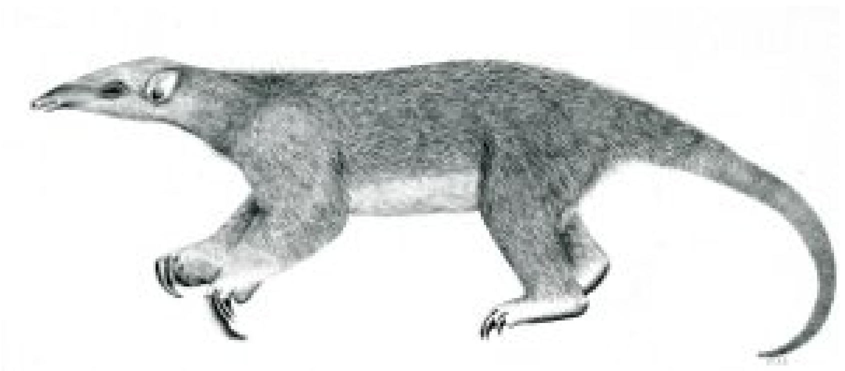
Tranalyzer2

regex_pcre

PCRE



Tranalyzer Development Team

Contents

1 regex_pcre **1**
1.1 Description 1
1.2 Dependencies 1
1.3 Configuration Flags 1
1.4 Flow File Output 3
1.5 Plugin Report Output 3

1 regex_pcre

1.1 Description

The regex_pcre plugin provides a full PCRE compatible regex engine.

1.2 Dependencies

1.2.1 External Libraries

This plugin depends on the **pcre** library.

Ubuntu: `sudo apt-get install libpcre3-dev`

OpenSUSE: `sudo zypper install pcre-devel`

Mac OS X: `brew install pcre`

1.2.2 Other Plugins

If `LABELSCANS=1`, then this plugin requires the `tcpFlags` plugin.

1.2.3 Required Files

The file `regexfile.txt` is required. See Section 1.3.3 for more details.

1.3 Configuration Flags

1.3.1 regfile_pcre.h

The compiler constants in *regfile_pcre.h* control the pre-processing and compilation of the rule sets supplied in the regex file during the initialisation phase of Tranalyzer.

Name	Default	Description
`RULE_OPTIMIZE`	0	0: No opt rules allocated 1: Allocate opt rule structure & compile regex
`REGEX_MODE`	`PCRE_DOTALL`	Regex compile time options

1.3.2 regex_pcre.h

The compiler constants in *regex_pcre.h* control the execution and the output the rule matches.

Variable	Default	Description	Flags
`EXPERTMODE`	0	0: Alarm with highest severity: class type & severity, 1: full info	
`PKTTIME`	0	0: no time, 1: timestamp when rule matched	
`LABELSCANS`	0	0: No scans, 1: label scans (depends on `tcpFlags`)	
`MAXREGPOS`	30	Maximal # of matches stored / flow	
`OVECCOUNT`	1	regex internal: maximal # of regex output vectors	

Variable	Default	Description	Flags
REXPOSIX_FILE	"regexfile.txt"	Name of regex file under *./tranalyzer/plugins*	

1.3.3 regexfile.txt

The *regexfile.txt* file has the following format:

```
# ID Predecessor Flags ANDMask  ANDPin ClassID Severity Sel Dir Proto srcPort   dstPort offset
    Regex
# single rule
1   0   0x80    0x0000  0x0000  15  3   0x8b    0x0001  6   0   80  0   \x6A.{1,}\x6B\x3C\x24\x0B\
    x60\x6A.*
# single rule
3   1   0x80    0x0000  0x0000  15  3   0x82    0x0001  6   0   80  8   \x31\xDB\x8D\x43\x0D\xCD\
    x80\x66.*\x31
# root rules to following tree
202 0   0x11    0x0000  0x0000  20  4   0x41    0x0001  6   0   80  20  ^http
203 0   0x10    0x0000  0x0000  20  4   0x41    0x0001  6   0   80  20  GET
# sucessors and predesessors
204 202 0x01    0x0000  0x0001  43  2   0x85    0x0001  6   0   445 0   Volume Serial Number
204 203 0x40    0x0000  0x0002  40  2   0x8f    0x0001  6   666 666 0   (?i)Command completed(?-i)
# successors 20t5 & 205 to 204 AND ruleset
205 204 0x81    0x0003  0x0000  40  3   0x00    0x0001  0   0   20  0   ^get .*porno.*
206 204 0x80    0x0002  0x0000  35  3   0x00    0x0000  0   0   21  0   ^FTP
```

Lines starting with a '#' denote a comment line and will be ignored. All kind of rule trees can be formed using rules also acting on multiple packets using different ID's and Predecessor as outlined in the example above. Regex rules with the same ID denote combined predecessors to other rules. Default is an OR operation unless ANDPin bits are set. These bits denote the different inputs to a bitwise AND. The output is then provided to the successor rule which compares with the ANDMask bit field whether all necessary rules are matched. Then an evaluation of the successor rule can take place. Thus, arbitrary rule trees can be constructed and results of predecessors can be used for multiple successor rules. The variable Flags controls the basic PCRE rule interpretation and the flow alarm production (see the table below), e.g. only if bit eight is set and alarm flow output is produced. ClassID and Severity denote information being printed in the flow file if the rule fires.

Flags	Description
2^0 (=0x01)	PCRE_CASELESS
2^1 (=0x02)	PCRE_MULTILINE
2^2 (=0x04)	PCRE_DOTALL
2^3 (=0x08)	PCRE_EXTENDED
2^4 (=0x10)	Internal state: successor found
2^5 (=0x20)	Internal state: predecessor matched
2^6 (=0x40)	Preserve alarm in queue for later use
2^7 (=0x80)	Print alarm in flow file

The Sel column controls the header selection of a rule in the lower nibble and the start of regex evaluation in the higher nibble. The position of the bits in the control byte are outlined below:

Sel	Description
2^0 (=0x01)	Activate dir field
2^1 (=0x02)	Activate L4Proto field
2^2 (=0x04)	Activate srcPort field
2^3 (=0x08)	Activate dstPort field
2^4 (=0x10)	Header start: Layer 2
2^5 (=0x20)	Header start: Layer 3
2^6 (=0x40)	Header start: Layer 4
2^7 (=0x80)	Header start: Layer 7

The higher nibble selects which flow direction (A=0 or B=1), protocol, source and destination port will be evaluated per rule, all others will be ignored. The `dir` field might contain other bits meaning more selection options in future. The `offset` column depicts the start of the regex evaluation from the selected header start, default value 0. The `Regex` column accepts a full PCRE regex term. If the regex is not correct, the rule will be discarded displaying an error message in the Tranalyzer report.

1.4 Flow File Output

The `regex_pcre` plugin outputs the following columns:

Column	Type	Description	Flags
`RgxCnt`	U16	Regexp match count	
`RgxClTyp`	U8	Classtype	`EXPERTMODE=0`
`RgxSev`	U8	Severity	`EXPERTMODE=0`
`RgxN_B_RID_Amsk_F_CT_Sv`	R(4xU16_H8_2xU8)	Packet, byte position, regfile ID, AND mask, flags, classtype, severity	`EXPERTMODE=1&& PKTTIME=0`
`RgxT_N_B_RID_Amsk_F_CT_Sv`	R(TS_4xU16_H8_2xU8)	Time, packet, byte position, regfile ID, AND mask, flags, classtype, severity	`EXPERTMODE=1&& PKTTIME=1`

1.5 Plugin Report Output

The following information is reported:

- Number of alarms