Automatic Annotation of Confidential Data in Java Programs

Iulia Bastys   Pauline Bolignano   Franco Raimondi   Daniel Schoepe

CHALMERS   amazon

Securing applications

FUZZING

Symbolic execution

Access control

Information flow control

Testing

Manual code inspection

Securing applications

FUZZING

Symbolic execution

Access control

Formal guarantees

Information flow control

Testing

Manual code inspection

IFC in a nutshell

- explicit flows:

```
x := y
```

- implicit flows:

```
if (x) then
   y := true
else
   y := false
```

IFC in a nutshell

- explicit flows:

```
    public        secret
x            := y
```

✗

- implicit flows:

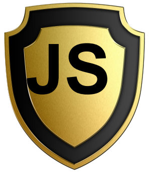
```
        secret
if (y           ) then
          public
    x             := true
else
          public
    x             := false
```

✗

Plethora of IFC trackers

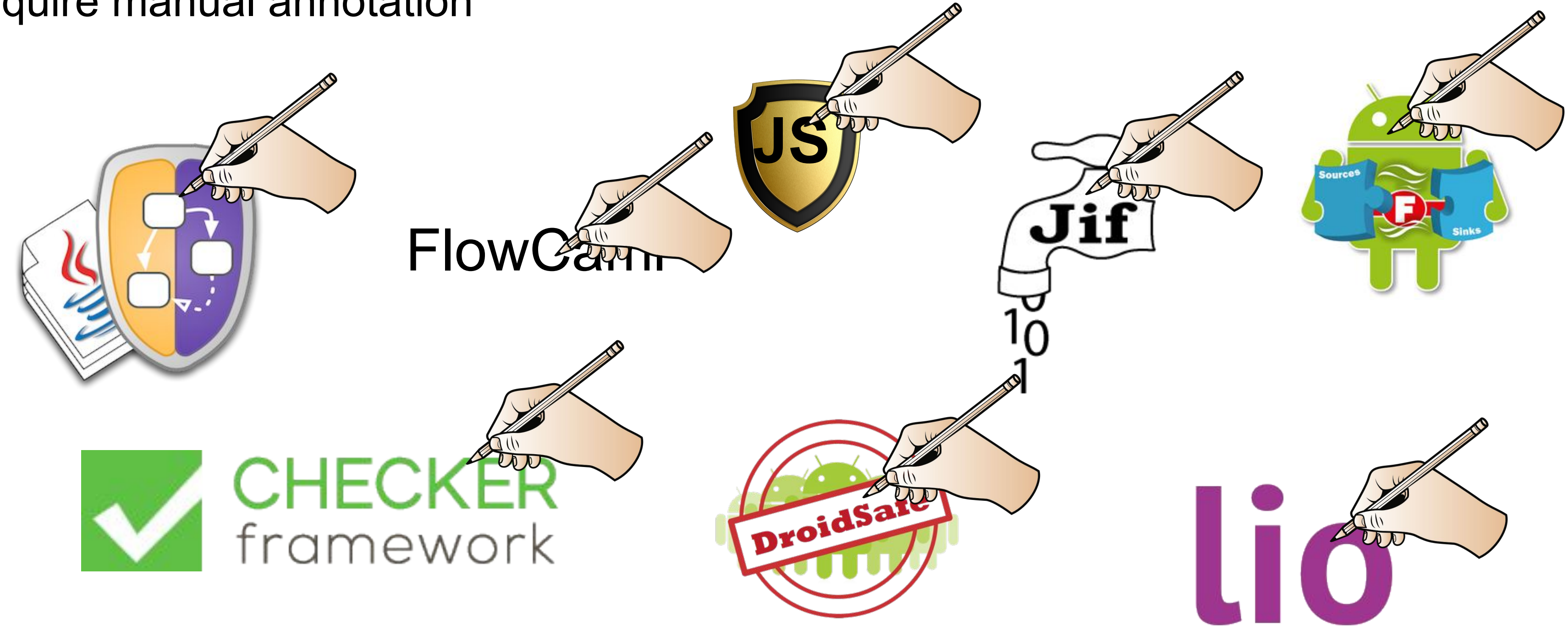
- JavaScript, Java, OCaml, Haskell, etc.
- dynamic, static, hybrid

FlowCaml



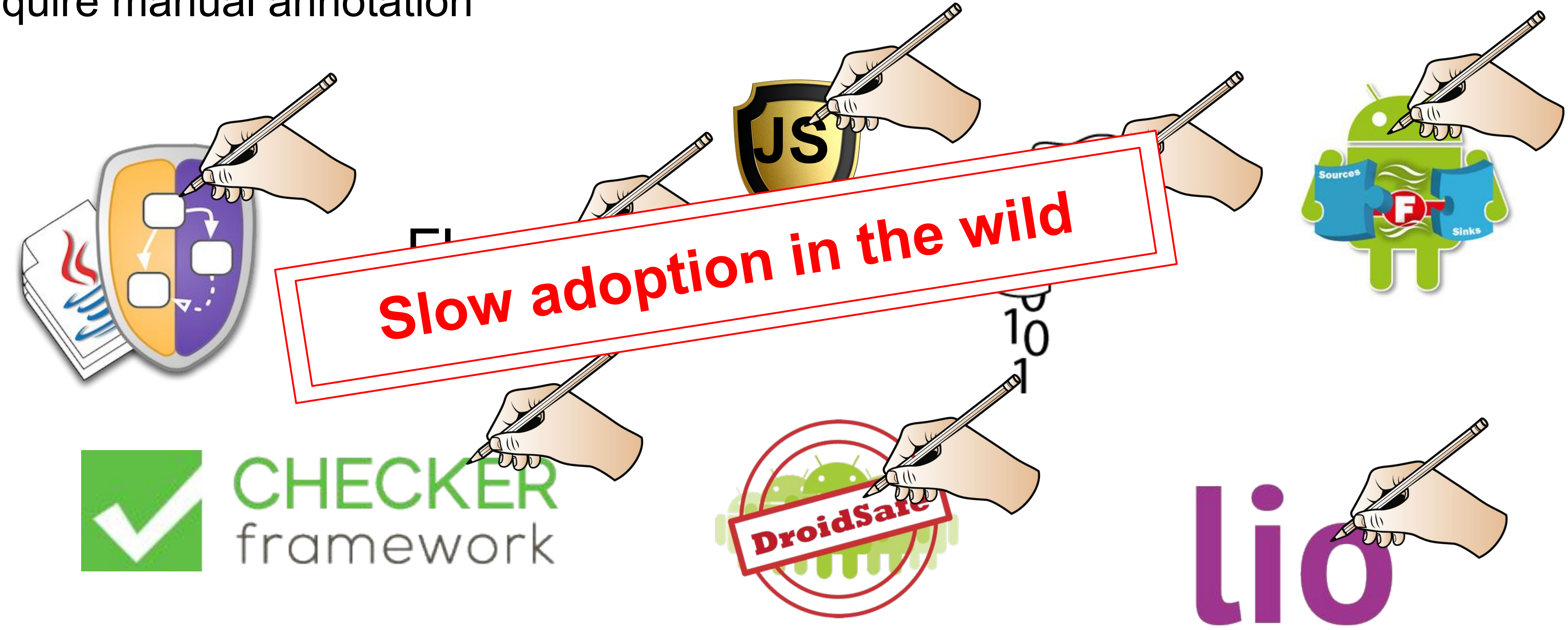
Plethora of IFC trackers

- require manual annotation



Plethora of IFC trackers

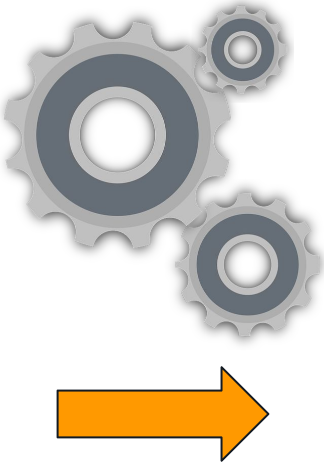
- require manual annotation



Slow adoption in the wild

Bridge the gap: Automatically annotate secret data

```
public String myMethod() {
  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```



```
public String myMethod() {
                    secret
  String high              = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

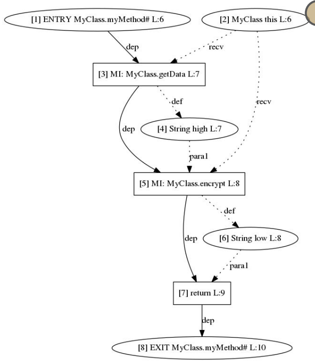
Which data is secret?

```
encrypt(secret)

secret = decrypt(...)
```

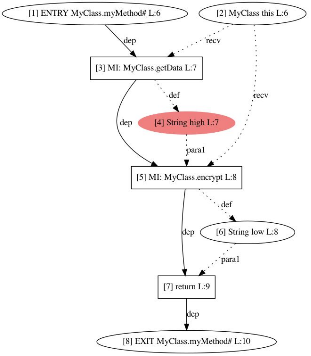
Three-step approach

```
public String myMethod() {
  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

1 Groum generation


[1] ENTRY MyClass.myMethod# L:6
[2] MyClass this L:6
dep
recv
[3] MI: MyClass.getData L:7
def
recv
dep
[4] String high L:7
para1
[5] MI: MyClass.encrypt L:8
def
dep
[6] String low L:8
para1
[7] return L:9
dep
[8] EXIT MyClass.myMethod# L:10

2 Datalog DFA


[1] ENTRY MyClass.myMethod# L:6
[2] MyClass this L:6
dep
recv
[3] MI: MyClass.getData L:7
def
recv
dep
[4] String high L:7
para1
[5] MI: MyClass.encrypt L:8
def
dep
[6] String low L:8
para1
[7] return L:9
dep
[8] EXIT MyClass.myMethod# L:10

3 Annotation

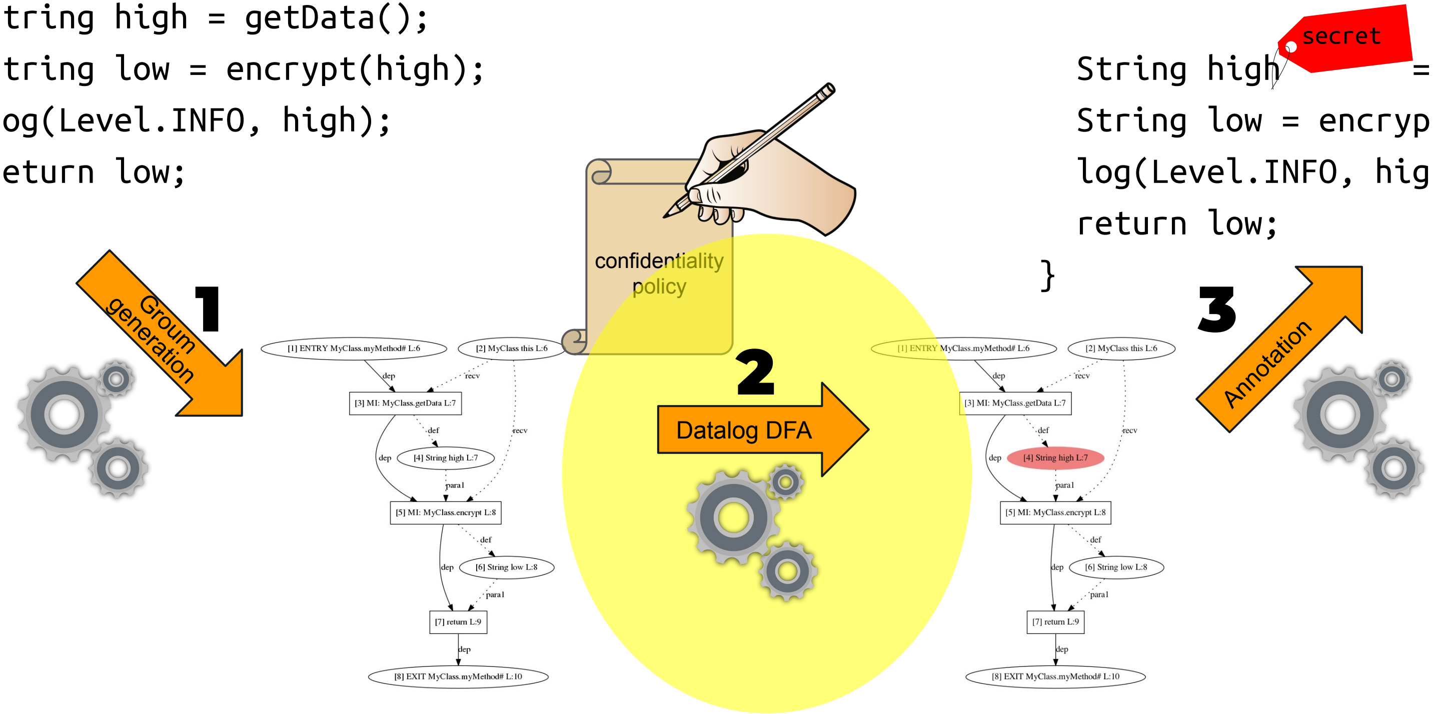
secret

```
public String myMethod() {

  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

Three-step approach

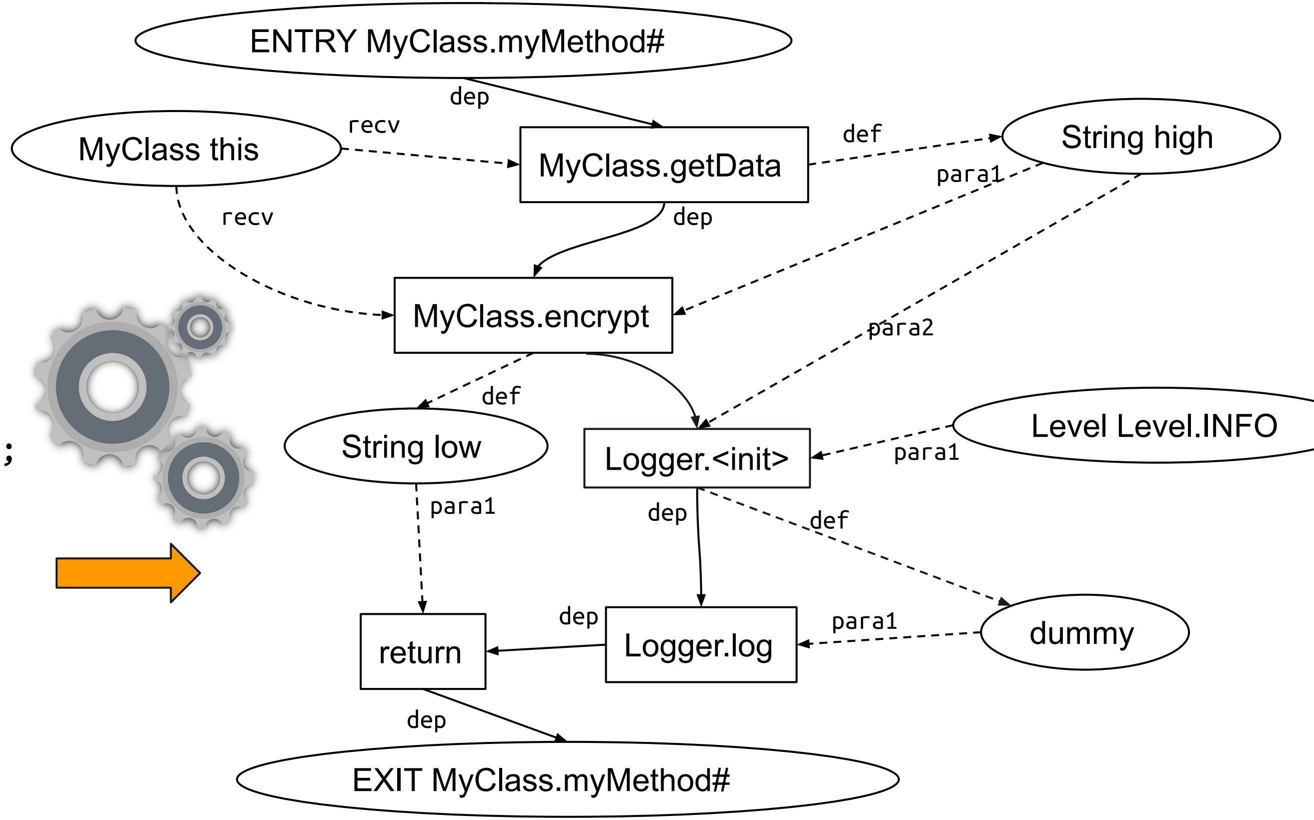
```
public String myMethod() {
  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

```
public String myMethod() {

  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

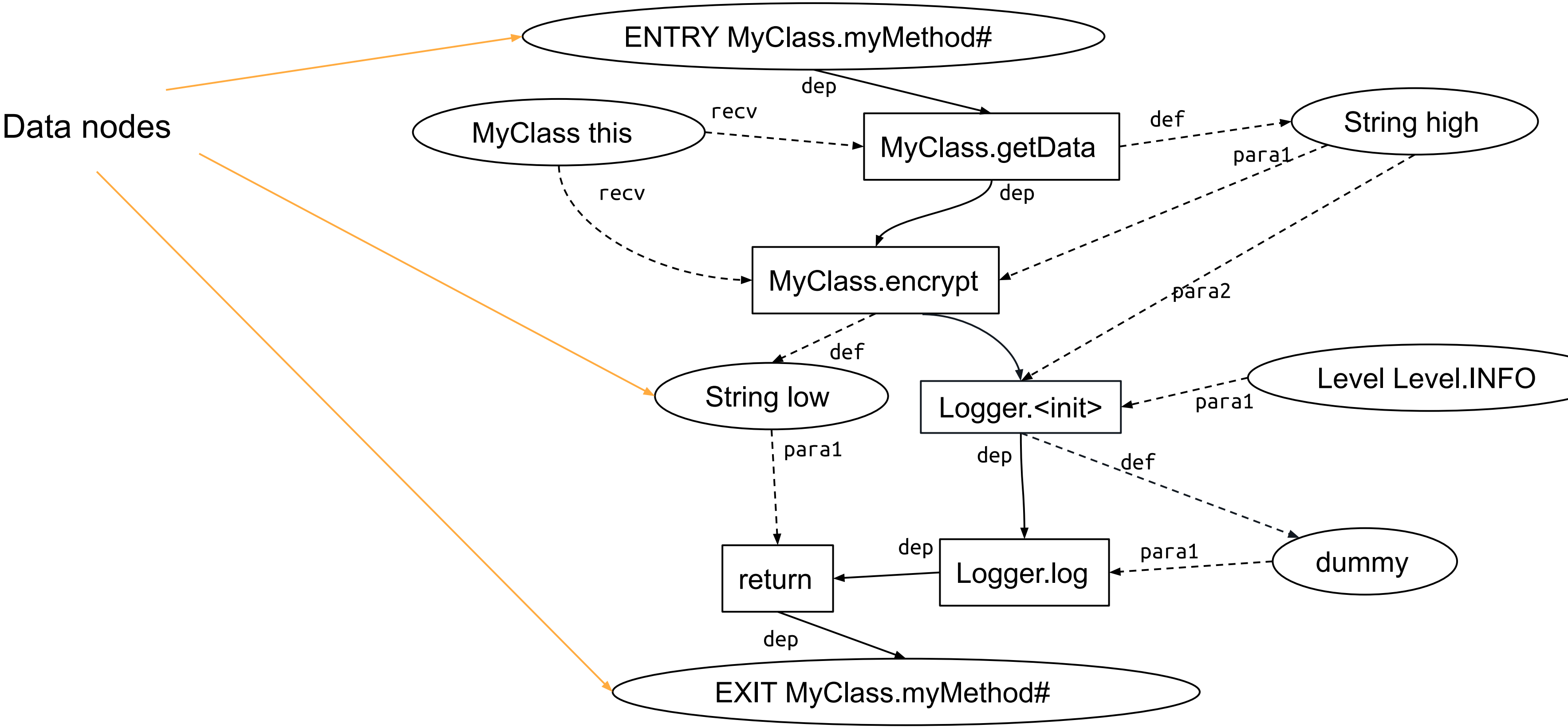


1. Groums

```
public String myMethod() {
  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```

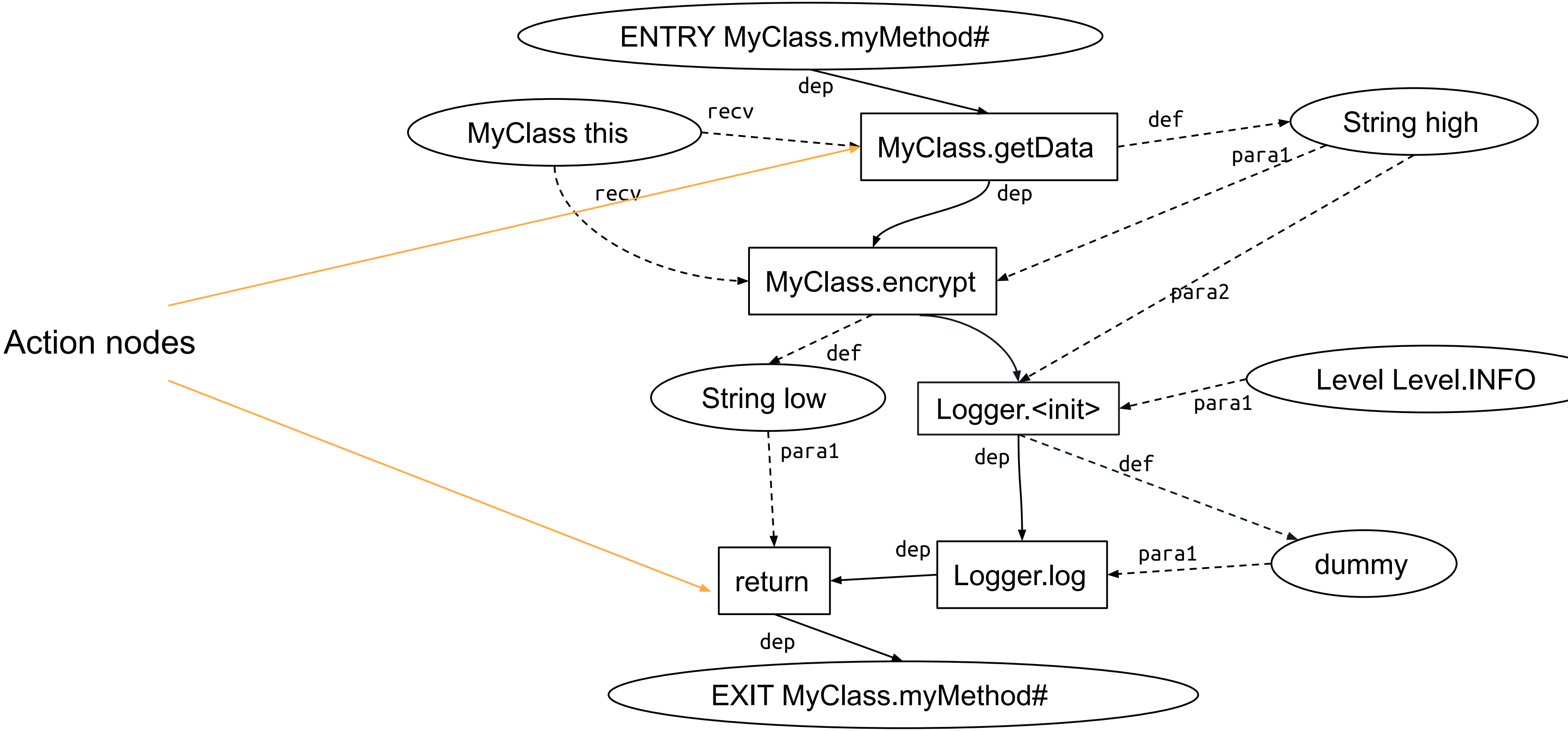


1. Grooms



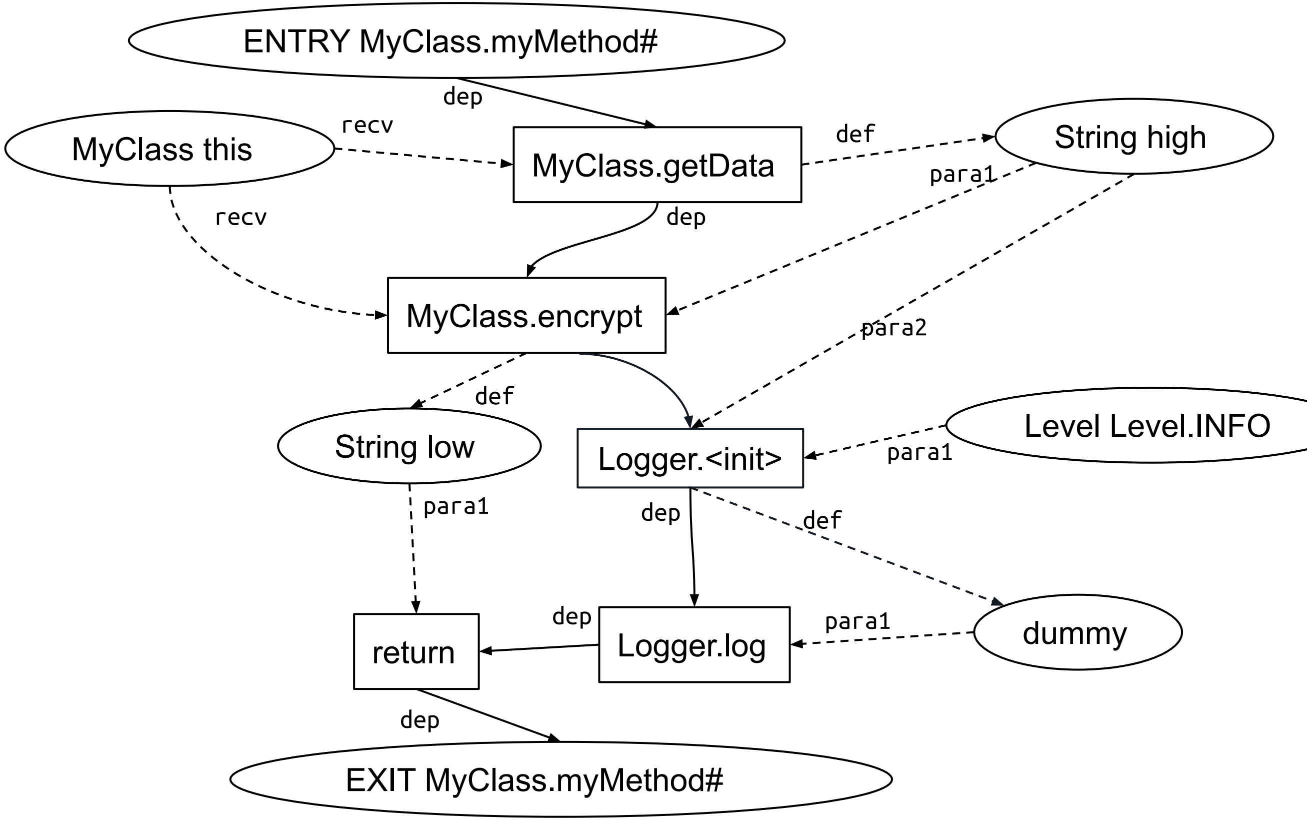
Data nodes

1. Groums

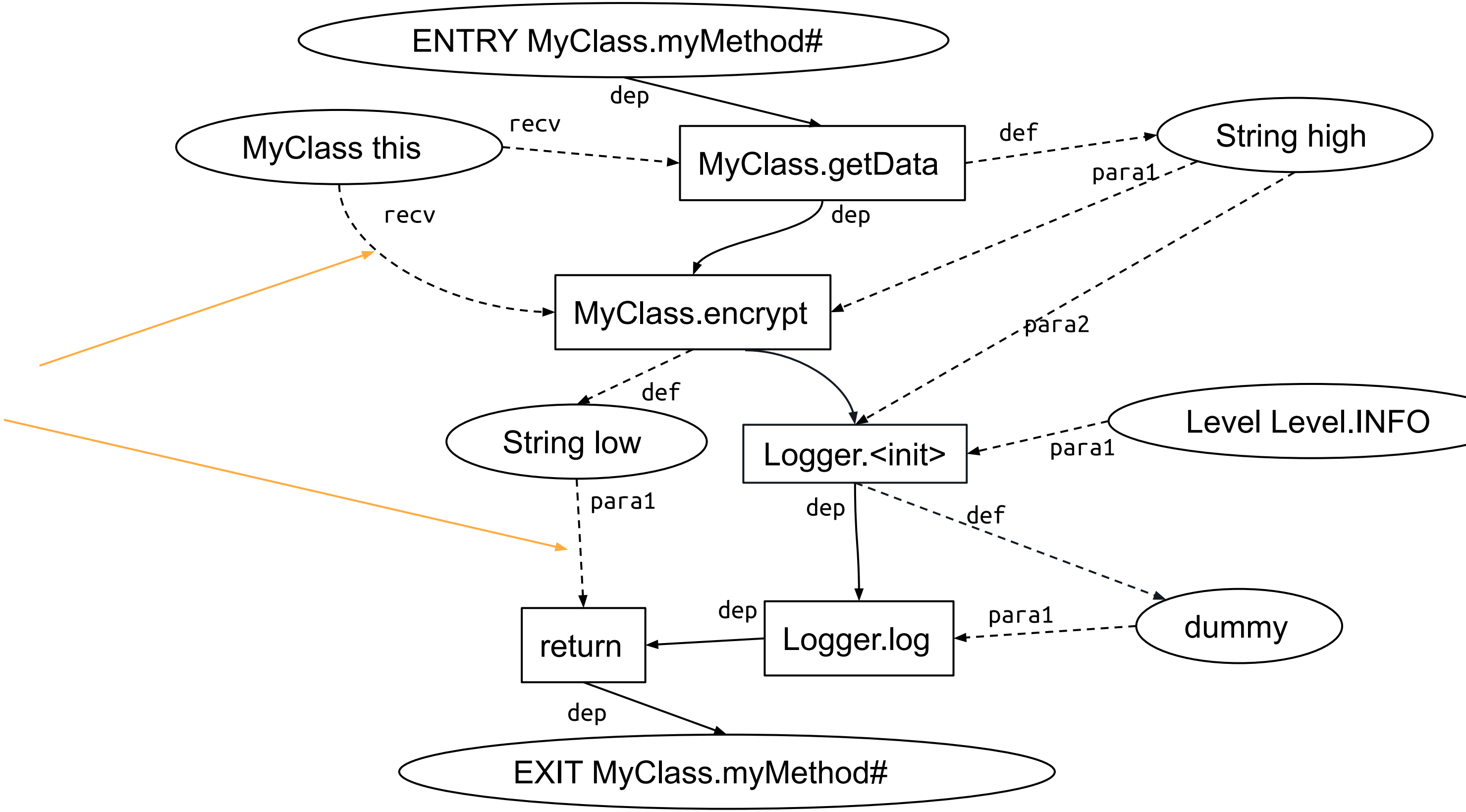

ENTRY MyClass.myMethod#
dep
recv
MyClass this
MyClass.getData
def
String high
para1
recv
dep
MyClass.encrypt
para2
Action nodes
def
String low
Logger.<init>
Level Level.INFO
para1
para1
dep
def
dep
return
Logger.log
para1
dummy
dep
EXIT MyClass.myMethod#

1. Groums

Control nodes (not here)

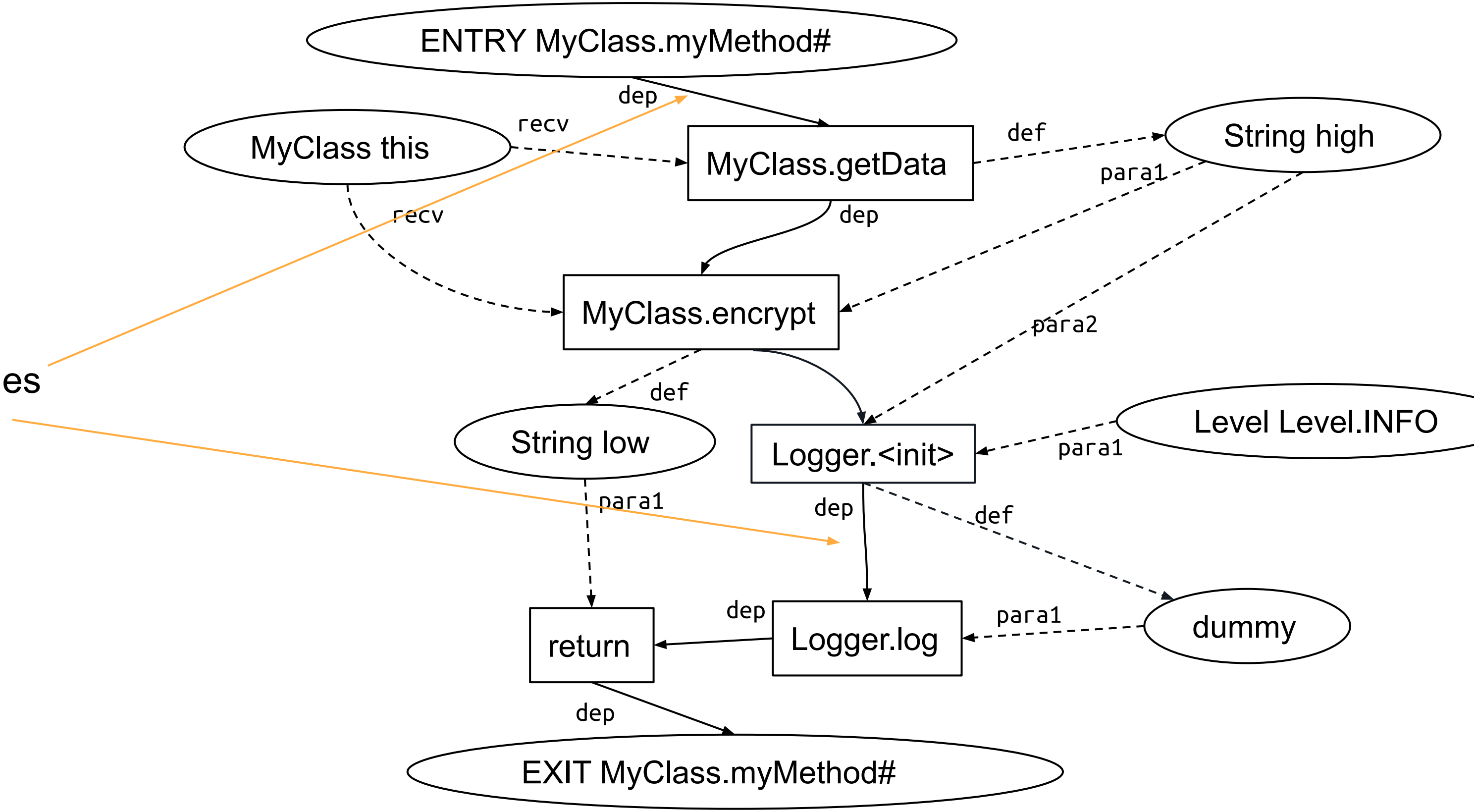


1. Groums



Data flow edges (dotted)

1. Groums

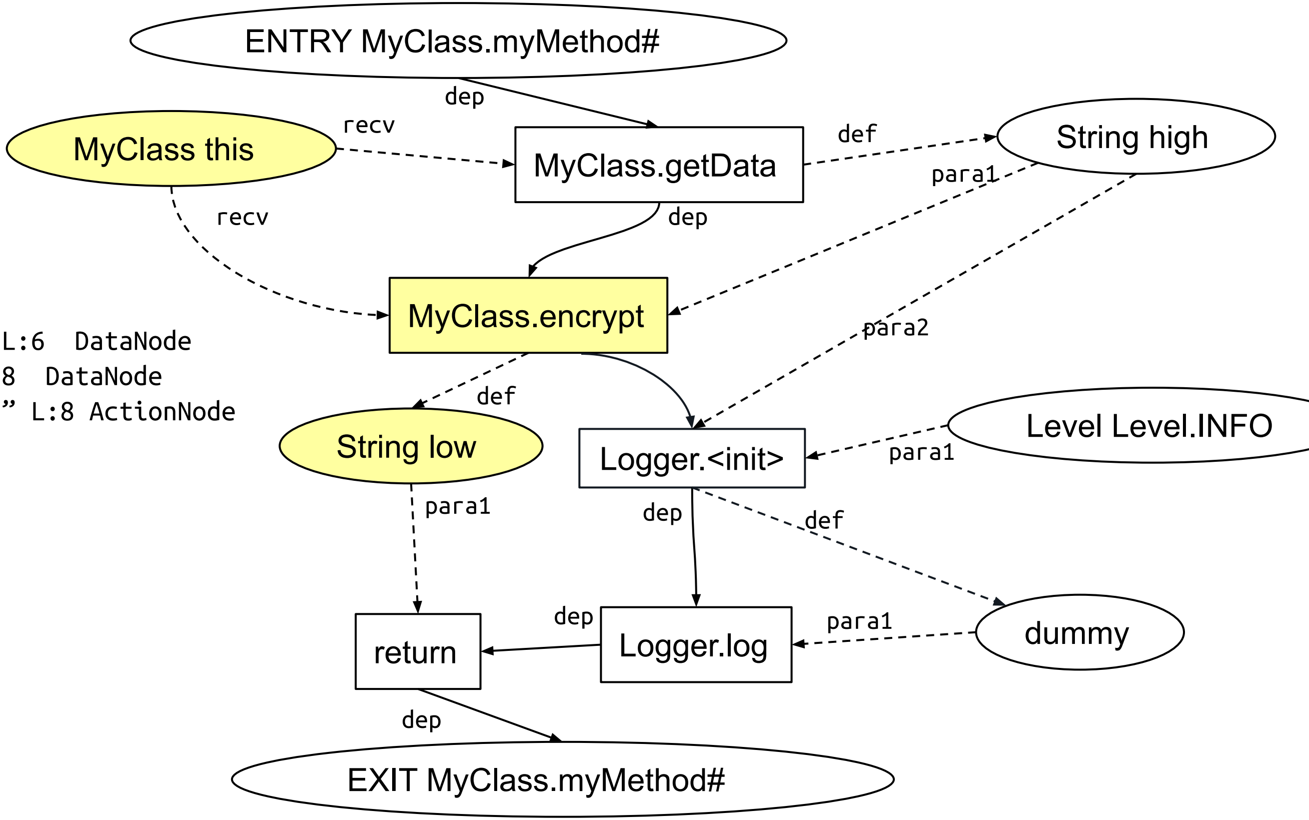


Control flow edges (solid)

2. Datalog DFA | Groum encoding

Node.facts

```
MyClass.myMethod#  2  “MyClass this”  L:6  DataNode
MyClass.myMethod#  6  “String low”  L:8  DataNode
MyClass.myMethod#  5  “MyClass.encrypt” L:8 ActionNode
...
```



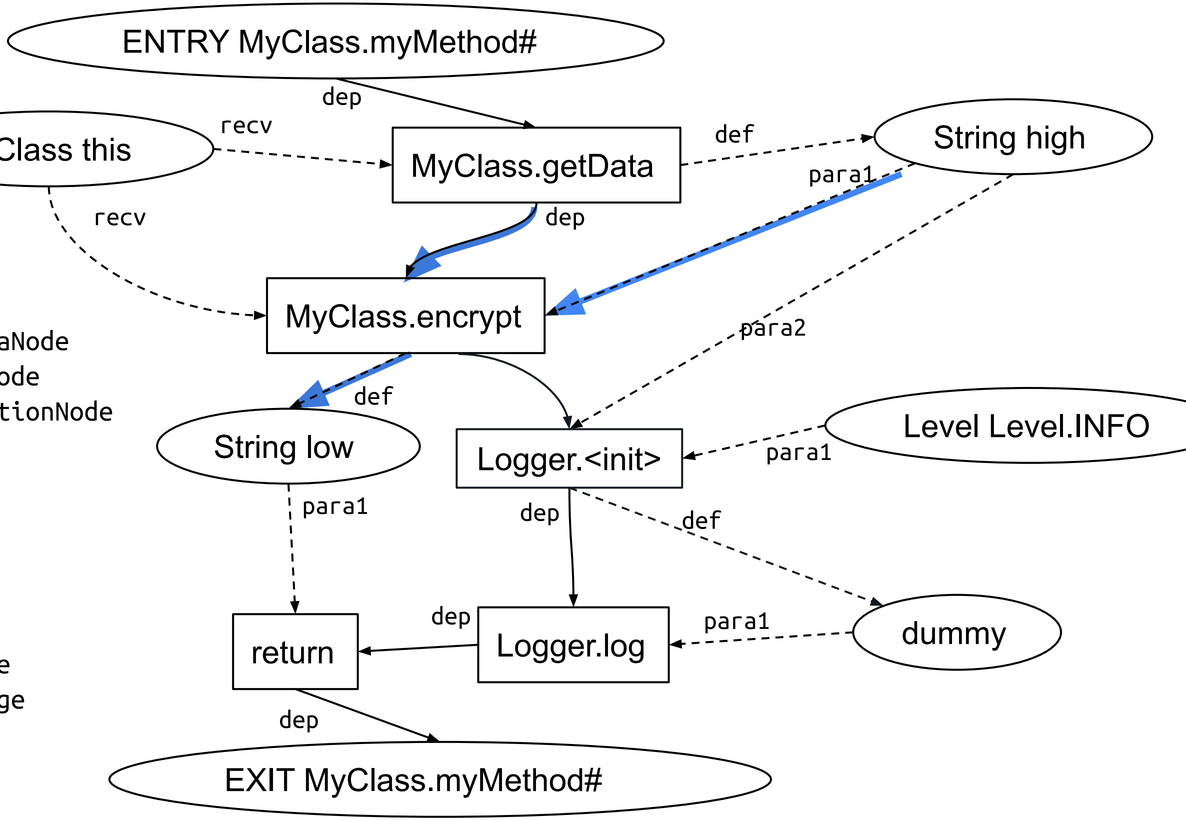
2. Datalog DFA | Groum encoding

Node.facts

```
MyClass.myMethod#  2  “MyClass this”  L:6  DataNode
MyClass.myMethod#  6  “String low”  L:8  DataNode
MyClass.myMethod#  5  “MyClass.encrypt” L:8 ActionNode
...
```

Edge.facts

```
MyClass.myMethod#  5  6 def  {}  DataFlowEdge
MyClass.myMethod#  4  5 para1  {}  DataFlowEdge
MyClass.myMethod#  3  5 dep  {}  ControlFlowEdge
...
```



Datalog: short intro

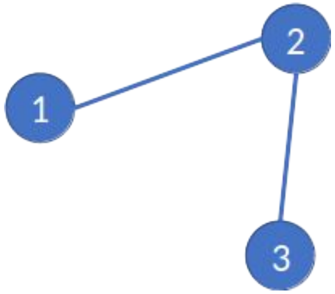
```
.decl edge(x:number, y:number)
.input edge

.decl path(x:number, y:number)
.output path

path(x, y) :- edge(x, y).
path(x, y) :- path(x, z), edge(z, y).
```

`edge.facts`

```
1        2
2        3
```

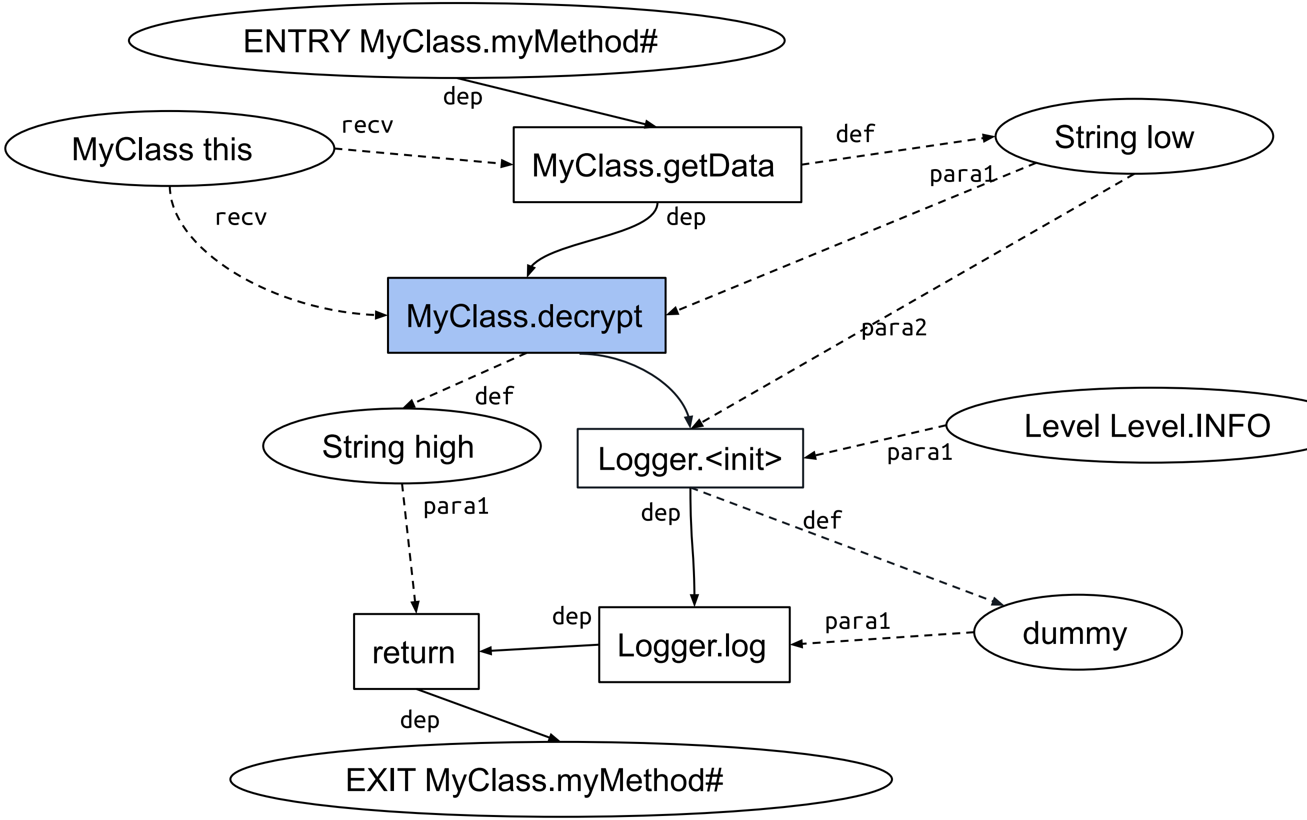
`path.csv`

```
1        2
2        3
1        3
```



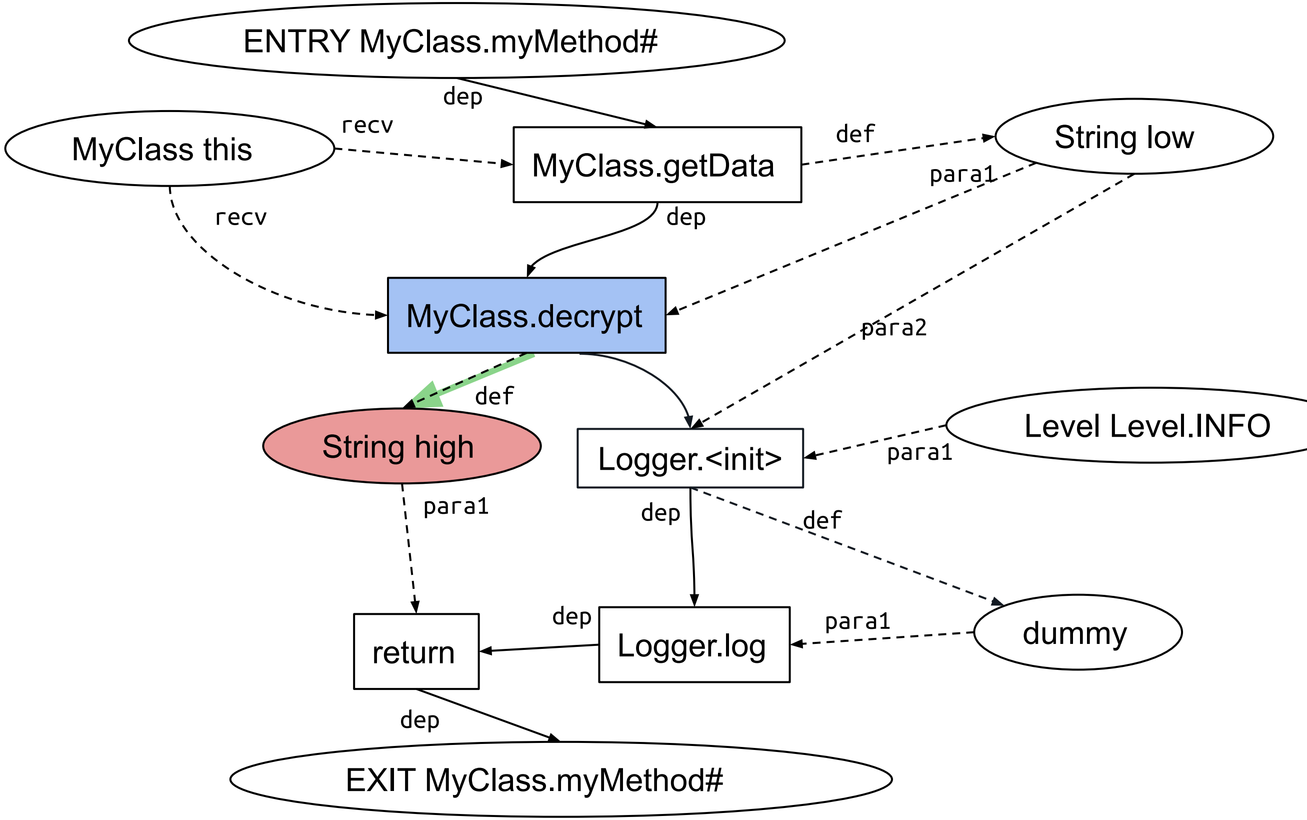
2. Datalog DFA | Initial data annotation

```
secret = decrypt(...)
```



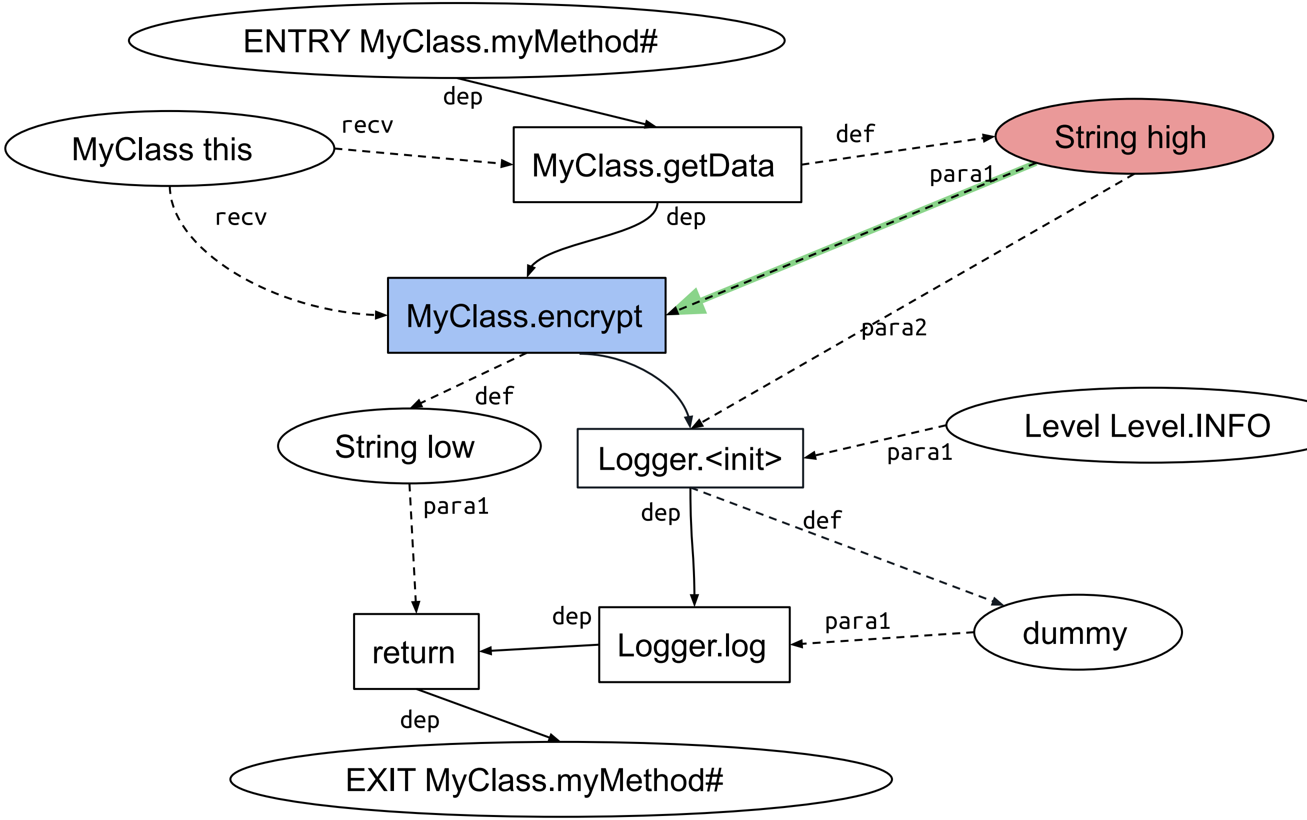
2. Datalog DFA | Initial data annotation

```
secret = decrypt(...)
```



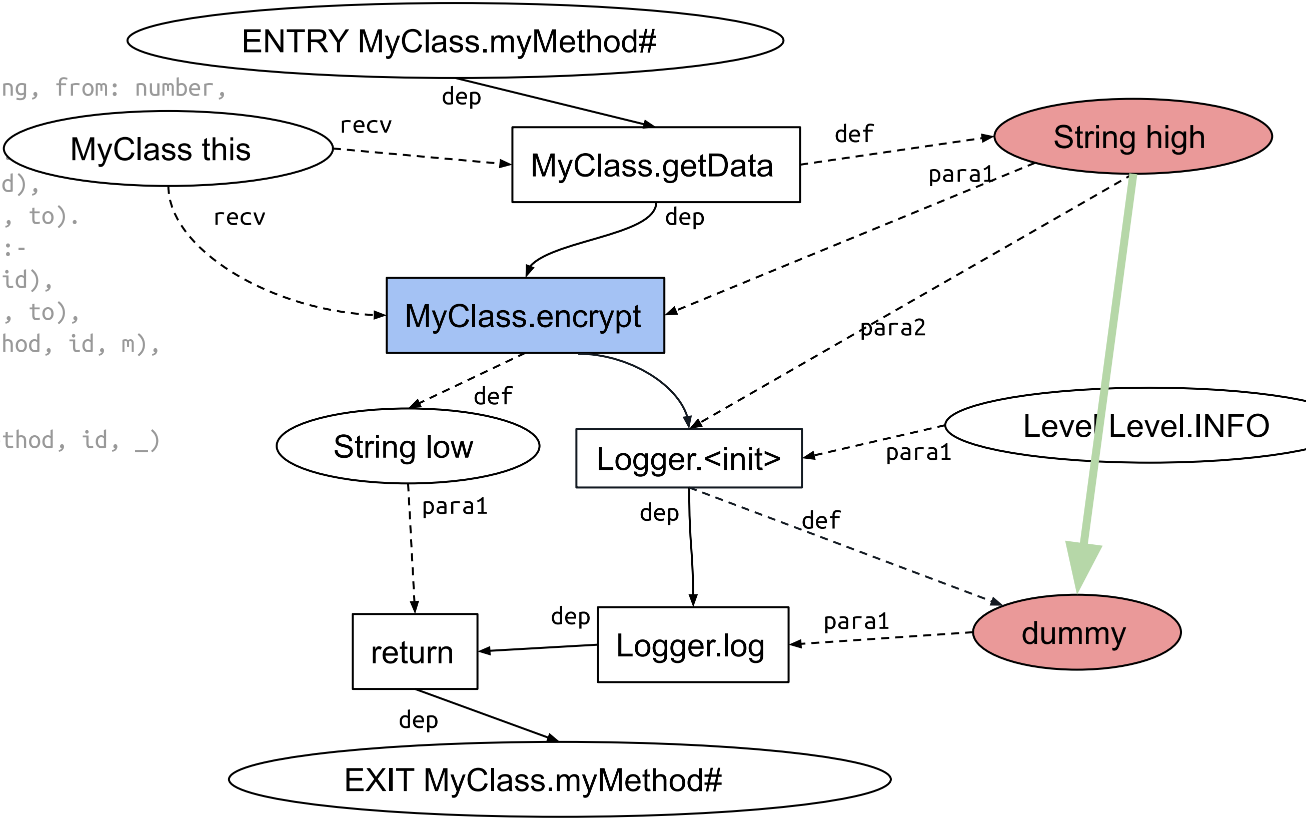
2. Datalog DFA | Initial data annotation

```
encrypt(secret)
```



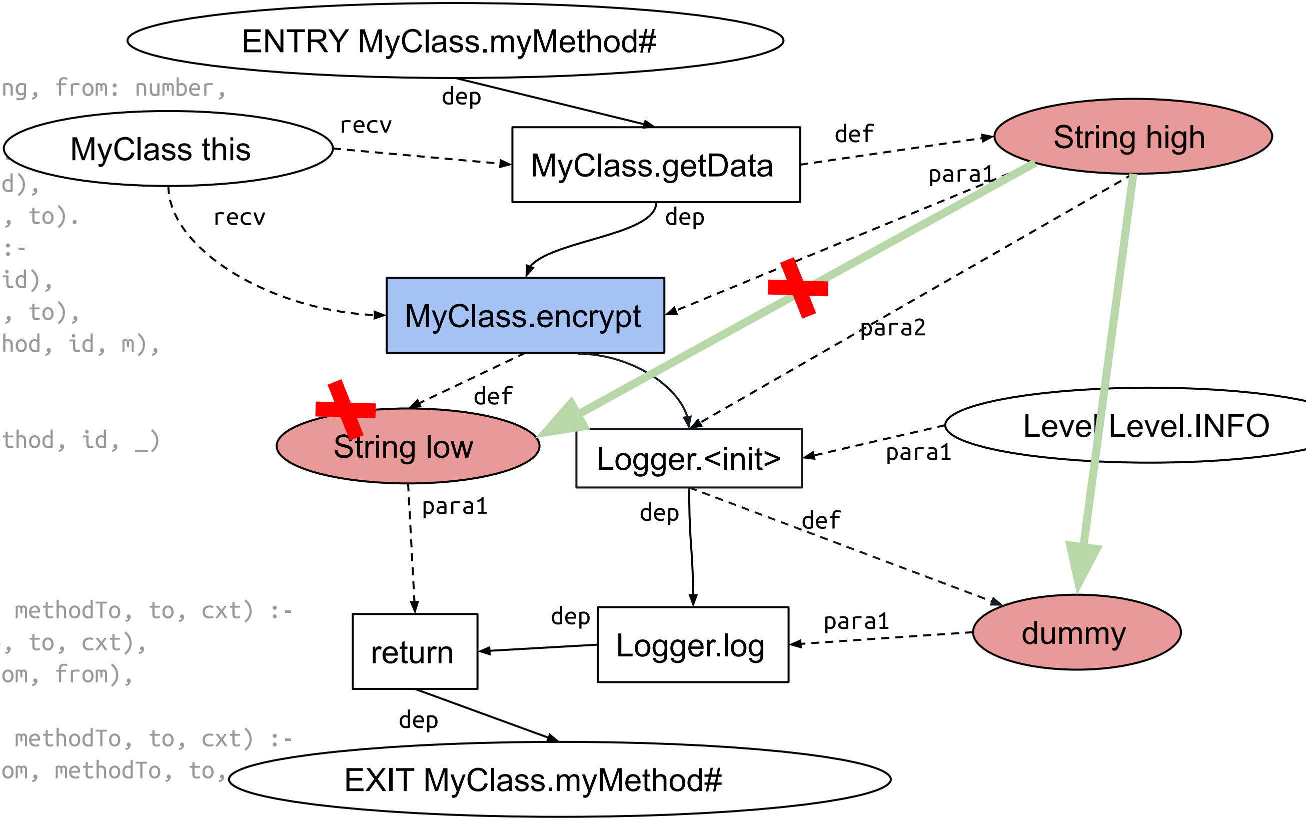
2. Datalog DFA | Annotation propagation

```
.decl EGroumIntraDataFlowEdge(method: String, from: number,
to: number)
EGroumIntraDataFlowEdge(method, from, to) :-
EGroumReceiverDataFlowEdge(method, from, id),
   EGroumDefinitionDataFlowEdge(method, id, to).
EGroumIntraDataFlowEdge(method, from, to) :-
EGroumParameterDataFlowEdge(method, from, id),
   EGroumDefinitionDataFlowEdge(method, id, to),
   ( (EGroumMethodInvocationActionNode(method, id, m),
      Method(g),
      !contains(g, m));
      !EGroumMethodInvocationActionNode(method, id, _)
   ).
...
```



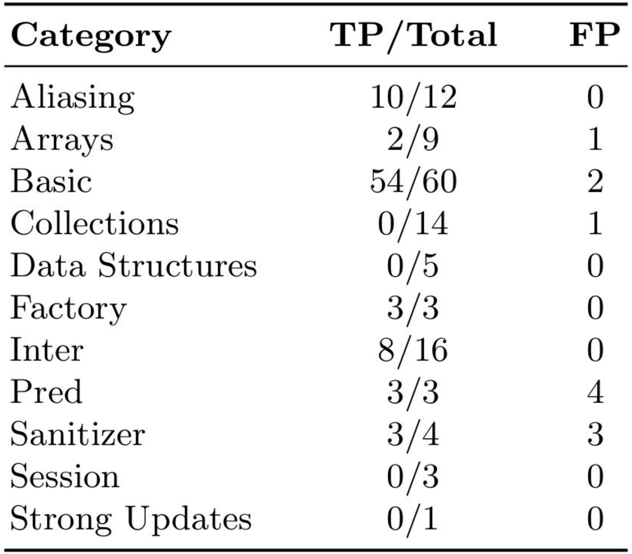
2. Datalog DFA | Annotation propagation

```
.decl EGroumIntraDataFlowEdge(method: String, from: number,
to: number)
EGroumIntraDataFlowEdge(method, from, to)
EGroumReceiverDataFlowEdge(method, from, id),
   EGroumDefinitionDataFlowEdge(method, id, to).
EGroumIntraDataFlowEdge(method, from, to) :-
EGroumParameterDataFlowEdge(method, from, id),
   EGroumDefinitionDataFlowEdge(method, id, to),
   ( (EGroumMethodInvocationActionNode(method, id, m),
      Method(g),
      !contains(g, m));
      !EGroumMethodInvocationActionNode(method, id, _)
   ).
...

.decl ConfidentialDataFlowPath(…)
ConfidentialDataFlowPath(methodFrom, from, methodTo, to, cxt) :-
   DataFlowPath(methodFrom, from, methodTo, to, cxt),
   ConfidentialVarsFromMethodData(methodFrom, from),
   !IsDeclassified(methodTo, to).
ConfidentialDataFlowPath(methodFrom, from, methodTo, to, cxt) :-
   ConfidentialDataFlowEdge(methodFrom, from, methodTo, to,
   !IsDeclassified(methodTo, to).
...
```



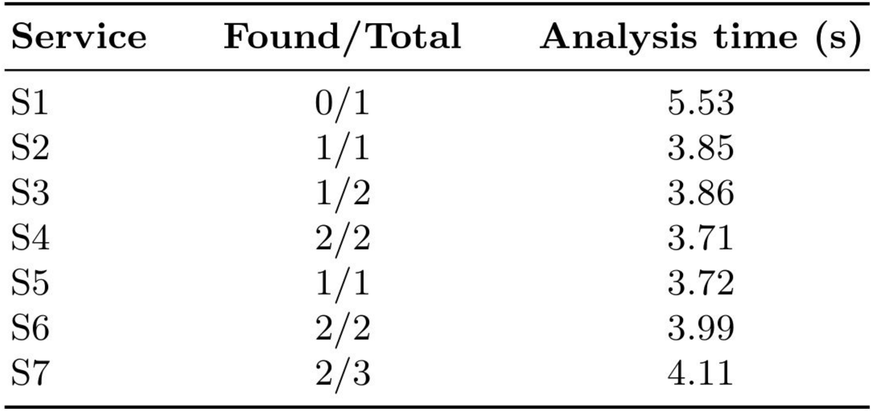
Evaluation

SecuriBench-micro benchmark



Category	TP/Total	FP
Aliasing	10/12	0
Arrays	2/9	1
Basic	54/60	2
Collections	0/14	1
Data Structures	0/5	0
Factory	3/3	0
Inter	8/16	0
Pred	3/3	4
Sanitizer	3/4	3
Session	0/3	0
Strong Updates	0/1	0

Amazon annotated code bases

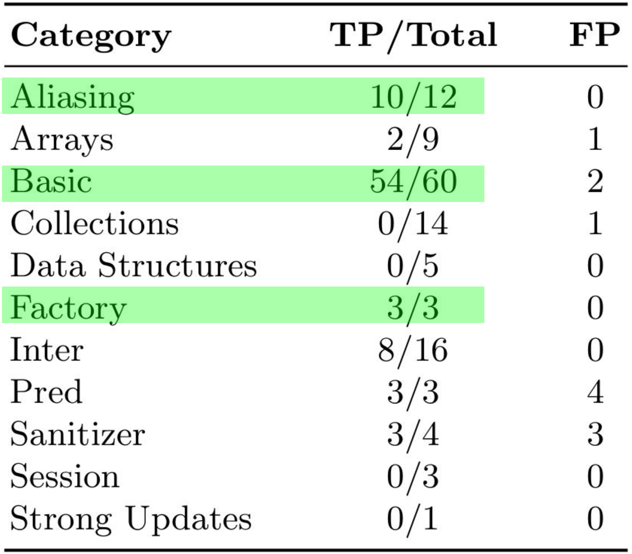


Service	Found/Total	Analysis time (s)
S1	0/1	5.53
S2	1/1	3.85
S3	1/2	3.86
S4	2/2	3.71
S5	1/1	3.72
S6	2/2	3.99
S7	2/3	4.11

9/12

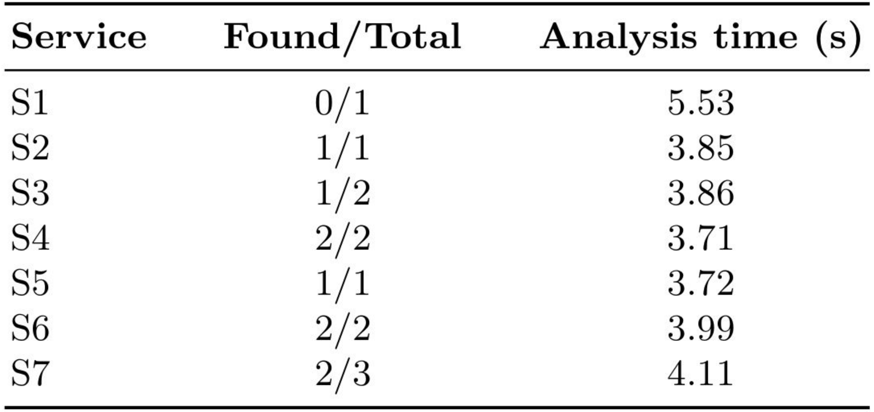
Evaluation: Promising results

SecuriBench-micro benchmark



Category	TP/Total	FP
Aliasing	10/12	0
Arrays	2/9	1
Basic	54/60	2
Collections	0/14	1
Data Structures	0/5	0
Factory	3/3	0
Inter	8/16	0
Pred	3/3	4
Sanitizer	3/4	3
Session	0/3	0
Strong Updates	0/1	0

Amazon annotated code bases



Service	Found/Total	Analysis time (s)
S1	0/1	5.53
S2	1/1	3.85
S3	1/2	3.86
S4	2/2	3.71
S5	1/1	3.72
S6	2/2	3.99
S7	2/3	4.11

9/12

Other features and limitations

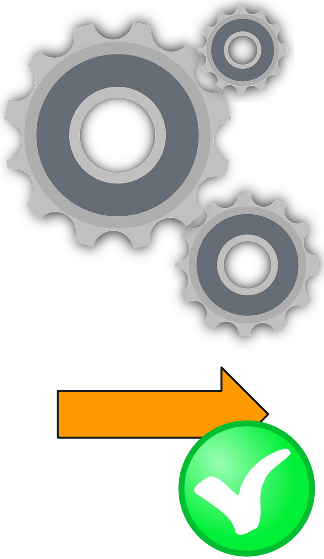
+ inter-procedural analysis

- arrays
- class fields
- step 3
- backwards analysis
- ...

```
public String myMethod() {
  String high1 = getData();
                       secret
  String high2 =          high1;
  String low = encrypt(high2);
  log(Level.INFO, high2);
  return low;
}
```

Conclusion

```
public String myMethod() {
  String high = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```



```
public String myMethod() {

  String high        = getData();
  String low = encrypt(high);
  log(Level.INFO, high);
  return low;
}
```