NAME

rarc – **ra** client resource file.

SYNOPSIS

rarc

DESCRIPTION

Ra* clients will open this file if its in the users $HOME directory, or in the $ARGUSHOME directory, and parse it to set common configuration options. All of these values will be overriden by options set on the command line, or in the file specified using the '-F conffile' option.

Values can be quoted to make string denotation easier, however, the parser does not require that string values be quoted. To support this, the parse will remove " (double quote) characters from input strings, so do not use this character in strings themselves.

Values specified as "" will be treated as a NULL string, and the parser will ignore the variable setting.

RA_ARGUS_SERVER

All ra* clients can attach to a remote server, and collect argus data in real time. This variable can be a name or a dot notation IP address. Optionally you can specify a port number using a ':' and then providing the port number desired.

RA_ARGUS_SERVER=localhost:561

PID FILE SUPPORT

Any ra* program can generate a pid file, which can be used to control the number of instances that the system can support.

Creating a system pid file may require priviledges that may not be inappropriate for all cases. By specifying **RA_PID_PATH**, you can create personal pid files that will enforce your own policy for your own use of the ra* programs.

When configured to generate a pid file for a ra* program, if a file called ra*.pid (where ra* is the name of the program in question) exists in the RA_PID_PATH directory, and a program exists with a pid that matches the one contained in the file, then the program will not start. If the pid does not exist, then the ra* program replaces the value in the file, with its own pid. If a pid file does not exist, then the ra* program will create it in the RA_PID_PATH directory, if it can. The end result is that the system will support only one instanace of the program, based on name, running at a time.

The default value is to not generate a pid. The default path for the pid file, is /var/run.

No Commandline equivalent

RA_SET_PID="no"
RA_PID_PATH="/var/run"

RA_OUTPUT_FILE

All ra* clients can support writing output as Argus Records into a file or stdout. Stdout is specified as '-'.

RA_OUTPUT_FILE="filename"

RA_TIMERANGE

All ra* clients can support input filtering on a time range. The format is:
timeSpecification[-timeSpecification]

where the format of a timeSpecification can be:
[[[yy/]mm/]dd.]hh[:mm[:ss]]
[yy/]mm/dd

RA_TIMERANGE="55/12/04.00:00:01-55/12/04.23:59:59"
RA_TIMERANGE="12/04-12/05"

RA_RUN_TIME

All ra* clients can support running for a number of seconds, while attached to a remote source of argus data. This is a type of polling. The default is zero (0), which means run indefinately.

RA_RUN_TIME=0

RA_PRINT_MAN_RECORDS

Specify if ra* clients should print management records by default. This does not affect management record processing, nor down stream management record propagation.

Commandline equivalents: -M [no]man

RA_PRINT_MAN_RECORDS=no **RA_PRINT_EVENT_RECORDS**=no

RA_PRINT_LABELS

Most ra* clients are designed to print argus records out in ASCII, with each client supporting its own output formats. For ra() like clients, this variable will generate column headers as labels. The number is the number of lines between repeated header labeling. Setting this value to zero (0) will cause the labels to be printed once. If you don't want labels, comment this line out, delete it or set the value to -1.

RA_PRINT_LABELS=0

RA_FIELD_DELIMITER

Most ra* clients are designed to print argus records out in ASCII, with each client supporting its own output formats. For ra() like clients, this variable can overide the default field delimiter, which are variable spans of space (' '), to be any character. The most common are expected to be '' for tabs, and ',' for comma separated fields.

RA_FIELD_DELIMITER=','

RA_PRINT_NAMES

For **ra(1)** like clients, this variable will control the translation of various numbers to names, such as address hostnames, port service names and/or protocol names. There can be a huge performance impact with name lookup, so the default is to not resolve hostnames.

RA_PRINT_NAMES=port

Other valid options are **none** to print no names, **proto** to translate the protocol names, **port** to translate port names, and **all** to translate all the fields. An invalid option will default to **port**, silently.

RA_CIDR_ADDRESS_FORMAT

Use this variable to specify whether **ra()** clients, when printing numeric IP addresses, will print them as CIDR addresses, or not. CIDR notation is constructed from the IP address and the prefix size, the latter being the number of leading 1 bits of the routing prefix. The IP address is expressed according to the standards of IPv4 or IPv6. It is followed by a separator character, the forward slash (/) character, and the prefix size expressed as a decimal number.

Argus IPv4 data contains the CIDR mask length, when its less than 32, and ra* programs will by default provides the "/masklen" suffix when the mask is less than 32.

This maybe confusing for some data processors, which would rather not see the "/masklen" never, or all the time. Use this option to specify changes in the default printing stratgy.

Accepatable values for this variable are:
"no" - do not provide the CIDR mask length (legacy mode) [default]
"yes" - print CIDR mask length when less than 32
"strict" - always print CIDR mask length

RA_CIDR_ADDRESS_FORMAT="no"

RA_ASN_PRINT_FORMAT

All ra() clients can print and process AS Numbers that have been added to the records through metadata labeling, or were a part of the original Netflow to argus conversion process..

RFC 5396 specifies 3 formats for representing AS Numbers, and all 3 are acceptable formats. These format are:
"asplain" - 2 and 4-byte ASNs are printed as decimal integers.
"asdot+" - 2 and 4-byte ASNs are printed using a dot notation.
"asdot" - 2 byte ASNs are printed as decimal, and 4-byte ASNs
are printed using a dotted notation..

The default is 'asplain'.

No Commandline equivalent

RA_ASN_PRINT_FORMAT="asplain"

RA_PRINT_RESPONSE_DATA

For ra() like clients, this variable will include the response data that is provided by Argus. This is protocol and state specific.

RA_PRINT_RESPONSE_DATA=no

RA_PRINT_UNIX_TIME

For ra() like clients, this variable will force the timestamp to be in Unix time format, which is an integer representing the number of elapsed seconds since the epoch.

RA_PRINT_UNIX_TIME=no

RA_TIME_FORMAT

For ra() like clients, the format that is used to print timestamps, is based on the strftime() library call, with an extension to print fractions of a sec using "%f". The default is "%T.%f". You can overide this default time format by setting this variable. This string must conform to the format specified in strftime(). Malformed strings can generate interesting output, so be aware with this one, and don't forget the '.' when doing fractions of a second.

RA_TIME_FORMAT="%T.%f"

RA_TZ

The timezone used for timestamps is specified by the tzset() library routines, and is normally specified by factors such as the TZ environment variable found on most machines. You can override the TZ environment variable by specifying a time zone using this variable. The format of this string must conform to the format specified by tzset(3).

RA_TZ="EST5EDT4,M3.2.0/02,M11.1.0/02"
RA_TZ="PST8PDT"

RA_USEC_PRECISION

For ra() like clients, this variable is used to override the time format of the timestamp. This variable specifies the number of decimal places that will be printed as the fractional part of the time. Argus collects usec precision, and so a maximum value of 6 is supported. To not print the fractional part, specify the value zero (0).

RA_USEC_PRECISION=6

RA_USERDATA_ENCODE

Argus can capture user data, and the argus clients can print, merge, filter, and strip user data from argus records. When printing out the user data contents, using tools such as ra.1, the type of encoding used to print the buffers can be specified here. This is available because many user data buffers are not printable text, and other representations may be more appropriate.

Supported values are "Ascii", "Obfuscate", "Hex", "Encode32" or "Encode64". The default is "Ascii".

Obfuscate is an extension to the Ascii print, that attempts to over-write plain text passwords, encountered in the user data, with 'x's.

Commandline equivalent: -M printer=<printer>

RA_USERDATA_ENCODE=Ascii

RA_FILTER

You can provide a filter expression here, if you like. It should be limited to 2K in length. The default is to not filter. See ra(1) for the format of the filter expression.

RA_FILTER=""

SASL SUPPPORT

When argus is compiled with SASL support, ra* clients may be required to authenticate to the argus server before the argus will accept the connection. This variable will allow one to set the user and authorization id's, if needed. Although not the best practice, you can provide a password through the RA_AUTH_PASS variable. If you do this, you should protect the contents of this file. The format for this variable is:

RA_USER_AUTH="user_id/authorization_id"
RA_AUTH_PASS="password"

The clients can specify a part of the negotiation of the security policy that argus uses. This is controlled through the use of a minimum and maximum allowable protection strength values. Set these variable to control this policy.

RA_MIN_SSF=0
RA_MAX_SSF=128

RA_DEBUG_LEVEL

If compiled to support this option, ra* clients are capable of generating a lot of use [full | less | whatever] debug information. The default value is zero (0).

RA_DEBUG_LEVEL=0

RA_CONNECT_TIME

Some ra style clients use a non-blocking method to connect to remote data sources, so the user many need to control how long to wait if a remote source doesn't respond. This variable sets the number of seconds to wait. This number should be set to a reasonable value (5 < value < 60). The default value is 10 seconds.

RA_CONNECT_TIME=10

RA_SORT_ALGORITHMS

Many ra* programs sort records as a part of their function. Programs like rasort.1, providing explicit command-line options to specify the sort algorithms and their order, using the

Use this configuration directive to specify the default sorting algorithm table for your ra* programs. The default sort algorithm is record start time "stime".

RA_SORT_ALGORITHMS="stime "

RA_TIMEOUT_INTERVAL

Some ra* clients have a timeout based function. Ratop, as an example, times out flows and removes them from screen at a fixed interval. This variable can be set using the RA_TIMEOUT_INTERVAL variable, which is a float in seconds. 60.0 seconds is the default.

RA_TIMEOUT_INTERVAL=60.0

RA_UPDATE_INTERVAL

Some ra* clients have an interval based function. Ratop, as an example, can refresh the screen at a fixed interval. This variable can be set using the RA_UPDATE_INTERVAL variable, which is a float in seconds. 0.5 seconds is the default.

RA_UPDATE_INTERVAL=0.5

RA_PRINT_ETHERNET_VENDORS

All ra* clients have the ability to print vendor names for the vendor part of ethernet addresses that are in flow records. ra* programs get its strings for the ethernet vendors using Wireshark 'manuf' files. One is provided with the distribution, and installed into /usr/local/argus.

No Commandline equivalent

RA_PRINT_ETHERNET_VENDORS="no"
RA_ETHERNET_VENDORS="/usr/local/argus/wireshark.manuf.txt"

RA_DELEGATED_IP

All ra* clients have the ability to print country codes for the IP addresses that are in a flow record. Country codes are generated from the ARIN delegated address space files. Specify the location of your DELEGATED_IP file here.

No Commandline equivalent

RA_DELEGATED_IP="/usr/local/argus/delegated-ipv4-latest"

RA_RELIABLE_CONNECT

All ra* clients can reliably connect to remote data sources. This causes the ra* program to try to reconnect to lost remote sources every 5 seconds, indefinately. This causes ra* program to not terminate but retry connection attempts when they fail.

This feature is implemented using threads, and so threads support must be compiled in.

No Commandline equivalent

RA_RELIABLE_CONNECT=no

MYSQL SUPPORT

Many ra* clients can connect and use a MySQL database, either reading for writing. This may require references to remotes database hosts, databases, tables, and mysql account names and passwords.

Default values for these variables can be set here. support must be compiled in.

Commandline equivalents:
-r mysql://[username[:password]@]hostname[:port]/database/tablename
-w mysql://[username[:password]@]hostname[:port]/database/tablename
-u username:password

RA_DATABASE="argus"
RA_DB_TABLE="table"

RA_DB_USER="carter"
RA_DB_PASS="whatever"

Those ra* clients that can create database tables may need to specify a table type or rather, a database engine other than the defaul, MyISAM.

Commandline equivalents:
-M mysql_engine=tableType
Current tableTypes are
MyISAM
InnoDB
Merge
Memory
Archive
NDB
Federated
CSV

MYSQL_DB_ENGINE="MyISAM"

COLOR SUPPORT

For ra* programs that use curses, these variables defined color schemes and color assignments.

Argus uses a sixteen color palette, with 8 monotone and 8 accent colors, plus 16 colors of gray. Currently these color values are hard coded. New versions should allow you to provide color definitions for all internal values using a 256 Xterm color wheel, to assign foreground and background colors. But we're not there yet

RA_COLOR_SUPPORT="yes"
RA_COLOR_CONFIG="/usr/carter/.racolor.conf"

DIRECTION SUPPORT

Many ra* clients process flow records based on source and destination properties. TCP and UDP ports values can be used to assign direction, and are best used for well-known ports (< 1024), values that are in the /etc/services defintions, and the reserved ports (> 1023, < 49151).

The syntax is:
RA_PORT_DIRECTION="services"
RA_PORT_DIRECTION="services,wellknown"
RA_PORT_DIRECTION="services,wellknown,registered"

We recommend the wellknown and services options, as they are a bit more discriminating. If there are ports that you know are services that are in the registered port range, we suggest that you add them to your /etc/services file rather than include the registered port range; only because the registered range is so large. However, this option is applied only to flow in which the direction is ambiguous, and as such, corrections based on the logic should have minimum effect on analytics.

RA_PORT_DIRECTION="services,wellknown"

Sites use locality for a number of features, such as access control, and this support is intended to support visualization, and analytics.

Currently, you can identify a collection of IP addresses that represent RA_LOCAL, and are specified using an iana-address-file formatted file. (See ralabel.conf)

RA_LOCAL="/usr/local/argus/local.addrs"

When locality information is available, programs like ra(), and as the assignement of source when there is ambiguity in the flow record as to who is the actual initiator or receiver of the flow.

When locality information is available, programs like ra(), and ratop() can use that information to make display decisions, such

RA_LOCAL_DIRECTION provides the logic for using the locality information to assign flow direction. You can force the local address to be either the source (src) or the destination (dst).

The syntax is:
RA_LOCAL_DIRECTION="local:src"
RA_LOCAL_DIRECTION="local:dst"

RA_LOCAL_DIRECTION="suggest:src"
RA_LOCAL_DIRECTION="force:src

COPYRIGHT

Copyright (c) 2000-2014 QoSient. All rights reserved.

SEE ALSO

ra(1)