Oh SCAP Please User Guide

Theodore Olsauskas-Warren – U5195918
October 2014

Introduction

This document aims to be an end to end users guide for Oh SCAP Please (OSP). It will attempt to guide the user through the extraction and insertion process, data configuration and report generation. It will also look at how the user can interact with the final report to find the information relevant to their concerns.

Before You Begin

OSP has several external dependencies which are not included in the program itself. If you're reading this, you probably acquired the Python source OSP from the Github repository. Before you begin using OSP, there are three main external requirements: Python 2.6.x, Psycopg2 and a PostgreSQL installation.

OSP was developed with Python 2.6 primarily for maximum compatibility reasons, many distributions of Linux come with Python 2.6 automatically installed. If not, obtaining Python is usually easiest via your distribution of choice's package manager.

OSP uses the Psycopg2 library for interfacing with the PSQL database instance. Psycopg2 can be downloaded directly from the developer website at: http://initd.org/psycopg/ Alternatively, it can be installed by the Python package download manager, instructions can be found by following the aforementioned link. The most recent version of Psycopg2 is preferred (2.5.4), however OSP should function with any 2.5.x version.

For storing data, OSP uses PostgreSQL (PSQL), again this is for maximum compatibility. PSQL is often included in Linux distributions as either an extra module available during install (as in Ubuntu) or from the various package managers. The PSQL instance requires no special configuration to run and simply uses the default public schema of a database you specify. Information on acquiring PSQL and performing a basic install can be found at: http://www.postgresql.org/

Insertion and Extraction

The first step when using OSP is to extract information from your OpenSCAP result files and extract them into the database. As of writing, OSP supports both XCCDF and OVAL results files. It is important to note that these are the XML files OpenSCAP generates, and not the HTML visual report files.

Once you have assembled the XML files you wish to insert, OSP can be run in insersion mode via the following command:

```
python OSP.py -i -f <your files>
```

Other command line parameters are also accepted here, these enable you to specify the target database, the database user, log file location and other options. To see a full list of available parameters you can print the OSP command line argument information by using the command:

```
python OSP.py -h
```

Once extraction and insertion has completed, you should see a message printed to the terminal indicating that all database insertions have been completed. It is here that you should review the terminal output, as well as the log file (default location OSP.log in the folder from where these commands were run) for any errors that may have occurred. You may also connect manually to the database you selected (default is OSP) and confirm that insertion was successful, you may also review the data that OSP extracted by perusing the tables present in the database.

Configuring Data Scope

The OSP program has been designed to store large amounts of historical result data. Whilst you are able to filter finely through results in the generated HTML report, it is important that before generating the report you coarsely refine the data included in the HTML file. As the report OSP generates has been designed with portability in mind, all information extracted from the database is present within the file. Thus, to keep the file size manageable, it is recommended that you use the coarse refining so as to only include the relevant data.

This refining is performed by specifying refinement constrains in the configuration file. An example file is included with OSP called osp.cfg, though any file may be specified when generating the report. In this file you can narrow the data included in the report, this can be performed by specifying start and end dates from which to include data from, as well as other properties. More information regarding the configuration file is available in the aforementioned included example.

If you are dealing with smaller amounts of data, or all the data you have is relevant, then you will not need to perform this step and may simply ignore the configuration file.

Generating the Report

After inserting your data, and potentially creating your configuration file, the next step is to generate the HTML report. This is performed by running OSP in output mode by using the following command.

```
Python OSP.py -o <output file>
```

As per the insertion command, extra parameters can be added to the command, including specifying the location of a configuration file. For a full list, please refer to the command line help by running the aforementioned command.

After running this command, you should receive a message in your terminal regarding the successful creation of the report. At this point you should take time to review the messages contained in the log file to check for any issues.

At this point, you should have a HTML file containing the report created in the location you specified above. It should be noted again that only a single file is generated, and all of the data as specified by the configuration file is present within this file. This HTML file is therefore completely portable. There is only one caveat: some of the libraries used by the HTML file must be retrieved over the internet and thus the HTML file requires internet connectivity for full functionality.

Using the Generated Report

When you open your generated report you are greeted with the dashboard view, as seen in Figure 1.

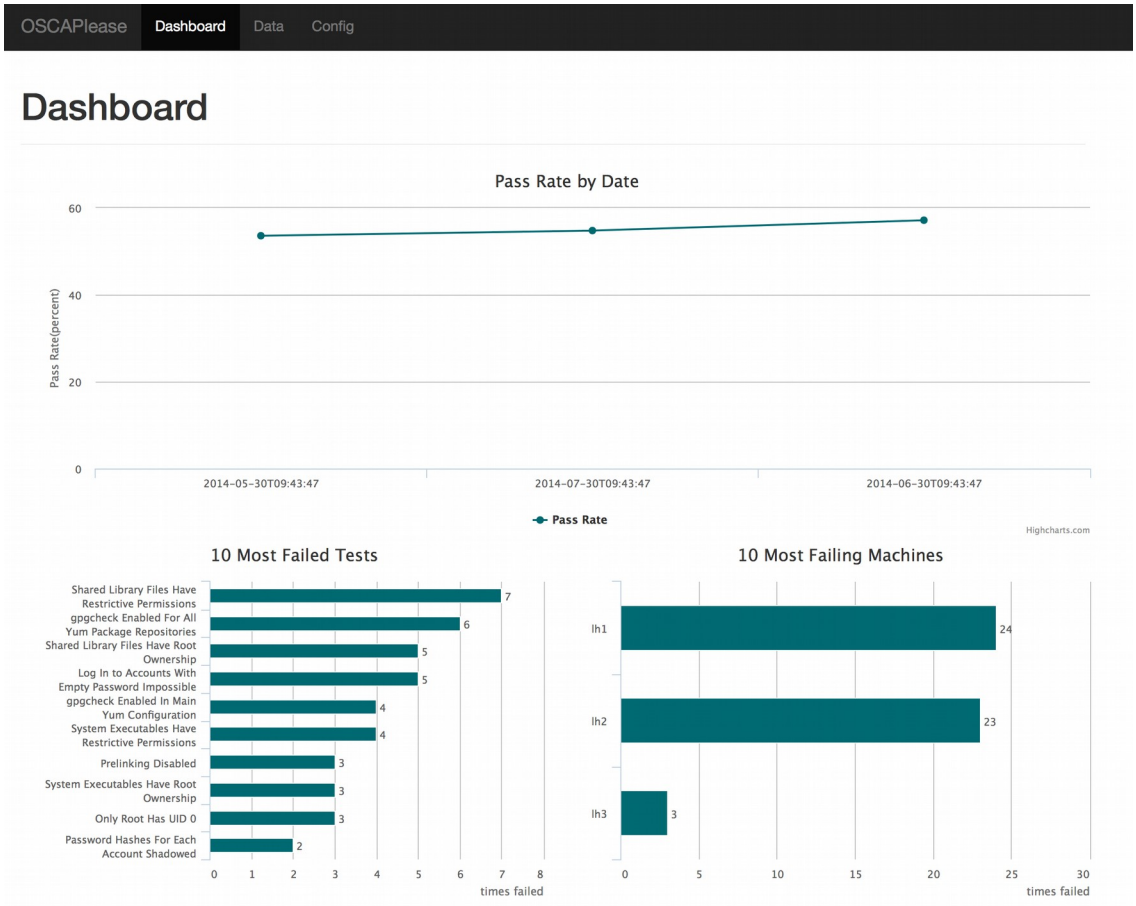

Figure 1: OSP Report Dashboard View

In this view you can instantly get a high level idea of your results. The large graph in the upper-centre displays the overall pass rate of all machines as a function of time, enabling you to view your security compliance history. The smaller graph in the lower left identifies which tests have the highest raw number of fails, whilst the one of the right shows which machines have the highest number of raw fails. These graphs should enable you to quickly identify compliance trouble areas.

Along the top you will notice a navigation bar, clicking on the tab marked “Data” brings you to the data view, as shown in Figure 2.

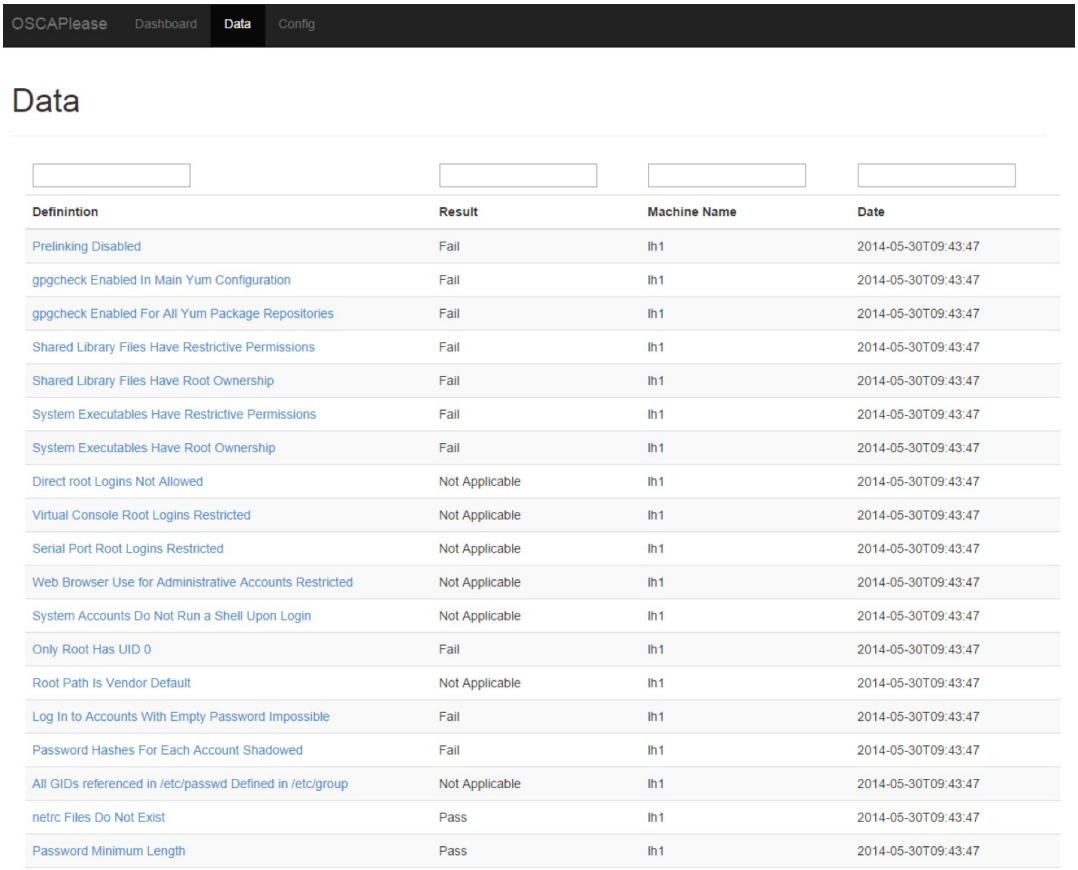

Figure 2: The OSP report data view

This view allows you to see all of the test results that OSP is using to generate the data visible on the dashboard. It shows the name of the test, the date it was run, the machine it was run on and the result of the test. This view is useful for drilling further down into the results, to identify exactly where specific compliance issues are occurring. In this view you are able to filter the data by using the text boxes above the data columns. Each text box filters the column directly below it, and each filter can be used concurrently, enabling you to exactly pinpoint specific test results.

By hovering over a test name, as show in Figure 3, you can also view the information of the definition used to run the test. Definitions contain more specific information about what the test was checking for, in some cases they can also contain resolution information. If you see a failed test, be sure to look at the definition information.

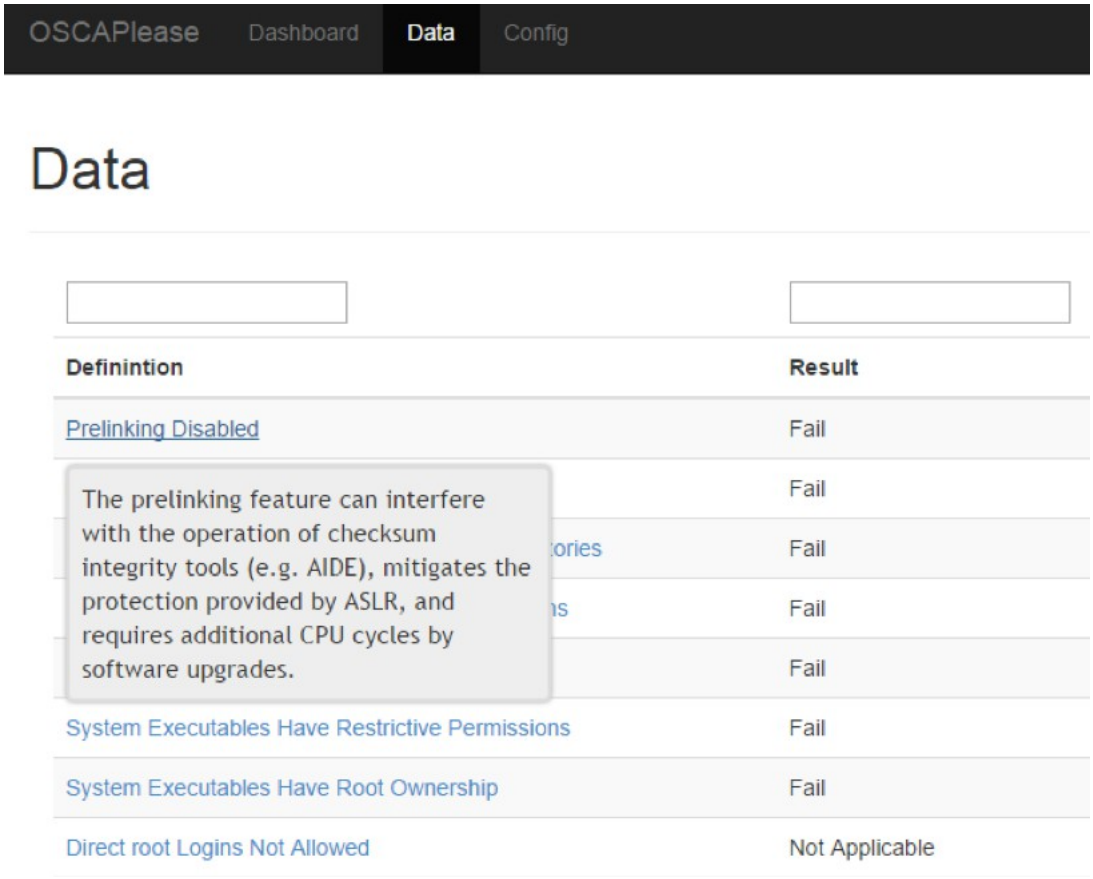

Figure 3: Definition information obtained by hovering over a test name

The final tab in the top navigation bar simply displays the configuration information present in the config file used to coarsely refine the report at creation time. This enables you to be sure exactly which information from the database has been included in the report.

Troubleshooting

If at any stage of operation, OSP should not behave as you should expect, you first course of action should to always check the log file, often there will be a simple configuration error preventing insertion of data or the creation of the report. If the log does not indicate the cause of the issue, try increasing the logging verbosity level (check the command line argument help to see how to do this). By increasing this to the maximum of 5, you should be able to see exactly where OSP is failing. If this still doesn't enable you to solve the issue, please feel free to create an issue on the Github repository. You can access the link by clicking the OSCAPlease name in the top left of the report, or directly at: https://github.com/ANU-RH2/oscap-please

Conclusion

By reading this document hopefully you have a basic, end to end understanding of how to use OSP to consolidate and report on your OpenSCAP results, enabling you to more effectively identify and fix compliance issues.