

Getting Started with Application Layer Protocol Inspection

The following topics describe how to configure application layer protocol inspection.

- Application Layer Protocol Inspection, on page 1
- Configure Application Layer Protocol Inspection, on page 9
- Configure Regular Expressions, on page 15
- Monitoring Inspection Policies, on page 19
- History for Application Inspection, on page 20

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 ASA to do a deep packet inspection instead of passing the packet through the fast path. As a result, inspection engines can affect overall throughput. Several common inspection engines are enabled on the ASA by default, but you might need to enable others depending on your network.

The following topics explain application inspection in more detail.

When to Use Application Protocol Inspection

When a user establishes a connection, the ASA checks the packet against ACLs, 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 ASA.

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 ASA 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 ASA monitors sessions to identify the dynamic port assignments, and permits data exchange on these ports for the duration of the specific session.

Inspection Policy Maps

You can configure special actions for many application inspections using an *inspection policy map*. These maps are optional: you can enable inspection for a protocol that supports inspection policy maps without configuring a map. These maps are needed only if you want something other than the default inspection actions.

An inspection policy map consists of one or more of the following elements. The exact options available for an inspection policy map depends on the application.

- Traffic matching criteria—You match application traffic to criteria specific to the application, such as a URL string, for which you then enable actions.

  For some traffic matching criteria, you use regular expressions to match text inside a packet. Be sure to create and test the regular expressions before you configure the policy map, either singly or grouped together in a regular expression class map.

- Inspection class map—Some inspection policy maps let you use an inspection class map to include multiple traffic matching criteria. You then identify the inspection class map in the inspection policy map and enable actions for the class as a whole. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that you can create more complex match criteria and you can reuse class maps. However, you cannot set different actions for different matches.

- Parameters—Parameters affect the behavior of the inspection engine.

The following topics provide more details.

Replacing an In-Use Inspection Policy Map

If you have an inspection enabled with a policy map in a service policy, replacing the policy map is a two-step process. First, you must remove the inspection. Then, you add it back with the new policy map name.

For example, to replace sip-map1 with sip-map2 in SIP inspection, use the following command sequence:

```
hostname(config)# policy-map test
hostname(config-pmap)# class sip
hostname(config-pmap-c)# no inspect sip sip-map1
hostname(config-pmap-c)# inspect sip sip-map2
```

How Multiple Traffic Classes are Handled

You can specify multiple inspection class maps or direct matches in the inspection policy map.

If a packet matches multiple different classes or direct matches, then the order in which the ASA applies the actions is determined by internal ASA rules, and not by the order they are added to the inspection policy map. The internal rules are determined by the application type and the logical progression of parsing a packet, and are not user-configurable. For example for HTTP traffic, parsing a Request Method field precedes parsing the Header Host Length field; an action for the Request Method field occurs before the action for the Header Host Length field. For example, the following match commands can be entered in any order, but the **match request method get** command is matched first.

```
match request header host length gt 100
  reset
match request method get
  log
```

If an action drops a packet, then no further actions are performed in the inspection policy map. For example, if the first action is to reset the connection, then it will never match any further match criteria. If the first action is to log the packet, then a second action, such as resetting the connection, can occur.

If a packet matches multiple match criteria that are the same, then they are matched in the order they appear in the policy map. For example, for a packet with the header length of 1001, it will match the first command below, and be logged, and then will match the second command and be reset. If you reverse the order of the two **match** commands, then the packet will be dropped and the connection reset before it can match the second **match** command; it will never be logged.

```
match request header length gt 100
  log
match request header length gt 1000
  reset
```

A class map is determined to be the same type as another class map or direct match based on the lowest priority match option in the class map (the priority is based on the internal rules). If a class map has the same type of lowest priority match option as another class map, then the class maps are matched according to the order they are added to the policy map. If the lowest priority match for each class map is different, then the class map with the higher priority match option is matched first. For example, the following three class maps contain two types of **match** commands: **match request-cmd** (higher priority) and **match filename** (lower priority). The ftp3 class map includes both commands, but it is ranked according to the lowest priority command, **match filename**. The ftp1 class map includes the highest priority command, so it is matched first, regardless of the order in the policy map. The ftp3 class map is ranked as being of the same priority as the ftp2 class map, which also contains the **match filename** command. They are matched according to the order in the policy map: ftp3 and then ftp2.

```
class-map type inspect ftp match-all ftp1
  match request-cmd get
class-map type inspect ftp match-all ftp2
  match filename regex abc
class-map type inspect ftp match-all ftp3
  match request-cmd get
  match filename regex abc

policy-map type inspect ftp ftp
  class ftp3
    log
  class ftp2
    log
  class ftp1
    log
```

Guidelines for Application Inspection

Failover

State information for multimedia sessions that require inspection are not passed over the state link for stateful failover. The exceptions are GTP, M3UA, and SIP, which are replicated over the state link. You must configure strict application server process (ASP) state checking in M3UA inspection to get stateful failover.

Clustering

The following inspections are not supported in clustering:

- CTIQBE
- H323, H225, and RAS
- IPsec passthrough
- MGCP
- MMP
- RTSP
- SCCP (Skinny)
- WAAS

IPv6

Supports IPv6 for the following inspections:

- Diameter
- DNS over UDP
- FTP
- GTP
- HTTP
- ICMP
- IPsec pass-through
- IPv6
- M3UA
- SCCP (Skinny)
- SCTP
- SIP
- SMTP
- VXLAN

Supports NAT64 for the following inspections:

- DNS over UDP
- FTP
- HTTP
- ICMP

- SCTP

Additional Guidelines

- Some inspection engines do not support PAT, NAT, outside NAT, or NAT between same security interfaces. For more information about NAT support, see Default Inspections and NAT Limitations, on page 5.
- For all the application inspections, the ASA 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.
- Inspected protocols are subject to advanced TCP-state tracking, and the TCP state of these connections is not automatically replicated. While these connections are replicated to the standby unit, there is a best-effort attempt to re-establish a TCP state.
- If the system determines that a TCP connection requires inspection, the system clears all TCP options except for the MSS and selective-acknowledgment (SACK) options on the packets before inspecting them. Other options are cleared even if you allow them in a TCP map applied to the connections.
- TCP/UDP Traffic directed to the ASA (to an interface) is inspected by default. However, ICMP traffic directed to an interface is never inspected, even if you enable ICMP inspection. Thus, a ping (echo request) to an interface can fail under specific circumstances, such as when the echo request comes from a source that the ASA can reach through a backup default route.

Defaults for Application Inspection

The following topics explain the default operations for application inspection.

Default Inspections and NAT Limitations

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.

The following table 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. In this table:

- Inspection engines that are enabled by default for the default port are in bold.
- The ASA is in compliance with the indicated 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 ASA does not enforce the order.

Table 1: Supported Application Inspection Engines

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
CTIQBE	TCP/2748	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
DCERPC	TCP/135	No NAT64.	—	—
Diameter	TCP/3868 TCP/5868 (for TCP/TLS) SCTP/3868	No NAT/PAT.	RFC 6733	Requires the Carrier license.
DNS over UDP DNS over TCP	UDP/53 UDP/443 TCP/53	No NAT support is available for name resolution through WINS.	RFC 1123	You must enable DNS/TCP inspection in the DNS inspection policy map to inspect DNS over TCP. UDP/443 is used for Cisco Umbrella DNSCrypt sessions only.
FTP	TCP/21	(Clustering) No static PAT.	RFC 959	—
GTP	UDP/3386 (GTPv0) UDP/2123 (GTPv1+)	No extended PAT. No NAT.	—	Requires the Carrier license.
H.323 H.225 and **RAS**	TCP/1720 UDP/1718 UDP (RAS) 1718-1719	(Clustering) No static PAT. No extended PAT. No NAT on same security interfaces. No NAT64.	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	ICMP	—	—	ICMP traffic directed to an ASA interface is never inspected.
ICMP ERROR	ICMP	—	—	—
ILS (LDAP)	TCP/389	No extended PAT. No NAT64.	—	—

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
Instant Messaging (IM)	Varies by client	No extended PAT. No NAT64.	RFC 3860	—
IP Options	RSVP	No NAT64.	RFC 791, RFC 2113	—
IPsec Pass Through	UDP/500	No PAT. No NAT64.	—	—
IPv6	—	No NAT64.	RFC 2460	—
LISP	—	No NAT or PAT.	—	—
M3UA	SCTP/2905	No NAT or PAT for embedded addresses.	RFC 4666	Requires the Carrier license.
MGCP	UDP/2427, 2727	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2705bis-05	—
MMP	TCP/5443	No extended PAT. No NAT64.	—	—
NetBIOS Name Server over IP	UDP/137, 138 (Source ports)	No extended PAT. No NAT64.	—	NetBIOS is supported by performing NAT of the packets for NBNS UDP port 137 and NBDS UDP port 138.
PPTP	TCP/1723	No NAT64. (Clustering) No static PAT.	RFC 2637	—
RADIUS Accounting	UDP/1646	No NAT64.	RFC 2865	—
RSH	TCP/514	No PAT. No NAT64. (Clustering) No static PAT.	Berkeley UNIX	—
RTSP	TCP/554	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2326, 2327, 1889	No handling for HTTP cloaking.

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
SCTP	SCTP	—	RFC 4960	Requires the Carrier license. Although you can do static network object NAT on SCTP traffic (no dyamic NAT/PAT), the inspection engine is not used for NAT.
SIP	TCP/5060 UDP/5060	No NAT/PAT on interfaces with the same, or lower to higher, security levels. No extended PAT. No NAT64 or NAT46. (Clustering) No static PAT.	RFC 2543	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SKINNY (SCCP)	TCP/2000	No NAT on same security interfaces. No extended PAT. No NAT64, NAT46, or NAT66. (Clustering) No static PAT.	—	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SMTP and **ESMTP**	TCP/25	No NAT64.	RFC 821, 1123	—
SNMP	UDP/161, 162 UDP/4161 on platforms that also run Secure Firewall eXtensible Operating System (FXOS).	No NAT or PAT.	RFC 1155, 1157, 1212, 1213, 1215	v.2 RFC 1902-1908; v.3 RFC 2570-2580. On FXOS platforms, if you configure SNMP, this inspection is enabled automatically, and you cannot disable it.
SQL*Net	TCP/1521	No extended PAT. No NAT64. (Clustering) No static PAT.	—	v.1 and v.2.
STUN	TCP/3478 UDP/3478	(WebRTC) Static NAT/PAT44 only. (Cisco Spark) Static NAT/PAT44 and 64; and dynamic NAT/PAT.	RFC 5245, 5389	—
Sun RPC	TCP/111 UDP/111	No extended PAT. No NAT64.	—	—
TFTP	UDP/69	No NAT64. (Clustering) No static PAT.	RFC 1350	Payload IP addresses are not translated.

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
WAAS	TCP/1- 65535	No extended PAT. No NAT64.	—	—
XDMCP	UDP/177	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
VXLAN	UDP/4789	Not applicable	RFC 7348	Virtual Extensible Local Area Network.

Default Inspection Policy Maps

Some inspection types use hidden default policy maps. For example, if you enable ESMTP inspection without specifying a map, _default_esmtp_map is used.

The default inspection is described in the sections that explain each inspection type. You can view these default maps using the **show running-config all policy-map** command.

DNS inspection is the only one that uses an explicitly-configured default map, preset_dns_map.

Configure Application Layer Protocol Inspection

You configure application inspection in service policies.

Inspection is enabled by default globally on all interfaces for some applications on their standard ports and protocols. See Default Inspections and NAT Limitations, on page 5 for more information on default inspections. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Before you begin

For some applications, you can perform special actions when you enable inspection by configuring inspection policy maps. The table later in this procedure shows which protocols allow inspection policy maps, with pointers to the instructions on configuring them. If you want to configure these advanced features, create the map before configuring inspection.

Procedure

Step 1 Unless you are adding inspection to an existing class map, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

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

Example:

```
hostname(config)# class-map dns_class_map
hostname(config-cmap)# match access-list dns
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). You can have more than one inspection on the inspection_default class only, and you might want to simply edit the existing global policy that applies the inspection defaults. If you are using this class map in either the default policy or for a new service policy, you can skip this step. For detailed information on which class map to choose, see Choosing the Right Traffic Class for Inspection, on page 14.

For information on matching statements, see Create a Layer 3/4 Class Map for Through Traffic. For RADIUS accounting inspection, which uses a management layer 3/4 class, see Configure RADIUS Accounting Inspection.

Step 2 Add or edit a Layer 3/4 policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

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

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

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

Example:

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

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

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. Do not add another class that matches SNMP.

Step 4 Enable application inspection: **inspect** *protocol*

The *protocol* is one of the following values:

Table 2: Inspection Protocol Keywords

Keywords	Notes
ctiqbe	See CTIQBE Inspection.
dcerpc [*map_name*]	See DCERPC Inspection. If you added a DCERPC inspection policy map according to Configure a DCERPC Inspection Policy Map, identify the map name in this command.

Keywords	Notes
diameter [*map_name*] [**tls-proxy** *proxy_name*]	See Diameter Inspection. If you added a Diameter inspection policy map according to Configure a Diameter Inspection Policy Map, identify the map name in this command. **tls-proxy** *proxy_name* identifies the TLS proxy to use for this inspection. You need a TLS proxy only if you want to enable inspection of encrypted traffic.
dns [*map_name*] [**dynamic-filter-snoop**]	See DNS Inspection. If you added a DNS inspection policy map according to Configure DNS Inspection Policy Map, identify the map name in this command. The default DNS inspection policy map name is "preset_dns_map." **dynamic-filter-snoop** enables dynamic filter snooping, which is used exclusively by the Botnet Traffic Filter. Include this keyword only if you use Botnet Traffic Filtering. We suggest that you enable DNS snooping only on interfaces where external DNS requests are going. Enabling DNS snooping on all UDP DNS traffic, including that going to an internal DNS server, creates unnecessary load on the ASA.
esmtp [*map_name*]	See SMTP and Extended SMTP Inspection. If you added an ESMTP inspection policy map according to Configure an ESMTP Inspection Policy Map, identify the map name in this command.
ftp [**strict** [*map_name*]]	See FTP Inspection. Use the **strict** keyword to increase the security of protected networks by preventing web browsers from sending embedded commands in FTP requests. See Strict FTP for more information. If you added an FTP inspection policy map according to Configure an FTP Inspection Policy Map, identify the map name in this command.
gtp [*map_name*]	See GTP Inspection Overview. If you added a GTP inspection policy map according to Configure a GTP Inspection Policy Map, identify the map name in this command.
h323 h225 [*map_name*]	See H.323 Inspection. If you added an H323 inspection policy map according to Configure H.323 Inspection Policy Map, identify the map name in this command.
h323 ras [*map_name*]	See H.323 Inspection. If you added an H323 inspection policy map according to Configure H.323 Inspection Policy Map, identify the map name in this command.

Keywords	Notes
http [*map_name*]	See HTTP Inspection. If you added an HTTP inspection policy map according to Configure an HTTP Inspection Policy Map, identify the map name in this command.
icmp	See ICMP Inspection.
icmp error	See ICMP Error Inspection.
ils	See ILS Inspection.
im [*map_name*]	See Instant Messaging Inspection. If you added an Instant Messaging inspection policy map, identify the map name in this command.
ip-options [*map_name*]	See IP Options Inspection. If you added an IP Options inspection policy map according to Configure an IP Options Inspection Policy Map, identify the map name in this command.
ipsec-pass-thru [*map_name*]	See IPsec Pass Through Inspection. If you added an IPsec Pass Through inspection policy map according to Configure an IPsec Pass Through Inspection Policy Map, identify the map name in this command.
ipv6 [*map_name*]	See IPv6 Inspection. If you added an IPv6 inspection policy map according to Configure an IPv6 Inspection Policy Map, identify the map name in this command.
lisp [*map_name*]	For detailed information on configuring LISP, including inspection, see the clustering chapter in the general configuration guide. If you added a LISP inspection policy map, identify the map name in this command.
m3ua [*map_name*]	See M3UA Inspection. If you added an M3UA inspection policy map according to Configure an M3UA Inspection Policy Map, identify the map name in this command.
mgcp [*map_name*]	See MGCP Inspection. If you added an MGCP inspection policy map according to Configure an MGCP Inspection Policy Map, identify the map name in this command.
netbios [*map_name*]	See NetBIOS Inspection. If you added a NetBIOS inspection policy map, identify the map name in this command.

Keywords	Notes
pptp	See PPTP Inspection.
radius-accounting *map_name*	See RADIUS Accounting Inspection Overview. The **radius-accounting** keyword is only available for a management class map. You must specify a RADIUS accounting inspection policy map; see Configure a RADIUS Accounting Inspection Policy Map.
rsh	See RSH Inspection.
rtsp [*map_name*]	See RTSP Inspection. If you added a RTSP inspection policy map according to Configure RTSP Inspection Policy Map, identify the map name in this command.
sctp [*map_name*]	See SCTP Application Layer Inspection. If you added an SCTP inspection policy map according to Configure an SCTP Inspection Policy Map, identify the map name in this command.
sip [*map_name*] [**tls-proxy** *proxy_name*]	See SIP Inspection. If you added a SIP inspection policy map according to Configure SIP Inspection Policy Map, identify the map name in this command. **tls-proxy** *proxy_name* identifies the TLS proxy to use for this inspection. You need a TLS proxy only if you want to enable inspection of encrypted traffic.
skinny [*map_name*]	See Skinny (SCCP) Inspection. If you added a Skinny inspection policy map according to Configure a Skinny (SCCP) Inspection Policy Map, identify the map name in this command.
snmp [*map_name*]	See SNMP Inspection. If you added an SNMP inspection policy map, identify the map name in this command.
sqlnet	See SQL*Net Inspection.
stun	See STUN Inspection.
sunrpc	See Sun RPC Inspection. 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	See TFTP Inspection.
waas	Enables TCP option 33 parsing. Use when deploying Cisco Wide Area Application Services products.

Keywords	Notes
xdmcp	See XDMCP Inspection.
vxlan	See VXLAN Inspection.

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

Example:

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

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

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

Example:

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

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

Choosing the Right Traffic Class for Inspection

The default Layer 3/4 class map for through traffic is called "inspection_default." It matches traffic using a special **match** command, **match default-inspection-traffic**, to match the default protocols and ports for each application protocol. This traffic class (along with **match any**, which is not typically used for inspection) matches both IPv4 and IPv6 traffic for inspections that support IPv6. See Guidelines for Application Inspection, on page 3 for a list of IPv6-enabled inspections.

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 ACL are ignored.

Tip We suggest that you only inspect traffic on ports on which you expect application traffic; if you inspect all traffic, for example using **match any**, the ASA performance can be impacted.

If you want to match non-standard ports, then create a new class map for the non-standard ports. See Default Inspections and NAT Limitations, on page 5 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. 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 ACL 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
```

Configure Regular Expressions

Regular expressions define pattern matching for text strings. You can use these expressions in some protocol inspection maps to match packets based on strings such as URLs or the contents of particular header fields.

Create a Regular Expression

A regular expression matches text strings either literally as an exact string, or by using *metacharacters* so that you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet.

Before you begin

Use **Ctrl+V** to escape all of the special characters in the CLI, such as question mark (?) or a tab. For example, type **d[Ctrl+V]?g** to enter **d?g** in the configuration.

See the **regex** command in the command reference for performance impact information when matching a regular expression to packets. In general, matching against long input strings, or trying to match a large number of regular expressions, will reduce system performance.

Note As an optimization, the ASA searches on the deobfuscated URL. Deobfuscation compresses multiple forward slashes (/) into a single slash. For strings that commonly use double slashes, like “http://”, be sure to search for “http:/” instead.

The following table lists the metacharacters that have special meanings.

Table 3: Regular Expression Metacharacters

Character	Description	Notes
.	Dot	Matches any single character. For example, **d.g** matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.
(*exp*)	Subexpression	A subexpression segregates characters from surrounding characters, so that you can use other metacharacters on the subexpression. For example, **d(o|a)g** matches dog and dag, but **do|ag** matches do and ag. A subexpression can also be used with repeat quantifiers to differentiate the characters meant for repetition. For example, **ab(xy){3}z** matches abxyxyxyz.
|	Alternation	Matches either expression it separates. For example, **dog|cat** matches dog or cat.
?	Question mark	A quantifier that indicates that there are 0 or 1 of the previous expression. For example, **lo?se** matches lse or lose.
*	Asterisk	A quantifier that indicates that there are 0, 1 or any number of the previous expression. For example, **lo*se** matches lse, lose, loose, and so on.
+	Plus	A quantifier that indicates that there is at least 1 of the previous expression. For example, **lo+se** matches lose and loose, but not lse.
{*x*} or {*x*,}	Minimum repeat quantifier	Repeat at least *x* times. For example, **ab(xy){2,}z** matches abxyxyz, abxyxyxyz, and so on.
[*abc*]	Character class	Matches any character in the brackets. For example, **[abc]** matches a, b, or c.
[^*abc*]	Negated character class	Matches a single character that is not contained within the brackets. For example, **[^abc]** matches any character other than a, b, or c. **[^A-Z]** matches any single character that is not an uppercase letter.

Character	Description	Notes
[*a-c*]	Character range class	Matches any character in the range. **[a-z]** matches any lowercase letter. You can mix characters and ranges: **[abcq-z]** matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [**a-cq-z]**. The dash (-) character is literal only if it is the last or the first character within the brackets: **[abc-]** or **[-abc]**.
“”	Quotation marks	Preserves trailing or leading spaces in the string. For example, **“ test”** preserves the leading space when it looks for a match.
^	Caret	Specifies the beginning of a line.
\	Escape character	When used with a metacharacter, matches a literal character. For example, **\[** matches the left square bracket.
char	Character	When character is not a metacharacter, matches the literal character.
\r	Carriage return	Matches a carriage return 0x0d.
\n	Newline	Matches a new line 0x0a.
\t	Tab	Matches a tab 0x09.
\f	Formfeed	Matches a form feed 0x0c.
\x*NN*	Escaped hexadecimal number	Matches an ASCII character using hexadecimal (exactly two digits).
NNN	Escaped octal number	Matches an ASCII character as octal (exactly three digits). For example, the character 040 represents a space.

Procedure

Step 1 Test a regular expression to make sure it matches what you think it will match: **test regex** *input_text regular_expression*

Where the *input_text* argument is a string you want to match using the regular expression, up to 201 characters in length. The *regular_expression* argument can be up to 100 characters in length.

Use **Ctrl+V** to escape all of the special characters in the CLI. For example, to enter a tab in the input text in the **test regex** command, you must enter **test regex “test[Ctrl+V Tab]” “test\t”**.

If the regular expression matches the input text, you see the following message:

```
INFO: Regular expression match succeeded.
```

If the regular expression does not match the input text, you see the following message:

```
INFO: Regular expression match failed.
```

Step 2 To add a regular expression after you tested it, enter the following command: **regex** *name regular_expression*

Where the *name* argument can be up to 40 characters in length. The *regular_expression* argument can be up to 100 characters in length.

Examples

The following example creates two regular expressions for use in an inspection policy map:

```
hostname(config)# regex url_example example\.com
hostname(config)# regex url_example2 example2\.com
```

Create a Regular Expression Class Map

A regular expression class map identifies one or more regular expression. It is simply a collection of regular expression objects. You can use a regular expression class map in many cases in replace of a regular expression object.

Procedure

Step 1 Create the regular expression class map: **class-map type regex match-any** *class_map_name*

Where *class_map_name* is a string up to 40 characters in length. The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map.

The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the regular expressions.

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

Step 3 Identify the regular expressions you want to include by entering the following command for each regular expression: **match regex** *regex_name*

Examples

The following example creates two regular expressions, and adds them to a regular expression class map. Traffic matches the class map if it includes the string "example.com" or "example2.com."

```
hostname(config)# regex url_example example\.com
hostname(config)# regex url_example2 example2\.com
hostname(config)# class-map type regex match-any URLs
hostname(config-cmap)# match regex url_example
```

```
hostname(config-cmap)# match regex url_example2
```

Monitoring Inspection Policies

To monitor inspection service policies, enter the following commands. See the command reference on Cisco.com for detailed syntax and examples.

- **show service-policy inspect** *protocol*

  Displays statistics for inspection service policies. The *protocol* is the protocol from the inspect command, for example **dns**. However, not all inspection protocols show statistics with this command. For example:

```
asa# show service-policy inspect dns

Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: dns preset_dns_map, packet 0, lock fail 0, drop 0, reset-drop 0,
5-min-pkt-rate 0 pkts/sec, v6-fail-close 0
        message-length maximum client auto, drop 0
        message-length maximum 512, drop 0
        dns-guard, count 0
        protocol-enforcement, drop 0
        nat-rewrite, count 0
asa#
```

- **show conn**

  Shows current connections for traffic passing through the device. This command has a wide range of keywords so that you can get information about various protocols.

- Additional commands for specific inspected protocols:
  - **show ctiqbe**

    Displays information about the media connections allocated by the CTIQBE inspection engine

  - **show h225**

    Displays information for H.225 sessions.

  - **show h245**

    Displays information for H.245 sessions established by endpoints using slow start.

  - **show h323 ras**

    Displays connection information for H.323 RAS sessions established between a gatekeeper and its H.323 endpoint.

  - **show mgcp** {**commands** | **sessions** }

    Displays the number of MGCP commands in the command queue or the number of existing MGCP sessions.

  - **show sip**

    Displays information for SIP sessions.

- **show skinny**

  Displays information for Skinny (SCCP) sessions.

- **show sunrpc-server active**

  Displays the pinholes opened for Sun RPC services.

History for Application Inspection

Feature Name	Releases	Description
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.