STONESOUP Phase 3
Test Case Creation User Guide
12 December 2014

IARPA STONESOUP PHASE 3
TEST CASE CREATION USER GUIDE

Table of Contents

List of Figures

List of Tables

12 December 2014

1 Test Case

A test case for the purposes of this document is a collection of files that are encapsulated in a compressed archive that allow for the automated compilation and execution of a program with inputs and outputs defined in a metadata file.

This document is a reference guide to the required structure of test cases that can be run through the Test and Evaluation Analysis and Execution System (TEXAS) developed for the IARPA STONESOUP program.

1.1 Test Case Naming Convention

While the metadata specification captures an extremely detailed set of parameters that comprise a target test case, the content does not allow itself to be easily searched in a file system. The test case naming standard aims to capture the important parameters that describe the test case in a compressed format. The test case naming standard does not replace the information captured in the metadata, but rather provide access to the most common parameters of interest when identifying a test case archive. This information is designed to be both human readable and is used by the TEXAS system when aggregating results from the database. However, while running a test case through TEXAS the name is simply treated as a string with no special meaning, so if there is no plan to use the test case in the database then a custom naming format is plausible.

The archive containing the test case files should be named with the following naming convention if it has not had bad I/O Pairs injected into it.

```
<language>-<program name shorted to 4 characters>.tar.gz
```

For example, Wireshark which is a C program would be named the following:

```
C-WIRE.tar.gz.
```

A Java program such as Lucene would be named the following:

```
J-LUCE.tar.gz.
```

An injected archive naming convention takes on the following form in **Figure 1**.
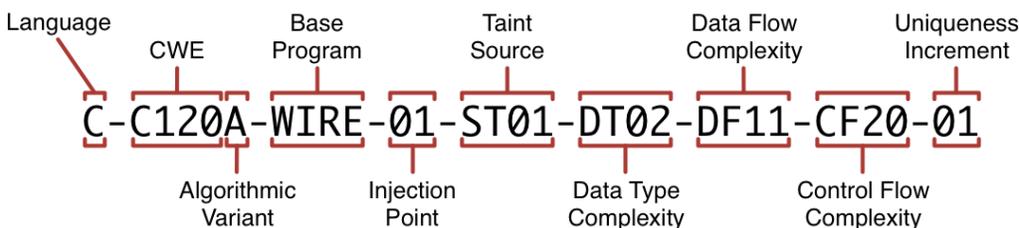
Figure 1   Test Case Naming Converntion - Example

Texas supports two program input languages. It supports either C programs or Java programs. These have been shorted to a single character for identification purposes in the naming standard.

```
C for C Programs
J for Java Programs
```

The CWE portion stands for the MITRE common weakness enumeration.

12 December 2014

The Algorithmic variants further refine a given CWE by mapping to a code snipped that has been injected into a target base program.

The Base program portion denotes the name of the given base program that has been injected into.

The injection points are defined by a two digit number and are specific to a given base program. Two or more base programs may share the same identifier however these identifiers reference difference injection points in the base programs.

Taint sources are denoted by the letters ST and are followed by a two digit number for 4 characters total. The following **Table 1   ST Identifers by Language,** defines each taint source.

Table 1   ST Identifers by Language

Identifier	Name	Java	C/Binary
01	ENVIRONMENT_VARIABLE	Yes	Yes
02	FILE_CONTENTS	Yes	Yes
03	SOCKET	Yes	Yes
04	SHARED_MEMORY	Yes	Yes

Data Type Complexity is a 4-character field and defined by a DT and a two-digit number. These two digit numbers are defined in **Table 2   DT Identifers by Language**.

Table 2   DT Identifers by Language

Identifier	Name	Java	C/Binary
01	ARRAY	Yes	Yes
02	SIMPLE	Yes	Yes
03	VOID_POINTER	Yes	Yes
04	HEAP_POINTER	No	Yes
05	STRUCT	No	Yes
06	TYPEDEF	No	Yes
07	UNION	No	Yes

Data Flow Complexities are defined with a DF identifier and a two number code as defined by **Table 3   DF Identifers by Language**.

Table 3   DF Identifers by Language

Identifier	Name	Group	Java	C/Binary
01	ADDRESS_ALIAS_1	ADDRESS_ALIAS	No	Yes
02	ADDRESS_ALIAS_2	ADDRESS_ALIAS	No	Yes
03	ADDRESS_ALIAS_10	ADDRESS_ALIAS	No	Yes
04	ADDRESS_ALIAS_50	ADDRESS_ALIAS	No	Yes
05	ADDRESS_AS_CONSTANT	N/A	Yes	Yes
06	ADDRESS_AS_FUNCTION_RETURN_VALUE	N/A	Yes	Yes
07	INDEX_ALIAS_1	INDEX_ALIAS	Yes	Yes
08	INDEX_ALIAS_2	INDEX_ALIAS	No	Yes
09	INDEX_ALIAS_10	INDEX_ALIAS	No	Yes
10	INDEX_ALIAS_50	INDEX_ALIAS	No	Yes
11	BASIC	N/A	Yes	Yes
12	VAR_ARG_LIST	N/A	Yes	Yes
13	ADDRESS_AS_LINEAR_EXPRESSION	N/A	No	Yes
14	ADDRESS_AS_NONLINEAR_EXPRESSION	N/A	No	Yes
15	ADDRESS_AS_VARIABLE	N/A	No	Yes
16	BUFFER_ADDRESS_ARRAY_INDEX	N/A	No	Yes
17	BUFFER_ADDRESS_POINTER	N/A	No	Yes
18	JAVA_GENERICS	N/A	Yes	No

Control Flow Complexities are defined with a CF identifier and a two number code as defined by **Table 4   CF Identifers by Language**.

Table 4   CF Identifers by Language

Identifier	Name	Group	Java	C/Binary
01	CALLBACK	N/A	Yes	Yes
02	INDIRECTLY_RECURSIVE	N/A	Yes	Yes
03	INFINITE_LOOP	N/A	Yes	Yes

Identifier	Name	Group	Java	C/Binary
04	INTERCLASS_1	INTERCLASS	Yes	No
05	INTERCLASS_2	INTERCLASS	Yes	No
06	INTERCASS_10	INTERCLASS	Yes	No
07	INTERCLASS_50	INTERCLASS	Yes	No
08	INTERFILE_1	INTERFILE	No	No
09	INTERFILE_2	INTERFILE	No	No
10	INTERFILE_10	INTERFILE	No	No
11	INTERFILE_50	INTERFILE	No	No
12	INTERPROCEDURAL_1	INTERPROCEDURAL	Yes	Yes
13	INTERPROCEDURAL_2	INTERPROCEDURAL	Yes	Yes
14	INTERPROCEDURAL_10	INTERPROCEDURAL	Yes	Yes
15	INTERPROCEDURAL_50	INTERPROCEDURAL	Yes	Yes
16	INTERRUPT	N/A	Yes	No
17	INTERUPT_CONTINUE	N/A	Yes	No
18	POINTER_TO_FUNCTION	N/A	No	Yes
19	RECURSIVE	N/A	Yes	Yes
20	SEQUENCE	N/A	Yes	Yes
21	AT_EXIT	N/A	No	No
22	MACROS	N/A	No	Yes
23	SET_JUMP_LONG_JUMP	N/A	No	Yes
24	UNCONDITIONAL_JUMP	N/A	No	Yes
25	BREAK_WITH_LABEL	N/A	Yes	No
26	FUNCTION_INVOCATION_OVERLOAD	N/A	Yes	No

Finally A unique increment which is comprised of two digits and starts at 01 is added to the end of each test case name to prevent the above parameters from having a possible collision.

It is important to note that it is not the values of a given field that are important to TEXAS but rather the order of those fields. If values are presented in the wrong order certain parts of the TEXAS archive processer may fail.

1.1.1 Archive

When creating an archive tar ball the xml file, the parent folder and the tarball of the parent folder should all share the same name as the name in the metadata.

For example.

```
C-WIRE.tar.gz
     C-WIRE/
          C-WIRE.XML
                Name=C-WIRE
```

1.1.2 Metadata XML

The metadata xml file inside the archive package should be named the same name as the archive. Using the same examples as above the Metadata xml file for Lucene would be named

```
  J-LUCE.xml
```

and the xml file for Wireshark would be named

```
  C-Wire.xml
```

An IO pair should take the following format:

Good IO Pair

```
 <Language>-<Program name in 4 characters>-<Type either good or bad>-<number>
```

For example, good I/O Pair number 4 for Wireshark would be named

```
  C-WIRE-GOOD-04
```

1.2 Archival Guidelines

By default TEXAS accepts and executes GZIP tar archives by default however it should support other UNIX based compression algorithms as well. GZIP compressed files can be normally identified by their tar.gz extension.

---

 12 December 2014

2   Package Structure

A test package must contain at least the following structural elements inside the archive. When all of these files and directories are packaged up into a tar ball and gziped they can be considered a "Test Package"

2.1   Top Level Directory

The test case files must be in a single top-level directory inside of the archive.

2.2   Metadata Xml File

This xml file is in the metadata format described in later sections and documented in Appendix A. This metadata contains all the instructions needed for TEXAS to build, execute and score a given test case. It also contains the Input and Output pairs (I/O PAIRS) used to perform the testing. This xml file may also contain any necessary pre-process and post-process or co-process that must be run with a test case to ensure that it functions and can be tested.

2.3   Install Directory

This is the empty directory that a given test program is built and installed to. Its default name is install but this can be changed in the metadata.

2.4   Scripts Directory

This directory contains any scripts needed to support or run a test case. Its default name is scripts but this can be changed in the metadata. This folder gets added to the path for pre-process and post-process, co-process, and scoring scripts as well as the main execution script.

2.5   I/O Data Directory

This directory contains all the "good" inputs used to test various inputs for the programs, it is also the directory that the all pre-process and post-process, co-process, and scoring scripts as well as the main execution script execute from.. Additionally, any scoring checks with Actual elements that reference files will accept relative paths from this directory. This directories default name is `testData` however it can be changed in the metadata.

2.6   Score Data Directory

This directory contain all the "good" outputs that scoring checks can be written against. For example, if the scoring check is an md5 sum. The expected sum for that IO Pair may be stored in a text file in this subdirectory. Additionally, any scoring checks with Expected elements that reference files will accept relative paths from this file. This directory's default name is `testOutput` however it can be changed in the metadata.

3 Metadata Structure

The full schema can be found in **Appendix A: Metadata Schema**.

3.1 Project

The project element is the root element of the schema and contains all the information necessary to describe a test case.

3.1.1 Schema

```
<xs:element name="project" type="ProjectType"/>
<xs:complexType name="ProjectType">
   <xs:sequence>
      <xs:element minOccurs="0" name="description" type="xs:string"/>
      <xs:element minOccurs="0" name="additional_information"
         type="AdditionalInformationType"/>
      <xs:element minOccurs="0" name="structure"
         type="TestCaseStructureType"/>
      <xs:element minOccurs="0" name="resource_limits"
         type="ResourceLimitsGroupType"/>
      <xs:element minOccurs="0" name="run_command" type="xs:string"/>
      <xs:element minOccurs="0" name="weakness"
         type="WeaknessType" maxOccurs="unbounded"/>
      <xs:element name="build_commands" type="BuildCommandType"/>
      <xs:element minOccurs="0" name="output_checks"
         type="OutputCheckGroupType"></xs:element>
      <xs:element minOccurs="0" name="iopairs"
         type="IOPairGroupType"></xs:element>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="base_program" type="xs:string"/>
   <xs:attribute name="language" type="LanguageType" use="required"/>
   <xs:attribute name="targeted_architecture"
      type="ArchitectureType" default="ANY"/>
   <xs:attribute name="targeted_os" type="OperatingSystemType"
      default="LINUX"/>
   <xs:attribute name="test_case_type" type="TestCaseType"
      default="ALL" use="optional"/>
</xs:complexType>
```

3.1.2 Attributes
This project element has 3 user settable attributes.

3.1.2.1 name

Required: True

Default: This is the name of the Test Case. It should mirror the name of the archive, top level directory and the XML file in which it is contained.

3.1.2.2 base_program

Required: False

Default: None

This is the name of the base program that a weakness has been injected or found in.

3.1.2.3 language

Required: True

           12 December 2014

Default: None

Acceptable Inputs:

```
        C
        C++
        Java
```

This is the language of the program. Valid inputs are C, C++, Java

3.1.2.4   targeted_architecture

Required: False

Default: ANY

Acceptable Input:

```
        ANY
        x86
        x86_64
```

This defines the targeted architecture of the program.

3.1.2.5   targeted_os

Required: False

Default: LINUX

Acceptable Inputs:

```
         ALL
        LINUX
        WINDOWS
        OSX
        ANDROID
        IOS
```

This is the targeted OS of the project.

3.1.2.6   test_case_type

Required: False

Default: All

Acceptable Inputs:

```
        All
        BINARY
        SOURCE
```

3.1.3   Children

3.1.3.1   Description

Required: False

Approved for public release; distribution unlimited.

This element contains a textual description of the test case.

```
<description>This is a sample description for a project.</description>
```

3.1.3.2   structure

Required: False

See 3.2 below for more information on this element.

3.1.3.3   run_command

This element contains the base command that is used to execute an IO Pair. You may use environment variables in this command, as they will be expanded at run time. It is important to note that an IO Pairs run command may override this or may have contents from the I/O Pair process appended to it.

```
<run_command> bin/wireshark input/input.pcap</run_command>
```

3.1.3.4   build_commands

Required: False

See 3.3 below for more information on this element

3.1.3.5   iopairs

Required: False

See 3.4 below for more information on this element

3.1.3.6   additional_information

Required: False

This simple string provides a place for additional information

3.1.3.7   resource_limits

Required: False

Acceptable Inputs:

```
RLIMIT_AS

RLIMIT_NOFILE

RLIMIT_NOPROC

RLIMIT_CPU
```

This simple string defines the resource limits

3.1.4   Samples

```
<project name="SAMPLE-PROGRAM" language="Java">
    <!—Children -->
</project>
```

3.2   Structure

This element defines the test case structure type

---

                   12 December 2014

3.2.1 Schema

```
<xs:complexType name="TestCaseStructureType">
      <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="binary"
              type="BinaryDistributionType"/>
      </xs:sequence>
      <xs:attribute default="src" name="source" type="xs:string"/>
      <xs:attribute default="install" name="install" type="xs:string"/>
      <xs:attribute default="scripts" name="scripts" type="xs:string"/>
      <xs:attribute default="testData" name="iodata" type="xs:string"/>
      <xs:attribute default="testOutput" name="scoredata" type="xs:string"/>
  </xs:complexType>
```

3.2.2 Attributes

3.2.2.1 source

 Required: False

 Default: src

This attribute is the name of the folder where a programs source code resides.

3.2.2.2 install

 Required: False

 Default: install

This attribute is the name of the folder where the program is installed

3.2.2.3 scripts

 Required: False

 Default: scripts

This attribute is the name of folder where any supporting scripts are stored

3.2.2.4 iodata

 Required: False

 Default: testData

This attribute is the name of folder where input data files are stored

3.2.2.5 scoredata

 Required: False

 Default: *testOutput*

This attribute is the name of folder where the input data files are stored

3.3 Build Commands

Build command elements may contain multiple command elements and are executed in sequential order

---

3.3.1 Schema

```
<xs:complexType name="BuildCommandType">
    <xs:sequence maxOccurs="unbounded">
        <xs:element name="command" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="ldflags" type="xs:string"/>
    <xs:attribute name="libs" type="xs:string"/>
    <xs:attribute name="cflags" type="xs:string"/>
    <xs:attribute name="cppflags" type="xs:string"/>
</xs:complexType>
```

3.3.2 Attributes

3.3.2.1 Ldflags

Required: False

Default: None

This attribute is a basic string type

3.3.2.2 Libs

Required: False

Default: None

This attribute is a basic string type

3.3.2.3 Cflags

Required: False

Default: None

This attribute is a basic string type

3.3.2.4 Cpfflfags

Required: False

Default: None

This attribute is a basic string type

3.3.3 Children

3.3.3.1 Command

Required: True

Command is a simple string of the command you wish to run. This is passed to the system shell at execution time. You may use environment variables here as they are expanded at run time.

```
<command>./configure</command>
```

3.3.4 Samples

```
<buildcommands>
    <command>./configure</command>
    <command>make</command>
```

```
        <command>make install</command>
</buildcommands>
```

3.4   I/O PAIRS

I/O Pairs are comprised of several I/O Pair element objects.

3.4.1   Schema

```
<xs:complexType name="IOPairGroupType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="iopair" type="IOPairType"/>
        </xs:sequence>
    </xs:complexType>
```

3.4.2   Children

3.4.2.1   Iopair

 Required: True

Each I/O Pair is comprised of several sub elements needed for I/O Pair execution. For more information see section 3.5

3.5   I/O Pair

Io pairs are comprised of several elements the most important of which are the pre-, co- and post-process elements. Each of these elements can have an unlimited number of processes in them to complete a given task. Each contained process is executed in sequential order. A preprocess is run before the execution of the main run command. A co-process is executed alongside the run command in parallel and a post process is run after the run command finishes its execution. These elements are not required to exist since only a run command element is needed to have a valid IO pair. NOTE: When running a server program, the last post-process should provide a normal shutdown of the server process, either a Kill or a SIGTERM to clean up the system.

3.5.1   Schema

```
<xs:complexType name="IOPairType">
    < !—children></children>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="type" type="IOPairCategoryType" use="required"/>
        <xs:attribute default="300" name="timeout" type="xs:int"/>
        <xs:attribute default="NONE" name="reset_scheme" type="ResetType"/>
        <xs:attribute name="derived_from" type="xs:string"/>
    </xs:complexType>
```

3.5.2   Attributes

3.5.2.1   name

 Required: True

 Default:

The name of a given IO pair

3.5.2.2   type

 Required: True

Approved for public release; distribution unlimited.   12 December 2014

Default:

Allowed Values:

```
GOOD
BAD
```

If it's a good or bad Input.

3.5.2.3   timeout

Required: False

Default: 300

Default timeout in seconds to allow an IO pair to execute the run command

3.5.2.4   reset_scheme

Required: False

Default: NONE

Allowed Values:

```
REBOOT
REPROVISION
NONE
```

Simple string defining reset type for a given IO pair

3.5.2.5   derived_from

Required: False

Default: NONE

Simple string defining the name a given I/O Pair may be derived from

3.5.3   Children

3.5.3.1   Description

Required: False

This simple string element is a short description of what the io pair does.

An example:

```
<description>Open a saved pcap file</description>
```

3.5.3.2   technical_impact

Required: True

Available Inputs:

```
NONE
UNSPECIFIED
READ_FILE
READ_APPLICATION_DATA
GAIN_PRIVILEGES
HIDE_ACTIVITIES
EXECUTE_UNATHORIZED_CODE
MODIFY_FILES
MODIFY_APPLICATION_DATA
BYPASS_PROTECTION_MECHANISM
ALTER_EXECUTION_LOGIC
UNEXPECTED_STATE
  DOS_UNCONTROLLED_EXIT
  DOS_AMPLIFICATION
  DOS_INSTABILITY
  DOS_BLOCKING
  DOS_RESOURCE_CONSUPTION
```

This simple string element is a short description of the technical impact

3.5.3.3 run_command

Required: False

This is a simple string, which will be executed by the system. It is acceptable to use environment variables, as they will be expanded at run time. This element also has an attribute

3.5.3.3.1   prepend_command

Required: False

Default: True

This Boolean attribute determines if the project run command is prepended to the I/O Pair run command

3.5.3.4 pre_process

Required: False

This element is described in section 3.6

3.5.3.5 co_process

Required: False

This element is described in section 3.6

3.5.3.6 post_process

Required: False

Approved for public release; distribution unlimited.   12 December 2014

This element is described in section 3.6

3.5.3.7 *scoring_formula*

 Required: True

This element is described in section 0

3.5.3.8 *output_check*

 Required: False

This element is described in section 3.8

3.6 Execution Group

An execution group is not an actual element. It is the base type for the pre_process co_process and post_process elements.

3.6.1 Children

3.6.1.1 *process*
Each execution group contains a process element, which is simply a holder for multiple element types. These types can be any of the following elements.

3.6.1.1.1 script

Each script element contains a simple string command, which is executed by the system. You may use environment variables here as they will be expanded at run time.

```
<script>wget 127.0.0.1</script>
```

3.6.1.1.2 Database

Each database element contains a database command that is to be executed in either a postgress or mysql database.

```
<database>SELECT version();</database>
```

3.6.1.1.3 File

Each file element will take the contents and write it out to a file

```
<file encoding="STRING" path="file.txt" type="USE_TEXT">Hello world</file>
```

3.6.1.1.4 Socket

This element will take the contents and write them to a system socket on the port and address specified in the applicable attributes.

```
<socket port=9999 address=127.0.0.1>Hello world</socket>
```

 12 December 2014

3.7 Scoring Formula

```
Schema

    <xs:complexType name="ScoringFormulaType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="formula"
type="FormulaType"> </xs:element>
        </xs:sequence>
        <xs:attribute name="default" type="xs:string" use="required"/>
    </xs:complexType>
```

3.7.1 Attributes

3.7.1.1 Default

Required: True

Default: This attribute is a basic string type

3.7.1.2 Children

3.7.1.2.1 Formula

Required: False

The formula is a basic string type and specifies the scoring formula to use. This child also has an attribute named default.

Required: True

Default:

This attribute is a basic string type.

3.8 Output Checks

This element operates similar to the execution groups and has two possible children.

3.8.1 Schema

```
    <xs:complexType name="OutputCheckGroupType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="output_check"/>
        </xs:sequence>
    </xs:complexType>
<xs:complexType name="SimpleOutputCheckType">
        <xs:simpleContent>
            <xs:extension base="ValueWithType">
                <xs:attribute name="name" type="xs:string"/>
                <xs:attribute name="check_type" type="SimpleCheckType"/>
                <xs:attribute name="rescorable" type="xs:boolean"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CompareOutputCheckType">
        <xs:sequence>
            <xs:element name="actual" type="ActualOutputType"/>
            <xs:element name="expected" type="ExpectedOutputType"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="check_type" type="CompareCheckType" use="required"/>
        <xs:attribute name="rescorable" type="xs:boolean"/>
    </xs:complexType>
```

3.8.2 Children

3.8.2.1 simple_check

Required: True

The simple check checks either a system provided variable or a Boolean output.

3.8.2.1.1 Attributes

3.8.2.1.1.1 check_type

Required: True

Default:

This attribute defines the type of check to be performed.

Available Inputs:

```
EXACT_MATCH
CONTAINS
REGEX
SCRIPT
EXISTS
SPECIAL
```

3.8.2.1.1.2 encoding

Required: True

Default:

This defines the encoding of the input.

Available Inputs:

```
STRING

BASE64
```

3.8.2.1.1.3 name

Required: True

Default:

This simple string defines the type of return.

Available Inputs:

```
TIMEOUT

CONTROLLED EXIT
```

3.8.2.1.1.4 type

Required: True

Default:

This defines direct text or contents of a file are checked.

Available Inputs:

```
USE_TEXT

USE_FILE_CONTENTS
```

3.8.2.1.2   Sample

```
<simple_check check_type="SPECIAL" encoding="STRING" name="TIMEOUT" type="USE_TEXT">
DOES_NOT_RETURN</simple_check>
```

3.8.2.2   compare_check

Required: True

The compare checks two provided inputs based on the check type.

3.8.2.2.1   Attributes

3.8.2.2.1.1 name

This simple string should be set to the I/O Pair name

3.8.2.2.1.2 check_type

Required: True

Default:

This attribute defines the type of check to be performed.

Available Inputs:

```
EXACT_MATCH
CONTAINS
REGEX
SCRIPT
EXISTS
SPECIAL
```

3.8.2.2.2   Children

3.8.2.2.2.1 actual

Required: True

This child stores the location of the output from the execution of the text case.

3.8.2.2.2.1.1 Attributes:

3.8.2.2.2.1.1.1 output_type

Required: True

Default:

This attribute defines the output type to check.

Available Inputs:

```
STANDARD_OUT
STANDARD_ERROR
FILE_CONTENTS
FILE_NAME
RETURN_CODE
```

3.8.2.2.2.1.1.2 encoding

Required: True

Default:

This defines the encoding of the input.

Available Inputs:

```
STRING
BASE64
```

3.8.2.2.2.1.1.3 input_id_ref

Required: False

Default:

This simple string should be used in when checking STDOUT or STDERR and is used to find the log files of a given process.

3.8.2.2.2.2 *expected*

This Child stores the expected output for a given test case run.

3.8.2.2.2.2.1 Attributes:

3.8.2.2.2.2.1.1 *type*

Required: True

Default:

This defines direct text or contents of a file are checked.

Available Inputs:

```
USE_TEXT

USE_FILE_CONTENTS
```

3.8.2.2.3   Sample

```
<compare_check name="GOOD-10" check_type="CONTAINS">

      <actual output_type="STANDARD_OUT" encoding="STRING" input_id_ref="SIKULI-GOOD-10"></actual>

       <expected type="USE_FILE_CONTENTS">good-10/output.txt</expected>

</compare_check>

      <simple_check check_type="SPECIAL" encoding="STRING" name="TIMEOUT"
type="USE_TEXT">DOES_NOT_RETURN</simple_check>

      <simple_check check_type="SPECIAL" encoding="STRING" name="CONTROLLED_EXIT"
type="USE_TEXT">CONTROLLED_EXIT</simple_check>

</compare_check>
```

4   Special Considerations

Test case size can play large role in how long it takes to run a test case, particularly for console programs that tend to execute quickly. Since TEXAS will automatically create a results archive after every run of analyze or I/O pair it is important to pay attention to how large the base archive is and what files are being preserved.

Appendix A. Metadata Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
    targetNamespace="urn:stonesoup:project" xmlns="urn:stonesoup:project">

    <xs:element name="project" type="ProjectType"/>
    <xs:complexType name="ActualOutputType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="output_type" type="OutputType"
use="required"/>
                <xs:attribute name="input_id_ref" type="xs:string"/>
                <xs:attribute name="encoding" type="EncodingType"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>s
    <xs:complexType name="AdditionalInformationType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="1"
name="keyvalue" type="KeyValuePairType"
            />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BinaryDistributionType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="architecture"
type="ArchitectureType" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="BuildCommandType">
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="command" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="ldflags" type="xs:string"/>
        <xs:attribute name="libs" type="xs:string"/>
        <xs:attribute name="cflags" type="xs:string"/>
        <xs:attribute name="cppflags" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="CodeLocationType">
        <xs:sequence>bu
            <xs:element minOccurs="0" name="method" type="MethodType">
</xs:element>
        </xs:sequence>
        <xs:attribute name="file_name" type="xs:string" use="required"/>
        <xs:attribute name="line_number" type="xs:int"/>
    </xs:complexType>
    <xs:complexType name="CodeLocationWithDescriptionType">
        <xs:complexContent>
            <xs:extension base="CodeLocationType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="description"
type="xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ControlFlowLocationType">
        <xs:complexContent>
```

 12 December 2014

```
                <xs:extension base="CodeLocationType">
                    <xs:attribute name="type" type="ControlFlowType"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DatabaseContentType">
            <xs:simpleContent>
                <xs:extension base="NamedValueWithType">
                    <xs:attribute name="server" type="xs:string" use="required"/>
                    <xs:attribute name="port" type="xs:int" use="optional"/>
                    <xs:attribute name="database_type" type="DatabaseType"
use="required"/>
                    <xs:attribute name="database_name" type="xs:string"
use="required"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="DataFlowLocationType">
            <xs:complexContent>
                <xs:extension base="CodeLocationType">
                    <xs:attribute name="type" type="DataFlowType"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DataTypeLocationType">
            <xs:complexContent>
                <xs:extension base="CodeLocationType">
                    <xs:attribute name="type" type="DataType"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="EnvironmentVariableType">
            <xs:simpleContent>
                <xs:extension base="NamedValueWithType">
                    <xs:attribute name="key" type="xs:string" use="required"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="ExecutionGroupType">
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" ref="processes"/>
            </xs:sequence>
            <xs:attribute default="SEQUENCE" name="scheme"
type="ExecutionSchemeType"> </xs:attribute>
        </xs:complexType>
        <xs:complexType name="ExpectedOutputType">
            <xs:simpleContent>
                <xs:extension base="ValueWithType"/>
            </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="FeaturesType">
            <xs:sequence maxOccurs="1" minOccurs="1">
                <xs:element name="taint_source"
type="TaintSourceLocationType"> </xs:element>
                <xs:element maxOccurs="unbounded" minOccurs="0" ref="complexity"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="FileContentsType">
            <xs:simpleContent>
                <xs:extension base="NamedValueWithType">
                    <xs:attribute name="path" type="xs:string" use="required"/>
                </xs:extension>
            </xs:simpleContent>
```

```
    </xs:complexType>
    <xs:complexType name="FormulaType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="IOPairGroupType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="iopair" type="IOPairType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IOPairType">
        <xs:sequence>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="technical_impact"
type="TechnicalImpactType" maxOccurs="unbounded"/>
            <xs:element minOccurs="0" name="resource_limits"
type="ResourceLimitsGroupType"/>
            <xs:element name="pre_processes" type="ExecutionGroupType"
minOccurs="0"> </xs:element>
            <xs:element name="run_command"
type="ValueWithPrependType"> </xs:element>
            <xs:element name="co_processes" type="ExecutionGroupType"
minOccurs="0"> </xs:element>
            <xs:element name="post_processes"
type="ExecutionGroupType" minOccurs="0"> </xs:element>
            <xs:element name="scoring_formula" type="ScoringFormulaType"/>
            <xs:element minOccurs="0" name="output_checks"
type="OutputCheckGroupType"> </xs:element>

        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="type" type="IOPairCategoryType" use="required"/>
        <xs:attribute default="300" name="timeout" type="xs:int"/>
        <xs:attribute default="NONE" name="reset_scheme" type="ResetType"/>
        <xs:attribute name="derived_from" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="KeyValuePairType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="key" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="MethodType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
name="parameter" type="xs:string">
            </xs:element>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="classname" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="NamedValueWithType">
        <xs:simpleContent>
            <xs:extension base="ValueWithType">
                <xs:attribute name="input_id" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="OutputCheckGroupType">
```

```xml
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" ref="output_check"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="SimpleOutputCheckType">
                <xs:simpleContent>
                    <xs:extension base="ValueWithType">
                        <xs:attribute name="name" type="xs:string"/>
                        <xs:attribute name="check_type" type="SimpleCheckType"/>
                        <xs:attribute name="rescorable" type="xs:boolean"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
            <xs:complexType name="CompareOutputCheckType">
                <xs:sequence>
                    <xs:element name="actual" type="ActualOutputType"/>
                    <xs:element name="expected" type="ExpectedOutputType"/>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required"/>
                <xs:attribute name="check_type" type="CompareCheckType" use="required"/>
                <xs:attribute name="rescorable" type="xs:boolean"/>
            </xs:complexType>
            <xs:complexType name="ParameterType">
                <xs:attribute name="name" type="xs:string"/>
                <xs:attribute name="type" type="xs:string" use="required"/>
            </xs:complexType>
            <xs:complexType name="ProcessType">
                <xs:choice>
                    <xs:element name="environment_variable"
type="EnvironmentVariableType"/>
                    <xs:element name="database" type="DatabaseContentType"/>
                    <xs:element name="file" type="FileContentsType"/>
                    <xs:element name="socket" type="SocketType"/>
                    <xs:element name="script" type="ScriptType"/>
                </xs:choice>
                <xs:attribute default="0" name="delay_before" type="xs:int"/>
                <xs:attribute default="0" name="delay_after" type="xs:int"/>
            </xs:complexType>
            <xs:complexType name="ProjectType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="description" type="xs:string"/>
                    <xs:element minOccurs="0" name="additional_information"
type="AdditionalInformationType"/>
                    <xs:element minOccurs="0" name="structure"
type="TestCaseStructureType"/>
                    <xs:element minOccurs="0" name="resource_limits"
type="ResourceLimitsGroupType"/>
                    <xs:element minOccurs="0" name="run_command" type="xs:string"/>
                    <xs:element minOccurs="0" name="weakness"
type="WeaknessType" maxOccurs="unbounded"/>
                    <xs:element name="build_commands" type="BuildCommandType"/>
                    <xs:element minOccurs="0" name="output_checks"
type="OutputCheckGroupType"> </xs:element>
                    <xs:element minOccurs="0" name="iopairs"
type="IOPairGroupType"> </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required"/>
                <xs:attribute name="base_program" type="xs:string"/>
                <xs:attribute name="language" type="LanguageType" use="required"/>
                <xs:attribute name="targeted_architecture"
type="ArchitectureType" default="ANY"/>
                <xs:attribute name="targeted_os" type="OperatingSystemType"
default="LINUX"/>
```

```
            <xs:attribute name="test_case_type" type="TestCaseType"
default="ALL" use="optional"/>
    </xs:complexType>
    <xs:complexType name="ResourceLimitsGroupType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="limit"
type="ResourceLimitKeyValuePairType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ResourceLimitKeyValuePairType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="type" type="ResourceLimitType"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="ScriptType">
        <xs:simpleContent>
            <xs:extension base="NamedValueWithType">
                <xs:attribute default="false" name="background"
type="xs:boolean"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="ScoringFormulaType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="formula"
type="FormulaType"> </xs:element>
        </xs:sequence>
        <xs:attribute name="default" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="SocketType">
        <xs:simpleContent>
            <xs:extension base="NamedValueWithType">
                <xs:attribute name="address" type="xs:string" use="required"/>
                <xs:attribute name="port" type="xs:int" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TaintSourceLocationType">
        <xs:complexContent>
            <xs:extension base="CodeLocationType">
                <xs:attribute name="type" type="TaintSourceType"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TestCaseStructureType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="binary"
                type="BinaryDistributionType"/>
        </xs:sequence>
        <xs:attribute default="src" name="source" type="xs:string"/>
        <xs:attribute default="install" name="install" type="xs:string"/>
        <xs:attribute default="scripts" name="scripts" type="xs:string"/>
        <xs:attribute default="testData" name="iodata" type="xs:string"/>
        <xs:attribute default="testOutput" name="scoredata" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="ValueWithType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="type" default="USE_TEXT"
type="ValueType"> </xs:attribute>
                <xs:attribute default="STRING" name="encoding"
```

```
type="EncodingType"> </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="ValueWithPrependType">
        <xs:simpleContent>
            <xs:extension base="ValueWithType">
                <xs:attribute default="true"
name="preprend_runcommand" type="xs:boolean"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="VariantType">
        <xs:sequence>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element minOccurs="0" name="crossover_point"
type="CodeLocationWithDescriptionType"> </xs:element>
            <xs:element minOccurs="0" name="interaction_point"
                type="CodeLocationWithDescriptionType"> </xs:element>
            <xs:element minOccurs="0" name="trigger_point"
type="CodeLocationWithDescriptionType">
            </xs:element>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="cwe" type="CWEType"/>
    </xs:complexType>
    <xs:complexType name="WeaknessType">
        <xs:sequence>
            <xs:element name="variant" type="VariantType"/>
            <xs:element name="seed_injection_point"
type="CodeLocationType" minOccurs="0"/>
            <xs:element name="features" type="FeaturesType"/>
        </xs:sequence>
        <xs:attribute name="generation_method"
type="GenerationMethodType" use="required"/>
    </xs:complexType>
    <xs:simpleType name="ArchitectureType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ANY"/>
            <xs:enumeration value="x86"/>
            <xs:enumeration value="x86_64"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CheckType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="EXACT_MATCH"/>
            <xs:enumeration value="CONTAINS"/>
            <xs:enumeration value="REGEX"/>
            <xs:enumeration value="SCRIPT"/>
            <xs:enumeration value="EXISTS"/>
            <xs:enumeration value="SPECIAL"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SimpleCheckType">
        <xs:restriction base="CheckType">
            <xs:enumeration value="SCRIPT"/>
            <xs:enumeration value="EXISTS"/>
            <xs:enumeration value="SPECIAL"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CompareCheckType">
        <xs:restriction base="CheckType">
            <xs:enumeration value="EXACT_MATCH"/>
```

```
                <xs:enumeration value="CONTAINS"/>
                <xs:enumeration value="REGEX"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ControlFlowType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="CALLBACK"/>
                <xs:enumeration value="INDIRECTLY_RECURSIVE"/>
                <xs:enumeration value="INFINITE_LOOP"/>
                <xs:enumeration value="INTERCLASS_1"/>
                <xs:enumeration value="INTERCLASS_2"/>
                <xs:enumeration value="INTERCLASS_10"/>
                <xs:enumeration value="INTERCLASS_50"/>
                <xs:enumeration value="INTERFILE_1"/>
                <xs:enumeration value="INTERFILE_2"/>
                <xs:enumeration value="INTERFILE_10"/>
                <xs:enumeration value="INTERFILE_50"/>
                <xs:enumeration value="INTERPROCEDURAL_1"/>
                <xs:enumeration value="INTERPROCEDURAL_2"/>
                <xs:enumeration value="INTERPROCEDURAL_10"/>
                <xs:enumeration value="INTERPROCEDURAL_50"/>
                <xs:enumeration value="INTERRUPT"/>
                <xs:enumeration value="INTERRUPT_CONTINUE"/>
                <xs:enumeration value="POINTER_TO_FUNCTION"/>
                <xs:enumeration value="RECURSIVE"/>
                <xs:enumeration value="SEQUENCE"/>
                <xs:enumeration value="AT_EXIT"/>
                <xs:enumeration value="MACROS"/>
                <xs:enumeration value="SET_JUMP_LONG_JUMP"/>
                <xs:enumeration value="UNCONDITIONAL_JUMP"/>
                <xs:enumeration value="BREAK_WITH_LABEL"/>
                <xs:enumeration value="FUNCTION_INVOCATION_OVERLOAD"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CWEType">
        <xs:restriction base="xs:string">
                <xs:pattern value="[A-Z]{3}-[0-9]{1,3}"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DataType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="ARRAY"/>
                <xs:enumeration value="SIMPLE"/>
                <xs:enumeration value="VOID_POINTER"/>
                <xs:enumeration value="HEAP_POINTER"/>
                <xs:enumeration value="STRUCT"/>
                <xs:enumeration value="TYPEDEF"/>
                <xs:enumeration value="UNION"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DatabaseType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="MSSQL"/>
                <xs:enumeration value="MYSQL"/>
                <xs:enumeration value="POSTGRESQL"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DataFlowType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="ADDRESS_ALIAS_1"/>
                <xs:enumeration value="ADDRESS_ALIAS_2"/>
                <xs:enumeration value="ADDRESS_ALIAS_10"/>
                <xs:enumeration value="ADDRESS_ALIAS_50"/>
```

```xml
            <xs:enumeration value="ADDRESS_AS_CONSTANT"/>
            <xs:enumeration value="ADDRESS_AS_FUNCTION_RETURN_VALUE"/>
            <xs:enumeration value="INDEX_ALIAS_1"/>
            <xs:enumeration value="INDEX_ALIAS_2"/>
            <xs:enumeration value="INDEX_ALIAS_10"/>
            <xs:enumeration value="INDEX_ALIAS_50"/>
            <xs:enumeration value="BASIC"/>
            <xs:enumeration value="VAR_ARG_LIST"/>
            <xs:enumeration value="ADDRESS_AS_LINEAR_EXPRESSION"/>
            <xs:enumeration value="ADDRESS_AS_NONLINEAR_EXPRESSION"/>
            <xs:enumeration value="ADDRESS_AS_VARIABLE"/>
            <xs:enumeration value="BUFFER_ADDRESS_ARRAY_INDEX"/>
            <xs:enumeration value="BUFFER_ADDRESS_POINTER"/>
            <xs:enumeration value="JAVA_GENERICS"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="EncodingType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="STRING"/>
            <xs:enumeration value="BASE64"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ExecutionSchemeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SEQUENCE"/>
            <xs:enumeration value="PARALLEL"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="GenerationMethodType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAULT_INJECTED"/>
            <xs:enumeration value="REAL_WORLD"/>
            <xs:enumeration value="ENGINEERED"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="IOPairCategoryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="GOOD"/>
            <xs:enumeration value="BAD"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="LanguageType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="C"/>
            <xs:enumeration value="C++"/>
            <xs:enumeration value="JAVA"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="OperatingSystemType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ALL"/>
            <xs:enumeration value="LINUX"/>
            <xs:enumeration value="WINDOWS"/>
            <xs:enumeration value="OSX"/>
            <xs:enumeration value="ANDROID"/>
            <xs:enumeration value="IOS"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="OutputType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="STANDARD_OUT"/>
            <xs:enumeration value="STANDARD_ERROR"/>
            <xs:enumeration value="FILE_CONTENTS"/>
```

```xml
                <xs:enumeration value="FILE_NAME"/>
                <xs:enumeration value="RETURN_CODE"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ResetType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="NONE"/>
                <xs:enumeration value="REBOOT"/>
                <xs:enumeration value="REPROVISION"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ResourceLimitType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="RLIMIT_AS"/>
                <xs:enumeration value="RLIMIT_NOFILE"/>
                <xs:enumeration value="RLIMIT_NOPROC"/>
                <xs:enumeration value="RLIMIT_CPU"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="TaintSourceType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="ENVIRONMENT_VARIABLE"/>
                <xs:enumeration value="FILE_CONTENTS"/>
                <xs:enumeration value="SOCKET"/>
                <xs:enumeration value="SHARED_MEMORY"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="TechnicalImpactType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="NONE"/>
                <xs:enumeration value="UNSPECIFIED"/>
                <xs:enumeration value="READ_FILE"/>
                <xs:enumeration value="READ_APPLICATION_DATA"/>
                <xs:enumeration value="GAIN_PRIVILEGES"/>
                <xs:enumeration value="HIDE_ACTIVITIES"/>
                <xs:enumeration value="EXECUTE_UNAUTHORIZED_CODE"/>
                <xs:enumeration value="MODIFY_FILES"/>
                <xs:enumeration value="MODIFY_APPLICATION_DATA"/>
                <xs:enumeration value="BYPASS_PROTECTION_MECHANISM"/>
                <xs:enumeration value="ALTER_EXECUTION_LOGIC"/>
                <xs:enumeration value="UNEXPECTED_STATE"/>
                <xs:enumeration value="DOS_UNCONTROLLED_EXIT"/>
                <xs:enumeration value="DOS_AMPLIFICATION"/>
                <xs:enumeration value="DOS_INSTABILITY"/>
                <xs:enumeration value="DOS_BLOCKING"/>
                <xs:enumeration value="DOS_RESOURCE_CONSUMPTION"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="TestCaseType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="ALL"/>
                <xs:enumeration value="BINARY"/>
                <xs:enumeration value="SOURCE"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ValueType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="USE_TEXT"/>
                <xs:enumeration value="USE_FILE_CONTENTS"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="process" substitutionGroup="processes"
type="ProcessType"/>
```

```
    <xs:element name="process_group" substitutionGroup="processes"
type="ExecutionGroupType"/>
    <xs:element abstract="true" name="processes"/>
    <xs:element name="output_check" abstract="true"/>
    <xs:element name="simple_check" substitutionGroup="output_check"
type="SimpleOutputCheckType"/>
    <xs:element name="compare_check" substitutionGroup="output_check"
type="CompareOutputCheckType"/>
    <xs:element abstract="true" name="complexity"/>
    <xs:element name="control_flow" substitutionGroup="complexity"
type="ControlFlowLocationType"/>
    <xs:element name="data_flow" substitutionGroup="complexity"
type="DataFlowLocationType"/>
    <xs:element name="data_type" substitutionGroup="complexity"
type="DataTypeLocationType"/>
</xs:schema>
```

                   12 December 2014