DIY:
Java Static Analysis

#sonarqube
#sonarlint

Nicolas PERU - @benzonico

sonarsource

Ego boost

- Nicolas PERU - @benzonico
  - Java developer@SonarSource
  - Developer in language team
  - Geneva Jug enthusiast
  - Cycle around the Leman

Sonar Java Plugin

- Back Story

sonarsource

Challenge

Get the language.

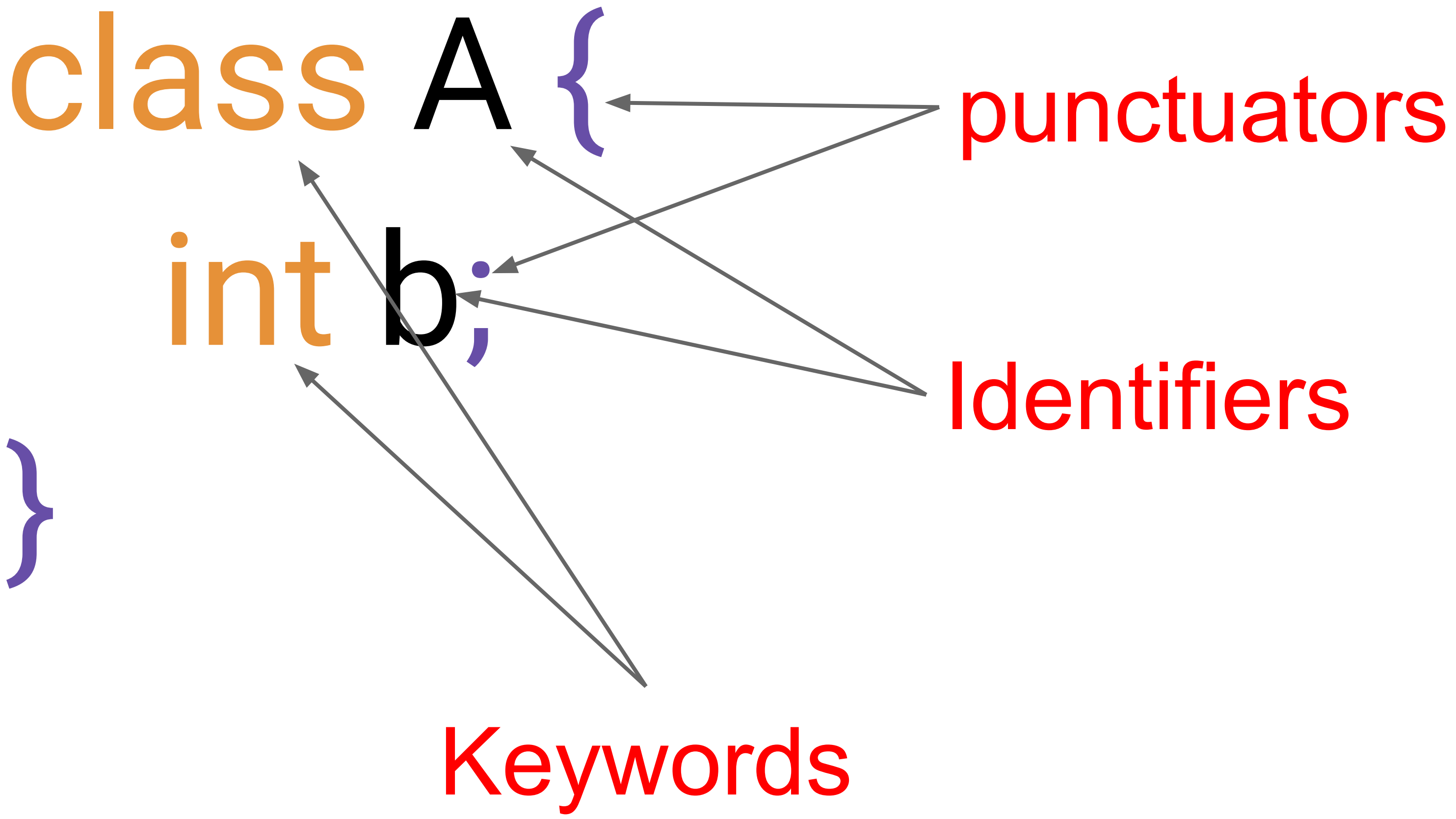
```
import java.io.*;
import java.net.*;
import java.security.*;
import protection;
public class client {
  public void sendAuthentication(String us
      OutputStream outStream) throws IOExce
    OutputStream out = new DataOutpu
    DataOutputStream out = new Date()).getTime();
    long t1 = (new Date()).getTime();
    double q1 = Math.random();
    byte[] protected1 = Protection.ma
    long t2 = (new Date()).getTime()
    double q2 = Math.random();
    byte[] protected2 = Protection.
    out.writeUTF(user);
    out.writeInt(protected1.lengt
    out.write(protected2);
    out.flush();
 }
public static void main(St
    String host = args[0];
    int port = 7999;
    String user = "John";
    String password = "Shr
    Socket s = new Socke

    client client = new
    client.sendAuthent
```

sonarsource

Lexical Analysis

```
class A {
    int b;
}
```

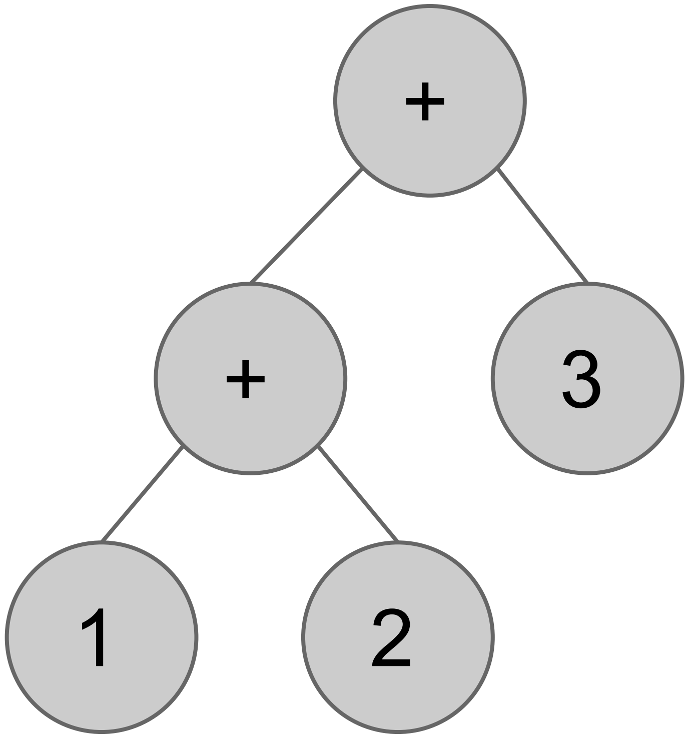
Syntactic Analysis



```
class A {
  int b;
}
```

Syntax Tree

1 + 2 + 3



```
interface BinaryExpressionTree {

 ExpressionTree leftOperand();

 SyntaxToken operatorToken();

 ExpressionTree rightOperand();

}
```

Semantic Analysis

```
class A {
  int b;
  A(int b) {
    this.b = b;
  }
}
```

sonarsource

Your turn now : Custom rules !

Beyond Semantic: Symbolic Execution

Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```

Beyond Semantic: Symbolic Execution

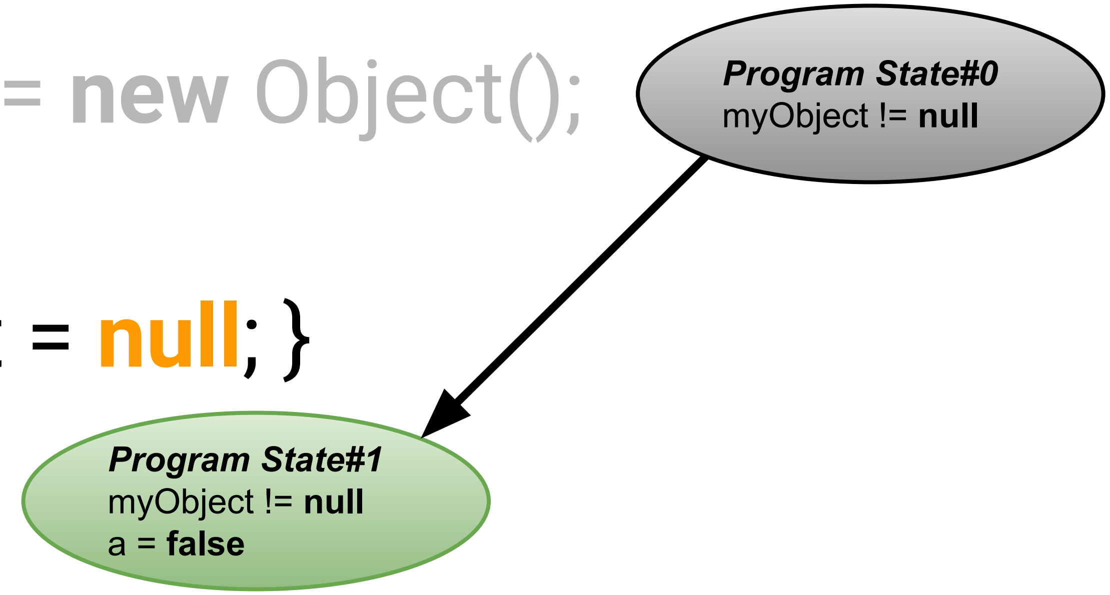
```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```

Program State#0
myObject != **null**

sonarsource

Beyond Semantic: Symbolic Execution

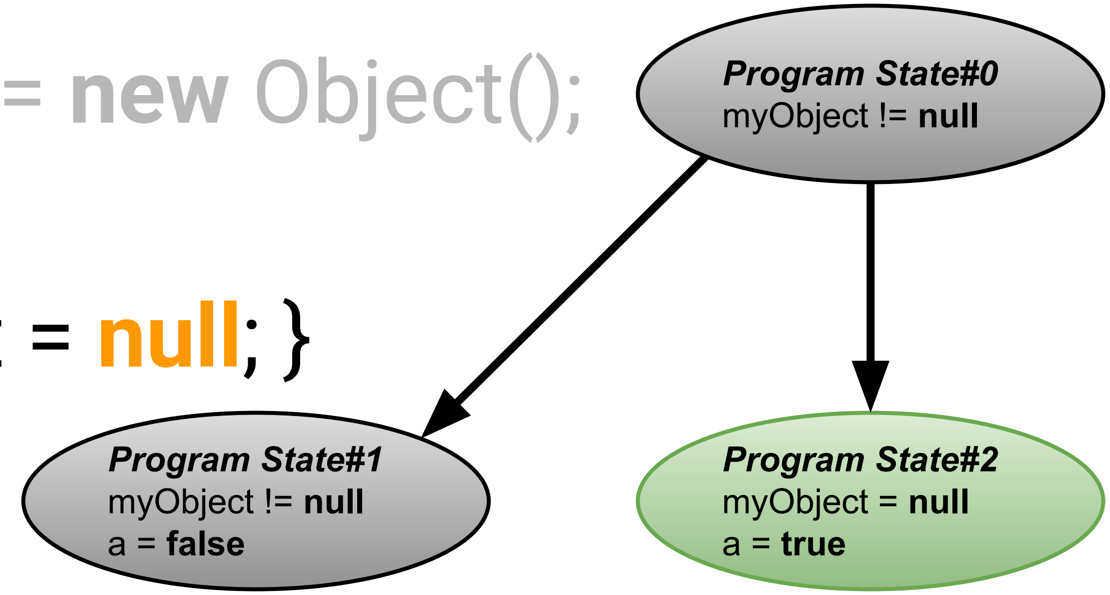
```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



sonarsource

Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



Beyond Semantic: Symbolic Execution

```
Object myObject =                t();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```

Program State#1
myObject != **null**
a = **false**

Program State#2
myObject = **null**
a = **true**

sonarsource

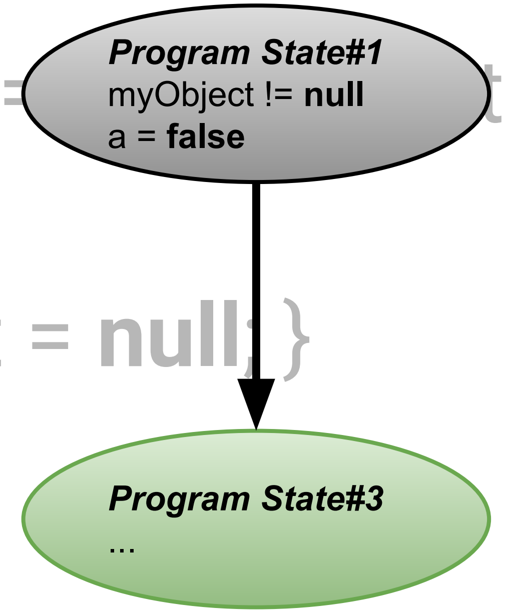
Beyond Semantic: Symbolic Execution

Program State#1
myObject != **null**
a = **false**

```
Object myObject =                t();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```

Beyond Semantic: Symbolic Execution

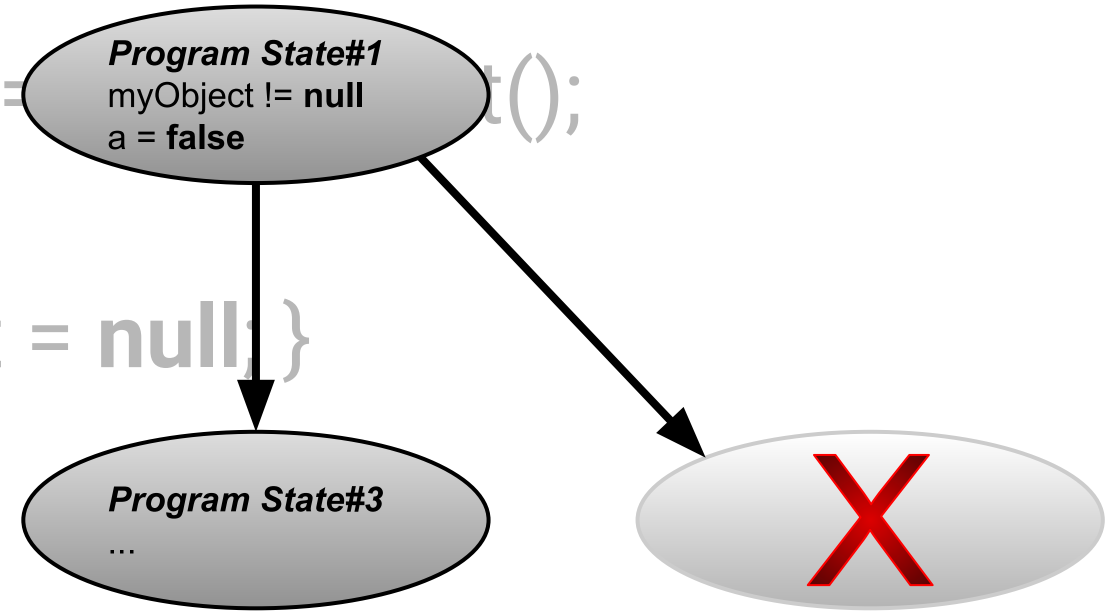
```
Object myObject =                         t();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



sonarsource

Beyond Semantic: Symbolic Execution

```
Object myObject =                        t();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



sonarsource

Beyond Semantic: Symbolic Execution

```
Object myObject =                    t();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```

Program State#1
myObject != **null**
a = **false**

Program State#2
myObject = **null**
a = **true**

Beyond Semantic: Symbolic Execution

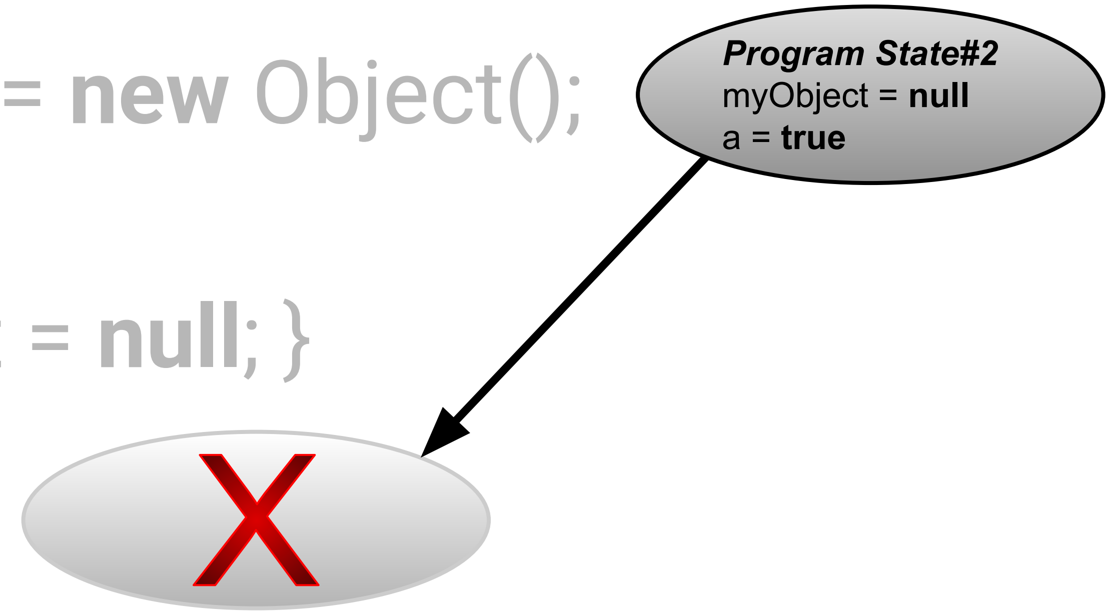
```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
   myObject.toString();
}
```

Program State#2
myObject = **null**
a = **true**

sonarsource

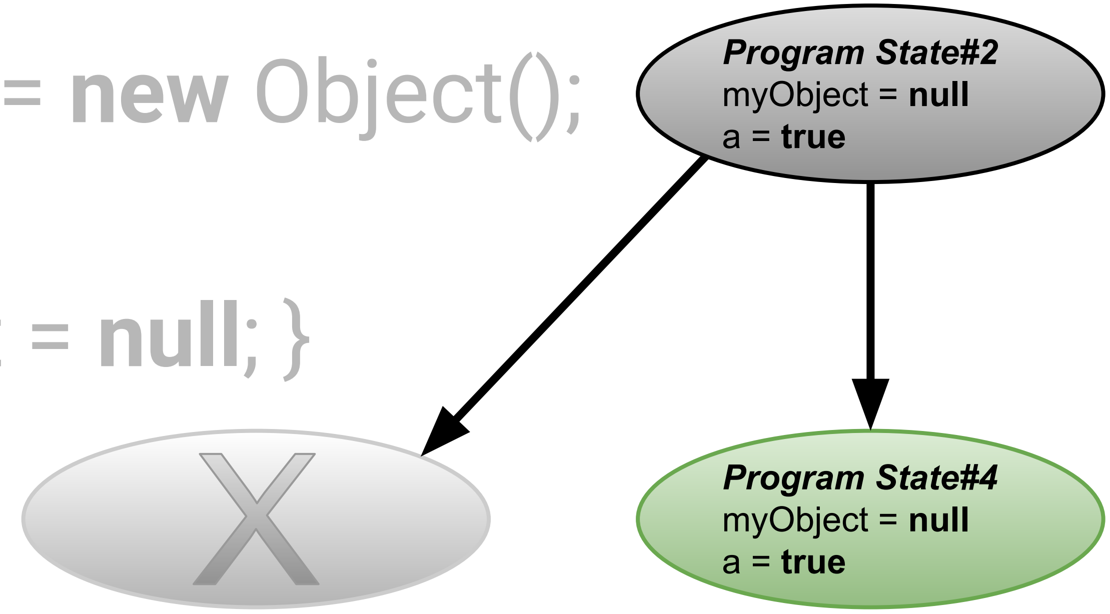
Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



Beyond Semantic: Symbolic Execution

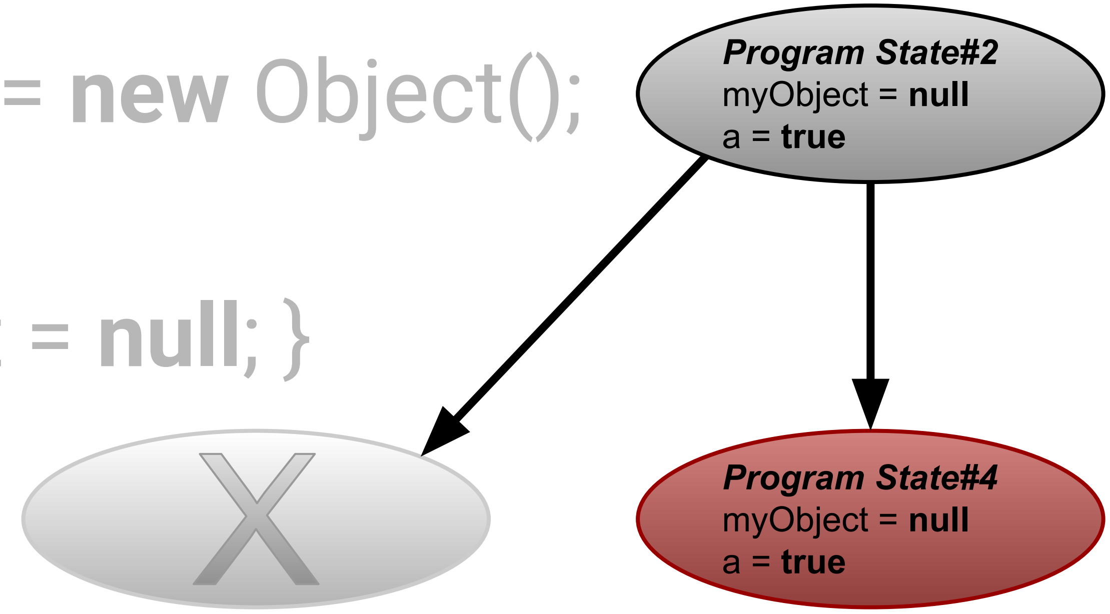
```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString();
}
```



sonarsource

Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
// ...
if ( a ) { myObject = null; }
// ...
if ( !a ) { /* ... */ }
else {
    myObject.toString(); // NPE
}
```



sonarsource

Symbolic Execution challenges

- Complex conditions

```
if (a + 1 < (b * 10 - 39) ) {
  if ( b > a / 10 + 4 ) { … } // Always true
}
```

sonarsource™

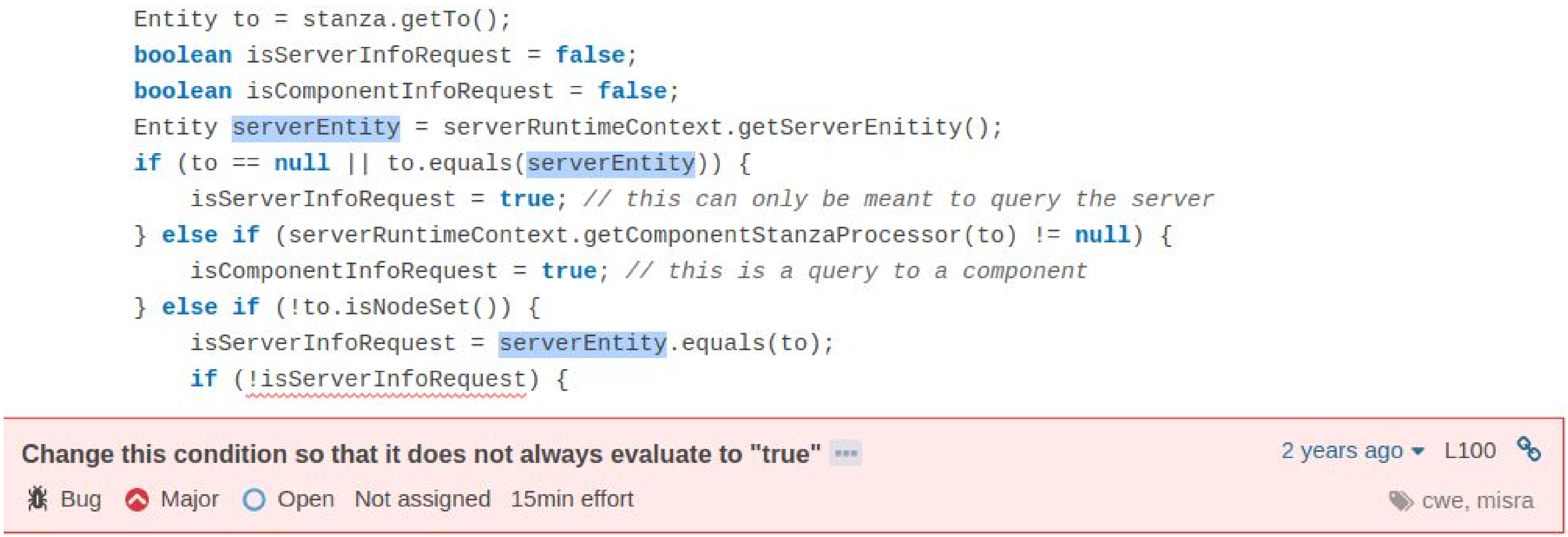
Symbolic Execution challenges

- Complex conditions

```
if (a + 1 < (b * 10 - 39) ) {
  if ( b > a / 10 + 4 ) { … } // Always true
}
```

- Explosion of states

Uhoh ?!



```
Entity to = stanza.getTo();
boolean isServerInfoRequest = false;
boolean isComponentInfoRequest = false;
Entity serverEntity = serverRuntimeContext.getServerEnitity();
if (to == null || to.equals(serverEntity)) {
    isServerInfoRequest = true; // this can only be meant to query the server
} else if (serverRuntimeContext.getComponentStanzaProcessor(to) != null) {
    isComponentInfoRequest = true; // this is a query to a component
} else if (!to.isNodeSet()) {
    isServerInfoRequest = serverEntity.equals(to);
    if (!isServerInfoRequest) {
```

Change this condition so that it does not always evaluate to "true"

2 years ago L100

Bug Major Open Not assigned 15min effort

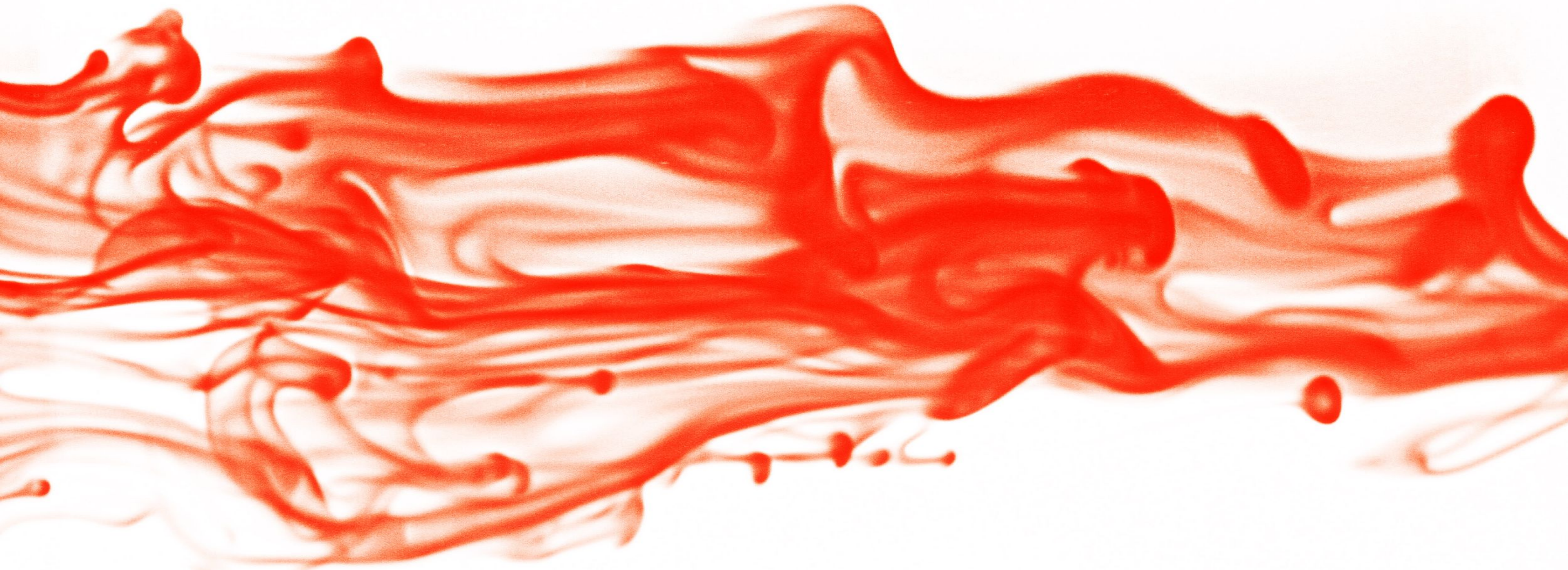
cwe, misra

From apache vysper

sonarsource

What’s next ?

Taint Analysis for vulnerabilities



jobs@sonarsource.com