



O QUE É?

● O que é

- Query Language para identificar vulnerabilidades e erros no código
- Ferramenta de análise de variantes
- Criada pela empresa Semmle, comprada pelo Github em 2019
 - Desenvolvimento do CodeQL como ferramenta open source começou a ser feito após ser comprada
- Gratuito para projetos open source
 - Pode ser rodado localmente de graça para qualquer projeto*
- Diversas queries disponíveis publicamente



*Exceto em algumas condições: <https://securitylab.github.com/tools/codeql/>

● Motivos para aprender

- 180 vulnerabilidades reportadas em 2021 pelo GH SecurityLab (51 CVEs)
- Programas de Bug Bounty:
 - The Bug Slayer (Nova Vulnerabilidade – Até 5000 USD)
 - Ache 4 vulnerabilidades, consiga um CVE para cada e escreva reports.
 - All for one, one for all (Nova Query – Até 6000 USD)
 - Identificar uma classe de vulnerabilidades com alta precisão
 - Bounty médio de 1800 USD
 - 267.800 USD pagos

[SecurityLab/Bounties](#) para mais detalhes

[Hackerone: Github Security Lab](#)

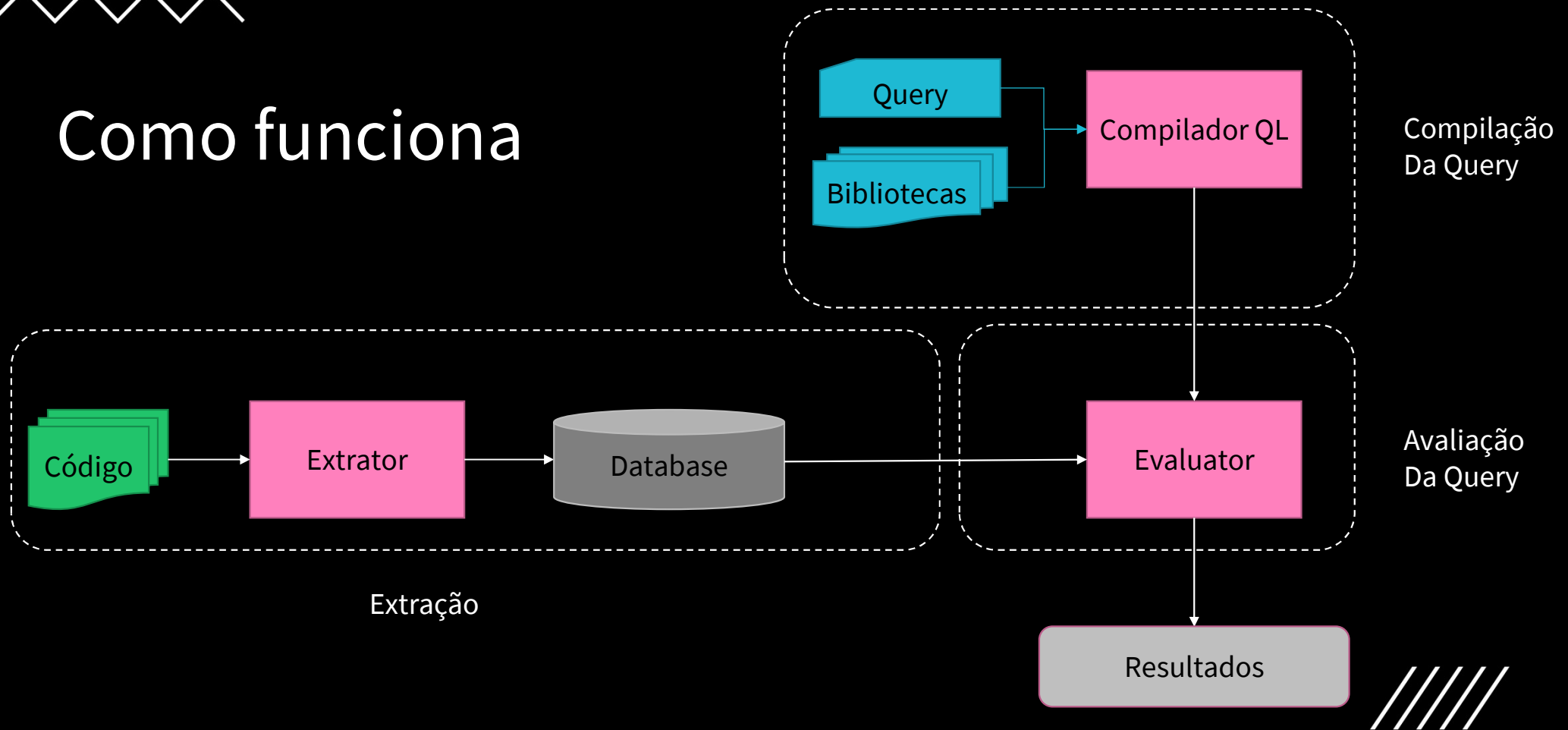


● Linguagens Suportadas

- C/C++
- C#
- Go
- Java
- Javascript/TypeScript
- Python



Como funciona





AS QUERIES



Tipos de Query



Static Analysis

Taint Analysis

DataFlow Analysis

Control Flow Graph Analysis

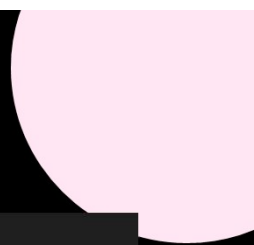
● A estrutura de uma query

- **import** – Importa bibliotecas
- **from** – Declaração de variáveis que terão valores usados para cálculos. Ex: Function, FunctionCall, VariableAccess, Variable, Expr, etc.
- **where** – Condições a serem satisfazidas, é a parte central de uma query.
- **select** – Como e o que vai ser mostrado na saída. O modo mais simples é uma tabela, mas existem outros






EXEMPLE DE QUERY

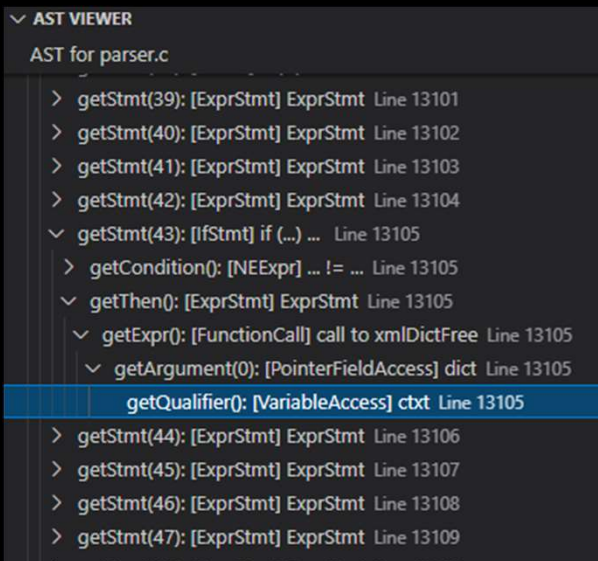


```
✓ /**  
 * @name Block  
 * @kind problem  
 * @problem.severity warning  
 * @id cpp/example/block  
 */  
  
import cpp  
  
from BlockStmt b, int n  
where n = b.getNumStmt()  
select b, "This is a block with " + n +
```



Abstract Syntax Tree (AST)

Descreve a representação do código para o CodeQL



```
[libxml2-292-custom source archive] > home > estevam > fuzzing > codeql_stuff > projetos > libxml2 > C pa
13094     if (ctxt->validate) {
13095         ctxt->vctxt.error = ctx->vctxt.error;
13096         ctxt->vctxt.warning = ctx->vctxt.warning;
13097     } else {
13098         ctxt->vctxt.error = NULL;
13099         ctxt->vctxt.warning = NULL;
13100     }
13101     ctxt->vctxt.nodeTab = NULL;
13102     ctxt->vctxt.nodeNr = 0;
13103     ctxt->vctxt.nodeMax = 0;
13104     ctxt->vctxt.node = NULL;
13105     if (ctxt->dict != NULL) xmlDictFree(ctx->dict);
13106     ctxt->dict = ctx->dict;
13107     ctxt->str_xml = xmlDictLookup(ctxt->dict, BAD_CAST "xml", 3);
13108     ctxt->str_xmlns = xmlDictLookup(ctxt->dict, BAD_CAST "xmlns", 5);
13109     ctxt->str_xml_ns = xmlDictLookup(ctxt->dict, XML_XML_NAMESPACE, 36);
13110     ctxt->dictNames = ctx->dictNames;
13111     ctxt->attsDefault = ctx->attsDefault;
13112     ctxt->attsSpecial = ctx->attsSpecial;
13113     ctxt->linenumbers = ctx->linenumbers;
13114
13115     xmlParseContent(ctxt);
13116
```



● Configurando o ambiente

- Instalar o Visual Studio Code (VSCode)
- Instalar a extensão do CodeQL para o VSCode
- Criar/Baixar uma database de Código
 - <https://lgtm.com> é o site principal para baixar databases
- <https://github.com/Es7evam/workshop-codeql>






REVISANDO USE-AFTER- FREE



File: uaf.cpp

```
1  #include <stdlib.h>
2
3  int main(void){
4      int a;
5      int *x;
6      free(x);
7      a = *x;
8  }
```





PRÁTICA!!

● Recursos Úteis

- [Queries CWE C++](#)
- [Github Security Lab](#)
- [Documentação CodeQL](#)
- [Exemplos/Snippets oficiais](#)
- [Guias – CodeQL for C and C++](#)



● Blogposts interessantes

- [Hunting for XSS with CodeQL](#)
- [Sequoia Variant Analysis](#)
- [Vulnerability Hunting With SemmleQL \(MSRC\) Pt.2](#)

- [Learning CodeQL - Repositório com links](#)

