

# Intro to CodeQL

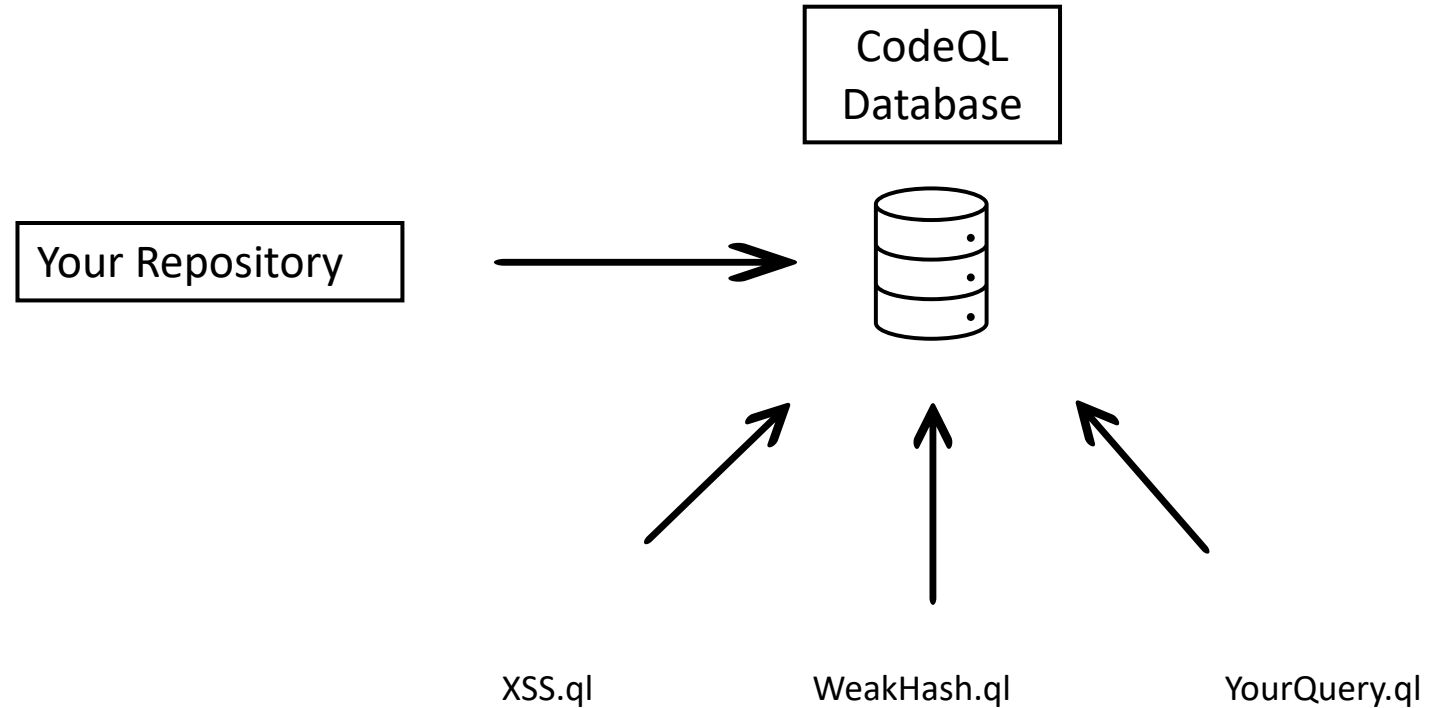
[tinyurl.com/bsides-codeql-workshop](https://tinyurl.com/bsides-codeql-workshop)

# An Example

- [Deserialization risks in use of BinaryFormatter and related types - .NET | Microsoft Learn](#)
- "BinaryFormatter deserializing user input is bad"
  - What is user input? What does that look like in code?
  - How does that user input get to the Deserialize call?
  - What if it's sanitized before the call?

# What is CodeQL

- Similar tools
  - Roslyn
  - SonarCloud
- "Code as Data"
  - Abstract Syntax Tree
  - **Dataflows**

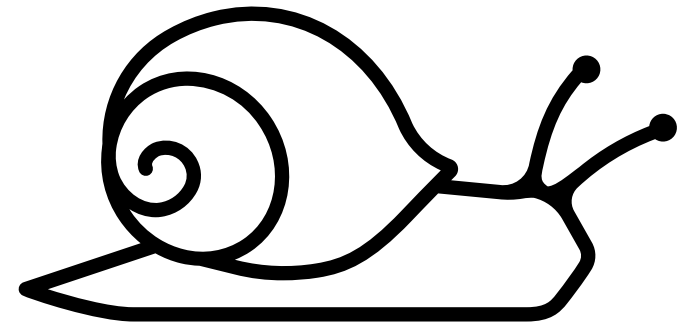


# When to use CodeQL?

- Security: Bug Classes as queries
- At Microsoft:
  - SDL (Security Development Lifecycle) rules
  - Security incident response
  - Other efforts:
    - Security research, eg: "how many services are using x method"
    - [CBOM for Cryptography](#)

# When NOT to use CodeQL

- Slower to run on large databases
- High learning curve
- Other tools: Semgrep, Roslyn



# Getting a CodeQL Database

## Option 1 – Existing Repository

- [Github has >200,000 databases for repositories](#)
- Any repository with github code scanning enabled

## Option 2 – Create your own

- [codeql database create](#)

# Logistics/software

- Download the latest codeql cli: [github/codeql-cli-binaries: Binaries for the CodeQL CLI](#)
- Download VSCode: <https://code.visualstudio.com/>
- Download CodeQL Extension for VSCode: [Installing CodeQL for Visual Studio Code - GitHub Docs](#)
- Clone workshop repository: [chanel-y/BSides-CodeQL101: Starter Files and Docs for CodeQL Workshop for BSides Vancouver 2024 \(github.com\)](#)
- Clone sample project repository: [chanel-y/BSides-Sample-Project \(github.com\)](#)

# Demo + Walkthrough

Creating a CodeQL database and writing our first query



# DataFlow/Path Queries

# Why Dataflow?

```
public RSA CreateWeakKey()  
{  
    RSA key = RSA.Create(1024);  
    return key;  
}
```

# Why Dataflow?

```
public RSA CreateWeakKey1()
{
    int weakKeySize = 1024;
    RSA key = RSA.Create(weakKeySize);
    return key;
}

public RSA CreateKey(int keySize)
{
    RSA key = RSA.Create(weakKeySize);
    return key;
}

public RSA CreateWeakKey2()
{
    int weakKeySize = 1024;
    return CreateKey(weakKeySize);
}
```

# DataFlow Syntax

```
module MyFlowConfiguration implements DataFlow::ConfigSig {  
  predicate isSource(DataFlow::Node source) { ... }  
  predicate isSink(DataFlow::Node sink) { ... }  
}
```

```
module MyFlow = DataFlow::Global<MyFlowConfiguration>;
```

```
from DataFlow::Node source, DataFlow::Node sink  
where MyFlow::flow(source, sink)  
select source, "Dataflow to $@.", sink, sink.toString()
```

# DataFlow vs TaintTracking

## DataFlow

```
source = "mysource";  
intermediary = source  
mySink(intermediary);
```

## TaintTracking

```
source = "mysource";  
intermediary = source + "something";  
mySink(intermediary);
```

# DataFlow vs TaintTracking Syntax

## DataFlow

```
module MyFlowConfiguration implements DataFlow::ConfigSig {  
  predicate isSource(DataFlow::Node source) { ... }  
  predicate isSink(DataFlow::Node sink) { ... }  
}
```

```
module MyFlow = DataFlow::Global<MyFlowConfiguration>;
```

```
from DataFlow::Node source, DataFlow::Node sink  
where MyFlow::flow(source, sink)  
select source, "Dataflow to sink"
```

## TaintTracking

```
module MyFlowConfiguration implements DataFlow::ConfigSig {  
  predicate isSource(DataFlow::Node source) { ... }  
  predicate isSink(DataFlow::Node sink) { ... }  
}
```

```
module MyFlow = TaintTracking::Global<MyFlowConfiguration>;
```

```
from DataFlow::Node source, DataFlow::Node sink  
where MyFlow::flow(source, sink)  
select source, "Dataflow to sink"
```

# Path Queries!

« 1 / 1 » Detects use of binary deserialization on codeql\_querywriting\_demo\_codeql - finished in 1 seconds (1 results) [2/20/2024, 9:52:40 AM] [Open UserInputToBinaryDeserialization.ql](#)

alerts ▼ 1 result ☐ Show results in Problems view

Message

user input flows to BinaryFormatter deserialize [BinaryFormatterDeserialize.cs:29:42](#)

WithBinder.ql RsaKeyCreationWithInsufficientKeySize.ql CodeQL Query Results X ▶ □ ...

« 1 / 1 » Detects use of binary deserialization on codeql\_querywriting\_demo\_codeql - finished in 1 seconds (1 results) [2/19/2024, 4:17:24 PM] [Open UserInputToBinaryDeserialization.ql](#)

alerts ▼ 1 result ☐ Show results in Problems view

Message

user input flows to BinaryFormatter deserialize [BinaryFormatterDeserialize.cs:29:42](#)

Path

- 1 access to property Body : Stream [BinaryFormatterDeserialize.cs:26:32](#)
- 2 call to method ToString : String [BinaryFormatterDeserialize.cs:26:32](#)
- 3 access to local variable userInput : String [BinaryFormatterDeserialize.cs:27:44](#)
- 4 object creation of type FileStream : FileStream [BinaryFormatterDeserialize.cs:27:29](#)
- 5 access to local variable fs [BinaryFormatterDeserialize.cs:29:42](#)

# Path Query Syntax

## "Normal" DataFlow

@kind problem

...

```
module MyFlowConfiguration implements DataFlow::ConfigSig {  
  predicate isSource(DataFlow::Node source) { ... }  
  predicate isSink(DataFlow::Node sink) { ... }  
}
```

```
module MyFlow = DataFlow::Global<MyFlowConfiguration>;
```

```
from DataFlow::Node source, DataFlow::Node sink  
where MyFlow::flow(source, sink)  
select source, "Dataflow to sink."
```

## Path Query

@kind path-problem

...

```
module MyFlowConfiguration implements DataFlow::ConfigSig {  
  predicate isSource(DataFlow::Node source) { ... }  
  predicate isSink(DataFlow::Node sink) { ... }  
}
```

```
module MyFlow = DataFlow::Global<MyFlowConfiguration>;
```

```
Import MyFlow::PathGraph
```

```
from MyFlow::Node source, MyFlow::Node sink  
where MyFlow::flow(source, sink)  
select sink.getNode(), source, sink, "Dataflow to sink."
```



# Writing a DataFlow Query

- Define a source
- Define a sink
- Define any additional steps and barriers
- Decide whether dataflow vs tainttracking and problem vs path-problem

# DataFlow/Path Query Exercises

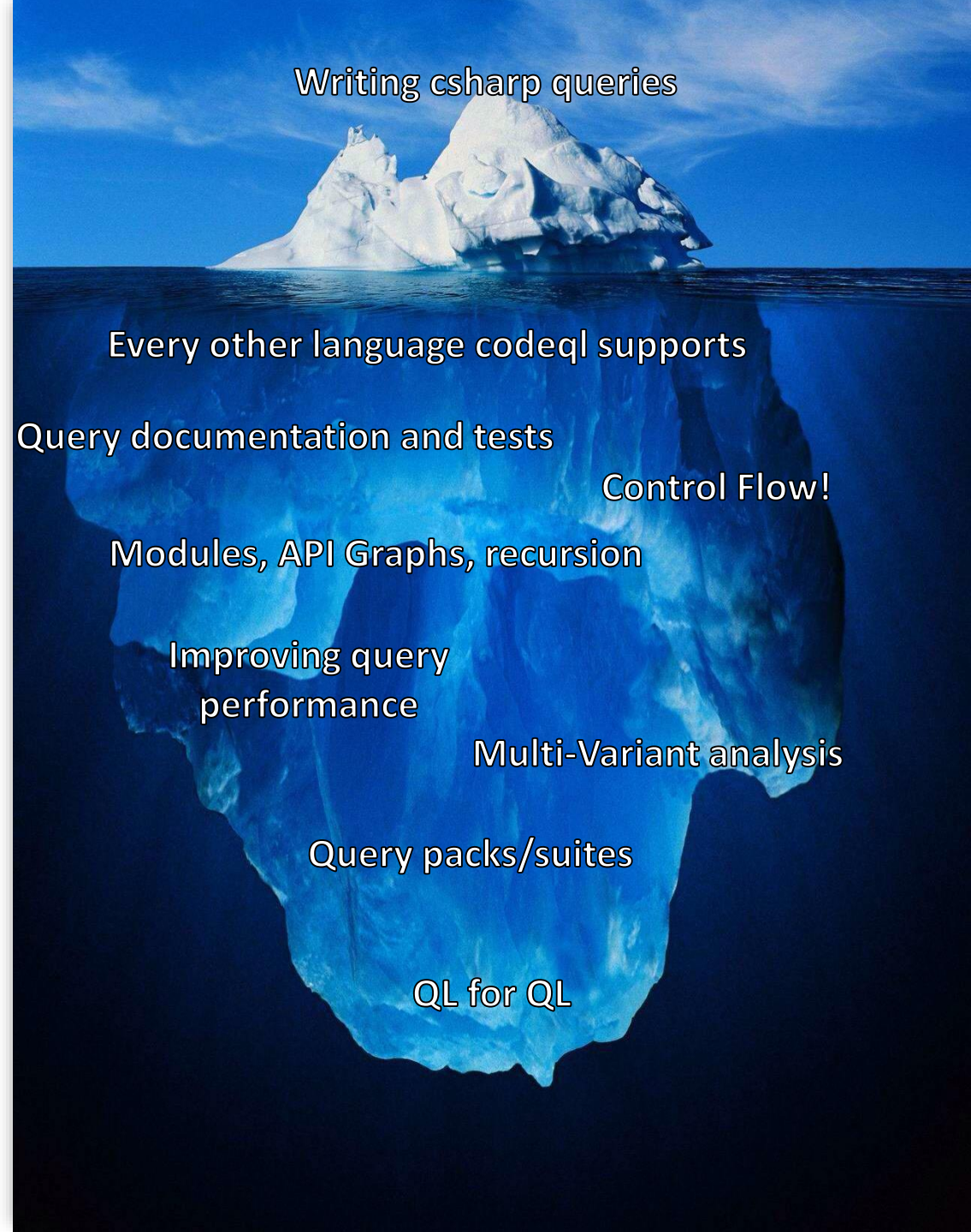
You've been Intro'd to  
CodeQL, now what?

# Getting Familiar with QL

- "I don't know what this is in CodeQL"
  - CTRL-F through github's existing queries
  - Use the AST to see where to start
- "I don't know how to use this class/predicate in CodeQL"
  - CTRL-F through github's existing queries
  - Standard Library: [CodeQL standard libraries \(github.com\)](https://github.com/github/codeql/blob/master/standard-libraries/CodeQLStandardLibraries.md)
- I'm truly, fully stuck
  - Public github security slack: [codeql-writing \(Channel\) - GitHub Security Lab – Slack](https://github.com/github/codeql-writers)
  - CodeQL Repository Discussion: [github/codeql · Discussions · GitHub](https://github.com/github/codeql/discussions)
  - Email me!

# Additional CodeQL Topics to Explore

- Query Performance: [Troubleshooting query performance — CodeQL \(github.com\)](#)
- Unit tests for CodeQL: [Testing custom queries - GitHub Docs](#)
- QLPacks: [Creating and working with CodeQL packs - GitHub Docs](#)
- Multi-repository variant analysis: [Running CodeQL queries at scale with multi-repository variant analysis - GitHub Docs](#)



Writing csharp queries

Every other language codeql supports

Query documentation and tests

Control Flow!

Modules, API Graphs, recursion

Improving query  
performance

Multi-Variant analysis

Query packs/suites

QL for QL

# Additional Learning Resources

- Github CodeQL docs: [CodeQL overview — CodeQL \(github.com\)](#)
- [QL tutorials — CodeQL \(github.com\)](#)
- Github CodeQL CTFs: [Capture the flag | GitHub Security Lab](#)
- A Practical Introduction to CodeQL by jorgectf: [Practical Introduction to CodeQL :: jorgectf — blog](#)
- CodeQL Zero to Hero (part 1 of 3): [CodeQL zero to hero part 1: The fundamentals of static analysis for vulnerability research - The GitHub Blog](#)

# Thanks!

Email: [chanelyoung@microsoft.com](mailto:chanelyoung@microsoft.com)

LinkedIn: [chanelyoung99](#)