Kind of guide to the Pandora FMS's log agent

Jose Navarro

January 19, 2007

Contents

Abstract

The Pandora 1.2 agent's code have been improved in order to increase its capabilities for monitoring and analyzing text files that grow and are rotated, like typically log files do. Perl regular expressions and code can be used in the design of modules. From this aproximation, every new line of the monitored text is considered an individual piece of information. Another secondary improvements have been done to Pandora FMS, like the use of regular expressions to define alerts on data string modules.

1 Notes

This chapter is documentation for developers of the Pandora project, and its main objective is to share information about the inners workings of the code. It also can be used as a installation and configuration guide till the official documentation is attached to the trunk branch of the project.

2 Introduction

Till version 1.2, pandora was able to treat several types of data, including numeric, incremetal numeric and strings. However, string data type lacks some funcionalities for monitoring log files.

The aim of this pack of code, or development branch, is to strength the capabilities of Pandora for monitoring text files that are increased and rotated with the time, like log files.

In particular, the main objectives of this development branch are:

- pandora agents should be able to analize new lines of a text log file, and only the new ones
- pandora agents should treat each new line as a independent element/data unit
- pandora agents need to be aware about rotation of log files in order to not lose information
- alerts capabilities on string data types has to be improved
- graphic representation capabilites on string data modules has to be improved

For a technical summary of new technical features, please refer to section 4 on the page 3.

3 Architecture

Funcionalities described in the Introduction chapter can be implemented in several ways. The approach presented in this document propose that the pandora agent can control log files, detect which new lines are created since its last execution, and process differents modules on EACH of the lines.

On this branch of development, pandora_agent.sh calls to another script just before copying data files to the server. This script is a simple perl script named pandora_agent_log.pl, that uses another file as a configuration file, pandora_agent_log.conf.

pandora_agent_log.pl and pandora_agent_log.conf are very similar to pandora_agent.sh and pandora_agent.conf. In the future, its funcionalities are intended to be included in the agent code. Now, they are presented sepparated just for clarity in the development.

In order to monitor only the new lines that are added to a text file between two executions of the agent, index files are used. This files contain a pointer to the last byte read in the last execution. If the index file does not exist, Pandora will create it, but *will not analyze lines in this first interaction.* This avoid overloading of the agent the first time that tries to analyze a log file with, maybe, thousands of lines. See the section of the index file for more information and more uses of this file.

4 Summary of technical features

4.1 line by line text file monitoring

The log agent can monitor every new line of a text file, executing all the modules associated. Every line of the log generates a XML data file that is introduced in the Pandora FMS system with the information of all the modules executed.

For more information, refer to the commands module_log, module_exec in the section 6.

4.2 rotating log file monitoring

The log agent detects when a log file has been rotated since de last execution, analyzing the new lines of the log file but also the lines not analyzed in the rotated file. This process is transparent to Pandora.

For more information, refer to the commands module_log_rotated in the section 6.

4.3 alerts on generic_data_string

Pandora FMS can now use alerts on generic_data_string modules, firing the alert when a perl regular expression is matched.

For more information, refer to the section 8.

4.4 storing even repeated module data

The default behaviour of Pandora is not to store repeated and consecutive data of a module in the database. This issue could be a problem when managing log files. A new feature has been development in order to store all the values returned by a module, even the repeated and consecutive ones. Note that this feature can apply to *any* type of module.

For more information, refer to the commands module_store_all_data in the section 6.

4.5 timestamp forging

By default, the timestamp of a data XML file is determined by the time in which the agent has been executed. A new feature has been added that allows to forge the timestamp of the XML file. Note that this feature can apply to *any* type of module.

For more information, refer to the commands module_log_timestamp in the section 6.

4.6 counters

The log agent analyze only the group of new lines appended to a log file between executions, but it analyze one by one. Counters allow to count the number of these lines that match a regular expressions.

For more information, refer to the commands module_log_counter in the section 6. Refer also to the Index file section 7

5 pandora_agent_log.pl

Some of the code used for this script, mainly the index files manegement, is based in the part of the code of pandora_server/bin/ pandora_snmpconsole.pl

this script performs the following actions in the following order:

- load the pandora_agent_log.conf configuration file, that is described later in this chapter
- for each log file to be monitored loops through the next steps
- loads or creates an index file. Each index file stores information regarding the state of the log file in the last execution of the agent. Indexes are stored in $PANDORA_HOME. If this file does not exists, it is created pointing to the end of the log file. Nothing more is done for this log file.
- the script make some checks over the log file to see if it has been rewritten and/or rotated. If it has been rotated, rotated log file is processed first to recover last lines.
- loops for the NEW lines of the log (since last agent's execution). For each line, a data file in data_out is created.
- each module associated with that log file is executed against the new log line, and data is written in the corresponding data file
- checksum is performed on data files
- index file is updated

6 pandora_agent_log.conf

the structure of this file is the same as the structure of pandora_agent.conf, but some extensions has been added.

- module_log [LOG FILE] : only the modules with 'module_log' are considered. [LOG FILE] is the file, path included, of the log to be analysed. Different modules can be associated to the same log file. A module can only be associated (for now) to a log file.

- module_log_timestamp : timestamp of the data file can be rewritten using this module. The overriding timestamp is the result of processing 'module_exec' on the log line. Modules with module_log_timestamp are not further considered as pandora modules. So, they require no name, description, data type, ...

- module_log_rotated [LOG FILE] : tells the agent what wil be the name of the rotated log file. Everytime that the agent detects a rotation in the main log file, it will analize the last lines of [LOG FILE]. You don't need to put it in every module associated with a log file: in one is enough.

- module_exec [EXPR] : expression to be executed on every new line for this module. For now, EXPR is a perl expression. You can use the variable $LINE to represent the new log line. F.ex.,
  '`module_exec $LINE =~ y/A-Z/a-z/; return $LINE;`' lowercase the log lines before be stored in pandora database.

- module_store_all_data : the default behaviour of pandora is not to store in the database repeated values captures by the agent. This options override that behaviour, forcing pandora to store ALL data. NOTE: please note that this parameter can be used with all kind of modules, not just module_log ones.

- module_log_counter : makes the current module a counter. A counter is a special module that: (a) does neather create a module in the database nor in the web console (b) counts how many lines from the current execution of the agent matches the expression in `module_exec`. For example, if the monitored log file /tmp/jarl.log has 5 new lines, this modules will return a number from 0 to 5 depending on the number of lines that match `module_exec`. The results are stored in the index file of the corresponding log. See the index file section for more information.

7 Index file

This section describes the functions and syntaxis of the index files used for the log agent.

7.1 objectives

Index files has the following main objectives:

- to store a pointer to the last line monitored in the last execution of the agent. Every time the agent executes, looks for this pointer and only analizes the new lines added.

- to store information that cannot be introduced directly in Pandora FMS, like counters. This information requires another agent to be treated and introduced in the system.

One index file *should* exist for every text file monitored. If this index file does not exist, the log agent will create one, *but will not analyze the existing lines.* This means that the default behaviour of the log agent the first time it is executed is:

- to jump all the lines of the monitored log without analyzing them, and
- to write a index file that makes that the agent will monitor *only the new lines* the next time it is executed

This behaviour is very useful the first time Pandora is executed to monitor existing huge log files, avoiding overloading the first time the log agent tries to analyze (maybe) thousands of lines.

7.2 syntax

The index file is a text file that is stored in the same folder as the agent script. Its name is the complete path to the log file, where the '/' and '\' are substituted for '_', and a '.index' extension is appended. For example, the index file corresponding to the text file '/home/user/teletubi.txt' should be '_home_user_teletubi.txt.index'.

The contents of this index file are organized in lines:

1. `last-byte-read 0 0` the last two zeros are not used already
2. `counter value-of-the-counter module-name` where the first 'counter' is literally this string, the value of the counter is the number of lines of the last agent execution matches module_exec of the module `module-name`. There is a line with this format for every counter present in the configuration file that has a value of counter different than zero.
3. `# log-file-name` the character '#' and the complete path to the log file

NOTE: please, note that the lines corresponding to the counters only appear when the value of the counter is greater than zero, and that are actualized with every execution of the log agent. If you want to use this information to build a pandora module, place this module in the agent log or in an agent that executes with the same frequency than it.

8 alert configuration on generic_data_string modules

It is possible to configure alerts on generic_data_string modules using perl regular expressions.

NOTE: please note that, although this feature has been development in this development branch, it can be used in all generic_data_string modules, not only in the module_log ones.

For the configuration of this feature, a new field has been added to the 'Alert association form' of the pandora console: 'Perl expression'. A succesful matching of the generic_data_string data will trigger an alert.

Regular expressions has to have Perl syntax, for example:

- `word` : matches the word 'word'
- `^#` : matches lines beginning with '#'
- `(d{1,3}\.){3}\d{1,3}` : matches IP addresses

9 Example

9.1 objectives

Next pandora_agent_log.conf performs the following actions:

- monitors log file /tmp/log1.log with two modules. First one, returns the line, and second one makes a simple substitution.
- all log lines of /tmp/log1.log are stored and displayed in database, even repeated and consecutives ones. for /tmp/log1.log, a rotated file is configured, /tmp/log1.log.0
- /tmp/log2.log log file is also monitored. A generic_data module is configured.
- for /tmp/log2.log, a fixed timestamp is forced, so all data is registered in database as captured '2006/9/25 1:1:1'
- for /tmp/log2.log, create a counter that counts how many lines contain the letter `a`, and a module named 'modcountlog2' that introduces this information in Pandora.

9.2 pandora_agent_log.conf

```
module_begin
module_name log1
module_descripcion log 1 log file
module_log /tmp/log1.log
module_log_rotated /tmp/log1.log.0
module_store_all_data
module_type generic_data_string
module_exec return $LINE
module_end

module_begin
module_name log1_subst
module_descripcion simple substitution in log 1
module_log /tmp/log1.log
```

```
module_type generic_data_string
module_exec $LINE =~ s/o/X/g; return $LINE
module_end

module_begin
module_name log2
module_description log2 - generic_data
module_log /tmp/log2.log
module_type generic_data
module_exec return $LINE;
module_end

module_begin
module_log /tmp/log2.log
module_log_timestamp
module_exec return '2006/9/25 1:1:1';
module_end

module_begin
module_log /tmp/log2.log
module_log_counter
module_name countlog2
module_description counter on log2. This is not considered a module for Pandora.
module_exec return ($LINE=~/a/)?1:0
module_end

module_begin
module_name modcountlog2
module_description density of lines containing the letter a
module_type generic_data
module_exec grep countlog2 _tmp_log2.log.index | cut -d' ' -f2
module_end
```

10 Installation

Till the merging of this development branch to the trunk code of Pandora FMS, some tricks and tips are needed for the installation process. This installation process consist on this steps:

10.1 Installing Pandora FMS 1.2

The code of the svn is based in Pandora FMS 1.2, so begin installing this. The svn branch for the log agent does not contain the entire system due for performance reasons and, let's say it, inexperience and lazyness of myself.

10.2 Substitute some branches

The relevant branches of the log agent (modified) are:

- pandora_console
- pandora_agents/linux
- pandora_server

10.3 Follow the normal installation process

ditto

10.4 Notes on database structure

For the agent log to work properly, two new fields have been added to the database (with respect to Pandora FMS 1.2). For your information, these are the two tables affected:

field ‘perl_expr’ in talerta_agente_modulo

```
# Table: 'talerta_agente_modulo'
#

CREATE TABLE `talerta_agente_modulo` (
  `id_aam` int(11) unsigned NOT NULL auto_increment,
  `id_agente_modulo` int(11) NOT NULL default '0',
  `id_alerta` int(11) NOT NULL default '0',
  `al_campo1` varchar(255) default '',
  `al_campo2` varchar(255) default '',
  `al_campo3` mediumtext default '',
  `descripcion` varchar(255) default '',
  `dis_max` bigint(12) default NULL,
  `dis_min` bigint(12) default NULL,
  `time_threshold` int(11) NOT NULL default '0',
  `last_fired` datetime NOT NULL default '0000-00-00 00:00:00',
  `max_alerts` int(4) NOT NULL default '1',
  `times_fired` int(11) NOT NULL default '0',
  `module_type` int(11) NOT NULL default '0',
  `min_alerts` int(4) NOT NULL default '0',
  `internal_counter` int(4) default '0',
  `perl_expr` text default NULL,
  PRIMARY KEY  (`id_aam`)
) TYPE=InnoDB;
```

field ‘store_all_data’ in tagente_modulo

```
# Database: pandora
# Table: 'tagente_modulo'
#
CREATE TABLE `tagente_modulo` (
  `id_agente_modulo` bigint(100) unsigned NOT NULL auto_increment,
  `id_agente` int(11) NOT NULL default '0',
  `id_tipo_modulo` int(11) NOT NULL default '0',
  `descripcion` varchar(100) NOT NULL default '',
  `nombre` varchar(100) NOT NULL default '',
  `max` bigint(20) default '0',
  `min` bigint(20) default '0',
  `module_interval` int(4) unsigned default '0',
  `tcp_port` int(4) unsigned default '0',
  `tcp_send` varchar(150) default '',
  `tcp_rcv` varchar(100) default '',
  `snmp_community` varchar(100) default '',
  `snmp_oid` varchar(255) default '0',
  `ip_target` varchar(100) default '',
  `id_module_group` int(4) unsigned default '0',
  `flag` tinyint(3) unsigned default '0',
  `store_all_data` bool default '0',
  PRIMARY KEY  (`id_agente_modulo`)
) TYPE=InnoDB;
```