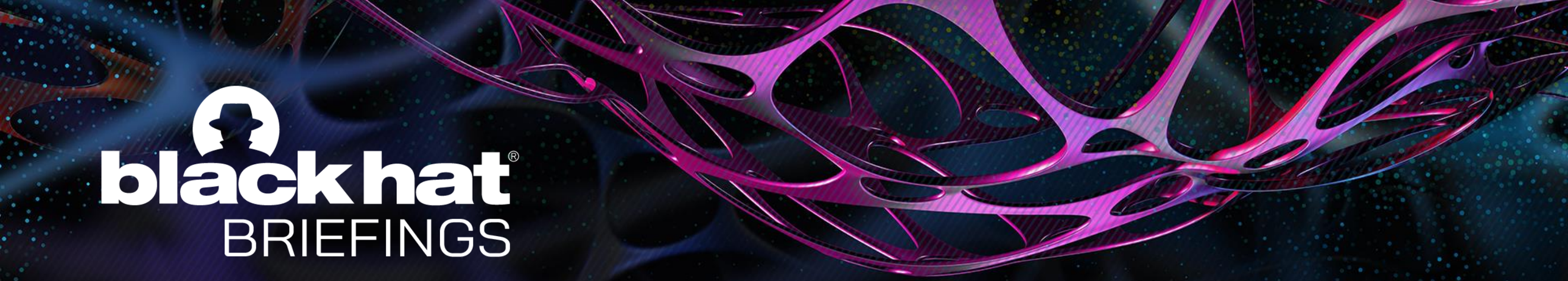


More Flows, More Bugs: Empowering SAST with LLMs and Customized DFA

Yuan Luo & Zhaojun Chen & Yi Sun & Rhettxie

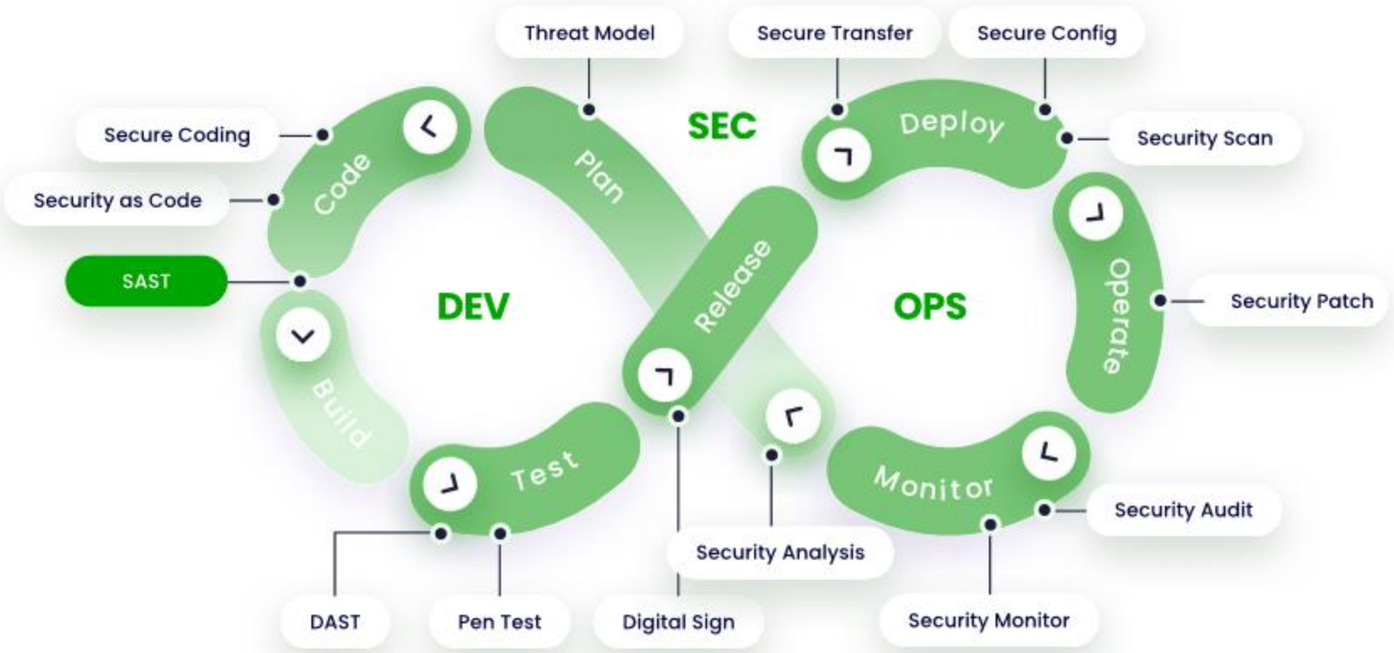
@Tencent Security YunDing Lab

Outline

- Introduction to SAST
- How to use LLM to recognize Sources & Sinks
- DFA (Data Flow Analysis) Enhancement
- Results

What is SAST?

- Static Application Security Testing (SAST)
  - Analyze source code without execution
- Scans code for bugs
  - Such as SQL injection or XSS, acting like an X-ray
- Essential part of DevSecOps
  - Integrated into CI/CD pipelines
- Popular tools
  - CodeQL/Fortify/SonarQube/Checkmarx/...



DevSecOps Lifecycle Diagram

Source: https://www.mend.io/blog/sast-static-application-security-testing/

What is CodeQL?

Background

- Founded in 2006, GitHub acquired CodeQL in 2019
- Queries and libraries are open-source
- The core CodeQL engine is proprietary

Workflow

- Preparing the code
- Creating a CodeQL database
- Running CodeQL queries against the database
- Interpreting the query results

Source code

JavaConverter.java

```
public static Object deserialize (InputStream is)
    throws IOException {
  ObjectInputStream ois = new ObjectInputStream(is);
  return ois.readObject();
}
```

Build database

```
$ # Clone the project
$ git clone https://github.com/m-y-mo/struts_9805


$ # Create a CodeQL database
$ codeql database create ./struts_db -s ./struts_9805 \
    -j 0 -l java --command "mvn -B -DskipTests \
    -DskipAssembly"
```

QL query

UnsafeDeserialization.ql

```
from PathNode source, PathNode sink
where flowPath(source, sink)
select sink.getNode().(UnsafeDeserializationSink)
    .getMethodAccess(),
  source, sink, "Unsafe deserialization of $@.",
  source.getNode(), "user input"
```

Query Results

QL Query Results

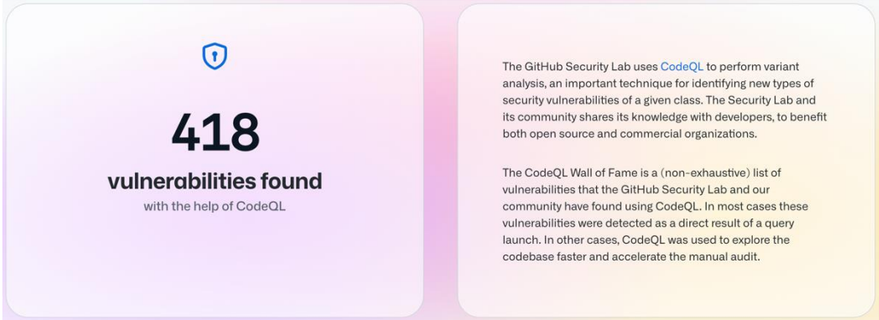
alerts

- Unsafe deserialization of user input.
- Unsafe deserialization of user input.
  - Path
    1. getContent(...) : InputStream
    2. getContentAsStream(...) : InputStream
    3. toBufferedInputStream(...) : InputStream
    4. getInputStream(...) : InputStream
    5. is : InputStream
    6. ois
  - Path
- Unsafe deserialization of user input.

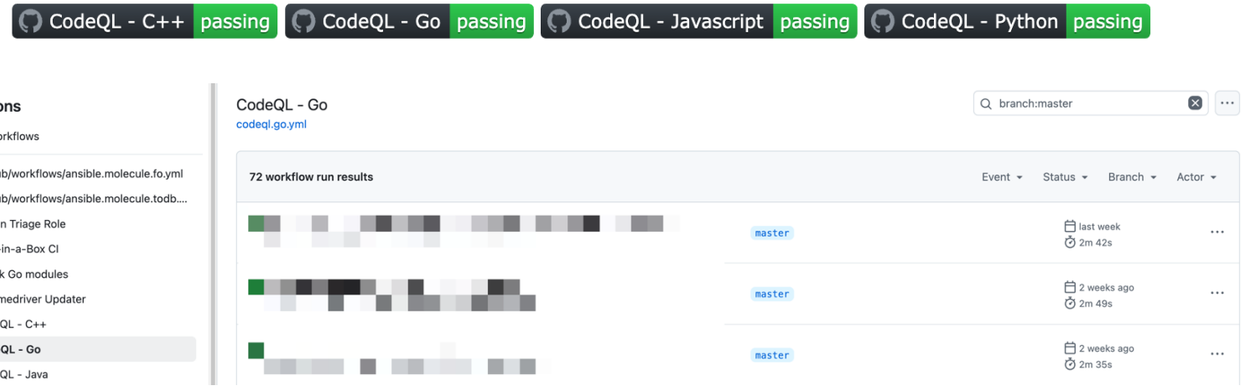
Source: https://codeql.github.com/

No bugs anymore?

Many vulnerabilities found…



Periodically perform code scanning on the default branch and pull requests



However, SAST tools may overlook ...

Our attempts to detect recent critical RCE vulnerabilities using CodeQL revealed two main causes for **false negatives** :

1. Incomplete **source and sink** coverage in built-in propagation rules.
2. Disruptions in data flow due to insufficient support for certain **language features**.

CVE	Description	Root causes for missed detection
CVE-2024-47552 CVE-2024-56180 CNVD-2023-45001	Apache Seata, Apache EventMesh, Alibaba Nacos JRaft vulnerability; other affected applications include Apache Ignite and Apache HugeGraph.	Missing source rule
CVE-2024-37084	Spring Cloud Data Flow Remote Code Execution Vulnerability	Missing summary rule
CVE-2024-22263	Spring Cloud Data Flow Arbitrary File Write Vulnerability	Missing summary rule
CVE-2023-52251	Kafka UI Background Messages Groovy Code Execution Vulnerability	Code pre-generation
CVE-2023-34050	Spring AMQP Deserialization Vulnerability	Asynchronous Method Reference
CVE-2023-37582	Apache RocketMQ NameServer Remote Code Execution Vulnerability	Reflection Cross-Thread
CVE-2023-33246	Apache RocketMQ Remote Code Execution Vulnerability	Reflection Cross-Thread
CVE-2023-46604	Apache ActiveMQ Remote Code Execution Vulnerability	Cross-Thread Missing sink rule
CVE-2023-25194	Apache Kafka JAAS JNDI Injection Vulnerability	Missing source rule Missing summary rule Missing sink rule
…	…	…


How to use LLM to recognize Sources & Sinks

Current Methods Depend on Human Effort

Manual Definition

Community Contributions

codeql / go / ql / lib / ext / **github.com.beego.beego.client.orm.model.yml**

Code | Blame | 60 lines (59 loc) · 4.86 KB ·

```
9     - addsTo:
10        pack: codeql/go-all
11        extensible: sourceModel
12      data:
13        - ["group:beego-orm", "DB", True, "Query", "", "", "ReturnValue[0]", "database", "manual"]
```

Labor-intensive. Any automated methods?

Where to find Sources and Sinks?

Developers implement functionality using third-party frameworks (using APIs).

```go
// HTTP, REST Client
client := resty.New()
defer client.Close()

res, err := client.R().
    EnableTrace().
    Get("https://httpbin.org/get")
fmt.Println(err, res)
fmt.Println(res.Request.TraceInfo())
```

The API implementation is included in the framework's open-source code

```go
// Get method does GET HTTP request. It's defined in section 9.3.1 of [RFC 9110].
//
// [RFC 9110]: https://datatracker.ietf.org/doc/html/rfc9110.html#section-9.3.1
func (r *Request) Get(url string) (*Response, error) {
        return r.Execute(MethodGet, url)
}
```

Scan open-source frameworks and detect Sources and Sinks

How LLMs can help?

- Discover Agent

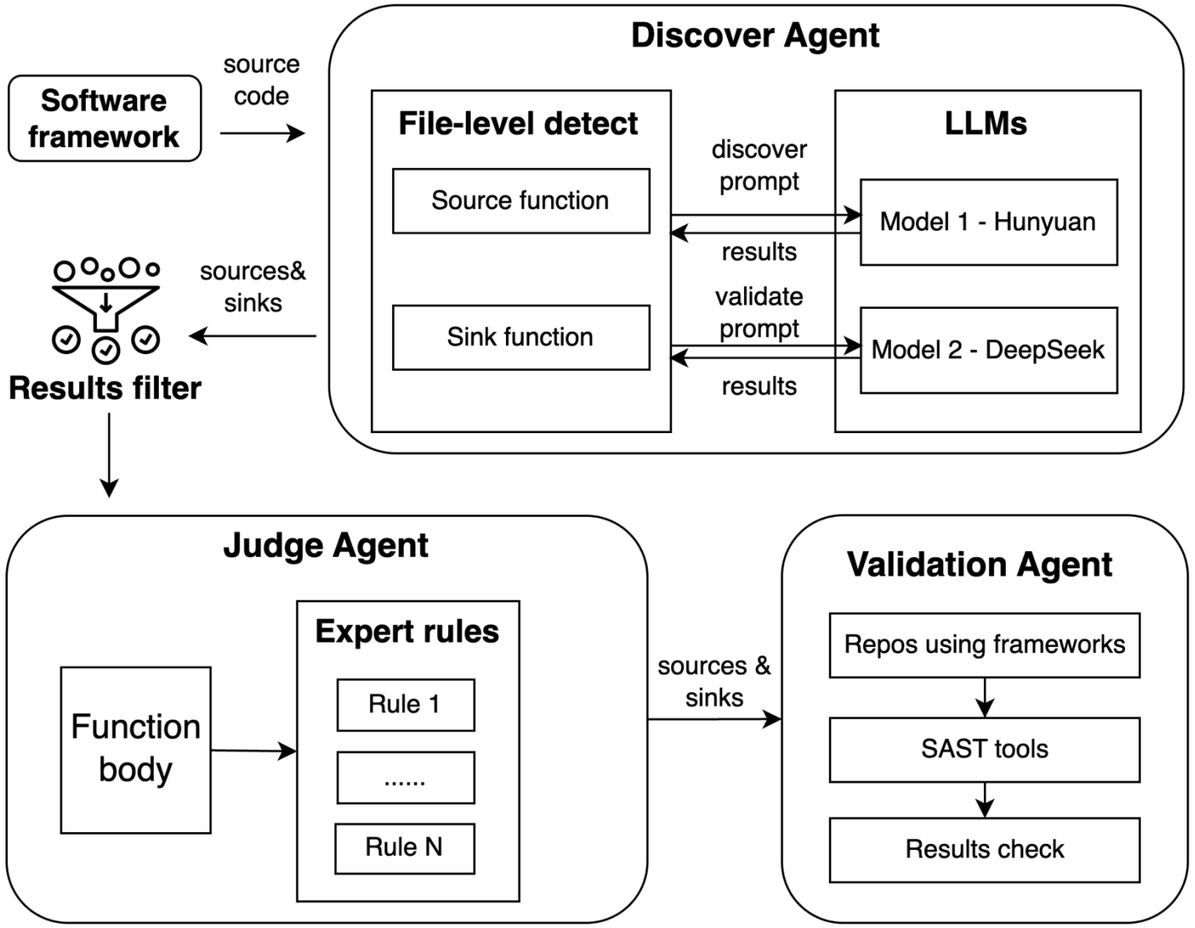
  - Discover possible functions from frameworks

- Judge Agent

  - Use expert rules to verify

- Validation Agent

  - Verify sources/sinks are used in real-world repos



The workflow of our method

Discover Agent – How to find source/sink functions?

File-Level Coarse-Grained Filtering

Source code file

resty / request.go

Code | Blame | 1787 lines (1626 loc) · 54.3 KB

```
1289    // HTTP verb method starts here
1290    //___________________________________________________________________
1291
1292    // Get method does GET HTTP request. It's defined in section 9.3.1 of [RFC 9110].
1293    //
1294    // [RFC 9110]: https://datatracker.ietf.org/doc/html/rfc9110.html#section-9.3.1
1295    func (r *Request) Get(url string) (*Response, error) {
1296            return r.Execute(MethodGet, url)
1297    }
1298
1299    // Head method does HEAD HTTP request. It's defined in section 9.3.2 of [RFC 9110].
1300    //
1301    // [RFC 9110]: https://datatracker.ietf.org/doc/html/rfc9110.html#section-9.3.2
1302    func (r *Request) Head(url string) (*Response, error) {
1303            return r.Execute(MethodHead, url)
1304    }
1305
1306    // Post method does POST HTTP request. It's defined in section 9.3.3 of [RFC 9110].
1307    //
1308    // [RFC 9110]: https://datatracker.ietf.org/doc/html/rfc9110.html#section-9.3.3
1309    func (r *Request) Post(url string) (*Response, error) {
1310            return r.Execute(MethodPost, url)
1311    }
```

LLM prompt. The prompts vary depending on different LLMs.

You are a cybersecurity expert. You are given a source code file written in the Go programming language. Identify functions in the source code that send HTTP requests. Such functions could be labeled as potential HTTP sinks for taint analysis. If no such functions exist, answer "None" directly. If such functions exist, return the function name, concrete code context, and confidence score as a JSON in the following format:
{ "method": <method name>, "code": <code context that send HTTP requests>, "score": <confidence score>}.
Do not provide explanations or comments; just output the JSON. If there are multiple functions, list all functions.
[start of Go source code]
{source code}
[end of Go source code]

Tencent HunYuan 腾讯混元

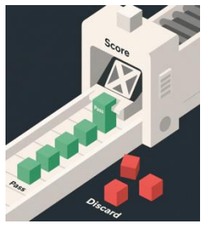
LLM results

{ "method": "Get", "code": "func (r *Request) Get(url string) (*Response, error) { return r.Execute(MethodGet, url) }", "score": 0.9 }

{ "method": "AddRetryConditions", "code": "func (r *Request) AddRetryConditions(conditions ...RetryConditionFunc) *Request { r.retryConditions = append(r.retryConditions, conditions...) return r}", "score": 0.1 }

Score filter

You can change the threshold to get more or fewer functions.



Functions list

{ "method": "Get"}
{ "method": "Post"}

Discover Agent – How to remove false positives?

■ Function-Level Filtering

Function name and body

```
func (r *Request) Get(url string) (*Response, error) {
        return r.Execute(MethodGet, url)
}
```

LLM prompt for SSRF Sink detection

You are a cybersecurity expert. Determine if the following function is a sink for SSRF vulnerabilities by checking if it sends HTTP requests. If it does, answer "YES". If not, answer "NO". Do not explain or comment.

Source/Sink functions for each framework

```
9.0K Sep  6  2024 apisix.csv
 89K Aug  5  2024 argo-cd.csv
 27K Oct 21  2024 beego.csv
 54K Aug  9  2024 caddy.csv
6.0K Aug 21  2024 chi.csv
 13K Aug  6  2024 cloudquery.csv
5.6K Sep  6  2024 droplet.csv
 51K Aug  6  2024 dubbo.csv
 18K Oct 18  2024 echo.csv
 30K Aug  6  2024 ekuiper.csv
6.7K Aug 12  2024 fasthttp.csv
 34K Aug 14  2024 gf.csv
 13K Aug 22  2024 gin.csv
368K Aug  3  2024 gitea.csv
2.6K Aug 30  2024 go-grpc-middleware.csv
 54K Aug  2  2024 gogs.csv
 11K Aug 20  2024 go-restful.csv
  24 Aug  2  2024 gorm.csv
 21K Aug 13  2024 go-zero.csv
```

deepseek

Discard, if it does not meet standards

Judge Agent

We need to combine expert experience

- The function should be publicly accessible. (Source/Sink)
- The function should not read authentication credentials and key information. (Source/Sink)
- The function should have return values that propagate tainted data. (Source)
- The return value of the function should not be of the Bool type. (Source)
- The function should accept inputs from untrusted sources. For example, user input (web forms, cookies, URL parameters), external files, network data, environment variables, etc. (Source)
- The function should create or execute a SQL query. (Sink-SQL)
- The function should send HTTP requests. (Sink-SSRF)
- …

+

Function body

```
func (r *Request) Get(url string) (*Response, error) {
        return r.Execute(MethodGet, url)
}
```

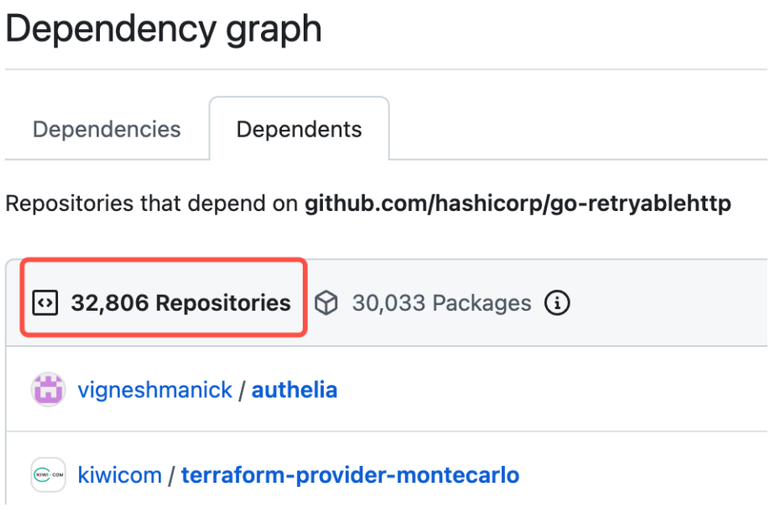
⬇

Conduct expert rule checks on functions using LLMs

Validation Agent - How to ensure functions are used in real-world repos?

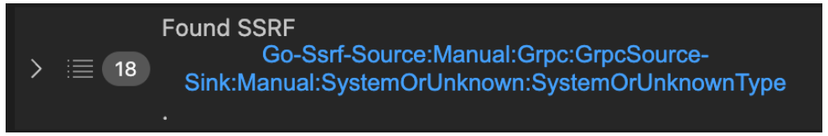
■ Run queries on real-world repos

Retrieve the framework's dependent repositories



Get the source code of these repositories

- Incorporate identified sources and sinks into SAST tools (e.g., CodeQL)
- Verify the presence of these sources and sinks



Finally, we get new sources and sinks!

DFA (Data Flow Analysis) Enhancement

CodeQL DFA Implementation Mechanism

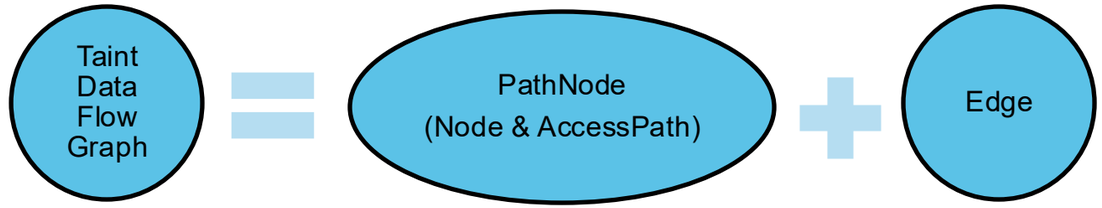
- What is the execution principle of DFA (Data flow analysis) queries?

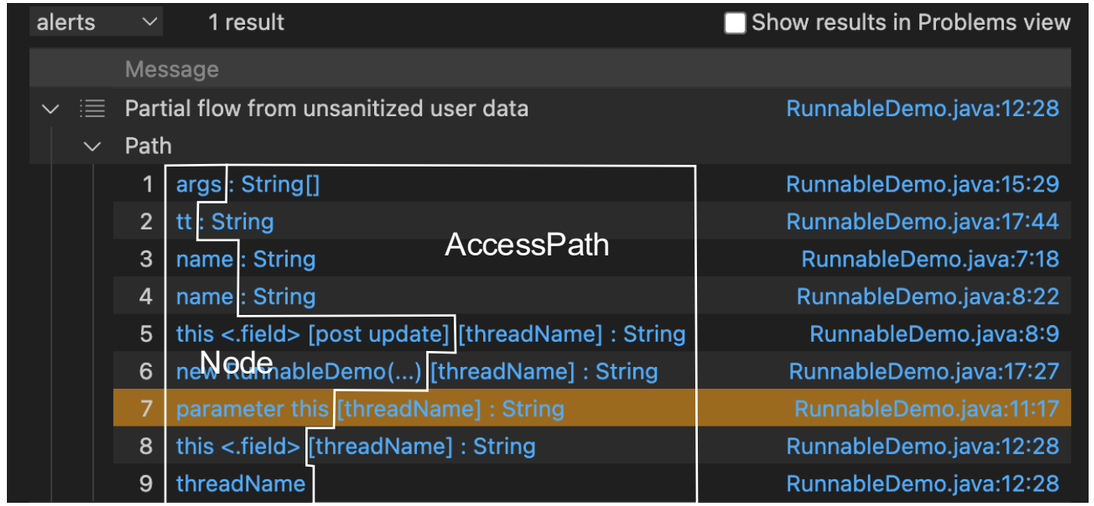
Data Flow Example

```
1   /**
2   * @kind path-problem
3   */
4   import java
5   import semmle.code.java.dataflow.TaintTracking
6
7
8   module MyTaintTrackingConfig implements DataFlow::ConfigSig {
9
        Quick Evaluation: isSource
10      predicate isSource(DataFlow::Node source) {
11          exists(Method m | m.hasName("main") and
12              m.getAParameter() = source.asParameter()
13          )
14      }
15
        Quick Evaluation: isSink
16      predicate isSink(DataFlow::Node sink) {
17          exists(MethodCall ma | sink.asExpr() = ma.getAnArgument() and
18              ma.getCallee().getDeclaringType().hasQualifiedName("java.io", "PrintStream")
19          )
20      }
21  }
22
23  module MyTaintTrackingFlow = TaintTracking::Global<MyTaintTrackingConfig>;
24  import MyTaintTrackingFlow::PathGraph
25
26  from MyTaintTrackingFlow::PathNode source, MyTaintTrackingFlow::PathNode sink
27  where MyTaintTrackingFlow::flowPath(source, sink)
28  select sink, source, sink, "$@", source, source.toString()
```

?

DFA Result

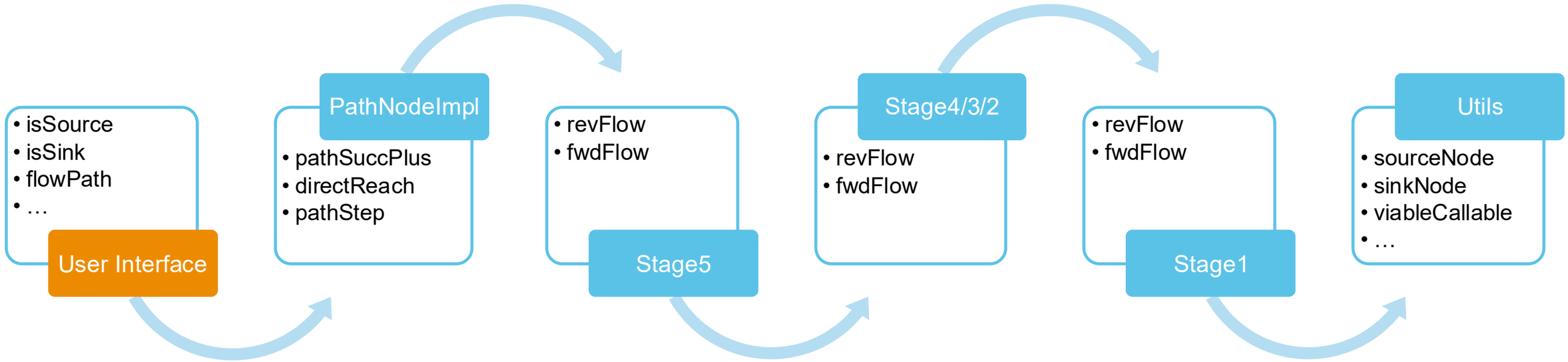




In the following examples, note that:

1. CodeQL version is 2.17.2.
2. We define that the arguments to the main method serve as the ***source***, and the arguments to the System.out.println method serve as the ***sink***.

CodeQL DFA Implementation Mechanism



- The user implements taint analysis by inheriting the ***DataFlow::ConfigSig*** interface and then calling the ***flowPath*** interface.
- The ***fwdFlow*** begins at the source to identify potential data flow propagation points, while ***revFlow*** starts from the sink to trace back the origins of propagation points. They target the ***Node***, and together they form the complete ***source-to-sink*** path.
- From Stage 5 to Stage 1, the logic is similar, differing only in context, with the ultimate goal of obtaining the ***AccessPath***.

CodeQL DFA Implementation Mechanism

How to calculate the Node?

Step	Demo	Description
Source		Taint analysis source node.
Local Flow	mid = "taint"; node = mid;	Intra-procedural analysis, if a mid-node exists and can propagate to a node within the program, then that node is also considered a point in the flow.
Jump		Custom extension methods provided for users can be implemented through ***AdditionalValueStep***.
Store	node.filed = mid; node[x] = mid;	Assign values to fields, arrays, collections, maps, etc.
Load	node = mid.filed; node = mid[x];	Retrieve values from fields, arrays, collections, maps, and so on.

Call In	public void m(param){
	this.f=param;
	}
	o.m(arg);
		Inter-procedural analysis, propagation during method invocation:
		1. Propagation from actual argument ***arg*** to formal parameter ***param***.
		2. Propagation from the method call qualifier (***o***) to the ***this*** parameter of the method.

Call Out	public Object m(param){
	ret=source;
	return ret;
	};
	r=o.m(arg);
		Specifically, the return value ret is not propagated from parameters, but from sources such as source points. In this case:
		1. ***ret*** propagates to the expression ***o.m(arg)***.

Call Through	public Object m(param, o){
	ret=param;
	o.field=param;
	return ret;
	};
	r=o.m(arg, obj);
		Specifically, the return value ret is propagated from the parameters. In this example:
		1. ***ret*** propagates to the expression ***o.m(arg, obj).***
		2. ***o[post update][field]*** propagates to ***obj[post update][field].***

```
private predicate fwdFlow(NodeEx node, Cc cc) {
  // source
  sourceNode(node, _) and
  if hasSourceCallCtx() then cc = true else cc = false
  or
  // local flow
  exists(NodeEx mid | fwdFlow(mid, cc) |
    localFlowStepEx(mid, node, _) or
    additionalLocalFlowStep(mid, node, _) or
    additionalLocalStateStep(mid, _, node, _)
  )
  or
  // jump
  exists(NodeEx mid | fwdFlow(mid, _) and cc = false |
    jumpStepEx(mid, node) or
    additionalJumpStep(mid, node, _) or
    additionalJumpStateStep(mid, _, node, _)
  )
  or
  // store
  exists(NodeEx mid |
    useFieldFlow() and
    fwdFlow(mid, cc) and
    storeEx(mid, _, node, _, _)
  )
  or
  // read
  exists(ContentSet c |
    fwdFlowReadSet(c, node, cc) and
    fwdFlowConsCandSet(c, _)
  )
  or
  // flow into a callable
  fwdFlowIn(_, _, _, node) and
  cc = true
  or
  // flow out of a callable
  fwdFlowOut(_, node, false) and
  cc = false
  or
  // flow through a callable
  exists(DataFlowCall call |
    fwdFlowOutFromArg(call, node) and
    fwdFlowIsEntered(call, cc)
  )
}
```

CodeQL DFA Implementation Mechanism

What is AccessPath?

Content

- Content is a description of the way data may be stored inside an object.
- Different object type has different content, as follows:

Object Type	Demo Code	Content
Field	Person p = new Person(); p.name = taint;	name
Array	String[] ts = new String[1]; ts[0] = taint;	[]
Collection	ArrayList<String> ts = new ArrayList<>(); ts.add(taint);	<element>
MapKey	HashMap<String, int> ts = new HashMap<String, int>(); ts.put(taint,1);	<map.key>
MapValue	HashMap<int, String> ts = new HashMap<int, String>(); ts.put(1, taint);	<map.value>

- Assuming that ***taint*** is a taint ***Node***, type String.

AccessPath

- AccessPath records the ***content*** propagation relationships for the four types of nodes (Field, Array, Collection, and Map) in Store/Load propagation. It consists of ***a content list*** and ***a type***.

Object Type	Demo Code	AccessPath
Field	Person p = new Person(); p.name = taint;	[name] : String
Array	String[] ts = new String[1]; ts[0] = taint;	[[]] : String
Collection	ArrayList<String> ts = new ArrayList<>(); ts.add(taint);	[<element>]: String
MapKey	HashMap<String, int> ts = new HashMap<String, int>(); ts.put(taint,1);	[<map.key>] : String
MapValue	HashMap<int, String> ts = new HashMap<int, String>(); ts.put(1, taint);	[<map.value>] : String

- The red mark indicates the corresponding ***Node***.

CodeQL DFA Implementation Mechanism

How to calculate the AccessPath?

Stage	AP	Description	Demo
Stage1	Unit	No explicit access paths; tracks candidate content operations (storeStepCand, readStepCand) that will form access paths.	-
Stage2	Boolean	Boolean access paths (true for non-empty, false for empty), introducing coarse pruning based on whether any dereference exists.	true
Stage3	ApproxAccessPath Front	Single-content approximation (ApproxAccessPathFront), tracking the first content operation to refine pruning.	[] <element> <map.key> <map.value> approximated field a to approximated field z
Stage4	AccessPathFront	Precise single-content tracking (AccessPathFront), with content clearing to eliminate invalid paths.	[] <element> <map.key> <map.value> ps1 ps2 ps3 ps4
Stage5	AccessPathApprox	Type-safe access paths (AccessPathApprox), tracking up to two contents or an approximated tail, with type checking and cost-based pruning.	[ps1, [], …(3)] [ps2, <element>, …(3)] [ps3, <map.value>, …(3)] [ps4, <map.key>, …(3)]
PathNodeImpl	AccessPath	Precise access paths (AccessPath), representing full sequences of contents (or approximations for expensive cases), integrated into the final path graph for accurate data flow paths.	[ps1, [], name] : String [ps2, <element>, name] : String [ps3, <map.value>, name] : String [ps4, <map.key>, name] : String

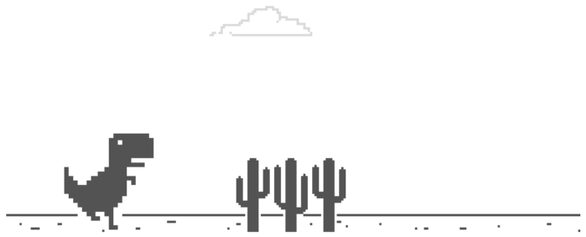
```java
public class Person {
    private String name;
    …
}

public class Demo {
    private Person[] ps1 = new Person[1];
    private ArrayList<Person> ps2 =new ArrayList<>();
    private HashMap<Integer, Person> ps3 = new HashMap<Integer, Person>();
    private HashMap<Person, Integer> ps4 = new HashMap<Person, Integer>();
    …

    public static void main(String[] args) throws Exception {
        Demo demo = new Demo();
    }
}
```

Language-specific Limitations

- Java as an example

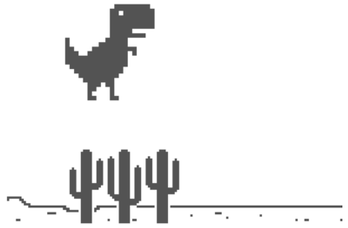


Cross Thread

Reflection

Value Passing

…



Java Cross-Thread

In the following examples, we agree that:
1. The parameters of the static main method serve as ***source***.
2. The parameters of the System.out.println method serve as ***sink***.

Runnable Instance Constructor Call

```
1   import java.lang.Runnable;
2
3   public class RunnableDemo implements Runnable {
4       private String threadName;
5
6       RunnableDemo(String name){threadName = name;}
7
8       public void run(){
9           System.out.println(threadName);
10      }
11
12      public static void main(String[] args) throws Exception {
13          String tt = args[0];
14          RunnableDemo T1 = new RunnableDemo(tt);
15          Thread t = new Thread(T1);
16          t.start();
17      }
18  }
```

sink

jump

source

#Quick_evaluation_of_predicate_test_fwdFlow0

#	node	cc	step	pre
1	args	false	source	args
2	args	false	local flow	args
3	...[...]	false	local flow	args
4	tt	false	local flow	...[...]
5	name	true	flow into	tt
6	name	true	local flow	name
7	this <.field> [post update]	true	store	name
8	new RunnableDemo(...)	false	flow through	this <.field> [post update]
9	parameter this	false	jump	new RunnableDemo(...)
10	this <.field>	false	local flow	parameter this
11	threadName	false	read	this <.field>

- When the taint node (***tt***) is positioned in the constructor call of a Runnable subclass.
- Jump from the ***constructor call expression*** of a Runnable subclass to the instance parameter (***this***) of ***run*** method.

Java Cross-Thread

- How to jump?

Jump API: **AdditionalValueStep**

```
/** Value step from the constructor call of a `Runnable` to the instance parameter (this) of `run`.
*
* Class MyRunnable implements Runnable{
* public void run(){}
* }
* MyRunnable m = new MyRunnable(xxx)
* additional step:
* from: new MyRunnable(xxx)
* to: MyRunnable#run#this
*/
private class RunnableConstructorCallToRunStep extends AdditionalValueStep {
        override predicate step(Node pred, Node succ) {
                exists(ConstructorCall cc, Method m |
                        m.getDeclaringType() = cc.getConstructedType().getSourceDeclaration()
                        and cc.getConstructedType().getAnAncestor().hasQualifiedName("java.lang", "Runnable")
                        and m.hasName("run")
                |
                        pred.asExpr() = cc
                        and succ.(InstanceParameterNode).getEnclosingCallable() = m
                )
        }
}
```

Java Cross-Thread

Thread start

```
1   import java.lang.Thread;
2
3   public class ThreadDemo extends Thread {
4       private String threadName;
5
6       ThreadDemo() {}
7
8       public void setThreadName(String name){threadName=name;}
9
10      public void run() {
11          System.out.println(threadName);
12      }
13                  jump
14      public static void main(String[] args) throws Exception {
15          String tt = args[0];
16          ThreadDemo T1 = new ThreadDemo();
17          T1.setThreadName(tt);
18          T1.start();
19      }
20  }
```

#Quick_evaluation_of_predicate_test_fwdFlow0

#	node	cc	step	pre
1	args	false	source	args
2	args	false	local flow	args
3	...[...]	false	local flow	args
4	tt	false	local flow	...[...]
5	name	true	flow into	tt
6	name	true	local flow	name
7	this <.field> [post update]	true	store	name
8	T1 [post update]	false	flow through	this <.field> [post update]
9	T1	false	local flow	T1 [post update]
10	parameter this	false	jump	T1
11	this <.field>	false	local flow	parameter this
12	threadName	false	read	this <.field>

- When the taint node (***tt***) is positioned after the constructor call of a Thread subclass and before the start method.
- Jump from the Thread instance (***T1***) initiating the start() call to the ***this*** reference within the run method's execution context.

Java Cross-Thread

Field Update

```
1  import java.lang.Thread;
2
3  public class ThreadDemoWhile extends Thread {
4      private String threadName;
5
6      ThreadDemoWhile() {}
7
8      public void setThreadName(String name){this.threadName=name;}
9
10     public void run(){
11         while(true){
12             try{
13                 Thread.sleep(1000);
14                 System.out.println(this.threadName);
15             }catch (Exception e){}
16         }
17     }
18
19     public static void main(String[] args) throws Exception {
20         String tt = args[0];
21         ThreadDemoWhile T1 = new ThreadDemoWhile();
22         T1.start();
23         T1.setThreadName(tt);
24     }
25 }
```

jump

#	node	cc	step	pre
1	args	false	source	args
2	args	false	local flow	args
3	...[...]	false	local flow	args
4	tt	false	local flow	...[...]
5	name	true	flow into	tt
6	name	true	local flow	name
7	this [post update]	true	store	name
8	parameter this	false	jump	this [post update]
9	this	false	local flow	parameter this
10	this.threadName	false	read	this

- When the taint node(***tt***) is positioned after the constructor call and the start method.
- Jump from the Runnable instance PostUpdateNode parameter (***this[post update]***) of a store operation to the instance parameter (***this***) of the run method.

Java Reflection

Invoke Call

```
1   import java.lang.reflect.Method;
2
3   public class InvokeDemo {
4       private String name;
5
6       public void setName(String name){ this.name = name; }
7
8       public String getName(){ return name; }
9
10      private static void util(Object obj, String method, String s) throws Exception{
11          Class c = obj.getClass();
12          Method m = c.getMethod(method);
13          m.invoke(obj, s);
14      }
15
16      public static void main(String[] args) throws Exception {
17          String arg = args[0];
18          InvokeDemo tt = new InvokeDemo();
19
20          util(tt, "setName", arg);
21          String name = tt.getName();
22          System.out.println(name);
23      }
24  }
```

Call Through

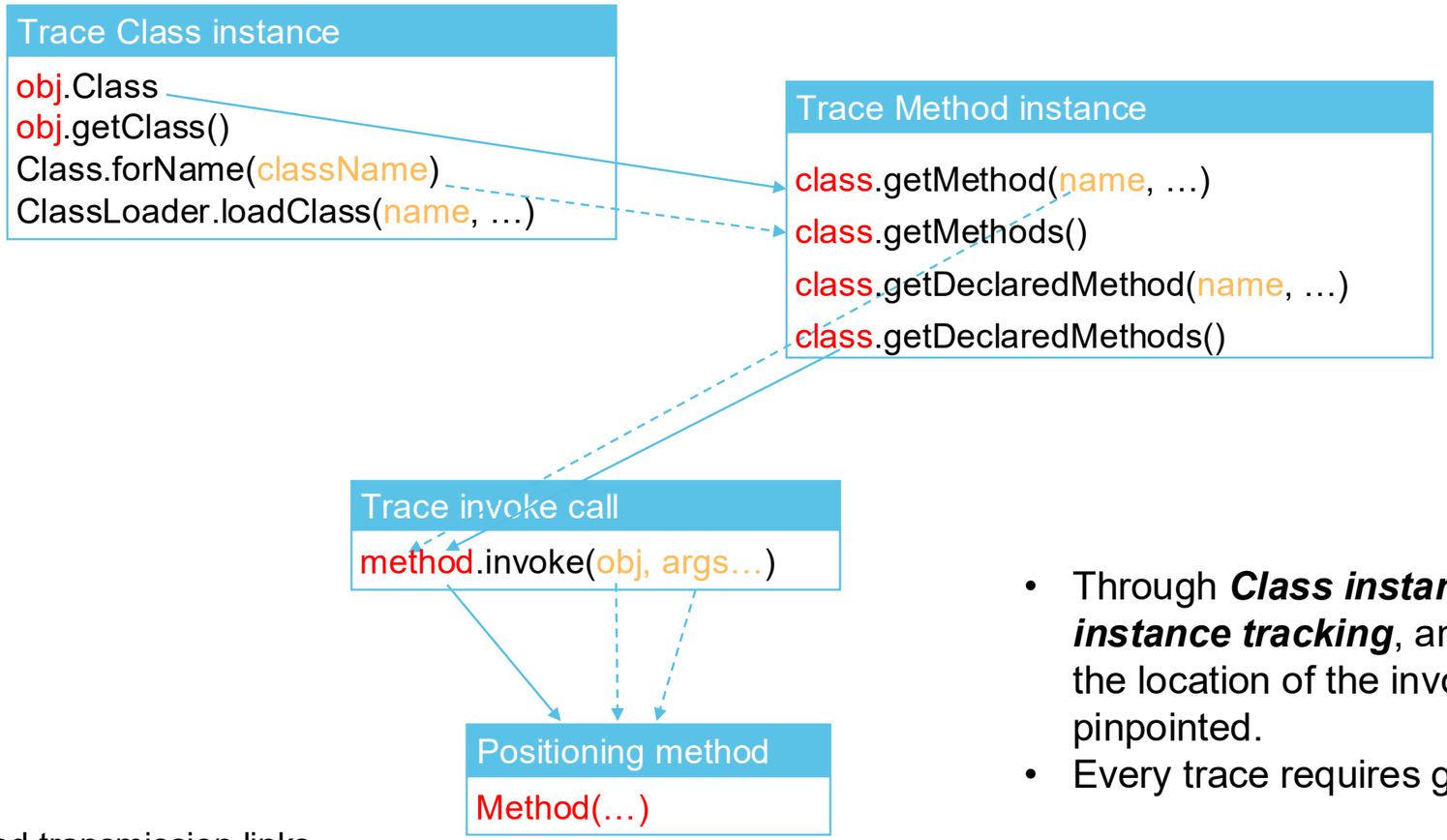
Call In

To enable Java reflection analysis in CodeQL, including operations like method invocation, we face the following difficulties:

1. How to locate method invoked.
2. The previous propagation flow rules for ***Call In/Call Through*** in Data Flow are no longer applicable and require patching.

Java Reflection

- **Challenge1: How to locate method invoked ?**



- Through ***Class instance tracking***, ***Method instance tracking***, and ***Invoke call tracking***, the location of the invoked method can be pinpointed.
- Every trace requires global DataFlow.

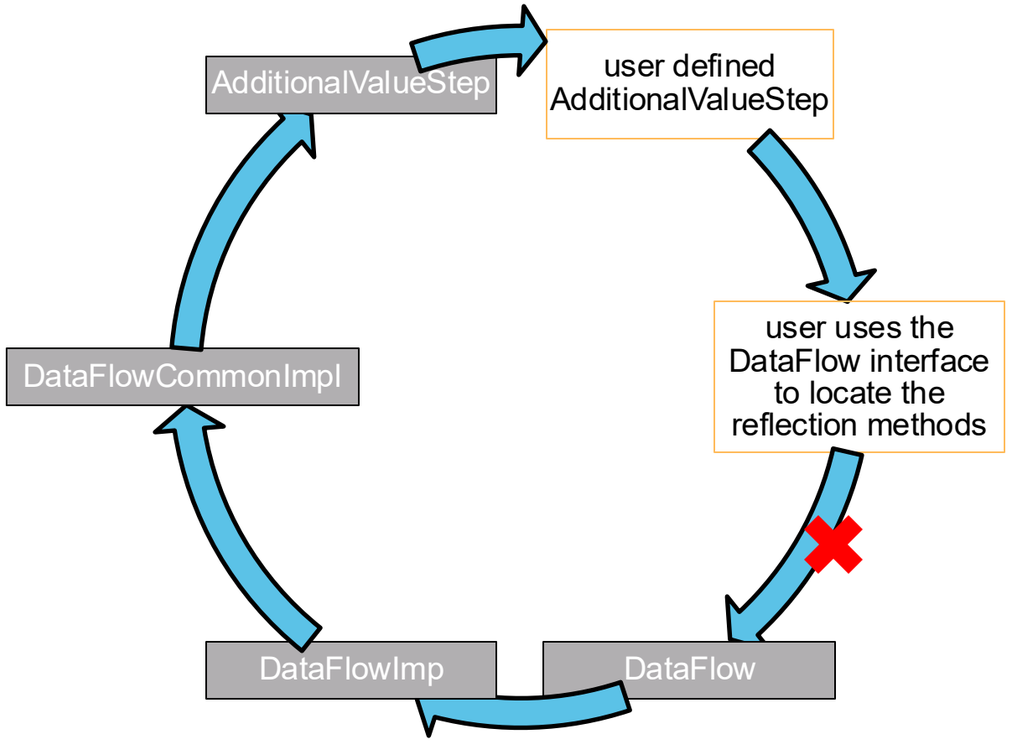
- Solid line indicates confirmed transmission links.
- Dashed line indicates potential transmission links.

Java Reflection

- **Challenge2: How to Patch CodeQL data flow analysis (DFA)?**

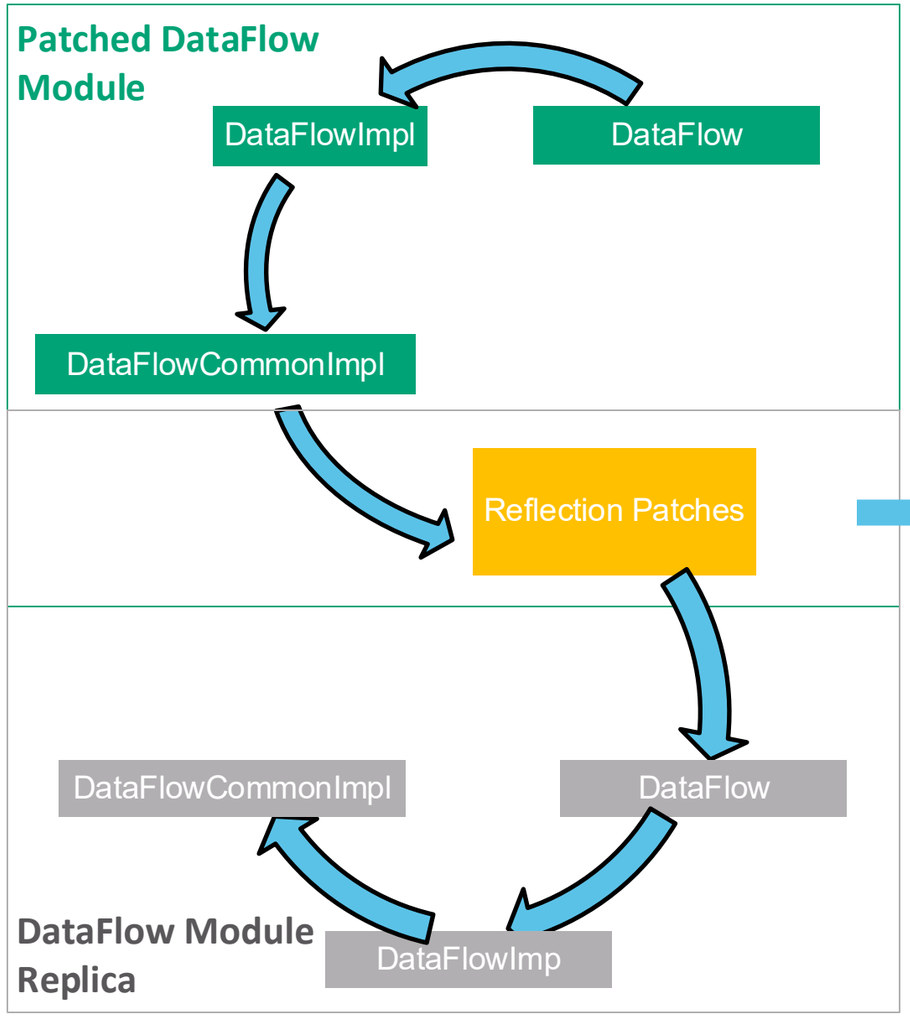
AdditionalValueStep does not work because of ***Non-monotonic recursion***.

1. To achieve reflection analysis, we need to use the global DataFlow.
2. And then connect the Call In/Call Through rules with AdditionalValueStep.
3. But the global DataFlow depends on AdditionalValueStep.



Java Reflection

- **Challenge2: Patch CodeQL data flow analysis (DFA).**



Invoke reflection data propagation policy

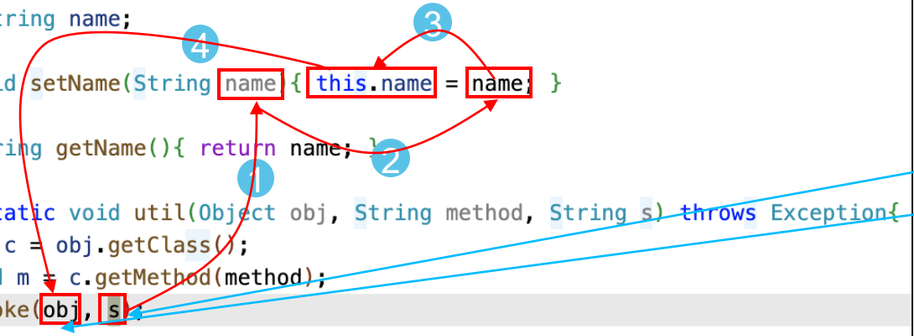
```
r = method.invoke(obj, arg1, …)
Method(param1, …){ this …; return ret;}
```

Step	Demo	Policy
Call In		1. ***arg*** propagates to the ***param.*** 2. ***obj*** propagates to ***parameter this.***
Call Through	public object m(param,…){ … return ret; } r = m.invoke(obj, arg, …);	1. ***ret*** propagates to the method call expression ***(m.invoke(obj, arg, …))*** 2. If there is a ***PostUpdateNode*** in the method, it should also be propagated to the corresponding **obj/arg[post update]**.

Java Reflection

Invoke Call

```
1   import java.lang.reflect.Method;
2
3   public class InvokeDemo {
4       private String name;
5
6       public void setName(String name){ this.name = name; }
7
8       public String getName(){ return name; }
9
10      private static void util(Object obj, String method, String s) throws Exception{
11          Class c = obj.getClass();
12          Method m = c.getMethod(method);
13          m.invoke(obj, s);
14      }
15
16      public static void main(String[] args) throws Exception {
17          String arg = args[0];
18          InvokeDemo tt = new InvokeDemo();
19
20          util(tt, "setName", arg);
21          String name = tt.getName();
22          System.out.println(name);
23      }
24  }
```



1	args : String[]	InvokeDemo.java:16:29
2	arg : String	InvokeDemo.java:20:29
3	s : String	InvokeDemo.java:10:57
4	s : String	InvokeDemo.java:13:23
5	obj [post update] : InvokeDemo [name] : String	InvokeDemo.java:13:18
6	tt [post update] : InvokeDemo [name] : String	InvokeDemo.java:20:14
7	tt : InvokeDemo [name] : String	InvokeDemo.java:21:23
8	parameter this : InvokeDemo [name] : String	InvokeDemo.java:8:19
9	this <.field> : InvokeDemo [name] : String	InvokeDemo.java:8:37
10	name : String	InvokeDemo.java:8:37
11	getName(...) : String	InvokeDemo.java:21:23
12	name	InvokeDemo.java:22:28

- As can be seen from the results, by supporting Java reflection, taint node ***s*** can propagate to ***obj[name].***

Java Value Passing

What is Java Value Passing?

In Java, the way to pass actual parameters to methods is ***pass-by-value***:

- If the parameter is a primitive type, it's straightforward; what is passed is a copy of the literal value of the primitive type, and a copy is created.
- If the parameter is not a primitive type, what is passed is a copy of the address value in the heap of the object referenced by the actual parameter, and similarly, a copy is created.

```
1   import java.io.*;
2   import java.nio.Buffer;
3
4   public class DemoObject {
5       public static void main(String[] args) {
6           String arg = args[0];
7           Person p = new Person("origin");
8           System.out.println("before change " + "p:" + p + " name: " + p.getName());
9           change(p, arg);
10          System.out.println("after change  " + "p:" + p + " name: " + p.getName());
11      }
12
13      public static void change(Person tp, String nn) {
14          tp.setName(nn);
15      }
16  }
```

```
m0d9@ src % git:(master) ✗ java DemoObject aaa
before change p:Person@15db9742 name: origin
after change  p:Person@15db9742 name: aaa
```

Java Value Passing

What is the problem?

However, the CodeQL Java parameter passing model may miss some instances when multiple copies of non-primitive type parameters exist.

- Both ***p*** and ***d.a*** are instances of the Person class, pointing to the same object, which is a copy of the heap address of that object.
- In the CodeQL analysis flow, the analysis considers that ***d.a*** has changed but cannot track ***pn***.
- Actually, the final ***pn*** is also tainted.

```java
import java.io.*;

public class DemoField {
    Person a;

    DemoField(String, Person a){
        this.a = a;
    }

    public void setPersonName(String n){
        this.a.setName(n);
    }

    public static void main(String[] args) {
        String arg0 = args[0];

        Person p = new Person("test");
        DemoField d = new DemoField(p);
        System.out.println("p:"+p);
        System.out.println("d.a:"+p);

        d.setPersonName(arg0);
        String pn = p.getName();
        System.out.println(pn);
    }
}
```

```java
public class Person{
    private String name;

    public Person(String n){
        this.name = n;
    }

    public void setName(String n){
        this.name = n;
    }

    public String getName(){
        return this.name;
    }
}
```

```
m0d9@ src % git:(master) ✗ java DemoField aaa
p:Person@15db9742
d.a:Person@15db9742
aaa
```

Java Value Passing

How to support multiple copies of non-primitive type parameters value passing?

Taking the copy stored in a ***Field*** as an example.

1. Locate the ***field*** that is ***non-primitive***.
2. For this field, find its ***store*** operations, identifying both ***non-PostUpdateNode*** and *PostUpdateNode* nodes.
3. For ***non-PostUpdateNode store*** operations, use ***global data flow*** to locate the parameter ***param*** and the actual argument ***arg***.
4. For another ***store*** operation of ***PostUpdateNode***, add a mapping from this node to ***arg***.

Tips:

- Implementation may lead to **non-monotonic recursion**, as shown in the Java Reflection Solution.

```
1  import java.io.*;
2
3  public class DemoField {
4      Person a;
5
6      DemoField(String, Person a) {
7          this.a = a;
8      }
9
10     public void setPersonName(String n){
11         this.a.setName(n);
12     }
13
14     public static void main(String[] args) {
15         String arg0 = args[0];
16
17         Person p = new Person("test");
18         DemoField d = new DemoField(p);
19         System.out.println("p:"+p);
20         System.out.println("d.a:"+p);
21
22         d.setPersonName(arg0);
23         String pn = p.getName();
24         System.out.println(pn);
25     }
26 }
```

```
1  public class Person{
2      private String name;
3
4      public Person(String n){
5          this.name = n;
6      }
7
8      public void setName(String n){
9          this.name = n;
10     }
11
12     public String getName(){
13         return this.name;
14     }
15 }
```

Java Value Passing

- Why must use global data flow?

Inter-procedural Call

```
public class DemoFieldCall {
    String name;
    Person a;

    public static Person getPerson(Person a){
        return a;
    }

    DemoFieldCall(String name, Person a){
        this.name = name;
        this.a = getPerson(a);
    }

    public void setPersonName(String n){
        this.a.setName(n);
        // System.out.println(this.a);
    }

    public static void main(String[] args) {
        String arg0 = args[0];

        Person p = new Person("test");
        DemoFieldCall d = new DemoFieldCall("test" , p);
        System.out.println("p:"+p);
        System.out.println("d.a:"+p);

        d.setPersonName(arg0);
        // a.printName();
        String pn = p.getName();
        System.out.println(pn);
    }
}
```

Array Store & Read

```
public class DemoFieldStoreAndRead1 {
    String name;
    Person a;

    DemoFieldStoreAndRead1(String name, Person a){
        this.name = name;
        Person[] ps = new Person[2];
        ps[0] = a;
        this.a = ps[0];
    }

    public void setPersonName(String n){
        this.a.setName(n);
        // System.out.println(this.a);
    }

    public static void main(String[] args) {
        String arg0 = args[0];

        Person p = new Person("test");
        DemoFieldStoreAndRead1 d = new DemoFieldStoreAndRead1("test" , p);
        System.out.println("p:"+p);
        System.out.println("d.a:"+p);

        d.setPersonName(arg0);
        // a.printName();
        String pn = p.getName();
        System.out.println(pn);
    }
}
```

Map Store & Read

```
public class DemoFieldStoreAndRead4 {
    String name;
    Person a;

    DemoFieldStoreAndRead4(String name, Person a){
        this.name = name;
        HashMap<Integer, Person> ps = new HashMap<Integer, Person>();
        ps.put(1, a);
        this.a = ps.get(1);
    }

    public void setPersonName(String n){
        this.a.setName(n);
        // System.out.println(this.a);
    }

    public static void main(String[] args) {
        String arg0 = args[0];

        Person p = new Person("test");
        DemoFieldStoreAndRead4 d = new DemoFieldStoreAndRead4("test" , p);
        System.out.println("p:"+p);
        System.out.println("d.a:"+p);

        d.setPersonName(arg0);
        // a.printName();
        String pn = p.getName();
        System.out.println(pn);
    }
}
```

Results of research

Newly discovered sources/sinks

Statistics on Newly Discovered Sources and Sinks for Popular Golang Frameworks

Framework	Sources	Sinks
github.com/Beego/Beego/v2	32	-
github.com/ClickHouse/clickhouse-go	-	4
github.com/gocraft/dbr	-	20
github.com/labstack/echo	3	-
github.com/valyala/fasthttp	14	-
github.com/gofiber/fiber	10	-
github.com/gogf/gf	12	-
github.com/go-gorp/gorp	-	6
github.com/zeromicro/go-zero	1	-
github.com/kataras/iris	19	-
gopkg.in/macaron.v1	11	-
github.com/jackc/pgx	-	6
github.com/gobuffalo/pop/v6	-	6
github.com/go-resty/resty/v2	-	8
github.com/hashicorp/go-retryablehttp	-	10
github.com/aarondl/sqlboiler	-	13
github.com/jmoiron/sqlx	-	2
github.com/MASTERMINDS/squirrel	-	4
github.com/upper/db	-	10

Scanned 5,000+ repositories, detecting a >15% increase in data flows

```
total 2.8G
-rw-r--r-- 1 root root 6.8M Aug 25  2024 actions-runner-controller.zip
-rw-r--r-- 1 root root 5.6M Aug 25  2024 alaz.zip
-rw-r--r-- 1 root root 5.3M Aug 25  2024 apko.zip
-rw-r--r-- 1 root root  38M Aug 24  2024 argo-cd.zip
-rw-r--r-- 1 root root  25M Aug 24  2024 argo-workflows.zip
-rw-r--r-- 1 root root 9.4M Aug 24  2024 authelia.zip
-rw-r--r-- 1 root root  13M Aug 25  2024 aws-otel-collector.zip
-rw-r--r-- 1 root root  12M Aug 25  2024 bacalhau.zip
-rw-r--r-- 1 root root 1.6M Aug 25  2024 bbscope.zip
-rw-r--r-- 1 root root  53M Aug 24  2024 beats.zip
-rw-r--r-- 1 root root 252M Aug 25  2024 bk-bcs.zip
-rw-r--r-- 1 root root  39M Aug 25  2024 boundary.zip
-rw-r--r-- 1 root root 6.0M Aug 25  2024 brigade.zip
-rw-r--r-- 1 root root 4.9M Aug 25  2024 buildpacks.zip
-rw-r--r-- 1 root root  39M Aug 25  2024 cds.zip
```

Case study - CVE-2024-45387

SQL injection vulnerability in Traffic Ops in Apache Traffic Control

CVE-2024-45387 Detail

Description

An SQL injection vulnerability in Traffic Ops in Apache Traffic Control <= 8.0.1, >= 8.0.0 allows a privileged user with role "admin", "federation", "operations", "portal", or "steering" to execute arbitrary SQL against the database by sending a specially-crafted PUT request. Users are recommended to upgrade to version Apache Traffic Control 8.0.2 if you run an affected version of Traffic Ops.

Metrics | CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NVD	**NIST:** NVD	**Base Score:** 8.8 HIGH	**Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
C	**CNA:** Apache Software Foundation	**Base Score:** 9.9 CRITICAL	**Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

Automated Vulnerability Scanning with CodeQL

Apache Traffic Control

APACHE TRAFFIC CONTROL

Apache Traffic Control allows you to build a large scale content delivery network using open source. Built around Apache Traffic Server as the caching software, Traffic Control implements all the core functions of a modern CDN.

Slack | join #traffic-control | Follow @trafficctrlcdn | Apache Traffic Control | 163

release v8.0.2 | commits since v8.0.2 223

Build Status [1]

CDN-in-a-Box CI failing | docs passing

Code Status [1]

Weasel License checks passing | Go Format failing | Go Vet passing

CodeQL - C++ passing | CodeQL - Go passing | CodeQL - Java failing | CodeQL - Javascript passing

CodeQL - Python passing

CVE-2024-45387 – Data flow

Read parameters from user input

trafficcontrol / traffic_ops / traffic_ops_golang / api / **shared_handlers.go**

```go
func GetCombinedParams(r *http.Request) (map[string]string, error) {
	combinedParams := make(map[string]string)
	q := r.URL.Query()
	for k, v := range q {
		combinedParams[k] = v[0] //we take the first value and do not sup
	}

	ctx := r.Context()
	pathParams, err := GetPathParams(ctx)
	if err != nil {
		return combinedParams, fmt.Errorf("no path parameters: %s", err)
	}
	//path parameters will overwrite query parameters
	for k, v := range pathParams {
		combinedParams[k] = v
	}

	return combinedParams, nil
}
```

Process parameters from HTTP requests

trafficcontrol / traffic_ops / traffic_ops_golang / api / **api.go**

```go
// AllParams takes the request (in which the router has inserted context for path parameters), and an array of parameters required t
// This is a helper for the common case; not using this in unusual cases is perfectly acceptable.
func AllParams(req *http.Request, required []string, ints []string) (map[string]string, map[string]int, error, error, int) {
	params, err := GetCombinedParams(req)
	if err != nil {
		return nil, nil, nil, errors.New("getting combined URI parameters: " + err.Error()), http.StatusInternalServerError
	}
	params = StripParamJSON(params)
	if err := ParamsHaveRequired(params, required); err != nil {
		return nil, nil, errors.New("required parameters missing: " + err.Error()), nil, http.StatusBadRequest
	}
	intParams, err := IntParams(params, ints)
	if err != nil {
		return nil, nil, errors.New("getting integer parameters: " + err.Error()), nil, http.StatusBadRequest
	}
	return params, intParams, nil, nil, 0
}
```

Validate request parameters

trafficcontrol / traffic_ops / traffic_ops_golang / api / **info.go**

```go
func NewInfo(r *http.Request, requiredParams []string, intParamNames []string) (*Info, error, error, int) {
	db, err := GetDB(r.Context())
	if err != nil {
		return &Info{Tx: &sqlx.Tx{}}, fmt.Errorf("getting db: %w", err), nil, http.StatusInternalServerError
	}
	cfg, err := GetConfig(r.Context())
	if err != nil {
		return &Info{Tx: &sqlx.Tx{}}, fmt.Errorf("getting config: %w", err), nil, http.StatusInternalServerError
	}
	tv, err := GetTrafficVault(r.Context())
	if err != nil {
		return &Info{Tx: &sqlx.Tx{}}, fmt.Errorf("getting TrafficVault: %w", err), nil, http.StatusInternalServe
	}
	reqID, err := getReqID(r.Context())
	if err != nil {
		return &Info{Tx: &sqlx.Tx{}}, fmt.Errorf("getting reqID: %w", err), nil, http.StatusInternalServerError
	}
	version := GetRequestedAPIVersion(r.URL.Path)

	user, err := auth.GetCurrentUser(r.Context())
	if err != nil {
		return &Info{Tx: &sqlx.Tx{}}, fmt.Errorf("getting user: %w", err), nil, http.StatusInternalServerError
	}
	params, intParams, userErr, sysErr, errCode := AllParams(r, requiredParams, intParamNames)
	if userErr != nil || sysErr != nil {
		return &Info{Tx: &sqlx.Tx{}}, userErr, sysErr, errCode
	}
```

Insert comments into the database

trafficcontrol / traffic_ops / traffic_ops_golang / deliveryservice / request / comment / **comments.go**

```go
// Update is used to modify an existing DeliveryServiceRequestCommentV5 in the database.
func Update(w http.ResponseWriter, r *http.Request) {
	var deliveryServiceRequestComment tc.DeliveryServiceRequestCommentV5

	inf, userErr, sysErr, errCode := api.NewInfo(r, []string{"id"}, nil)
```

QueryRowx

```go
err := inf.Tx.QueryRowx(selectQuery() + `WHERE dsrc.id=` + inf.Params["id"]).StructScan(&current)
```

The sink function was omitted...

The *QueryRowx* function in the *sqlx* framework was omitted in CodeQL

```
- addsTo:
    pack: codeql/go-all
    extensible: sinkModel
  data:
    - ["github.com/jmoiron/sqlx", "DB", True, "Get", "", "", "Argument[1]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "DB", True, "MustExec", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "DB", True, "NamedExec", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "DB", True, "NamedQuery", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "DB", True, "Queryx", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "DB", True, "Select", "", "", "Argument[1]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "Get", "", "", "Argument[1]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "MustExec", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "NamedExec", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "NamedQuery", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "Queryx", "", "", "Argument[0]", "sql-injection", "manual"]
    - ["github.com/jmoiron/sqlx", "Tx", True, "Select", "", "", "Argument[1]", "sql-injection", "manual"]
```

Our method found this sink and thus the vulnerability

Vulnerable code snippet

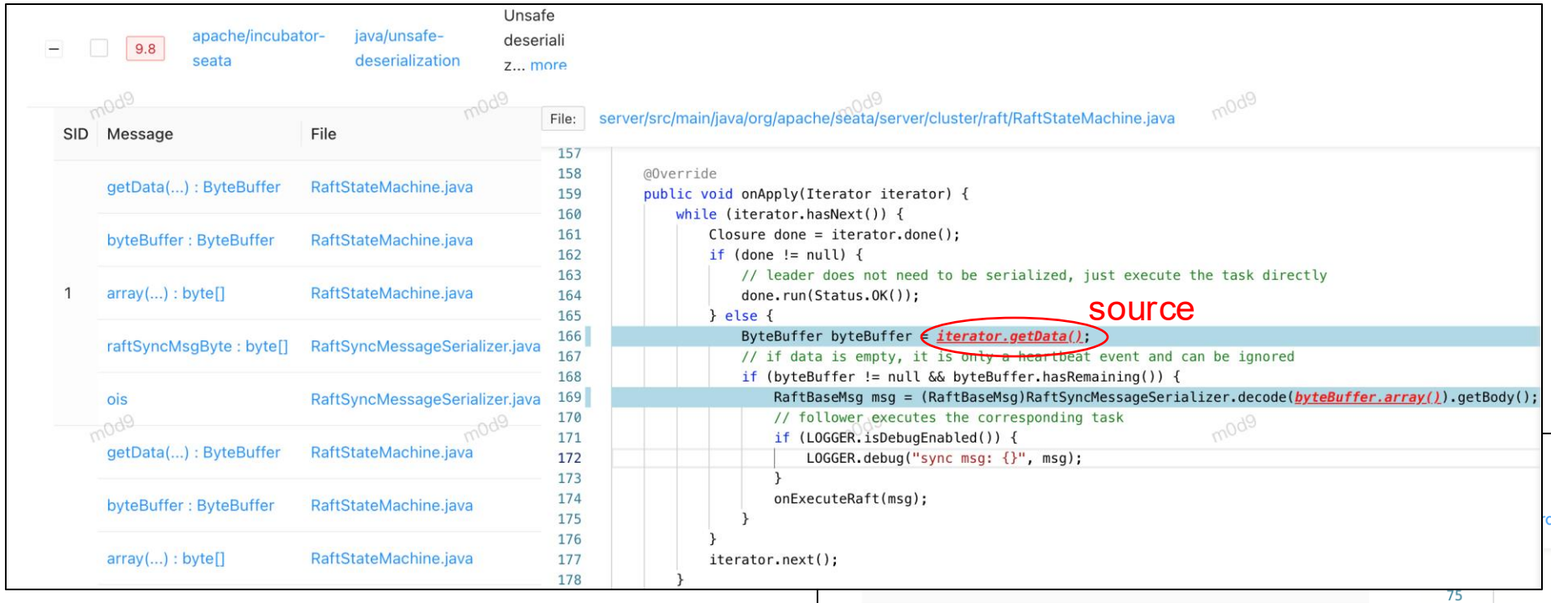
```
err := inf.Tx.QueryRowx(selectQuery() +
`WHERE dsrc.id=` +
inf.Params["id"]).StructScan(&current)
```

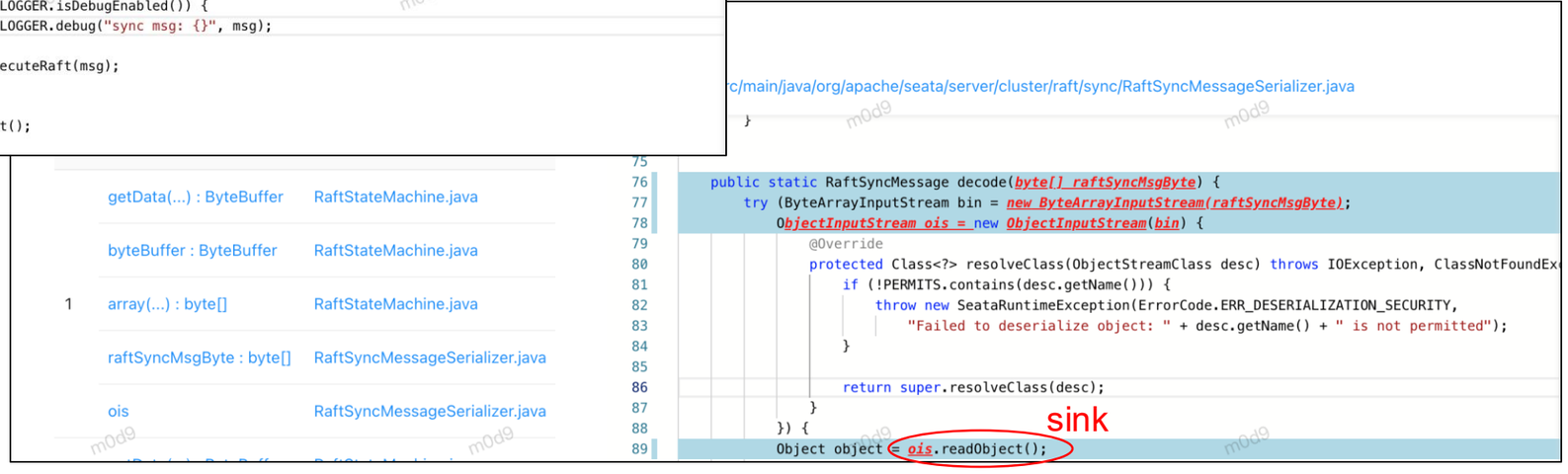
Our method reported the sink function


```
 {
   ""method"": ""DB.Get"",
   ""code"": ""func (db *DB) Get(dest interface{}, query string, args ...interface{}) error {\n    return Get(db, dest, query, a
rgs...)\n}"",
   ""score"": 1.0
 },
 {
   ""method"": ""Tx.Queryx"",
   ""code"": ""func (tx *Tx) Queryx(query string, args ...interface{}) (*Rows, error) {\n    r, err := tx.Tx.Query(query, args..
.)\n    if err != nil {\n        return nil, err\n    }\n    return &Rows{Rows: r, unsafe: tx.unsafe, Mapper: tx.Mapper}, err\n}"
",
   ""score"": 1.0
 },
 {
   ""method"": ""Tx.QueryRowx"",
   ""code"": ""func (tx *Tx) QueryRowx(query string, args ...interface{}) *Row {\n    rows, err := tx.Tx.Query(query, args...)\n
   return &Row{rows: rows, err: err, unsafe: tx.unsafe, Mapper: tx.Mapper}\n}"",
   ""score"": 1.0
 },
 {
   ""method"": ""Tx.MustExec"",
   ""code"": ""func (tx *Tx) MustExec(query string, args ...interface{}) sql.Result {\n    return MustExec(tx, query, args...)\n
}"",
   ""score"": 1.0
 },
 {
   ""method"": ""Tx.Select"",
   ""code"": ""func (tx *Tx) Select(dest interface{}, query string, args ...interface{}) error {\n    return Select(tx, dest, qu
ery, args...)\n}"",
   ""score"": 1.0
 },
 {
   ""method"": ""Tx.Get"",
   ""code"": ""func (tx *Tx) Get(dest interface{}, query string, args ...interface{}) error {\n    return Get(tx, dest, query, a
rgs...)\n}"",
   ""score"": 1.0
 },
```

Reproduced historical CVEs

CVE-2024-47552 CVE-2024-56180 CNVD-2023-45001	Apache Seata, Apache EventMesh, Alibaba Nacos JRaft vulnerability; other affected applications include Apache Ignite and Apache HugeGraph.	Missing source rule





Reproduced historical CVEs

CVE-2023-46604	Apache ActiveMQ Remote Code Execution Vulnerability	Cross-Thread Missing sink rule

```
56   public class TcpTransport extends TransportThreadSupport implements Transport, Service, Runnable {
614      protected void initializeStreams() throws Exception {
615          TcpBufferedInputStream buffIn = new TcpBufferedInputStream(socket.getInputStream(), ioBufferSize) {
```

source

1

socket.getInputStream
TcpTransport#initializeStreams
TcpTransport#connect
TcpTransport#doStart
TransportThreadSupport#doStart

2

```
25   public abstract class TransportThreadSupport extends TransportSupport implements Runnable {
40       protected void doStart() throws Exception {
41           runner = new Thread(null, this, "ActiveMQ Transport: " + toString(), stackSize);
42           runner.setDaemon(daemon);
43           runner.start();
44       }
```

3

Cross-Thread Jump

```
56   public class TcpTransport extends TransportThreadSupport implements Transport, Service, Runnable
210      public void run() {
211          LOG.trace("TCP consumer thread for " + this + " starting");
212          this.runnerThread=Thread.currentThread();
213          try {
214              while (!isStopped() && !isStopping()) {
215                  doRun();
```

4

TcpTransport#run
TcpTransport#doRun
TcpTransport#readCommand
OpenWireFormat#unmarshal
OpenWireFormat#doUnmarshal
BaseDataStreamMarshaller#tightUnmarsalThrowable
BaseDataStreamMarshaller#createThrowable

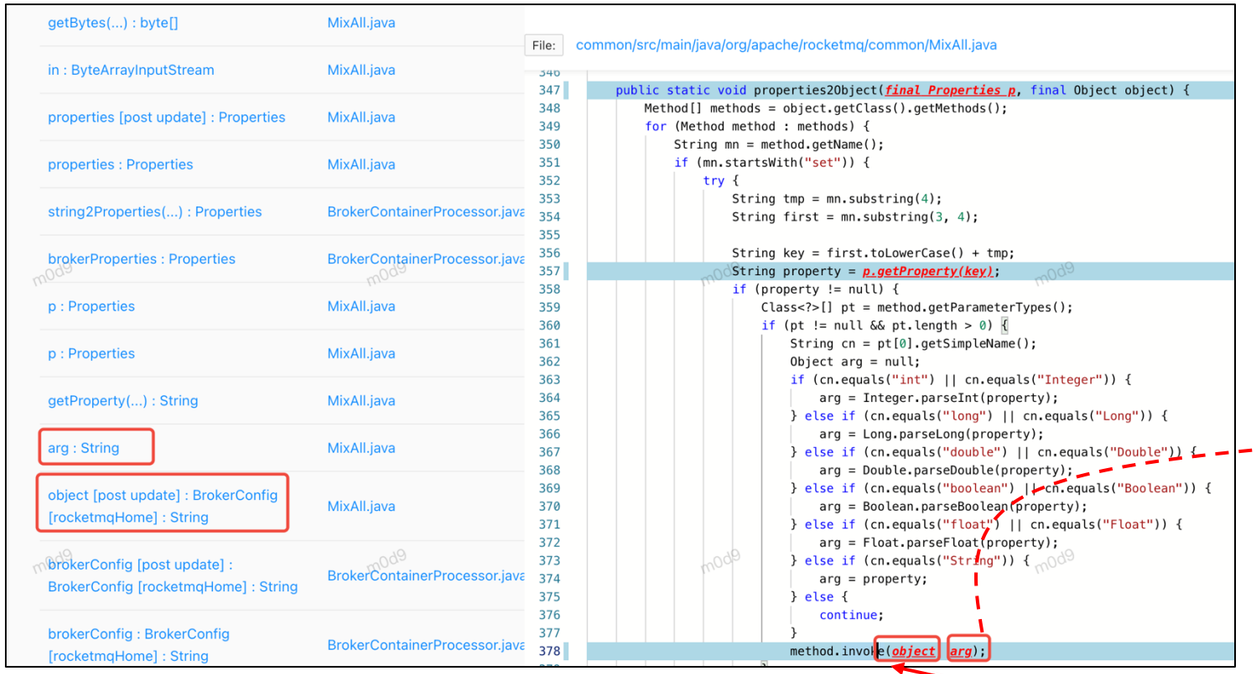
5

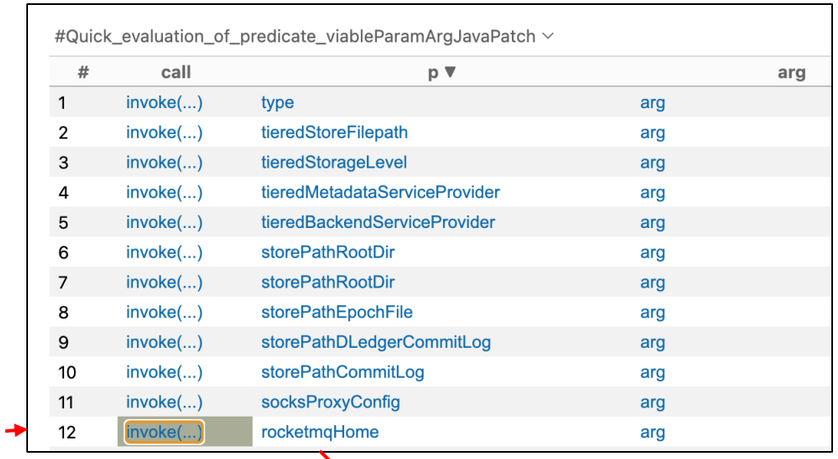
```
31   public abstract class BaseDataStreamMarshaller implements DataStreamMarshaller {
230      private Throwable createThrowable(String className, String message) {
231          try {
232              Class clazz = Class.forName(className, false, BaseDataStreamMarshaller.class.getClassLoader());
233              OpenWireUtil.validateIsThrowable(clazz);
234              Constructor constructor = clazz.getConstructor(new Class[] {String.class});
235              return (Throwable)constructor.newInstance(new Object[] {message});
236          } catch (IllegalArgumentException e) {
237              return e;
238          } catch (Throwable e) {
239              return new Throwable(className + ": " + message);
240          }
```

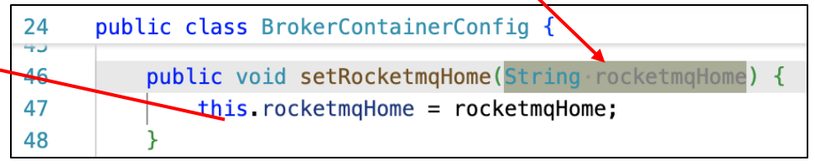
sink

Reproduced historical CVEs

CVE-2023-37582	Apache RocketMQ NameServer Remote Code Execution Vulnerability	Reflection Cross-Thread







- Just explain the Java reflection analysis in it

CVEs we discovered

- 5 new disclosed vulnerabilities, some cases below

CVE-2024-45387 Detail

Description

An SQL injection vulnerability in Traffic O
"operations", "portal", or "steering" to ex
recommended to upgrade to version Apa

Metrics

CVSS Version 4.0

NVD enrichment efforts reference publicly ava

CVSS 3.x Severity and Vector String

NIST: NVD

CNA: Apache Software Foundation

CVE-2024-45794 Detail

Description

devtron is an open source tool int
could utilize and exploit SQL Inje
been addressed in version 0.7.2 a

Metrics

CVSS Versi

NVD enrichment efforts reference pu

CVSS 3.x Severity and Vecto

NIST: NVD

CNA: GitHub, Inc.

CVE-2024-43406 Detail

Description

LF Edge eKuiper is a lightweight IoT data analytics and stream processing engine running on resource-constraint edge devices. A user could utilize and exploit SQL Injection to allow the execution of malicious SQL query via Get method in sqlKvStore. This vulnerability is fixed in 1.14.2.

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

CNA: GitHub, Inc. **Base Score:** 8.8 HIGH **Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Takeaways

- Semantic analysis of code in SAST is particularly suitable for LLM-assisted analysis, and their combination is a research direction.
- CodeQL's data flow analysis mechanism is highly representative, serving as a good start for learning data flow analysis.
- CodeQL's data flow analysis is not perfect and can be studied, modified, and improved.

Q & A

leonyuanluo@tencent.com
zzzzjchen@tencent.com
landonsun@tencent.com
jitxie@tencent.com


black hat
BRIEFINGS
AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS
Thank you!
#BHUSA @BlackHatEvents