Project 2 - Developing a web-based SNMP browser

Managing and securing computer networks

Floriane Magera (S111295)
Fabrice Servais (S111093)

April 15, 2015

1 Detecting the agents

To enhance the previous script, we took the one provided to us and modify it in order to allow it to write the results in the XML, as in the first part of the project.

2 Application

Our website is available at the page :

http://bee.run.montefiore.ulg.ac.be/~nms01/

To develop our application, we used the MVC pattern.

2.1 Model

The model is implemented by a SQLite database and retains the agents and the list of objects for each agent, which allows a quite fast retrieval of information, it is a cache. This database has a timeout of two hours, if we try to access it after that, it will retrieve the information again and update the cache:

- For the agents, the list is stored in the `/model` directory in a XML file, which is read before inserting the agents in the database. The XML file is indeed adding one more step, from retrieving the agents to store them in the database. An improvement could be to store them directly in the database, however, in the first part of the project, we stored them in the XML format. To keep coherence (and by lack of time), we kept this transitional and read from it to populate the database.
- To list of OID's contained in an agent are got by running a serie of `snmpgetnext`, beginning at the root. We did not used `snmpwalk` because it seems that it retrieved less objects than `snmpgetnext`. These objects are directly stored in the cache as the retrieval is finished. Same as the agents' cache, it is refreshed after two hours. In the database, an agent is always recognized by the 4-tuple (ip, port, version, sec_name). We also provide phpLiteAdmin to see in a more easier way the content of the database[1].

  When the list is returned to the controller, the array containing the OID's is exploded to form a tree-structured array, an "OID-tree like". For exemple, `array("1.2.1" => "a", "1.2.2" => "b")` becomes (more or less) `array("1" => array("2" => array("1" => "a", "2" => "b")))`. This function `explodeTree($array, $delimiter = '.', $baseval = false)` comes from an external source[2], making the development faster since we do not have to solve the problem ourselves.

2.2 Controller

The controller treats the client requests by retrieving from the model the information needed. In the case of the request of an OID value, it sends the SNMP request directly in the network.

[1] http://bee.run.montefiore.ulg.ac.be/~nms01/phpliteadmin.php - Password: admin
[2] http://kvz.io/blog/2007/10/03/convert-anything-to-tree-structures-in-php/

We tried to make the requests to the database as light as possible, that is why we only retrieve all the mib list for an agent, even if we know the OID of the requested object. The model transforms the list in a form of tree, which allows efficient access from the controller.

We also decided that the informations needed to identify an SNMP agent are its ip address, its port number, the SNMP version used and the community name. The other informations about SNMPv3 agents are kept local and never communicated to the client, it is a (maybe too) simple way to avoid communicating the passwords to the client.

2.3 View

The view is simple, it displays the data the controller collected.

3 Feedback

We think the projects were interesting, it allowed us to learn new things : we had never coded in python, neither heard about cron jobs. And creating a website is always a good exercice, as we don't do it often, but we think it is essential to have some experience in that field too.

The SNMP subject is sure important and helped us realise that it is not "simple" as its name could let us think, but we would also have enjoyed a subject more related to security. Even if we have the project of the course "Introduction to computer securiy".