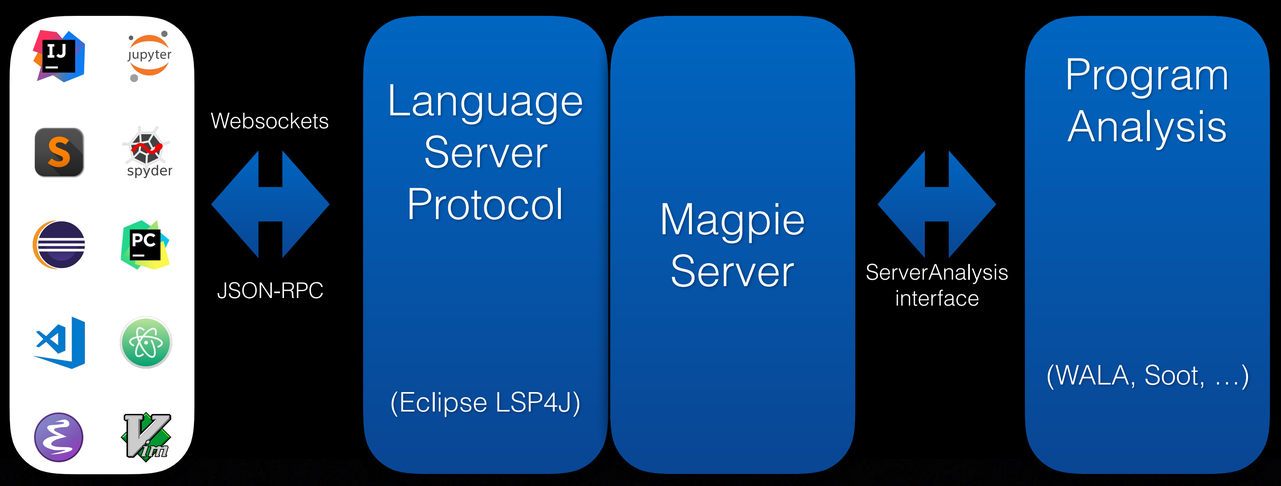
How to Use MagpieBridge:
A General Approach to Integrating Static Analyses into IDEs and Editors

https://github.com/MagpieBridge



Additional MagpieBridge Functionalities:

- IR converter
  - adapt WALA Java IR to Soot IR
  - use WALA's precise source map
  - similar 3-address IRs
- IntelliJ LSP client
  - support popular IDE
  - Java-based
  - supports related information

Actual JavaScript Taint Analysis IDE Integration Code

Basic analysis code looks as usual, with little change to be suitable for the IDE.
Code taken directly from WALA tutorial given in 2016*.

This analysis finds tainted flows using graph reachability on a flow- and context-sensitive system dependence graph.

1. Add analysis to MagpieServer using the `ServerAnalysis` interface

Analysis specifies applicable language

Analysis given arbitrary name for IDE

```
public class CreateSDGForJavaScript {

    public static MagpieServer bridge() {
        MagpieServer bridge = new MagpieServer();
        bridge.addAnalysis("js", new ServerAnalysis() {
            @Override
            public String source() {
                return "JSTaintDemo";
            }

            @Override
            public void analyze(Collection<? extends Module> files, MagpieServer server) {
                try {
Collection<Module> m2 = HashSetFactory.make();
files.forEach((m) -> m2.add(m));
m2.add(JSCallGraphBuilderUtil.getPrologueFile("prologue.js"));
Module[] modules = m2.toArray(new Module[files.size()]);
JavaScriptLoaderFactory loaders = new JavaScriptLoaderFactory(new CAstRhinoTranslatorFactory());
JSCFABuilder builder = JSCallGraphBuilderUtil.makeCGBuilder(loaders, modules, CGBuilderType.ONE_CFA, AstIRFactory.makeDefaultFactory());
CallGraph CG = builder.makeCallGraph(builder.getOptions());
SDG<InstanceKey> SDG = new SDG<InstanceKey>(CG, builder.getPointerAnalysis(), new JavaScriptModRef<InstanceKey>(), DataDependenceOptions.NO_BASE_NO_HEAP_NO_EXCEPTIONS, ControlDependenceOptions.NONE);
Set<List<Statement>> paths = Analysis.getPaths(SDG, Analysis.documentUrlSource, Analysis.documentWriteSink);
Set<AnalysisResult> results = HashSetFactory.make();
paths.forEach((path) ->  {
    List<AnalysisResult> pr = new LinkedList<>();
    for(int i = 0; i < path.size(); i++) {
        int idx = i;
        Statement last = path.get(idx);
        if (Print.getPosition(last) == null) {
            continue;
        }
        pr.add(new AnalysisResult() {
            @Override
            public Kind kind() {
                return Kind.Diagnostic;
            }

            @Override
            public String toString(boolean useMarkdown) {
                if (isSink(pr)) {
                    return "tainted sink";
                } else if (isSource(pr)) {
                    return "tainted source";
                } else {
                    return "tainted flow step";
                }
            }

            private boolean isSource(List<AnalysisResult> pr) {
                return pr.indexOf(this) == pr.size()-1;
            }

            private boolean isSink(List<AnalysisResult> pr) {
                return pr.indexOf(this) == 0;
            }


            @Override
            public Position position() {
                return Print.getPosition(last);
            }

            @Override
            public Iterable<Pair<Position, String>> related() {
                List<Pair<Position, String>> info = new LinkedList<>();
                path.forEach((p) -> {
                    try {
                        Position pos = Print.getPosition(p);
                        if (pos != null) {
                            String text = new SourceBuffer(pos).toString();
                            if (text != null) {
                                info.add(Pair.make(pos, text.trim()));
                            }
                        }
                    } catch (IOException e) {
                        // ignore error
                    }
                });
                Collections.reverse(info);
                return info;
            }

            @Override
            public DiagnosticSeverity severity() {
                return
                    isSink(pr)?
                       DiagnosticSeverity.Error:
                       isSource(pr)?
                          DiagnosticSeverity.Warning:
                          DiagnosticSeverity.Information;
            }

            @Override
            public Pair<Position, String> repair() {
                // not implemented stub
                return null;
            }

            @Override
            public String code() {
                try {
                    return new SourceBuffer(position()).toString();
                } catch (IOException e) {
                    return "unknown";
                }
            }

        });
    }
    results.addAll(pr);
});

                server.consume(results, "JSTaintDemo");
            } catch (WalaException | IllegalArgumentException | CancelException e) {
                MessageParams mp = new MessageParams();
                mp.setType(MessageType.Error);
                mp.setMessage(e.toString());
                server.getClient().showMessage(mp);
            }
        }

    });
    return bridge;
}

public static void main(String[] args) throws IOException, WalaException, CancelException {
    com.ibm.wala.cast.js.ipa.callgraph.JSCallGraphUtil.setTranslatorFactory(new CAstRhinoTranslatorFactory());
    MagpieServer bridge = bridge();
    bridge.launchOnStdio();
}
}
```

2. Adapt results using the `AnalysisResult` interface

`kind` controls how results are rendered: Diagnostics typically get underlines, and can show in any problems view

`toString` denotes the message to show, and can use markdown for some IDEs

`position` denotes where to show the message. vital that analysis frameworks be precise and accurate

`related`: analysis specifies arbitrary program items that it deems related to an issue. in this case, that is a complete tainted flow

`severity`: analysis specifies issue severity, which often controls rendered color (e.g. red for error)

`repair` denotes strings for repair by straightforward text substitution at precise code position

`code` denotes the problematic source code detected by the analysis

3. Launch MagpieServer

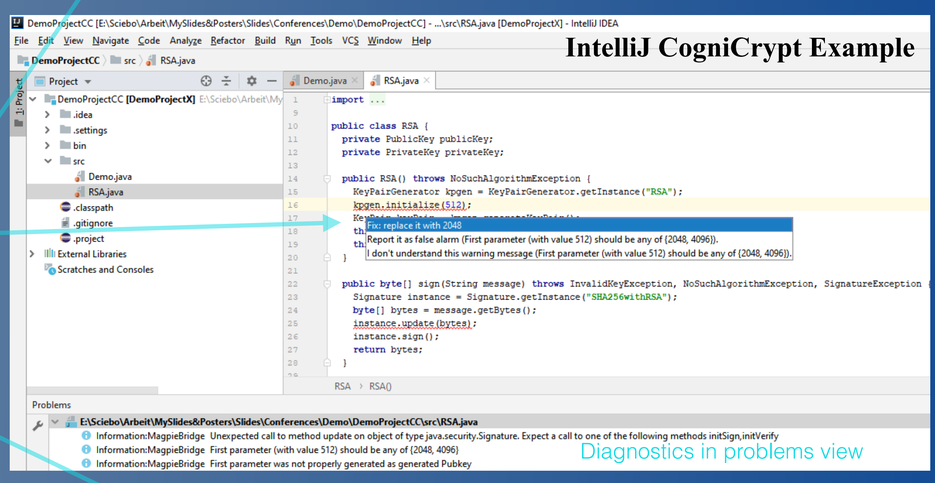
Monaco JS Taint Example

```
1  var document = { URL: "whatever",
2    write: function Document_prototype_write(x) { } };
3  var id = function _id(x) { return x; };
4  function Id() { this.id = id; }
5  function SubId() { }; SubId.prototype = new Id();
6
7  if (Math.random.call(null) > 0) {
8      var id1 = new Id();
9
10 [JSTaintDemo] tainted sink
11
12   • temp8535751275671584956.js(9, 15): document.URL
13   • temp8535751275671584956.js(10, 16): id1.id.call(document, url)
14   • temp8535751275671584956.js(3, 28): return x;
15   • temp8535751275671584956.js(15, 1): document.write(text)
16 document.write(text);
```

Monaco Python Tensors Example

```
32 def conv_net(x_dict, n_classes, dropout, reuse, is_training):
33     # Define a scope for reusing the variables
34     with tf.variable_scope('ConvNet', reuse=reuse):
35         # TF Estimator input is a dict, in case of multiple inputs
36         xxx = x_dict['images']
37
38         bad_x = tf.reshape(xxx, shape=[-1, 11, 28, 1])
39
40         # MNIST data input is a 1-D vector of 784 features (28*28 pixels)
41         # Reshape to match picture format [Height x Width x Channel]
42         # Tensor input become 4-D: [Batch Size, Height, Width, Channel]
43         z = tf.reshape(xxx, shape=[-1, 28, 28, 1])
44
45         # Convolution Layer with 32 filters and a kernel size of 5
46         conv1 = tf.layers.conv2d(z, 32, 5, activation=tf.nn.relu)
47         # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
48         conv1 = tf.layers.max_pooling2d(conv1, 2, 2)
49
50         # Convolution Layer with 64 filters and a kernel size of 3
51         conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
52         # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
53         conv2 = tf.layers.max_pooling2d(conv2, 2, 2)
54
55         bad_conv1 = tf.layers.conv2d(xxx, 32, 5, activation=tf.nn.relu)
56
57 Fix: replace it with tf.reshape(xxx, [-1, 28, 28, 1])
58
59 Report it as false alarm (Bad type to convolve pixel[n][28 * 28], needs 4 dimensions (possible fix: tf.reshape(xxx, [-1, 28, 28, 1]))).
```

IntelliJ CogniCrypt Example



Diagnostics in problems view

Visual Studio Code FlowDroid Example


```
@ServerEndpoint("/websocket-js")
public class JSTaintExampleWebsocket extends MagpieWebsocketServer {

    public JSTaintExampleWebsocket() {
        super(() -> CreateSDGForJavaScript.bridge());
        System.err.println("started server");
    }

}
```

Websockets allows MagpieBridge to communicate with web-based systems such as Microsoft Monaco. MagpieBridge deploys seamlessly inside an Apache Tomcat server.

*cited tutorial:https://conf.researchr.org/track/pldi-2016/Tutorials

Linghui Luo (Paderborn University) Julian Dolby (IBM Thomas J. Watson Research Center) Eric Bodden (Paderborn University & Fraunhofer IEM)