Inspection of Database, Directory, and Management Protocols

The following topics explain application inspection for database, directory, and management protocols. For information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection.

DCERPC Inspection

DCERPC inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add DCERPC inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe the DCERPC inspection engine.

DCERPC Overview

Microsoft Remote Procedure Call (MSRPC), based on 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.

The DCERPC inspection engine inspects 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 configurable timeouts.

DCE inspection supports the following universally unique identifiers (UUIDs) and messages:

- End point mapper (EPM) UUID. All EPM messages are supported.
- ISystemMapper UUID (non-EPM). Supported messages are:
  - RemoteCreateInstance opnum4
  - RemoteGetClassObject opnum3
- Any message that does not contain an IP address or port information because these messages do not require inspection.

Configure a DCERPC Inspection Policy Map

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.

Procedure

Step 1 Create a DCERPC inspection policy map: **policy-map type inspect dcerpc** *policy_map_name*

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

Step 2 (Optional) Add a description to the policy map: **description** *string*

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

a) Enter parameters configuration mode:

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

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **timeout pinhole** *hh:mm:ss*—Configures the timeout for DCERPC pinholes and override the global system pinhole timeout of two minutes. The timeout can be from 00:00:01 to 119:00:00.
- **endpoint-mapper** [**epm-service-only**] [**lookup-operation** [**timeout** *hh:mm:ss*]]—Configures options for the endpoint mapper traffic. 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. You can configure 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.

Examples

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
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

GTP Inspection

The following sections describe the GTP inspection engine.

Note GTP inspection requires a special license, which is not supported on all device models. For detailed information, see the tables in the licensing chapter of the general configuration guide.

GTP Inspection Overview

GPRS Tunneling Protocol is used in GSM, UMTS and LTE networks for general packet radio service (GPRS) traffic. GTP provides a tunnel control and management protocol to provide GPRS network access for a mobile station by creating, modifying, and deleting tunnels. GTP also uses a tunneling mechanism for carrying user data packets.

Service provider networks use GTP to tunnel multi-protocol packets through the GPRS backbone between endpoints. In GTPv0-1, GTP is used for signaling between gateway GPRS support nodes (GGSN) and serving GPRS support nodes (SGSN). The GGSN is the interface between the GPRS wireless data network and other networks. The SGSN performs mobility, data session management, and data compression.

You can use the ASA to provide protection against rogue roaming partners. Place the device between the home GGSN and visited SGSN endpoints and use GTP inspection on the traffic. GTP inspection works only on traffic between these endpoints. .

GTP and associated standards are defined by 3GPP (3rd Generation Partnership Project). For detailed information, see http://www.3gpp.org.

Defaults for GTP Inspection

GTP inspection is not enabled by default. However, if you enable it without specifying your own inspection map, a default map is used that provides the following processing. You need to configure a map only if you want different values.

- Errors are not permitted.
- The maximum number of requests is 200.
- The maximum number of tunnels is 500. This is equivalent to the number of PDP contexts (endpoints).
- The GSN timeout is 30 minutes.
- The PDP context timeout is 30 minutes.
- The request timeout is 1 minute.
- The signaling timeout is 30 minutes.
- The tunneling timeout is 1 hour.
- The T3 response timeout is 20 seconds.
- Unknown message IDs are dropped and logged.

  Messages are considered unknown if they are either undefined or are defined in GTP releases that the system does not support.

Configure GTP Inspection

GTP inspection is not enabled by default. You must configure it if you want GTP inspection.

Procedure

Step 1 Configure a GTP Inspection Policy Map, on page 4.

Step 2 Configure the GTP Inspection Service Policy, on page 7.

Step 3 (Optional) Configure RADIUS accounting inspection to protect against over-billing attacks. See RADIUS Accounting Inspection, on page 10.

Configure a GTP Inspection Policy Map

If you want to enforce additional parameters on GTP traffic, and the default map does not meet your needs, create and configure a GTP map.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 Create a GTP inspection policy map: **policy-map type inspect gtp** *policy_map_name*

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

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **apn regex** {*regex_name* | **class** *class_name*}—Matches the access point name (APN) against the specified regular expression or regular expression class.
- **match** [**not**] **message id** {*message_id* | **range** *message_id_1 message_id_2*}—Matches the message ID, which can be 1 to 255. You can specify a single ID or a range of IDs.
- **match** [**not**] **message length min** *bytes* **max** *bytes*—Matches messages where the length of the UDP payload (GTP header plus the rest of the message) is between the minimum and maximum values, from 1 to 65536.
- **match** [**not**] **version** {*version_id* | **range** *version_id_1 version_id_2*}—Matches the GTP version, which can be 0 to 255. You can specify a single version or a range of versions.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop all packets that match. Add the **log** keyword to also send a system log message.
- **rate-limit** *message_rate*—Limit the rate of messages. This option is available with **message id** only.

You can specify multiple **match** commands in the policy map. For information about the order of **match** commands, see How Multiple Traffic Classes are Handled.

Step 4 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

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

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **permit errors**—Allows invalid GTP packets or packets that otherwise would fail parsing and be dropped.
- **request-queue** *max_requests*—Sets the maximum number of GTP requests that will be queued waiting for a response. 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.

- **tunnel-limit** *max_tunnels*—Sets the maximum number of active GTP tunnels allowed. This is equivalent to the number of PDP contexts or endpoints. The default is 500. New requests will be dropped once the number of tunnels specified by this command is reached.
- **timeout** {**gsn** | **pdp-context** | **request** | **signaling** | **t3-response** | **tunnel**} *time*—Sets the idle timeout for the specified service (in hh:mm:ss format). To have no timeout, specify 0 for the number. Enter the command separately for each timeout.
  - **gsn**—The maximum period of inactivity before a GSN is removed.
  - **pdp-context**—The maximum period of inactivity before removing the PDP Context for a GTP session.
  - **request**—The maximum period of inactivity after which a request is removed from the request queue. Any subsequent responses to a dropped request will also be dropped.
  - **signaling**—The maximum period of inactivity before GTP signaling is removed.
  - **t3-response**—The maximum wait time for a response before removing the connection.
  - **tunnel**—The maximum period of inactivity for the GTP tunnel before it is torn down.

Step 5 While still in parameter configuration mode, configure IMSI prefix filtering, if desired:

mcc *country_code* **mnc** *network_code*

By default, GTP inspection does not check for valid Mobile Country Code (MCC)/Mobile Network Code (MNC) combinations. If you configure IMSI prefix filtering, the MCC and MNC in the IMSI of the received packet is compared with the configured MCC/MNC combinations and is dropped if it does not match.

The Mobile Country Code is a non-zero, three-digit value; add zeros as a prefix for one- or two-digit values. The Mobile Network Code is a two- or three-digit value.

Add all permitted MCC and MNC combinations. By default, the ASA 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*.

Step 6 While still in parameter configuration mode, configure GSN pooling, if desired.

permit-response to-object-group *SGSN_name* **from-object-group** *GSN_pool*

When the ASA performs GTP inspection, by default the ASA 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.

To configure GSN pooling and thus support load balancing, create a network object group that specifies the GSNs and specify this on the **from-object-group** parameter. Likewise, create a network object group for the SGSN and select it on the **to-object-group** parameter. 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 an object group that the responding GSN is permitted to send a GTP response to, the ASA permits the response.

The network object group can identify the GSN or SGSN by host address or by the subnet that contains them.

Example:

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 inspection 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)# policy-map type inspect gtp gtp-policy
hostname(config-pmap)# parameters
hostname(config-pmap-p)# permit-response to-object-group sgsn32
from-object-group gsnpool32
```

Example

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
```

Configure the GTP Inspection Service Policy

GTP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add GTP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

Step 1 If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map gtp_class_map
hostname(config-cmap)# match access-list gtp
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Create a Layer 3/4 Class Map for Through Traffic.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 class map you are using for GTP inspection: **class** *name*

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the *name*. Otherwise, you are specifying the class you created earlier in this procedure.

Step 4 Configure GTP inspection: **inspect gtp** [*gtp_policy_map*]

Where *gtp_policy_map* is the optional GTP inspection policy map. You need a map only if you want non-default inspection processing. For information on creating the inspection policy map, see Configure a GTP Inspection Policy Map, on page 4.

Example:

```
hostname(config-class)# no inspect gtp
hostname(config-class)# inspect gtp gtp-map
```

Note If you are editing the default global policy (or any in-use policy) to use a different inspection policy map, you must remove the GTP inspection with the **no inspect gtp** command, and then re-add it with the new inspection policy map name.

Step 5 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

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

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Monitoring GTP Inspection

To display the GTP configuration, enter the **show service-policy inspect gtp** command in privileged EXEC mode.

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

```
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
```

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

```
hostname# show service-policy inspect gtp statistics gsn 10.9.9.9
1 in use, 1 most used, timeout 0:30:00
 GTP GSN Statistics for 10.9.9.9, Idle 0:00:34, restart counter 0
  Tunnels Active                    0
  Tunnels Created                   1
  Tunnels Destroyed                 0
  Total Messages Received           1
                        Signalling Messages        Data Messages
  total received                    1                         0
  dropped                           0                         0
  forwarded                         1                         0
```

Use the **show service-policy inspect gtp pdp-context** command to display PDP context-related information. For example:

```
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:         10.0.1.1
    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 an MS user.

ILS Inspection

The Internet Locator Service (ILS) inspection engine provides NAT support for Microsoft NetMeeting, SiteServer, and Active Directory products that use LDAP to exchange directory information with an ILS server. You cannot use PAT with ILS inspection because only IP addresses are stored by an LDAP database.

For search responses, when the LDAP server is located outside, consider using NAT to allow internal peers to communicate locally while registered to external LDAP servers. If you do not need to use NAT, we recommend that you turn off the inspection engine to provide better performance.

Additional configuration may be necessary when the ILS server is located inside the ASA border. This would require a hole for outside clients to access the LDAP server on the specified port, typically TCP 389.

Note Because ILS traffic (H225 call signaling) only occurs on the secondary UDP channel, the TCP connection is disconnected after the TCP inactivity interval. By default, this interval is 60 minutes and can be adjusted using the TCP **timeout** command. In ASDM, this is on the **Configuration > Firewall > Advanced > Global Timeouts** pane.

ILS inspection has the following limitations:

- Referral requests and responses are not supported.
- Users in multiple directories are not unified.
- Single users having multiple identities in multiple directories cannot be recognized by NAT.

For information on enabling ILS inspection, see Configure Application Layer Protocol Inspection.

RADIUS Accounting Inspection

The following sections describe the RADIUS Accounting inspection engine.

RADIUS Accounting Inspection Overview

The purpose of RADIUS accounting inspection is to prevent over-billing attacks on GPRS networks that use RADIUS servers. Although you do not need the GTP/GPRS license to implement RADIUS accounting inspection, it has no purpose unless you are implementing GTP inspection and you have a GPRS setup.

The over-billing attack in GPRS networks results in consumers 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 ASA 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 ASA 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 ASA can validate the message. If the shared secret is not configured, the ASA 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, the ASA checks for the 3GPP-Session-Stop-Indicator in the Accounting Request STOP messages to properly handle secondary PDP contexts. Specifically, the ASA requires that the Accounting Request STOP messages include the 3GPP-SGSN-Address attribute before it will terminate the user sessions and all associated connections. Some third-party GGSNs might not send this attribute by default.

Configure RADIUS Accounting Inspection

RADIUS accounting inspection is not enabled by default. You must configure it if you want RADIUS accounting inspection.

Procedure

Step 1 Configure a RADIUS Accounting Inspection Policy Map, on page 11.

Step 2 Configure the RADIUS Accounting Inspection Service Policy, on page 13.

Configure a RADIUS Accounting Inspection Policy Map

You must create a RADIUS accounting inspection policy map to configure the attributes needed for the inspection.

Procedure

Step 1 Create a RADIUS accounting inspection policy map: **policy-map type inspect radius-accounting** *policy_map_name*

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

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode.

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

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

Step 4 Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option.

- **send response**—Instructs the ASA to send Accounting-Request Start and Stop messages to the sender of those messages (which are identified in the **host** command).
- **enable gprs**—Implement GPRS over-billing protection. The ASA checks for the 3GPP VSA 26-10415 attribute in the Accounting-Request Stop and Disconnect messages in order to properly handle secondary PDP contexts. If this attribute is present, then the ASA tears down all connections that have a source IP matching the User IP address on the configured interface.
- **validate-attribute** *number*—Additional criteria to use when building a table of user accounts when receiving Accounting-Request Start messages. These attributes help when the ASA decides whether to tear down connections.

  If you do not specify additional attributes to validate, the decision is based solely on the IP address in the Framed IP Address attribute. If you configure additional attributes, and the ASA receives a start accounting message that includes an address that is currently being tracked, but the other attributes to validate are different, then all connections started using the old attributes are torn down, on the assumption that the IP address has been reassigned to a new user.

  Values range from 1-191, and you can enter the command multiple times. For a list of attribute numbers and their descriptions, see http://www.iana.org/assignments/radius-types.
- **host** *ip_address* [**key** *secret*]—The IP address of the RADIUS server or GGSN. You can optionally include a secret key so that the ASA can validate the message. Without the key, only the IP address is checked. You can repeat this command to identify multiple RADIUS and GGSNs hosts. The ASA receives a copy of the RADIUS accounting messages from these hosts.
- **timeout users** *time*—Sets the idle timeout for users (in hh:mm:ss format). To have no timeout, specify 00:00:00. The default is one hour.

Example

```
policy-map type inspect radius-accounting radius-acct-pmap
  parameters
    send response
    enable gprs
    validate-attribute 31
    host 10.2.2.2 key 123456789
    host 10.1.1.1 key 12345
class-map type management radius-class
  match port udp eq radius-acct
policy-map global_policy
  class radius-class
    inspect radius-accounting radius-acct-pmap
```

Configure the RADIUS Accounting Inspection Service Policy

RADIUS accounting inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. Because RADIUS accounting inspection is for traffic directed to the ASA, you must configure it as a management inspection rule rather than a standard rule.

Procedure

Step 1 Create an L3/L4 management class map to identify the traffic for which you want to apply the inspection, and identify the matching traffic.

```
class-map type management name
match {port | access-list} parameter
```

Example:

```
hostname(config)# class-map type management radius-class-map
hostname(config-cmap)# match port udp eq radius-acct
```

In this example, the match is for the radius-acct UDP port, which is 1646. You can specify a different port, a range of ports (**match port udp range** *number1 number2*) or use **match access-list** *acl_name* and use an ACL.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 management class map you are using for RADIUS accounting inspection: **class** *name*

Example:

```
hostname(config-pmap)# class radius-class-map
```

Step 4 Configure RADIUS accounting inspection: **inspect radius-accounting** [*radius-accounting_policy_map*]

Where *radius_accounting_policy_map* is the RADIUS accounting inspection policy map you created in Configure a RADIUS Accounting Inspection Policy Map, on page 11.

Example:

```
hostname(config-class)# no inspect radius-accounting
hostname(config-class)# inspect radius-accounting radius-class-map
```

Note If you are editing an in-use policy to use a different inspection policy map, you must remove the RADIUS accounting inspection with the **no inspect radius-accounting** command, and then re-add it with the new inspection policy map name.

Step 5 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

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

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

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.

For information on enabling RSH inspection, see Configure Application Layer Protocol Inspection.

SNMP Inspection

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 ASA can deny SNMP versions 1, 2, 2c, or 3. You control the versions permitted by creating an SNMP map.

SNMP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add SNMP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

Create an SNMP map.

Use the **snmp-map** *map_name* command to create the map and enter SNMP map configuration mode, then the **deny version** *version* command to identify the versions to disallow. The version can be 1, 2, 2c, or 3.

Example:

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
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

SQL*Net Inspection

SQL*Net inspection is enabled by default. The inspection engine supports SQL*Net versions 1 and 2, but only the Transparent Network Substrate (TNS) format. Inspection does not support the Tabular Data Stream (TDS) format. SQL*Net messages are scanned for embedded addresses and ports, and NAT rewrite is applied when necessary.

The default port assignment for SQL*Net is 1521. This is the value used by Oracle for SQL*Net, but this value does not agree with IANA port assignments for Structured Query Language (SQL). If your application uses a different port, apply the SQL*Net inspection to a traffic class that includes that port.

Note Disable SQL*Net inspection when SQL data transfer occurs on the same port as the SQL control TCP port 1521. The security appliance acts as a proxy when SQL*Net inspection is enabled and reduces the client window size from 65000 to about 16000 causing data transfer issues.

For information on enabling SQL*Net inspection, see Configure Application Layer Protocol Inspection.

Sun RPC Inspection

This section describes Sun RPC application inspection.

Sun RPC Inspection Overview

Sun RPC protocol inspection is enabled by default. You simply need to manage the Sun RPC server table to identify which services are allowed to traverse the firewall. However, pinholing for NFS is done for any server even without the server table configuration.

Sun RPC is used by NFS and NIS. Sun RPC services can run on any port. When a client attempts to access a Sun RPC service on a server, it must learn the port that service is running on. It does this by querying the port mapper process, usually rpcbind, on the well-known port of 111.

The client sends the Sun RPC program number of the service and the port mapper process responds with the port number of the service. The client sends its Sun RPC queries to the server, specifying the port identified by the port mapper process. When the server replies, the ASA intercepts this packet and opens both embryonic TCP and UDP connections on that port.

NAT or PAT of Sun RPC payload information is not supported.

Manage Sun RPC Services

Use the Sun RPC services table to control Sun RPC traffic based on established Sun RPC sessions.

Procedure

Step 1 Configure the Sun RPC service properties.

sunrpc-server *interface_name ip_address mask* **service** *service_type* **protocol** {**tcp** | **udp**} *port*[*-port*] **timeout** *hh:mm:ss*

Where:

- *interface_name*—The interface through which traffic to the server flows.
- *ip_address mask*—The address of the Sun RPC server.
- **service** *service_type* —The service type on the server, which is the mapping between a specific service type and the port number used for the service. To determine the service type (for example, 100003), use the **sunrpcinfo** command at the UNIX or Linux command line on the Sun RPC server machine.
- **protocol** {**tcp** | **udp**}—Whether the service uses TCP or UDP.
- *port*[*-port*]—The port or range of ports used by the service. To specify a range of ports, separate the starting and ending port numbers in the range with a hyphen (for example, 111-113).
- **timeout** *hh:mm:ss*—The idle timeout for the pinhole opened for the connection by Sun RPC inspection.

Example:

For example, to create a timeout of 30 minutes to the Sun RPC server with the IP address 192.168.100.2, enter the following command. In this example, the Sun RPC server is on the inside interface using TCP port 111.

```
hostname(config)# sunrpc-server inside 192.168.100.2 255.255.255.255
service 100003 protocol tcp 111 timeout 00:30:00
```

Step 2 (Optional.) Monitor the pinholes created for these services.

To display the pinholes open for Sun RPC services, enter the **show sunrpc-server active** command. For example:

```
hostname# show sunrpc-server active
LOCAL FOREIGN SERVICE TIMEOUT
-----------------------------------------------
1 209.165.200.5/0 192.168.100.2/2049 100003 0:30:00
2 209.165.200.5/0 192.168.100.2/2049 100003 0:30:00
3 209.165.200.5/0 192.168.100.2/647 100005 0:30:00
4 209.165.200.5/0 192.168.100.2/650 100005 0:30:00
```

The entry in the LOCAL column shows the IP address of the client or server on the inside interface, while the value in the FOREIGN column shows the IP address of the client or server on the outside interface.

If necessary, you can clear these services using the **clear sunrpc-server active**

XDMCP Inspection

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 ASA must allow the TCP back connection from the Xhosted computer. To permit the back connection, you can use access rules to allow the TCP ports. Alternatively, you can use the **established** command on the ASA. 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 ASA can NAT if needed. XDCMP inspection does not support PAT.

For information on enabling XDMCP inspection, see Configure Application Layer Protocol Inspection.

VXLAN Inspection

Virtual Extensible Local Area Network (VXLAN) inspection works on VXLAN encapsulated traffic that passes through the ASA. It ensures that the VXLAN header format conforms to standards, dropping any malformed packets. VXLAN inspection is not done on traffic for which the ASA acts as a VXLAN Tunnel End Point (VTEP) or a VXLAN gateway, as those checks are done as a normal part of decapsulating VXLAN packets.

VXLAN packets are UDP, normally on port 4789. This port is part of the default-inspection-traffic class, so you can simply add VXLAN inspection to the inspection_default service policy rule. Alternatively, you can create a class for it using port or ACL matching.

History for Database, Directory, and Management Protocol Inspection

Feature Name	Releases	Feature Information
DCERPC inspection support for ISystemMapper UUID message RemoteGetClassObject opnum3.	9.4(1)	The ASA started supporting non-EPM DCERPC messages in release 8.3, supporting the ISystemMapper UUID message RemoteCreateInstance opnum4. This change extends support to the RemoteGetClassObject opnum3 message. We did not modify any commands.
VXLAN packet inspection	9.4(1)	The ASA can inspect the VXLAN header to enforce compliance with the standard format. We introduced the following command: **inspect vxlan**.