Soot Tutorial

Sunday, March 03, 2019 2:06 PM

Download soot jar file. sootclasses-trunk-jar-with-dependencies.jar

Create a new Java project in Eclipse.

Write an Analysis.java file.

```
import soot.Pack;
import soot.PackManager;
import soot.Scene;
import soot.SootClass;
import soot.SootResolver;
import soot.Transform;
import soot.options.Options;

public class Analysis {
	public static void main(String[] args) {
		Options.v().setPhaseOption("jb", "use-original-names:true");
		Scene.v().getSootClassPath();
		Scene.v().extendSootClassPath("C:\\eclipsews\\practice\\sootTutorial\\bin");
		System.out.println(Scene.v().getSootClassPath());
		Pack jtp=PackManager.v().getPack("jtp");
		jtp.add(new Transform("jtp.instrumenter", new AnalysisWrapper()));
		SootResolver.v().resolveClass("java.lang.CloneNotSupportedException",
		SootClass.SIGNATURES);
		Options.v().set_output_format(Options.output_format_jimple);
		soot.Main.main(args);
	}
}
```

Now, let us create another Wrapper file which will have our analysis.

```
import java.util.Iterator;
import java.util.Map;

import soot.Body;
import soot.BodyTransformer;
import soot.SootMethod;
import soot.Unit;
import soot.toolkits.graph.BriefUnitGraph;
import soot.toolkits.graph.UnitGraph;

public class AnalysisWrapper extends BodyTransformer {

	@Override
	protected void internalTransform(Body body, String phase, Map options) {
		// TODO Auto-generated method stub
		SootMethod sootMethod = body.getMethod();
		UnitGraph g = new BriefUnitGraph(sootMethod.getActiveBody());
		Iterator unitIt = g.iterator();
```

```
            while (unitIt.hasNext()) {
                Unit s = (Unit) unitIt.next();
                System.out.println(s.toString());
            }

        }
}
```

Provide a Test.java on which we will conduct the analysis.

```
public class Test {
    public static void main(String[] args)
    {
        String str = "CMI";
        int x,y;
        StringBuilder sb = new StringBuilder();

        for(int i = str.length() - 1; i >= 0; i--)
        {
          if (i == 0) sb.append('S');
          sb.append(str.charAt(i));
          x = 2;
        }
        System.out.println(sb.toString());
    }
}
```

Pass "Test" as argument while running your code.

Note that the output is in Jimple format. To understand, open and see the Jimple file from the source/sootoutput folder.

You may also try traversing through the CFG.

```
import java.util.Iterator;
import java.util.Map;

import polyglot.ast.Stmt;
import soot.Body;
import soot.BodyTransformer;
import soot.SootMethod;
import soot.Unit;
import soot.ValueBox;
import soot.toolkits.graph.BriefUnitGraph;
import soot.toolkits.graph.UnitGraph;

public class AnalysisWrapper extends BodyTransformer {

    @Override
    protected void internalTransform(Body body, String phase, Map options) {
        // TODO Auto-generated method stub
        SootMethod sootMethod = body.getMethod();
```

```
            UnitGraph g = new BriefUnitGraph(sootMethod.getActiveBody());
            Iterator unitIt = g.iterator();
            while (unitIt.hasNext()) {
                  soot.jimple.Stmt s = (soot.jimple.Stmt) unitIt.next();
                  System.out.println(s.toString());
                  /*if (!s.getDefBoxes().isEmpty()) {
                        Iterator defIt = s.getDefBoxes().iterator();
                        while(defIt.hasNext())
                        {
                              ValueBox defBox = (ValueBox)defIt.next();
                              System.out.println(defBox.getValue().toString());
                        }
                  }*/

            }

      }
}
```