SWRhapsody

CodeQL 部分使用记录

于5月 15, 2020由**SWRhapsody**发布

前言

CodeQL 是一个代码分析引擎，主要原理是通过对代码进行构建并创建一个数据库，然后通过编写 QL 查询语句来查询数据库。

相关链接

Doc查询
https://help.semmle.com/home/help/search-results-qldoc.html?addsearch=call
WIKI
https://help.semmle.com/wiki
Query Console + Database Download
https://lgtm.com/query

安装

官方文档

https://help.semmle.com/codeql/codeql-cli/procedures/get-started.html

首先创建一个 workspace 的文件夹

1. 下载 codeql-cli-binaries 到该文件夹下，解压并将解压出来的文件夹重命名为 codeql-cli [1]
2. 创建一个 codeql-go-repo 的文件夹，将codeql-go [2] 克隆到该文件夹下
3. 创建一个 codeql-repo 的文件夹，将codeql ql [3] 克隆到该文件夹下
4. 将 workspace\codeql-cli 添加到path 中，运行下面的命令来查看安装情况

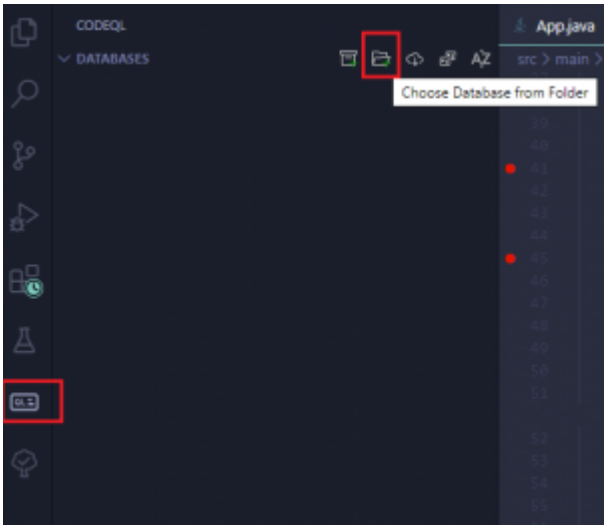
```
codeql version
codeql resolve languages
```

SWRhapsody

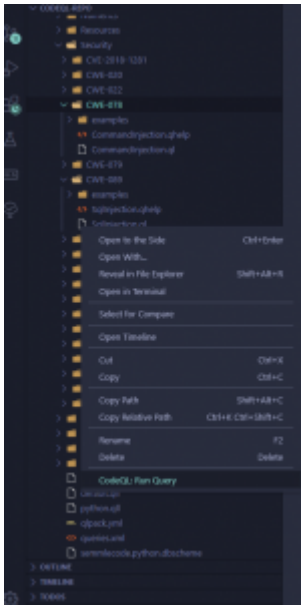
编写语句与调试的时候建议使用VSCode 的插件，在生成相关项目的根目录后运行(以python为例)

```
codeql database create -l=python workspace\db\{项目名}
```

生成数据库后在vscode 的插件中添加生成的数据库，选择你的 workspace\db\{项目名}

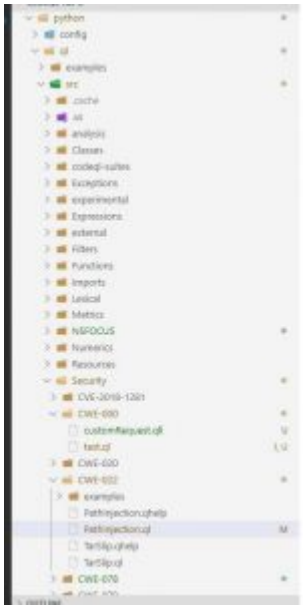


选中后再右键运行你想执行的语句（最底下的CodeQL：Run Query）



目录结构与文件类型

SWRhapsody



上图是 CodeQL python 部分的目录结构，python 相关的所有的 QL规则 和 QL 语句的库都在 python/ql/src 的目录下。src 目录下的每一个文件夹相当于一个包。

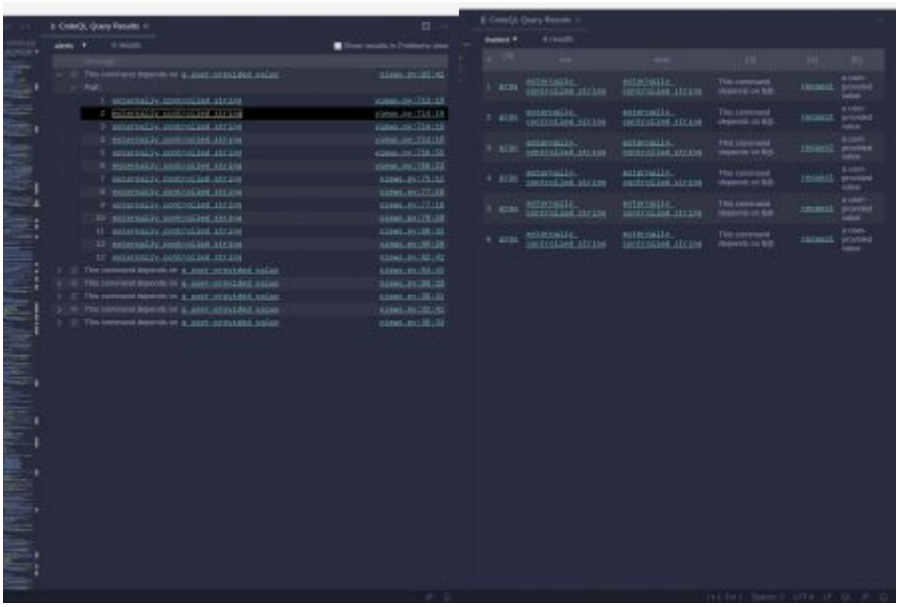
对于文件来说 *.ql 是一条单独的 QL 规则，*.qll 是一个 QL 语句的库文件，用于抽取通用的函数和功能，*.qhelp 是一个说明，相当于 Readme。
对于我们安全工作者可能主要的工作都在security目录下。

规则文件结构

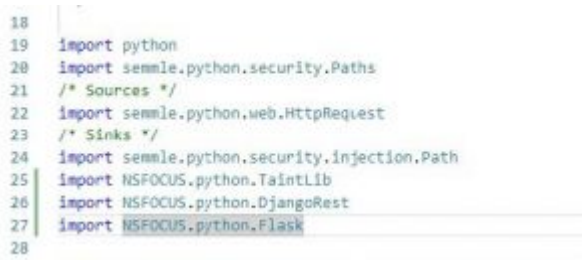
对于单个的 *.ql 文件，大体的结构如下

SWRhapsody

红色框内为对于这个规则的说明以及一些必要的信息，需要注意的是如果你想要在查询出来的结果中有比较好的展示，这部分是必须的。下图中左图是带这一段后的结果，右图是不带这一段的结果。可以看到如果没有这一段的注释，CodeQL的插件是不会帮你把污点传播的路径详细列出来，只会展示开始和结束。PS. 经常会出现同一条链重复出现的情况，不知道是不是我语句写的有问题。



绿色框内是 import 部分



```
18
19  import python
20  import semmle.python.security.Paths
21  /* Sources */
22  import semmle.python.web.HttpRequest
23  /* Sinks */
24  import semmle.python.security.injection.Path
25  import NSFOCUS.python.TaintLib
26  import NSFOCUS.python.DjangoRest
27  import NSFOCUS.python.Flask
28
```

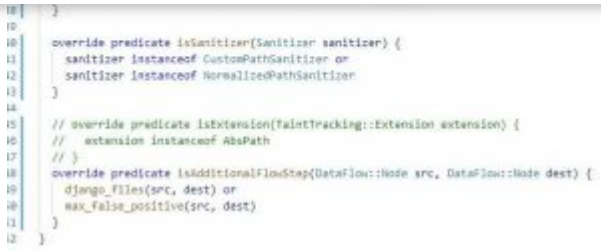
上图是 python\ql\src\Security\CWE-022\PathInjection.ql 中的一部分

```
1  import NSFOCUS.python.Flask
```

等于在 PathInjection.ql 中引入 python\ql\src\NSFOCUS\python\Flask.qll 库中所有的 public 的函数。

红色框内为本规则中的其他函数，taintconfig 一般也放在这一段

SWRhapsody



最后为黑色框内放置规则的主体查询语句。

其他

CodeQL 的和普通的正则规则是不一样的，不是越多越好，而是越精越好。同一类型的漏洞只用一条规则，此规则的泛用性越强越好。

坑

使用过程中目前踩到的坑：

1. CodeQL 对于 Java 支持最好的是 maven 和 gradle。如果你使用其他编译工具如 bazel，CodeQL 可能根本无法编译它的数据库。
2. CodeQL 自带的JDK 是 13 版本的，并且自带 JDK 的 lib 中的class 比官方的JDK要少一些，不从脚本启动直接使用会报各种错误。
3. Vscode CodeQL 这个插件特别吃内存，有的时候一条语句能吃 5G 的内存。
4. CodeQL（v2.1.0） 对于 python 的支持很差，不支持 partial 对于部分 django rest_framework 的分析存在错误，这些viewset 的父类识别出来
是 builtin object 而不是具体的 rest_framework 的类。
5. 在对 python 的解析过程中有一定的bug，不知道后续版本会不会改（>2.1.0），os.path.join 用时候用 Value::named 搜不出来，但先 from os.path import join 就可以找到 join。
6. 在给官方提 issue 时官方人员有聊到使用CodeQL 最好把你要审计软件的依赖都装齐，不然会出现污点传播无法继续的情况。
7. CodeQL 对于 python 包的解析有点麻烦，比如你的文件中有 import org.app.test， 你的目录结构可能需要是 folder_xxx/org/** ，并在在 folder_xxx 中运行 codeql database create，直接在 org 目录下会导致工具无法正确识别引用。

Reference

[1] https://github.com/github/codeql-cli-binaries/releases

[2] https://github.com/github/codeql-go

SWRhapsody

分类: CHEAT SHEET EXERCISE

0 条评论

发表评论

名称 *

电子邮件 *

网站

在想些什么?

发表评论

近期文章

CVE-2020-5410

CodeQL部分源码简读

服务器与网关的不一致

CodeQL 部分使用记录

近期评论

文章归档

2020年8月

2020年6月

2020年5月

2020年3月

2020年1月

2019年12月

2019年11月

2019年8月

2019年7月

2019年5月

2019年4月

2019年1月

2018年11月

2018年10月

2018年9月

2018年4月

2018年3月

SWRhapsody

分类目录

- Article Collection
- Cheat Sheet
- cryptography
- Exercise
- Exploit
- HackTheBox
- Penetration Test
- Uncategorized

相关文章

EXERCISE

CodeQL部分源码简读

Introduction CodeQL 也用了不少时间了，从最初的不会 阅读更多…

在复现 CVE-2019-15012 遇到了一个非常坑的地方，使用 J 阅读更多…

EXERCISE

OpenRASP

前言 记得我还没想过要从事安全行业时看过一个黑客的博客，博主说他找到了 阅读更多…