

Configuring Application Layer Protocol Inspection

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 security appliance to do a deep packet inspection instead of passing the packet through the fast path (see the “[Stateful Inspection Overview](#)” section on page 1-4 for more information about the fast path). As a result, inspection engines can affect overall throughput.

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

- [Inspection Engine Overview, page 22-2](#)
 - [When to Use Application Protocol Inspection, page 22-2](#)
 - [Inspection Limitations, page 22-2](#)
 - [Default Inspection Policy, page 22-3](#)
- [Configuring Application Inspection, page 22-5](#)
- [CTIQBE Inspection, page 22-8](#)
- [DNS Inspection, page 22-10](#)
- [FTP Inspection, page 22-18](#)
- [GTP Inspection, page 22-21](#)
- [H.323 Inspection, page 22-27](#)
- [HTTP Inspection, page 22-30](#)
- [ICMP Inspection, page 22-34](#)
- [ICMP Error Inspection, page 22-34](#)
- [ILS Inspection, page 22-34](#)
- [MGCP Inspection, page 22-35](#)
- [NetBIOS Inspection, page 22-39](#)
- [PPTP Inspection, page 22-39](#)
- [RSH Inspection, page 22-40](#)
- [RTSP Inspection, page 22-40](#)
- [SIP Inspection, page 22-42](#)

- [Skinny \(SCCP\) Inspection, page 22-44](#)
- [SMTP and Extended SMTP Inspection, page 22-46](#)
- [SNMP Inspection, page 22-47](#)
- [SQL*Net Inspection, page 22-48](#)
- [Sun RPC Inspection, page 22-48](#)
- [TFTP Inspection, page 22-51](#)
- [XDMCP Inspection, page 22-51](#)

Inspection Engine Overview

This section includes the following topics:

- [When to Use Application Protocol Inspection, page 22-2](#)
- [Inspection Limitations, page 22-2](#)
- [Default Inspection Policy, page 22-3](#)

When to Use Application Protocol Inspection

When a user establishes a connection, the security appliance checks the packet against access lists, creates an address translation, and creates an entry for the session in the fast path, so that further packets can bypass time-consuming checks. However, the fast path relies on predictable port numbers and does not perform address translations inside a packet.

Many protocols open secondary TCP or UDP ports. The initial session on a well-known port is used to negotiate dynamically assigned port numbers.

Other applications embed an IP address in the packet that needs to match the source address that is normally translated when it goes through the security appliance.

If you use applications like these, then you need to enable application inspection.

When you enable application inspection for a service that embeds IP addresses, the security appliance translates embedded addresses and updates any checksum or other fields that are affected by the translation.

When you enable application inspection for a service that uses dynamically assigned ports, the security appliance monitors sessions to identify the dynamic port assignments, and permits data exchange on these ports for the duration of the specific session.

Inspection Limitations

See the following limitations for application protocol inspection:

- State information for multimedia sessions that require inspection are not passed over the state link for stateful failover. The exception is GTP, which is replicated over the state link.
- Some inspection engines do not support PAT, NAT, outside NAT, or NAT between same security interfaces.

- For all the application inspections, the adaptive security appliance limits the number of simultaneous, active data connections to 200 connections. For example, if an FTP client opens multiple secondary connections, the FTP inspection engine allows only 200 active connections and the 201 connection is dropped and the adaptive security appliance generates a system error message.

Default Inspection Policy

By default, the configuration includes a policy that matches all default application inspection traffic and applies inspection to the traffic on all interfaces (a global policy). Default application inspection traffic includes traffic to the default ports for each protocol. You can only apply one global policy, so if you want to alter the global policy, for example, to apply inspection to non-standard ports, or to add inspections that are not enabled by default, you need to either edit the default policy or disable it and apply a new one.

[Table 22-1](#) lists all inspections supported, the default ports used in the default class map, and the inspection engines that are on by default, shown in bold. This table also notes any NAT limitations.

Table 22-1 Supported Application Inspection Engines

Application ¹	Default Port	NAT Limitations	Standards ²	Comments
CTIQBE	TCP/2748	—	—	—
DNS over UDP	UDP/53	No NAT support is available for name resolution through WINS.	RFC 1123	No PTR records are changed.
FTP	TCP/21	—	RFC 959	—
GTP	UDP/3386 UDP/2123	—	—	Requires a special license.
H.323 H.225 and RAS	TCP/1720 UDP/1718 UDP (RAS) 1718-1719	No NAT on same security interfaces. No static PAT.	ITU-T H.323, H.245, H225.0, Q.931, Q.932	—
HTTP	TCP/80	—	RFC 2616	Beware of MTU limitations stripping ActiveX and Java. If the MTU is too small to allow the Java or ActiveX tag to be included in one packet, stripping may not occur.
ICMP	—	—	—	All ICMP traffic is matched in the default class map.
ICMP ERROR	—	—	—	All ICMP traffic is matched in the default class map.
ILS (LDAP)	TCP/389	No PAT.	—	—
MGCP	UDP/2427, 2727	—	RFC 2705bis-05	—
NetBIOS Name Server over IP	UDP/137, 138 (Source ports)	—	—	NetBIOS is supported by performing NAT of the packets for NBNS UDP port 137 and NBDS UDP port 138.
PPTP	TCP/1723	—	RFC 2637	—

Table 22-1 Supported Application Inspection Engines (continued)

Application ¹	Default Port	NAT Limitations	Standards ²	Comments
RSH	TCP/514	No PAT	Berkeley UNIX	—
RTSP	TCP/554	No PAT. No outside NAT.	RFC 2326, 2327, 1889	No handling for HTTP cloaking.
SIP	TCP/5060 UDP/5060	No outside NAT. No NAT on same security interfaces.	RFC 2543	—
SKINNY (SCCP)	TCP/2000	No outside NAT. No NAT on same security interfaces.	—	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SMTP and ESMTP	TCP/25	—	RFC 821, 1123	—
SNMP	UDP/161, 162	No NAT or PAT.	RFC 1155, 1157, 1212, 1213, 1215	v.2 RFC 1902-1908; v.3 RFC 2570-2580.
SQL*Net	TCP/1521	—	—	v.1 and v.2.
Sun RPC over UDP and TCP	UDP/111	No NAT or PAT.	—	The default class map includes UDP port 111; if you want to enable Sun RPC inspection for TCP port 111, you need to create a new class map that matches TCP port 111, add the class to the policy, and then apply the inspect sunrpc command to that class.
TFTP	UDP/69	—	RFC 1350	Payload IP addresses are not translated.
XDCMP	UDP/177	No NAT or PAT.	—	—

1. Inspection engines that are enabled by default for the default port are in bold.
2. The security appliance is in compliance with these standards, but it does not enforce compliance on packets being inspected. For example, FTP commands are supposed to be in a particular order, but the security appliance does not enforce the order.

The default policy configuration includes the following commands:

```
class-map inspection_default
  match default-inspection-traffic
policy-map global_policy
  class inspection_default
    inspect dns maximum-length 512
    inspect ftp
    inspect h323 h225
    inspect h323 ras
    inspect rsh
    inspect smtp
    inspect sqlnet
    inspect skinny
    inspect sunrpc
    inspect xdmcp
    inspect sip
    inspect netbios
    inspect tftp
  service-policy global_policy global
```

Configuring Application Inspection

This feature uses Modular Policy Framework, so that implementing application inspection consists of identifying traffic, applying inspections to the traffic, and activating inspections on an interface. See [Chapter 18, “Using Modular Policy Framework,”](#) for more information.

Inspection is enabled by default for some applications. See the [“Default Inspection Policy”](#) section for more information. Use this section to modify your inspection policy.

To configure application inspection, perform the following steps:

- Step 1** To identify the traffic to which you want to apply inspections, add a class map using the **class-map** command. See the [“Identifying Traffic Using a Class Map”](#) section on page 18-2 for detailed information.

The default class map is called “inspection_default.” It matches traffic using a special **match** command, **match default-inspection-traffic**, to match the default ports for each application protocol.

You can specify a **match access-list** command along with the **match default-inspection-traffic** command to narrow the matched traffic to specific IP addresses. Because the **match default-inspection-traffic** command specifies the ports to match, any ports in the access list are ignored.

If you want to match non-standard ports, then create a new class map for the non-standard ports. See the [“Default Inspection Policy”](#) section on page 22-3 for the standard ports for each inspection engine. You can combine multiple class maps in the same policy if desired, so you can create one class map to match certain traffic, and another to match different traffic. However, if traffic matches a class map that contains an inspection command, and then matches another class map that also has an inspection command, only the first matching class is used. For example, SNMP matches the inspection_default class. To enable SNMP inspection, enable SNMP inspection for the default class in [Step 5](#). Do not add another class that matches SNMP.

For example, to limit inspection to traffic from 10.1.1.0 to 192.168.1.0 using the default class map, enter the following commands:

```
hostname(config)# access-list inspect extended permit ip 10.1.1.0 255.255.255.0
192.168.1.0 255.255.255.0
hostname(config)# class-map inspection_default
hostname(config-cmap)# match access-list inspect
```

View the entire class map using the following command:

```
hostname(config-cmap)# show running-config class-map inspection_default
!
class-map inspection_default
  match default-inspection-traffic
  match access-list inspect
!
```

To inspect FTP traffic on port 21 as well as 1056 (a non-standard port), create an access list that specifies the ports, and assign it to a new class map:

```
hostname(config)# access-list ftp_inspect extended permit tcp any any eq 21
hostname(config)# access-list ftp_inspect extended permit tcp any any eq 1056
hostname(config)# class-map new_inspection
hostname(config-cmap)# match access-list ftp_inspect
```

- Step 2** (Optional) Some inspection engines let you control additional parameters when you apply the inspection to the traffic. Before you apply the inspection in the policy map configuration, configure additional parameters using the following command:

```
hostname(config)# application-map application_map_name
```

where *application* is the type of application inspection. You can use the following map commands:

- **ftp-map**—See the “Configuring an FTP Map for Additional Inspection Control” section on page 22-19 for detailed information about commands available in the ftp-map configuration mode.
- **gtp-map**—See the “Configuring a GTP Map for Additional Inspection Control” section on page 22-22 for detailed information about commands available in the gtp-map configuration mode.
- **http-map**—See the “Configuring an HTTP Map for Additional Inspection Control” section on page 22-31 for detailed information about commands available in the http-map configuration mode.
- **mgcp-map**—See the “Configuring an MGCP Map for Additional Inspection Control” section on page 22-37 for detailed information about commands available in the mgcp-map configuration mode.
- **snmp-map**—See the “SNMP Inspection” section on page 22-47 for detailed information about commands available in the snmp-map configuration mode. If you enable SNMP inspection, you must use an SNMP map that identifies the SNMP versions you want to deny.

Step 3 To add or edit a policy map that sets the actions to take with the class map traffic, enter the following command:

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

The default policy map is called “global_policy.” This policy map includes the default inspections listed in the “Default Inspection Policy” section on page 22-3. If you want to modify the default policy (for example, to add or delete an inspection, or to identify an additional class map for your actions), then enter **global_policy** as the name.

Step 4 To identify the class map from Step 1 to which you want to assign an action, enter the following command:

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

If you are editing the default policy map, it includes the inspection_default class map. You can edit the actions for this class by entering **inspection_default** as the name. To add an additional class map to this policy map, identify a different name. You can combine multiple class maps in the same policy if desired, so you can create one class map to match certain traffic, and another to match different traffic. However, if traffic matches a class map that contains an inspection command, and then matches another class map that also has an inspection command, only the first matching class is used. For example, SNMP matches the inspection_default class map. To enable SNMP inspection, enable SNMP inspection for the default class in Step 5. Do not add another class that matches SNMP.

Step 5 Enable application inspection by entering the following command:

```
hostname(config-pmap-c)# inspect protocol
```

The *protocol* is one of the following values:

Table 22-2 Protocol Keywords

Keywords	Notes
ctiqbe	—
dns [maximum-length <i>max-pkt-length</i>]	To change the maximum DNS packet length from the default (512), use the maximum-length keyword and replace <i>max-pkt-length</i> with a numeric value. Longer packets are dropped. To disable checking the DNS packet length, enter the inspect dns command without the maximum-length keyword.

Table 22-2 Protocol Keywords

Keywords	Notes
<code>esmtpt</code>	—
<code>ftp [strict [map_name]]</code>	Use the strict keyword to increase the security of protected networks by preventing web browsers from sending embedded commands in FTP requests. See the “Using the strict Option” section on page 22-18 for more information. If you added an FTP map in Step 2, identify the map name in this command. For detailed information about adding an FTP map, see the “Configuring an FTP Map for Additional Inspection Control” section on page 22-19.
<code>gtp [map_name]</code>	If you added a GTP map in Step 2, identify the map name in this command. For detailed information about adding a GTP map, see the “Configuring a GTP Map for Additional Inspection Control” section on page 22-22
<code>h323 h225</code>	—
<code>h323 ras</code>	—
<code>http [map_name]</code>	If you added an HTTP map in Step 2, identify the map name in this command. For detailed information about adding an HTTP map, see the “Configuring an HTTP Map for Additional Inspection Control” section on page 22-31.
<code>icmp</code>	—
<code>icmp error</code>	—
<code>ils</code>	—
<code>mgcp [map_name]</code>	If you added an MGCP map in Step 2, identify the map name in this command. For detailed information about adding an MGCP map, see the “Configuring an MGCP Map for Additional Inspection Control” section on page 22-37.
<code>netbios</code>	—
<code>pptp</code>	—
<code>rsh</code>	—
<code>rtsp</code>	—
<code>sip</code>	—
<code>skinny</code>	—
<code>snmp map_name</code>	The SNMP inspection engine requires an SNMP map to deny specified SNMP versions. For detailed information about adding an SNMP map, see the “SNMP Inspection” section on page 22-47.
<code>sqlnet</code>	—
<code>sunrpc</code>	The default class map includes UDP port 111; if you want to enable Sun RPC inspection for TCP port 111, you need to create a new class map that matches TCP port 111, add the class to the policy, and then apply the inspect sunrpc command to that class.

Table 22-2 Protocol Keywords

Keywords	Notes
tftp	—
xmcp	—

Step 6 To activate the policy map on one or more interfaces, enter the following command:

```
hostname(config)# service-policy polycymap_name {global | interface interface_name}
```

Where **global** applies the policy map to all interfaces, and **interface** applies the policy to one interface. By default, the default policy map, “global_policy,” is applied globally. 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.

CTIQBE Inspection

This section describes CTIQBE application inspection. This section includes the following topics:

- [CTIQBE Inspection Overview, page 22-8](#)
- [Limitations and Restrictions, page 22-8](#)
- [Verifying and Monitoring CTIQBE Inspection, page 22-9](#)

CTIQBE Inspection Overview

CTIQBE protocol inspection supports NAT, PAT, and bidirectional NAT. This enables Cisco IP SoftPhone and other Cisco TAPI/JTAPI applications to work successfully with Cisco CallManager for call setup across the security appliance.

TAPI and JTAPI are used by many Cisco VoIP applications. CTIQBE is used by Cisco TSP to communicate with Cisco CallManager.

Limitations and Restrictions

The following summarizes limitations that apply when using CTIQBE application inspection:

- CTIQBE application inspection does not support configurations with the **alias** command.
- Stateful failover of CTIQBE calls is not supported.
- Entering the **debug ctique** command may delay message transmission, which may have a performance impact in a real-time environment. When you enable this debugging or logging and Cisco IP SoftPhone seems unable to complete call setup through the security appliance, increase the timeout values in the Cisco TSP settings on the system running Cisco IP SoftPhone.

The following summarizes special considerations when using CTIQBE application inspection in specific scenarios:

- If two Cisco IP SoftPhones are registered with different Cisco CallManagers, which are connected to different interfaces of the security appliance, calls between these two phones fails.

- When Cisco CallManager is located on the higher security interface compared to Cisco IP SoftPhones, if NAT or outside NAT is required for the Cisco CallManager IP address, the mapping must be static as Cisco IP SoftPhone requires the Cisco CallManager IP address to be specified explicitly in its Cisco TSP configuration on the PC.
- When using PAT or Outside PAT, if the Cisco CallManager IP address is to be translated, its TCP port 2748 must be statically mapped to the same port of the PAT (interface) address for Cisco IP SoftPhone registrations to succeed. The CTIQBE listening port (TCP 2748) is fixed and is not user-configurable on Cisco CallManager, Cisco IP SoftPhone, or Cisco TSP.

Verifying and Monitoring CTIQBE Inspection

The **show ctiqbe** command displays information regarding the CTIQBE sessions established across the security appliance. It shows information about the media connections allocated by the CTIQBE inspection engine.

The following is sample output from the **show ctiqbe** command under the following conditions. There is only one active CTIQBE session setup across the security appliance. It is established between an internal CTI device (for example, a Cisco IP SoftPhone) at local address 10.0.0.99 and an external Cisco CallManager at 172.29.1.77, where TCP port 2748 is the Cisco CallManager. The heartbeat interval for the session is 120 seconds.

```
hostname# # show ctiqbe

Total: 1
-----
LOCAL                FOREIGN              STATE    HEARTBEAT
-----
1                    10.0.0.99/1117     172.29.1.77/2748    1        120
-----
RTP/RTCP: PAT xlates: mapped to 172.29.1.99(1028 - 1029)
-----
MEDIA: Device ID 27      Call ID 0
      Foreign 172.29.1.99      (1028 - 1029)
      Local   172.29.1.88      (26822 - 26823)
-----
```

The CTI device has already registered with the CallManager. The device internal address and RTP listening port is PATed to 172.29.1.99 UDP port 1028. Its RTCP listening port is PATed to UDP 1029.

The line beginning with **RTP/RTCP: PAT xlates:** appears only if an internal CTI device has registered with an external CallManager and the CTI device address and ports are PATed to that external interface. This line does not appear if the CallManager is located on an internal interface, or if the internal CTI device address and ports are translated to the same external interface that is used by the CallManager.

The output indicates a call has been established between this CTI device and another phone at 172.29.1.88. The RTP and RTCP listening ports of the other phone are UDP 26822 and 26823. The other phone locates on the same interface as the CallManager because the security appliance does not maintain a CTIQBE session record associated with the second phone and CallManager. The active call leg on the CTI device side can be identified with Device ID 27 and Call ID 0.

The following is sample output from the **show xlate debug** command for these CTIBQE connections:

```
hostname# show xlate debug
3 in use, 3 most used
Flags: D - DNS, d - dump, I - identity, i - inside, n - no random,
      r - portmap, s - static
TCP PAT from inside:10.0.0.99/1117 to outside:172.29.1.99/1025 flags ri idle 0:00:22
timeout 0:00:30
```

```

UDP PAT from inside:10.0.0.99/16908 to outside:172.29.1.99/1028 flags ri idle 0:00:00
timeout 0:04:10
UDP PAT from inside:10.0.0.99/16909 to outside:172.29.1.99/1029 flags ri idle 0:00:23
timeout 0:04:10

```

The **show conn state ctiqbe** command displays the status of CTIQBE connections. In the output, the media connections allocated by the CTIQBE inspection engine are denoted by a ‘C’ flag. The following is sample output from the **show conn state ctiqbe** command:

```

hostname# show conn state ctiqbe
1 in use, 10 most used
hostname# show conn state ctiqbe detail
1 in use, 10 most used
Flags: A - awaiting inside ACK to SYN, a - awaiting outside ACK to SYN,
       B - initial SYN from outside, C - CTIQBE media, D - DNS, d - dump,
       E - outside back connection, F - outside FIN, f - inside FIN,
       G - group, g - MGCP, H - H.323, h - H.225.0, I - inbound data,
       i - incomplete, J - GTP, j - GTP data, k - Skinny media,
       M - SMTP data, m - SIP media, O - outbound data, P - inside back connection,
       q - SQL*Net data, R - outside acknowledged FIN,
       r - UDP RPC, r - inside acknowledged FIN, S - awaiting inside SYN,
       s - awaiting outside SYN, T - SIP, t - SIP transient, U - up

```

DNS Inspection

This section describes DNS application inspection. This section includes the following topics:

- [How DNS Application Inspection Works, page 22-10](#)
- [How DNS Rewrite Works, page 22-11](#)
- [Configuring DNS Rewrite, page 22-12](#)
- [Verifying and Monitoring DNS Inspection, page 22-17](#)

How DNS Application Inspection Works

The security appliance tears down the DNS session associated with a DNS query as soon as the DNS reply is forwarded by the security appliance. The security appliance also monitors the message exchange to ensure that the ID of the DNS reply matches the ID of the DNS query.

When DNS inspection is enabled, which is the default, the security appliance performs the following additional tasks:

- Translates the DNS record based on the configuration completed using the **alias**, **static** and **nat** commands (DNS Rewrite). Translation only applies to the A-record in the DNS reply; therefore, DNS Rewrite does not affect reverse lookups, which request the PTR record.



Note DNS Rewrite is not applicable for PAT because multiple PAT rules are applicable for each A-record and the PAT rule to use is ambiguous.

- Enforces the maximum DNS message length (the default is 512 bytes and the maximum length is 65535 bytes). The security appliance performs reassembly as needed to verify that the packet length is less than the maximum length configured. The security appliance drops the packet if it exceeds the maximum length.



Note If you enter the **inspect dns** command without the **maximum-length** option, DNS packet size is not checked

- Enforces a domain-name length of 255 bytes and a label length of 63 bytes.
- Verifies the integrity of the domain-name referred to by the pointer if compression pointers are encountered in the DNS message.
- Checks to see if a compression pointer loop exists.

A single connection is created for multiple DNS sessions, as long as they are between the same two hosts, and the sessions have the same 5-tuple (source/destination IP address, source/destination port, and protocol). DNS identification is tracked by *app_id*, and the idle timer for each *app_id* runs independently.

Because the *app_id* expires independently, a legitimate DNS response can only pass through the security appliance within a limited period of time and there is no resource build-up. However, if you enter the **show conn** command, you will see the idle timer of a DNS connection being reset by a new DNS session. This is due to the nature of the shared DNS connection and is by design.

How DNS Rewrite Works

When DNS inspection is enabled, DNS rewrite provides full support for NAT of DNS messages originating from any interface.

If a client on an inside network requests DNS resolution of an inside address from a DNS server on an outside interface, the DNS A-record is translated correctly. If the DNS inspection engine is disabled, the A-record is not translated.

As long as DNS inspection remains enabled, you can configure DNS rewrite using the **alias**, **static**, or **nat** commands. For details about the configuration required see the [“Configuring DNS Rewrite” section on page 22-12](#).

DNS Rewrite performs two functions:

- Translating a public address (the routable or “mapped” address) in a DNS reply to a private address (the “real” address) when the DNS client is on a private interface.
- Translating a private address to a public address when the DNS client is on the public interface.

In [Figure 22-1](#), the DNS server resides on the external (ISP) network. The real address of the server (192.168.100.1) has been mapped using the **static** command to the ISP-assigned address (209.165.200.5). When a web client on the inside interface attempts to access the web server with the URL `http://server.example.com`, the host running the web client sends a DNS request to the DNS server to resolve the IP address of the web server. The security appliance translates the non-routable source address in the IP header and forwards the request to the ISP network on its outside interface. When the DNS reply is returned, the security appliance applies address translation not only to the destination address, but also to the embedded IP address of the web server, which is contained in the A-record in the DNS reply. As a result, the web client on the inside network gets the correct address for connecting to the web server on the inside network. For the exact NAT and DNS configuration for this example, see [Figure 22-1](#). For configuration instructions for scenarios similar to this one, see the [“Configuring DNS Rewrite with Two NAT Zones” section on page 22-13](#).

Figure 22-1 Translating the Address in a DNS Reply (DNS Rewrite)



DNS rewrite also works if the client making the DNS request is on a DMZ network and the DNS server is on an inside interface. For an illustration and configuration instructions for this scenario, see the “[DNS Rewrite with Three NAT Zones](#)” section on page 22-14.

Configuring DNS Rewrite

You configure DNS rewrite using the **alias**, **static**, or **nat** commands. The **alias** and **static** command can be used interchangeably; however, we recommend using the **static** command for new deployments because it is more precise and unambiguous. Also, DNS rewrite is optional when using the **static** command.

This section describes how to use the **alias** and **static** commands to configure DNS rewrite. It provides configuration procedures for using the **static** command in a simple scenario and in a more complex scenario. Using the **nat** command is similar to using the **static** command except that DNS Rewrite is based on dynamic translation instead of a static mapping.

This section includes the following topics:

- [Using the Static Command for DNS Rewrite, page 22-13](#)
- [Using the Static Command for DNS Rewrite, page 22-13](#)
- [Configuring DNS Rewrite with Two NAT Zones, page 22-13](#)
- [DNS Rewrite with Three NAT Zones, page 22-14](#)
- [Configuring DNS Rewrite with Three NAT Zones, page 22-16](#)

For detailed syntax and additional functions for the **alias**, **nat**, and **static** command, see the appropriate command page in the *Cisco Security Appliance Command Reference*.

Using the Static Command for DNS Rewrite

The **static** command causes addresses on an IP network residing on a specific interface to be translated into addresses on another IP network on a different interface. The syntax for this command is as follows:

```
hostname(config)# static (real_ifc,mapped_ifc) mapped-address real-address dns
```

The following example specifies that the address 192.168.100.10 on the inside interface is translated into 209.165.200.5 on the outside interface:

```
hostname(config)# static (inside,outside) 209.165.200.225 192.168.100.10 dns
```

**Note**

Using the **nat** command is similar to using the **static** command except that DNS Rewrite is based on dynamic translation instead of a static mapping.

Using the Alias Command for DNS Rewrite

The **alias** command causes the security appliance to translate addresses on an IP network residing on any interface into addresses on another IP network connected through a different interface. The syntax for this command is as follows:

```
hostname(config)# alias (interface_name) mapped-address real-address
```

The following example specifies that the real address (192.168.100.10) on any interface except the inside interface will be translated to the mapped address (209.165.200.225) on the inside interface. Notice that the location of 192.168.100.10 is not precisely defined.

```
hostname(config)# alias (inside) 209.165.200.225 192.168.100.10
```

**Note**

If you use the **alias** command to configure DNS Rewrite, proxy ARP will be performed for the mapped address. To prevent this, disable Proxy ARP by entering the **sysopt noproxyarp** command after entering the **alias** command.

Configuring DNS Rewrite with Two NAT Zones

To implement a DNS Rewrite scenario similar to the one shown in [Figure 22-1](#), perform the following steps:

Step 1 Create a static translation for the web server, as follows:

```
hostname(config)# static (real_ifc,mapped_ifc) mapped-address real-address netmask  
255.255.255.255 dns
```

where the arguments are as follows:

- *real_ifc*—The name of the interface connected to the real addresses.
- *mapped_ifc*—The name of the interface where you want the addresses to be mapped.
- *mapped-address*—The translated IP address of the web server.
- *real-address*—The real IP address of the web server.

Step 2 Create an access list that permits traffic to the port that the web server listens to for HTTP requests.

```
hostname(config)# access-list acl-name extended permit tcp any host mapped-address eq port
```

where the arguments are as follows:

acl-name—The name you give the access list.

mapped-address—The translated IP address of the web server.

port—The TCP port that the web server listens to for HTTP requests.

Step 3 Apply the access list created in [Step 2](#) to the mapped interface. To do so, use the **access-group** command, as follows:

```
hostname(config)# access-group acl-name in interface mapped_ifc
```

Step 4 If DNS inspection is disabled or if you want to change the maximum DNS packet length, configure DNS inspection. DNS application inspection is enabled by default with a maximum DNS packet length of 512 bytes. For configuration instructions, see the “[Configuring Application Inspection](#)” section on [page 22-5](#).

Step 5 On the public DNS server, add an A-record for the web server, such as:

```
domain-qualified-hostname. IN A mapped-address
```

where *domain-qualified-hostname* is the hostname with a domain suffix, as in `server.example.com`. The period after the hostname is important. *mapped-address* is the translated IP address of the web server.

The following example configures the security appliance for the scenario shown in [Figure 22-1](#). It assumes DNS inspection is already enabled.

```
hostname(config)# static (inside,outside) 209.165.200.225 192.168.100.1 netmask
255.255.255.255 dns
hostname(config)# access-list 101 permit tcp any host 209.165.200.225 eq www
hostname(config)# access-group 101 in interface outside
```

This configuration requires the following A-record on the DNS server:

```
server.example.com. IN A 209.165.200.225
```

DNS Rewrite with Three NAT Zones

[Figure 22-2](#) provides a more complex scenario to illustrate how DNS inspection allows NAT to operate transparently with a DNS server with minimal configuration. For configuration instructions for scenarios like this one, see the “[Configuring DNS Rewrite with Three NAT Zones](#)” section on [page 22-16](#).

Figure 22-2 *DNS Rewrite with Three NAT Zones*



In [Figure 22-2](#), a web server, `server.example.com`, has the real address `192.168.100.10` on the DMZ interface of the security appliance. A web client with the IP address `10.10.10.25` is on the inside interface and a public DNS server is on the outside interface. The site NAT policies are as follows:

- The outside DNS server holds the authoritative address record for `server.example.com`.
- Hosts on the outside network can contact the web server with the domain name `server.example.com` through the outside DNS server or with the IP address `209.165.200.5`.
- Clients on the inside network can access the web server with the domain name `server.example.com` through the outside DNS server or with the IP address `192.168.100.10`.

When a host or client on any interface accesses the DMZ web server, it queries the public DNS server for the A-record of `server.example.com`. The DNS server returns the A-record showing that `server.example.com` binds to address `209.165.200.5`.

When a web client on the *outside* network attempts to access `http://server.example.com`, the sequence of events is as follows:

1. The host running the web client sends the DNS server a request for the IP address of `server.example.com`.
2. The DNS server responds with the IP address `209.165.200.225` in the reply.
3. The web client sends its HTTP request to `209.165.200.225`.
4. The packet from the outside host reaches the security appliance at the outside interface.
5. The static rule translates the address `209.165.200.225` to `192.168.100.10` and the security appliance directs the packet to the web server on the DMZ.

When a web client on the *inside* network attempts to access `http://server.example.com`, the sequence of events is as follows:

1. The host running the web client sends the DNS server a request for the IP address of `server.example.com`.
2. The DNS server responds with the IP address `209.165.200.225` in the reply.
3. The security appliance receives the DNS reply and submits it to the DNS application inspection engine.
4. The DNS application inspection engine does the following:
 - a. Searches for any NAT rule to undo the translation of the embedded A-record address “[outside]:209.165.200.5”. In this example, it finds the following static configuration:

```
static (dmz,outside) 209.165.200.225 192.168.100.10 dns
```

- b. Uses the static rule to rewrite the A-record as follows because the **dns** option is included:

```
[outside]:209.165.200.225 --> [dmz]:192.168.100.10
```



Note If the **dns** option were not included with the **static** command, DNS Rewrite would not be performed and other processing for the packet continues.

- c. Searches for any NAT to translate the web server address, `[dmz]:192.168.100.10`, when communicating with the inside web client.

No NAT rule is applicable, so application inspection completes.

If a NAT rule (`nat` or `static`) were applicable, the **dns** option must also be specified. If the **dns** option were not specified, the A-record rewrite in step **b** would be reverted and other processing for the packet continues.

- The security appliance sends the HTTP request to server.example.com on the DMZ interface.

Configuring DNS Rewrite with Three NAT Zones

To enable the NAT policies for the scenario in [Figure 22-2](#), perform the following steps:

- Step 1** Create a static translation for the web server on the DMZ network, as follows:

```
hostname(config)# static (dmz,outside) mapped-address real-address dns
```

where the arguments are as follows:

- dmz*—The name of the DMZ interface of the security appliance.
- outside*—The name of the outside interface of the security appliance.
- mapped-address*—The translated IP address of the web server.
- real-address*—The real IP address of the web server.

- Step 2** Create an access list that permits traffic to the port that the web server listens to for HTTP requests.

```
hostname(config)# access-list acl-name extended permit tcp any host mapped-address eq port
```

where the arguments are as follows:

acl-name—The name you give the access list.

mapped-address—The translated IP address of the web server.

port—The TCP port that the web server listens to for HTTP requests.

- Step 3** Apply the access list created in [Step 2](#) to the outside interface. To do so, use the **access-group** command, as follows:

```
hostname(config)# access-group acl-name in interface outside
```

- Step 4** If DNS inspection is disabled or if you want to change the maximum DNS packet length, configure DNS inspection. DNS application inspection is enabled by default with a maximum DNS packet length of 512 bytes. For configuration instructions, see the “[Configuring Application Inspection](#)” section on [page 22-5](#).

- Step 5** On the public DNS server, add an A-record for the web server, such as:

```
domain-qualified-hostname. IN A mapped-address
```

where *domain-qualified-hostname* is the hostname with a domain suffix, as in server.example.com. The period after the hostname is important. *mapped-address* is the translated IP address of the web server.

The following example configures the security appliance for the scenario shown in [Figure 22-2](#). It assumes DNS inspection is already enabled.

```
hostname(config)# static (dmz,outside) 209.165.200.225 192.168.100.10 dns
hostname(config)# access-list 101 permit tcp any host 209.165.200.225 eq www
hostname(config)# access-group 101 in interface outside
```

This configuration requires the following A-record on the DNS server:

```
server.example.com. IN A 209.165.200.225
```

Verifying and Monitoring DNS Inspection

To view information about the current DNS connections, enter the following command:

```
hostname# show conn
```

For connections using a DNS server, the source port of the connection may be replaced by the IP address of DNS server in the show conn command output.

A single connection is created for multiple DNS sessions, as long as they are between the same two hosts, and the sessions have the same 5-tuple (source/destination IP address, source/destination port, and protocol). DNS identification is tracked by app_id, and the idle timer for each app_id runs independently.

Because the app_id expires independently, a legitimate DNS response can only pass through the security appliance within a limited period of time and there is no resource build-up. However, when you enter the **show conn** command, you see the idle timer of a DNS connection being reset by a new DNS session. This is due to the nature of the shared DNS connection and is by design.

To display the statistics for DNS application inspection, enter the **show service-policy** command. The following is sample output from the **show service-policy** command:

```
hostname# show service-policy
Interface outside:
  Service-policy: sample_policy
  Class-map: dns_port
    Inspect: dns maximum-length 1500, packet 0, drop 0, reset-drop 0
```

FTP Inspection

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

- [FTP Inspection Overview, page 22-18](#)
- [Using the strict Option, page 22-18](#)
- [Configuring an FTP Map for Additional Inspection Control, page 22-19](#)
- [Verifying and Monitoring FTP Inspection, page 22-20](#)

FTP Inspection Overview

The FTP application inspection inspects the FTP sessions and performs four tasks:

- Prepares dynamic secondary data connection
- Tracks the FTP command-response sequence
- Generates an audit trail
- Translates the embedded IP address

FTP application inspection prepares secondary channels for FTP data transfer. Ports for these channels are negotiated through PORT or PASV commands. The channels are allocated in response to a file upload, a file download, or a directory listing event.

**Note**

If you disable FTP inspection engines with the **no inspect ftp** command, outbound users can start connections only in passive mode, and all inbound FTP is disabled.

Using the strict Option

Using the **strict** option with the **inspect ftp** command increases the security of protected networks by preventing web browsers from sending embedded commands in FTP requests.



Note

To specify FTP commands that are not permitted to pass through the security appliance, create an FTP map according to the [“Configuring an FTP Map for Additional Inspection Control”](#) section on page 22-19.

After you enable the **strict** option on an interface, FTP inspection enforces the following behavior:

- An FTP command must be acknowledged before the security appliance allows a new command.
- The security appliance drops connections that send embedded commands.
- The 227 and PORT commands are checked to ensure they do not appear in an error string.



Caution

Using the **strict** option may cause the failure of FTP clients that are not strictly compliant with FTP RFCs.

If the **strict** option is enabled, each FTP command and response sequence is tracked for the following anomalous activity:

- Truncated command—Number of commas in the PORT and PASV reply command is checked to see if it is five. If it is not five, then the PORT command is assumed to be truncated and the TCP connection is closed.
- Incorrect command—Checks the FTP command to see if it ends with <CR><LF> characters, as required by the RFC. If it does not, the connection is closed.
- Size of RETR and STOR commands—These are checked against a fixed constant. If the size is greater, then an error message is logged and the connection is closed.
- Command spoofing—The PORT command should always be sent from the client. The TCP connection is denied if a PORT command is sent from the server.
- Reply spoofing—PASV reply command (227) should always be sent from the server. The TCP connection is denied if a PASV reply command is sent from the client. This prevents the security hole when the user executes “227 xxxxx a1, a2, a3, a4, p1, p2.”
- TCP stream editing—The security appliance closes the connection if it detects TCP stream editing.
- Invalid port negotiation—The negotiated dynamic port value is checked to see if it is less than 1024. As port numbers in the range from 1 to 1024 are reserved for well-known connections, if the negotiated port falls in this range, then the TCP connection is freed.
- Command pipelining—The number of characters present after the port numbers in the PORT and PASV reply command is cross checked with a constant value of 8. If it is more than 8, then the TCP connection is closed.
- The security appliance replaces the FTP server response to the SYST command with a series of Xs. to prevent the server from revealing its system type to FTP clients. To override this default behavior, use the **no mask-syst-reply** command in the FTP map.

Configuring an FTP Map for Additional Inspection Control

If you want FTP inspection to allow FTP servers to reveal their system type to FTP clients, and limit the allowed FTP commands, then create and configure an FTP map. You can then apply the FTP map when you enable FTP inspection according to the “[Configuring Application Inspection](#)” section on page 22-5.

To create an FTP map, perform the following steps:

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

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

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

Step 2 To add a text description to this FTP map, enter the following command:

```
hostname(config-ftp-map)# description text
```

Step 3 If you want to allow FTP servers from revealing their system type to FTP clients in responses to SYST messages, use the **no** form of the **mask-syst-reply** command, as follows:

```
hostname(config-ftp-map)# no mask-syst-reply
hostname(config-ftp-map)#
```



Note By default, when FTP inspection is enabled, responses to SYST messages are masked. If you disable SYST response masking, you can reenable it with the **mask-syst-response** command.

Step 4 If you want to disallow specific FTP commands, use the following command:

```
hostname(config-ftp-map)# request-command deny ftp_command [ftp_command...]
hostname(config-ftp-map)#
```

where *ftp_command* with one or more FTP commands that you want to restrict. See [Table 22-3](#) for a list of the FTP commands that you can restrict.

Table 22-3 FTP Map request-command deny Options

request-command deny Option	Purpose
appe	Disallows the command that appends to a file.
cdup	Disallows the command that changes to the parent directory of the current working directory.
dele	Disallows the command that deletes a file on the server.
get	Disallows the client command for retrieving a file from the server.
help	Disallows the command that provides help information.
mkd	Disallows the command that makes a directory on the server.
put	Disallows the client command for sending a file to the server.
rmd	Disallows the command that deletes a directory on the server.
rnfr	Disallows the command that specifies rename-from filename.
rnto	Disallows the command that specifies rename-to filename.
site	Disallows the command that are specific to the server system. Usually used for remote administration.
stou	Disallows the command that stores a file using a unique file name.

The following example shows how to define an FTP map:

```
hostname(config-cmap)# ftp-map sample_map
hostname(config-ftp-map)# request-command deny put stou appe
```

Verifying and Monitoring FTP Inspection

FTP application inspection generates the following log messages:

- An Audit record 302002 is generated for each file that is retrieved or uploaded.
- The FTP command is checked to see if it is RETR or STOR and the retrieve and store commands are logged.
- The username is obtained by looking up a table providing the IP address.
- The username, source IP address, destination IP address, NAT address, and the file operation are logged.
- Audit record 201005 is generated if the secondary dynamic channel preparation failed due to memory shortage.

In conjunction with NAT, the FTP application inspection translates the IP address within the application payload. This is described in detail in RFC 959.

GTP Inspection

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

- [GTP Inspection Overview, page 22-21](#)
- [Configuring a GTP Map for Additional Inspection Control, page 22-22](#)
- [Verifying and Monitoring GTP Inspection, page 22-26](#)

**Note**

GTP inspection requires a special license. If you enter GTP-related commands on a security appliance without the required license, the 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 22-3](#)).

Figure 22-3 *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 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 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 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 Inspection” section on page 22-5](#).

Step 1 To create a GTP map, enter the following command:

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

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

Step 2 To add a text description to this GTP map, enter the following command:

```
hostname(config-gtp-map)# description text
```

Step 3 To drop specified GTP messages, enter the following command:

```
hostname(config-gtp-map)# drop {apn access_point_name | message message_id | version
version}
```

where the **apn** keyword drops GTP messages with the specified *access_point_name*.

The **message** keyword drops specific GTP messages identified by the *message_id*. The *message_id* is an alphanumeric identifier between 1 and 255.

The **version** keyword drops GTP messages with the specified *version*. Use 0 to identify Version 0 and 1 to identify Version 1. Version 0 of GTP uses port 2123, while Version 1 uses port 3386.

By default, all messages with valid message IDs, APNs, and version are inspected. Any APN is allowed.

Step 4 To identify the mobile country code and the mobile network code for IMSI prefix filtering, enter the following command:

```
hostname(config-gtp-map)# mcc country_code mnc network_code
```

where the **mcc** *country_code* argument is a non-zero, three-digit value identifying the mobile country code. One or two-digit entries will be prepended by 0 to create a three-digit value.

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 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*.

- Step 5** To filter GTP packets that do not meet the configured maximum and minimum length, enter the following command:

```
hostname(config-gtp-map)# message-length min min_bytes max max_bytes
```

where the **min min_bytes** argument specifies the minimum number of bytes allowed in the UDP payload, from 1 to 65536.

The **max max_bytes** argument specifies the maximum number of bytes allowed in the UDP payload, from 1 to 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 allow invalid GTP packets or packets that otherwise would fail parsing and be dropped, enter the following command:

```
hostname(config-gtp-map)# permit errors
```

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

- Step 7** To enable support for GSN pooling, use the **permit response** command.

If the security appliance performs GTP inspection, by default the 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 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 security appliance permits the response.

1. To create an object to represent the pool of load-balancing GSNs, perform the following steps:
 - a. Use the **object-group network** 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)#
```

- b. 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)#
```

2. 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)#
```

3. To allow GTP responses from any GSN in the network object representing the GSN pool, defined in [Step 1](#), to the network object representing the SGSN, defined in [Step 2](#), 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
```

- Step 8** 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.

Step 9 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 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 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.

Step 10 To specify the maximum number of GTP tunnels allowed to be active on the 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 define a GTP map:

```
hostname(config-cmap)# gtp-map sample_map
hostname(config-gtp-map)# request-queue 300
hostname(config-gtp-map)# permit mcc 111 mnc 222
hostname(config-gtp-map)# message-length min 20 max 300
hostname(config-gtp-map)# drop message 20
hostname(config-gtp-map)# tunnel-limit 10000
```

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 Security Appliance 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.

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
  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
```

H.323 Inspection

This section describes the H.323 application inspection. This section includes the following topics:

- [H.323 Inspection Overview, page 22-27](#)
- [How H.323 Works, page 22-27](#)
- [Limitations and Restrictions, page 22-28](#)
- [Configuring H.323 and H.225 Timeout Values, page 22-28](#)
- [Verifying and Monitoring H.323 Inspection, page 22-29](#)

H.323 Inspection Overview

H.323 inspection provides support for H.323 compliant applications such as Cisco CallManager and VocalTec Gatekeeper. H.323 is a suite of protocols defined by the International Telecommunication Union for multimedia conferences over LANs. The security appliance supports H.323 through Version 4, including H.323 v3 feature Multiple Calls on One Call Signaling Channel.

With H323 inspection enabled, the security appliance supports multiple calls on the same call signaling channel, a feature introduced with H.323 Version 3. This feature reduces call setup time and reduces the use of ports on the security appliance.

The two major functions of H.323 inspection are as follows:

- NAT the necessary embedded IPv4 addresses in the H.225 and H.245 messages. Because H.323 messages are encoded in PER encoding format, the security appliance uses an ASN.1 decoder to decode the H.323 messages.
- Dynamically allocate the negotiated H.245 and RTP/RTCP connections.

How H.323 Works

The H.323 collection of protocols collectively may use up to two TCP connection and four to six UDP connections. FastConnect uses only one TCP connection, and RAS uses a single UDP connection for registration, admissions, and status.

An H.323 client may initially establish a TCP connection to an H.323 server using TCP port 1720 to request Q.931 call setup. As part of the call setup process, the H.323 terminal supplies a port number to the client to use for an H.245 TCP connection. In environments where H.323 gatekeeper is in use, the initial packet is transmitted using UDP.

H.323 inspection monitors the Q.931 TCP connection to determine the H.245 port number. If the H.323 terminals are not using FastConnect, the security appliance dynamically allocates the H.245 connection based on the inspection of the H.225 messages.

Within each H.245 message, the H.323 endpoints exchange port numbers that are used for subsequent UDP data streams. H.323 inspection inspects the H.245 messages to identify these ports and dynamically creates connections for the media exchange. RTP uses the negotiated port number, while RTCP uses the next higher port number.

The H.323 control channel handles H.225 and H.245 and H.323 RAS. H.323 inspection uses the following ports.

- 1718—Gate Keeper Discovery UDP port
- 1719—RAS UDP port
- 1720—TCP Control Port

You must permit traffic for the well-known H.323 port 1720 for the H.225 call signaling; however, the H.245 signaling ports are negotiated between the endpoints in the H.225 signaling. When an H.323 gatekeeper is used, the security appliance opens an H.225 connection based on inspection of the ACF message.

After inspecting the H.225 messages, the security appliance opens the H.245 channel and then inspects traffic sent over the H.245 channel as well. All H.245 messages passing through the security appliance undergo H.245 application inspection, which translates embedded IP addresses and opens the media channels negotiated in H.245 messages.

The H.323 ITU standard requires that a TPKT header, defining the length of the message, precede the H.225 and H.245, before being passed on to the reliable connection. Because the TPKT header does not necessarily need to be sent in the same TCP packet as H.225 and H.245 messages, the security appliance must remember the TPKT length to process and decode the messages properly. For each connection, the security appliance keeps a record that contains the TPKT length for the next expected message.

If the security appliance needs to perform NAT on IP addresses in messages, it changes the checksum, the UUIE length, and the TPKT, if it is included in the TCP packet with the H.225 message. If the TPKT is sent in a separate TCP packet, the security appliance proxy ACKs that TPKT and appends a new TPKT to the H.245 message with the new length.

**Note**

The security appliance does not support TCP options in the Proxy ACK for the TPKT.

Each UDP connection with a packet going through H.323 inspection is marked as an H.323 connection and times out with the H.323 timeout as configured with the **timeout** command.

Limitations and Restrictions

The following are some of the known issues and limitations when using H.323 application inspection:

- Static PAT may not properly translate IP addresses embedded in optional fields within H.323 messages. If you experience this kind of problem, do not use static PAT with H.323.
- H.323 application inspection is not supported with NAT between same-security-level interfaces.
- When a NetMeeting client registers with an H.323 gatekeeper and tries to call an H.323 gateway that is also registered with the H.323 gatekeeper, the connection is established but no voice is heard in either direction. This problem is unrelated to the security appliance.
- If you configure a network static address where the network static address is the same as a third-party netmask and address, then any outbound H.323 connection fails.

Configuring H.323 and H.225 Timeout Values

To configure the idle time after which an H.225 signalling connection is closed, use the **timeout h225** command. The default for H.225 timeout is one hour.

To configure the idle time after which an H.323 control connection is closed, use the **timeout h323** command. The default is five minutes.

Verifying and Monitoring H.323 Inspection

This section describes how to display information about H.323 sessions. This section includes the following topics:

- [Monitoring H.225 Sessions, page 22-29](#)
- [Monitoring H.245 Sessions, page 22-29](#)
- [Monitoring H.323 RAS Sessions, page 22-30](#)

Monitoring H.225 Sessions

The **show h225** command displays information for H.225 sessions established across the security appliance. Along with the **debug h323 h225 event**, **debug h323 h245 event**, and **show local-host** commands, this command is used for troubleshooting H.323 inspection engine issues.

Before entering the **show h225**, **show h245**, or **show h323-ras** commands, we recommend that you configure the **pager** command. If there are a lot of session records and the **pager** command is not configured, it may take a while for the **show** command output to reach its end. If there is an abnormally large number of connections, check that the sessions are timing out based on the default timeout values or the values set by you. If they are not, then there is a problem that needs to be investigated.

The following is sample output from the **show h225** command:

```
hostname# show h225
Total H.323 Calls: 1
1 Concurrent Call(s) for
  Local: 10.130.56.3/1040 Foreign: 172.30.254.203/1720
  1. CRV 9861
  Local: 10.130.56.3/1040 Foreign: 172.30.254.203/1720
0 Concurrent Call(s) for
  Local: 10.130.56.4/1050 Foreign: 172.30.254.205/1720
```

This output indicates that there is currently 1 active H.323 call going through the security appliance between the local endpoint 10.130.56.3 and foreign host 172.30.254.203, and for these particular endpoints, there is 1 concurrent call between them, with a CRV for that call of 9861.

For the local endpoint 10.130.56.4 and foreign host 172.30.254.205, there are 0 concurrent calls. This means that there is no active call between the endpoints even though the H.225 session still exists. This could happen if, at the time of the **show h225** command, the call has already ended but the H.225 session has not yet been deleted. Alternately, it could mean that the two endpoints still have a TCP connection opened between them because they set “maintainConnection” to TRUE, so the session is kept open until they set it to FALSE again, or until the session times out based on the H.225 timeout value in your configuration.

Monitoring H.245 Sessions

The **show h245** command displays information for H.245 sessions established across the security appliance by endpoints using slow start. Slow start is when the two endpoints of a call open another TCP control channel for H.245. Fast start is where the H.245 messages are exchanged as part of the H.225 messages on the H.225 control channel.) Along with the **debug h323 h245 event**, **debug h323 h225 event**, and **show local-host** commands, this command is used for troubleshooting H.323 inspection engine issues.

The following is sample output from the **show h245** command:

```
hostname# show h245
Total: 1
LOCAL          TPKT    FOREIGN          TPKT
1 10.130.56.3/1041 0 172.30.254.203/1245 0
MEDIA: LCN 258 Foreign 172.30.254.203 RTP 49608 RTCP 49609
      Local 10.130.56.3 RTP 49608 RTCP 49609
MEDIA: LCN 259 Foreign 172.30.254.203 RTP 49606 RTCP 49607
      Local 10.130.56.3 RTP 49606 RTCP 49607
```

There is currently one H.245 control session active across the security appliance. The local endpoint is 10.130.56.3, and we are expecting the next packet from this endpoint to have a TPKT header because the TPKT value is 0. The TKTP header is a 4-byte header preceding each H.225/H.245 message. It gives the length of the message, including the 4-byte header. The foreign host endpoint is 172.30.254.203, and we are expecting the next packet from this endpoint to have a TPKT header because the TPKT value is 0.

The media negotiated between these endpoints have an LCN of 258 with the foreign RTP IP address/port pair of 172.30.254.203/49608 and an RTCP IP address/port of 172.30.254.203/49609 with a local RTP IP address/port pair of 10.130.56.3/49608 and an RTCP port of 49609.

The second LCN of 259 has a foreign RTP IP address/port pair of 172.30.254.203/49606 and an RTCP IP address/port pair of 172.30.254.203/49607 with a local RTP IP address/port pair of 10.130.56.3/49606 and RTCP port of 49607.

Monitoring H.323 RAS Sessions

The **show h323-ras** command displays information for H.323 RAS sessions established across the security appliance between a gatekeeper and its H.323 endpoint. Along with the **debug h323 ras event** and **show local-host** commands, this command is used for troubleshooting H.323 RAS inspection engine issues.

The **show h323-ras** command displays connection information for troubleshooting H.323 inspection engine issues. The following is sample output from the **show h323-ras** command:

```
hostname# show h323-ras
Total: 1
      GK                Caller
      172.30.254.214  10.130.56.14
```

This output shows that there is one active registration between the gatekeeper 172.30.254.214 and its client 10.130.56.14.

HTTP Inspection

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

- [HTTP Inspection Overview, page 22-30](#)
- [Configuring an HTTP Map for Additional Inspection Control, page 22-31](#)

HTTP Inspection Overview

Use the HTTP inspection engine to protect against specific attacks and other threats that may be associated with HTTP traffic. HTTP inspection performs several functions:

- Enhanced HTTP inspection
- URL screening through N2H2 or Websense
- Java and ActiveX filtering

The latter two features are configured in conjunction with the **filter** command. For more information about filtering, see [Chapter 17, “Applying Filtering Services.”](#)



Note

The **no inspect http** command also disables the **filter url** command.

The enhanced HTTP inspection feature, which is also known as an application firewall and is available when you configure an HTTP map (see “[Configuring an HTTP Map for Additional Inspection Control](#)”), can help prevent attackers from using HTTP messages for circumventing network security policy. It verifies the following for all HTTP messages:

- Conformance to RFC 2616
- Use of RFC-defined methods only.
- Compliance with the additional criteria.

Configuring an HTTP Map for Additional Inspection Control

To specify actions when a message violates a parameter, create an HTTP map. You can then apply the HTTP map when you enable HTTP inspection according to the “[Configuring Application Inspection](#)” section on page 22-5.



Note

When you enable HTTP inspection with an HTTP map, strict HTTP inspection with the action reset and log is enabled by default. You can change the actions performed in response to inspection failure, but you cannot disable strict inspection as long as the HTTP map remains enabled.

To create an HTTP map, perform the following steps:

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

```
hostname(config-cmap) # http-map map_name
hostname(config-http-map) #
```

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

Step 2 To restrict HTTP traffic based on the length of the HTTP message body, enter the following command:

```
hostname(config-http-map) # content-length {[min bytes] [max bytes]} action {allow | reset
| drop} [log]
```

where the **min bytes** argument specifies the minimum content length allowed, from 1 to 65535

The **max bytes** argument specifies the maximum content length allowed, from 1 to 50000000.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Step 3 To restrict HTTP traffic based on the content type of the HTTP message, enter the following command:

```
hostname(config-http-map) # content-type-verification [match-req-rsp] action {allow | reset
| drop} [log]
```

where the **match-req-rsp** keyword verifies that the content-type field in the HTTP response matches the accept field in the corresponding HTTP request message.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

This command enables the following checks:

- Verifies that the value of the header content-type is in the internal list of supported content types.

For a list of content types, see the *Cisco Security Appliance Command Reference*.

- Verifies that the header content-type matches the actual content in the data or entity body portion of the message.
- The `match-req-rsp` keyword enables an additional check that verifies the content-type field in the HTTP response matches the `accept` field in the corresponding HTTP request message.

Step 4 To restrict HTTP traffic based on the HTTP header length, enter the following command:

```
hostname(config-http-map)# max-header-length {[request bytes] [response bytes]} action
{allow | reset | drop} [log]
```

where the **request bytes** argument sets the maximum header length for request messages, from 1 to 65535.

The **response bytes** argument sets the maximum header length for response messages, from 1 to 65535.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Step 5 To restrict HTTP traffic based on the length of the URI in the HTTP request message, enter the following command:

```
hostname(config-http-map)# max-uri-length bytes action {allow | reset | drop} [log]
```

where the **bytes** argument sets the maximum URI length, from 1 to 65535.

Step 6 To restrict HTTP traffic by specifying a restricted application category, enter the following command:

```
hostname(config-http-map)# port-misuse {im | p2p | tunneling | default} action {allow |
reset | drop} [log]
```

where the **im** keyword restricts traffic in the instant messaging application category. The applications checked for are Yahoo Messenger, AIM, and MSN IM.

The **p2p** keyword restricts traffic in the peer-to-peer application category. The Kazaa application is checked.

The **tunneling** keyword restricts traffic in the tunneling application category. The applications checked for are: HTTPPort/HTTHost, GNU Httptunnel, GotoMyPC, Firethru, and Http-tunnel.com Client.

Specify **default** to set the default action taken by the security appliance when the traffic does *not* match the application types listed. The default action is to allow connections without logging.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Enter the **port-misuse** command once for each setting you wish to apply.

Step 7 To restrict HTTP traffic based on the HTTP request method, enter the following command:

```
hostname(config-http-map)# request-method {ext {ext_methods | default} | rfc {rfc_methods
| default}} action {allow | reset | drop} [log]
```

where the **ext** keyword specifies the extension methods. See the CLI help for a list of extension methods. Specify **default** to set the default action taken by the security appliance when the traffic does *not* match the extension methods listed. The default action is to allow connections without logging.

The **rfc** keyword specifies the RFC 2616 supported methods. See the CLI help for a list of RFC methods. Specify **default** to set the default action taken by the security appliance when the traffic does *not* match the RFC methods listed. The default action is to allow connections without logging.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Enter the **request-method** command once for each setting you wish to apply.

Step 8 To allow forwarding of non-compliant HTTP traffic, enter the following command:

```
hostname(config-http-map)# strict-http action {allow | reset | drop} [log]
```

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Although strict HTTP inspection cannot be disabled, the **strict-http action allow** command causes the security appliance to allow forwarding of non-compliant HTTP traffic. This command overrides the default behavior, which is to deny forwarding of non-compliant HTTP traffic.

Step 9 To restrict HTTP traffic by specifying a transfer encoding type, enter the following command:

```
hostname(config-http-map)# transfer-encoding type {chunked | compress | deflate | gzip | identity | default} action {allow | reset | drop} [log]
```

where the **chunked** keyword identifies the transfer encoding type in which the message body is transferred as a series of chunks.

The **compress** keyword identifies the transfer encoding type in which the message body is transferred using UNIX file compression.

The **deflate** keyword identifies the transfer encoding type in which the message body is transferred using zlib format (RFC 1950) and deflate compression (RFC 1951).

The **gzip** keyword identifies the transfer encoding type in which the message body is transferred using GNU zip (RFC 1952).

The **identity** keyword identifies connections in which the message body is no transfer encoding is performed.

Specify **default** to set the default action taken by the security appliance when the traffic does *not* match the transfer encoding types listed. The default action is to allow connections without logging.

The **action** keyword specifies the action taken when a message fails this inspection: **allow**, **reset**, or **drop**.

Specify the **log** keyword to generate a system message when a message fails this inspection.

Enter the **transfer-encoding** command once for each setting you wish to apply.

The following example shows how to define an HTTP map:

```
hostname(config-cmap)# http-map sample_map
hostname(config-http-map)# content-length min 100 max 2000 action reset log
hostname(config-http-map)# content-type-verification match-req-rsp action reset log
hostname(config-http-map)# max-header-length request 100 action reset log
hostname(config-http-map)# max-uri-length 100 action reset log
hostname(config-http-map)# policy-map sample_policy
```

ICMP Inspection

The ICMP inspection engine allows ICMP traffic to have a “session” so it can be inspected like TCP and UDP traffic. Without the ICMP inspection engine, we recommend that you do not allow ICMP through the security appliance in an access list. Without stateful inspection, ICMP can be used to attack your network. The ICMP inspection engine ensures that there is only one response for each request, and that the sequence number is correct.

ICMP Error Inspection

When this feature is enabled, the security appliance creates translation sessions for intermediate hops that send ICMP error messages, based on the NAT configuration. The security appliance overwrites the packet with the translated IP addresses.

When disabled, the security appliance does not create translation sessions for intermediate nodes that generate ICMP error messages. ICMP error messages generated by the intermediate nodes between the inside host and the security appliance reach the outside host without consuming any additional NAT resource. This is undesirable when an outside host uses the traceroute command to trace the hops to the destination on the inside of the security appliance. When the security appliance does not translate the intermediate hops, all the intermediate hops appear with the mapped destination IP address.

The ICMP payload is scanned to retrieve the five-tuple from the original packet. Using the retrieved five-tuple, a lookup is performed to determine the original address of the client. The ICMP error inspection engine makes the following changes to the ICMP packet:

- In the IP Header, the mapped IP is changed to the real IP (Destination Address) and the IP checksum is modified.
- In the ICMP Header, the ICMP checksum is modified due to the changes in the ICMP packet.
- In the Payload, the following changes are made:
 - Original packet mapped IP is changed to the real IP
 - Original packet mapped port is changed to the real Port
 - Original packet IP checksum is recalculated

ILS Inspection

The 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.

The security appliance supports NAT for ILS, which is used to register and locate endpoints in the ILS or SiteServer Directory. PAT cannot be supported because only IP addresses are stored by an LDAP database.

For search responses, when the LDAP server is located outside, NAT should be considered to allow internal peers to communicate locally while registered to external LDAP servers. For such search responses, xlates are searched first, and then DNAT entries to obtain the correct address. If both of these searches fail, then the address is not changed. For sites using NAT 0 (no NAT) and not expecting DNAT interaction, we recommend that the inspection engine be turned off to provide better performance.

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

Because ILS traffic 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 **timeout** command.

ILS/LDAP follows a client/server model with sessions handled over a single TCP connection. Depending on the client's actions, several of these sessions may be created.

During connection negotiation time, a BIND PDU is sent from the client to the server. Once a successful BIND RESPONSE from the server is received, other operational messages may be exchanged (such as ADD, DEL, SEARCH, or MODIFY) to perform operations on the ILS Directory. The ADD REQUEST and SEARCH RESPONSE PDUs may contain IP addresses of NetMeeting peers, used by H.323 (SETUP and CONNECT messages) to establish the NetMeeting sessions. Microsoft NetMeeting v2.X and v3.X provides ILS support.

The ILS inspection performs the following operations:

- Decodes the LDAP REQUEST/RESPONSE PDUs using the BER decode functions
- Parses the LDAP packet
- Extracts IP addresses
- Translates IP addresses as necessary
- Encodes the PDU with translated addresses using BER encode functions
- Copies the newly encoded PDU back to the TCP packet
- Performs incremental TCP checksum and sequence number adjustment

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

**Note**

Because H225 call signalling traffic only occurs on the secondary UDP channel, the TCP connection is disconnected after the interval specified by the TCP **timeout** command. By default, this interval is set at 60 minutes.

MGCP Inspection

This section describes MGCP application inspection. This section includes the following topics:

- [MGCP Inspection Overview, page 22-36](#)
- [Configuring an MGCP Map for Additional Inspection Control, page 22-37](#)
- [Configuring MGCP Timeout Values, page 22-38](#)
- [Verifying and Monitoring MGCP Inspection, page 22-39](#)

MGCP Inspection Overview

MGCP is a master/slave protocol used to control media gateways from external call control elements called media gateway controllers or call agents. A media gateway is typically a network element that provides conversion between the audio signals carried on telephone circuits and data packets carried over the Internet or over other packet networks. Using NAT and PAT with MGCP lets you support a large number of devices on an internal network with a limited set of external (global) addresses. Examples of media gateways are:

- Trunking gateways, that interface between the telephone network and a Voice over IP network. Such gateways typically manage a large number of digital circuits.
- Residential gateways, that provide a traditional analog (RJ11) interface to a Voice over IP network. Examples of residential gateways include cable modem/cable set-top boxes, xDSL devices, broad-band wireless devices.
- Business gateways, that provide a traditional digital PBX interface or an integrated soft PBX interface to a Voice over IP network.

MGCP messages are transmitted over UDP. A response is sent back to the source address (IP address and UDP port number) of the command, but the response may not arrive from the same address as the command was sent to. This can happen when multiple call agents are being used in a failover configuration and the call agent that received the command has passed control to a backup call agent, which then sends the response. [Figure 22-4](#) illustrates how NAT can be used with MGCP.

Figure 22-4 Using NAT with MGCP



MGCP endpoints are physical or virtual sources and destinations for data. Media gateways contain endpoints on which the call agent can create, modify and delete connections to establish and control media sessions with other multimedia endpoints. Also, the call agent can instruct the endpoints to detect certain events and generate signals. The endpoints automatically communicate changes in service state to the call agent.

MGCP transactions are composed of a command and a mandatory response. There are eight types of commands:

- CreateConnection
- ModifyConnection
- DeleteConnection
- NotificationRequest
- Notify
- AuditEndpoint
- AuditConnection
- RestartInProgress

The first four commands are sent by the call agent to the gateway. The Notify command is sent by the gateway to the call agent. The gateway may also send a DeleteConnection. The registration of the MGCP gateway with the call agent is achieved by the RestartInProgress command. The AuditEndpoint and the AuditConnection commands are sent by the call agent to the gateway.

All commands are composed of a Command header, optionally followed by a session description. All responses are composed of a Response header, optionally followed by a session description.

- The port on which the gateway receives commands from the call agent. Gateways usually listen to UDP port 2427.
- The port on which the call agent receives commands from the gateway. Call agents usually listen to UDP port 2727.



Note

MGCP inspection does not support the use of different IP addresses for MGCP signaling and RTP data. A common and recommended practice is to send RTP data from a resilient IP address, such as a loopback or virtual IP address; however, the security appliance requires the RTP data to come from the same address as MGCP signalling.

Configuring an MGCP Map for Additional Inspection Control

If the network has multiple call agents and gateways for which the security appliance has to open pinholes, create an MGCP map. You can then apply the MGCP map when you enable MGCP inspection according to the [“Configuring Application Inspection” section on page 22-5](#)

To create an MGCP map, perform the following steps:

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

```
hostname(config-cmap) # mgcp-map map_name
hostname(config-mgcp-map) #
```

where *map_name* is the name of the MGCP map. The system enters MGCP map configuration mode and the CLI prompt changes accordingly.

Step 2 To configure the call agents, enter the following command for each call agent:

```
hostname(config-mgcp-map) # call-agent ip_address group_id
```

Use the **call-agent** command to specify a group of call agents that can manage one or more gateways. The call agent group information is used to open connections for the call agents in the group (other than the one a gateway sends a command to) so that any of the call agents can send the response. Call agents with the same *group_id* belong to the same group. A call agent may belong to more than one group. The *group_id* option is a number from 0 to 4294967295. The *ip_address* option specifies the IP address of the call agent.

**Note**

MGCP call agents send AUEP messages to determine if MGCP end points are present. This establishes a flow through the security appliance and allows MGCP end points to register with the call agent.

Step 3 To configure the gateways, enter the following command for each gateway:

```
hostname(config-mgcp-map)# gateway ip_address group_id
```

Use the **gateway** command to specify which group of call agents are managing a particular gateway. The IP address of the gateway is specified with the *ip_address* option. The *group_id* option is a number from 0 to 4294967295 that must correspond with the *group_id* of the call agents that are managing the gateway. A gateway may only belong to one group.

Step 4 If you want to change the maximum number of commands allowed in the MGCP command queue, enter the following command:

```
hostname(config-mgcp-map)# command-queue command_limit
```

The following example shows how to define an MGCP map:

```
hostname(config-cmap)# mgcp-map sample_map
hostname(config-mgcp-map)# call-agent 10.10.11.5 101
hostname(config-mgcp-map)# call-agent 10.10.11.6 101
hostname(config-mgcp-map)# call-agent 10.10.11.7 102
hostname(config-mgcp-map)# call-agent 10.10.11.8 102
hostname(config-mgcp-map)# gateway 10.10.10.115 101
hostname(config-mgcp-map)# gateway 10.10.10.116 102
hostname(config-mgcp-map)# gateway 10.10.10.117 102
hostname(config-mgcp-map)# command-queue 150
```

Configuring MGCP Timeout Values

The **timeout mgcp command** lets you set the interval for inactivity after which an MGCP media connection is closed. The default is 5 minutes.

The **timeout mgcp-pat** command lets you set the timeout for PAT xlates. Because MGCP does not have a keepalive mechanism, if you use non-Cisco MGCP gateways (call agents), the PAT xlates are torn down after the default timeout interval, which is 30 seconds.

Verifying and Monitoring MGCP Inspection

The **show mgcp commands** command lists the number of MGCP commands in the command queue. The **show mgcp sessions** command lists the number of existing MGCP sessions. The **detail** option includes additional information about each command (or session) in the output. The following is sample output from the **show mgcp commands** command:

```
hostname# show mgcp commands
1 in use, 1 most used, 200 maximum allowed
CRCX, gateway IP: host-pc-2, transaction ID: 2052, idle: 0:00:07
```

The following is sample output from the **show mgcp detail** command.

```
hostname# show mgcp commands detail
1 in use, 1 most used, 200 maximum allowed
CRCX, idle: 0:00:10
    Gateway IP      host-pc-2
    Transaction ID  2052
    Endpoint name   aaln/1
    Call ID         9876543210abcdef
    Connection ID
    Media IP        192.168.5.7
    Media port      6058
```

The following is sample output from the **show mgcp sessions** command.

```
hostname# show mgcp sessions
1 in use, 1 most used
Gateway IP host-pc-2, connection ID 6789af54c9, active 0:00:11
```

The following is sample output from the **show mgcp sessions detail** command.

```
hostname# show mgcp sessions detail
1 in use, 1 most used
Session active 0:00:14
    Gateway IP      host-pc-2
    Call ID         9876543210abcdef
    Connection ID   6789af54c9
    Endpoint name   aaln/1
    Media lcl port  6166
    Media rmt IP    192.168.5.7
    Media rmt port  6058
```

NetBIOS Inspection

NetBIOS inspection is enabled by default. The NetBios inspection engine translates IP addresses in the NetBios name service (NBNS) packets according to the security appliance NAT configuration.

PPTP Inspection

PPTP is a protocol for tunneling PPP traffic. A PPTP session is composed of one TCP channel and usually two PPTP GRE tunnels. The TCP channel is the control channel used for negotiating and managing the PPTP GRE tunnels. The GRE tunnels carries PPP sessions between the two hosts.

When enabled, PPTP application inspection inspects PPTP protocol packets and dynamically creates the GRE connections and xlates necessary to permit PPTP traffic. Only Version 1, as defined in RFC 2637, is supported.

PAT is only performed for the modified version of GRE [RFC 2637] when negotiated over the PPTP TCP control channel. Port Address Translation is *not* performed for the unmodified version of GRE [RFC 1701, RFC 1702].

Specifically, the security appliance inspects the PPTP version announcements and the outgoing call request/response sequence. Only PPTP Version 1, as defined in RFC 2637, is inspected. Further inspection on the TCP control channel is disabled if the version announced by either side is not Version 1. In addition, the outgoing-call request and reply sequence are tracked. Connections and xlates are dynamic allocated as necessary to permit subsequent secondary GRE data traffic.

The PPTP inspection engine must be enabled for PPTP traffic to be translated by PAT. Additionally, PAT is only performed for a modified version of GRE (RFC2637) and only if it is negotiated over the PPTP TCP control channel. PAT is not performed for the unmodified version of GRE (RFC 1701 and RFC 1702).

As described in RFC 2637, the PPTP protocol is mainly used for the tunneling of PPP sessions initiated from a modem bank PAC (PPTP Access Concentrator) to the headend PNS (PPTP Network Server). When used this way, the PAC is the remote client and the PNS is the server.

However, when used for VPN by Windows, the interaction is inverted. The PNS is a remote single-user PC that initiates connection to the head-end PAC to gain access to a central network.

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.

RTSP Inspection

This section describes RTSP application inspection. This section includes the following topics:

- [RTSP Inspection Overview, page 22-40](#)
- [Using RealPlayer, page 22-41](#)
- [Restrictions and Limitations, page 22-41](#)

RTSP Inspection Overview

The RTSP inspection engine lets the security appliance pass RTSP packets. RTSP is used by RealAudio, RealNetworks, Apple QuickTime 4, RealPlayer, and Cisco IP/TV connections.



Note

For Cisco IP/TV, use RTSP TCP port 554 and TCP 8554.

RTSP applications use the well-known port 554 with TCP (rarely UDP) as a control channel. The security appliance only supports TCP, in conformity with RFC 2326. This TCP control channel is used to negotiate the data channels that is used to transmit audio/video traffic, depending on the transport mode that is configured on the client.

The supported RDT transports are: rtp/avp, rtp/avp/udp, x-real-rdt, x-real-rdt/udp, and x-pn-tng/udp.

The security appliance parses Setup response messages with a status code of 200. If the response message is travelling inbound, the server is outside relative to the security appliance and dynamic channels need to be opened for connections coming inbound from the server. If the response message is outbound, then the security appliance does not need to open dynamic channels.

Because RFC 2326 does not require that the client and server ports must be in the SETUP response message, the security appliance keeps state and remembers the client ports in the SETUP message. QuickTime places the client ports in the SETUP message and then the server responds with only the server ports.

RTSP inspection does not support PAT or dual-NAT. Also, the security appliance cannot recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.

Using RealPlayer

When using RealPlayer, it is important to properly configure transport mode. For the security appliance, add an **access-list** command from the server to the client or vice versa. For RealPlayer, change transport mode by clicking **Options>Preferences>Transport>RTSP Settings**.

If using TCP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use TCP for all content** check boxes. On the security appliance, there is no need to configure the inspection engine.

If using UDP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use UDP for static content** check boxes, and for live content not available via Multicast. On the security appliance, add an **inspect rtsp port** command.

Restrictions and Limitations

The following restrictions apply to the **inspect rtsp** command.

- The security appliance does not support multicast RTSP or RTSP messages over UDP.
- PAT is not supported.
- The security appliance does not have the ability to recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.
- The security appliance cannot perform NAT on RTSP messages because the embedded IP addresses are contained in the SDP files as part of HTTP or RTSP messages. Packets could be fragmented and security appliance cannot perform NAT on fragmented packets.
- With Cisco IP/TV, the number of translates the security appliance performs on the SDP part of the message is proportional to the number of program listings in the Content Manager (each program listing can have at least six embedded IP addresses).
- You can configure NAT for Apple QuickTime 4 or RealPlayer. Cisco IP/TV only works with NAT if the Viewer and Content Manager are on the outside network and the server is on the inside network.

SIP Inspection

This section describes SIP application inspection. This section includes the following topics:

- [SIP Inspection Overview, page 22-42](#)
- [SIP Instant Messaging, page 22-42](#)
- [Configuring SIP Timeout Values, page 22-43](#)
- [Verifying and Monitoring SIP Inspection, page 22-44](#)

SIP Inspection Overview

SIP, as defined by the IETF, enables call handling sessions, particularly two-party audio conferences, or “calls.” SIP works with SDP for call signalling. SDP specifies the ports for the media stream. Using SIP, the security appliance can support any SIP VoIP gateways and VoIP proxy servers. SIP and SDP are defined in the following RFCs:

- SIP: Session Initiation Protocol, RFC 2543
- SDP: Session Description Protocol, RFC 2327

To support SIP calls through the security appliance, signaling messages for the media connection addresses, media ports, and embryonic connections for the media must be inspected, because while the signaling is sent over a well-known destination port (UDP/TCP 5060), the media streams are dynamically allocated. Also, SIP embeds IP addresses in the user-data portion of the IP packet. SIP inspection applies NAT for these embedded IP addresses.

The following limitations and restrictions apply when using PAT with SIP:

- If a remote endpoint tries to register with a SIP proxy on a network protected by the security appliance, the registration fails under very specific conditions, as follows:
 - PAT is configured for the remote endpoint.
 - The SIP registrar server is on the outside network.
 - The port is missing in the contact field in the REGISTER message sent by the endpoint to the proxy server.
- If a SIP device transmits a packet in which the SDP portion has an IP address in the owner/creator field (o=) that is different than the IP address in the connection field (c=), the IP address in the o= field may not be properly translated. This is due to a limitation in the SIP protocol, which does not provide a port value in the o= field.

SIP Instant Messaging

Instant Messaging refers to the transfer of messages between users in near real-time. SIP supports the Chat feature on Windows XP using Windows Messenger RTC Client version 4.7.0105 only. The MESSAGE/INFO methods and 202 Accept response are used to support IM as defined in the following RFCs:

- Session Initiation Protocol (SIP)-Specific Event Notification, RFC 3265
- Session Initiation Protocol (SIP) Extension for Instant Messaging, RFC 3428

MESSAGE/INFO requests can come in at any time after registration/subscription. For example, two users can be online at any time, but not chat for hours. Therefore, the SIP inspection engine opens pinholes that time out according to the configured SIP timeout value. This value must be configured at least five minutes longer than the subscription duration. The subscription duration is defined in the Contact Expires value and is typically 30 minutes.

Because MESSAGE/INFO requests are typically sent using a dynamically allocated port other than port 5060, they are required to go through the SIP inspection engine.

**Note**

Only the Chat feature is currently supported. Whiteboard, File Transfer, and Application Sharing are not supported. RTC Client 5.0 is not supported.

SIP inspection translates the SIP text-based messages, recalculates the content length for the SDP portion of the message, and recalculates the packet length and checksum. It dynamically opens media connections for ports specified in the SDP portion of the SIP message as address/ports on which the endpoint should listen.

SIP inspection has a database with indices CALL_ID/FROM/TO from the SIP payload. These indices identify the call, the source, and the destination. This database contains the media addresses and media ports found in the SDP media information fields and the media type. There can be multiple media addresses and ports for a session. The security appliance opens RTP/RTCP connections between the two endpoints using these media addresses/ports.

The well-known port 5060 must be used on the initial call setup (INVITE) message; however, subsequent messages may not have this port number. The SIP inspection engine opens signaling connection pinholes, and marks these connections as SIP connections. This is done for the messages to reach the SIP application and be translated.

As a call is set up, the SIP session is in the “transient” state until the media address and media port is received from the called endpoint in a Response message indicating the RTP port the called endpoint listens on. If there is a failure to receive the response messages within one minute, the signaling connection is torn down.

Once the final handshake is made, the call state is moved to active and the signaling connection remains until a BYE message is received.

If an inside endpoint initiates a call to an outside endpoint, a media hole is opened to the outside interface to allow RTP/RTCP UDP packets to flow to the inside endpoint media address and media port specified in the INVITE message from the inside endpoint. Unsolicited RTP/RTCP UDP packets to an inside interface does not traverse the security appliance, unless the security appliance configuration specifically allows it.

Configuring SIP Timeout Values

The media connections are torn down within two minutes after the connection becomes idle. This is, however, a configurable timeout and can be set for a shorter or longer period of time. To configure the timeout for the SIP control connection, enter the following command:

```
hostname(config)# timeout sip hh:mm:ss
```

This command configures the idle timeout after which a SIP control connection is closed.

To configure the timeout for the SIP media connection, enter the following command:

```
hostname(config)# timeout sip_media hh:mm:ss
```

This command configures the idle timeout after which a SIP media connection is closed.

Verifying and Monitoring SIP Inspection

The **show sip** command assists in troubleshooting SIP inspection engine issues and is described with the **inspect protocol sip udp 5060** command. The **show timeout sip** command displays the timeout value of the designated protocol.

The **show sip** command displays information for SIP sessions established across the security appliance. Along with the **debug sip** and **show local-host** commands, this command is used for troubleshooting SIP inspection engine issues.



Note

We recommend that you configure the **pager** command before entering the **show sip** command. If there are a lot of SIP session records and the **pager** command is not configured, it takes a while for the **show sip** command output to reach its end.

The following is sample output from the **show sip** command:

```
hostname# show sip
Total: 2
call-id c3943000-960ca-2e43-228f@10.130.56.44
    state Call init, idle 0:00:01
call-id c3943000-860ca-7e1f-11f7@10.130.56.45
    state Active, idle 0:00:06
```

This sample shows two active SIP sessions on the security appliance (as shown in the Total field). Each call-id represents a call.

The first session, with the call-id c3943000-960ca-2e43-228f@10.130.56.44, is in the state Call Init, which means the session is still in call setup. Call setup is not complete until a final response to the call has been received. For instance, the caller has already sent the INVITE, and maybe received a 100 Response, but has not yet seen the 200 OK, so the call setup is not complete yet. Any non-1xx response message is considered a final response. This session has been idle for 1 second.

The second session is in the state Active, in which call setup is complete and the endpoints are exchanging media. This session has been idle for 6 seconds.

Skinny (SCCP) Inspection

This section describes SCCP application inspection. This section includes the following topics:

- [SCCP Inspection Overview, page 22-44](#)
- [Supporting Cisco IP Phones, page 22-45](#)
- [Restrictions and Limitations, page 22-45](#)
- [Verifying and Monitoring SCCP Inspection, page 22-46](#)

SCCP Inspection Overview

Skinny (SCCP) is a simplified protocol used in VoIP networks. Cisco IP Phones using SCCP can coexist in an H.323 environment. When used with Cisco CallManager, the SCCP client can interoperate with H.323 compliant terminals. Application layer functions in the security appliance recognize SCCP Version 3.3. There are 5 versions of the SCCP protocol: 2.4, 3.0.4, 3.1.1, 3.2, and 3.3.2. The security appliance supports all versions through Version 3.3.2.

The security appliance supports PAT and NAT for SCCP. PAT is necessary if you have more IP phones than global IP addresses for the IP phones to use. By supporting NAT and PAT of SCCP Signaling packets, Skinny application inspection ensures that all SCCP signalling and media packets can traverse the security appliance.

Normal traffic between Cisco CallManager and Cisco IP Phones uses SCCP and is handled by SCCP inspection without any special configuration. The security appliance also supports DHCP options 150 and 66, which it accomplishes by sending the location of a TFTP server to Cisco IP Phones and other DHCP clients. Cisco IP Phones might also include DHCP option 3 in their requests, which sets the default route. For more information, see the [“Using Cisco IP Phones with a DHCP Server”](#) section on page 8-27.

Supporting Cisco IP Phones

In topologies where Cisco CallManager is located on the higher security interface with respect to the Cisco IP Phones, if NAT is required for the Cisco CallManager IP address, the mapping must be **static** as a Cisco IP Phone requires the Cisco CallManager IP address to be specified explicitly in its configuration. An static identity entry allows the Cisco CallManager on the higher security interface to accept registrations from the Cisco IP Phones.

Cisco IP Phones require access to a TFTP server to download the configuration information they need to connect to the Cisco CallManager server.

When the Cisco IP Phones are on a lower security interface compared to the TFTP server, you must use an access list to connect to the protected TFTP server on UDP port 69. While you do need a static entry for the TFTP server, this does not have to be an identity static entry. When using NAT, an identity static entry maps to the same IP address. When using PAT, it maps to the same IP address and port.

When the Cisco IP Phones are on a *higher* security interface compared to the TFTP server and Cisco CallManager, no access list or static entry is required to allow the Cisco IP Phones to initiate the connection.

Restrictions and Limitations

The following are limitations that apply to the current version of PAT and NAT support for SCCP:

- PAT does not work with configurations containing the **alias** command.
- Outside NAT or PAT is *not* supported.

If the address of an internal Cisco CallManager is configured for NAT or PAT to a different IP address or port, registrations for external Cisco IP Phones fail because the security appliance currently does not support NAT or PAT for the file content transferred over TFTP. Although the security appliance supports NAT of TFTP messages and opens a pinhole for the TFTP file, the security appliance cannot translate the Cisco CallManager IP address and port embedded in the Cisco IP Phone configuration files that are transferred by TFTP during phone registration.



Note

The security appliance supports stateful failover of SCCP calls except for calls that are in the middle of call setup.

Verifying and Monitoring SCCP Inspection

The **show skinny** command assists in troubleshooting SCCP (Skinny) inspection engine issues. The following is sample output from the **show skinny** command under the following conditions. There are two active Skinny sessions set up across the security appliance. The first one is established between an internal Cisco IP Phone at local address 10.0.0.11 and an external Cisco CallManager at 172.18.1.33. TCP port 2000 is the CallManager. The second one is established between another internal Cisco IP Phone at local address 10.0.0.22 and the same Cisco CallManager.

```
hostname# show skinny
          LOCAL                FOREIGN                STATE
-----
1      10.0.0.11/52238         172.18.1.33/2000         1
    MEDIA 10.0.0.11/22948         172.18.1.22/20798
2      10.0.0.22/52232         172.18.1.33/2000         1
    MEDIA 10.0.0.22/20798         172.18.1.11/22948
```

The output indicates that a call has been established between two internal Cisco IP Phones. The RTP listening ports of the first and second phones are UDP 22948 and 20798 respectively.

The following is sample output from the **show xlate debug** command for these Skinny connections:

```
hostname# show xlate debug
2 in use, 2 most used
Flags: D - DNS, d - dump, I - identity, i - inside, n - no random,
       r - portmap, s - static
NAT from inside:10.0.0.11 to outside:172.18.1.11 flags si idle 0:00:16 timeout 0:05:00
NAT from inside:10.0.0.22 to outside:172.18.1.22 flags si idle 0:00:14 timeout 0:05:00
```

SMTP and Extended SMTP Inspection

ESMTP application inspection provides improved protection against SMTP-based attacks by restricting the types of SMTP commands that can pass through the security appliance and by adding monitoring capabilities.

ESMTP is an enhancement to the SMTP protocol and is similar in most respects to SMTP. For convenience, the term SMTP is used in this document to refer to both SMTP and ESMTP. The application inspection process for extended SMTP is similar to SMTP application inspection and includes support for SMTP sessions. Most commands used in an extended SMTP session are the same as those used in an SMTP session but an ESMTP session is considerably faster and offers more options related to reliability and security, such as delivery status notification.

Extended SMTP application inspection adds support for eight extended SMTP commands, including AUTH, EHLO, ETRN, HELP, SAML, SEND, SOML and VRFY. Along with the support for seven RFC 821 commands (DATA, HELO, MAIL, NOOP, QUIT, RCPT, RSET), the security appliance supports a total of fifteen SMTP commands.

Other extended SMTP commands, such as ATRN, STARTTLS, ONEX, VERB, CHUNKING, and private extensions are not supported. Unsupported commands are translated into Xs, which are rejected by the internal server. This results in a message such as “500 Command unknown: 'XXX'.” Incomplete commands are discarded.

The ESMTP inspection engine changes the characters in the server SMTP banner to asterisks except for the “2”, “0”, “0” characters. Carriage return (CR) and linefeed (LF) characters are ignored.

With SMTP inspection enabled, a Telnet session used for interactive SMTP may hang if the following rules are not observed: SMTP commands must be at least four characters in length; must be terminated with carriage return and line feed; and must wait for a response before issuing the next reply.

An SMTP server responds to client requests with numeric reply codes and optional human-readable strings. SMTP application inspection controls and reduces the commands that the user can use as well as the messages that the server returns. SMTP inspection performs three primary tasks:

- Restricts SMTP requests to seven basic SMTP commands and eight extended commands.
- Monitors the SMTP command-response sequence.
- Generates an audit trail—Audit record 108002 is generated when invalid character embedded in the mail address is replaced. For more information, see RFC 821.

SMTP inspection monitors the command and response sequence for the following anomalous signatures:

- Truncated commands.
- Incorrect command termination (not terminated with <CR><LR>).
- The MAIL and RCPT commands specify who are the sender and the receiver of the mail. Mail addresses are scanned for strange characters. The pipeline character (|) is deleted (changed to a blank space) and “<” ,”>” are only allowed if they are used to define a mail address (“>” must be preceded by “<”).
- Unexpected transition by the SMTP server.
- For unknown commands, the security appliance changes all the characters in the packet to X. In this case, the server generates an error code to the client. Because of the change in the packed, the TCP checksum has to be recalculated or adjusted.
- TCP stream editing.
- Command pipelining.

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 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 Inspection”](#) section on page 22-5.

To create an SNMP 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
```

SQL*Net Inspection

SQL*Net inspection is enabled by default.

The SQL*Net protocol consists of different packet types that the security appliance handles to make the data stream appear consistent to the Oracle applications on either side of the security appliance.

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). Use the **class-map** command to apply SQL*Net inspection to a range of port numbers.



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.

The security appliance translates all addresses and looks in the packets for all embedded ports to open for SQL*Net Version 1.

For SQL*Net Version 2, all DATA or REDIRECT packets that immediately follow REDIRECT packets with a zero data length will be fixed up.

The packets that need fix-up contain embedded host/port addresses in the following format:

```
(ADDRESS=(PROTOCOL=tcp) (DEV=6) (HOST=a.b.c.d) (PORT=a))
```

SQL*Net Version 2 TNSFrame types (Connect, Accept, Refuse, Resend, and Marker) will not be scanned for addresses to NAT nor will inspection open dynamic connections for any embedded ports in the packet.

SQL*Net Version 2 TNSFrames, Redirect, and Data packets will be scanned for ports to open and addresses to NAT, if preceded by a REDIRECT TNSFrame type with a zero data length for the payload. When the Redirect message with data length zero passes through the security appliance, a flag will be set in the connection data Structure to expect the Data or Redirect message that follows to be translated and ports to be dynamically opened. If one of the TNS frames in the preceding paragraph arrive after the Redirect message, the flag will be reset.

The SQL*Net inspection engine will recalculate the checksum, change IP, TCP lengths, and readjust Sequence Numbers and Acknowledgment Numbers using the delta of the length of the new and old message.

SQL*Net Version 1 is assumed for all other cases. TNSFrame types (Connect, Accept, Refuse, Resend, Marker, Redirect, and Data) and all packets will be scanned for ports and addresses. Addresses will be translated and port connections will be opened.

Sun RPC Inspection

This section describes Sun RPC application inspection. This section includes the following topics:

- [Sun RPC Inspection Overview, page 22-49](#)
- [Managing Sun RPC Services, page 22-49](#)
- [Verifying and Monitoring Sun RPC Inspection, page 22-50](#)

Sun RPC Inspection Overview

The Sun RPC inspection engine enables or disables application inspection for the Sun RPC protocol. Sun RPC is used by NFS and NIS. Sun RPC services can run on any port. When a client attempts to access an 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 security appliance intercepts this packet and opens both embryonic TCP and UDP connections on that port.



Note

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

Managing Sun RPC Services

Use the Sun RPC services table to control Sun RPC traffic through the security appliance based on established Sun RPC sessions. To create entries in the Sun RPC services table, use the `sunrpc-server` command in global configuration mode:

```
hostname(config)# sunrpc-server interface_name ip_address mask service service_type
protocol {tcp | udp} port[-port] timeout hh:mm:ss
```

You can use this command to specify the timeout after which the pinhole that was opened by Sun RPC application inspection will be closed. 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:

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

This command specifies that the pinhole that was opened by Sun RPC application inspection will be closed after 30 minutes. In this example, the Sun RPC server is on the inside interface using TCP port 111. You can also specify UDP, a different port number, or a range of ports. To specify a range of ports, separate the starting and ending port numbers in the range with a hyphen (for example, 111-113).

The service type identifies the mapping between a specific service type and the port number used for the service. To determine the service type, which in this example is 100003, use the `sunrpcinfo` command at the UNIX or Linux command line on the Sun RPC server machine.

To clear the Sun RPC configuration, enter the following command.

```
hostname(config)# clear configure sunrpc-server
```

This removes the configuration performed using the `sunrpc-server` command. The `sunrpc-server` command allows pinholes to be created with a specified timeout.

To clear the active Sun RPC services, enter the following command:

```
hostname(config)# clear sunrpc-server active
```

This clears the pinholes that are opened by Sun RPC application inspection for specific services, such as NFS or NIS.

Verifying and Monitoring Sun RPC Inspection

The sample output in this section is for a Sun RPC server with an IP address of 192.168.100.2 on the inside interface and a Sun RPC client with an IP address of 209.168.200.5 on the outside interface.

To view information about the current Sun RPC connections, enter the **show conn** command. The following is sample output from the **show conn** command:

```
hostname# show conn
15 in use, 21 most used
UDP out 209.165.200.5:800 in 192.168.100.2:2049 idle 0:00:04 flags -
UDP out 209.165.200.5:714 in 192.168.100.2:111 idle 0:00:04 flags -
UDP out 209.165.200.5:712 in 192.168.100.2:647 idle 0:00:05 flags -
UDP out 192.168.100.2:0 in 209.165.200.5:714 idle 0:00:05 flags i
hostname(config)#
```

To display the information about the Sun RPC service table configuration, enter the **show running-config sunrpc-server** command. The following is sample output from the **show running-config sunrpc-server** command:

```
hostname(config)# show running-config sunrpc-server
sunrpc-server inside 192.168.100.2 255.255.255.255 service 100003 protocol UDP port 111
timeout 0:30:00
sunrpc-server inside 192.168.100.2 255.255.255.255 service 100005 protocol UDP port 111
timeout 0:30:00
```

This output shows that a timeout interval of 30 minutes is configured on UDP port 111 for the Sun RPC server with the IP address 192.168.100.2 on the inside interface.

To display the pinholes open for Sun RPC services, enter the **show sunrpc-server active** command. The following is sample output from **show sunrpc-server active** command:

```
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.

To view information about the Sun RPC services running on a Sun RPC server, enter the **rpcinfo -p** command from the Linux or UNIX server command line. The following is sample output from the **rpcinfo -p** command:

```
sunrpcserver:~ # rpcinfo -p
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100024 1 udp 632 status
100024 1 tcp 635 status
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100003 2 tcp 2049 nfs
100003 3 tcp 2049 nfs
100021 1 udp 32771 nlockmgr
```

```

100021 3 udp 32771 nlockmgr
100021 4 udp 32771 nlockmgr
100021 1 tcp 32852 nlockmgr
100021 3 tcp 32852 nlockmgr
100021 4 tcp 32852 nlockmgr
100005 1 udp 647 mountd
100005 1 tcp 650 mountd
100005 2 udp 647 mountd
100005 2 tcp 650 mountd
100005 3 udp 647 mountd
100005 3 tcp 650 mountd

```

In this output, port 647 corresponds to the mountd daemon running over UDP. The mountd process would more commonly be using port 32780. The mountd process running over TCP uses port 650 in this example.

TFTP Inspection

TFTP inspection is enabled by default.

TFTP, described in RFC 1350, is a simple protocol to read and write files between a TFTP server and client.

The security appliance inspects TFTP traffic and dynamically creates connections and translations, if necessary, to permit file transfer between a TFTP client and server. Specifically, the inspection engine inspects TFTP read request (RRQ), write request (WRQ), and error notification (ERROR).

A dynamic secondary channel and a PAT translation, if necessary, are allocated on a reception of a valid read (RRQ) or write (WRQ) request. This secondary channel is subsequently used by TFTP for file transfer or error notification.

Only the TFTP server can initiate traffic over the secondary channel, and at most one incomplete secondary channel can exist between the TFTP client and server. An error notification from the server closes the secondary channel.

TFTP inspection must be enabled if static PAT is used to redirect TFTP traffic.

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 security appliance must allow the TCP back connection from the Xhosted computer. To permit the back connection, use the **established** command on the 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 security appliance can NAT if needed. XDCMP inspection does not support PAT.

