UNIVERSITY OF SOUTH WALES

Faculty of Computing, Engineering and Science

Computing Individual Project

Automate Manual Task for Penetration Testing

Brenno Candido, 13205978

July 15, 2015

Contents

1 Abstract **3**

2 Implementation **4**

2.1 Descriptor File 4

2.2 Expand the Descriptor File 5

2.3 Taskrunner 6

2.4 Variable 7

2.5 SMTP Plugin 8

3 Results **8**

4 Evaluation **8**

5 Conclusion **8**

1 Abstract

Penetration Testers usually have to make their own scripts when they are testing a determined system or service. The proposal of this project is to create a simple management console tool to speedup time during the manual process of a Penetration Test. This console application will be able to run a collection of predefined scripts, other console tool, and predefined commands in a modular (plug-in) manner. The main characteristics of the project are: easily adaptable and expandable due to a number of possible types of heterogeneous tests. The current project is focused on researching and creating a Simple Mail Transfer Protocol Plugin (SMTP Plugin).

2 Implementation

To reach the goals of this project, a python script called Taskrunner was developed as well as a xml structure was defined to be interpreted by the Taskrunner. This xml structure is called a descriptor file. It basically contains a number of tasks/checks desired to be executed by the script. The following session will discuss the main parts of the tool implemented i.e descriptor file and the script file. Firstly, what is the description file, how to interpret it and how to expand it. Secondly this report will explain the usage of the Taskrunner, how it works, and how the script will interact with the descriptor file.

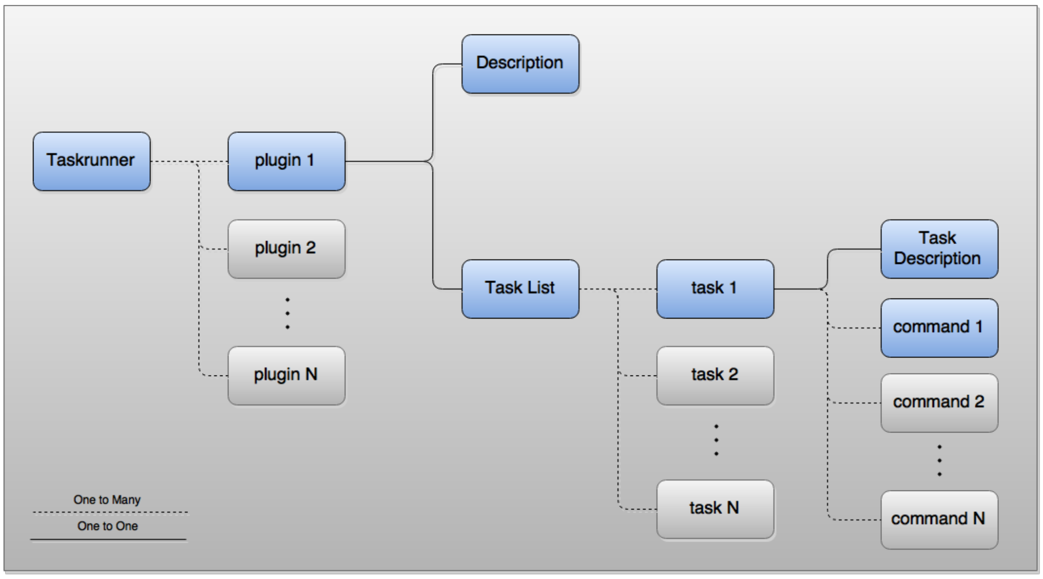
2.1 Descriptor File

The descriptor file was created to provide good usage and easy extensibility. This file contains plugins to be executed, and each plugin is composed by tasks, descriptions and commands. In order to achieve a better understand of the descriptor architecture, this section will explain the structure hierarchy and the element compositions.

The structure hierarchy is composed by a combination of elements. The main elements used are: plugin, task and command, and their composition are explained below:

- Each plugin has a plugin name, plugin description and a list of task;
- Each task has a description and one or more commands to be executed; and
- A command is defined by a shell command, a predefined script or other console tools

The structure of the descriptor file can be seen in the Figura 2.1.



Extensible Markup Language also known as XML provides a nice and clean way to describe the structure of the descriptor file. Although there are other possibles formats that fits this structure like JSON or other text-based format, XML is easy to read and understand, and it make easier for expand the functionalities such as add task/checks against HTTP Server, Web Servers, DNS Servers. This project will also provide a SMTP Plugin that will task/check against SMTP Servers. More information is available in the SMTP Plugin session.

2.2 Expand the Descriptor File

One of the biggest advantages of implementing the descriptor file in XML is due to the fact that XML provides a flexible structure to add and modify data. This facility allows Pen Testers to add and modify/check tasks in a simple way. For example, if the Pen Tester wants to add different tasks or add a new plugin to perform a range of different attacks against a Web server, an HTTP Server. The Pen Tester can also create a plugin that perform an only checkup to gain time during the penetration test. In order to achieve this asset, a simple XML file is shown below to provide a basic view of the Descriptor structure.

```
<taskrunner>
    <plugin name="myPlugin">
        <description> My Plugin Description </description>
        <taskList>
            <task>
                <description>My Task Description </description>
                <command> echo My Task </command>
            </task>
        </taskList>
    </plugin>
</taskrunner>
```

All Tasks has one or more shell commands to be executed, and it is nice to add a description of each task. It is recommended because Taskrunner provides a list of task descriptions, and this feature improves the task visualization. Plugins are composed by a list of tasks and a plugin description. The description is optional, however it is recommended to add as well as it is recommended for task descriptions. Therefore, to add a new task in a plugin, a task tag should be added inside the taskList tag, and the new task tag should be composed by a description and a command.

In almost the same way that tasks are added, it is possible to expand the Descriptor to contain many plugins by adding new plugins. To add a new plugin, a plugin tag should be added inside the taskrunner tag. The new plugin tag should contain a plugin description and a task list. A taskList tag is composed by many tasks tag.

2.3 Taskrunner

Taskrunner is a python script developed to interpret and execute the tasks specified in the Descriptor file. Python was chosen due to the fact that is easy and simple. Python also provides a fast development, and it is easy to read and to understand. Furthermore, python comes with many useful built-in tools like XML Parsers and String methods, which makes the development easier and faster than programming in a language without these built-methods, such as C Language. Taskrunner provides a command line interaction with console tools such as Terminal and Command Prompt. This section will focus on how make Taskrunner execute the plugins/tasks specified in described in the Descriptor file.

Taskrunner should be executed via a command line. In order to execute the script make sure that Taskrunner has permission to execute. To execute, a Descriptor file and a Plugin must be informed with option -D and option -P, respectively. In the example below, myDescriptor.xml is the Descriptor file's name, and myPlugin is the Plugin's name. It is important to notice that all plugins must have a name.

Taskrunner comes with functionalities to improve user experience and time performance. The plugin can also provide a description list of each task on the Descriptor file using option –list. It can be very useful for descriptor files which many people might have access, and when plugin/tasks are constantly modified. By default all the tasks are executed sequentially, and it can lead to bottleneck, since tasks are not linked to each other, for example a considerable number of task can take a time to finish. Therefore, to increase the time performance, the tasks can be executed in parallel using the option –parallel. Be aware that the –parallel option only parallelize tasks i.e. the commands inside a task is executed sequentially.

Command-line executions:

```
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin  --list
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin  --parallel
```

Descriptor:

```
<taskrunner>
  <plugin name="myPlugin"> ... </plugin>
</taskrunner>
```

2.4 Variable

Commands-line arguments such as IP addresses and Port numbers are used many time in some stage a penetration test. Those types of command-line arguments is an example of possible variable. In order to create reusable plugins in differents projects, the variable concept was introduced in the Descriptor file. Variables are a simple string substitution that allow the user pass values through command line.

Variables are defined into the command tag in the Descriptor file, and they are declared with predefined structure. The variable declaration in the Descriptor file is composed by two parts: a variable marker followed by a variable name. Firstly, a variable marker is the percent sign (%). The variable name could be any string of numbers or/and characters. Therefore, to actually use the variable at the time to execute the script a option with the variable name has to be passed as a command-line argument. To demonstrate a variable usage see the next example.

Command-line executions:

```
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin -PATH /etc
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin -ip
    192.168.1.1
$ ./taskrunner.py -D myDescriptor.xml -P myPlugin -PATH /etc
    -ip 192.168.1.1
```

Descriptor:

```
<command> ls %PATH </command>
<command> nmap %ip </command>
```

2.5 SMTP Plugin

3 Results

4 Evaluation

5 Conclusion

References