SEMANTIC ANALYSIS

- Catch bugs
- Catch security issues

```
752  });
753
754  function elementExtractor(tagName, type) {
755    return new RegExp(
756        `<${tagName} [^>]*['"]${type}['"][^>]*>([\\s\\S]+?)</${tagName}>`,
```

This regular expression is constructed from a `[ 2 Values ]`.

`user-provided value`

`user-provided value`

```
757        'gm');
758  }
759
```

SEMANTIC ANALYSIS

- Can dive across layers and layers of code to find weird paths developers are not going to see
- Can be adapted to many languages and detect errors like deserialization, unsafe usage of user params, etc

JavaConverter.java

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

UnsafeDeserialization.ql

```
from DataFlow::PathNode source, DataFlow::PathNode
    sink, UnsafeDeserializationConfig conf
where conf.hasFlowPath(source, sink)
select sink.getNode().(UnsafeDeserializationSink)
    .getMethodAccess(),
  source, sink, "Unsafe deserialization of $@.",
source.getNode(), "user input"
```

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