INFO0056-1 : Managing and Securing Computer Networks
Assignment : SNMP monitor (final version)

Jean-Francois Grailet & Raphaël Javaux

April 8, 2014

1 Introduction

This report will explain the final version of our SNMP monitor. Compared to the previous version, we added the information retrieval, the logging and a trap listener, along a supplementary Java class to send a trap to our monitor when the RAM usage is above 90%.

To make things simple, we decided to keep the same structure for the report as for our intermediate version; you should also find content similarities between both reports, but this should avoid you to read again our first report. The biggest changes are, of course, related to the features added to this final version.

2 General architecture of the monitor

Inside the src/ folder, you will now find 5 packages, described below (in alphabetical order).

2.1 discovery

This package was already present in our first version and is still dedicated to the discovery of SNMP agents in the given address range. The code has been reduced compared to our first version, due to several corrections and the introduction of the snmp package (see below). It also controls the retrieval of the variable tree (by calling a RemoteAgent method, as explained below). Otherwise, the key ideas remain the same as before.

2.2 monitor

Just like in our intermediate version, this package deals with command line parameters and launches the main thread of the software (DiscoveryThread) all along with the trap listener (TrapsListener).

2.3 retrieval

This package contains only one class : RemoteAgent. This class manages a detected agent, by providing methods to retrieve OIDs, maintain them, schedule their refresh and log them. It also implements the process of retrieving the entire OID tree. This process is triggered by the main monitor thread, not by the RemoteAgent object itself.

2.4 snmp

We introduced the snmp to reduce the amount of code related to SNMP4j.
It consists of a main abstract class, SNMPLink and 3 children classes, one for each SNMP version (SNMPv1Link, SNMPv2cLink and SNMPv3Link).
There is two additional interfaces, SNMPParameters and SNMPv3Parameters, the first one is associated with SNMPv1Link and SNMPv2cLink whereas the second ont is associated with SNMPv3Link. This helped us to considerably reduce the amount of code needed for the discovery, the information retrieval and the trap transmission processes.

2.5 traps

The traps package provides one class to listen for SNMP traps and to log them (*TrapsListener*) and one class to probe the RAM usage and enventually send a trap to the monitor (*TrapsDaemon*).

3 Comments about our implementation

3.1 Discovery of SNMP agents

The monitor uses a thread pool[1] to continously probe the entire address range. We use a thread pool to limite resource usage with large address ranges.

- For IPs on which no agent is known, a thread will be used to send first a SNMPv3 GETNEXT PDU to detect SNMPv3, then a SNMPv2c one, and finally a SNMPv1 one if none of the previous PDUs gets a response. If a response has been received, the entire OID tree of the agent will be retrieved ;

- For IPs on which an agent is already known from a previous scan, only the OID tree will be refreshed, as the project statement requires to detect new variables.

One the full address range has been tested and all threads terminated, the monitor waits 5 seconds before restarting the scanning.

3.2 Information retrieval

Information retrieval is done by the RemoteAgent class. This class has a timer to schedule the refresh of detected variables.
The delay between two updates of the same variable is computed using an adaptive algorithm, similar to the congestion avoidance algorithm of TCP Reno :

[1]The number of threads in the pool can be set by the user by introducing an eleventh input parameter when launching the monitor, see section 5.

- when the value of a variable changed, the delay for the next refresh is divided by two ;
- when the value of a variable did not change, the delay for the next refresh is multiplied by two ;
- if the refreshment of the variable failed, its update is rescheduled with the same delay. After 2 retries, the variable is removed from the monitored variables set and will not be rescheduled ;
- the new so computed delay must be between five seconds (lower limit) and one hour (upper limit).

3.3 Traps

The reception of traps is done through the implementation of the *CommandResponder* interface of SNMP4j.
The Java program which sends traps monitors the free RAM on the computer it is running on. When free RAM fells below 10% of the total available RAM, it sends a trap to the host given as command line argument.

4 Compilation

To compile our code, put the src/ folder and the build.xml file in your working directory and type the command ant. This will compile the monitor and produces a build/ folder with the compiled sources along a SNMPMonitor.jar file.

If you need, for some reason, to clean the working directory from our compiled sources, just type ant clean. This will delete the .jar as well as the build folder.

BEWARE : the build file assumes that the .jar providing the SNMP4j library is located at /opt/snmp4j-2.2.5/dist, just like on *Bee* in the Network Lab at Montefiore Institute. If you compile our code elsewhere, make sure to edit the build file accordingly.

5 Execution

To launch the monitor, use the following command :

```
java -jar Monitor.jar [0] ... [9]
```

Where [0] ... [9] are the input parameters, in the same order as in the assignment. In other words, if you wish to use the given parameters, use the following command[2] :

```
java -jar Monitor.jar ./logs 139.165.222.0/24 public
v3user authPriv SHA1 stupidAuthPassword AES
stupidPrivPassword 7005
```

To edit the number of maximum threads used for discovery, add parameter [10] at the end of the line (must be a positive integer). In other words, if you wish to use at most N = 128 threads (besides main thread), you will use the following command :

[2]The *./log* dir must exist.

```
java -jar Monitor.jar ./logs 139.165.222.0/24 public
v3user authPriv SHA1 stupidAuthPassword AES
stupidPrivPassword 7005 128
```

At launch, the monitor will write on the standard output a reminder of the input parameters it parsed, with a more convenient display, then write a new line on the standard output each time it detect a new agent in the address range or it receives a trap. Of course, the monitor will also update the logs in the given output directory when necessary, but will not display anything about OIDs in the console.

To launch the traps daemon, use the folowing command :

```
java -jar TrapsDaemon.jar <monitor host> <monitor port>
<community name>
```

If the monitor is running on the same host, use the following command :

```
java -jar TrapsDaemon.jar 127.0.0.1 7005 public
```

6 Feedback

We both needed to work around 35-40 hours each on this project.
The main difficulty was to go with the badly documented SNMP4j library. Most of our time fell in finding the correct usage of this library.
Given the amount of time needed to complete the project, the educational gain was pretty poor since this project is a simple network application as we have already done multiple times before.