CHAPTER 42

Configuring Inspection for Management Application Protocols

This chapter describes how to configure application layer protocol inspection. Inspection engines are required for services that embed IP addressing information in the user data packet or that open secondary channels on dynamically assigned ports. These protocols require the adaptive security appliance to do a deep packet inspection instead of passing the packet through the fast path. As a result, inspection engines can affect overall throughput.

Several common inspection engines are enabled on the adaptive security appliance by default, but you might need to enable others depending on your network. This chapter includes the following sections:

- DCERPC Inspection, page 42-1
- GTP Inspection, page 42-3
- RADIUS Accounting Inspection, page 42-9
- RSH Inspection, page 42-11
- SNMP Inspection, page 42-11
- XDMCP Inspection, page 42-12

DCERPC Inspection

This section describes the DCERPC inspection engine. This section includes the following topics:

- DCERPC Overview, page 42-1
- Configuring a DCERPC Inspection Policy Map for Additional Inspection Control, page 42-2

DCERPC Overview

DCERPC is a protocol widely used by Microsoft distributed client and server applications that allows software clients to execute programs on a server remotely.

This typically involves a client querying a server called the Endpoint Mapper listening on a well known port number for the dynamically allocated network information of a required service. The client then sets up a secondary connection to the server instance providing the service. The security appliance allows the appropriate port number and network address and also applies NAT, if needed, for the secondary connection.

DCERPC inspect maps inspect for native TCP communication between the EPM and client on well known TCP port 135. Map and lookup operations of the EPM are supported for clients. Client and server can be located in any security zone. The embedded server IP address and Port number are received from the applicable EPM response messages. Since a client may attempt multiple connections to the server port returned by EPM, multiple use of pinholes are allowed, which have user configurable timeouts.

Note DCERPC inspection only supports communication between the EPM and clients to open pinholes through theadaptive security appliance. Clients using RPC communication that does not use the EPM is not supported with DCERPC inspection.

Configuring a DCERPC Inspection Policy Map for Additional Inspection Control

To specify additional DCERPC inspection parameters, create a DCERPC inspection policy map. You can then apply the inspection policy map when you enable DCERPC inspection.

To create a DCERPC inspection policy map, perform the following steps:

Step 1 Create a DCERPC inspection policy map, enter the following command:

```
hostname(config)# policy-map type inspect dcerpc policy_map_name
hostname(config-pmap)#
```

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

Step 3 To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b. To configure the timeout for DCERPC pinholes and override the global system pinhole timeout of two minutes, enter the following command:

```
hostname(config-pmap-p)# timeout pinhole hh:mm:ss
```

Where the *hh:mm:ss* argument is the timeout for pinhole connections. Value is between 0:0:1 and 1193:0:0.

c. To configure options for the endpoint mapper traffic, enter the following command:

```
hostname(config-pmap-p)# endpoint-mapper [epm-service-only] [lookup-operation
[timeout hh:mm:ss]]
```

Where the *hh:mm:ss* argument is the timeout for pinholes generated from the lookup operation. If no timeout is configured for the lookup operation, the timeout pinhole command or the default is used. The **epm-service-only** keyword enforces endpoint mapper service during binding so that only its service traffic is processed. The **lookup-operation** keyword enables the lookup operation of the endpoint mapper service.

The following example shows how to define a DCERPC inspection policy map with the timeout configured for DCERPC pinholes.

```
hostname(config)# policy-map type inspect dcerpc dcerpc_map
hostname(config-pmap)# timeout pinhole 0:10:00

hostname(config)# class-map dcerpc
hostname(config-cmap)# match port tcp eq 135

hostname(config)# policy-map global-policy
hostname(config-pmap)# class dcerpc
hostname(config-pmap-c)# inspect dcerpc dcerpc-map

hostname(config)# service-policy global-policy global
```

GTP Inspection

This section describes the GTP inspection engine. This section includes the following topics:

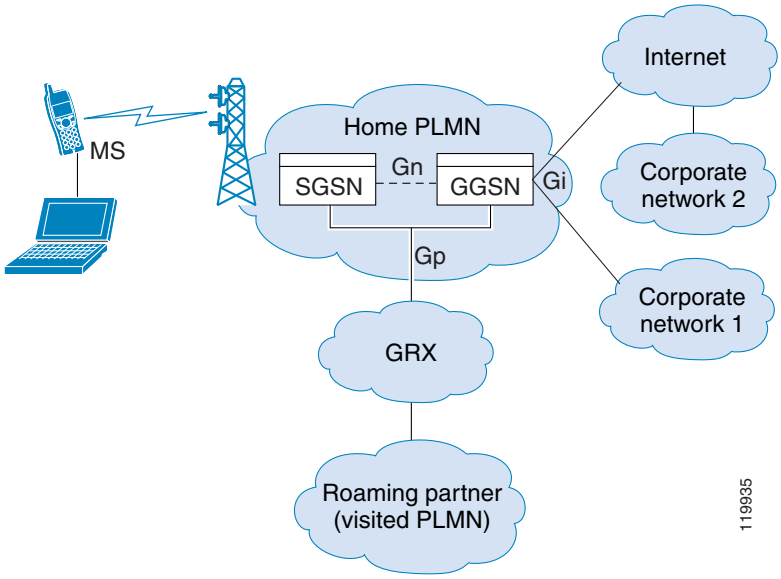
- GTP Inspection Overview, page 42-3
- Configuring a GTP Inspection Policy Map for Additional Inspection Control, page 42-4
- Verifying and Monitoring GTP Inspection, page 42-8

Note GTP inspection requires a special license. If you enter GTP-related commands on a adaptive security appliance without the required license, the adaptive security appliance displays an error message.

GTP Inspection Overview

GPRS provides uninterrupted connectivity for mobile subscribers between GSM networks and corporate networks or the Internet. The GGSN is the interface between the GPRS wireless data network and other networks. The SGSN performs mobility, data session management, and data compression (See Figure 42-1).

Figure 42-1 GPRS Tunneling Protocol



The UMTS is the commercial convergence of fixed-line telephony, mobile, Internet and computer technology. UTRAN is the networking protocol used for implementing wireless networks in this system. GTP allows multi-protocol packets to be tunneled through a UMTS/GPRS backbone between a GGSN, an SGSN and the UTRAN.

GTP does not include any inherent security or encryption of user data, but using GTP with the adaptive security appliance helps protect your network against these risks.

The SGSN is logically connected to a GGSN using GTP. GTP allows multiprotocol packets to be tunneled through the GPRS backbone between GSNs. GTP provides a tunnel control and management protocol that allows the SGSN to provide GPRS network access for a mobile station by creating, modifying, and deleting tunnels. GTP uses a tunneling mechanism to provide a service for carrying user data packets.

Note When using GTP with failover, if a GTP connection is established and the active unit fails before data is transmitted over the tunnel, the GTP data connection (with a "j" flag set) is not replicated to the standby unit. This occurs because the active unit does not replicate embryonic connections to the standby unit.

Configuring a GTP Inspection Policy Map for Additional Inspection Control

If you want to enforce additional parameters on GTP traffic, create and configure a GTP map. If you do not specify a map with the **inspect gtp** command, the adaptive security appliance uses the default GTP map, which is preconfigured with the following default values:

- **request-queue 200**
- **timeout gsn 0:30:00**
- **timeout pdp-context 0:30:00**
- **timeout request 0:01:00**

- **timeout signaling 0:30:00**
- **timeout tunnel 0:01:00**
- **tunnel-limit 500**

To create and configure a GTP map, perform the following steps. You can then apply the GTP map when you enable GTP inspection according to the "Configuring Application Layer Protocol Inspection" section on page 38-6.

Step 1 Create a GTP inspection policy map, enter the following command:

```
hostname(config)# policy-map type inspect gtp policy_map_name
hostname(config-pmap)#
```

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) To add a description to the policy map, enter the following command:

```
hostname(config-pmap)# description string
```

Step 3 To match an Access Point name, enter the following command:

```
hostname(config-pmap)# match [not] apn regex [regex_name | class regex_class_name]
```

Step 4 To match a message ID, enter the following command:

```
hostname(config-pmap)# match [not] message id [message_id | range lower_range upper_range]
```

Where the *message_id* is an alphanumeric identifier between 1 and 255. The *lower_range* is lower range of message IDs. The *upper_range* is the upper range of message IDs.

Step 5 To match a message length, enter the following command:

```
hostname(config-pmap)# match [not] message length min min_length max max_length
```

Where the *min_length* and *max_length* are both between 1 and 65536. The length specified by this command is the sum of the GTP header and the rest of the message, which is the payload of the UDP packet.

Step 6 To match the version, enter the following command:

```
hostname(config-pmap)# match [not] version [version_id | range lower_range upper_range]
```

Where the *version_id* is between 0and 255. The *lower_range* is lower range of versions. The *upper_range* is the upper range of versions.

Step 7 To configure parameters that affect the inspection engine, perform the following steps:

a. To enter parameters configuration mode, enter the following command:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

The **mnc** *network_code* argument is a two or three-digit value identifying the network code.

By default, the security appliance does not check for valid MCC/MNC combinations. This command is used for IMSI Prefix filtering. The MCC and MNC in the IMSI of the received packet is compared with the MCC/MNC configured with this command and is dropped if it does not match.

This command must be used to enable IMSI Prefix filtering. You can configure multiple instances to specify permitted MCC and MNC combinations. By default, the adaptive security appliance does not check the validity of MNC and MCC combinations, so you must verify the validity of the combinations configured. To find more information about MCC and MNC codes, see the ITU E.212 recommendation, *Identification Plan for Land Mobile Stations*.

b. To allow invalid GTP packets or packets that otherwise would fail parsing and be dropped, enter the following command:

```
hostname(config-pmap-p)# permit errors
```

By default, all invalid packets or packets that failed, during parsing, are dropped.

c. To enable support for GSN pooling, use the **permit response** command.

If the adaptive security appliance performs GTP inspection, by default the adaptive security appliance drops GTP responses from GSNs that were not specified in the GTP request. This situation occurs when you use load-balancing among a pool of GSNs to provide efficiency and scalability of GPRS.

You can enable support for GSN pooling by using the **permit response** command. This command configures the adaptive security appliance to allow responses from any of a designated set of GSNs, regardless of the GSN to which a GTP request was sent. You identify the pool of load-balancing GSNs as a network object. Likewise, you identify the SGSN as a network object. If the GSN responding belongs to the same object group as the GSN that the GTP request was sent to and if the SGSN is in a object group that the responding GSN is permitted to send a GTP response to, the adaptive security appliance permits the response.

d. To create an object to represent the pool of load-balancing GSNs, perform the following steps:

Use the **object-group** command to define a new network object group representing the pool of load-balancing GSNs.

```
hostname(config)# object-group network GSN-pool-name
hostname(config-network)#
```

For example, the following command creates an object group named gsnpool32:

```
hostname(config)# object-group network gsnpool32
hostname(config-network)#
```

e. Use the **network-object** command to specify the load-balancing GSNs. You can do so with one **network-object** command per GSN, using the **host** keyword. You can also using **network-object** command to identify whole networks containing GSNs that perform load balancing.

```
hostname(config-network)# network-object host IP-address
```

For example, the following commands create three network objects representing individual hosts:

```
hostname(config-network)# network-object host 192.168.100.1
hostname(config-network)# network-object host 192.168.100.2
hostname(config-network)# network-object host 192.168.100.3
hostname(config-network)#
```

f. To create an object to represent the SGSN that the load-balancing GSNs are permitted to respond to, perform the following steps:

a. Use the **object-group** command to define a new network object group that will represent the SGSN that sends GTP requests to the GSN pool.

```
hostname(config)# object-group network SGSN-name
hostname(config-network)#
```

For example, the following command creates an object group named sgsn32:

```
hostname(config)# object-group network sgsn32
hostname(config-network)#
```

b. Use the **network-object** command with the **host** keyword to identify the SGSN.

```
hostname(config-network)# network-object host IP-address
```

For example, the following command creates a network objects representing the SGSN:

```
hostname(config-network)# network-object host 192.168.50.100
hostname(config-network)#
```

g. To allow GTP responses from any GSN in the network object representing the GSN pool, defined in c., d, to the network object representing the SGSN, defined in c., f., enter the following commands:

```
hostname(config)# gtp-map map_name
hostname(config-gtp-map)# permit response to-object-group SGSN-name from-object-group
GSN-pool-name
```

For example, the following command permits GTP responses from any host in the object group named gsnpool32 to the host in the object group named sgsn32:

```
hostname(config-gtp-map)# permit response to-object-group sgsn32 from-object-group
gsnpool32
```

The following example shows how to support GSN pooling by defining network objects for the GSN pool and the SGSN. An entire Class C network is defined as the GSN pool but you can identify multiple individual IP addresses, one per **network-object** command, instead of identifying whole networks. The example then modifies a GTP map to permit responses from the GSN pool to the SGSN.

```
hostname(config)# object-group network gsnpool32
hostname(config-network)# network-object 192.168.100.0 255.255.255.0
hostname(config)# object-group network sgsn32
hostname(config-network)# network-object host 192.168.50.100
hostname(config)# gtp-map gtp-policy
hostname(config-gtp-map)# permit response to-object-group sgsn32 from-object-group
gsnpool32
```

h. To specify the maximum number of GTP requests that will be queued waiting for a response, enter the following command:

```
hostname(config-gtp-map)# request-queue max_requests
```

where the *max_requests* argument sets the maximum number of GTP requests that will be queued waiting for a response, from 1 to 4294967295. The default is 200.

When the limit has been reached and a new request arrives, the request that has been in the queue for the longest time is removed. The Error Indication, the Version Not Supported and the SGSN Context Acknowledge messages are not considered as requests and do not enter the request queue to wait for a response.

i. To change the inactivity timers for a GTP session, enter the following command:

```
hostname(config-gtp-map)# timeout {gsn | pdp-context | request | signaling | tunnel}
hh:mm:ss
```

Enter this command separately for each timeout.

The **gsn** keyword specifies the period of inactivity after which a GSN will be removed.

The **pdp-context** keyword specifies the maximum period of time allowed before beginning to receive the PDP context.

The **request** keyword specifies the maximum period of time allowed before beginning to receive the GTP message.

The **signaling** keyword specifies the period of inactivity after which the GTP signaling will be removed.

The **tunnel** keyword specifies the period of inactivity after which the GTP tunnel will be torn down.

The *hh*:*mm*:*ss* argument is the timeout where *hh* specifies the hour, *mm* specifies the minutes, and *ss* specifies the seconds. The value **0** means never tear down.

j. To specify the maximum number of GTP tunnels allowed to be active on the adaptive security appliance, enter the following command:

```
hostname(config-gtp-map)# tunnel-limit max_tunnels
```

where the *max_tunnels* argument is the maximum number of tunnels allowed, from 1 to 4294967295. The default is 500.

New requests will be dropped once the number of tunnels specified by this command is reached.

The following example shows how to limit the number of tunnels in the network:

```
hostname(config)# policy-map type inspect gtp gmap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# tunnel-limit 3000

hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect gtp gmap

hostname(config)# service-policy global_policy global
```

Verifying and Monitoring GTP Inspection

To display GTP configuration, enter the **show service-policy inspect gtp** command in privileged EXEC mode. For the detailed syntax for this command, see the command page in the *Cisco ASA 5500 Series Command Reference*.

Use the **show service-policy inspect gtp statistics** command to show the statistics for GTP inspection. The following is sample output from the **show service-policy inspect gtp statistics** command:

```
hostname# show service-policy inspect gtp statistics
GPRS GTP Statistics:
  version_not_support               0     msg_too_short             0
  unknown_msg                       0     unexpected_sig_msg        0
  unexpected_data_msg               0     ie_duplicated             0
  mandatory_ie_missing              0     mandatory_ie_incorrect    0
  optional_ie_incorrect             0     ie_unknown                0
  ie_out_of_order                   0     ie_unexpected             0
  total_forwarded                   0     total_dropped             0
  signalling_msg_dropped            0     data_msg_dropped          0
  signalling_msg_forwarded          0     data_msg_forwarded        0
  total created_pdp                 0     total deleted_pdp         0
  total created_pdpmcb              0     total deleted_pdpmcb      0
  pdp_non_existent                  0
```

You can use the vertical bar (|) to filter the display. Type **?|** for more display filtering options.

The following is sample GSN output from the **show service-policy inspect gtp statistics gsn** command:

```
hostname# show service-policy inspect gtp statistics gsn 9.9.9.9
1 in use, 1 most used, timeout 0:00:00

GTP GSN Statistics for 9.9.9.9, Idle 0:00:00, restart counter 0
    Tunnels Active 0Tunnels Created 0
    Tunnels Destroyed 0
    Total Messages Received 2
    Signaling Messages Data Messages
    total received 2 0
    dropped 0 0
    forwarded 2 0
```

Use the **show service-policy inspect gtp pdp-context** command to display PDP context-related information. The following is sample output from the **show service-policy inspect gtp pdp-context** command:

```
hostname# show service-policy inspect gtp pdp-context detail
1 in use, 1 most used, timeout 0:00:00

Version TID                 MS Addr        SGSN Addr    Idle      APN
v1      1234567890123425         10.0.1.1        10.0.0.2 0:00:13  gprs.cisco.com

    user_name (IMSI): 214365870921435    MS address:         1.1.1.1
    primary pdp: Y                       nsapi: 2
    sgsn_addr_signal:        10.0.0.2    sgsn_addr_data:          10.0.0.2
    ggsn_addr_signal:        10.1.1.1    ggsn_addr_data:          10.1.1.1
    sgsn control teid:     0x000001d1    sgsn data teid:        0x000001d3
    ggsn control teid:     0x6306ffa0    ggsn data teid:        0x6305f9fc
    seq_tpdu_up:                  0    seq_tpdu_down:                 0
    signal_sequence:              0
    upstream_signal_flow:         0     upstream_data_flow:          0
    downstream_signal_flow:       0     downstream_data_flow:        0
    RAupdate_flow:                0
```

The PDP context is identified by the tunnel ID, which is a combination of the values for IMSI and NSAPI. A GTP tunnel is defined by two associated PDP contexts in different GSN nodes and is identified with a Tunnel ID. A GTP tunnel is necessary to forward packets between an external packet data network and a MS user.

You can use the vertical bar (|) to filter the display, as in the following example:

```
hostname# show service-policy gtp statistics  |  grep gsn
```

RADIUS Accounting Inspection

This section describes the IM inspection engine. This section includes the following topics:

- RADIUS Accounting Inspection Overview, page 42-9
- Configuring a RADIUS Inspection Policy Map for Additional Inspection Control, page 42-10

RADIUS Accounting Inspection Overview

One of the well known problems is the over-billing attack in GPRS networks. The over-billing attack can cause consumers anger and frustration by being billed for services that they have not used. In this case, a malicious attacker sets up a connection to a server and obtains an IP address from the SGSN.

When the attacker ends the call, the malicious server will still send packets to it, which gets dropped by the GGSN, but the connection from the server remains active. The IP address assigned to the malicious attacker gets released and reassigned to a legitimate user who will then get billed for services that the attacker will use.

RADIUS accounting inspection prevents this type of attack by ensuring the traffic seen by the GGSN is legitimate. With the RADIUS accounting feature properly configured, the security appliance tears down a connection based on matching the Framed IP attribute in the Radius Accounting Request Start message with the Radius Accounting Request Stop message. When the Stop message is seen with the matching IP address in the Framed IP attribute, the security appliance looks for all connections with the source matching the IP address.

You have the option to configure a secret pre-shared key with the RADIUS server so the security appliance can validate the message. If the shared secret is not configured, the security appliance does not need to validate the source of the message and will only check that the source IP address is one of the configured addresses allowed to send the RADIUS messages.

Note When using RADIUS accounting inspection with GPRS enabled, theadaptive security appliance checks for the 3GPP-Session-Stop-Indicator in the Accounting Request STOP messages to properly handle secondary PDP contexts. Specifically, the adaptive security appliance requires that the Accounting Request STOP messages include the 3GPP-SGSN-Address attribute before it will temrinate the user sessions and all associated connections. Some third-party GGSNs might not send this attribute by default.

Configuring a RADIUS Inspection Policy Map for Additional Inspection Control

In order to use this feature, the **radius-accounting-map** will need to be specified in the **policy-map type management** and then applied to the service-policy using the new **control-plane** keyword to specify that this traffic is for to-the-box inspection.

The following example shows the complete set of commands in context to properly configure this feature:

Step 1 Configure the class map and the port:

```
class-map type management c1
  match port udp eq 1888
```

Step 2 Create the policy map, and configure the parameters for RADIUS accounting inspection using the parameter command to access the proper mode to configure the attributes, host, and key.

```
policy-map type inspect radius-accounting radius_accounting_map
  parameters
    host 10.1.1.1 inside key 123456789
    send response
    enable gprs
    validate-attribute 22
```

Step 3 Configure the service policy and control-plane keywords.

```
policy-map type management global_policy
  class c1
     inspect radius-accounting radius_accounting_map

service-policy global_policy control-plane abc global
```

RSH Inspection

RSH inspection is enabled by default. The RSH protocol uses a TCP connection from the RSH client to the RSH server on TCP port 514. The client and server negotiate the TCP port number where the client listens for the STDERR output stream. RSH inspection supports NAT of the negotiated port number if necessary.

SNMP Inspection

This section describes the IM inspection engine. This section includes the following topics:

- SNMP Inspection Overview, page 42-11
- Configuring an SNMP Inspection Policy Map for Additional Inspection Control, page 42-11

SNMP Inspection Overview

SNMP application inspection lets you restrict SNMP traffic to a specific version of SNMP. Earlier versions of SNMP are less secure; therefore, denying certain SNMP versions may be required by your security policy. The adaptive security appliance can deny SNMP versions 1, 2, 2c, or 3. You control the versions permitted by creating an SNMP map.

You then apply the SNMP map when you enable SNMP inspection according to the "Configuring Application Layer Protocol Inspection" section on page 38-6.

Configuring an SNMP Inspection Policy Map for Additional Inspection Control

To create an SNMP inspection policy map, perform the following steps:

Step 1 To create an SNMP map, enter the following command:

```
hostname(config)# snmp-map map_name
hostname(config-snmp-map)#
```

where *map_name* is the name of the SNMP map. The CLI enters SNMP map configuration mode.

Step 2 To specify the versions of SNMP to deny, enter the following command for each version:

```
hostname(config-snmp-map)# deny version version
hostname(config-snmp-map)#
```

where *version* is 1, 2, 2c, or 3.

The following example denies SNMP Versions 1 and 2:

```
hostname(config)# snmp-map sample_map
hostname(config-snmp-map)# deny version 1
hostname(config-snmp-map)# deny version 2
```

XDMCP Inspection

XDMCP inspection is enabled by default; however, the XDMCP inspection engine is dependent upon proper configuration of the **established** command.

XDMCP is a protocol that uses UDP port 177 to negotiate X sessions, which use TCP when established.

For successful negotiation and start of an XWindows session, the adaptive security appliance must allow the TCP back connection from the Xhosted computer. To permit the back connection, use the **established** command on the adaptive security appliance. Once XDMCP negotiates the port to send the display, The **established** command is consulted to verify if this back connection should be permitted.

During the XWindows session, the manager talks to the display Xserver on the well-known port 6000 | n. Each display has a separate connection to the Xserver, as a result of the following terminal setting.

```
setenv DISPLAY Xserver:n
```

where *n* is the display number.

When XDMCP is used, the display is negotiated using IP addresses, which the adaptive security appliance can NAT if needed. XDCMP inspection does not support PAT.