Chapter 27

Intrusion Prevention System (IPS)

This chapter describes the TCPware Intrusion Prevention System (IPS). This security feature monitors network and/or system activities for malicious or unwanted behavior and can react, in real-time, to block or prevent those activities. IPS is highly flexible and customizable. When an attack is detected, pre-configured rules will block an intruder's IP address from accessing the TCPware system, prevent an intruder from accessing a specific application, or both. The time period that the filter is in place is configurable. An API is provided so that TCPware customers can incorporate the IPS functionality into user-written applications.

IPS is implemented by instrumenting components (e.g, TCPware SSH or FTP, or user-supplied components) with a Process Software supplied API that allows them to report events, such as invalid login attempts, to the FILTER_SERVER process. The filter server, started when TCPware starts, maintains the component rulesets and lists of events, based on the originating address for the offending connection, and when defined limits are reached, creates and sets timed filters in the TCPware kernel to filter that traffic.

IPS Operation

All of the operating parameters such as the definition of the rules, the number of events/unit time to trigger a filter, the duration of a filter, etc. are all defined by component configuration files.

Events are recorded per source address, per rule, per destination address, per component. This provides the ability to have differing filtering criteria for different interfaces (for example, an internal network vs. an external network). Addresses or networks may be excluded from consideration when an event is logged. This feature allows, for example, different settings to be used for internal vs. external networks.

Events “age”; after a time period, old events are discarded from the list of events so that infrequent event occurrences (e.g., mistyping a password) have less chance of inadvertently causing a filter to be set. Note that when a filter is triggered for an address and rule, the list of known events for that rule and address are deleted.

Multiple filters may be set in sequence for a component/rule/source address/destination address as events are logged. The purpose of this is to make a filter progressively longer. For example, the first filter set for an address and rule might be 5 minutes long; the next, 10 minutes long; the next, 15 minutes long; etc., up to 5 filter times.

Configuring IPS

IPS is configured in two steps:

1. Enabling IPS and configuring the main process-specific parameters of the FILTER_SERVER process (for example, the size of the mailbox used by applications to communicate with the FILTER_SERVER process).
2. Editing the FILTER_SERVER configuration files to set the operating parameters of IPS; for example, the applications that will use IPS and setting the rule parameters for reporting events.

Configuring Process-Specific Parameters

The TCPware CNFNET procedure is used to enable and disable IPS, as well as to configure the process-specific parameters for the FILTER_SERVER process. The process-specific parameters that are set/modified are the size of the FILTER_SERVER mailbox and some of the process-specific quotas for the FILTER_SERVER process. These parameters set the following logical names:

TCPWARE_FILTER_SERVER_TQELM

TCPWARE_FILTER_SERVER_ASTLM

TCPWARE_FILTER_SERVER_MBX_MSGS

Determining the Correct FILTER_SERVER Process Quotas

It is important to correctly determine the correct process quotas for the FILTER_SERVER process. High-volume systems, for example, an E-mail server where SMTP may detect many anomalies, may log large numbers of events in a short time. If the TQELM and ASTLM quotas for FILTER_SERVER are too low, the FILTER_SERVER process could enter MUTEX state and hang, preventing any events from being logged and possibly leading to other problems such as processes hanging while trying to log events.

The amount of additional TQELM quota in addition to the default value (specified via the PQL_DTQELM SYSGEN parameter) required for the FILTER_SERVER process can be calculated as follows:

- 1 for automated hourly reporting
- 1 for automated 24-hour maintenance
- 1 for each source address per rule per component for which an event has been received. These timers are used to clean up internal address structures and disappear after 24 hours of inactivity from that address.
- 1 for each non-empty event queue per source address per rule per component. These timers ar used to delete aged events from the event queue.

Thus, the event frequency must be anticipated and the quotas adjusted appropriately for each installation. The hourly FILTER_SERVER logs will be of use for determining traffic patterns.

The ASTLMquota tends to track the value for TQELM closely, but should have an additional 10% added to it for other uses.

Both the ASTLM and TQELM quotas are controlled by logical names described int he previous section. Both of the ASTLM and TQELM values default to 500.

Determining the Correct FILTER_SEVER Mailbox Size

In addition to setting the TQELM and ASTLM process quotas correctly, the size of the mailbox used for communication with the FILTER_SERVER process must be correctly determined. Failure to do can result in events reported by instrumented components being lost. The mailbox is sized to handle 400 simultaneous event messages by default.

Once the mailbox size has been configured, either the system must be rebooted to allow the new mailbox size to be used (this is the preferred method), or the following procedure can be used to avoid a reboot in the near term:

3. Stop IPS (NETCU SET IPS /STOP).
4. Stop all applications using IPS (e.g., telnet sessions, ftp session, etc.).
5. Delete the old mailbox by running TCPWARE:DELMBX.EXE.
6. Start IPS (NETCU SET IPS/START).
7. Start any other applications previously stopped.

Filter Server Main Configuration

The filter server is configured using a main configuration file and per-component configuration files. The main configuration file is used to set overall configuration options for filter server operation, while the per-component configuration files contain configuration information for each instrumented component such as the ruleset to use, the prototype filter to be set, etc. Per-component configuration files are referenced by the main configuration file by using the “INCLUDE” keyword.

Sample configuration files are supplied in the TCPware distribution and must be copied and modified as necessary to conform to the particular site’s security profile and interface configuration. These files are copied to the TCPWARE: directory when TCPware is installed. Once these have been copied and modified, the filter server configuration may be reloaded via the NETCU SET IPS/RELOAD command. The template files supplied are:

FILTER_SERVER_CONFIG.TEMPLATE

SSH_FILTER_CONFIG.TEMPLATE

IMAP_FILTER_CONFIG.TEMPLATE

POP3_FILTER_CONFIG.TEMPLATE

SNMP_FILTER_CONFIG.TEMPLATE

SMTP_FILTER_CONFIG.TEMPLATE

TELNET_FILTER_CONFIG.TEMPLATE

REXEC_FILTER_CONFIG.TEMPLATE

RLOGIN_FILTER_CONFIG.TEMPLATE

RSHELL_FILTER_CONFIG.TEMPLATE

The following table lists the main configuration file keywords. These are found in the file TCPWARE:FILTER_SERVER_CONFIG.TXT:

Table 7-1 Main configuration Keywords

Keyword	**Default**	**Description**
BLOCK_AT_DESTINATION_PORT	NO	If set to YES, indicates that all filters generated by the filter server will be for a specific destination port (the equivalent of a filter line of “EQ <portnumber>”). The port number to be used is specified for each component in the per-component configuration file. If set NO, all filters generated by the filter server will deny access to all destination ports.
DEBUG *value*	0	Indicates the amount of debug to output. Zero means no debug, while higher number mean more debug. This value should ordinarily never be set above 4 without direction from Process Software.
ENTERPRISE_STRING		Defines the location in the MIB tree that the trap used to send filter logging events via SNMP pertains to.
GENERIC_TRAP_ID		An integer representing the generic trap value when filter logging events are sent via SNMP.
INCLUDE *filename*		Specifies a per-component configuration file to load. Any number of INCLUDE statements may occur in the main configuration file.

Table 7-1 Main configuration Keywords

Keyword	Default	Description
LOG_TO_LOGFILE	NO	If YES, information log messages are sent to the log file specified by the *logfile* keyword.
LOG_TO_OPCOM	NO	If YES, informational messages are reported via OPCOM.
LOG_TO_SNMP	NO	If YES, informational messages are reported via an SNMP trap.
LOGFILE		Specifies the log file used when the *log_to_logfile* keyword is specified.
OPCOM_TARGET	NETWORK, SECURITY	Specifies the list of operator types to which events are written when the keyword LOG_TO_OPCOM is set. This is a comma-separated list, and may contain any of the values which are valid for the VMS REPLY/ENABLE command.
SPECIFIC_TRAP_ID		An integer representing the specific trap value when filter logging events are sent via SNMP.

Filter Server Per-Component Configuration File

The per-component configuration files are loaded using the INCLUDE keyword in the main filter server configuration file. Each of these configuration files have the following format. The definition must begin with a COMPONENT keyword. Comments lines begin with a “#” character.

```
COMPONENT component-name
     DESTINATION_ADDRESS
     EXCLUDE_ADDRESS
     DESTINATION_PORT
     PROTO_FILTER
RULE rulename
     DESTINATION_ADDRESS
     DESTINATION_PORT
     MAX_COUNT
     DELTA_TIME
     FILTER_DURATIONS
RULE rulename
```

```
MAX_COUNT
DELTA_TIME
FILTER_DURATIONS
```

Each component may have as many rules defined for it as are appropriate for the component. However, the more rules defined for a component, the more complex it may be to instrument the component to actually report those rules. All entries in configuration files are not case-sensitive.

The following table shows the keywords for a per-component configuration file:

Table 7-2 Per-Component Configuration Keywords

Keyword	Scope	Description
COMPONENT *component-name*	component	Name of the component to which this applies (e.g., SSH).
DELTA_TIME *time*	rule	Time, in seconds, where if *max_count* events are received for a rule from the same address, that will cause a filter to be set for that address. This is also the time for aging events. If the age of an event exceeds *delta_time* seconds, the event is dropped from the event list.
DESTINATION_ADDRESS *address*	component or rule	Destination IP address (the TCPware interface address) in CIDR format to check. If *destination_address* occurs before the first rule in the per-component configuration file, it will be used as a default for any rule for the component that doesn't have a destination address specified. Note: multiple *destination_address* lines may be specified at the component level if all the interfaces specified have the same filtering criteria.

Table 7-2 Per-Component Configuration Keywords

Keyword	Scope	Description
DESTINATION_PORT *port*	component or rule	Optional destination port. This will only be effective if the keyword *BLOCK_AT_DESTINATION_PORT* is set in the main configuration file.log
EXCLUDE_ADDRESS *address*	component or rule	A source address/mask in CIDR format from which events are ignored. This allows, for example, events from an internal network to be ignored while counting events from external networks. Multiple *EXCLUDE_ADDRESS* lines may be specified for each rule.
FILTER_DURATIONS *list*	rule	List of durations for filters. This is a comma-delimited list of up to five filter durations, and it must be terminated with a -1.
MAX_COUNT *count*	rule	Maximum number of events from the same address for a specific rule over *delta_time* seconds that will trigger a filter.
PROTO_FILTER	component	This is a prototype filter to be used to build the filter set against an interface when a filter is triggered. The format of this filter is the same as used in a filter file, with one exception as noted below. **NOTE**: The source address & subnet mask and destination address & subnet mask for the prototype filter must be specified in CIDR format. This is to maintain consistency with the other address/subnet mask combinations in, for example, the *EXCLUDE_ADDRESS* keywords.
RULE *rulename*	component	The user-defined name for a rule.

Sample Main Configuration File

```
#==========================================================================
#
#        FILTER_SERVER_CONFIG.TEMPLATE
#
#==========================================================================
#
#        The following parameter determines the level of debug information
#        written to the debug log file.  This should normally be set to a
#        value of 2 or less, and shouldn't be set above 4 without the
#        recommendation of Process Software, as higher debug levels will
#        negatively impact the filter server (and possibly, system)
#        performance.  The debug messages will be found in the file
#        TCPware:FILTER_SERVER.OUT.
#
debug 4
#
#        The following parameters define the logging locations.  Note
#        that debug messages are not written to the logging locations.
#
#        The first two parameters are used when logging to a log file.
#
logfile tcpware:filter_logfile.log
log_to_logfile    yes
#
#        The next parameter is used when logging to OPCOM.
#
log_to_opcom    yes
opcom_target    NETWORK,SECURITY,OPER3

#
#        The next parameters are used when logging via SNMP.  Details
#        on the values for enterprise_string, generic_trap_id and
#        specific_trap_id can be found in chapter 23 of the TCPware
#        Administrators Guide.
#
log_to_snmp   no
# enterprise_string
# generic_trap_id
# specific_trap_id
#
#        The following parameter determines how filters are created.  If
#        set to YES, then the destination port field is added to the filter
#        (e.g., "192.168.0.11 eq 22").  If set to NO, then no source
#        port field is added, which will cause the filter to block all
#        traffic of the specified protocol from the source address.  If
#        not set, default is NO.
#
# block_at_destination_port yes
#
#==========================================================================
#
#        The following lines defihe the individual configuration files
#        for each configured component that uses the filter server
#
#==========================================================================
#
include tcpware:ftp_filter_config.txt
include tcpware:imap_filter_config.txt
include tcpware:pop3_filter_config.txt
include tcpware:smtp_filter_config.txt
include tcpware:snmp_filter_config.txt
include tcpware:ssh_filter_config.txt
#
```

For this configuration:

- Debug will be reported at level 4 (this produces fairly detailed information, normally useful only by Process Software when debugging a problem).
- Log messages will be logged to TCPWARE:FILTER_LOGFILE.LOG and OPCOM.
- When filters are logged, the destination port specified in the per-configuration files will be used.
- Per-component configuration files for the TCPware FTP, IMAP, POP3, SMTP, SNMP and SSH servers will be loaded.

Sample Component Configuration File

The following is a configuration file for the SSH component:

```
component ssh
    proto_filter "deny tcp 192.168.0.100/32 192.168.0.11/24 log"
    destination_address 192.168.0.16/32
    exclude_address 192.168.0/24
    destination_port 22
    rule   ssh_bogus_id
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
    rule   ssh_authfailed
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
    rule   ssh_authfailed
           destination_address  192.168.10.2/32
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
    rule   ssh_userauth
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
    rule   ssh_invaliduser
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
```

For component SSH, a “deny tcp” filter will be used. The source address/mask and destination address/mask parts of the prototype filter are ignored and are overwritten by the actual data specified by the source information gathered from the event that triggered the filter, and by the destination address/mask/port information specified by the corresponding keywords in this file. Events from the 192.168.0 network are all excluded from being counted.

To examine the first three rules specified above:

The rule is “ssh_bogus_id”.   Since no address or mask is specified for this rule, it will use the default destination address of 192.168.0.16 and mask of 255.255.255.255 specified at the beginning of the component configuration. The rule states that if 10 events from the same source address are seen over 90 seconds, a filter is created using the *proto_filter* specified above. The first filter is 5 minutes long, the second, 10 minutes, and so on, until at the 5th time, a permanent filter is put in place for the address and interface that is causing the problem.

The second rule is "ssh_authfailed", and applies to events received as a result of connections on the interface with the default address of 192.168.0.16 and mask of 255.255.255.0, respectively.

The third rule is also "ssh_authfailed", but applies to events received as a result of connections on the interface with the address 192.168.10.2, using a mask of 255.255.0.0. The *max_count* and *delta_time* parameters are different for this interface than for the previous *ssh_authfailed* rule in the system.

The remaining rules for this component will use the default address 192.168.0.16 and mask of 255.255.255.0.

Note! If a rule specifies a destination address for an interface which does not currently exist, events for that interface will be dropped until the interface becomes available.

The ***component*** keyword may occur exactly once in the configuration file. The following example shows a configuration file for component ftp for 5 interfaces (EIA-0, EIA-1, PSD-0, PSD-1, PSD-2):

```
=======================================================================
#
#       FTP_FILTER_CONFIG.TXT
#
#       Filter server configuration file for the FTP component.
#
#======================================================================
component ftp
         proto_filter "deny tcp 192.168.0.100/32 192.168.0.1/24 log"
        destination_port 21

#
#  For EIA-0 and EIA-1:
#
        destination_address 192.168.0.29/32
        destination_address 192.168.0.25/32
        rule    ftp_invaliduser
                max_count       10
                delta_time      300
                filter_durations  300,600,1800,3600,-1
        rule    ftp_userauth
                max_count       21
                delta_time      180
                filter_durations  300,600,1800,3600,-1
        rule    ftp_authfailed
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
        rule    ftp_timeout
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
#
#  For PSD-0:
#
        rule    ftp_invaliduser
                max_count       10
                delta_time      300
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.28/32
                destination_port  1521
        rule    ftp_userauth
                max_count       21
                delta_time      180
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.28/32
                destination_port  1521
        rule    ftp_authfailed
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.28/32
                destination_port  1521
        rule    ftp_timeout
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.28/32
                destination_port  1521
#
#  For PSD-1:
#
        rule    ftp_invaliduser
                max_count       10
                delta_time      300
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.27/32
                exclude_address  192.168.0.0/24
        rule    ftp_userauth
                max_count       21
                delta_time      180
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.27/32
                exclude_address  192.168.0.0/24
        rule    ftp_authfailed
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.27/32
                exclude_address  192.168.0.0/24
        rule    ftp_timeout
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.27/32
                exclude_address  192.168.0.0/24
#
#  For PSD-2:
#
        rule    ftp_invaliduser
                max_count       10
                delta_time      300
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.21/32
        rule    ftp_userauth
                max_count       21
                delta_time      180
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.21/32
        rule    ftp_authfailed
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.21/32
        rule    ftp_timeout
                max_count       21
                delta_time      90
                filter_durations  300,600,1800,3600,-1
                destination_address 192.168.0.21/32
```

The above example shows some configuration options for the system with 4 interfaces. Specifically:

- Interfaces IEA-0 and EIA-1 will use identical rules, because they didn’t specifiy destination addresses within their rulesets and the destination addresses for EIA-0 and EIA-1 were specified at the component level. All other interface rules specified their own destination addresses at the rule level, so they will use specific rules for those specific addresses.
- A default port of 21 was been specified for all interfaces. However, interface PSD-0 has specified a port of 1521, so that port will be used for PSD-0 only. All other interfaces will use the default port of 21.
- Interface PSD-1 has an ***exclude_address*** specified for net 192.168.0.0/24. All events for PSD-1 that originated from that source net will be excluded from being counted by IPS. All other interfaces will count events from that network.

Note! Filters for TCPware pseudo device (PSD-n) interfaces will be set on the actual physical interface to which the pseudo device maps. However, the ***destination_address*** field in the rule definition must specify the address for the pseudo device.

Configuring IPS for Paired Network Interfaces (Pair Gain)

To configure IPS for a Pair Gain environment where multiple interfaces are treated as a common link set, the rules are fairly simple.

- Each physical and pseudo interface must be specified in the configuration files via *destination_address* rules for each interface.
- All physical interfaces are treated equally. When an event is logged for any interface in the set, it's as if it was logged against each interface in the set. Thus, when a filter is set on any interface in the set, the same filter is set on all physical interfaces in the set.
- Filters are set only on the physical interfaces. Since pseudo devices (PSD-*nnn*) are not true interfaces, they cannot have filters set on them.
- When a filter is created as a result of events coming in via a pseudo device, the destination address shown in the filter (using the NETCU SHOW FILTER command) will show the destination address for the pseudo device.

When NETCU is used to perform any of the following tasks:

- Create a Pair Gain set via SET INTERFACE /COMMON_LINK
- Stop an interface via STOP/IP
- Start an interface via START/IP

IPS is notified of the change being made. This allows the FILTER_SERVER process to reevaluate all interfaces it knows about, so it can determine if modifications must be made to Pair Gain sets about which it currently knows.

The following example shows a configuration for the SSH component for a Pair Gain configuration that consists of EWA-0, EWA-1, and PSD-0 where PSD-0's physical interface is EWA-1:

```
component ssh
    proto_filter "deny tcp 192.168.0.100/32 192.168.0.11/24 log"
    #
    # EWA-0's address
    #
    destination_address 192.168.0.70/32
    #
    # EWA-1's address
    #
    destination_address 192.168.0.71/32
    #
    # PSD-0's address
    #
    destination_address 192.168.0.72/32
    #
    destination_port 22
    rule   ssh_bogus_id
           max_count        10
           delta_time       90
           filter_durations  300,600,1800,3600,-1
    rule   ssh_authfailed
```

```
                max_count        10
                delta_time       90
                filter_durations  300,600,1800,3600,-1
         rule   ssh_authfailed
                destination_address  192.168.10.2/32
                max_count        10
                delta_time       90
                filter_durations  300,600,1800,3600,-1
         rule   ssh_userauth
                max_count        10
                delta_time       90
                filter_durations  300,600,1800,3600,-1
         rule   ssh_invaliduser
                max_count        10
                delta_time       90
                filter_durations  300,600,1800,3600,-1
```

Using the above configuration, the next item illustrates a filter being set due to events that occurred on line PSD-0:

```
RAPTOR_$
%%%%%%%%%%  OPCOM  29-OCT-2009 13:00:55.77  %%%%%%%%%%%    (from node VOODOO at 29-OCT-2009
13:00:59.12)
Message from user JOHNDOE on VOODOO
FILTER_SERVER: Filter queued on EWA-0 (192.168.0.70/32) at 29-OCT-2009 13:00:59.12
               Component: ssh, Rule: ssh_bogus_id
deny      tcp    192.168.0.11/32
                 192.168.0.72/32      eq 22
                 FLTSVR,LOG
                 START: 29-OCT-2009 13:00:59.12  END: 29-OCT-2009 14:00:59.12


RAPTOR_$
%%%%%%%%%%%  OPCOM  29-OCT-2009 13:00:55.80  %%%%%%%%%%%    (from node VOODOO at 29-OCT-2009
13:00:59.15)
Message from user JOHNDOE on VOODOO
FILTER_SERVER: Filter queued on EWA-1 (192.168.0.71/32) at 29-OCT-2009 13:00:59.15
               Component: ssh, Rule: ssh_bogus_id
deny      tcp    192.168.0.11/32
                 192.168.0.72/32      eq 22
                 FLTSVR,LOG
                 START: 29-OCT-2009 13:00:59.15  END: 29-OCT-2009 14:00:59.15


RAPTOR_$
```

Note some things illustrated above:

- Each physical address (EWA-0 and EWA-1) had a filter set on it.
- No filter was set on interface PSD-0 because it is a pseudo interface.
- The destination address for each event is that of interface PSD-0, since that was the source of the events that caused the filters to be set.

Filter Reporting via OPCOM and Log File

When a filter is set for an address/rule/destination/component, an informational message will appear either in OPCOM (if ***LOG_TO_OPCOM*** is set) or in the logfile (if ***LOG_TO_LOGFILE*** is set). The following message illustrates an OPCOM message, but the message to a logfile will have the same format.

```
TWEET_$
%%%%%%%%%%%  OPCOM  16-MAY-2008 10:33:19.74  %%%%%%%%%%%
Message from user SYSTEM on TWEET
FILTER_SERVER: Filter queued on EIA-0 (192.168.0.16) at 16-MAY-2009
10:33:19.74
               Component: ssh, Rule: ssh_bogus_id
deny      tcp    192.168.0.16/32
                 192.168.0.0/24      eq 22
                 FLTSVR,LOG
                 START: 16-MAY-2009 10:33:19  END: 16-MAY-2009 10:38:19
TWEET_$
```

This message is in essentially the same format as that when a NETCU SHOW FILTER command is performed:

```
$ netcu show filter eia-0
TCPware(R) for OpenVMS Packet Filter List for EIA-0:

Logging to OPCOM at 5 second intervals (normal format)

             Source           Source   Destination     Destination
Action Proto Address/Mask     Port     Address/Mask    Port     Option       Hits
------ ----- --------------   ----     --------------- ----     ------       ----
deny     tcp 192.168.0.16              192.168.0.22
             192.168.0.255             255.255.255.255          FLTSVR,LOG     0

permit  ip   0.0.0.0                   0.0.0.0
             0.0.0.0                   0.0.0.0                  FLTSVR         0
```

$

Note the second filter (the “permit ip” filter) that is shown. If there are currently no filters set for an interface when the filter server determines it needs to set a filter, it will add an explicit “permit ip” filter. This is done because the existence of any filter in a list of filters causes TCPware to act as if a “deny everything” filter terminates the list. The “permit ip” filter will essentially prevent that problem from happening.

Filter Reporting via SNMP

When logging a filter via SNMP, the configuration keywords *ENTERPRISE_STRING, GENERIC_TRAP_ID,* and *SPECIFIC_TRAP_ID* must be specified (as well as the keyword *LOG_TO_SNMP*). In addition, the SNMP configuration file must be properly set up on the TCPware system as specified in chapter 7 of the *TCPware Management Guide*.

When a filter is logged, the following fields will be reported:

FILTER_SERVER: Filter queued on <*interface*> (<*address*>) at <*time*>

COMPONENT=*component-name*

RULE=*rulename*

ACTION=*actionname* (e.g., “deny”)

PROTOCOL=*protocol* (e.g., “TCP”)

SOURCE=*source address in CIDR format*

SOURCE_PORT= *operator port* (e.g., “EQ 22”)

DESTINATION=*destination address in CIDR format*

DEST_PORT=operator *port* (e.g., “EQ 22”)

START=*VMS absolute time*

END=*VMS absolute time*

Correcting a Filter List

If a filter is inadvertently created by the filter server, the system manager should first correct the configuration problem (if one exists) that allowed the filter to be incorrectly set. Then, the system manager may retrieve the current list of filters in “manual filter form” that can be edited then reloaded onto the interface. The list is retrieved via the NETCU SHOW FILTER <*interface*>/ EXTRACT_FILTER command. For example:

This message is in essentially the same format as that when a NETCU SHOW FILTER command is performed:

```
$ netcu show filter eia-0
TCPware(R) for OpenVMS Packet Filter List for EIA-0:

Logging to OPCOM at 5 second intervals (normal format)

             Source          Source   Destination     Destination
Action Proto Address/Mask    Port     Address/Mask    Port     Option       Hits
------ ----- --------------  ----     --------------- ----     ------       ----
deny     tcp 192.168.0.16             192.168.0.22
             192.168.0.255            255.255.255.255          FLTSVR,LOG     0

permit  ip   0.0.0.0                  0.0.0.0
             0.0.0.0                  0.0.0.0                  FLTSVR         0
```

$

Note the second filter (the “permit ip” filter) that is shown. If there are currently no filters set for an interface when the filter server determines it needs to set a filter, it will add an explicit “permit ip” filter. This is done because the existence of any filter in a list of filters causes TCPware to act as if a “deny everything” filter terminates the list. The “permit ip” filter will essentially prevent that problem from happening.

```
$ netcu show filter eia-0/extract_filter=filter.txt
$ netcu show filter eia-0
TCPware(R) for OpenVMS Packet Filter List for EIA-0:

Logging to OPCOM at 5 second intervals (normal format)

             Source          Source   Destination     Destination
Action Proto Address/Mask    Port     Address/Mask    Port     Option       Hits
------ ----- --------------  ----     --------------- ----     ------       ----
```

```
deny     tcp 192.168.0.14            192.168.0.22      22
             255.255.255.255         255.255.255.0           FLTSVR,LOG     0

deny     tcp 192.168.0.45            192.168.0.22
             255.255.255.255         255.255.255.0                          0
permit   ip  0.0.0.0                 0.0.0.0
             0.0.0.0                 0.0.0.0                 FLTSVR         0


$ type filter.txt
#
#   FILTER.TXT
#
#   Generated 16-MAY-2009 10:51:31
#
#=====================================================================
deny tcp 192.168.0.14 255.255.255.255 192.168.0.22 225.255.255.0 eq 22 start “16-MAY-2009
10:33:19” end “16-MAY-2009 10:38:19”LOG
deny tcp 192.168.0.45 255.255.255.255.192.168.0.22 255.255.255.0
permit ip 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
$ <edit to remove the first (filter) line>
$ netcu set filter eia-0 filter.dat


$ netcu show filter eia-0
TCPware(R) for OpenVMS Packet Filter List for EIA-0:

Logging to OPCOM at 5 second intervals (normal format)

             Source          Source   Destination     Destination
Action Proto Address/Mask    Port     Address/Mask    Port     Option       Hits
------ ----- -------------   ----     --------------- ----     ------       ----
deny     tcp 192.168.0.45             192.168.0.22
             255.255.255.255          255.255.255.0                           0
permit   ip  0.0.0.0                  0.0.0.0
             0.0.0.0                  0.0.0.0                FLTSVR           0

$
```

Configuring PMDF to use IPS on TCPware

The IMAP, POP3 and SMTP servers referred to in the configuration template files above refer to the TCPware servers only. Beginning with PMDF V6.4, PMDF has been instrumented to use IPS. The following PMDF template files are available in the PMDF_TABLE directory:

FILTER_SERVER_PMDF_IMAP.TEMPLATE

FILTER_SERVER_PMDF_POP3.TEMPLATE

FILTER_SERVER_PMDF_SMTP.TEMPLATE

These files should be copied to TCPWARE:*.TXT and modified as appropriate for your installation. Edit TCPWARE:FILTER_SERVER_CONFIG.TXT to add INCLUDE lines for these component files, and comment out the INCLUDE lines for the standard TCPware IMAP, POP and SMTP files.

Next, make sure the appropriate PMDF images are installed. The legacy IMAP and POP servers (PMDF_EXE:IMAPD.EXE and PMDF_EXE:POP3D.EXE) are already installed. The msgstore IMAP and POP servers (PMDF_EXE:IMAP_SERVER.EXE, PMDF_EXE:POP_SERVER.EXE), as well as the SMTP server (PMDF_EXE:TCP_SMTP_SERVER.EXE) are not installed, so they will need to be added to your PMDF_COM:PMDF_SITE_STARTUP.COM file if your PMDF installation uses them. These must all be installed using the /OPEN qualifier.

At this point, define the logical name PMDF_DO_FILTER_SERVER to 1, using the /SYSTEM qualifier (this can be put in PMDF_COM:PMDF_SITE_STARTUP.COM as well).

Finally, restart IPS via the TCPware NETCU SET IPS/RESTART command.

Note! Note: the PMDF_TABLE:FILTER_SERVER_PMDF_POP3.TEMPLATE file as distributed in PMDF V6.4 has the incorrect port number of 143 specified. Make sure port 110 is specified in this file.

Controlling the Filter Server

The filter server is started at system startup time. However, it can be controlled using the $ NETCU SET IPS command. The valid commands and their uses are:

Table 7-3 Filter Server NETCU SET Commands

Command	**Description**
/DEBUG_LEVEL=*level*	Change the debug level for the server. See the description for the *debug* main configuration keyword.
/RELOAD	Re-read and parse the configuration files. Note that this will not wipe out existing event and rule information; it will simply update it so no potential filter information will be lost.
/RESTART	Stop and restart the filter server. All existing event and rule information will be lost and reloaded from the configuration files.
/START	Start the filter server if it’s not already running.
/STOP	Stop the filter server from running. All existing event and rule information will be lost.

Table 7-4 Filter Server NETCU SHOW Commands

Command	**Description**
/CONFIG *{ = filename }*	Defaults to CONFIG.TXT. Prints out the current configuration and status. This normally all the information a system manager needs.

Table 7-4 Filter Server NETCU SHOW Commands

Command	Description
/EXTENDED_CONFIG *{ = filename }*	Defaults to EXTENDED_CONFIG.TXT. Used primarily for debugging purposes by Process Software, this prints out a detailed configuration and status report. Much of this additional data will be meaningless to most users.

The current configuration of the filter server may also be displayed using the $ NETCU SHOW IPS /CONFIG=<*filename*> command. For example:

```
$ netcu show ips/config=server_stats.out
$ type server_stats.out
Filter server snapshot

Debug level 6
Block at destination port or system: PORT
Log to: OPCOM   SNMP trap
Component  ssh
    Rule ssh_bogus_id
        dest address:       192.168.0.16/32
        interface:          se0
        max event count:    10
        delta time:          0 00:01:30.00
        filter durations:   300  600  1800  3600  -1
        Address 192.168.0.11/32
            number of still-queued events:   1
            number of filters created:       0
            Address entry to be deleted:     N/A
            Event
                event time: 29-APR-2008 10:00:12.41
                expires:    29-APR-2008 10:01:42.41
    Rule ssh_authfailed
        dest address:     192.168.0.16/32
        interface:        se0
        max event count:  10
        delta time:          0 00:01:30.00
        filter durations:   300  600  1800  3600  -1
    Rule ssh_userauth
        dest address:     192.168.0.16
        interface:        se0
        max event count:  10
        delta time:          0 00:01:30.00
        filter durations:   300  600  1800  3600  -1
    Rule ssh_invaliduser
        dest address:     192.168.0.16/32
        interface:        se0
        max event count: 10
        delta time:         0 00:01:30.00
        filter durations:  300  600  1800  3600  -1
    Rule ssh_invalid_id_msg
        dest address:     192.168.0.16/32
        interface:        se0
        max event count: 5
        delta time:         0 00:02:00.00
        filter durations:  300  600  1800  3600  -1
```

Filter Server Files

The following files are associated with the filter server:

TCPWARE:FILTER_SERVER_HOURLY_LOG.*yyyymmdd*

This file is an hourly activity log for the filter server. The file extension changes daily at midnight to reflect the current day. What follows is a sample log segment for one hour:

```
Filter server hourly snapshot for hour 2 of 05/18/2008

Component  ssh

    Rule ssh_bogus_id
        number of hits 0

    Rule ssh_authfailed
        number of hits 0

    Rule ssh_userauth
        number of hits 0

    Rule ssh_invaliduser
        number of hits 2

        Address 192.168.0.10/32
            number of still-queued events:   0
            number of all events:            0
            number of filters created:       1
            Address entry to be deleted:     18-MAY-2008 05:55:45.35

        Address 192.168.0.204
            number of still-queued events:   0
            number of all events:            2
            number of filters created:       0
            Address entry to be deleted:     18-MAY-2008 06:22:03.97
```

This log is showing that during the hour 01:00-02:00, 2 different source addresses were being tracked by the filter server.:

The first address (192.168.0.10) had a filter created sometime in the last 4 hours (the time it takes an address to have no activity before its records are deleted by the filter server). The log indicates the address entry is to be deleted in 3 hours if there is no more activity; therefore, the filter was actually set in the previous hour (looking at the previous hour's entry in the log file will confirm this).

The second address (192.168.0.204) had 2 events during the hour that never triggered a filter and were deleted after they aged. This address entry is scheduled to be deleted in 4 hours if there's no more activity for it.

TCPWARE:FILTER_SERVER_CONFIG.TXT

This is the main filter server configuration file. Optionally, the server will use the logical name FILTER_SERVER_CONFIG to determine the name of the main configuration file.

TCPWARE:FILTER_SERVER.OUT

This file contains any output resulting from starting the filter server (e.g., the output from any DCL commands executed to start it) and all debug messages.

IPS Logicals

The following logical names are used by the FILTER_SERVER process within IPS: They are generally created and set by CNFNET, which is used to configure IPS running parameters, and each of these logical names must have the /SYSTEM attribute.

FILTER_SERVER_CONFIG

Defined by the user to be the name of the FILTER_SERVER configuration file.

TCPWARE_FILTER_SERVER_ASTLM

Defined by CNFNET, this defines the size of the ASTLM quota for the FILTER_SERVER process. This must not be set by hand; CNFNET must be used to modify the value of this logical name.

TCPWARE_FILTER_SERVER_MBX_MSGS

Defined by CNFNET, this defines the size of the FILTER_SERVER mailbox. If this value is set too low, event messages may be lost. This must not be set by hand; CNFNET must be used to modify the value of this logical name.

TCPWARE_FILTER_SERVER_QUOTA_CHECK_TIME

Defined and set by the system administrator, this defines the value, in seconds, of how often the FILTER_SERVER process checks its TQELM and ASTLM parameters. If more than 90% of the quote value for either of these parameters is exceeded, a warning message is output to OPCOM. This is useful for determining the proper settings for the TQELM and ASTLM quotas for the FILTER_SERVER process.

TCPWARE_FILTER_SERVER_TQELM

Defined by IPS_CONTROL, this defines the size of the TQELM quota for the FILTER_SERVER process. This must not be set by hand; CNFNET must be used to modify the value of this logical name.

Instrumenting a User-Written Application with IPS

When instrumenting an application (aka, a component), there are several steps to be followed:

- The user determines the component-specific parameters. These include:
    - The prototype filter to be used when a filter is created. This is the same format as that used when using a filter file. All filter features are supported, with the exception of the *ESTABLISHED* and *REPEATING* keywords. Note that the source address/mask/port and destination address/mask/port fields of the filter will be overwritten during creation of the filter, according to the other parameters set in the configuration file.
    - Whether the filters created will block at the destination port or simply block all traffic from the source system (the *BLOCK_AT_DESTINATION_PORT* keyword).
    - The logging to be used.
- The user determines the ruleset:
    - The user determines what rules are to be supported. There's no limit on the number of rules the filter server can maintain; the limit is depends on how complex you want to make the component.
    - For each rule, you need to determine:
        - The name of the rule. This string (maximum length 35 characters) will be used by the filter server and by the call to the filter server API call *send_filter_event*.
        - The number of events/unit time that will trigger a filter (the *MAX_COUNT* and *DELTA_TIME* fields).
        - The duration(s) of a filter. Up to 5 may be chosen, and the list must end with -1.
- The user creates the component-specific configuration file, then adds a reference to it via the *INCLUDE* keyword in the main filter server configuration file. At this point, the filter server can be made aware of this new (or updated) configuration by using the TCPware NETCU SET IPS/RELOAD command.

Note! The filter server configuration may be reloaded multiple times without causing problems for the filter server.

Filter Server API

There are two calls available in the filter server API. The function prototypes are defined in TCPWARE_COMMON:[TCPWARE.INCLUDE]FILTER_SERVER_API.H. The first call is used to register with the filter server:

```
int filter_server_register(char *component, 0, 0)
```

where

component is the name of the component

The remaining two arguments are there for future expansion, and are ignored, but must be specified as zero.

The return values from this function are 1 (success) or 0 (an error occurred; most likely, this is because the filter server isn't running). Normally this function is called once when the first event is logged. However, if an error is returned, it may be called again when additional events are logged.

Note! The application that's registering MUST be an installed image, using /OPEN or /SHARED. It doesn't need to be installed with privileges. This is an attempt to help cut down on bogus applications registering with the server; it takes a conscious effort by the system manager to do this and therefore, to control this.

The next function is used to format and send events to the filter server:

```
int send_filter_event(char *rule,
                      char *source_address,
                      u_short source_port,
                      char *dest_address)
```

where

rule is the name of the rule to be enforced. This must correspond to a *rule* keyword specified in the per-component configuration file for the component. If a match cannot be made, the event will be ignored by the filter server.

source_address is the address of the system that caused the event to be logged (e.g, “192.168.0.1”). This may be in ipv4 or ipv6 format, but must be of the same address family as that of the *destination_address* specified for the component in the per-component configuration file. Note that this is an address only. Do not specify address mask bits (e.g., “192.168.0.1/24”) with it.

source_port is the source port on the originating system.

dest_address is the destination address of the socket used to communicate to *source_address*. This information may be obtained by performing a *getsockname* function on the socket. Note that this is an address only. Do not specify address mask bits (e.g., “192.168.0.11/24”) with it.

To include these routines in your application, link using the library TCPWARE_COMMON:[TCPWARE]FILTER_SERVER_API.OLB.

The following is an example of code used to send events to the filter server:

```
void ssh_send_filter_event(int code, char *addr, int port, char *dest_addr)
{
    char *rule;
    static int filter_server = -1;

    if (filter_server == -1)
        filter_server = filter_server_register("SSH", 0, 0);

    if (!filter_server)
        return;

    switch(code)
    {
        case LGI$_NOSUCHUSER:
        case LGI$_NOTVALID:
            rule = "SSH_INVALIDUSER";
            break;

        case LGI$_USERAUTH:
            rule = "SSH_USERAUTH";
            break;

        case LGI$_DISUSER:
        case LGI$_ACNTEXPIR:
        case LGI$_RESTRICT:
        case LGI$_INVPWD:
        case LGI$_PWDEXPIR:
            rule = "SSH_AUTHFAILED";
            break;

        default:
            printf("Unrecognized status code %d", code));
            return;
    }

    send_filter_event(rule, addr, (unsigned short)port, dest_addr);
}
```