NAME

argus – audit record generation and utilization system

SYNOPSIS

argus [**options**] [**filter expression**]

COPYRIGHT

DESCRIPTION

Argus is a data network transaction auditing tool that categorizes and tracks network packets that match the libpcap filter *expression* into a protocol-specific network flow transaction model. **Argus** reports on the transactions that it discovers, as periodic network flow data, that is suitable for historical and near real-time processing for forensics, trending and alarm/alerting.

Designed to run as a daemon, **argus** reads packets directly from a network interface, classifies the packets into network transacations and appends the resulting network flow data to a log file or open socket connected to an **argus** client (such as **ra(1)**). **Argus** can also read packet information from **tcpdump(1)** , **snoop(1)** , **NLANR's Moat Time Sequence Header** or **Endaces ERF** raw packet files. **Argus** can also be configured to append its transaction logs to stdout.

Argus can provide address based access control for its socket connection facility using **tcp_wrappers** , and it can provide strong authentication and confidentiality protection using **SASL2** technology. Refer to package documentation to enable each of these services.

OPTIONS

–A Generate application byte metrics in each audit record.

–b Dump the compiled packet-matching code to stdout and stop. This is used to debug filter expressions.

–B <addr> Specify the bind interface address for remote access. Acceptable values are IP version 4 addresses. The default is to bind to INADDR_ANY address.

–c <dir> Specify a chroot directory to use after privilege access is complete. Must be super user to use this option.

–C Run argus in control plane capture mode. This sets the interface packet snap length to capture full packets, and to enable detailed flow tracking for supported control plane protocols.

–d Run argus as a daemon. This will cause argus to do the things that Unix daemons do and return, if there were no errors, with argus running as a detached process.

–D <level> Print debug messages to stderr. The higher the **<level>** the more information printed. Acceptable levels are 1-8.

–e <value> Specify the source identifier for this **argus**. Acceptable values are numbers, strings, hostnames or ip address. Double quotes around the parameter are used to designate the string format. The longest supported string is 4 characeters long. Be sure and single quote or 'escape' the double quotes so that the shell doesn't gooble up the delimiters.

This option sets a global Source identifier that can be overriden by specific -i options.

```
argus -e '"arg1"'
argus -e \"arg2\"
```

–f When reading packets from a packet capture file, the -f option causes argus to not stop when end of file is reached, but rather to wait for additional packets to be appended to the input. The -f option is ignored if the standard input is a pipe, but not if it is a FIFO.

−F Use *conffile* as a source of configuration information. Options set in this file override any other specification, and so this is the last word on option values.

−g <group> Specify a group name to change to after privilege access is complete.

−h Print an explanation of all the arguments.

−i <interface> Specify the physical network **<interface>** to be audited. The default is the first network interface that is up and running.

The syntax for specifying the interface is the same format used in the argus.conf.5 file. The optional source id specification can be an IPv4 address, an integer, or a string denoted using double quotes.

```
 -i interface[/srcid]
 -i all[/srcid]
 -i dup:en0,en1/"ap01"          ( en0 and en1 are in ingress and egress interfaces )
 -i bond:en0,en1/2.3.4.5        ( en0 and en1 are bonded interfaces )
 -i en0 en1                     ( equivalent '-i bond:en0,en1' )
 -i dup:[bond:en0,en1],en2/3    ( in this case 3 is the srcid )
 -i en0/"en0" -i en1/"en1"      ( equivalent '-i ind:en0/srcid,en1/srcid' )
```

−J Generate packet peformance data in each audit record.

−M <secs> Specify the interval in <secs> of argus status records. These records are used to report the internal status of argus itself. The default is 300 seconds.

−m Provide MAC addresses information in **argus** records.

−N <packet count>|<packet range>
Specify the number of packets to process. You can give an absolute number, or a range with the syntax "start-stop". Examples are:

```
  -N 27        - read the first 27 packets.
  -N 1034-1434 - read 100 packets starting with 1034.
```

−O Turn off Berkeley Packet Filter optimizer. No reason to do this unless you think the optimizer generates bad code.

−p Do not set the physical network interface in promiscuous mode. If the interface is already in promiscuous mode, this option may have no effect. Do this to audit only the traffic coming to and from the system argus is running on.

−P <portnum> Specifies the **<portnum>** for remote client connection. The default is to not support remote access. Setting the value to zero (0) will forceably turn off the facility.

−r <[type:]file [type:]file ... >
Read from **tcpdump(1) , snoop(1)** or **NLANR's Moat Time Sequence Header** (tsh) packet capture files. If the packet capture file is a **tsh** format file, then the **-t** option must also be used. The file "-" specifies stdin as the source of packets.

The **type** provides the opportunity to specify what type of packet source to expect and process. Supported types are '' (default) and 'cisco', where argus will process the payload of packets as netflow records, when found.

Argus will read from only one input packet file at a time, and will open the files in lexigraphic order. Care should be taken to ensure that the timestamps in the packets are ordered, or unexpected behavior may result. If the **−r** option is specified, **argus** will not put down a **listen(2)** to support remote access.

−R Generate argus records such that response times can be derived from transaction data.

−s <bytes> Specify the packet snaplen.

−S <secs> Specify the status reporting interval in <secs> for all traffic flows.

−t Indicate that the expected packet capture input file is a **NLANR's Moat Time Sequence Header** (tsh) packet capture file.

−T timescale
Specify a playback timescale for realtime processing of input packets.

−u <user> Specify an account name to change to after privilege access is complete.

−U Specify the number of user bytes to capture.

−w <file | stream ["filter"]> Append transaction status records to *output-file* or write records to the URL based stream. Supported stream URLs are 'argus-udp://host[:port]', where the default port is 561. An *output-file* of '-' directs **argus** to write the resulting *argus-file* output to *stdout*.

−X Clear existing argus configuration. This removes any initialization done prior to encountering this flag. Allows you to eliminate the effects of the */etc/argus.conf* file, or any argus.conf files that may have been loaded.

−Z Collect packet size information. This options turns on packet size reporting for all flows. Argus will provide the mean, max, min and standard deviation of the packet sizes seen during the flow status interval.

expression
This **tcpdump(1)** expression specifies which transactions will be selected. If no *expression* is given, all transactions are selected. Otherwise, only transactions for which *expression* is 'true' will be dumped. For a complete *expression* format description, please refer to the **tcpdump(1)** man page.

SIGNALS

Argus catches a number of **signal(3)** events. The three signals **SIGHUP**, **SIGINT**, and **SIGTERM** cause **argus** to exit, writing TIMEDOUT status records for all currently active transactions. The signal **SIGUSR1** will turn on **debug** reporting, and subsequent **SIGUSR1** signals, will increment the **debug-level**. The signal **SIGUSR2** will cause **argus** to turn off all **debug** reporting.

FILES

```
/etc/argus.conf        - argus daemon configuration file
/var/run/argus.#.#.pid  - PID file
```

EXAMPLES

Run **argus** as a daemon, writing all its transaction status reports to *output-file*. This is the typical mode.

```
argus -d -e `hostname` -w output-file
```

If ICMP traffic is not of interest to you, you can filter out ICMP packets on input.

```
argus -w output-file - ip and not icmp
```

Argus supports both input filtering and output filtering, and argus supports multiple output streams, each with their own independant filters. Output streams can be written to udp based sockets, to unicast or multicast addresses.

If you are interested in tracking IP traffic only (input filter) and want to report ICMP traffic to one output stream, and all other IP traffic in another output stream.

```
argus -w argus-udp://224.0.20.21:561 "icmp" \
    -w argus-udp://224.0.20.21:562 "not icmp" - ip
```

Audit the network activity that is flowing between the two gateway routers, whose ethernet addresses are 00:08:03:2D:42:01 and 00:00:0C:18:29:F1. Without specifying an *output-file*, it is assumed that the transaction status reports will be written to a remote client. In this case we have changed the port that the remote client will use to port 430/tcp.

```
argus -P 430 ether host (0:8:3:2d:42:1 and 0:0:c:18:29:f1) &
```

Audit each individual ICMP ECHO transaction from data in <dir>. You would do this to gather Round Trip Time (RTT) data within your network. Append the output to *output-file*.

argus -R dir -w *output-file* **"echo" - icmp**

Audit all NFS transactions involving the server *fileserver* and increase the reporting interval to 3600 seconds (to provide high data reduction). Append the output to *output-file*.

argus -S 3600 -w *output-file* **- host fileserver and udp and port 2049** *&*

Import flow data from pcap file containing Cisco flow data packets. Write output to stdout, to a *ra.1* instance.

argus -r *cisco:pcap-file* **-w - | ra**

AUTHORS

Carter Bullard (carter@qosient.com)

SEE ALSO

hosts_access(5), **hosts_options**(5), **tcpd**(8), **tcpdump**(1)