codeql-1

CodeQL

数据库
以XNU内核为例，下载好codeql工具后，创建数据库只需要输入如下命令

```
codeql database create xnu-10.15.1 --language=cpp --command="make install SDKROOT=macosx ARCH_CONFIGS=X86_64 KERNEL_CONFIGS=RELEASE"
```

- 编译
  之后在VS Code里面安装CodeQL插件，并选择目标数据库即可。需要注意的是，在编译之前需要确认是全新编译而不是增量编译，否则会导致只有一部分编译结果被保存到数据库。
- 创建项目
  首先用vscode打开文件夹，并克隆如下依赖仓库到当前目录，vscode会自动检测

```
git clone https://github.com/github/codeql/

mkdir queries && cd queries
```

在queries目录下创建queries.xml文件，内容如下

```
<queries language="cpp"/>
```

在queries目录下创建qlpack.xml文件，内容如下

```
name: xnu

version: 0.0.0
```

```
libraryPathDependencies: codeql-cpp

extractor: cpp
```

之后就可以在queries下面编写查询语句了，当然也可以使用Github提供的脚手架项目 https://github.com/github/vscode-codeql-starter/ 。编写完成后在Command Palette中输入CodeQL: Run Query即可执行。

- 查询语句

CodeQL可以完成的查询功能很多，其语法和SQL类似，并额外扩充了函数和类的定义语法（这一部分语法则和Java/C++类似）。需要用户有比较高的漏洞抽象能力。一些真实的CVE查询语句可以在这里找到: https://github.com/github/securitylab

查询可能的格式化字符串漏洞

如果sprintf函数的fmt参数不是常量字符串，就选择出来

```
import cpp

from FunctionCall fc

where

fc.getTarget().getQualifiedName() = "sprintf" and

not fc.getArgument(1) instanceof StringLiteral

select fc, "sprintf called with variable format string."
```

- 进行数据流分析

需要定义source和Sink，下列查询用于检查是否有类型为tcphdr的参数被作为source流向了mbuf_copydata的参数

```
import cpp

import semmle.code.cpp.dataflow.TaintTracking

import DataFlow::PathGraph
```

```
class Config extends TaintTracking::Configuration {

Config() { this = "tcphdr_flow" }


override predicate isSource(DataFlow::Node source) {

source.asExpr().getType().stripType().getName() = "tcphdr"

}



override predicate isSink(DataFlow::Node sink) {

exists (FunctionCall call

| call.getArgument(2) = sink.asExpr() and

call.getTarget().getName() = "mbuf_copydata")

}

}


from Config cfg, DataFlow::PathNode source, DataFlow::PathNode sink


where cfg.hasFlowPath(source, sink)


select sink, source, sink, "tcp"
```

参考链接

codeQL官方文档 https://help.semmle.com/codeql/codeql-for-vscode.html
Github Security Lab https://github.com/github/securitylab