Application Layer Preprocessors

The following topics explain application layer preprocessors and how to configure them:

Introduction to Application Layer Preprocessors

Application layer protocols can represent the same data in a variety of ways. The Firepower System provides application layer protocol decoders that normalize specific types of packet data into formats that the intrusion rules engine can analyze. Normalizing application-layer protocol encodings allows the rules engine to effectively apply the same content-related rules to packets whose data is represented differently and obtain meaningful results.

When an intrusion rule or rule argument requires a disabled preprocessor, the system automatically uses it with its current configuration even though it remains disabled in the network analysis policy's web interface.

Note that preprocessors do not generate events in most cases unless you enable the accompanying preprocessor rules in an intrusion policy.

License Requirements for Application Layer Preprocessors

FTD License

Threat

Classic License

Protection

Requirements and Prerequisites for Application Layer Preprocessors

Model Support

Any.

Supported Domains

Any

User Roles

- Admin
- Intrusion Admin

The DCE/RPC Preprocessor

The DCE/RPC protocol allows processes on separate network hosts to communicate as if the processes were on the same host. These inter-process communications are commonly transported between hosts over TCP and UDP. Within the TCP transport, DCE/RPC might also be further encapsulated in the Windows Server Message Block (SMB) protocol or in Samba, an open-source SMB implementation used for inter-process communication in a mixed environment comprised of Windows and UNIX- or Linux-like operating systems. In addition, Windows IIS web servers on your network might use IIS RPC over HTTP, which provides distributed communication through a firewall, to proxy TCP-transported DCE/RPC traffic.

Note that descriptions of DCE/RPC preprocessor options and functionality include the Microsoft implementation of DCE/RPC known as MSRPC; descriptions of SMB options and functionality refer to both SMB and Samba.

Although most DCE/RPC exploits occur in DCE/RPC client requests targeted for DCE/RPC servers, which could be practically any host on your network that is running Windows or Samba, exploits can also occur in server responses. The DCE/RPC preprocessor detects DCE/RPC requests and responses encapsulated in TCP, UDP, and SMB transports, including TCP-transported DCE/RPC using version 1 RPC over HTTP. The preprocessor analyzes DCE/RPC data streams and detects anomalous behavior and evasion techniques in DCE/RPC traffic. It also analyzes SMB data streams and detects anomalous SMB behavior and evasion techniques.

The DCE/RPC preprocessor also desegments SMB and defragments DCE/RPC in addition to the IP defragmentation provided by the IP defragmentation preprocessor and the TCP stream reassembly provided by the TCP stream preprocessor.

Finally, the DCE/RPC preprocessor normalizes DCE/RPC traffic for processing by the rules engine.

Connectionless and Connection-Oriented DCE/RPC Traffic

DCE/RPC messages comply with one of two distinct DCE/RPC Protocol Data Unit (PDU) protocols:

connection-oriented DCE/RPC PDU protocol

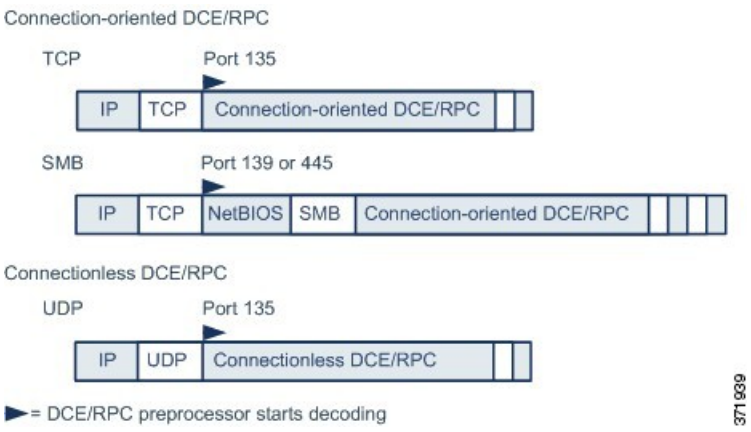
The DCE/RPC preprocessor detects connection-oriented DCE/RPC in the TCP, SMB, and RPC over HTTP transports.

connectionless DCE/RPC PDU protocol

The DCE/RPC preprocessor detects connectionless DCE/RPC in the UDP transport.

The two DCE/RPC PDU protocols have their own unique headers and data characteristics. For example, the connection-oriented DCE/RPC header length is typically 24 bytes and the connectionless DCE/RPC header length is fixed at 80 bytes. Also, correct fragment order of fragmented connectionless DCE/RPC cannot be handled by a connectionless transport and, instead, must be ensured by connectionless DCE/RPC header values; in contrast, the transport protocol ensures correct fragment order for connection-oriented DCE/RPC. The DCE/RPC preprocessor uses these and other protocol-specific characteristics to monitor both protocols for anomalies and other evasion techniques, and to decode and defragment traffic before passing it to the rules engine.

The following diagram illustrates the point at which the DCE/RPC preprocessor begins processing DCE/RPC traffic for the different transports.



Note the following in the figure:

- The well-known TCP or UDP port 135 identifies DCE/RPC traffic in the TCP and UDP transports.
- The figure does not include RPC over HTTP.

  For RPC over HTTP, connection-oriented DCE/RPC is transported directly over TCP as shown in the figure after an initial setup sequence over HTTP.
- The DCE/RPC preprocessor typically receives SMB traffic on the well-known TCP port 139 for the NetBIOS Session Service or the similarly implemented well-known Windows port 445.

Because SMB has many functions other than transporting DCE/RPC, the preprocessor first tests whether the SMB traffic is carrying DCE/RPC traffic and stops processing if it is not or continues processing if it is.

- IP encapsulates all DCE/RPC transports.
- TCP transports all connection-oriented DCE/RPC.
- UDP transports connectionless DCE/RPC.

DCE/RPC Target-Based Policies

Windows and Samba DCE/RPC implementations differ significantly. For example, all versions of Windows use the DCE/RPC context ID in the first fragment when defragmenting DCE/RPC traffic, and all versions of Samba use the context ID in the last fragment. As another example, Windows Vista uses the opnum (operation number) header field in the first fragment to identify a specific function call, and Samba and all other Windows versions use the opnum field in the last fragment.

There are also significant differences in Windows and Samba SMB implementations. For example, Windows recognizes the SMB OPEN and READ commands when working with named pipes, but Samba does not recognize these commands.

When you enable the DCE/RPC preprocessor, you automatically enable a default target-based policy. Optionally, you can add target-based policies that target other hosts running different Windows or Samba versions. The default target-based policy applies to any host not included in another target-based policy.

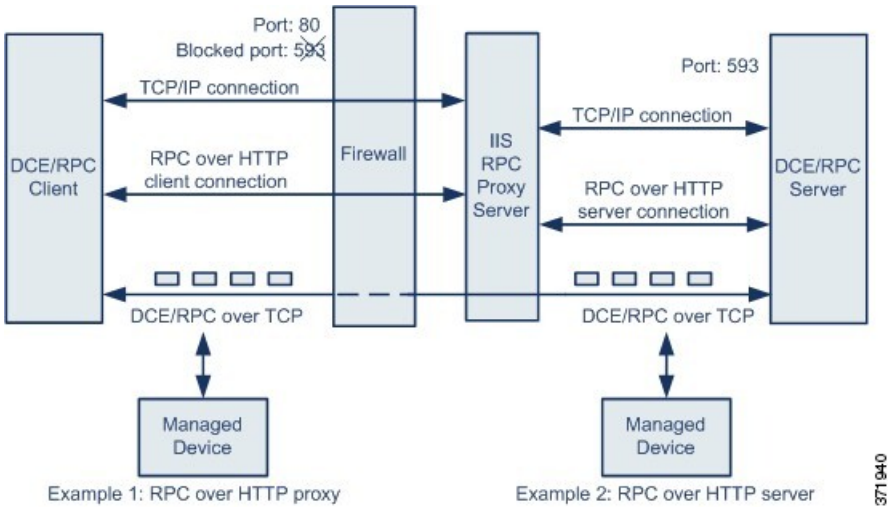
In each target-based policy, you can:

- enable one or more transports and specify *detection ports* for each
- enable and specify *auto-detection ports*
- set the preprocessor to detect when there is an attempt to connect to one or more shared SMB resources that you identify
- configure the preprocessor to detect files in SMB traffic and to inspect a specified number of bytes in a detected file
- modify an advanced option that should be modified only by a user with SMB protocol expertise; this option lets you set the preprocessor to detect when a number of chained SMB AndX commands exceed a specified maximum number

In addition to enabling SMB traffic file detection in the DCE/RPC preprocessor, you can configure a file policy to optionally capture and block these files, or submit them to the Cisco AMP cloud for dynamic analysis. Within that policy, you must create a file rule with an **Action** of **Detect Files** or **Block Files** and a selected **Application Protocol** of **Any** or **NetBIOS-ssn (SMB)**.

RPC over HTTP Transport

Microsoft RPC over HTTP allows you to tunnel DCE/RPC traffic through a firewall as shown in the following diagram. The DCE/RPC preprocessor detects version 1 of Microsoft RPC over HTTP.



The Microsoft IIS proxy server and the DCE/RPC server can be on the same host or on different hosts. Separate proxy and server options provide for both cases. Note the following in the figure:

- The DCE/RPC server monitors port 593 for DCE/RPC client traffic, but the firewall blocks port 593.

  Firewalls typically block port 593 by default.
- RPC over HTTP transports DCE/RPC over HTTP using well-known HTTP port 80, which firewalls are likely to permit.
- Example 1 shows that you would choose the **RPC over HTTP proxy** option to monitor traffic between the DCE/RPC client and the Microsoft IIS RPC proxy server.
- Example 2 shows that you would choose the **RPC over HTTP server** option when the Microsoft IIS RPC proxy server and the DCE/RPC server are located on different hosts and the device monitors traffic between the two servers.
- Traffic is comprised solely of connection-oriented DCE/RPC over TCP after RPC over HTTP completes the proxied setup between the DCE/RPC client and server.

DCE/RPC Global Options

Global DCE/RPC preprocessor options control how the preprocessor functions. Note that, except for the **Memory Cap Reached** and **Auto-Detect Policy on SMB Session** options, modifying these options could have a negative impact on performance or detection capability. You should not modify them unless you have a thorough understanding of the preprocessor and the interaction between the preprocessor and enabled DCE/RPC rules.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Maximum Fragment Size

When **Enable Defragmentation** is selected, specifies the maximum DCE/RPC fragment length allowed. The preprocessor truncates larger fragments for processing purposes to the specified size before defragmenting but does not alter the actual packet. A blank field disables this option.

Make sure that the **Maximum Fragment Size** option is greater than or equal to the depth to which the rules need to detect.

Reassembly Threshold

When **Enable Defragmentation** is selected, 0 disables this option, or specifies a minimum number of fragmented DCE/RPC bytes and, if applicable, segmented SMB bytes to queue before sending a reassembled packet to the rules engine. A low value increases the likelihood of early detection but could have a negative impact on performance. You should test for performance impact if you enable this option.

Make sure that the **Reassembly Threshold** option is greater than or equal to the depth to which the rules need to detect.

Enable Defragmentation

Specifies whether to defragment fragmented DCE/RPC traffic. When disabled, the preprocessor still detects anomalies and sends DCE/RPC data to the rules engine, but at the risk of missing exploits in fragmented DCE/RPC data.

Although this option provides the flexibility of not defragmenting DCE/RPC traffic, most DCE/RPC exploits attempt to take advantage of fragmentation to hide the exploit. Disabling this option would bypass most known exploits, resulting in a large number of false negatives.

Memory Cap Reached

Detects when the maximum memory limit allocated to the preprocessor is reached or exceeded. When the maximum memory cap is reached or exceeded, the preprocessor frees all pending data associated with the session that caused the memory cap event and ignores the rest of that session.

You can enable rule 133:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Auto-Detect Policy on SMB Session

Detects the Windows or Samba version that is identified in SMB `Session Setup AndX` requests and responses. When the detected version is different from the Windows or Samba version configured for the **Policy** configuration option, the detected version overrides the configured version for that session only.

For example, if you set **Policy** to Windows XP and the preprocessor detects Windows Vista, the preprocessor uses a Windows Vista policy for that session. Other settings remain in effect.

When the DCE/RPC transport is not SMB (that is, when the transport is TCP or UDP), the version cannot be detected and the policy cannot be automatically configured.

To enable this option, choose one of the following from the drop-down list:

- Choose **Client** to inspect server-to-client traffic for the policy type.
- Choose **Server** to inspect client-to-server traffic for the policy type.
- Choose **Both** to inspect server-to-client and client-to-server traffic for the policy type.

Legacy SMB Inspection Mode

Specifies which SMB versions to inspect. When **Legacy SMB Inspection Mode** is enabled, the DCE/RPC preprocessor inspects only SMB Version 1 traffic. When this option is disabled, the DCE/RPC preprocessor inspects traffic that uses SMB Versions 1, 2, and 3.

Related Topics

Basic content and protected_content Keyword Arguments
Overview: The byte_jump and byte_test Keywords

DCE/RPC Target-Based Policy Options

In each target-based policy, you can enable one or more of the TCP, UDP, SMB, and RPC over HTTP transports. When you enable a transport, you must also specify one or more *detection ports*, that is, ports that are known to carry DCE/RPC traffic.

Cisco recommends that you use the default detection ports, which are either well-known ports or otherwise commonly-used ports for each protocol. You would add detection ports only if you detected DCE/RPC traffic on a non-default port.

You can specify ports for one or more transports in any combination in a Windows target-based policy to match the traffic on your network, but you can only specify ports for the SMB transport in a Samba target-based policy.

Note You must enable at least one DCE/RPC transport in the default target-based policy except when you have added a DCE/RPC target-based policy that has at least one transport enabled. For example, you might want to specify the hosts for all DCE/RPC implementations and not have the default target-based policy deploy to unspecified hosts, in which case you would not enable a transport for the default target-based policy.

Optionally, you can also enable and specify *auto-detection ports*, that is, ports that the preprocessor tests first to determine if they carry DCE/RPC traffic and continues processing only when it detects DCE/RPC traffic.

When you enable auto-detection ports, ensure that they are set to the port range from 1024 to 65535 to cover the entire ephemeral port range.

Note that auto-detection occurs only for ports not already identified by transport detection ports.

It is unlikely that you would enable or specify auto-detection ports for the RPC over HTTP Proxy Auto-Detect Ports option or the SMB Auto-Detect Ports option because there is little likelihood that traffic for either would occur or even be possible except on the specified default detection ports.

Each target-based policy allows you to specify the various options below. If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Networks

The host IP addresses where you want to deploy the DCE/RPC target-based server policy. Also named the **Server Address** field in the Add Target pop-up window when you add a target-based policy.

You can specify a single IP address or address block, or a comma-separated list of either or both. You can configure up to 255 total profiles including the default policy.

Note The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or CIDR block/prefix length for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, 0.0.0.0/0 or ::/0).

Policy

The Windows or Samba DCE/RPC implementation used by the targeted host or hosts on your monitored network segment.

Note that you can enable the **Auto-Detect Policy on SMB Session** global option to automatically override the setting for this option on a per session basis when SMB is the DCE/RPC transport.

SMB Invalid Shares

Identifies one or more SMB shared resources the preprocessor will detect when there is an attempt to connect to a shared resource that you specify. You can specify multiple shares in a comma-separated list and, optionally, you can enclose shares in quotes, which was required in previous software versions but is no longer required; for example:

```
"C$", D$, "admin", private
```

The preprocessor detects invalid shares in SMB traffic when you have enabled **SMB Ports**.

Note that in most cases you should append a dollar sign to a drive named by Windows that you identify as an invalid share. For example, identify drive C as C$ or "C$".

Note also that to detect SMB invalid shares, you must also enable **SMB Ports** or **SMB Auto-Detect Ports.**

You can enable rule 133:26 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

SMB Maximum AndX Chain

The maximum number of chained SMB AndX commands to permit. Typically, more than a few chained AndX commands represent anomalous behavior and could indicate an evasion attempt. Specify 1 to permit no chained commands or 0 to disable detecting the number of chained commands.

Note that the preprocessor first counts the number of chained commands and generates an event if accompanying SMB preprocessor rules are enabled and the number of chained commands equals or exceeds the configured value. It then continues processing.

Caution Only someone who is expert in the SMB protocol should modify the setting for the **SMB Maximum AndX Chains** option.

You can enable rule 133:20 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

RPC proxy traffic only

Enabling **RPC over HTTP Proxy Ports** indicates whether detected client-side RPC over HTTP traffic is proxy traffic only or might include other web server traffic. For example, port 80 could carry both proxy and other web server traffic.

When this option is disabled, both proxy and other web server traffic are expected. Enable this option, for example, if the server is a dedicated proxy server. When enabled, the preprocessor tests traffic to determine if it carries DCE/RPC, ignores the traffic if it does not, and continues processing if it does. Note that enabling this option adds functionality only if the **RPC over HTTP Proxy Ports** check box is also enabled.

RPC over HTTP Proxy Ports

Enables detection of DCE/RPC traffic tunneled by RPC over HTTP over each specified port when your managed device is positioned between the DCE/RPC client and the Microsoft IIS RPC proxy server.

When enabled, you can add any ports where you see DCE/RPC traffic, although this is unlikely to be necessary because web servers typically use the default port for both DCE/RPC and other traffic. When enabled, you would not enable **RPC over HTTP Proxy Auto-Detect Ports**, but you would enable the **RPC Proxy Traffic Only** when detected client-side RPC over HTTP traffic is proxy traffic only and does not include other web server traffic.

Note You would rarely, if ever, select this option.

RPC over HTTP Server Ports

Enables detection of DCE/RPC traffic tunneled by RPC over HTTP on each specified port when the Microsoft IIS RPC proxy server and the DCE/RPC server are located on different hosts and the device monitors traffic between the two servers.

Typically, when you enable this option you should also enable **RPC over HTTP Server Auto-Detect Ports** with a port range from 1025 to 65535 for that option even if you are not aware of any proxy web servers on your network. Note that the RPC over HTTP server port is sometimes reconfigured, in which case you should add the reconfigured server port to port list for this option.

TCP Ports

Enables detection of DCE/RPC traffic in TCP on each specified port.

Legitimate DCE/RPC traffic and exploits might use a wide variety of ports, and other ports above port 1024 are common. Typically, when this option is enabled you should also enable **TCP Auto-Detect Ports** with a port range from 1025 to 65535 for that option.

UDP Ports

Enables detection of DCE/RPC traffic in UDP on each specified port.

Legitimate DCE/RPC traffic and exploits might use a wide variety of ports, and other ports above port 1024 are common. Typically, when this option is enabled you should also enable **UDP Auto-Detect Ports** with a port range from 1025 to 65535 for that option.

SMB Ports

Enables detection of DCE/RPC traffic in SMB on each specified port.

You could encounter SMB traffic using the default detection ports. Other ports are rare. Typically, use the default settings.

Note that you can enable the **Auto-Detect Policy on SMB Session** global option to automatically override the policy type configured for a targeted policy on a per session basis when SMB is the DCE/RPC transport.

RPC over HTTP Proxy Auto-Detect Ports

Enables auto-detection of DCE/RPC traffic tunneled by RPC over HTTP on the specified ports when your managed device is positioned between the DCE/RPC client and the Microsoft IIS RPC proxy server.

When enabled, you would typically specify a port range from 1025 to 65535 to cover the entire range of ephemeral ports.

RPC over HTTP Server Auto-Detect Ports

Enables auto-detection of DCE/RPC traffic tunneled by RPC over HTTP on the specified ports when the Microsoft IIS RPC proxy server and the DCE/RPC server are located on different hosts and the device monitors traffic between the two servers.

TCP Auto-Detect Ports

Enables auto-detection of DCE/RPC traffic in TCP on the specified ports.

UDP Auto-Detect Ports

Enables auto-detection of DCE/RPC traffic in UDP on each specified port.

SMB Auto-Detect Ports

Enables auto-detection of DCE/RPC traffic in SMB.

Note You would rarely, if ever, select this option.

SMB File Inspection

Enables inspection of SMB traffic for file detection. You have the following options:

- Select **Off** to disable file inspection.
- Select **Only** to inspect file data without inspecting the DCE/RPC traffic in SMB. Selecting this option can improve performance over inspecting both files and DCE/RPC traffic.
- Select **On** to inspect both files and the DCE/RPC traffic in SMB. Selecting this option can impact performance.

Inspection of SMB traffic for the following is not supported:

- files transferred concurrently in a single TCP or SMB session
- files transferred across multiple TCP or SMB sessions

- files transferred with non-contiguous data, such as when message signing is negotiated
- files transferred with different data at the same offset, overlapping the data
- files opened on a remote client for editing that the client saves to the file server

SMB File Inspection Depth

If **SMB File Inspection** is set to **Only** or **On**, the number of bytes inspected when a file is detected in SMB traffic. Specify one of the following:

- a positive value
- `0` to inspect the entire file
- `-1` to disable file inspection

Enter a value in this field equal to or smaller than the one defined in the File and Malware Settings section of the Advanced tab in your access control policy. If you set a value for this option larger than the one defined for **Limit the number of bytes inspected when doing file type detection**, the system uses the access control policy setting as the functional maximum.

If **SMB File Inspection** is set to **Off**, this field is disabled.

Related Topics

Firepower System IP Address Conventions

Traffic-Associated DCE/RPC Rules

Most DCE/RPC preprocessor rules trigger against anomalies and evasion techniques detected in SMB, connection-oriented DCE/RPC, or connectionless DCE/RPC traffic. The following table identifies the rules that you can enable for each type of traffic.

Table 1: Traffic-Associated DCE/RPC Rules

Traffic	Preprocessor Rule GID:SID
SMB	133:2 through 133:26, and 133:48 through 133:59
Connection-Oriented DCE/RPC	133:27 through 133:39
Detect Connectionless DCE/RPC	133:40 through 133:43

Configuring the DCE/RPC Preprocessor

You configure the DCE/RPC preprocessor by modifying any of the global options that control how the preprocessor functions, and by specifying one or more target-based server policies that identify the DCE/RPC servers on your network by IP address and by either the Windows or Samba version running on them. Target-based policy configuration also includes enabling transport protocols, specifying the ports carrying DCE/RPC traffic to those hosts, and setting other server-specific options.

The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Before you begin

- Confirm that networks you want to identify in a custom target-based policy match or are a subset of the networks, zones, and VLANs handled by its parent network analysis policy. See Advanced Settings for Network Analysis Policies for more information.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel on the left.

Step 4 If **DCE/RPC Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **DCE/RPC Configuration**.

Step 6 Modify the options in the **Global Settings** section; see DCE/RPC Global Options, on page 5.

Step 7 You have the following choices:

- Add a server profile — Click **Add** ( ) next to **Servers**. Specify one or more IP addresses in the **Server Address** field, then click **OK**.
- Delete a server profile — Click **Delete** ( ) next to the policy.
- Edit a server profile — Click the configured address for the profile under **Servers**, or click **default**. You can modify any of the settings in the **Configuration** section; see DCE/RPC Target-Based Policy Options, on page 7.

Step 8 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate intrusion events, enable DCE/RPC preprocessor rules (GID 132 or 133). For more information, see Setting Intrusion Rule States, DCE/RPC Global Options, on page 5, DCE/RPC Target-Based Policy Options, on page 7, and Traffic-Associated DCE/RPC Rules, on page 11.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Firepower System IP Address Conventions

File and Malware Inspection Performance and Storage Options
DCE/RPC Keywords
Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

The DNS Preprocessor

The DNS preprocessor inspects DNS name server responses for the following specific exploits:

- Overflow attempts on RData text fields
- Obsolete DNS resource record types
- Experimental DNS resource record types

The most common type of DNS name server response provides one or more IP addresses that correspond to domain names in the query that prompted the response. Other types of server responses provide, for example, the destination of an email message or the location of a name server that can provide information not available from the server originally queried.

A DNS response is comprised of:

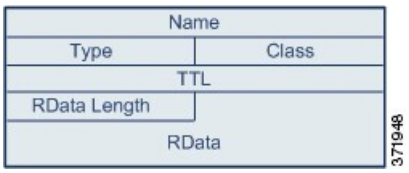
- a message header
- a Question section that contains one or more requests
- three sections that respond to requests in the Question section
  - Answer
  - Authority
  - Additional Information.

Responses in these three sections reflect the information in *resource records* (RR) maintained on the name server. The following table describes these three sections.

Table 2: DNS Name Server RR Responses

This section...	Includes...	For example...
Answer	Optionally, one or more resource records that provide a specific answer to a query	The IP address corresponding to a domain name
Authority	Optionally, one or more resource records that point to an authoritative name server	The name of an authoritative name server for the response
Additional Information	Optionally, one or more resource records that provided additional information related to the Answer sections	The IP address of another server to query

There are many types of resource records, all adhering to the following structure:



Theoretically, any type of resource record can be used in the Answer, Authority, or Additional Information section of a name server response message. The DNS preprocessor inspects any resource record in each of the three response sections for the exploits it detects.

The Type and RData resource record fields are of particular importance to the DNS preprocessor. The Type field identifies the type of resource record. The RData (resource data) field provides the response content. The size and content of the RData field differ depending on the type of resource record.

DNS messages typically use the UDP transport protocol but also use TCP when the message type requires reliable delivery or the message size exceeds UDP capabilities. The DNS preprocessor inspects DNS server responses in both UDP and TCP traffic.

The DNS preprocessor does not inspect TCP sessions picked up in midstream, and ceases inspection if a session loses state because of dropped packets.

DNS Preprocessor Options

Ports

This field specifies the source port or ports the DNS preprocessor should monitor for DNS server responses. Separate multiple ports with commas.

The typical port to configure for the DNS preprocessor is well-known port 53, which DNS name servers use for DNS messages in both UDP and TCP.

Detect Overflow attempts on RData Text fields

When the resource record type is TXT (text), the RData field is a variable-length ASCII text field.

When selected, this option detects a specific vulnerability identified by entry CVE-2006-3441 in MITRE's Current Vulnerabilities and Exposures database. This is a known vulnerability in Microsoft Windows 2000 Service Pack 4, Windows XP Service Pack 1 and Service Pack 2, and Windows Server 2003 Service Pack 1. An attacker can exploit this vulnerability and take complete control of a host by sending or otherwise causing the host to receive a maliciously crafted name server response that causes a miscalculation in the length of an RData text field, resulting in a buffer overflow.

You should enable this option when your network might include hosts running operating systems that have not been upgraded to correct this vulnerability.

You can enable rule 131:3 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect Obsolete DNS RR Types

RFC 1035 identifies several resource record types as obsolete. Because these are obsolete record types, some systems do not account for them and may be open to exploits. You would not expect to encounter these record types in normal DNS responses unless you have purposely configured your network to include them.

You can configure the system to detect known obsolete resource record types. The following table lists and describes these record types.

Table 3: Obsolete DNS Resource Record Types

RR Type	Code	Description
3	MD	a mail destination
4	MF	a mail forwarder

You can enable rule 131:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detecting Experimental DNS RR Types

RFC 1035 identifies several resource record types as experimental. Because these are experimental record types, some systems do not account for them and may be open to exploits. You would not expect to encounter these record types in normal DNS responses unless you have purposely configured your network to include them.

You can configure the system to detect known experimental resource record types. The following table lists and describes these record types.

Table 4: Experimental DNS Resource Record Types

RR Type	Code	Description
7	MB	a mailbox domain name
8	MG	a mail group member
9	MR	a mail rename domain name
10	NUL	a null resource record

You can enable rule 131:2 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Configuring the DNS Preprocessor

In a multidomain deployment, the system displays policies created in the current domain, which you can edit. It also displays policies created in ancestor domains, which you cannot edit. To view and edit policies created in a lower domain, switch to that domain.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **DNS Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **DNS Configuration**.

Step 6 Modify the settings as described in DNS Preprocessor Options, on page 14.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate intrusion events, enable DNS preprocessor rules (GID 131). For more information, see Setting Intrusion Rule States and DNS Preprocessor Options, on page 14.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Layers in Intrusion and Network Analysis Policies
Conflicts and Changes: Network Analysis and Intrusion Policies

The FTP/Telnet Decoder

The FTP/Telnet decoder analyzes FTP and telnet data streams, normalizing FTP and telnet commands before processing by the rules engine.

Global FTP and Telnet Options

You can set global options to determine whether the FTP/Telnet decoder performs stateful or stateless inspection of packets, whether the decoder detects encrypted FTP or telnet sessions, and whether the decoder continues to check a data stream after it encounters encrypted data.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Stateful Inspection

When selected, causes the FTP/Telnet decoder to save state and provide session context for individual packets and only inspect reassembled sessions. When cleared, analyzes each individual packet without session context.

To check for FTP data transfers, this option must be selected.

Detect Encrypted Traffic

Detects encrypted telnet and FTP sessions.

You can enable rules 125:7 and 126:2 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Continue to Inspect Encrypted Data

Instructs the preprocessor to continue checking a data stream after it is encrypted, looking for eventual decrypted data that can be processed.

Telnet Options

You can enable or disable normalization of telnet commands by the FTP/Telnet decoder, enable or disable a specific anomaly case, and set the threshold number of Are You There (AYT) attacks to permit.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Ports

Indicates the ports whose telnet traffic you want to normalize. Telnet typically connects to TCP port 23. In the interface, list multiple ports separated by commas.

Caution Because encrypted traffic (SSL) cannot be decoded, adding port 22 (SSH) could yield unexpected results.

Normalize

Normalizes telnet traffic to the specified ports.

Detect Anomalies

Enables detection of Telnet SB (subnegotiation begin) without the corresponding SE (subnegotiation end).

Telnet supports subnegotiation, which begins with SB (subnegotiation begin) and must end with an SE (subnegotiation end). However, certain implementations of Telnet servers will ignore the SB without a corresponding SE. This is anomalous behavior that could be an evasion case. Because FTP uses the Telnet protocol on the control connection, it is also susceptible to this behavior.

You can enable rule 126:3 to generate an event and, in an inline deployment, drop offending packets when this anomaly is detected in Telnet traffic, and rule 125:9 when it is detected on the FTP command channel. See Setting Intrusion Rule States.

Are You There Attack Threshold Number

Detects when the number of consecutive AYT commands exceeds the specified threshold. Cisco recommends that you set the AYT threshold to a value no higher than the default value.

You can enable rule 126:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Server-Level FTP Options

You can set options for decoding on multiple FTP servers. Each server profile you create contains the server IP address and the ports on the server where traffic should be monitored. You can specify which FTP commands to validate and which to ignore for a particular server, and set maximum parameter lengths for commands. You can also set the specific command syntax the decoder should validate against for particular commands and set alternate maximum command parameter lengths.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Networks

Use this option to specify one or more IP addresses of FTP servers.

You can specify a single IP address or address block, or a comma-separated list comprised of either or both. You can configure up to 1024 characters, and you can specify up to 255 profiles including the default profile.

Note The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or address block for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, 0.0.0.0/0 or ::/0).

Ports

Use this option to specify the ports on the FTP server where the managed device should monitor traffic. In the interface, list multiple ports separated by commas. Port 21 is the well-known port for FTP traffic.

File Get Commands

Use this option to define the FTP commands used to transfer files from server to client. Do not change these values unless directed to do so by Support.

Caution Do not modify the **File Get Commands** field unless directed to by Support.

File Put Commands

Use this option to define the FTP commands used to transfer files from client to server. Do not change these values unless directed to do so by Support.

Caution Do not modify the **File Put Commands** field unless directed to by Support.

Additional FTP Commands

Use this line to specify the additional commands that the decoder should detect. Separate additional commands by spaces.

Additional commands you may want to add include `XPWD`, `XCWD`, `XCUP`, `XMKD`, and `XRMD`. For more information on these commands, see RFC 775, the Directory oriented FTP commands specification by the Network Working Group.

Default Max Parameter Length

Use this option to detect the maximum parameter length for commands where an alternate maximum parameter length has not been set. You can add as many alternative maximum parameter lengths as needed.

You can enable rule 125:3 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Alternate Max Parameter Length

Use this option to specify commands where you want to detect a different maximum parameter length, and to specify the maximum parameter length for those commands. Click **Add** to add lines where you can specify a different maximum parameter length to detect for particular commands.

Check Commands for String Format Attacks

Use this option to check the specified commands for string format attacks.

You can enable rule 125:5 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Command Validity

Use this option to enter a valid format for a specific command. Click **Add** to add a command validation line.

You can enable rules 125:2 and 125:4 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Ignore FTP Transfers

Use this option to improve performance on FTP data transfers by disabling all inspection other than state inspection on the data transfer channel.

Note To inspect data transfers, the global FTP/Telnet **Stateful Inspection** option must be selected.

Detect Telnet Escape Codes within FTP Commands

Use this option to detect when telnet commands are used over the FTP command channel.

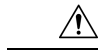
You can enable rule 125:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Ignore Erase Commands during Normalization

When **Detect Telnet Escape Codes within FTP Commands** is selected, use this option to ignore telnet character and line erase commands when normalizing FTP traffic. The setting should match how the FTP server handles telnet erase commands. Note that newer FTP servers typically ignore telnet erase commands, while older servers typically process them.

Troubleshooting Option: Log FTP Command Validation Configuration

Support might ask you during a troubleshooting call to configure your system to print the configuration information for each FTP command listed for the server.



Caution Do not enable **Log FTP Command Validation Configuration** unless instructed to do so by Support.

FTP Command Validation Statements

When setting up a validation statement for an FTP command, you can specify a group of alternative parameters by separating the parameters with spaces. You can also create a binary OR relationship between two parameters by separating them with a pipe character (`|`) in the validation statement. Surrounding parameters by square brackets (`[]`) indicates that those parameters are optional. Surrounding parameters with curly brackets (`{}`) indicates that those parameters are required.

You can create FTP command parameter validation statements to validate the syntax of a parameter received as part of an FTP communication.

Any of the parameters listed in the following table can be used in FTP command parameter validation statements.

Table 5: FTP Command Parameters

If you use...	The following validation occurs...
`int`	The represented parameter must be an integer.
`number`	The represented parameter must be an integer between 1 and 255.
`char` _chars	The represented parameter must be a single character and a member of the characters specified in the _chars argument. For example, defining the command validity for `MODE` with the validation statement `char` SBC checks that the parameter for the `MODE` command comprises the character `S` (representing Stream mode), the character `B` (representing Block mode), or the character `C` (representing Compressed mode).
`date` _datefmt	If _datefmt contains `#`, the represented parameter must be a number. If _datefmt contains `C`, the represented parameter must be a character. If _datefmt contains literal strings, the represented parameter must match the literal string.
`string`	The represented parameter must be a string.

If you use...	The following validation occurs...
`host_port`	The represented parameter must be a valid host port specifier as defined by RFC 959, the File Transfer Protocol specification by the Network Working Group.

You can combine the syntax in the table above as needed to create parameter validation statements that correctly validate each FTP command where you need to validate traffic.

Note When you include a complex expression in a TYPE command, surround it by spaces. Also, surround each operand within the expression by spaces. For example, type `char A | B` , not `char A|B`.

Related Topics

Server-Level FTP Options, on page 18
Firepower System IP Address Conventions
FTP Command Validation Statements, on page 20

Client-Level FTP Options

Use these options to configure custom FTP client profiles. If an option description does not include a preprocessor rule, the option is not associated with a preprocessor rule.

Networks

Use this option to specify one or more IP addresses of FTP clients.

You can specify a single IP address or address block, or a comma-separated list comprised of either or both. You can specify up to 1024 characters, and you can specify up to 255 profiles including the default profile.

Note The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or address block for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, 0.0.0.0/0 or ::/0).

Max Response Length

Use this option to specify the maximum allowed response length to an FTP command accepted by the client. This can detect basic buffer overflows.

You can enable rule 125:6 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect FTP Bounce Attempts

Use this option to detect FTP bounce attacks.

You can enable rule 125:8 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Allow FTP Bounce to

Use this option to configure a list of additional hosts and ports on those hosts on which FTP PORT commands should not be treated as FTP bounce attacks.

Detect Telnet Escape Codes within FTP Commands

Use this option to detect when telnet commands are used over the FTP command channel.

You can enable rule 125:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Ignore Erase Commands During Normalization

When **Detect Telnet Escape Codes within FTP Commands** is selected, use this option to ignore telnet character and line erase commands when normalizing FTP traffic.The setting should match how the FTP client handles telnet erase commands. Note that newer FTP clients typically ignore telnet erase commands, while older clients typically process them.

Related Topics

Firepower System IP Address Conventions

Configuring the FTP/Telnet Decoder

You can configure client profiles for FTP clients to monitor FTP traffic from clients.

The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Before you begin

- Confirm that any networks you want to identify in a custom target-based policy match or are a subset of the networks, zones, and VLANs handled by its parent network analysis policy. See Advanced Settings for Network Analysis Policies for more information.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **FTP and Telnet Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **FTP and Telnet Configuration**.

Step 6 Set options in the **Global Settings** section as described in Global FTP and Telnet Options, on page 16.

Step 7 Set options in the **Telnet Settings** section as described in Telnet Options, on page 17.

Step 8 Manage FTP server profiles:

- Add a server profile — Click **Add** ( ) next to **FTP Server**. Specify one or more IP addresses for the client in the **Server Address** field and click **OK**. You can specify a single IP address or address block, or a comma-separated list of either or both. You can specify up to 1024 characters, and you can configure up to 255 policies, including the default policy.
- Edit a server profile — Click the configured address for a custom profile under **FTP Server**, or click **default**. You can modify the settings in the **Configuration** section; see Server-Level FTP Options, on page 18.
- Delete a server profile — Click **Delete** ( ) next to the profile.

Step 9 Manage FTP client profiles:

- Add a client profile — Click **Add** ( ) next to **FTP Client**. Specify one or more IP addresses for the client in the **Client Address** field and click **OK**. You can specify a single IP address or address block, or a comma-separated list of either or both. You can specify up to 1024 characters, and you can configure up to 255 policies, including the default policy.
- Edit a client profile — Click the configured address for a profile you have added under **FTP Client**, or click **default**. You can modify the settings in the Configuration page area; see Client-Level FTP Options, on page 21.
- Delete a client profile — Click **Delete** ( ) next to a custom profile.

Step 10 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate intrusion events, enable FTP and telnet preprocessor rules (GID 125 and 126). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Firepower System IP Address Conventions
Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

The HTTP Inspect Preprocessor

The HTTP Inspect preprocessor is responsible for:

- decoding and normalizing HTTP requests sent to and HTTP responses received from web servers on your network
- separating messages sent to web servers into URI, non-cookie header, cookie header, method, and message body components to improve performance of HTTP-related intrusion rules
- separating messages received from web servers into status code, status message, non-set-cookie header, cookie header, and response body components to improve performance of HTTP-related intrusion rules
- detecting possible URI-encoding attacks
- making the normalized data available for additional rule processing

HTTP traffic can be encoded in a variety of formats, making it difficult for rules to appropriately inspect. HTTP Inspect decodes 14 types of encoding, ensuring that your HTTP traffic gets the best inspection possible.

You can configure HTTP Inspect options globally, on a single server, or for a list of servers.

Note that the preprocessor engine performs HTTP normalization *statelessly*. That is, it normalizes HTTP strings on a packet-by-packet basis, and can only process HTTP strings that have been reassembled by the TCP stream preprocessor.

Global HTTP Normalization Options

The global HTTP options provided for the HTTP Inspect preprocessor control how the preprocessor functions. Use these options to enable or disable HTTP normalization when ports not specified as web server ports receive HTTP traffic.

Note the following:

- If you enable **Unlimited Decompression**, the **Maximum Compressed Data Depth** and **Maximum Decompressed Data Depth** options are automatically set to 65535 when you commit your changes.
- The highest value is used when the values for **Maximum Compressed Data Depth** or **Maximum Decompressed Data Depth** are different in:
  - the default network analysis policy
  - any other custom network analysis policy invoked by network analysis rules in the same access control policy

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Detect Anomalous HTTP Servers

Detects HTTP traffic sent to or received by ports not specified as web server ports.

Note If you turn this option on, be sure to list all ports that do receive HTTP traffic in a server profile on the HTTP Configuration page. If you do not, and you enable this option and the accompanying preprocessor rule, normal traffic to and from the server will generate events. The default server profile contains all ports normally used for HTTP traffic, but if you modified that profile, you may need to add those ports to another profile to prevent events from being generated.

You can enable rule 120:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect HTTP Proxy Servers

Detects HTTP traffic using proxy servers not defined by the **Allow HTTP Proxy Use** option.

You can enable rule 119:17 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Maximum Compressed Data Depth

Sets the maximum size of compressed data to decompress when **Inspect Compressed Data** (and, optionally, **Decompress SWF File (LZMA)**, **Decompress SWF File (Deflate)**, or **Decompress PDF File (Deflate)**) is enabled.

Maximum Decompressed Data Depth

Sets the maximum size of the normalized decompressed data when **Inspect Compressed Data** (and, optionally, **Decompress SWF File (LZMA)**, **Decompress SWF File (Deflate)**, or **Decompress PDF File (Deflate)**) is enabled.

Server-Level HTTP Normalization Options

You can set server-level options for each server you monitor, globally for all servers, or for a list of servers. Additionally, you can use a predefined server profile to set these options, or you can set them individually to meet the needs of your environment. Use these options, or one of the default profiles that set these options, to specify the HTTP server ports whose traffic you want to normalize, the amount of server response payload you want to normalize, and the types of encoding you want to normalize.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Networks

Use this option to specify the IP address of one or more servers. You can specify a single IP address or address block, or a comma-separated list comprised of either or both.

In addition to a limit of up to 255 total profiles, including the default profile, you can include up to 496 characters, or approximately 26 entries, in an HTTP server list, and specify a total of 256 address entries for all server profiles.

Note The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Note that the `default` setting in the default policy specifies all IP addresses on your monitored network segment that are not covered by another target-based policy. Therefore, you cannot and do not need to specify an IP address or CIDR block/prefix length for the default policy, and you cannot leave this setting blank in another policy or use address notation to represent `any` (for example, 0.0.0.0/0 or ::/0).

Ports

The ports whose HTTP traffic the preprocessor engine normalizes. Separate multiple port numbers with commas.

Oversize Dir Length

Detects URL directories longer than the specified value.

You can enable rule 119:15 to generate events and, in an inline deployment, drop offending packets when the preprocessor detects a request for a URL that is longer than the specified length.

Client Flow Depth

Specifies the number of bytes for rules to inspect in raw HTTP packets, including header and payload data, in client-side HTTP traffic defined in **Ports**. Client flow depth does not apply when HTTP content rule options within a rule inspect specific parts of a request message.

Specify any of the following:

- A positive value inspects the specified number of bytes in the first packet. If the first packet contains fewer bytes than specified, inspect the entire packet. Note that the specified value applies to both segmented and reassembled packets.

  Note also that a value of 300 typically eliminates inspection of large HTTP Cookies that appear at the end of many client request headers.

- 0 inspects all client-side traffic, including multiple packets in a session and exceeding the upper byte limit if necessary. Note that this value is likely to affect performance.

- -1 ignores all client-side traffic.

Server Flow Depth

Specifies the number of bytes for rules to inspect in raw HTTP packets in server-side HTTP traffic specified by **Ports**. Inspection includes the raw header and payload when **Inspect HTTP Responses** disabled and only the raw response body when **Inspect HTTP Response** is enabled.

Server flow depth specifies the number of bytes of raw server response data in a session for rules to inspect in server-side HTTP traffic defined in **Ports**. You can use this option to balance performance and the level of inspection of HTTP server response data. Server flow depth does not apply when HTTP content options within a rule inspect specific parts of a response message.

Unlike client flow depth, server flow depth specifies the number of bytes per HTTP response, not per HTTP request packet, for rules to inspect.

You can specify any of the following:

- A positive value:

  When **Inspect HTTP Responses** is **enabled**, inspects only the raw HTTP response body, and not raw HTTP headers; also inspects decompressed data when **Inspect Compressed Data** is enabled.

  When **Inspect HTTP Responses** is **disabled**, inspects the raw packet header and payload.

  If the session includes fewer response bytes than specified, rules fully inspect all response packets in a given session, across multiple packets as needed. If the session includes more response bytes than

specified, rules inspect only the specified number of bytes for that session, across multiple packets as needed.

Note that a small flow depth value may cause false negatives from rules that target server-side traffic defined in **Ports**. Most of these rules target either the HTTP header or content that is likely to be in the first hundred or so bytes of non-header data. Headers are usually under 300 bytes long, but header size may vary.

Note also that the specified value applies to both segmented and reassembled packets.

- 0 inspects the entire packet for all HTTP server-side traffic defined in **Ports**, including response data in a session that exceeds 65535 bytes.

  Note that this value is likely to affect performance.

- -1:

  When **Inspect HTTP Responses** is **enabled**, inspects only raw HTTP headers and not the raw HTTP response body.

  When **Inspect HTTP Responses** is **disabled**, ignores all server-side traffic defined in **Ports.**

Maximum Header Length

Detects a header field longer than the specified maximum number of bytes in an HTTP request; also in HTTP responses when **Inspect HTTP Responses** is enabled. A value of 0 disables this option. Specify a positive value to enable it.

You can enable rule 119:19 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States..

Maximum Number of Headers

Detects when the number of headers exceeds this setting in an HTTP request. A value of 0 disables this option. Specify a positive value to enable it.

You can enable rule 119:20 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States..

Maximum Number of Spaces

Detects when the number of white spaces in a folded line equals or exceeds this setting in an HTTP request. A value of 0 disables this option. Specify a positive value to enable it.

You can enable rule 119:26 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States..

HTTP Client Body Extraction Depth

Specifies the number of bytes to extract from the message body of an HTTP client request. You can use an intrusion rule to inspect the extracted data by selecting the `content` or `protected_content` keyword **HTTP Client Body** option.

Specify -1 to ignore the client body. Specify 0 to extract the entire client body. Note that identifying specific bytes to extract can improve system performance. Note also that you must specify a value greater than or equal to 0 for the **HTTP Client Body** option to function in an intrusion rule.

Small Chunk Size

Specifies the maximum number of bytes at which a chunk is considered small. Specify a positive value. A value of 0 disables detection of anomalous consecutive small segments. See the **Consecutive Small Chunks** option for more information.

Consecutive Small Chunks

Specifies how many consecutive small chunks represent an abnormally large number in client or server traffic that uses chunked transfer encoding. The **Small Chunk Size** option specifies the maximum size of a small chunk.

For example, set **Small Chunk Size** to 10 and **Consecutive Small Chunks** to 5 to detect 5 consecutive chunks of 10 bytes or less.

You can enable preprocessor rule 119:27 to generate events and, in an inline deployment, drop offending packets on excessive small chunks in client traffic, and rule 120:7 in server traffic. When **Small Chunk Size** is enabled and this option is set to 0 or 1, enabling these rules would trigger an event on every chunk of the specified size or less.

HTTP Methods

Specifies HTTP request methods in addition to GET and POST that you expect the system to encounter in traffic. Use a comma to separate multiple values.

Intrusion rules use the `content` or `protected_content` keyword with the **HTTP Method** argument to search for content in HTTP methods. You can enable rule 119:31to generate events and, in an inline deployment, drop offending packets when a method other than GET, POST, or a method configured for this option is encountered in traffic. See Setting Intrusion Rule States.

No Alerts

Disables intrusion events when accompanying preprocessor rules are enabled.

Note This option does **not** disable HTTP standard text rules and shared object rules.

Normalize HTTP Headers

When **Inspect HTTP Responses** is enabled, enables normalization of non-cookie data in request and response headers. When **Inspect HTTP Responses** is **not** enabled, enables normalization of the entire HTTP header, including cookies, in request and response headers.

Inspect HTTP Cookies

Enables extraction of cookies from HTTP request headers. Also enables extraction of set-cookie data from response headers when **Inspect HTTP Responses** is enabled. Disabling this option when cookie extraction is not required can improve performance.

Note that the `Cookie:` and `Set-Cookie:` header names, leading spaces on the header line, and the `CRLF` that terminates the header line are inspected as part of the header and not as part of the cookie.

Normalize Cookies in HTTP headers

Enables normalization of cookies in HTTP request headers. When **Inspect HTTP Responses** is enabled, also enables normalization of set-cookie data in response headers. You must select **Inspect HTTP Cookies** before selecting this options.

Allow HTTP Proxy Use

Allows the monitored web server to be used as an HTTP proxy. This option is used only in the inspection of HTTP requests.

Inspect URI Only

Inspects only the URI portion of the normalized HTTP request packet.

Inspect HTTP Responses

Enables extended inspection of HTTP responses so, in addition to decoding and normalizing HTTP request messages, the preprocessor extracts response fields for inspection by the rules engine. Enabling this option causes the system to extract the response header, body, status code, and so on, and also extracts set-cookie data when **Inspect HTTP Cookies** is enabled.

You can enable rules 120:2 and 120:3 to generate events and, in an inline deployment, drop offending packets, as follows:

Table 6: Inspect HTTP Response Rules

This rule...	Triggers when...
120:2	an invalid HTTP response status code occurs.
120:3	an HTTP response does not include Content-Length or Transfer-Encoding.

Normalize UTF Encodings to UTF-8

When **Inspect HTTP Responses** is enabled, detects UTF-16LE, UTF-16BE, UTF-32LE, and UTF32-BE encodings in HTTP responses and normalizes them to UTF-8.

You can enable rule 120:4 to generate events and, in an inline deployment, drop offending packets when UTF normalization fails.

Inspect Compressed Data

When **Inspect HTTP Responses** is enabled, enables decompression of gzip and deflate-compatible compressed data in the HTTP response body, and inspection of the normalized decompressed data. The system inspects chunked and non-chunked HTTP response data. The system inspects decompressed data packet by packet across multiple packets as needed; that is, the system does not combine the decompressed data from different packets for inspection. Decompression ends when **Maximum Compressed Data Depth**, **Maximum Decompressed Data Depth**, or the end of the compressed data is reached. Inspection of decompressed data ends when **Server Flow Depth** is reached unless you also select **Unlimited Decompression**. You can use the `file_data` rule keyword to inspect decompressed data.

You can enable rules 120:6 and 120:24 to generate events and, in an inline deployment, drop offending packets, as follows:

Table 7: Inspect Compressed HTTP Response Rules

This rule...	Triggers when...
120:6	decompression of a compressed HTTP response fails.
120:24	partial decompression of a compressed HTTP response fails.

Unlimited Decompression

When **Inspect Compressed Data** (and, optionally, **Decompress SWF File (LZMA)**, **Decompress SWF File (Deflate)**, or **Decompress PDF File (Deflate)**) is enabled, overrides **Maximum Decompressed Data Depth** across multiple packets; that is, this option enables unlimited decompression across multiple packets. Note that enabling this option does not affect **Maximum Compressed Data Depth** or **Maximum Decompressed Data Depth** within a single packet. Note also that enabling this option sets **Maximum Compressed Data Depth** and **Maximum Decompressed Data Depth** to 65535 when you commit your changes.

Normalize Javascript

When **Inspect HTTP Responses** is enabled, enables detection and normalization of Javascript within the HTTP response body. The preprocessor normalizes obfuscated Javascript data such as the unescape and decodeURI functions and the String.fromCharCode method. The preprocessor normalizes the following encodings within the unescape, decodeURI, and decodeURIComponent functions:

- %XX
- %uXXXX
- 0xXX
- \xXX
- \uXXXX

The preprocessor detects consecutive white spaces and normalizes them into a single space. When this option is enabled, a configuration field allows you to specify the maximum number of consecutive white spaces to permit in obfuscated Javascript data. You can enter a value from 1 to 65535. The value 0 disables event generation, regardless of whether the preprocessor rule (120:10) associated with this field is enabled.

The preprocessor also normalizes the Javascript plus (+) operator and concatenates strings using the operator.

You can use the `file_data` intrusion rule keyword to point intrusion rules to the normalized Javascript data.

You can enable rules 120:9, 120:10, and 120:11 to generate events and, in an inline deployment, drop offending packets, as follows:

Table 8: Normalize Javascript Option Rules

This rule...	Triggers when...
120:9	the obfuscation level within the preprocessor is greater than or equal to 2.
120:10	the number of consecutive white spaces in the Javascript obfuscated data is greater than or equal to the value configured for the maximum number of consecutive white spaces allowed.

This rule...	Triggers when...
120:11	escaped or encoded data includes more than one type of encoding.

Decompress SWF File (LZMA) and Decompress SWF File (Deflate)

When **HTTP Inspect Responses** is enabled, these options decompress the compressed portions of files located within the HTTP response body of HTTP requests.

Note You can **only** decompress the compressed portions of files found in HTTP GET responses.

- **Decompress SWF File (LZMA)** decompresses the LZMA-compatible compressed portions of Adobe ShockWave Flash (`.swf`) files
- **Decompress SWF File (Deflate)** decompresses the deflate-compatible compressed portions of Adobe ShockWave Flash (`.swf`) files

Decompression ends when **Maximum Compressed Data Depth**, **Maximum Decompressed Data Depth**, or the end of the compressed data is reached. Inspection of decompressed data ends when **Server Flow Depth** is reached unless you also select **Unlimited Decompression**. You can use the `file_data` intrusion rule keyword to inspect decompressed data.

You can enable rules 120:12 and 120:13 to generate events and, in an inline deployment, drop offending packets, as follows:

Table 9: Decompress SWF File Option Rules

This rule...	Triggers when...
120:12	deflate file decompression fails.
120:13	LZMA file decompression fails.

Decompress PDF File (Deflate)

When **HTTP Inspect Responses** is enabled, **Decompress PDF File (Deflate)** decompresses the deflate-compatible compressed portions of Portable Document Format (`.pdf`) files located within the HTTP response body of HTTP requests. The system can only decompress PDF files with the `/FlateDecode` stream filter. Other stream filters (including `/FlateDecode /FlateDecode`) are unsupported.

Note You can **only** decompress the compressed portions of files found in HTTP GET responses.

Decompression ends when **Maximum Compressed Data Depth**, **Maximum Decompressed Data Depth**, or the end of the compressed data is reached. Inspection of decompressed data ends when **Server Flow Depth** is reached unless you also select **Unlimited Decompression**. You can use the `file_data` intrusion rule keyword to inspect decompressed data.

You can enable rules 120:14, 120:15, 120:16, and 120:17 to generate events and, in an inline deployment, drop offending packets, as follows:

Table 10: Decompress PDF File (Deflate) Option Rules

This rule...	Triggers when...
120:14	file decompression fails.
120:15	file decompression fails due to an unsupported compression type.
120:16	file decompression fails due to an unsupported PDF stream filter.
120:17	file parsing fails.

Extract Original Client IP Address

Enables the examination of original client IP addresses during intrusion inspection. The system extracts the original client IP address from the X-Forwarded-For (XFF), True-Client-IP, or custom HTTP headers you define in the **XFF Header Priority** option. You can view the extracted original client IP address in the intrusion events table.

You can enable rules 119:23, 119:29, and 119:30 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States..

XFF Header Priority

Specifies the order in which the system processes original client IP headers when multiple headers are present in an HTTP request. By default, the system examines X-Forwarded-For (XFF) headers, then True-Client-IP headers. Use the up and down arrow icons beside each header type to adjust its priority.

This option also allows you to specify original client IP headers other than XFF or True-Client-IP for extraction and evaluation. Click **Add** to add custom header names to the priority list. The system only supports custom headers that use the same syntax as an XFF or True-Client-IP header.

Keep in mind the following when configuring this option:

- The system uses this priority order when evaluating original client IP address headers for both access control and intrusion inspection.
- If multiple original client IP headers are present, the system processes only the header with the highest priority.
- The XFF header contains a list of IP addresses, which represent the proxy servers through which the request has passed. To prevent spoofing, the system uses the last IP address in the list (that is, the address appended by the trusted proxy) as the original client IP address.

Log URI

Enables extraction of the raw URI, if present, from HTTP request packets and associates the URI with all intrusion events generated for the session.

When this option is enabled, you can display the first fifty characters of the extracted URI in the HTTP URI column of the intrusion events table view. You can display the complete URI, up to 2048 bytes, in the packet view.

Log Hostname

Enables extraction of the host name, if present, from the HTTP request Host header and associates the host name with all intrusion events generated for the session. When multiple Host headers are present, extracts the host name from the first header.

When this option is enabled, you can display the first fifty characters of the extracted host name in the HTTP Hostname column of the intrusion events table view. You can display the complete host name, up to 256 bytes, in the packet view.

You can enable rule 119:25 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Note that, when enabled, rule 119:24 triggers if it detects multiple Host headers in an HTTP request, regardless of the setting for this option.

Profile

Specifies the types of encoding that are normalized for HTTP traffic. The system provides a default profile appropriate for most servers, default profiles for Apache servers and IIS servers, and custom default settings that you can tailor to meet the needs of your monitored traffic:

- Select **All** to use the standard default profile, appropriate for all servers.
- Select **IIS** to use the system-provided IIS profile.
- Select **Apache** to use the system-provided Apache profile.
- Select **Custom** to create your own server profile.

Server-Level HTTP Normalization Encoding Options

When you set the HTTP server-level **Profile** option to `Custom`, you can specify the types of encoding that are normalized for HTTP traffic, and enable HTTP preprocessor rules to generate events against traffic containing the different encoding types.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

ASCII Encoding

Decodes encoded ASCII characters and specifies whether the rules engine generates an event on ASCII-encoded URIs.

You can enable rule 119:1 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

UTF-8 Encoding

Decodes standard UTF-8 Unicode sequences in the URI.

You can enable rule 119:6 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Microsoft %U Encoding

Decodes the IIS %u encoding scheme that uses %u followed by four characters where the 4 characters are a hex encoded value that correlates to an IIS Unicode codepoint.

Tip Legitimate clients rarely use %u encodings, so Cisco recommends decoding HTTP traffic encoded with %u encodings.

You can enable rule 119:3 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Bare Byte UTF-8 Encoding

Decodes bare byte encoding, which uses non-ASCII characters as valid values in decoding UTF-8 values.

Tip Bare byte encoding allows the user to emulate an IIS server and interpret non-standard encodings correctly. Cisco recommends enabling this option because no legitimate clients encode UTF-8 this way.

You can enable rule 119:4 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Microsoft IIS Encoding

Decodes using Unicode codepoint mapping.

Tip Cisco recommends enabling this option, because it is seen mainly in attacks and evasion attempts.

You can enable rule 119:7 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Double Encoding

Decodes IIS double encoded traffic by making two passes through the request URI performing decodes in each one. Cisco recommends enabling this option because it is usually found only in attack scenarios.

You can enable rule 119:2 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Multi-Slash Obfuscation

Normalizes multiple slashes in a row into a single slash.

You can enable rule 119:8 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

IIS Backslash Obfuscation

Normalizes backslashes to forward slashes.

You can enable rule 119:9 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Directory Traversal

Normalizes directory traversals and self-referential directories. If you enable the accompanying preprocessor rules to generate events against this type of traffic, it may generate false positives because some web sites refer to files using directory traversals.

You can enable rules 119:10 and 119:11 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Tab Obfuscation

Normalizes the non-RFC standard of using a tab for a space delimiter. Apache and other non-IIS web servers use the tab character (0x09) as a delimiter in URLs.

Note Regardless of the configuration for this option, the HTTP Inspect preprocessor treats a tab as white space if a space character (0x20) precedes it.

You can enable rule 119:12 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Invalid RFC Delimiter

Normalizes line breaks (\n) in URI data.

You can enable rule 119:13 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Webroot Directory Traversal

Detects directory traversals that traverse past the initial directory in the URL.

You can enable rule 119:18 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Tab URI Delimiter

Turns on the use of the tab character (0x09) as a delimiter for a URI. Apache, newer versions of IIS, and some other web servers use the tab character as a delimiter in URLs.

Note Regardless of the configuration for this option, the HTTP Inspect preprocessor treats a tab as white space if a space character (0x20) precedes it.

Non-RFC characters

Detects the non-RFC character list you add in the corresponding field when it appears within incoming or outgoing URI data. When modifying this field, use the hexadecimal format that represents the byte character.

If and when you configure this option, set the value with care. Using a character that is very common may overwhelm you with events.

You can enable rule 119:14 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Max Chunk Encoding Size

Detects abnormally large chunk sizes in URI data.

You can enable rules 119:16 and 119:22 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Disable Pipeline Decoding

Disables HTTP decoding for pipelined requests. When this option is disabled, performance is enhanced because HTTP requests waiting in the pipeline are not decoded or analyzed, and are only inspected using generic pattern matching.

Non-Strict URI Parsing

Enables non-strict URI parsing. Use this option only on servers that will accept non-standard URIs in the format "GET /index.html abc xo qr \n". Using this option, the decoder assumes that the URI is between the first and second space, even if there is no valid HTTP identifier after the second space.

Extended ASCII Encoding

Enables parsing of extended ASCII characters in an HTTP request URI. Note that this option is available in custom server profiles only, and not in the default profiles provided for Apache, IIS, or all servers.

Related Topics

Overview: HTTP content and protected_content Keyword Arguments
Firepower System IP Address Conventions

Configuring The HTTP Inspect Preprocessor

The system builds a separate network map for each leaf domain. In a multidomain deployment, using literal IP addresses to constrain this configuration can have unexpected results. Using override-enabled objects allows descendant domain administrators to tailor Global configurations to their local environments.

Before you begin

- Confirm that any networks you want to identify in a custom target-based policy match or are a subset of the networks, zones, and VLANs handled by its parent network analysis policy. See Advanced Settings for Network Analysis Policies for more information.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** () next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **HTTP Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **HTTP Configuration**.

Step 6 Modify the options in the Global Settings page area; see Global HTTP Normalization Options, on page 24.

Step 7 You have three choices:

- Add a server profile — Click **Add** ( ) in the **Servers** section. Specify one or more IP addresses for the client in the **Server Address** field, and click **OK**. You can specify a single IP address or address block, or a comma-separated list of either or both. You can include up to 496 characters in a list, specify a total of 256 address entries for all server profiles, and create a total of 255 profiles including the default profile.
- Edit a server profile — Click the configured address for a profile you have added under **Servers**, or click **default**. You can modify any of the settings in the **Configuration** section; see Server-Level HTTP Normalization Options, on page 25. If you choose **Custom** for the **Profile** value, you can also modify the encoding options described in Server-Level HTTP Normalization Encoding Options, on page 33.
- Delete a server profile — Click **Delete** ( ) next to a custom profile.

Step 8 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want generate events and, in an inline deployment, drop offending packets, enable HTTP preprocessor rules (GID 119). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

Additional HTTP Inspect Preprocessor Rules

You can enable the rules in the **Preprocessor Rule GID:SID** column of the following table to generate events for HTTP Inspect preprocessor rules that are not associated with specific configuration options.

Table 11: Additional HTTP Inspect Preprocessor Rules

Preprocessor Rule GID:SID	Triggers when...
119:21	an HTTP request header has more than one `content-length` field.

Preprocessor Rule GID:SID	Triggers when...
119:24	an HTTP request has more than one Host header.
119:28	an HTTP POST method has neither a `content-length` header nor chunked `transfer-encoding`.
119:32	HTTP version 0.9 is encountered in traffic. Note that the TCP stream configuration must also be enabled.
119:33	an HTTP URI includes an unescaped space.
119:34	a TCP connection contains 24 or more pipelined HTTP requests.
120:5	UTF-7 encoding is encountered in HTTP response traffic; UTF-7 should only appear where 7-bit parity is required, such as in SMTP traffic.
120:8	the `content-length` or chunk size is invalid.
120:18	an HTTP server response occurs before the client request.
120:19	an HTTP response includes multiple content lengths.
120:20	an HTTP response includes multiple content encodings.
120:25	an HTTP response includse invalid header folding.
120:26	a junk line occurs before an HTTP response header.
120:27	an HTTP response does not include an end of header.
120:28	an invalid chunk size occurs, or chunk size is followed by junk characters.

The Sun RPC Preprocessor

Remote Procedure Call (RPC) normalization takes fragmented RPC records and normalizes them to a single record so the rules engine can inspect the complete record. For example, an attacker may attempt to discover the port where RPC `admind` runs. Some UNIX hosts use RPC `admind` to perform remote distributed system tasks. If the host performs weak authentication, a malicious user could take control of remote administration. The standard text rule (GID: 1) with the Snort ID (SID) 575 detects this attack by searching for content in specific locations to identify inappropriate `portmap GETPORT` requests.

Sun RPC Preprocessor Options

Ports

Specify the ports whose traffic you want to normalize. In the interface, list multiple ports separated by commas. Typical RPC ports are 111 and 32771. If your network sends RPC traffic to other ports, consider adding them.

Detect fragmented RPC records

Detects RPC fragmented records.

You can enable rules 106:1 and 106:5 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect multiple records in one packet

Detects more than one RPC request per packet (or reassembled packet).

You can enable rule 106:2 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect fragmented record sums which exceed one fragment

Detects reassembled fragment record lengths that exceed the current packet length.

You can enable rule 106:3 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Detect single fragment records which exceed the size of one packet

Detects partial records

You can enable rule 106:4 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Configuring the Sun RPC Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **Sun RPC Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **Sun RPC Configuration**.

Step 6 Modify the settings described in Sun RPC Preprocessor Options, on page 38.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable Sun RPC preprocessor rules (GID 106). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

The SIP Preprocessor

The Session Initiation Protocol (SIP) provides call setup, modification, and teardown of one or more sessions for one or more users of client applications such as Internet telephony, multimedia conferencing, instant messaging, online gaming, and file transfer. A *method* field in each SIP request identifies the purpose of the request, and a Request-URI specifies where to send the request. A status code in each SIP response indicates the outcome of the requested action.

After calls are set up using SIP, the Real-time Transport Protocol (RTP) is responsible for subsequent audio and video communication; this part of the session is sometimes referred to as the call channel, the data channel, or the audio/video data channel. RTP uses the Session Description Protocol (SDP) within the SIP message body for data-channel parameter negotiation, session announcement, and session invitation.

The SIP preprocessor is responsible for:

- decoding and analyzing SIP 2.0 traffic
- extracting the SIP header and message body, including SDP data when present, and passing the extracted data to the rules engine for further inspection
- generating events when the following conditions are detected and the corresponding preprocessor rules are enabled:
  - anomalies and known vulnerabilities in SIP packets
  - out-of-order and invalid call sequences
- optionally, ignoring the call channel

The preprocessor identifies the RTP channel based on the port identified in the SDP message, which is embedded in the SIP message body, but the preprocessor does not provide RTP protocol inspection.

Note the following when using the SIP preprocessor:

- UDP typically carries media sessions supported by SIP. UDP stream preprocessing provides SIP session tracking for the SIP preprocessor.
- SIP rule keywords allow you to point to the SIP packet header or message body and to limit detection to packets for specific SIP methods or status codes.

SIP Preprocessor Options

For the following options, you can specify a positive value from 1 to 65535 bytes, or 0 to disable event generation for the option regardless of whether the associated rule is enabled.

- **Maximum Request URI Length**
- **Maximum Call ID Length**
- **Maximum Request Name Length**
- **Maximum From Length**
- **Maximum To Length**
- **Maximum Via Length**
- **Maximum Contact Length**
- **Maximum Content Length**

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Ports

Specifies the ports to inspect for SIP traffic. You can specify an integer from 0 to 65535. Separate multiple port numbers with commas.

Methods to Check

Specifies SIP methods to detect. You can specify any of the following currently defined SIP methods:

```
ack, benotify, bye, cancel, do, info, invite, join, message,
notify, options, prack, publish, quath, refer, register,
service, sprack, subscribe, unsubscribe, update
```

Methods are case-insensitive. The method name can include alphabetic characters, numbers, and the underscore character. No other special characters are permitted. Separate multiple methods with commas.

Because new SIP methods might be defined in the future, your configuration can include an alphabetic string that is not currently defined. The system supports up to 32 methods, including the 21 currently defined methods and an additional 11 methods. The system ignores any undefined methods that you might configure.

Note that, in addition to any methods you specify for this option, the 32 total methods includes methods specified using the `sip_method` keyword in intrusion rules.

Maximum Dialogs within a Session

Specifies the maximum number of dialogs allowed within a stream session. If more dialogs than this number are created, the oldest dialogs are dropped until the number of dialogs does not exceed the maximum number specified. You can specify an integer from 1 to 4194303.

You can enable rule 140:27 to generate events and, in an inline deployment, drop offending packets for this option. See Setting Intrusion Rule States.

Maximum Request URI Length

Specifies the maximum number of bytes to allow in the Request-URI header field. A Longer URI generates an event and, in an inline deployment, drops offending packets when rule 140:3 is enabled. The request URI field indicates the destination path or page for the request.

Maximum Call ID Length

Specifies the maximum number of bytes to allow in the request or response Call-ID header field. A longer Call-ID generates an event and, in an inline deployment, drops offending packets when rule 140:5 is enabled. The Call-ID field uniquely identifies the SIP session in requests and responses.

Maximum Request Name Length

Specifies the maximum number of bytes to allow in the request name, which is the name of the method specified in the CSeq transaction identifier. A longer request name generates an event and, in an inline deployment, drops offending packets when rule 140:7 is enabled.

Maximum From Length

Specifies the maximum number of bytes to allow in the request or response From header field. A longer From generates an event and, in an inline deployment, drops offending packets when rule 140:9 is enabled. The From field identifies the message initiator.

Maximum To Length

Specifies the maximum number of bytes to allow in the request or response To header field. A longer To generates an event and, in an inline deployment, drops offending packets when rule 140:11 is enabled. The To field identifies the message recipient.

Maximum Via Length

Specifies the maximum number of bytes to allow in the request or response Via header field. A longer Via generates an event and, in an inline deployment, drops offending packets when rule 140:13 is enabled. The Via field provides the path followed by the request and, in a response, receipt information.

Maximum Contact Length

Specifies the maximum number of bytes to allow in the request or response Contact header field. A longer Contactgenerates an event and, in an inline deployment, drops offending packets when rule 140:15 is enabled. The Contact field provides a URI that specifies the location to contact with subsequent messages.

Maximum Content Length

Specifies the maximum number of bytes to allow in the content of the request or response message body. Longer content generates an event and, in an inline deployment, drops offending packets when rule 140:16 is enabled.

Ignore Audio/Video Data Channel

Enables and disables inspection of data channel traffic. Note that the preprocessor continues inspection of other non-data-channel SIP traffic when you enable this option.

Related Topics

SIP Keywords

Configuring the SIP Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **SIP Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **SIP Configuration**.

Step 6 Modify the options described in SIP Preprocessor Options, on page 41.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable SIP preprocessor rules (GID 140). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

Additional SIP Preprocessor Rules

The SIP preprocessor rules in the following table are not associated with specific configuration options. As with other SIP preprocessor rules, you must enable these rules if you want them to generate events and, in an inline deployment, drop offending packets.

Table 12: Additional SIP Preprocessor Rules

Preprocessor Rule GID:SID	Triggers when...
140:1	the preprocessor is monitoring the maximum number of SIP sessions allowed by the system.
140:2	the required Request_URI field is empty in a SIP request.
140:4	the Call-ID header field is empty in a SIP request or response.
140:6	the value for the sequence number in the SIP request or response CSeq field is not a 32-bit unsigned integer less than 231.
140:8	the From header field is empty in a SIP request or response.
140:10	the To header field is empty in a SIP request or response.
140:12	the Via header field is empty in a SIP request or response
140:14	the required Contact header field is empty in a SIP request or response.
140:17	a single SIP request or response packet in UDP traffic contains multiple messages. Note that older SIP versions supported multiple messages, but SIP 2.0 supports only one message per packet.
140:18	the actual length of the message body in a SIP request or response in UDP traffic does not match the value specified in the Content-Length header field in a SIP request or response.
140:19	the preprocessor does not recognize a method name in the CSeq field of a SIP response.
140:20	the SIP server does not challenge an authenticated invite message. Note that this occurs in the case of the InviteReplay billing attack.
140:21	session information changes before the call is set up. Note that this occurs in the case of the FakeBusy billing attack.
140:22	the response status code is not a three-digit number.
140:23	the Content-Type header field does not specify a content type and the message body contains data.
140:24	the SIP version is not 1, 1.1, or 2.0.
140:25	the method specified in the CSeq header and the method field do not match in a SIP request.
140:26	the preprocessor does not recognize the method named in the SIP request method field.

The GTP Preprocessor

The General Service Packet Radio (GPRS) Tunneling Protocol (GTP) provides communication over a GTP core network. The GTP preprocessor detects anomalies in GTP traffic and forwards command channel signaling messages to the rules engine for inspection. You can use the `gtp_version`, `gtp_type`, and `gtp_info` rule keywords to inspect GTP command channel traffic for exploits.

A single configuration option allows you to modify the default setting for the ports that the preprocessor inspects for GTP command channel messages.

GTP Preprocessor Rules

You must enable the GTP preprocessor rules in the following table if you want them to generate events and, in an inline deployment, drop offending packets.

Table 13: GTP Preprocessor Rules

Preprocessor Rule GID:SID	Description
143:1	Generates an event when the preprocessor detects an invalid message length.
143:2	Generates an event when the preprocessor detects an invalid information element length.
143:3	Generates an event when the preprocessor detects information elements that are out of order.

Configuring the GTP Preprocessor

You can use this procedure to modify the ports the GTP preprocessor monitors for GTP command messages.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel on the left.

Step 4 If **GTP Command Channel Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **GTP Command Channel Configuration**.

Step 6 Enter a **Ports** value.

Separate multiple ports with commas.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to enable intrusion events, enable GTP preprocessor rules (GID 143). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

The IMAP Preprocessor

The Internet Message Application Protocol (IMAP) is used to retrieve email from a remote IMAP server. The IMAP preprocessor inspects server-to-client IMAP4 traffic and, when associated preprocessor rules are enabled, generates events on anomalous traffic. The preprocessor can also extract and decode email attachments in client-to-server IMAP4 traffic and send the attachment data to the rules engine. You can use the `file_data` keyword in an intrusion rule to point to the attachment data.

Extraction and decoding include multiple attachments, when present, and large attachments that span multiple packets.

IMAP Preprocessor Options

Note that decoding, or extraction when the MIME email attachment does not require decoding, includes multiple attachments when present, and large attachments that span multiple packets.

Note also that the highest value is used when the values for the **Base64 Decoding Depth**, **7-Bit/8-Bit/Binary Decoding Depth**, **Quoted-Printable Decoding Depth**, or **Unix-to-Unix Decoding Depth** options are different in:

- the default network analysis policy
- any other custom network analysis policy invoked by network analysis rules in the same access control policy

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Ports

Specifies the ports to inspect for IMAP traffic. You can specify an integer from 0 to 65535. Separate multiple port numbers with commas.

Base64 Decoding Depth

Specifies the maximum number of bytes to extract and decode from each Base64 encoded MIME email attachment. You can specify a positive value, or specify 0 to decode all the Base64 data. Specify -1 to ignore Base64 data.

Note that positive values not divisible by 4 are rounded up to the next multiple of 4 except for the values 65533, 65534, and 65535, which are rounded down to 65532.

When this option is enabled, you can enable rule 141:4 generate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

7-Bit/8-Bit/Binary Decoding Depth

Specifies the maximum bytes of data to extract from each MIME email attachment that does not require decoding. These attachment types include 7-bit, 8-bit, binary, and various multipart content types such as plain text, jpeg images, mp3 files, and so on. You can specify a positive value, or specify 0 to extract all data in the packet. Specify -1 to ignore non-decoded data.

When this option is enabled, you can enable rule 141:6 to generate events and, in an inline deployment, drop offending packets when extraction fails; extraction could fail, for example, because of corrupted data.

Quoted-Printable Decoding Depth

Specifies the maximum number of bytes to extract and decode from each quoted-printable (QP) encoded MIME email attachment. You can specify a positive value, or specify 0 to decode all QP encoded data in the packet. Specify -1 to ignore QP encoded data.

When this option is enabled, you can enable rule 141:5 to generate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Unix-to-Unix Decoding Depth

Specifies the maximum number of bytes to extract and decode from each Unix-to-Unix encoded (uuencoded) email attachment. You can specify a positive value, or specify 0 to decode all uuencoded data in the packet. Specify -1 to ignore uuencoded data.

When this option is enabled, you can enable rule 141:7 to generate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Related Topics

The file_data Keyword

Configuring the IMAP Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **IMAP Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **IMAP Configuration**.

Step 6 Modify the settings described in IMAP Preprocessor Options, on page 46.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to enable intrusion events, enable IMAP preprocessor rules (GID 141); see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Layers in Intrusion and Network Analysis Policies
Conflicts and Changes: Network Analysis and Intrusion Policies

Additional IMAP Preprocessor Rules

The IMAP preprocessor rules in the following table are not associated with specific configuration options. As with other IMAP preprocessor rules, you must enable these rules if you want them to generate events and, in an inline deployment, drop offending packets.

Table 14: Additional IMAP Preprocessor Rules

Preprocessor Rule GID:SID	Description
141:1	Generates an event when the preprocessor detects a client command that is not defined in RFC 3501.
141:2	Generates an event when the preprocessor detects a server response that is not defined in RFC 3501.
141:3	Generates an event when the preprocessor is using the maximum amount of memory allowed by the system. At this point, the preprocessor stops decoding until memory becomes available.

The POP Preprocessor

The Post Office Protocol (POP) is used to retrieve email from a remote POP mail server. The POP preprocessor inspects server-to-client POP3 traffic and, when associated preprocessor rules are enabled, generates events on anomalous traffic. The preprocessor can also extract and decode email attachments in client-to-server POP3 traffic and send the attachment data to the rules engine. You can use the `file_data` keyword in an intrusion rule to point to attachment data.

Extraction and decoding include multiple attachments, when present, and large attachments that span multiple packets.

POP Preprocessor Options

Note that decoding, or extraction when the MIME email attachment does not require decoding, includes multiple attachments when present, and large attachments that span multiple packets.

Note also that the highest value is used when the values for the **Base64 Decoding Depth**, **7-Bit/8-Bit/Binary Decoding Depth**, **Quoted-Printable Decoding Depth**, or **Unix-to-Unix Decoding Depth** options are different in:

- the default network analysis policy
- any other custom network analysis policy invoked by network analysis rules in the same access control policy

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Ports

Specifies the ports to inspect for POP traffic. You can specify an integer from 0 to 65535. Separate multiple port numbers with commas.

Base64 Decoding Depth

Specifies the maximum number of bytes to extract and decode from each Base64 encoded MIME email attachment. You can specify a positive value, or specify 0 to decode all the Base64 data. Specify -1 to ignore Base64 data.

Note that positive values not divisible by 4 are rounded up to the next multiple of 4 except for the values 65533, 65534, and 65535, which are rounded down to 65532.

When this option is enabled, you can enable rule 142:4 to generate an event and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

7-Bit/8-Bit/Binary Decoding Depth

Specifies the maximum bytes of data to extract from each MIME email attachment that does not require decoding. These attachment types include 7-bit, 8-bit, binary, and various multipart content types such as plain text, jpeg images, mp3 files, and so on. You can specify a positive value, or specify 0 to extract all data in the packet. Specify -1 to ignore non-decoded data.

When this option is enabled, you can enable rule 142:6 to generate an event and, in an inline deployment, drop offending packets when extraction fails; extraction could fail, for example, because of corrupted data.

Quoted-Printable Decoding Depth

Specifies the maximum number of bytes to extract and decode from each quoted-printable (QP) encoded MIME email attachment. You can specify a positive value, or specify 0 to decode all QP encoded data in the packet. Specify -1 to ignore QP encoded data.

When this option is enabled, you can enable rule 142:5 to generate an event and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Unix-to-Unix Decoding Depth

Specifies the maximum number of bytes to extract and decode from each Unix-to-Unix encoded (uuencoded) email attachment. You can specify a positive value, or specify 0 to decode all uuencoded data in the packet. Specify -1 to ignore uuencoded data.

When this option is enabled, you can enable rule 142:7 to generate an event and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies
The file_data Keyword

Configuring the POP Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **POP Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **POP Configuration**.

Step 6 Modify the settings described in POP Preprocessor Options, on page 49.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to enable intrusion events, enable POP preprocessor rules (GID 142). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

Additional POP Preprocessor Rules

The POP preprocessor rules in the following table are not associated with specific configuration options. As with other POP preprocessor rules, you must enable these rules if you want them to generate events and, in an inline deployment, drop offending packets.

Table 15: Additional POP Preprocessor Rules

Preprocessor Rule GID:SID	Description
142:1	Generates an event when the preprocessor detects a client command that is not defined in RFC 1939.
142:2	Generates an event when the preprocessor detects a server response that is not defined in RFC 1939.
142:3	Generates an event when the preprocessor is using the maximum amount of memory allowed by the system. At this point, the preprocessor stops decoding until memory becomes available.

The SMTP Preprocessor

The SMTP preprocessor instructs the rules engine to normalize SMTP commands. The preprocessor can also extract and decode email attachments in client-to-server traffic and, depending on the software version, extract email file names, addresses, and header data to provide context when displaying intrusion events triggered by SMTP traffic.

SMTP Preprocessor Options

You can enable or disable normalization, and you can configure options to control the types of anomalous traffic the SMTP decoder detects.

Note that decoding, or extraction when the MIME email attachment does not require decoding, includes multiple attachments when present, and large attachments that span multiple packets.

Note also that the highest value is used when the values for the **Base64 Decoding Depth**, **7-Bit/8-Bit/Binary Decoding Depth**, **Quoted-Printable Decoding Depth**, or **Unix-to-Unix Decoding Depth** options are different in:

- the default network analysis policy
- any other custom network analysis policy invoked by network analysis rules in the same access control policy

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Ports

Specifies the ports whose SMTP traffic you want to normalize. You can specify a value greater than or equal to 0. Separate multiple ports with commas.

Stateful Inspection

When selected, causes SMTP decoder to save state and provide session context for individual packets and only inspects reassembled sessions. When cleared, analyzes each individual packet without session context.

Normalize

When set to `All`, normalizes all commands. Checks for more than one space character after a command.

When set to `None`, normalizes no commands.

When set to `Cmds`, normalizes the commands listed in **Custom Commands**.

Custom Commands

When **Normalize** is set to `Cmds`, normalizes the listed commands.

Specify commands which should be normalized in the text box. Checks for more than one space character after a command.

The space (ASCII 0x20) and tab (ASCII 0x09) characters count as space characters for normalization purposes.

Ignore Data

Does not process mail data; processes only MIME mail header data.

Ignore TLS Data

Does not process data encrypted under the Transport Layer Security protocol.

No Alerts

Disables intrusion events when accompanying preprocessor rules are enabled.

Detect Unknown Commands

Detects unknown commands in SMTP traffic.

You can enable rule 124:5 to generate events and, in an inline deployment, drop offending packets for this option.

Max Command Line Len

Detects when an SMTP command line is longer than this value. Specify `0` to never detect command line length.

RFC 2821, the Network Working Group specification on the Simple Mail Transfer Protocol, recommends 512 as a maximum command line length.

You can enable rule 124:1 to generate events and, in an inline deployment, drop offending packets for this option.

Max Header Line Len

Detects when an SMTP data header line is longer than this value. Specify `0` to never detect data header line length.

You can enable rules 124:2 and 124:7 to generate events and, in an inline deployment, drop offending packets for this option.

Max Response Line Len

Detects when an SMTP response line is longer than this value. Specify `0` to never detect response line length.

RFC 2821 recommends 512 as a maximum response line length.

You can enable rule 124:3 to generate events and, in an inline deployment, drop offending packets for this option and also for **Alt Mac Command Line Len**, when that option is enabled.

Alt Max Command Line Len

Detects when the SMTP command line for any of the specified commands is longer than this value. Specify `0` to never detect command line length for the specified commands. Different default line lengths are set for numerous commands.

This setting overrides the Max Command Line Len setting for the specified commands.

You can enable rule 124:3 to generate events and, in an inline deployment, drop offending packets for this option and also for **Max Response Line Len** when that option is enabled.

Invalid Commands

Detects if these commands are sent from the client side.

You can enable rule 124:6 to generate events and, in an inline deployment, drop offending packets for this option and also for **Invalid Commands**.

Valid Commands

Permits commands in this list.

Even if this list is empty, the preprocessor permits the following valid commands: ATRN AUTH BDAT DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SIZE SOML STARTTLS TICK TIME TURN TURNME VERB VRFY XADR XAUTH XCIR XEXCH50 X-EXPS XGEN XLICENSE X-LINK2STATE XQUE XSTA XTRN XUSR

Note RCPT TO and MAIL FROM are SMTP commands. The preprocessor configuration uses command names of RCPT and MAIL, respectively. Within the code, the preprocessor maps RCPT and MAIL to the correct command name.

You can enable rule 124:4 to generate events and, in an inline deployment, drop offending packets for this option and also for **Invalid Commands** when that option is configured.

Data Commands

Lists commands that initiate sending data in the same way the SMTP DATA command sends data per RFC 5321. Separate multiple commands with spaces.

Binary Data Commands

Lists commands that initiate sending data in a way that is similar to how the BDAT command sends data per RFC 3030. Separate multiple commands with spaces.

Authentication Commands

Lists commands that initiate an authentication exchange between client and server. Separate multiple commands with spaces.

Detect xlink2state

Detects packets that are part of X-Link2State Microsoft Exchange buffer data overflow attacks. In inline deployments, the system can also drop those packets.

You can enable rule 124:8 to generate events and, in an inline deployment, drop offending packets for this option.

Base64 Decoding Depth

When **Ignore Data** is disabled, specifies the maximum number of bytes to extract and decode from each Base64 encoded MIME email attachment. You can specify from a positive value, or specify 0 to decode all the Base64 data. Specify -1 to ignore Base64 data. The preprocessor will not decode data when **Ignore Data** is selected.

Note that positive values not divisible by 4 are rounded up to the next multiple of 4 except for the values 65533, 65534, and 65535, which are rounded down to 65532.

When this option is enabled, you can enable rule 124:10 to ggenerate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Note that this option replaces the deprecated options **Enable MIME Decoding** and **Maximum MIME Decoding Depth**, which are still supported in existing intrusion policies for backward compatibility.

7-Bit/8-Bit/Binary Decoding Depth

When **Ignore Data** is disabled, specifies the maximum bytes of data to extract from each MIME email attachment that does not require decoding. These attachment types include 7-bit, 8-bit, binary, and various multipart content types such as plain text, jpeg images, mp3 files, and so on. You can specify a positive value,

or specify 0 to extract all data in the packet. Specify -1 to ignore non-decoded data. The preprocessor will not extract data when **Ignore Data** is selected.

Quoted-Printable Decoding Depth

When **Ignore Data** is disabled, specifies the maximum number of bytes to extract and decode from each quoted-printable (QP) encoded MIME email attachment.

You can specify from 1 to 65535 bytes, or specify 0 to decode all QP encoded data in the packet. Specify -1 to ignore QP encoded data. The preprocessor will not decode data when **Ignore Data** is selected.

When this option is enabled, you can enable rule 124:11 to generate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Unix-to-Unix Decoding Depth

When **Ignore Data** is disabled, specifies the maximum number of bytes to extract and decode from each Unix-to-Unix encoded (uuencoded) email attachment. You can specify from 1 to 65535 bytes, or specify 0 to decode all uuencoded data in the packet. Specify -1 to ignore uuencoded data. The preprocessor will not decode data when **Ignore Data** is selected.

When this option is enabled, you can enable rule 124:13 to generate events and, in an inline deployment, drop offending packets when decoding fails; decoding could fail, for example, because of incorrect encoding or corrupted data.

Log MIME Attachment Names

Enables extraction of MIME attachment file names from the MIME Content-Disposition header and associates the file names with all intrusion events generated for the session. Multiple file names are supported.

When this option is enabled, you can view file names associated with events in the Email Attachment column of the intrusion events table view.

Log To Addresses

Enables extraction of recipient email addresses from the SMTP RCPT TO command and associates the recipient addresses with all intrusion events generated for the session. Multiple recipients are supported.

When this option is enabled, you can view recipients associated with events in the Email Recipient column of the intrusion events table view.

Log From Addresses

Enables extraction of sender email addresses from the SMTP MAIL FROM command and associates the sender addresses with all intrusion events generated for the session. Multiple sender addresses are supported.

When this option is enabled, you can view senders associated with events in the Email Sender column of the intrusion events table view.

Log Headers

Enables extraction of email headers. The number of bytes to extract is determined by the value specified for **Header Log Depth**.

You can use the `content` or `protected_content` keyword to write intrusion rules that use email header data as a pattern. You can also view the extracted email header in the intrusion event packet view.

Header Log Depth

Specifies the number of bytes of the email header to extract when **Log Headers** is enabled. You can specify 0 to 20480 bytes. A value of 0 disables **Log Headers**.

Related Topics

Basic content and protected_content Keyword Arguments

Configuring SMTP Decoding

In a multidomain deployment, the system displays policies created in the current domain, which you can edit. It also displays policies created in ancestor domains, which you cannot edit. To view and edit policies created in a lower domain, switch to that domain.

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation pane.

Step 4 If **SMTP Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **SMTP Configuration**.

Step 6 Modify the options described in SMTP Preprocessor Options, on page 51.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to generate events and, in an inline deployment, drop offending packets, enable SMTP preprocessor rules (GID 124). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

The SSH Preprocessor

The SSH preprocessor detects:

- The Challenge-Response Buffer Overflow exploit
- The CRC-32 exploit
- The SecureCRT SSH Client Buffer Overflow exploit
- Protocol mismatches
- Incorrect SSH message direction
- Any version string other than version 1 or 2

Challenge-Response Buffer Overflow and CRC-32 attacks occur after the key exchange and are, therefore, encrypted. Both attacks send an uncharacteristically large payload of more than 20 KBytes to the server immediately after the authentication challenge. CRC-32 attacks apply only to SSH Version 1; Challenge-Response Buffer Overflow exploits apply only to SSH Version 2. The version string is read at the beginning of the session. Except for the difference in the version string, both attacks are handled in the same way.

The SecureCRT SSH exploit and protocol mismatch attacks occur when attempting to secure a connection, before the key exchange. The SecureCRT exploit sends an overly long protocol identifier string to the client that causes a buffer overflow. A protocol mismatch occurs when either a non-SSH client application attempts to connect to a secure SSH server or the server and client version numbers do not match.

You can configure the SSH preprocessor to inspect traffic on a specified port or list of ports, or to automatically detect SSH traffic. It will continue to inspect SSH traffic until either a specified number of encrypted packets has passed within a specified number of bytes, or until a specified maximum number of bytes is exceeded within the specified number of packets. If the maximum number of bytes is exceeded, it is assumed that a CRC-32 (SSH Version 1) or a Challenge-Response Buffer Overflow (SSH Version 2) attack has occurred. Note that without configuration the preprocessor detects any version string value other than version 1 or 2.

Also note the that the SSH preprocessor does not handle brute force attacks.

SSH Preprocessor Options

The preprocessor stops inspecting traffic for a session when either of the following occurs:

- a valid exchange between the server and the client has occurred for this number of encrypted packets; the connection continues.
- the **Number of Bytes Sent Without Server Response** is reached before the number of encrypted packets to inspect is reached; the assumption is made that there is an attack.

Each valid server response during **Number of Encrypted Packets to Inspect** resets the **Number of Bytes Sent Without Server Response** and the packet count continues.

Consider the following example SSH preprocessor configuration:

- **Server Ports**: 22
- **Autodetect Ports**: off

- **Maximum Length of Protocol Version String**: 80
- **Number of Encrypted Packets to Inspect**: 25
- **Number of Bytes Sent Without Server Response**: 19,600
- All detect options are enabled.

In the example, the preprocessor inspects traffic only on port 22. That is, auto-detection is disabled, so it inspects only on the specified port.

Additionally, the preprocessor in the example stops inspecting traffic when either of the following occurs:

- The client sends 25 encrypted packets which contain no more than 19,600 bytes, cumulative. The assumption is there is no attack.
- The client sends more than 19,600 bytes within 25 encrypted packets. In this case, the preprocessor considers the attack to be the Challenge-Response Buffer Overflow exploit because the session in the example is an SSH Version 2 session.

The preprocessor in the example will also detect any of the following that occur while it is processing traffic:

- a server overflow, triggered by a version string greater than 80 bytes and indicating a SecureCRT exploit
- a protocol mismatch
- a packet flowing in the wrong direction

Finally, the preprocessor will automatically detect any version string other than version 1 or version 2.

If no preprocessor rule is mentioned in the following descriptions, the option is not associated with a preprocessor rule.

Server Ports

Specifies on which ports the SSH preprocessor should inspect traffic.

You can configure a single port or a comma-separated list of ports.

Autodetect Ports

Sets the preprocessor to automatically detect SSH traffic.

When this option is selected, the preprocessor inspects all traffic for an SSH version number. It stops processing when neither the client nor the server packet contains a version number. When disabled, the preprocessor inspects only the traffic identified by the **Server Ports** option.

Number of Encrypted Packets to Inspect

Specifies the number of stream reassembled encrypted packets to examine per session.

Setting this option to zero will allow all traffic to pass.

Reducing the number of encrypted packets to inspect may result in some attacks escaping detection. Raising the number of encrypted packets to inspect may negatively affect performance.

Number of Bytes Sent Without Server Response

Specifies the maximum number of bytes an SSH client may send to a server without getting a response before assuming there is a Challenge-Response Buffer Overflow or CRC-32 attack.

Increase the value for this option if the preprocessor generates false positives on the Challenge-Response Buffer Overflow or CRC-32 exploit.

Maximum Length of Protocol Version String

Specifies the maximum number of bytes allowed in the server's version string before considering it to be a SecureCRT exploit.

Detect Challenge-Response Buffer Overflow Attack

Enables or disables detecting the Challenge-Response Buffer Overflow exploit.

You can enable rule 128:1 to generate events and, in an inline deployment, drop offending packets for this option. Note that an SFTP session can occasionally trigger rule 128:1.

Detect SSH1 CRC-32 Attack

Enables or disables detecting the CRC-32 exploit.

You can enable rule 128:2 to generate events and, in an inline deployment, drop offending packets for this option.

Detect Server Overflow

Enables or disables detecting the SecureCRT SSH Client Buffer Overflow exploit.

You can enable rule 128:3 togenerate events and, in an inline deployment, drop offending packets for this option.

Detect Protocol Mismatch

Enables or disables detecting protocol mismatches.

You can enable rule 128:4 to generate events and, in an inline deployment, drop offending packets for this option.

Detect Bad Message Direction

Enables or disables detecting when traffic flows in the wrong direction (that is, if the presumed server generates client traffic, or if a client generates server traffic).

You can enable rule 128:5 to generate events and, in an inline deployment, drop offending packets for this option.

Detect Payload Size Incorrect for the Given Payload

Enables or disables detecting packets with an incorrect payload size such as when the length specified in the SSH packet is not consistent with the total length specified in the IP header or the message is truncated, that is, there is not enough data for a full SSH header.

You can enable rule 128:6 to generate events and, in an inline deployment, drop offending packets for this option.

Detect Bad Version String

Note that, when enabled, the preprocessor detects without configuration any version string other than version 1 or 2.

You can enable rule 128:7 to generate events and, in an inline deployment, drop offending packets for this option.

Configuring the SSH Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **SSH Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **SSH Configuration**.

Step 6 Modify the options described in SSH Preprocessor Options, on page 57.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to enable intrusion events, enable SSH preprocessor rules (GID 128). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies

The SSL Preprocessor

The SSL preprocessor allows you to configure SSL inspection, which can block encrypted traffic, decrypt it, or inspect the traffic with access control. Whether or not you configure SSL inspection, the SSL preprocessor also analyzes SSL handshake messages when detected in traffic and determines when a session becomes

encrypted. Identifying encrypted traffic allows the system to stop intrusion and file inspection of encrypted payloads, which helps reduce false positives and improve performance.

The SSL preprocessor can also examine encrypted traffic to detect attempts to exploit the Heartbleed bug, and generate events when it detects such exploits.

You can suspend inspecting traffic for intrusions and malware once the session is encrypted. If you configure SSL inspection, the SSL preprocessor also identifies encrypted traffic you can block, decrypt, or inspect with access control.

Using the SSL preprocessor to decrypt encrypted traffic does not require a license. All other SSL preprocessor functionality, including halting inspection of encrypted payloads for malware and intrusions, and detecting Heartbleed bug exploits, requires a Protection license.

How SSL Preprocessing Works

The SSL preprocessor stops intrusion and file inspection of encrypted data, and inspects encrypted traffic with an SSL policy if you configure SSL inspection. This can help to eliminate false positives. The SSL preprocessor maintains state information as it inspects the SSL handshake, tracking both the state and SSL version for that session. When the preprocessor detects that a session state is encrypted, the system marks the traffic in that session as encrypted. You can configure the system to stop processing on all packets in an encrypted session when encryption is established, as well as generate an event when it detects an attempt to exploit the Heartbleed bug.

For each packet, the SSL preprocessor verifies that the traffic contains an IP header, a TCP header, and a TCP payload, and that it occurs on the ports specified for SSL preprocessing. For qualifying traffic, the following scenarios determine whether the traffic is encrypted:

- The system observes all packets in a session, **Server side data is trusted** is not enabled, and the session includes a Finished message from both the server and the client and at least one packet from each side with an Application record and without an Alert record.
- The system misses some of the traffic, **Server side data is trusted** is not enabled, and the session includes at least one packet from each side with an Application record that is not answered with an Alert record.
- The system observes all packets in a session, **Server side data is trusted** is enabled, and the session includes a Finished message from the client and at least one packet from the client with an Application record and without an Alert record.
- The system misses some of the traffic, **Server side data is trusted** is enabled, and the session includes at least one packet from the client with an Application record that is not answered with an Alert record.

If you choose to stop processing on encrypted traffic, the system ignores future packets in a session after it marks the session as encrypted.

In addition, during the SSL handshake, the preprocessor monitors heartbeat requests and responses. The preprocessor generates an event if it detects:

- a heartbeat request containing a payload length value greater than the payload itself
- a heartbeat response that is larger than the value stored in the Max Heartbeat Length field

Note You can add the `ssl_state` and `ssl_version` keywords to a rule to use SSL state or version information within the rule.

Related Topics

SSL Keywords
TLS/SSL Inspection Requirements

SSL Preprocessor Options

Note The system-provided network analysis policies enable the SSL preprocessor by default. Cisco recommends that you do not disable the SSL preprocessor in custom deployments if you expect encrypted traffic to cross your network.

Without SSL inspection configured, the system attempts to inspect encrypted traffic for malware and intrusions without decrypting it. When you enable the SSL preprocessor, it detects when a session becomes encrypted. After the SSL preprocessor is enabled, the rules engine can invoke the preprocessor to obtain SSL state and version information. If you enable rules using the `ssl_state` and `ssl_version` keywords in an intrusion policy, you should also enable the SSL preprocessor in that policy.

Ports

Specifies the ports, separated by commas, where the SSL preprocessor should monitor traffic for encrypted sessions. Only ports specified in this field will be checked for encrypted traffic.

Note If the SSL preprocessor detects non-SSL traffic over the ports specified for SSL monitoring, it tries to decode the traffic as SSL traffic, and then flags it as corrupt.

Stop inspecting encrypted traffic

Enables or disables inspection of traffic in a session after the session is marked as encrypted.

Enable this option to disable inspection and reassembly for encrypted sessions. The SSL preprocessor maintains state for the session so it can disable inspection of all traffic in the session. When this option is enabled a few packets of a session are verified to ensure the flow is encrypted after which deep inspection is bypassed. Every bypassed session increases the fast-forwarded flows count shown in the response of the **show snort statistics** command. Moreover, since deep inspection is bypassed, the initiator and responder bytes in the connection event are not accurate. They are less than the value of the actual session, since it only includes the packets inspected by Snort and it does not include any packets after the deep inspection is bypassed. This behavior holds good for connection summary events and all traffic values shown in the widgets.

The system only stops inspecting traffic in encrypted sessions if both:

- SSL preprocessing is enabled
- this option is selected

If you clear this option, you cannot modify the **Server side data is trusted** option.

Server side data is trusted

When Stop inspecting encrypted traffic is enabled, enables identification of encrypted traffic based only on the client-side traffic,

Max Heartbeat Length

By specifying a number of bytes, enables inspection of heartbeat requests and responses within the SSL handshake for Heartbleed bug exploit attempts. You can specify an integer from `1` to `65535`, or `0` to disable the option.

If the preprocessor detects a heartbeat request whose payload length is greater than the actual payload length and rule 137:3 is enabled, or a heartbeat response greater in size than the value configured for this option when rule 137:4 is enabled, the preprocessor generates an event and, in an inline deployment, drops offending packets.

Configuring the SSL Preprocessor

Step 1 Choose **Policies** > **Access Control**, then click **Network Analysis Policies** or **Policies** > **Access Control** > **Intrusion**, then click **Network Analysis Policies**.

Note If your custom user role limits access to the first path listed here, use the second path to access the policy.

Step 2 Click **Edit** ( ) next to the policy you want to edit.

If **View** ( ) appears instead, the configuration belongs to an ancestor domain, or you do not have permission to modify the configuration.

Step 3 Click **Settings** in the navigation panel.

Step 4 If **SSL Configuration** under **Application Layer Preprocessors** is disabled, click **Enabled**.

Step 5 Click **Edit** ( ) next to **SSL Configuration**.

Step 6 Modify any of the settings described in SSL Preprocessor Options, on page 62.

- Enter a value in the **Ports** field. Separate multiple values with commas.
- Check or clear the **Stop inspecting encrypted traffic** check box.
- If you checked **Stop inspecting encrypted traffic**, check or clear **Server side data is trusted**.
- Enter a value in the **Max Heartbeat Length** field.

Tip A value of `0` disables this option.

Step 7 To save changes you made in this policy since the last policy commit, click **Policy Information**, then click **Commit Changes**.

If you leave the policy without committing changes, cached changes since the last commit are discarded if you edit a different policy.

What to do next

- If you want to enable intrusion events, enable SSL preprocessor rules (GID 137). For more information, see Setting Intrusion Rule States.
- Deploy configuration changes; see Deploy Configuration Changes.

Related Topics

Managing Layers
Conflicts and Changes: Network Analysis and Intrusion Policies
TLS/SSL Inspection Requirements

SSL Preprocessor Rules

If you want to generate events and, in an inline deployment, drop offending packets, enable SSL preprocessor rules (GID 137).

The following table describes the SSL preprocessor rules you can enable.

Table 16: SSL Preprocessor Rules

Preprocessor Rule GID:SID	Description
137:1	Detects a ClientHello message after a ServerHello message, which is invalid and considered to be anomalous behavior.
137:2	Detects a ServerHello message without a ClientHello message when the SSL preprocessor option **Server side data is trusted** is disabled, which is invalid and considered to be anomalous behavior.
137:3	Detects a heartbeat request with a payload length greater than the payload itself when the SSL preprocessor option **Max Heartbeat Length** contains a non-zero value, which indicates an attempt to exploit the Heartbleed bug.
137:4	Detects a heartbeat response larger than a non-zero value specified in the SSL preprocessor option **Max Heartbeat Length**, which indicates an attempt to exploit the Heartbleed bug.