KDM Analytics

Working Together to Build Confidence

USER'S GUIDE

TOOL OUTPUT INTEGRATION FRAMEWORK

Document Version 1.0.9

Software Release 1.8.7

Beta

This document applies to Tool Output Integration Framework (TOIF) version 1.0 Beta and may apply to subsequent releases. To check for newer versions of this document, please contact KDM Analytics Customer Support (e-mail: support@kdmanayltics.com).

KDM Analytics Inc
3730 Richmond Rd
Suite 204
Ottawa, ON
Canada
(613) 627-1010
Internet E-Mail: support@kdmanayltics.com
Website: http://www.kdmanalytics.com

Contents

Introduction

The *Tool Output Integration Framework* (TOIF) is a powerful open source vulnerability detection platform. It allows users to analyze systems, for the purpose of performing defect sightings on a project. TOIF provides:

- Reference implementation for standard-based adaptors
- Further CWE normalization of vulnerability reports based on the Software Fault Patterns; adoption of SFPs
- Adoption of standard-based reporting of vulnerabilities
- Utilization of open source development to advance the SwA space
- A common protocol for exchanging vulnerability findings

TOIF is based on existing standard protocol for exchanging system facts, the OMG Knowledge Discovery Metamodel (KDM), now ISO/IEC 19506.

This document contains information about how to use TOIF. It includes procedures, notes, and other background information.

Audience

This document is intended to help system and software engineers, system analysts, security analysts, and system architects to use the *Tool Output Integration Framework* (TOIF) in performing defect sightings.

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation.

The table below identifies the formatting conventions used in KDM Analytics documents to represent different types of information.

Type of information	Formatting convention	Example
Slash characters in path names	Follow the UNIX convention (forward slash). Usually appears in `monospace font` as part of command-line input or output. **Note**: Substitute with a backward slash (\) for Windows platform.	/workspace/xcerces-c-win32

File path and names	`monospace font`	Double click on the `kdm-workbench.exe` file.
Information to be substituted with user-provided information	Description of information item to be supplied surrounded by angle brackets, `monospace font`	Extract the `KDM-Workbench-<version>-<platform version>`.zip file into a desired target folder.
Filename in descriptive text	*italics*	captured in the *autoconfig.properties* file
Step-by-step procedures.	1., 2., 3.,...	**1)** Click START **2)** Click **Advanced** tab
User interface fields and menu options	SMALL CAPITALS	Right click on MY COMPUTER and from
User interface command buttons	**Bold**	1) Click **Start**
Names of keys on the keyboard.	CAPITALS	SHIFT, CTRL, or ALT.
Key combinations for which the user must press and hold down one key and then press another	KEY+KEY	CTRL+P, or ALT+F4.

For more information on specialized terms used in the documentation, see the Glossary at the end of this document.

Getting Help

A PDF version of the User Guide is available directly from the KDM Analytics product distribution zip file.

The following sections provide details on when and how to contact KDM Analytics support if you encounter a problem and cannot resolve the issue after consulting the published information.

When to contact us

If you encounter a problem or deficiency working with our product and are unable to resolve it after consulting the published information, please contact KDM Analytics support by e-mail: support@kdmanayltics.com

Chapter 1

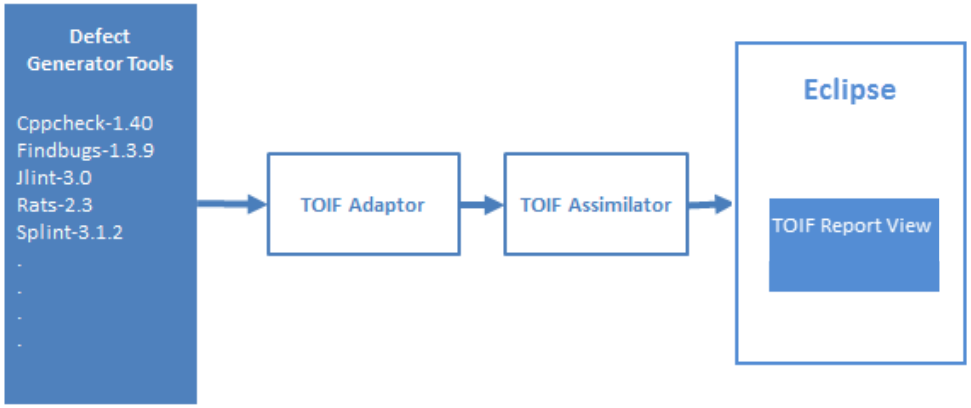
What is TOIF?

The *Tool Output Integration Framework* (TOIF) is a powerful open source vulnerability detection platform that provides analysts information of system defects with the ability to:

- Integrate multiple vulnerability detection tools as “data feeds” into the repository
- Collate findings from several tools
- Put vulnerability findings into the context of other facts about the system (such as metrics, architecture, design patterns, etc.)

How does TOIF work?

TOIF takes the output of defect generator tools and displays the results in *Eclipse*.



TOIF Components

TOIF includes the following components:

- TOIF Adaptor: *TOIF Adaptor* is used to collect the output from various vulnerability detection tools and convert their output into TOIF xml
- TOIF Assimilator: After running the *TOIF Adaptor* you need to run the Assimilator to merge TOIF findings and/or KDM data into a common fact-orientated repository or file
- TOIF Report View: Once you have your TOIF findings assimilated you view the *TOIF Report View* to display the results in *Eclipse*.

Chapter 2

Preparing to Install TOIF

The following sections provide the information required to install and get you working in the *Tool Output Integration Framework* (TOIF).

Installation Overview

This section provides the high-level steps for installing and running TOIF in *Eclipse*.

1) Gather the installation packages for *TOIF RCP*, and *Report View*.
2) Ensure that Eclipse 3.7.2 has been installed.
3) Read all of the information in this chapter before you install all the TOIF packages.
4) Unzip and install the *TOIF RCP* packages
5) Run the *TOIF Adaptor*.
6) Run the *TOIF Assimilator* against the TOIF/KDM files generated by the *TOIF Adaptor*.
7) Import your TOIF project into *Eclipse*.
8) Add the *TOIF Report View* to your Eclipse instance.
9) Generate defect data for viewing in the *TOIF Report View in Eclipse*.

Supported Deployment Configurations

The *TOIF RCP* is a standalone program. *TOIF Report View* should be used within an instance of *Eclipse*.

System Requirements

To install the TOIF your system must meet the minimum hardware and software requirements listed in the following sections.

Client Hardware

The following table lists the supported hardware platforms for TOIF.

Minimum	Recommended
500 MB free hard drive space	
2 GB RAM	8 GB RAM
Dual Core Processor	Quad core processor

Client Software

The following table lists the supported operating system platforms for TOIF.

Platform	Recommended
Linux	Linux
Microsoft Windows XP SP3 (or greater)	
Microsoft Windows 7 (32 and 64 bit)	

Chapter 3

Installing TOIF Packages

To install the *TOIF Adaptor*, *Assimilator* and *Report View* packages perform the following steps.

1) Download the `TOIF_RCP.zip`, and `TOIF_View_Updatesite.zip` files.
2) Run an unzip utility to extract the `TOIF_RCP.zip` into a desired target folder.

Note: Install the following defect generator tools according to their own instructions:

Cppcheck-1.40

Findbugs-1.3.9

Jlint-3.0

Rats-2.3

Splint-3.1.2

Make sure that the versions of these tools correspond to the same versions as the adaptors which you are using. The executable for each generator should be on the system path.

Chapter 4

Running the TOIF Adaptor

To run the TOIF Adaptor, perform the following tasks:

- Running the TOIF Adaptor from command line
- Integrating with a C or Java project's build

Running the TOIF Adaptor from the command line

To run the Adaptor from a command line, perform the following steps.

1) Open a command prompt.
2) Make sure that the TOIF_RCP (toif) command is on the PATH. Also, ensure that the generator tools are correctly installed.
3) Type `toif --adaptor=<Adaptor Name> --inputfile=<full path to input file> --outputdirectory=<path to output directory> --housekeeping=<path to housekeeping file> -- [Additional arguments].`

Where:

- `<Adaptor Name>` defines the name of the adaptor class. This is the adaptor that is to be used with the input source file. From this class, the framework is able to discover house keeping facts about the adaptor as well as which generator to call and what options to use. Typically the name is without the adaptor version number.
- `<full path to input file>` defines the full path to the input source file. In order for the adaptors to create all the facts for this file, a full path must be provided.
- `<path to output directory>` defines the path to the output directory. This is the directory where the subdirectories containing the TOIF XML file will be written.
- `<path to housekeeping file>` defines the path to the file containing the facts about the project's housekeeping. This file is specific to each adaptor and each project. This is because it is down to the user to provide the project details as well as which generator (scan tool) is running on the system.
- [Additional Arguments] defines any additional arguments that you may want. These must be entered after the TOIF Adaptor's required arguments and after a "--". These arguments can be included files or compilation options, and they will vary from generator to generator. For example, splint can take -I and -D options:

```
toif –-adaptor=SplintAdaptor –-inputfile=/home/user/foo.c --
outputdirectory=/home/user/output –housekeeping=housekeepingFile.txt
-- -I./includes -D_U_=
```

Integrating with C project's build

The best way to integrate the adaptors into the build is by wrapping the compiler and the adaptors into a script. When the compiler is called, the adaptors will be run for every source file used. To get the build process to use this wrapper instead of the compiler on its own, the compiler flag needs to be set during configuration of the make:

```
./configure CC=<myGccWrapper>
```

The make can then be continued as normal:

- `Make`
- `make install`

An example script has been provided in the examples folder (*Adaptors/examples/linux scripts*). However, the following directories will need to be changed within the script to suit your system.

- HOUSEKEEPING = `<the location of the housekeeping file>`
- OUTDIR = `<the output directory for the toif files>`

Integrating with Java project's build

It may be possible to integrate into a Java project's build by adding the following to the *build.xml* file.

```
<target name="check" depends="compile_src">
	<foreach param="file" target="run">
		<path>
			<fileset dir="${classes_dir}">
				<filename name="**/*.class" />
			</fileset>
		</path>
	</foreach>
</target>
<target name="run" >
	<exec executable="python">
		<arg value="<java adaptors python script>"/>
		<arg value="${file}"/>
		<arg value="${build_dir}"/>
	</exec>
</target>
```

This creates a new target which will find all the *.class* files in the destination directory of the project. For each file, the *javaAdaptors* python script will be run with the arguments that are specified.

Note: The value in the `<java adaptors python script>` needs to be changed to match where your python script is.

To run the adaptors and compile the project, perform the following steps.

Note: Example scripts are provided in the examples folder within the Adaptors directory (Adaptors/examples/linux scripts).

1) In the `javaAdaptors` python script modify the following options to suit your system.
   - OUTPUT = <path to the output directory for the toif files>
   - CLASSPATH = <path to the directory where the adaptors are located>
2) Open a command prompt.
3) Type `ant check`.

The output directory you defined in the *javaAdaptors* python script is created and contains all the toif files for your project.

Chapter 5

Running the TOIF Assimilator

The *TOIF assimilator* merges TOIF findings (toif files created by running the TOIF Adaptor) and/or KDM data into a common fact-orientated repository or file.

To run the *TOIF Assimilator*, perform the following steps.

1) Open a command prompt.
2) Make sure that the TOIF RCP (toif) is on the PATH.

   Do the following:

   - Type `toif --merge –outputfile=<output destination> --inputfile=[files or directories to merge...]`.

Note: The file extension for the output destination must be `.kdm`.

Where:

- `<-k>` defines that a .kdm file is to be used. The .kdm files are generally more mobile than a repository. If this option is used the path and filename of the .kdm file should be specified as the destination. For example,

  ```
  Toif –-merge –-outputfile=/dir/outputFile.kdm –-
  inputfile=/dir2/toifFiles/
  ```

- `[files or directories to merge...]` defines the toif files or tkdm files to be merged. Any number of toif files or tkdm files can be entered here. Tkdm files are not necessary if kdm data is not required. Alternatively, a directory can be specified which contains the toif or kdm files.

Warning: Be careful to ensure that the output file is not accidentally produced where the input files are being read from.

The resulting assimilated output data, from running the *TOIF Assimilator*, is used as the input to the *TOIF Report View*.

Chapter 6

Running the TOIF Report View

The *TOIF Report View* allows the assimilated output data to be viewed in *Eclipse*. The output data can be used by analysts who are performing defect sightings on a project.

Installing the TOIF Report View in Eclipse

To install the *TOIF Report View* in *Eclipse* perform the following steps.

1) Start Eclipse.
2) Choose the workspace location.

   This is home to the eclipse user/session data and also any projects which are created.
3) Close the welcome screen.
4) Click HELP in the tool bar menu and from the drop down menu select INSTALL NEW SOFTWARE....

   The *Available Software* dialog opens.
5) Click **Add...**.

   The *Install* dialog opens displaying the *Available Software* panel.
6) Click in the NAME: text field and enter a name for the new software source.

   For example, *TOIF Report*.
7) Click **Archive...**.

   The *Repository Archive* dialog opens.
8) Navigate the directories to find the location of the `TOIF_View_Updatesite.zip` file and then click **Open**.

   The *Repository Archive* dialog closes and the *Location:* text field in the *Add Repository* dialog is populated.
9) Click **OK** in the **Add Repository** dialog.

   The *Repository Archive* dialog closes and the *SFP/CWE* category appears under the *Name* column in the *Available Software* panel.

Note: The category is displayed only if the *Group items in category* check box is selected. Otherwise, the two features, *Extension Point for Stand Alone Report Tool* and *SFP/CWE Reporting* are displayed.

10) Click SFP/CWE to expand the category in the **Available Software** panel.

    Two features, *Extension Point for Stand Alone Report Tool* and *SFP/CWE Reporting* are displayed.

11) Click to select both of the features.

Note: For the stand-alone installation, both of these features are required.

12) Click **Next >**.

The *Install Details* panel is displayed.

13) Click **Next >**.

The *Review Licenses* panel is displayed.

14) Click **I accept the terms of the License Agreement** radio button in the **Install Details** panel.

This indicates that you agree to the license agreement.

15) Click **Finish**.

The *Install* dialog closes and the *TOIF Report View* installation starts. Once the installation is complete a dialog will appear requesting that Eclipse be re-started.

Note: If a dialog appears with a warning that the content is unsigned simply click *OK*. This allows the *TOIF Report View* installation to continue.

16) Click **Restart Now**.

Eclipse session will close and then re-open.

17) Click HELP in the Eclipse menu bar and from the drop down menu select ABOUT.

The *About Eclipse* dialog opens.

18) Click the **Installation Details** button.

The Eclipse Installation Details dialog opens.

19) Click the **Plug-ins** tab in the **Eclipse Installation Details** dialog and search the list of plug-ins for the `ReportView` plugin.

If the *Report View* plugin exists in the list of plugins then the *TOIF Report View* has been installed into Eclipse.

Importing Data into the TOIF Report View in Eclipse

To import the assimilated output data into the *TOIF Report View* in *Eclipse* perform the following steps.

1) Click FILE -> NEW -> PROJECT... in the Eclipse menu bar.

The *New Project* dialog opens allowing you to create a new project.

2) Click **General** folder to expand it and then select **Project** from the list

3) Click **Next>** in the **New Project** dialog.

The *New Project* dialog updates to display the Project panel.

4) Click in the PROJECT NAME: text field and type the name you want to give to the project.

   For example, *TOIF Project*.

5) Click **Finish** at the bottom of the **New Project** dialog.

   The project is displayed in the Navigator panel of the Eclipse session.

6) Right click on the project in the Navigator panel and from the drop down menu click IMPORT....

   The *Import* dialog opens.

7) Expand the SFP/CWE folder and navigate to IMPORT INTEGRATED SFP/CWE FILE.

8) Click IMPORT INTEGRATED SFP/CWE FILE.

   The *Import Integrated SFP/CWE File* is now selected.

Note: If the Assimilator was used to create a repository, select *Import SFP/CWE Repository*.

9) Click **Next>** in the **Import** dialog.

   The *Import* dialog updates.

10) Select the project that you just created in the list currently displayed in the Import dialog.

11) Click the **Browse…**.

    A dialog opens.

12) Navigate the directory to find the TOIF Data and select the *.kdm* file or repository depending on which option you selected when running the **TOIF Assimilator** (see page 13).

13) Click **Open**.

    The dialog closes.

14) Click **Finish**.

    The *Import* dialog closes and the *Import SFP/CWE Data* dialog appears to display the progress of importing the SFP/CWE data.

15) Click WINDOW -> SHOW VIEW -> OTHER... in the Eclipse menu bar.

    The *Show View* dialog opens

16) Click SFP/CWE folder and navigate to find TOIF REPORT VIEW.

17) Click TOIF REPORT VIEW to select it.

18) Click **OK**.

    The *Show View* dialog closes and after the model is finished loading, the *TOIF Report View* appears in the *Eclipse* List panel. The next step is to generate the defect data that will populate the *TOIF Report View*.

Chapter 7

Generate the Defect Data

To generate the defect data for viewing in the *TOIF Report View* in *Eclipse* perform the following steps.

1) Open Eclipse.
2) Click the project, that you added the assimilated output data to, in the Navigator panel.
3) Click PROJECT in the menu bar and from the drop down menu select SFP/CWE -> (RE)BUILD DEFECT MODEL.

A progress dialog opens to indicate that the defects are being loaded into the *TOIF Report View*. Once the progress dialog closes the defects can be displayed in the *TOIF Report View*.

If you want to view the source code of the file associated with a finding you need to first add the project sources to the applicable TOIF project in Eclipse. After adding the project sources you can double click on the finding in the *TOIF Report View* and the source code will be displayed in your default source code viewer.

Important: To view TOIF data in the *TOIF Report View* you have to perform the steps above every time you switch between TOIF projects or shutdown and re-start Eclipse (see "**Need to rebuild defect models to view TOIF project findings**" on page **Error! Bookmark not defined.**).

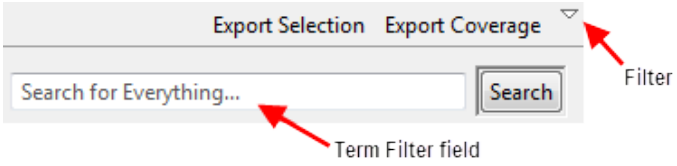
Chapter 8

TOIF Report View Interface

The following section provides the descriptions of the toolbar buttons and context-sensitive menu options in the *TOIF Report View* that can be used in performing a defect analysis on your project.

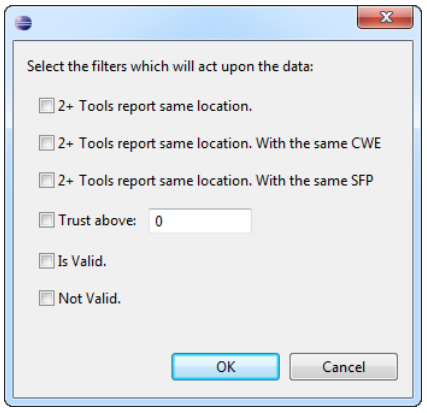
TOIF Report View toolbar

The TOIF Report View consists of a toolbar which is located in the top right corner contains the following buttons and fields:



Filter

Filters reduce the number of visible findings on the screen. The trust filter only shows findings with a trust above the set amount. The 2+ Tools filter options only displays findings where two tools found a finding in the same location, same location with same CWE, or same location with same SFP. The valid filter options displays the findings that have been marked as either IS A WEAKNESS or IS NOT A WEAKNESS.



Export Selection

Exports the selected elements to a .tsv format. This can then be imported into programs such as Excel. Note that this file is tab separated.

Export Coverage

Exports the entire data set as a Coverage Claims Representation (CCR). More information for this coverage report can be found at http://cwe.mitre.org/compatible/ccr.html.

Search and Term Filter Field

The *term filter field* is a search box for findings containing a specific term. Clicking the **Search** button executes the search on the terms included in CWE/SFP ids, Description contents, line numbers, or resource names of the findings. Only findings that contain the specific term(s) will be displayed in the TOIF View. Running the search without any terms in the term filter field will show all the findings in the data set.

TOIF Report View Context Sensitive Menu Options

Right clicking on data displayed in the *TOIF Report View* will display a drop down menu with the following options.

Not a Weakness

This marks that the finding is not actually a weakness.

Is a Weakness

This marks that the finding is a weakness.

Set Trust Level

This option is only available at the “Finding” level. It sets the level of trust for the selected finding. This level is propagated throughout the data set, marking any finding with the same CWE from the same tool with the specified value. Trust is an indication of how much faith the analyst has in the tools ability to accurately detect the defect.

Trace

This option is only available at the “Finding” level. If trace data is present, selecting this option will display the trace back as a dynamic menu which when clicked will take you to the various places within the code; tracing the route all the way to where the finding was generated.

Sorting

Provides options for sorting the data. The sorting options that are available are: Default Sorting, Sort by Count, and Sort by Trust.

More Information

This will take you to the mitre site for the selected CWE id.

CHAPTER 9

Known Limitations

Traceback displays numbers as reported by TOIF Adaptors

The traceback currently shows numbers as reported by the TOIF Adaptors unlike the code locations in the view list (which are normalized to kdm locations).

Glossary of Terms

Coverage Claims Representation

Coverage Claims Representation (CCR) is an XML document used for representing information about Common Weakness Enumeration.

Common Weakness Enumeration

Common Weakness Enumeration (CWE) is a software community project whose goal is to create a catalog of software weaknesses and vulnerabilities.

Software Fault Patterns

Software Fault Patters (SFP) is a generalized description of an identifiable family of computations:

- Described as patterns with an invariant core and variant parts
- Aligned with injury
- Aligned with operational views and risk through events
- Fully identifiable in code (discernible)
- Aligned with CWE
- With formally defined characteristics

Index