AppSec USA
2014
Denver, Colorado

From the ground up

Source-Sink tracing in the JVM

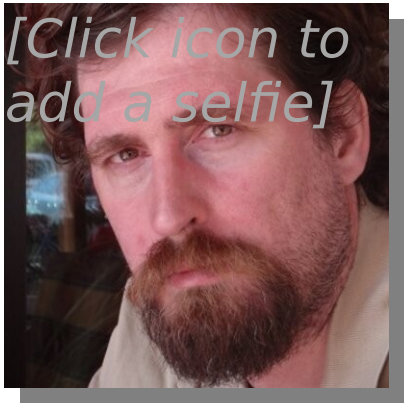
Steven van der Baan - @vdbaan

Me

- Steven van der Baan
- Dutch
- Lives in the UK
- Works @ 7Safe

& OWASP

(according to my wife)



History

- Appsec EU 2013 – Hamburg

- Source-Sink tracking
- Evil Bit

Jim Manico & Dinis Cruz

Basic Idea

- Every String that enters an application will get an 'Evil Bit' set.
- Every exit where that 'Evil String' leaves an application gets a warning.

- The application must not be changed to achieve this.
- Intended for testers (Staging).

Technologies

- Bytecode manipulation (Javassist)
  - Using “-Xbootclasspath:/p”
- AOP
- Using Javaagent (like Jrebel)
  - Using “-javaagent=<jar>”

  or
- Combination

ZAP policy

- “It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing as well as being a useful addition to an experienced pen testers toolbox.”

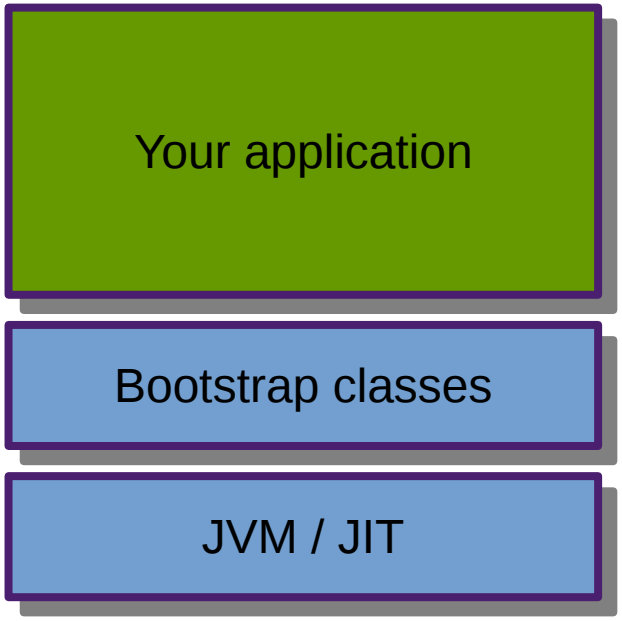
Basic Idea

```java
public class Example {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("What is your name?");
        String name = br.readLine();
        System.out.println("Hello to you, "+name);
    }
}
```

```
steven@ATH0:/tmp$ java org.owasp.securt.Example
What is your name?
Steven
Hello to you, Steven
steven@ATH0:/tmp$
```

```
steven@ATH0:/tmp$ printf "\x08\x08\x08\x08\x08\x08\x08\x08to our Hacker" \
> | java org.owasp.securt.Example
What is your name?
Hello to our Hacker
steven@ATH0:/tmp$
```

JVM

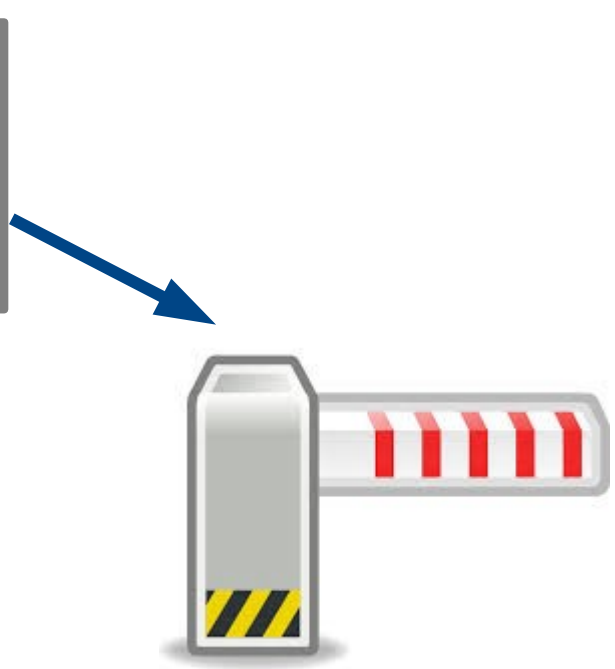


The 'evil' String

RFC3514 Compliant

Tracing

Your application



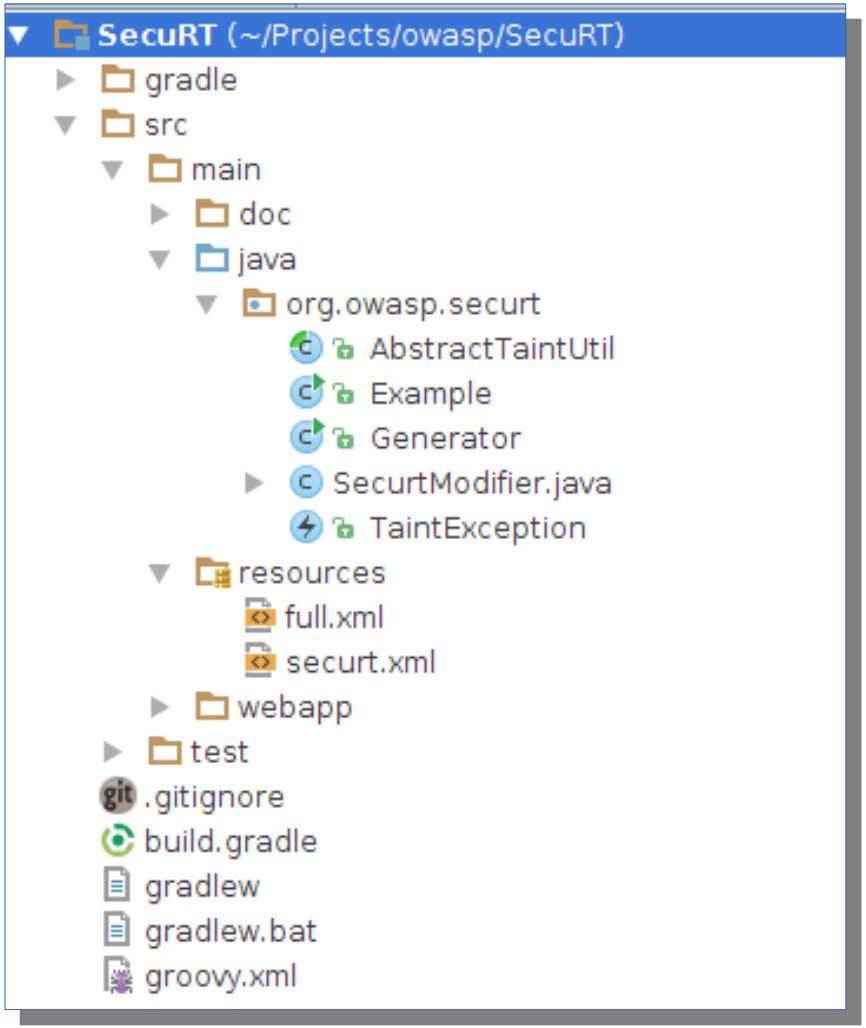
WARNING

TAINTING

Tainting

- Basic classes
  - Bootstrap classes
    - String
    - StringBuilder
    - ClassLoader
- Intercepting classes
  - All classes are loaded and instantiated by the ClassLoader.
    - Applies also to classes that are loaded through reflection.

Overview


SecuRT (~/Projects/owasp/SecuRT)
gradle
src
main
doc
java
org.owasp.securt
AbstractTaintUtil
Example
Generator
SecurtModifier.java
TaintException
resources
full.xml
securt.xml
webapp
test
.gitignore
build.gradle
gradlew
gradlew.bat
groovy.xml

Building

- Staged build process
  - Create project classes
  - Manipulate base classes
    - String, Stringbuilder, ClassLoader
  - Generate support classes
    - Logging, marking & checking, etc
  - Create JAR

Preparing java.lang.String

```
private void modifyString(String destPath) throws NotFoundException, CannotCompileException, IOException {
    ClassPool cp = ClassPool.getDefault();
    CtClass cc = cp.get("java.lang.String");

    CtField tainted = CtField.make("private boolean tainted;", cc);
    cc.addField(tainted);
    cc.addMethod(CtNewMethod.getter("isTainted", tainted));
    cc.addMethod(CtNewMethod.setter("setTaint", tainted));

    CtField trace = CtField.make("private StackTraceElement[] trace;", cc);
    cc.addField(trace);
    cc.addMethod(CtNewMethod.getter("getTrace", trace));
    cc.addMethod(CtNewMethod.setter("setTrace", trace));

    cc.writeFile(destPath);
    org.owasp.securt.AbstractTaintUtil.debug("Adapted: " + cc.getName());
}
```

Preparing java.lang.ClassLoader

```
private void changeClassLoader(String destPath) throws NotFoundException, CannotCompileException, IOException {
    ClassPool cp = ClassPool.getDefault();
    CtClass cc = cp.get("java.lang.ClassLoader");
    for (CtMethod method : cc.getDeclaredMethods()) {
        if (method.getName().equals("defineClass")) {
            if (method.getParameterTypes().length == 5 && method.getParameterTypes()[1].isArray()) {
                wrapMethod(cc, method);
            }
        }
    }

    CtMethod m = cc.getDeclaredMethod("defineClass",
            arguments("java.lang.String,java.nio.ByteBuffer,java.security.ProtectionDomain"));
    m.setName("wrappedDefineClass");
    cc.addMethod(CtMethod.make(
            "protected final Class defineClass(String name, java.nio.ByteBuffer b," +
                                              "java.security.ProtectionDomain protectionDomain) {"
                    + "    return wrappedDefineClass(name,b,protectionDomain);"
                    + "}", cc));
    cc.writeFile(destPath);
    org.owasp.securt.AbstractTaintUtil.debug("Adapted: " + cc.getName());
}
```

Wrapping method

```
private void wrapMethod(CtClass clazz, CtMethod method) throws NotFoundException, CannotCompileException {
    method.setName("wrappedDefineClass");
    CtMethod wrapper = CtNewMethod.make(Modifier.PROTECTED, method.getReturnType(), "defineClass",
                       method.getParameterTypes(), method.getExceptionTypes(), null, clazz);
    String code = "{"
            + " if (!$1.startsWith(\"javassist\") &&"
            // System classes
              + "     !$1.startsWith(\"org.apache.\") &&"
            + "     !$1.startsWith(\"org.junit.\") &&"
            + "     !$1.startsWith(\"sun.\") &&"
            + "     !$1.startsWith(\"java.\") &&"
              + "     !$1.startsWith(\"javax.\") &&"
            + "     !$1.startsWith(\"com.sun.\")) "
            + "{"
            + " org.owasp.securt.AbstractTaintUtil.debug(\"Checking: \"+$1);"
            + "      byte[] newBytes = org.owasp.securt.SecurtModifier.translate($1, $0, $2) ;"
            + "      if (newBytes != (byte[])null) {"
            + "         return wrappedDefineClass($1, newBytes, 0, newBytes.length, $5); "
            + "      }}"
            + "  return wrappedDefineClass($1, $2, $3, $4, $5); "
            + "  "
            + "}";
    wrapper.setBody(code);
    clazz.addMethod(wrapper);
}
```

Actual wrapping

```
public static byte[] translate(String className, ClassLoader loader, byte[] classfileBuffer)
        throws IOException, CannotCompileException {
    byte[] result = classfileBuffer;
    ClassPool cp = getClassPool();

    try {
        CtClass cc = cp.get(className);
        if (cc.isInterface())
            return result;

        if(classes.containsKey(cc.getName()))
            modify(classes,cc.getName(),cc);
        for (CtClass ctc : cc.getInterfaces()) {
            if (interfaces.containsKey(ctc.getName())) {
                AbstractTaintUtil.debug("Will have to change this class: " + className);
                modify(interfaces, ctc.getName(), cc);
                result = cc.toBytecode();
                AbstractTaintUtil.info("changed from " + classfileBuffer.length + " to " + result.length);
            }
        }
        CtClass object = cp.get("java.lang.Object");
        CtClass parent = cc;
        do {
            parent = parent.getSuperclass();
            if (abstracts.containsKey(parent.getName())) {
                AbstractTaintUtil.debug("Will have to change this class: " + className);
                modify(abstracts, parent.getName(), cc);
            }
        } while (parent != object);

        result = cc.toBytecode();
    } catch (NotFoundException e) {
        AbstractTaintUtil.debug(className +" is not in ClassPool");
    }

    return result;
}
```

- If class needs wrapping
  - Check interfaces of class against XML
  - Check parents of class against XML
- XML contains classes and methods which are a source or sink

XML

```
<securt>
    <sources>
        <source class="java.io.BufferedReader" method="readLine" arguments="boolean"/>
        <source class="java.io.RandomAccessFile">
            <method name="readUTF"/>
            <method name="readLine"/>
        </source>

    </sources>
    <sinks>
        <sink class="java.io.PrintStream">
            <method name="print" arguments="java.lang.String" vulnerable="1"/>
            <method name="println" arguments="java.lang.String" vulnerable="1"/>
        </sink>
        <sink class="java.io.PrintWriter">
            <method name="print" arguments="java.lang.String" vulnerable="1"/>
            <method name="println" arguments="java.lang.String" vulnerable="1"/>
        </sink>

        <sink class="java.io.DataOutput">
            <method name="write" arguments="int" vulnerable="1"/>
        </sink>
    </sinks>
    <runtime>
        <interfaces>
            <interface type="sink" class="java.sql.Statement" method="execute" arguments="java.lang.String" vulnerable="1"/>
            <interface type="sink" class="java.sql.Statement" method="executeQuery" arguments="java.lang.String" vulnerable="1"/>
            <interface type="source" class="javax.servlet.ServletRequest" method="getParameter" arguments="java.lang.String" vulnerable="1"/>
            <interface type="source" class="javax.servlet.http.HttpServletRequest" method="getParameter" arguments="java.lang.String" vulnerable="1"/>
        </interfaces>
        <abstracts>
            <abstract type="sink" class="javax.servlet.jsp.JspWriter">
                <method name="print" arguments="java.lang.String" vulnerable="1"/>
                <method name="println" arguments="java.lang.String" vulnerable="1"/>
            </abstract>
        </abstracts>
        <classes>
            <class type="source" class="org.codehaus.groovy.runtime.DefaultGroovyMethods"...>
            <class type="source" class="org.codehaus.groovy.runtime.typehandling.ShortTypeHandling"...>
        </classes>
    </runtime>
</securt>
```


PoC

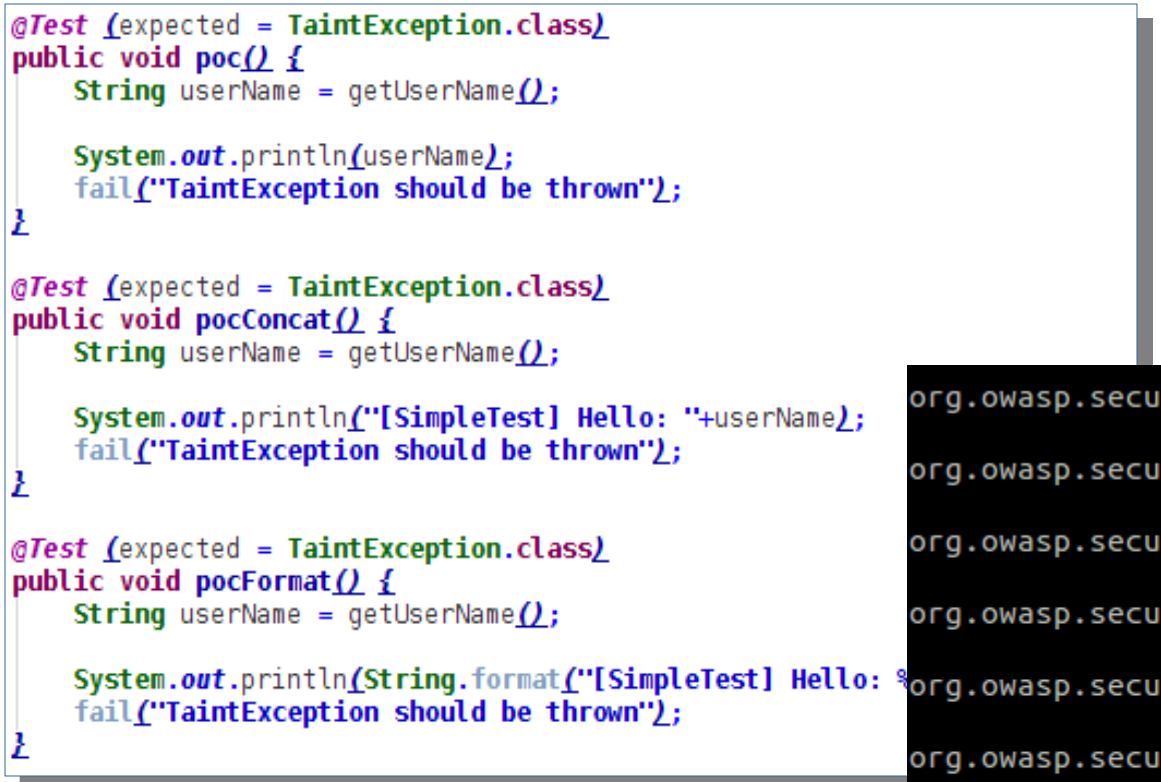
PoC - Setup

```
<securt>
    <sources>
        <source class="java.io.BufferedReader" method="readLine" arguments="boolean"/>
        <source class="java.io.RandomAccessFile">
            <method name="readUTF"/>
            <method name="readLine"/>
        </source>
```

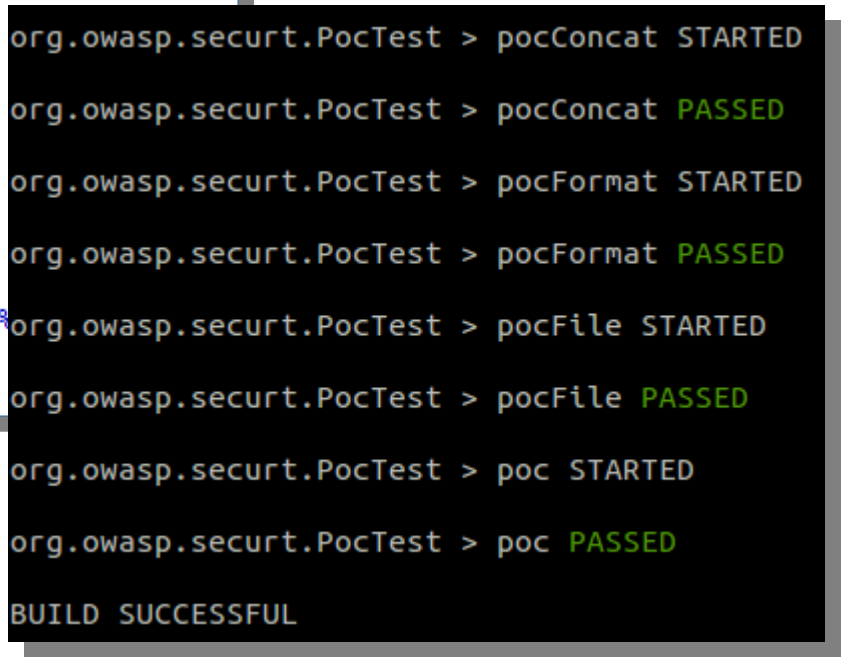
```
private String getUserName() {
    String userName = null;
    BufferedReader br = new BufferedReader(new StringReader("tainted String"));
    try {
        userName = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return userName;
}
```

```
test {
    jvmArgs = [
            "-Xbootclasspath/p:" + sourceSets.main.runtimeClasspath.getAsPath()
            ,"-DTHROW_EXCEPTION=true"
            ,"-DLOG_EXCEPTIONS=false"
            ,"-DSECURT_LOGLEVEL=debug"
    ].toList()

    testLogging {
        // Show that tests are running in the command-line output
        events 'started', 'passed', 'failed'
    }
}
```

Proofs on Concepts



```java
@Test (expected = TaintException.class)
public void poc() {
    String userName = getUserName();

    System.out.println(userName);
    fail("TaintException should be thrown");
}

@Test (expected = TaintException.class)
public void pocConcat() {
    String userName = getUserName();

    System.out.println("[SimpleTest] Hello: "+userName);
    fail("TaintException should be thrown");
}

@Test (expected = TaintException.class)
public void pocFormat() {
    String userName = getUserName();

    System.out.println(String.format("[SimpleTest] Hello: %
    fail("TaintException should be thrown");
}
```



```
org.owasp.securt.PocTest > pocConcat STARTED

org.owasp.securt.PocTest > pocConcat PASSED

org.owasp.securt.PocTest > pocFormat STARTED

org.owasp.securt.PocTest > pocFormat PASSED

org.owasp.securt.PocTest > pocFile STARTED

org.owasp.securt.PocTest > pocFile PASSED

org.owasp.securt.PocTest > poc STARTED

org.owasp.securt.PocTest > poc PASSED

BUILD SUCCESSFUL
```


SQL

PoC SQL

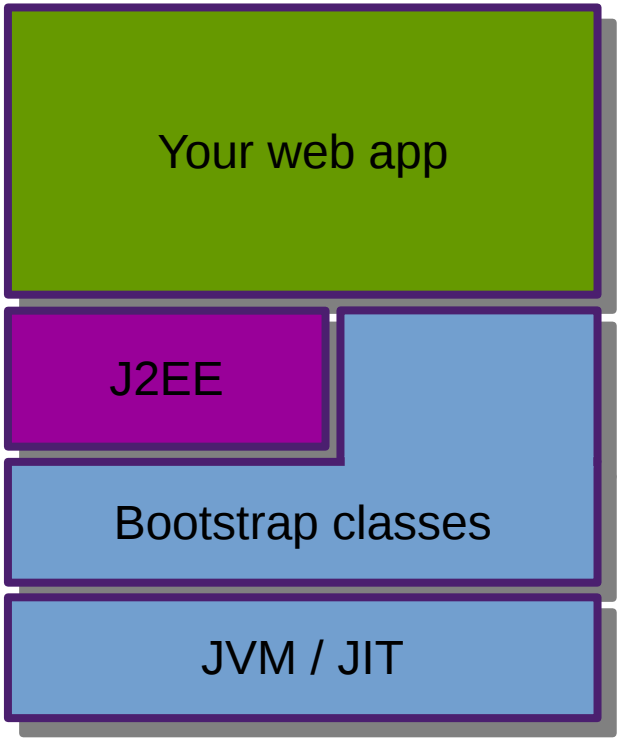
```
<runtime>
    <interfaces>
        <interface type="sink" class="java.sql.Statement" method="execute" arguments="java.lang.String" vulnerable="1"/>
        <interface type="sink" class="java.sql.Statement" method="executeQuery" arguments="java.lang.String" vulnerable="1"/>
```

```
@Test (expected = TaintException.class)
public void testSQL() {
    String sql = "SELECT * FROM contacts WHERE name='"+getUserName()+"'";
    String createSQL = "create table contacts (name varchar(45),email varchar(45),phone varchar(45))";

    Connection connection;
    try {
        Class.forName("org.hsqldb.jdbcDriver");
        connection = DriverManager.getConnection("jdbc:hsqldb:mem:mymemdb", "SA", "");
        connection.createStatement().executeUpdate(createSQL);

        Statement statement = null;
        ResultSet resultSet = null;

        statement = connection.createStatement();
        // it should fail here :)
        resultSet = statement.executeQuery(sql);

        resultSet.close();
        statement.close();
        connection.close();
        fail("Should not get here");
    } catch (SQLException e) {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        e.printStackTrace(System.err);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

```
org.owasp.securt.SQLTest > testSQL STARTED

org.owasp.securt.SQLTest > testSQL PASSED

BUILD SUCCESSFUL
```


J2EE

J2EE



J2EE Test Case

```java
@Test
public void testServlet() {
    request.setMethod("GET");
    request.setVersion("HTTP/1.0");
    request.setURI("/servlet/?yourName=my+servlet");

    try {
        response.parse(tester.getResponses(request.generate()));
        assertTrue(response.getMethod() == null);
        assertEquals(200, response.getStatus());
        assertEquals("\n\n<html>\n<body>\nValue of input is : my servlet\n</body>\n</html>\n",
                response.getContent());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
org.owasp.securt.WebTest > testName STARTED

org.owasp.securt.WebTest > testName FAILED
    junit.framework.AssertionFailedError at WebTest.java:64

org.owasp.securt.WebTest > testServlet STARTED

org.owasp.securt.WebTest > testServlet PASSED

2 tests completed, 1 failed
:test FAILED
```

For now only servlets and not JSP's :(

Output J2EE test

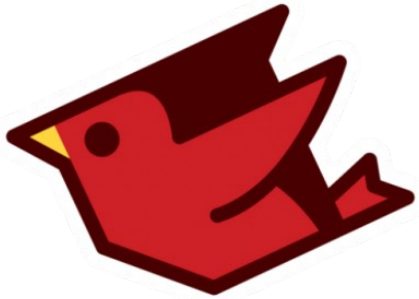
```
@Before
public void setUp() {
    tester.setResourceBase("./src/main/webapp/jsp");
    tester.addServlet(JspServlet.class, "*.jsp");
    tester.addServlet(textInput.class, "/servlet/*");
    try {
        tester.start();
    } catch (Exception e) {
        e.printStackTrace();
    }
    AbstractTaintUtil.setThrowException(false);
}
```

```
+<testcase name="testName" classname="org.owasp.securt.WebTest" time="2.181"></testcase>
  <testcase name="testServlet" classname="org.owasp.securt.WebTest" time="0.223"/>
+<testcase name="testHelloWorld" classname="org.owasp.securt.WebTest" time="0.224"></testcase>
  <system-out>[*] Throwing exception?false Taint detected </system-out>
+<system-err></system-err>
</testsuite>
```

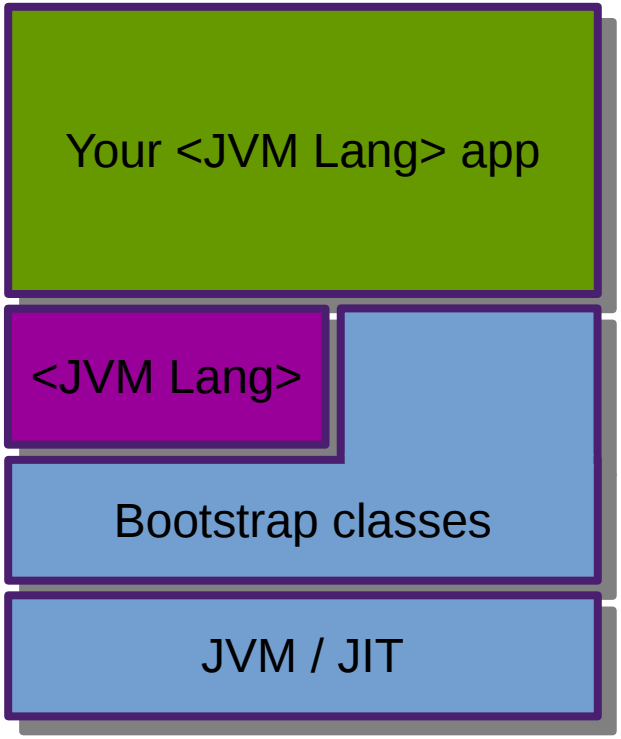
JVM Languages

Other JVM Languages

- Groovy
- Scala
- Jython
- Jruby
- Rhino



From a JVM perspective



Groovy

```groovy
// Groovy PoC
String fileContents = new File('src/test/resources/file.txt').text
println "The contents of the file is: " +fileContents
```

```
<classes>
    <class type="source" class="org.codehaus.groovy.runtime.DefaultGroovyMethods">
        <method name="getText" arguments="java.io.File"/>
    </class>
```

Groovyc...

```
(String)ShortTypeHandling.castToString(arrayOfCallSite[1].callGetProperty(arrayOfCallSite[2].callConstructor(File.class, "src/test/resources/file.txt")));
```

```
<class type="source" class="org.codehaus.groovy.runtime.typehandling.ShortTypeHandling">
    <method name="castToString" arguments="java.lang.Object"/>
</class>
```

ant -f Groovy.xml

```
<java classname="groovy.ui.GroovyMain" fork="true">
    <jvmarg value="-Xbootclasspath/p:build/distributions/SecuRT-1.0-shadow.jar"/>
    <jvmarg value="-DSECURT_LOGLEVEL=debug"/>
    <arg value="src/test/groovy/GroovyPoc.groovy"/>
    <classpath>
        <fileset dir="lib" includes="groovy-all-2.3.3.jar"/>
    </classpath>

</java>
```

```
[java] Taint detected
[java] The contents of the file is: testing123
```

Scala

```
object Poc {
  def main(args: Array[String]): Unit = {
    val source = scala.io.Source.fromFile("src/test/resources/file.txt")
    val lines = source.mkString
    source.close()
    println ("The contents of the file is: " + lines)
  }
}
```

```
<abstract type="source" class="scala.io.Source">
    <method name="mkString"/>
</abstract>
```

ant -f scala.xml

```
<java classname="Poc"
      classpathref="build.classpath" fork="true">
    <jvmarg value="-DSECURT_LOGLEVEL=debug"/>
    <jvmarg value="-Xbootclasspath/p:build/distributions/SecuRT-1.0-shadow.jar"/>
</java>
```

Jython

```
myfile = open('src/test/resources/file.txt')
lines = myfile.read()
print "The contents of the file is: %s" % lines
```

```
<abstract type="source" class="org.python.core.io.TextIOWrapper">
    <method name="read" arguments="int"/>
</abstract>
```

ant -f jython.xml

```
<java classname="org.python.util.jython" fork="true">
    <jvmarg value="-Xbootclasspath/p:build/distributions/SecuRT-1.0-shadow.jar"/>
    <arg value="src/test/jython/PythonPoc.py"/>
    <classpath>
        <fileset dir="lib" includes="jython-standalone-2.5.3.jar"/>
    </classpath>
</java>
```

Future

Future?

- JSP / JSF (as in compile on the fly)
- JRuby
- Optimize output
  - Show traces based on stacktraces
  - Combine stacktraces for joined tainted strings
- Incorporate JgraphX for visual display

Related Projects

Other

- Java Gravizapa
  - by Meder Kydyraliev
- JSR 308 (Checker framework)
- Chord

Thank you