Proxies in Cisco Multicloud Defense

First Published: 2025-04-03

Cisco Multicloud Defense Proxy

About Proxy

A proxy acts an intermediary in connecting a client or user to a target server or the internet. This represents the server in your network and enables caching, filtering, load balancing, and blocking access.

This is how a proxy works:

- A user accesses the network.
- A request is sent to a proxy server. The proxy server analyzes the request. It filters and modifies the request based on set rules.
- The proxy then forwards the request to the server or Internet.
- The proxy then receives the response from the server or Internet.
- The proxy processes/modifies the response and send it securely to the user.

The different types of proxies are:

1. Forward Proxy: This is used for anonymity and filtering. The proxy resides in front of a client and forwards the requests to the target server/internet. The proxy hides the client identity.
2. Reverse Proxy: This is used for load balancing and security purpose. The proxy reside in front of a server. The proxy forwards client requests to the target server. The proxy hides the server identity.
3. Transparent Proxy: This is used for monitoring networks. This proxy intercepts requests without making modifications to them.
4. Anonymous Proxy: This is used for anonymity. This proxy hides the user's IP address.
5. High Anonymity Proxy : This is also called Elite Proxy and is used for anonymity.This proxy hides the very presence of the proxy.

Multicloud Defense and Proxies

Multicloud Defense contains forward, reverse, transparent, and explicit proxies. These proxies are used to:

- Inspect content of encrypted flows
- Inspect access controls
- Inspect Layer 7 firewall, User ID, WAF, IDS/IPS, App ID, DLP, antivirus, antimalware, URL filtering and more.

The following proxies are used in Multicloud Defense - transparent proxy, explicit proxy, forward proxy, and reverse proxy. Application and stream proxies can be implemented as forward, reverse, transparent, or explicit proxies, based on the desired control, transparency, and configuration level, to address various operational needs.

Application Proxy

An application proxy in Multicloud Defense is used to handle HTTP/Websocket traffic. It operates at the application layer (Layer 7 of the OSI model) and is designed to process and forward requests for web applications. It understands and manipulates HTTP headers, which allows it to make decisions based on specific web application conditions.

Key Features of Application Proxy:

- Operates at the application layer (Layer 7)
- Can modify HTTP headers and content
- Supports SSL termination and initiation

Stream Proxy

A stream proxy in Multicloud Defense is used for handling TCP/UDP traffic. It operates at the transport layer (Layer 4 of the OSI model) and is suitable for proxying non-HTTP traffic, such as database queries, mail protocols, or any custom TCP/UDP applications. Stream proxies are generally simpler than application proxies because they do not intercept the data being transmitted, only the transport-layer information.

Key Features of Stream Proxy:

- Operates at the transport layer (Layer 4)
- Does not modify application data, only forwards it
- Useful for database servers, SMTP, or other custom protocols.

Differences between Application Proxy and Stream Proxy

- Layer of Operation: Application proxies operate at the application layer (Layer 7), while stream proxies operate at the transport layer (Layer 4).
- Data Processing: Application proxies can interpret HTTP headers and content, whereas stream proxies simply forward data without interpretation.
- Use Cases: Application proxies are used for web applications and services, while stream proxies are used for handling generic TCP/UDP traffic.

Choosing between an application proxy and a stream proxy depends on the type of traffic you need to handle and the level of control and modification you require. For web applications, an application proxy is appropriate. For non-HTTP services, such as database requests, a stream proxy is more suitable.

Transparent Proxy

Definition: A transparent proxy intercepts client requests without requiring any configuration on the client side. The client is unaware of the proxy's existence.

How it Works:

- Redirection: Traffic is redirected to the proxy through routing.
- Request Handling: The proxy intercepts the request and forwards it to the intended web server.
- Response: The proxy pretends to be the origin content server (OCS) and sends the response back to the client.

Configuration Requirements:

- Redirection Technology: Routing
- Certificate Installation: For HTTPS traffic, the proxy server's certificate must be installed in the workstation's trust store.

Example:

- HTTP Request: GET / HTTP/1.1
- Destination IP: Intended web server (DNS resolved by the client)

Explicit Proxy

Definition: An explicit proxy requires client-side configuration, where the client is explicitly aware of the proxy's existence and routes all traffic through it.

How it Works:

- Client Configuration: Each workstation and web browser must have its proxy settings pointing to the proxy server.
- Request Handling: The client sends requests directly to the proxy, which then forwards them to the origin server.
- Response: The proxy responds with its own IP information back to the client.

Configuration Requirements:

- Client Settings: Proxy settings must be configured on each client device.
- Certificate Installation: For HTTPS traffic, the proxy server's certificate must be installed in the workstation's trust store.

Example:

- HTTP Request: GET http://www.google.com/ HTTP/1.1
- Destination IP: Configured proxy server

Forward Proxy

Definition: A forward proxy handles requests from a group of clients to an unknown, untrusted, or arbitrary group of resources outside their control.

How it Works:

- Client Requests: Clients send requests to the forward proxy.

- Proxy Handling: The forward proxy processes the requests and forwards them to the appropriate external servers.
- Response: The proxy receives the response from the external server and sends it back to the client.

Deployment Modes:

- Transparent Mode: Routing
- Explicit Mode: Requires client-side configuration

Example:

- Use Case: A company uses a forward proxy to control and monitor employee internet access.

Reverse Proxy

Definition: A reverse proxy is deployed on the same network as the HTTP(S) servers and serves content on behalf of these servers.

How it Works:

- Client Requests: Clients send requests to the reverse proxy.
- Proxy Handling: The reverse proxy forwards the requests to the appropriate internal servers.
- Response: The internal servers send the response back to the reverse proxy, which then forwards it to the client.

Benefits:

- Security: Hides the internal server details from the clients
- Advanced Security: Allows for deep-packet inspection and advanced security such as IPS, Antimalware, and URL filtering to be applied to traffic.

Example:

- Use Case: A website uses a reverse proxy to inspect external traffic destined to public-facing resources in the customer's public cloud environment.

Considerations and Limitations of Proxies

Multicloud Defense Gateway offers robust proxying capabilities for managing network security.

Mixing and Matching Proxy Modes

Mixing and matching proxy modes on the same port in Multicloud Defense is not directly supported because Multicloud Defense treats HTTP and stream (TCP/UDP) traffic separately, and each listen on its own port. Each type of traffic (HTTP or stream) requires its own dedicated port, as they are handled in separate configuration contexts (http and stream blocks).

- HTTP and Stream Separation: Multicloud Defense configurations for HTTP and stream traffic are distinct and designed to operate independently. The http configuration is used for handling HTTP/HTTPS traffic,

while the stream configuration is used for TCP/UDP traffic. Each configuration listens on its own port, and Multicloud Defense cannot multiplex different types of traffic on the same port.

- Port Binding: A port can only be bound to one process or service at a time. Since HTTP and stream handling are different services in Multicloud Defense, they cannot share the same port.

Proxy Modes and Port Configuration

Single Port Limitation:

In Forward Proxy mode, you cannot mix different proxy modes on the same port. This means that if you have different sets of domains with varying requirements (e.g., some needing WebSocket headers and others not), you must configure them on separate ports.

WebSocket Handling:

If all traffic is running on port 443 and some domains require WebSocket headers while others do not, you must configure all proxies to handle WebSocket traffic. In this configuration, the default proxy mode is HTTPS. However, if additional WebSocket headers are detected, the proxy will upgrade the connection to WEBSOCKET_S and pass the headers to the server. This upgrade is managed on a session-by-session basis.

Application Layer Distinction

Traffic Differentiation:

The proxy cannot distinguish between different types of traffic at the application layer if they are coming in on the same port. This limitation means that mixed-mode application proxies cannot operate on the same port unless there is a method to differentiate the traffic at the application layer.

Layer 3 (L3) Limitation:

The application proxy does not understand lower-layer distinctions such as Layer 3. It relies on application-layer mechanisms to manage traffic.

Forward Proxy Limitation:

The concept of using Server Name Indication (SNI) for traffic distinction does not exist in the Forward Proxy implementation. As a result, mixed-mode application-based proxies cannot be run on the same port in Forward Proxy mode.

Reverse (Ingress) Proxy Mode: Key Considerations

Reverse Proxy Handling:

SNI Utilization: In Ingress (Reverse Proxy) mode, the proxy uses SNI to distinguish between different domains. This SNI is domain-specific and not a wildcard, allowing the proxy to handle traffic appropriately based on the domain name.

Practical Implications

Configuration Requirements:

Administrators must ensure that different proxy modes are configured on separate ports to avoid conflicts. For example, if you have a set of domains that require WebSocket headers and another set that does not, you must run these proxies on different ports.

Session Management:

The proxy's ability to upgrade connections to WEBSOCKET_S based on detected headers ensures that WebSocket traffic is handled correctly without manual intervention for each session.

Proxy Timeouts

Timeout Settings (configurable):

- All flows (proxy and non-proxy).
    - tcp_timeout_syn_sent: 30
    - tcp_timeout_syn_recv: 30
    - tcp_timeout_fin_wait: 30
    - tcp_timeout_close_wait: 30
    - tcp_timeout_last_ack: 30
    - tcp_timeout_time_wait: 30
    - tcp_timeout_max_retrans: 30
    - tcp_timeout_unacknowledged: 30
    - tcp_timeout_established: 180
    - udp_timeout: 10
    - udp_timeout_stream: 30
- No Config change settings.
    - tcp_timeout_syn_sent: 120
    - tcp_timeout_syn_recv: 60
    - tcp_timeout_established: 180
    - tcp_timeout_close: 10
    - tcp_timeout_close_wait: 60
    - tcp_timeout_fin_wait: 30
    - tcp_timeout_last_ack: 30
    - tcp_timeout_max_retrans: 300
    - tcp_timeout_time_wait: 30
    - tcp_timeout_unacknowledged: 300
- HTTP Keepalive: keepalive_timeout: 5

Timeout Settings (not configurable).

- Forward Proxy flows (TCP, TLS, HTTP, HTTPS, WEBSOCKET, WEBSOCKETS)
    - proxy_connect_connect_timeout: 180

    - proxy_connect_read_timeout: 180
    - proxy_connect_send_timeout: 180

- Reverse Proxy Flows (TCP, TLS, HTTP, HTTPS)
    - proxy_connect_timeout: 180
    - proxy_read_timeout: 180
    - proxy_send_timeout: 180

- Reverse Proxy flows (WEBSOCKET, WEBSOCKETS)
    - proxy_connect_timeout: 180
    - proxy_read_timeout: 1800
    - proxy_send_timeout: 180

Mixing and matching proxy modes on the same port in Multicloud Defense is not directly supported because Multicloud Defense treats HTTP and stream (TCP/UDP) traffic separately, and each listen on its own port. Each type of traffic (HTTP or stream) requires its own dedicated port, as they are handled in separate configuration contexts (http and stream blocks).

HTTP and Stream Separation: Multicloud Defense configurations for HTTP and stream traffic are distinct and designed to operate independently. The http configuration is used for handling HTTP/HTTPS traffic, while the stream configuration is used for TCP/UDP traffic. Each configuration listens on its own port, and Multicloud Defense cannot multiplex different types of traffic on the same port.

A port can only be bound to one process or service at a time. Since HTTP and stream handling are different services in Multicloud Defense, they cannot share the same port.