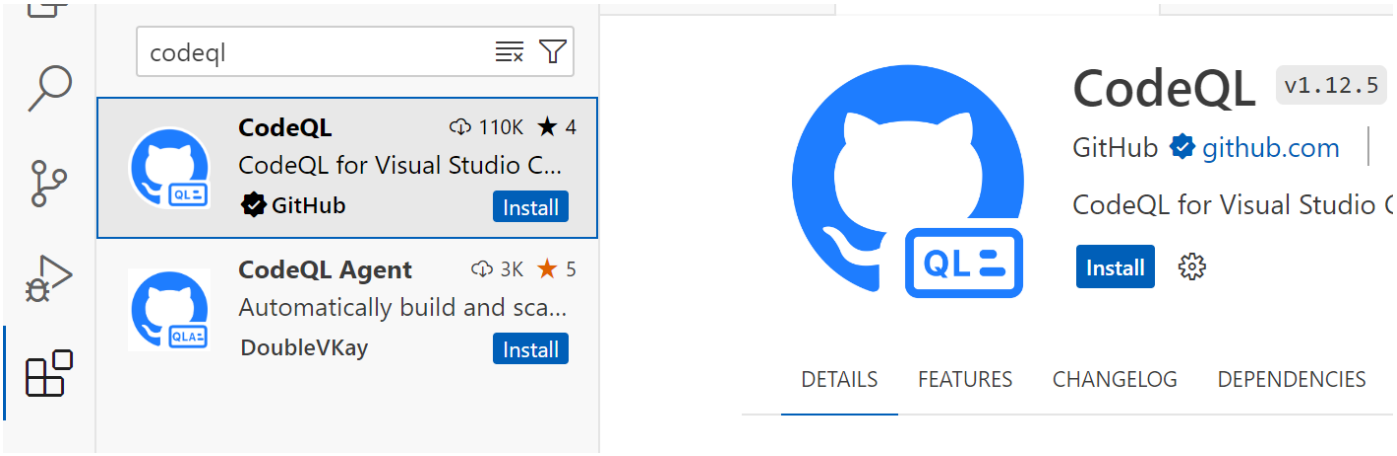
可以看到，对于需要进行详细污点分析的漏洞模式，借助 CodeQL 进行漏洞排查是极其有效的。但是 CodeQL 存在两个具有较大时间成本的环节：编译生成 AST 数据库、编写 QL 规则。当然我们不需要去编写QL规则，开源的规则库已经足够应付中大型项目代码的审计工作，根据数据库去匹配QL规则会更加耗时一些。

最后总结下使用 CodeQL 进行漏洞挖掘的主要流程：

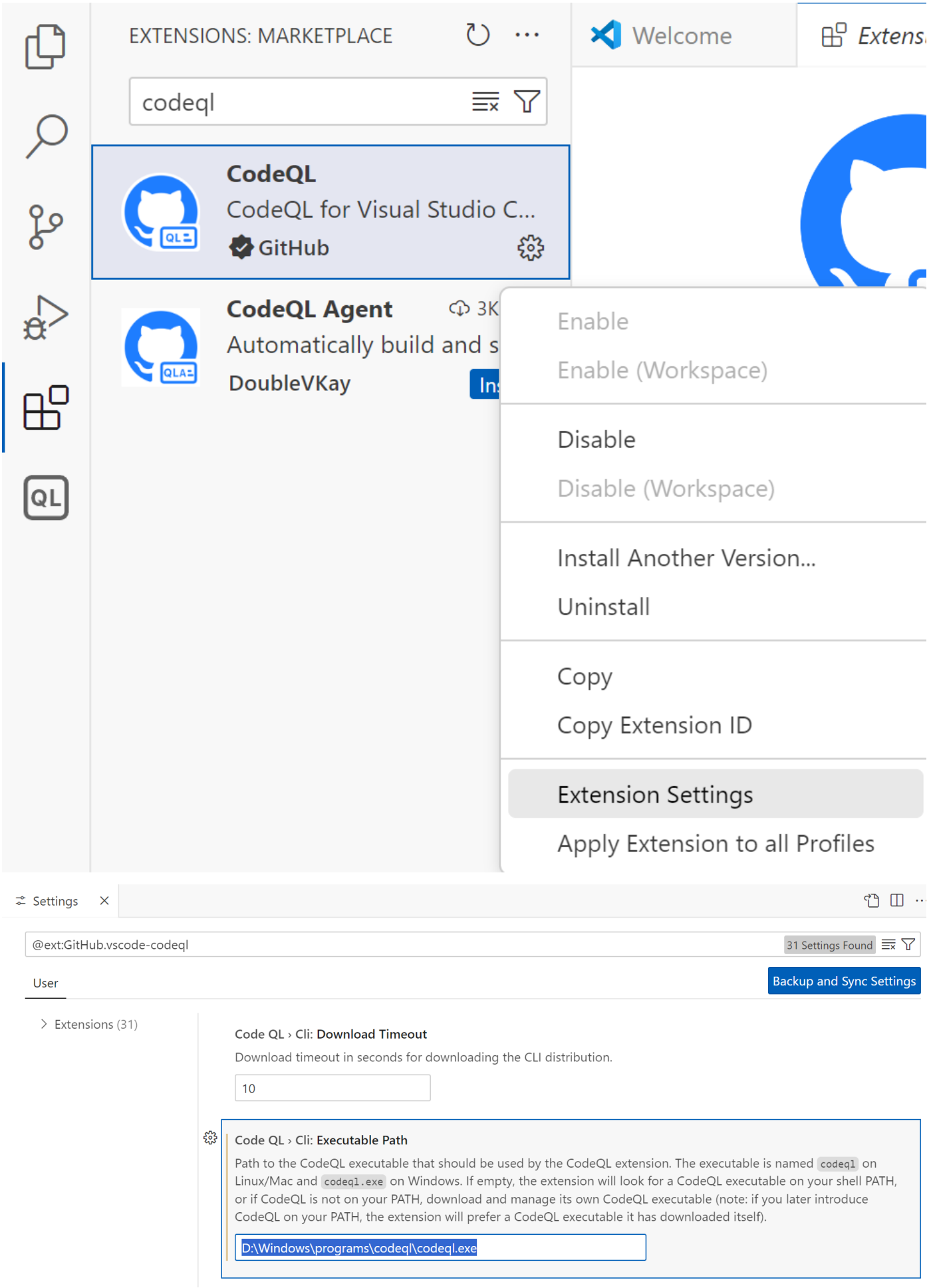
- 使用 CodeQL 命令，根据目标源代码编译生成 database，该 database 包含整个代码的 AST 树，CodeQL 就是查询该 database 来进行漏洞发现的；
- 根据已知漏洞给出的 CodeQL 规则，来查询目标 database 找出类似的漏洞。
- 根据 CodeQL 规则运行的结果进行代码审计，分析目标代码是否真正存在漏洞，确认漏洞的成因及触发条件。

四、VSCode插件代码审计

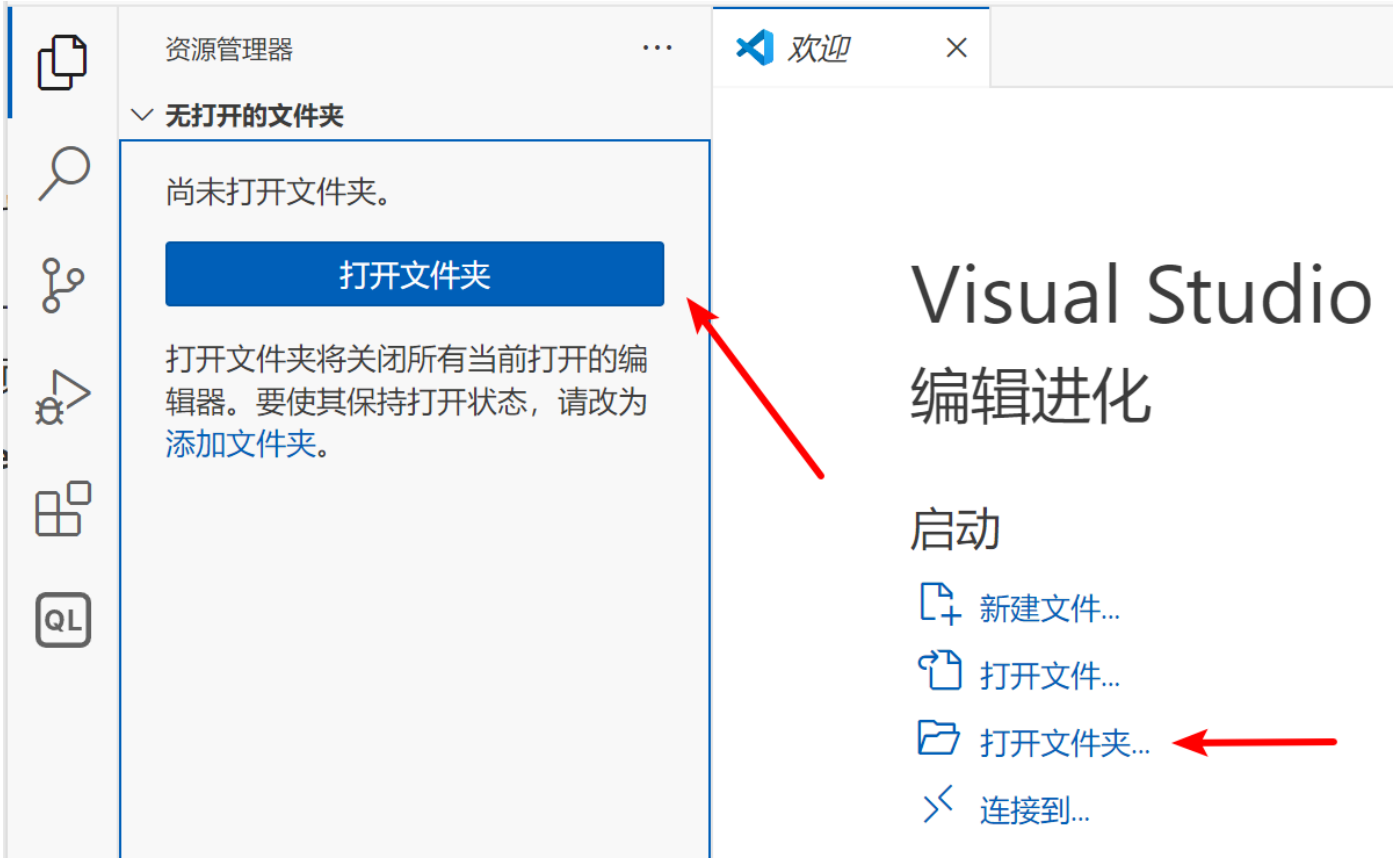
第一步，安装CodeQL插件



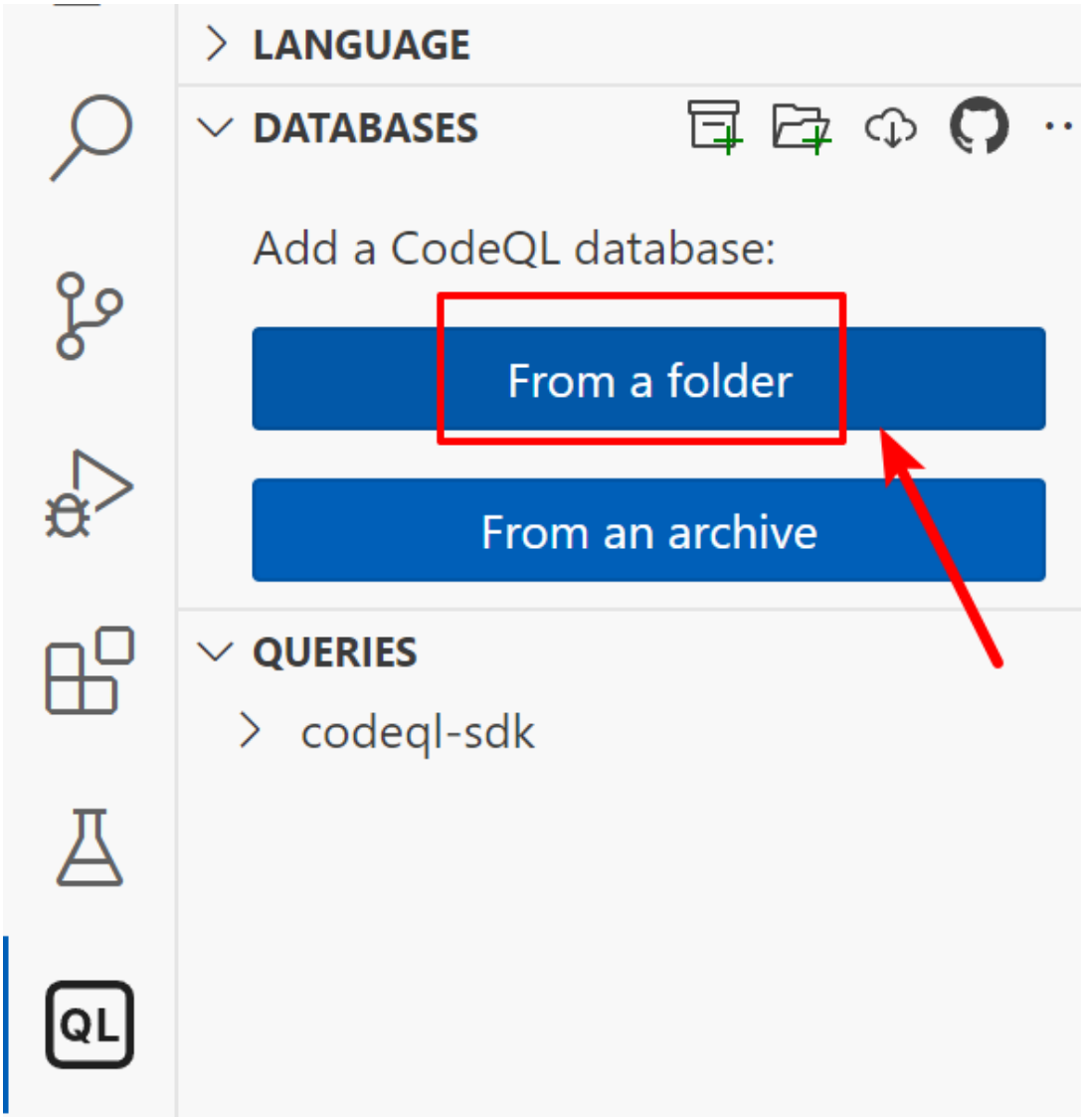
第二步，设置CodeQL命令路径



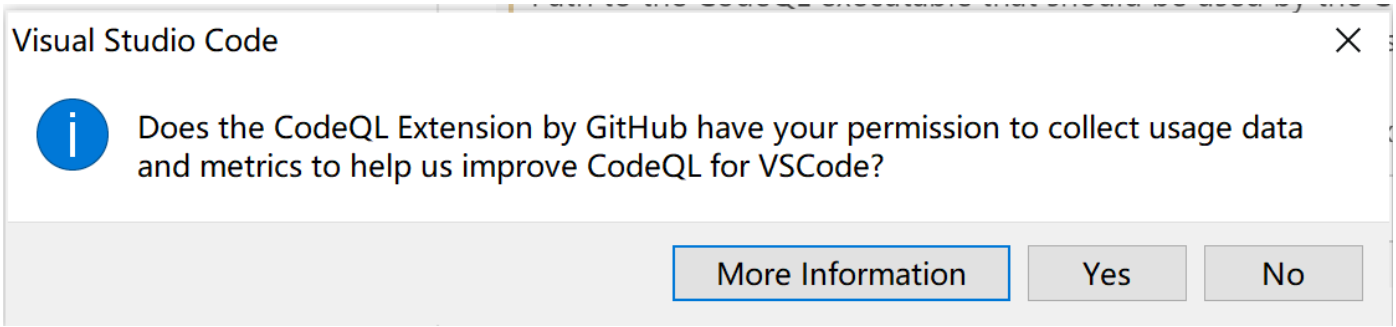
第三步，打开规则库目录C:\codeql\codeql-sdk，选任意一个打开文件夹即可



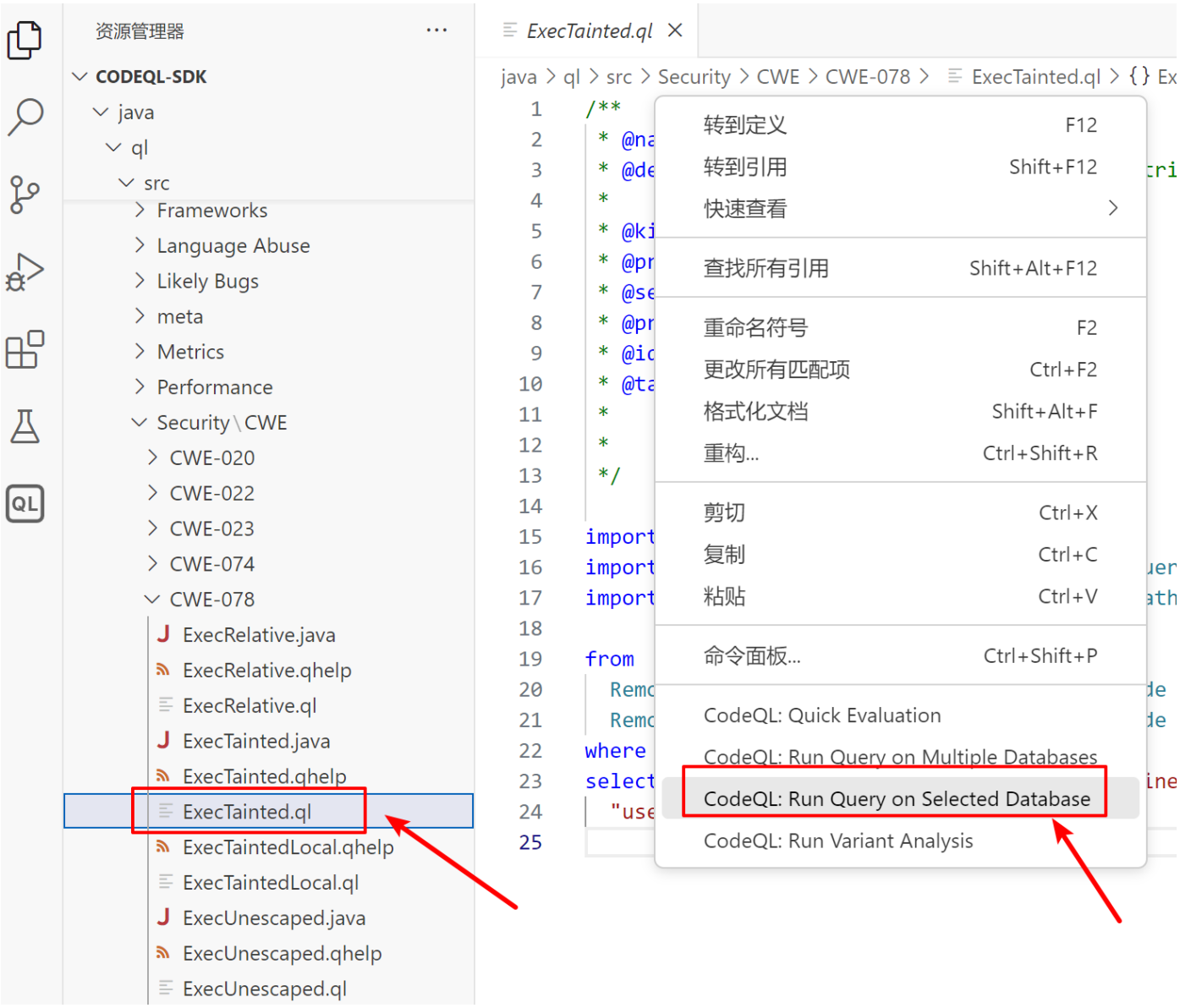
第四步，导入第三节生成的代码数据库java_test。



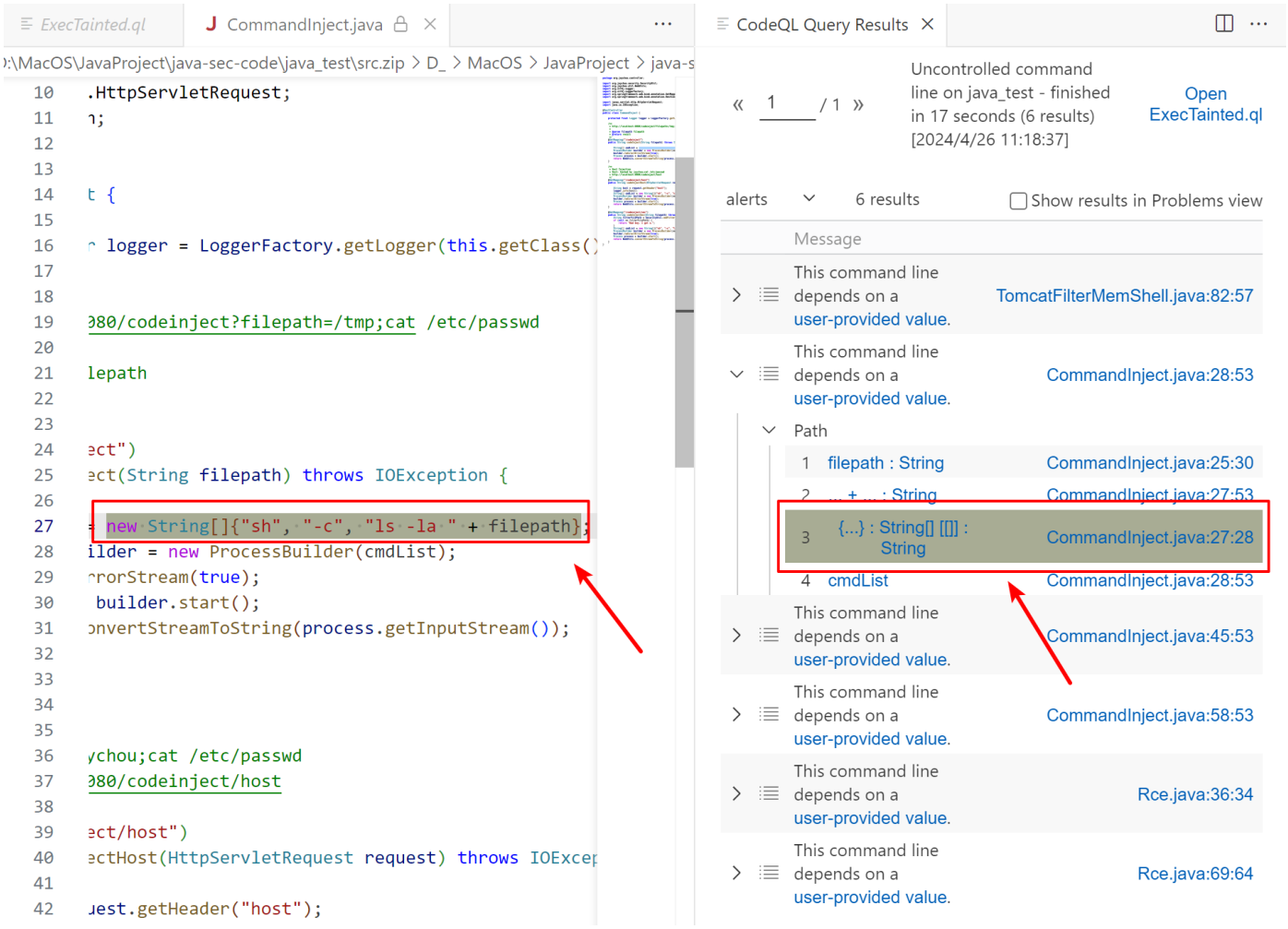
如果弹出下面的警告框，点击“No”。



第五步，根据规则库中的规则，验证代码数据库中的漏洞。vscode回到资源管理器，选择`codeql-sdk`目录下的`java\ql\src\Security\CWE\CWE-078`漏洞，选择其中一条ql规则，鼠标右键点击选择`CodeQL:Run Query on Selected Database`，开始查询。



第六步，根据查询给出的结果，查看代码哪里出现问题。



在分析结果中提示“Uncontrolled command line on java_test”，意思是，java_test中找到不受控制的命令行。

当然，也可以直接使用整个目录的代码查询规则，比如在`CWE-078`目录上点击鼠标右键，选择`CodeQL:Run Queries in Selected Files`，可以使用整个目录下所有的代码查询规则去分析代码数据库。