Security Configuration Guide for vEdge Routers, Cisco SD-WAN Releases 19.1, 19.2, and 19.3

First Published: 2019-05-23

Last Modified: 2019-08-20

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

What's New for Cisco SD-WAN

Note The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on RFP documentation, or language that is used by a referenced third-party product.

This chapter describes what's new in Cisco SD-WAN for each release.

What's New for Cisco SD-WAN Release 19.2.x

This section applies to Cisco vEdge devices.

Cisco is constantly enhancing the SD-WAN solution with every release and we try and keep the content in line with the latest enhancements. The following table lists new and modified features we documented in the Configuration, Command Reference, and Hardware Installation guides. For information on additional features and fixes that were committed to the SD-WAN solution, see the *Resolved and Open Bugs* section in the Release Notes.

Table 1: What's New for Cisco vEdge Device

Feature	Description
Getting Started	
API Cross-Site Request Forgery Prevention	This feature adds protection against Cross-Site Request Forgery (CSRF) that occurs when using Cisco SD-WAN REST APIs. This protection is provided by including a CSRF token with API requests. You can put requests on an allowed list so that they do not require protection if needed. See Cross-Site Request Forgery Prevention.
Systems and Interfaces	

Feature	Description
Secure Shell Authentication Using RSA Keys	This feature helps configure RSA keys by securing communication between a client and a Cisco SD-WAN server. See SSH Authentication using vManage on Cisco XE SD-WAN Devices. See Configure SSH Authentication.
Policies	
Packet Duplication for Noisy Channels	This feature helps mitigate packet loss over noisy channels, thereby maintaining high application QoE for voice and video. See Configure and Monitor Packet Duplication.
Control Traffic Flow Using Class of Service Values	This feature lets you control the flow of traffic into and out of a Cisco device's interface based on the conditions defined in the quality of service (QoS) map. A priority field and a layer 2 class of service (CoS) were added for configuring the re-write rule. See Configure Localized Data Policy for IPv4 Using Cisco vManage.
Security	
Secure Communication Using Pairwise IPsec Keys	This feature allows private pairwise IPSec session keys to be created and installed for secure communication between IPSec devices and its peers. For related information, see IPSec Pairwise Keys Overview.
Configure IKE-Enabled IPsec Tunnels	The pre-shared key needs to be at least 16 bytes in length. The IPsec tunnel establishment fails if the key size is less than 16 characters when the router is upgraded to version 19.2. See Configure IKE-Enabled IPsec Tunnels.
Network Optimization and High Availability	
Disaster Recovery for vManage	This feature helps you configure Cisco vManage in an active or standby mode to counteract hardware or software failures that may occur due to unforeseen circumstances. See Configure Disaster Recovery.
Share VNF Devices Across Service Chains	This feature lets you share Virtual Network Function (VNF) devices across service chains to improve resource utilisation and reduce resource fragmentation. See Share VNF Devices Across Service Chains.
Monitor Service Chain Health	This feature lets you configure periodic checks on the service chain data path and reports the overall status. To enable service chain health monitoring, NFVIS version 3.12.1 or later should be installed on all CSP devices in a cluster. See Monitor Service Chain Health.
Manage PNF Devices in Service Chains	This feature lets you add Physical Network Function (PNF) devices to a network, in addition to the Virtual Network function (VNF) devices. These PNF devices can be added to service chains and shared across service chains, service groups, and a cluster. Inclusion of PNF devices in the service chain can overcome the performance and scaling issues caused by using only VNF devices in a service chain. See Manage PNF Devices in Service Chains.

CHAPTER 2

Security Overview

Security is a critical element of today's networking infrastructure. Network administrators and security officers are hard pressed to defend their network against attacks and breaches. As a result of hybrid clouds and remote employee connectivity, the security perimeter around networks is disappearing. There are multiple problems with the traditional ways of securing networks, including:

- Very little emphasis is placed on ensuring the authenticity of the devices involved in the communication.
- Securing the links between a pair of devices involves tedious and manual setup of keys and shared passwords.
- Scalability and high availability solutions are often at odds with each other.

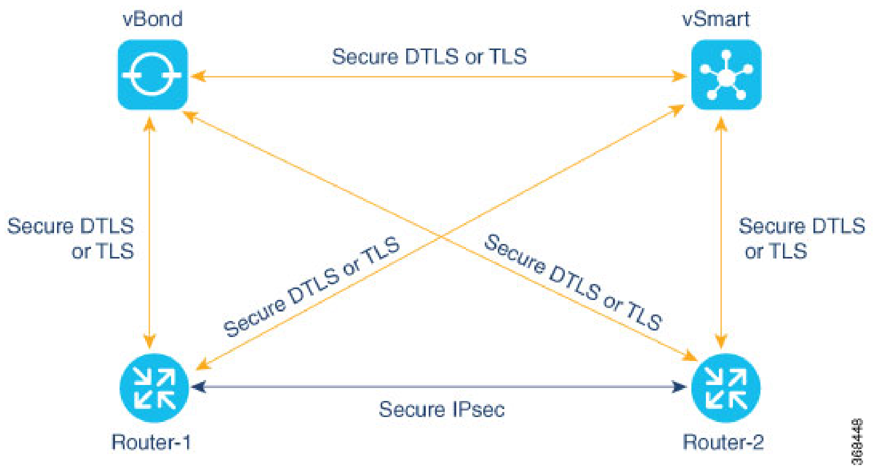
This chapter contains the following topics:

Cisco SD-WAN Security Components

The Cisco SD-WAN solution takes a fundamentally different approach to security, basing its core design around the following precepts:

- Authentication—The solution ensures that only authentic devices are allowed to send traffic to one another.
- Encryption—All communication between each pair of devices is automatically secure, completely eliminating the overhead involved in securing the links.
- Integrity—No group keys or key server issues are involved in securing the infrastructure.

These three components—authentication, encryption, and integrity—are key to securing the Cisco SD-WAN overlay network infrastructure.



The topics on Control Plane Security Overview and Data Plane Security Overview examine how authentication, encryption, and integrity are implemented throughout the Cisco SD-WAN overlay network. The security discussion refers to the following illustration of the components of the Cisco SD-WAN network—the vSmart controller, the vBond orchestrator, and the routers. The connections between these devices form the control plane (in orange) and the data plane (in purple), and it is these connections that need to be protected by appropriate measures to ensure the security of the network devices and all network traffic.

Security Provided by NAT Devices

While the primary purpose of NAT devices is to allow devices with private IP addresses in a local-area network (LAN) to communicate with devices in public address spaces, such as the Internet, NAT devices also inherently provide a level of security, functioning as hardware firewalls to prevent unwanted data traffic from passing through the routers and to the LAN networks in the service-side networks connected to the router.

To enhance the security at branch sites, you can place the router behind a NAT device. The router can interact with NAT devices configured with the following Session Traversal Utilities for NAT (STUN) methods, as defined in RFC 5389 :

- Full-cone NAT, or one-to-one NAT—This method maps an internal address and port pair to an external address and port. Any external host can send packets to LAN devices behind the router by addressing them to the external address and port.
- Address-restricted cone NAT, or restricted-cone NAT—This method also maps an internal address and port to and external address and port. However, an external host can send packets to the internal device only if the external address (and any port at that address) has received a packet from the internal address and port.
- Port-restricted cone NAT—This method is a stricter version of restricted-cone NAT, in which an external host can send packets to the internal address and port only if the external address and port pair has received a packet from that internal address and port. The external device must send packets from the specific port to the specific internal port.
- Symmetric NAT—With this method, each request from the same internal IP address and port to an external IP address and port is mapped to a unique external source IP address and port. If the same internal host sends a packet with the same source address and port but to a different destination, the NAT device creates a different mapping. Only an external host that receives a packet from an internal host can send

a packet back. The routers support symmetric NAT only on one side of the WAN tunnel. That is, only one of the NAT devices at either end of the tunnel can use symmetric NAT. When a router operates behind a NAT device running symmetric NAT, only one of the NAT devices at either end of the tunnel can use symmetric NAT. The router that is behind a symmetric NAT cannot establish a BFD tunnel with a remote router that is behind a symmetric NAT, an address-restricted NAT, or a port-restricted NAT. To allow a router to function behind a symmetric NAT, you must configure the vManage and vSmart controller control connections to use TLS. DTLS control connections do not work through a symmetric NAT.

Security for Connections to External Devices

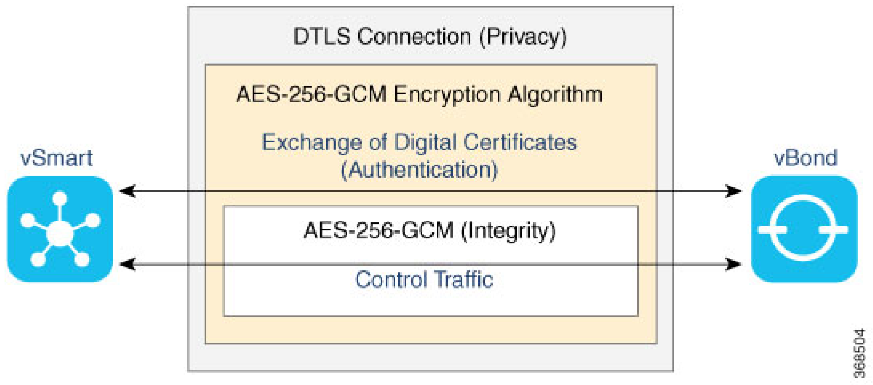
Cisco SD-WAN routers can use the standards-based Internet Key Exchange (IKE) protocol when establishing IPsec tunnels between a device within the overlay network and a device that is external to the overlay network, such as a cloud-hosted service or a remote user. The Cisco SD-WAN software supports IKE version 2, which performs mutual authentication and establishes and maintains security associations (SAs). IPsec provides confidentiality, data integrity, access control, and data source authentication for the traffic being exchanged over the IPsec tunnel.

Control Plane Security Overview

The control plane of any network is concerned with determining the network topology and defining how to direct packets. In a traditional network, the control plane operations of building and maintaining routing and forwarding tables and directing packets towards their destination are handled by routing and switching protocols, which typically offer few or no mechanisms for authenticating devices or for encrypting routing updates and other control information. In addition, the traditional methods for providing security are highly manual and do not scale. As examples, certificates are typically installed manually rather than in an automated fashion, and using preshared keys is not a very secure approach for providing device security.

The Cisco SD-WAN control plane has been designed with network and device security in mind. The foundation of the control plane is one of two security protocols derived from SSL (Secure Sockets Layer)—the Datagram Transport Layer Security (DTLS) protocol and the Transport Layer Security (TLS) protocol. The vSmart controller, which is the centralized brain of the Cisco SD-WAN solution, establishes and maintains DTLS or TLS connections to all Cisco SD-WAN devices in the overlay network: to the routers, the vBond orchestrators, to Cisco vManage, and to other vSmart controllers. These connections carry control plane traffic. DTLS or TLS provides communication privacy between Cisco SD-WAN devices in the network, using the Advanced Encryption Standard (AES-256) encryption algorithm to encrypt all control traffic sent over the connections.

The privacy and encryption in the control plane offered by DTLS and TLS provide a safe and secure foundation for the other two security components, authentication and integrity. To perform authentication, the Cisco SD-WAN devices exchange digital certificates. These certificates, which are either installed by the software or hard-coded into the hardware, depending on the device, identify the device and allow the devices themselves to automatically determine which ones belong in the network and which are imposters. For integrity, the DTLS or TLS connections run AES-256-GCM, a cryptographic secure hash algorithm which ensures that all control and data traffic sent over the connections has not been tampered with.


DTLS Connection (Privacy)
AES-256-GCM Encryption Algorithm
Exchange of Digital Certificates (Authentication)
vSmart
vBond
AES-256-GCM (Integrity)
Control Traffic
368504

The following are the control plane security components, which function in the privacy provided by DTLS or TLS connections:

- **AES-256-GCM** algorithm provides encryption services.
- **Digital certificates** are used for authentication.
- **AES-256-GCM** is responsible for ensuring integrity.

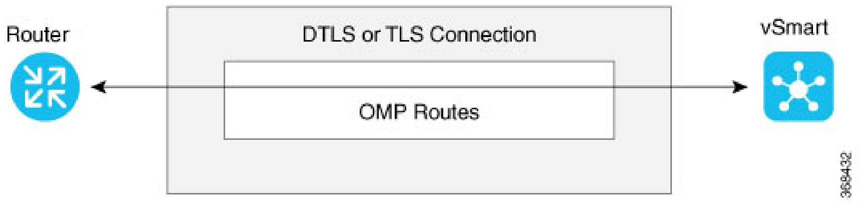
DTLS and TLS Infrastructure

Security protocols derived from SSL provide the foundation for the Cisco SD-WAN control plane infrastructure.

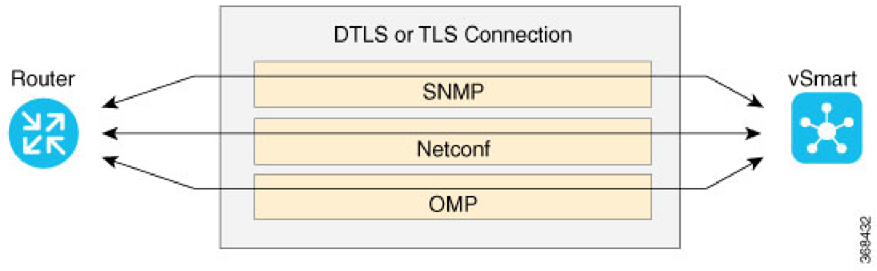
The first is the DTLS protocol, which is a transport privacy protocol for connectionless datagram protocols such as UDP, provides the foundation for the Cisco SD-WAN control plane infrastructure. It is based on the stream-oriented Transport Layer Security (TLS) protocol, which provides security for TCP-based traffic. (TLS itself evolved from SSL.) The Cisco SD-WAN infrastructure design uses DTLS running over UDP to avoid some of the issues with TCP, including the delays associated with stream protocols and some security issues. However, because UDP performs no handshaking and sends no acknowledgments, DTLS has to handle possible packet re-ordering, loss of datagrams, and data larger than the datagram packet size.

The control plane infrastructure can also be configured to run over TLS. This might be desirable in situations where the protections of TCP outweigh its issues. For example, firewalls generally offer better protection for TCP servers than for UDP servers.

The Cisco SD-WAN software implements the standard version of DTLS with UDP, which is defined in RFC 6347 . DTLS for use with other protocols is defined in a number of other RFCs . For TLS, the Cisco SD-WAN software implements the standard version defined in RFC 5246.



In the Cisco SD-WAN architecture, the Cisco SD-WAN devices use DTLS or TLS as a tunneling protocol, which is an application-level (Layer 4) tunneling protocol. When the vSmart controllers, vBond orchestrators, Cisco vManages, and routers join the network, they create provisional DTLS or TLS tunnels between them as part of the device authentication process. After the authentication process completes successfully, the provisional tunnels between the routers and vSmart controllers, and those between the vBond orchestrators and vSmart controllers, become permanent and remain up as long as the devices are active in the network. It is these authenticated, secure DTLS or TLS tunnels that are used by all the protocol applications running on the Cisco SD-WAN devices to transport their traffic. For example, an OMP session on a router communicates with an OMP session on a vSmart controller by sending plain IP traffic through the secure DTLS or TLS tunnel between the two devices. The Overlay Management Protocol is the Cisco SD-WAN control protocol used to exchange routing, policy, and management information among Cisco SD-WAN devices, as described in Overlay Routing Overview.



A Cisco SD-WAN daemon running on each vSmart controller and router creates and maintains the secure DTLS or TLS connections between the devices. This daemon is called vdaemon and is discussed later in this article. After the control plane DTLS or TLS connections are established between these devices, multiple protocols can create sessions to run and route their traffic over these connections—including OMP, Simple Network Management Protocol (SNMP), and Network Configuration Protocol (Netconf)—without needing to be concerned with any security-related issues. The session-related traffic is simply directed over the secure connection between the routers and vSmart controllers.

Control Plane Authentication

The Cisco SD-WAN control plane uses digital certificates with 2048-bit RSA keys to authenticate the Cisco SD-WAN routers in the network. The digital certificates are created, managed, and exchanged by standard components of the public key infrastructure (PKI):

- **Public keys**— These keys are generally known.
- **Private keys**— These keys are private. They reside on each Cisco SD-WAN router and cannot be retrieved from the router.
- **Certificates** signed by a root certification authority (CA)— The trust chain associated with the root CA needs to be present on all Cisco SD-WAN router.

In addition to standard PKI components, the Cisco SD-WAN router serial numbers and the router chassis numbers are used in the authentication processes.

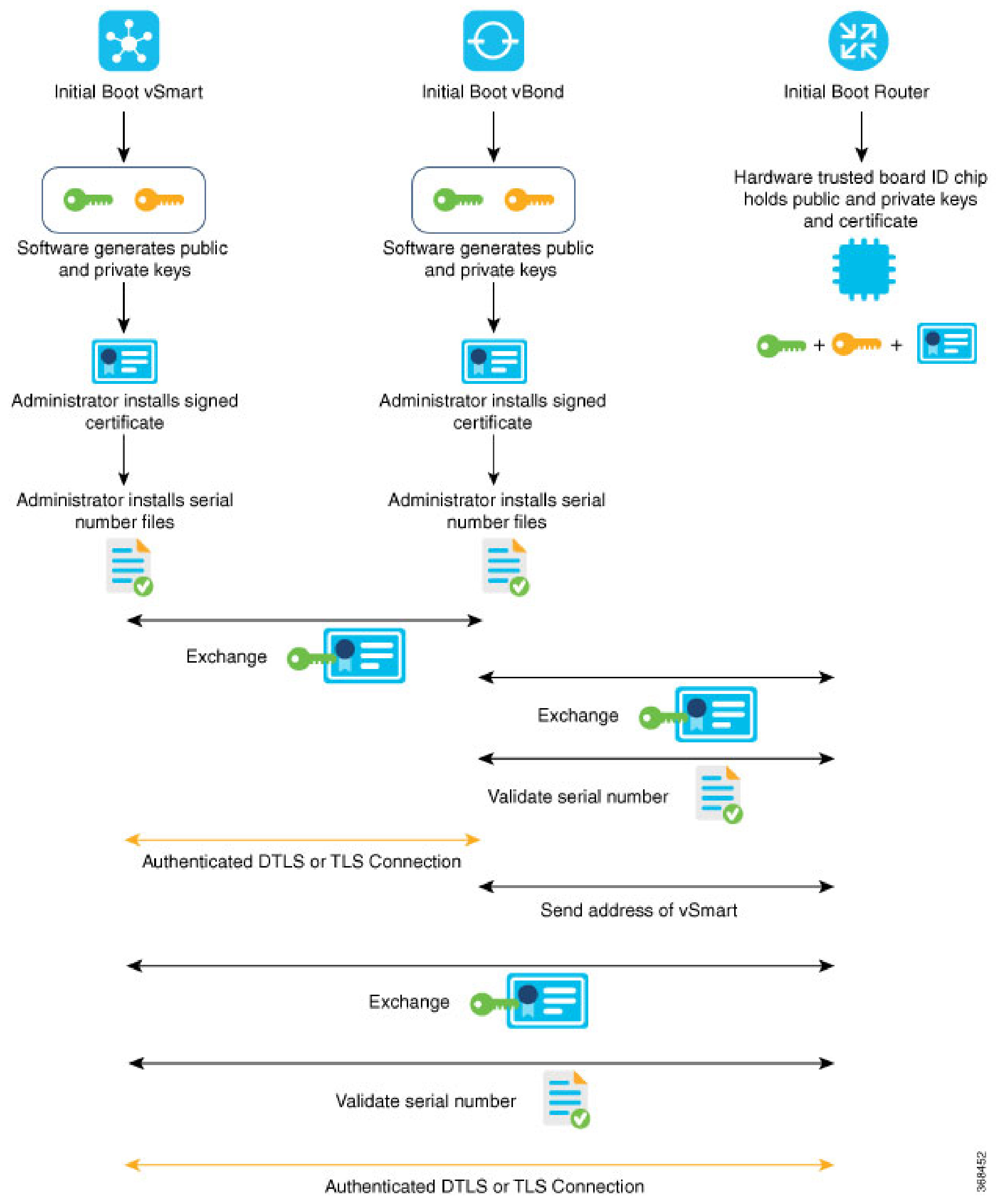
Let's first look at the PKI components that are involved in router authentication. On vEdge 100, 1000, and 2000 routers, the public and private keys and the certificates are managed automatically, by a Trusted Board ID chip that is built into the router. For vEdge 5000, instead of a Trusted Board ID chip, a Trusted Platform Module (TPM) is used. When the routers are manufactured, this chip is programmed with a signed certificate. This certificate includes the router's public key, its serial number, and the router's private key. When the routers boot up and join the network, they exchange their certificates (including the router's public key and serial number) with other Cisco SD-WAN routers as part of the router authentication process. Note that the router's private key always remains embedded in the router's chip, and it is never distributed, nor can it ever be retrieved from the router. In fact, any brute-force attempt to read the private key causes the chip to fail, thereby disabling all access to the router.

For vSmart controllers, vBond orchestrators, and Cisco vManage systems, the public and private keys and the certificates are managed manually. When you boot these routers for the first time, the Cisco SD-WAN software generates a unique private key–public key pair for each software image. The public key needs to be signed by the CA root. The network administrator then requests a signed certificate and manually installs it and the certificate chains on the vSmart controllers, vBond orchestrators, and Cisco vManage systems. A typical network might have only a small handful of vSmart controllers, vBond orchestrators, and Cisco vManages, so the burden of manually managing the keys and certificates on these routers is small.

To augment these standard PKI components, the Cisco SD-WAN software uses the router serial numbers in performing automatic router authentication. Specifically, it uses the router and vSmart serial numbers and the router chassis numbers. When a batch of routers is shipped, the manufacturer sends a text file that lists the serial numbers of the routers and the corresponding chassis numbers. For the vSmart controllers, when the network administrator receives the signed certificate, they should extract the serial numbers from the certificates and place them into a single text file, one serial number per line. Then the network administrator manually installs these two files. The file received from the manufacturer that lists all valid router serial and chassis numbers is uploaded and installed on vSmart controllers. Both the authorized serial number file and the file listing the vSmart serial numbers are uploaded and installed on vBond orchestrators. Then, during the automatic authentication process, as pairs of devices (routers and controllers) are establishing DTLS control connections, each device compares the serial numbers (and for routers, the chassis numbers) to those in the files installed on the router. A router allows a connection to be established only if the serial number or serial–chassis number combination (for a router) matches.

You can display the installed vSmart authorized serial numbers using the **show control valid-vsmarts** command on a vSmart controller or a router and the **show orchestrator valid-vsmarts** command on a vBond orchestrator. You can also run **show sdwan control valid-vsmarts** on Cisco IOS XE SD-WAN devices. You can display the installed router authorized serial and chassis number associations using the **show control valid-vedges** command on a vSmart controller and the **show orchestrator valid-devices** command on a vBond orchestrator

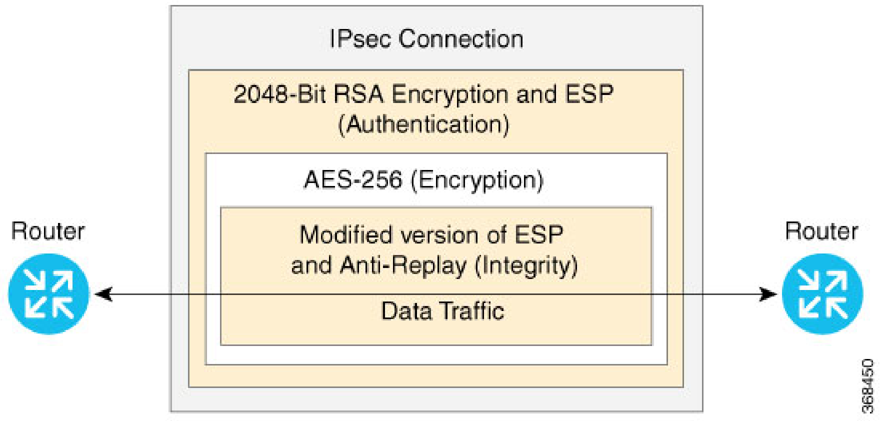
Now, let's look at how the PKI authentication components and the router serial and chassis numbers are used to authenticate router on the Cisco SD-WAN overlay network. When vSmart controllers, vBond orchestrators, and routers first boot up, they establish secure DTLS or TLS connections between the vSmart controllers and the routers. Over these connections, the devices authenticate each other, using the public and private keys, the signed certificates, and the routers serial numbers and performing a series of handshake operations to ensure that all the devices on the network are valid and not imposters. The following figure illustrates the key and certificate exchange that occurs when the Cisco SD-WAN devices boot. For details about the authentication that occurs during the bringup process, see Bringup Sequence of Events.


Initial Boot vSmart
Software generates public and private keys
Administrator installs signed certificate
Administrator installs serial number files
Initial Boot vBond
Software generates public and private keys
Administrator installs signed certificate
Administrator installs serial number files
Initial Boot Router
Hardware trusted board ID chip holds public and private keys and certificate
+
+
Exchange
Exchange
Validate serial number
Authenticated DTLS or TLS Connection
Send address of vSmart
Exchange
Validate serial number
Authenticated DTLS or TLS Connection
368452

Control Plane Encryption

Control plane encryption is done by either DTLS, which is based on the TLS protocol, or TLS. These protocol encrypt the control plane traffic that is sent across the connections between Cisco SD-WAN devices to validate the integrity of the data. TLS uses asymmetric cryptography for authenticating key exchange, symmetric encryption for confidentiality, and message authentication codes for message integrity.

A single Cisco SD-WAN device can have DTLS or TLS connections to multiple Cisco SD-WAN devices, so vdaemon creates a kernel route for each destination. For example, a router would typically have one kernel route, and hence one DTLS or TLS connection, for each vSmart controller. Similarly, a vSmart controller would have one kernel route and one DTLS or TLS connection for each router in its domain.


IPsec Connection
2048-Bit RSA Encryption and ESP (Authentication)
AES-256 (Encryption)
Modified version of ESP and Anti-Replay (Integrity)
Data Traffic
Router
Router
368450

Control Plane Integrity

The Cisco SD-WAN design implements control plane integrity by combining two security elements: SHA-1 or SHA-2 message digests, and public and private keys.

SHA-1 and SHA-2 are cryptographic hash functions that generate message digests (sometimes called simply digests) for each packet sent over a control plane connection. SHA-1 generates a 160-bit message digest. SHA-2 is a family that consists of six hash functions with digests that are 224, 256, 384, or 512 bits. The receiver then generates a digest for the packet, and if the two match, the packet is accepted as valid. Both SHA-1 and SHA-2 allow verification that the packet's contents have not been tampered with.

The second component of control plane integrity is the use of public and private keys. When a control plane connection is being established, a local Cisco SD-WAN device sends a challenge to a remote device. The remote device encrypts the challenge by signing it with its private key, and returns the signed challenge to the local device. The local device then uses the remote device's public key to verify that the received challenge matches the sent challenge.

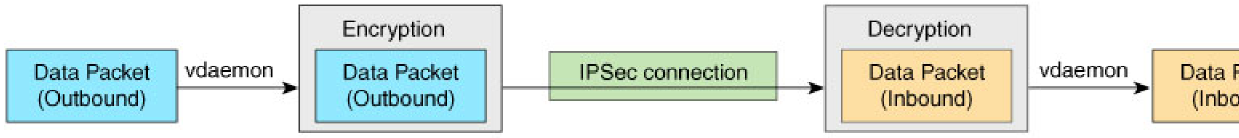
Then, once a control plane connection is up, keys are used to ensure that packets have been sent by a trusted host and were not inserted midstream by an untrusted source. The authenticity of each packet is verified through encryption and decryption with symmetric keys that were exchanged during the process of establishing the control connection.

Data Plane Security Overview

The data plane of any network is responsible for handling data packets that are transported across the network. (The data plane is also sometimes called the forwarding plane.) In a traditional network, data packets are typically sent directly over the Internet or another type of public IP cloud, or they could be sent through MPLS tunnels. If the routers in the Cisco SD-WAN overlay network were to send traffic over a public IP cloud, the transmission would be insecure. Anyone would be able to sniff the traffic, and it would be easy to implement various types of attacks, including man-in-the-middle (MITM) attacks.

The underlying foundation for security in the Cisco SD-WAN data plane is the security of the control plane. Because the control plane is secure—all devices are validated, and control traffic is encrypted and cannot be tampered with—we can be confident in using routes and other information learned from the control plane to create and maintain secure data paths throughout a network of routers.

The data plane provides the infrastructure for sending data traffic among the routers in the Cisco SD-WAN overlay network. Data plane traffic travels within secure Internet Security (IPsec) connections. The Cisco SD-WAN data plane implements the key security components of authentication, encryption, and integrity in the following ways:



- Authentication—As mentioned above, the Cisco SD-WAN control plane contributes the underlying infrastructure for data plane security. In addition, authentication is enforced by two other mechanisms:
    - In the traditional key exchange model, the vSmarts sends IPsec encryption keys to each edge device.

      In the pairwise keys model, the vSmart sends Diffie-Hellman public values to the edge devices and they generate pairwise IPsec encryption keys using ECDH and a P-384 curve. For more information, see Pairwise Keys , on page 38.
    - By default IPsec tunnel connections use a modified version of the Encapsulating Security Payload (ESP) protocol for authentication on IPsec tunnels.

- **Encryption**—A modified version of ESP protects the data packet's payload. This version of the protocol also checks the outer IP and UDP headers. Hence, this option supports an integrity check of the packet similar to the Authentication Header (AH) protocol. Data encryption is done using the AES-GCM-256 cipher.
- **Integrity**—To guarantee that data traffic is transmitted across the network without being tampered with, the data plane implements several mechanisms from the IPsec security protocol suite:
    - A modified version of the ESP protocol encapsulates the payload of data packets.
    - The modified version of ESP uses an AH-like mechanism to check the integrity of the outer IP and UDP headers. You can configure the integrity methods supported on each router, and this information is exchanged in the router's TLOC properties. If two peers advertise different authentication types, they negotiate the type to use, choosing the strongest method.
    - The anti-replay scheme protects against attacks in which an attacker duplicates encrypted packets.

Data Plane Authentication and Encryption

During the bringup of the overlay, the Cisco vSmart Controller establishes the information for edge routers to send data to each other. However before a pair of routers can exchange data traffic, they establish an IPsec connection between them, which they use as a secure communications channel. Since the Cisco vSmart Controller has authenticated the devices, the devices do not further authenticate each other.

Control plane communications have allowed the edge device to have enough information to establish IPsec tunnels. Edge devices simply send data through the tunnels. There is no authentication step.

In a traditional IPsec environment, key exchange is handled by the Internet Key Exchange (IKE) protocol. IKE first sets up secure communications channels between devices and then establishes security associations

(SAs) between each pair of devices that want to exchange data. IKE uses a Diffie-Hellman key exchange algorithm to generate a shared key that encrypts further IKE communication. To establish SAs, each device (n) exchanges keys with every other device in the network and creates per-pair keys, generating a unique key for each remote device. This scheme means that in a fully meshed network, each device has to manage n^2 key exchanges and (n-1) keys. As an example, in a 1,000-node network, 1,000,000 key exchanges are required to authenticate the devices, and each node is responsible for maintaining and managing 999 keys.

The discussion in the previous paragraph points out why an IKE-style key exchange does not scale as network size increases and why IKE could be a bottleneck in starting and in maintaining data exchange on a large network:

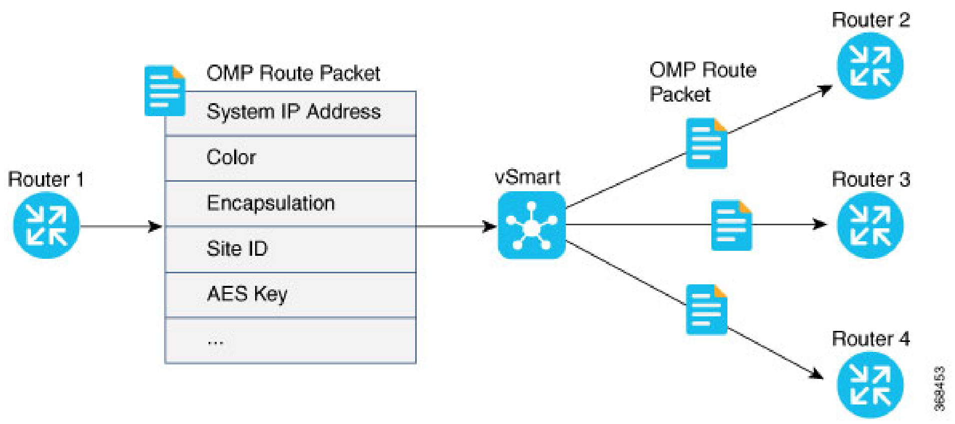
- The handshaking required to set up the communications channels is both time consuming and resource intensive.
- The processing required for the key exchange, especially in larger networks, can strain network resources and can take a long time.

The Cisco SD-WAN implementation of data plane authentication and encryption establishes SAs between each pair of devices that want to exchange data, but it dispenses with IKE altogether. Instead, to provide a scalable solution to data plane key exchange, the Cisco SD-WAN solution takes advantage of the fact that the DTLS control plane connections in the Cisco SD-WAN overlay network are known to be secure. Because the Cisco SD-WAN control plane establishes authenticated, encrypted, and tamperproof connections, there is no need in the data plane to set up secure communications channels to perform data plane authentication.

In the Cisco SD-WAN network for unicast traffic, data plane encryption is done by AES-256-GCM, a symmetric-key algorithm that uses the same key to encrypt outgoing packets and to decrypt incoming packets. Each router periodically generates an AES key for its data path (specifically, one key per TLOC) and transmits this key to the vSmart controller in OMP route packets, which are similar to IP route updates. These packets contain information that the vSmart controller uses to determine the network topology, including the router's TLOC (a tuple of the system IP address and traffic color) and AES key. The vSmart controller then places these OMP route packets into reachability advertisements that it sends to the other routers in the network. In this way, the AES keys for all the routers are distributed across the network. Even though the key exchange is symmetric, the routers use it in an asymmetric fashion. The result is a simple and scalable key exchange process that uses the Cisco vSmart Controller.

In Cisco SD-WAN Release 19.2.x and Cisco IOS XE SD-WAN Release 16.12.x onwards, Cisco SD-WAN supports IPSec pairwise keys that provide additional security. When IPSec pairwise keys are used, the edge router generates public and private Diffie-Hellman components and sends the public value to the vSmart for distribution to all other edge devices. For more information, see IPSec Pairwise Keys Overview, on page 37

For multicast traffic, data plane encryption is done by AES-256-CBC with SHA1-HMAC.



If control policies configured on a vSmart controller limit the communications channels between network devices, the reachability advertisements sent by the vSmart controller contain information only for the routers that they are allowed to exchange data with. So, a router learns the keys only for those routers that they are allowed to communicate with.

To further strengthen data plane authentication and encryption, routers regenerate their AES keys aggressively (by default, every 24 hours). Also, the key regeneration mechanism ensures that no data traffic is dropped when keys change.

In the Cisco SD-WAN overlay network, the liveness of SAs between router peers is tracked by monitoring BFD packets, which are periodically exchanged over the IPsec connection between the peers. IPsec relays the connection status to the vSmart controllers. If data connectivity between two peers is lost, the exchange of BFD packets stops, and from this, the vSmart controller learns that the connection has been lost.

The IPsec software has no explicit SA idle timeout, which specifies the time to wait before deleting SAs associated with inactive peers. Instead, an SA remains active as long as the IPsec connection between two routers is up, as determined by the periodic exchange of BFD packets between them. Also, the frequency with which SA keys are regenerated obviates the need to implement an implicit SA idle timeout.

In summary, the Cisco SD-WAN data plane authentication offers the following improvements over IKE:

- Because only n +1 keypaths are required rather than the n^2 required by IKE, the Cisco SD-WAN solution scales better as the network grows large.
- Keys are generated and refreshed locally, and key exchange is performed over a secure control plane.

Data Plane Integrity

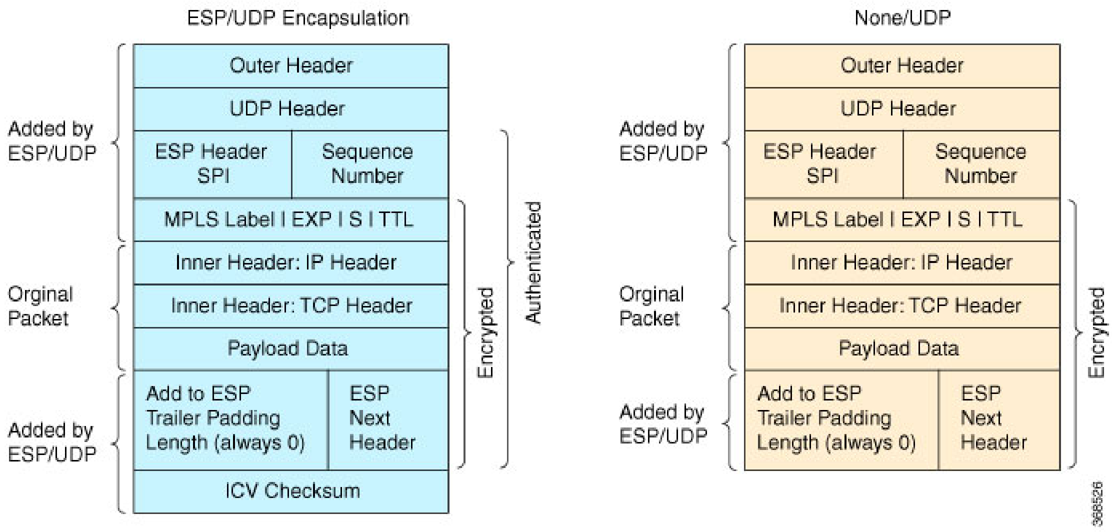
The following components contribute to the integrity of data packets in the Cisco SD-WAN data plane:

- ESP, which is a standard IPsec encryption protocol, protects (via encryption and authentication) the inner header, data packet payload, and ESP trailer in all data packets. The modifications to ESP also protect the outer IP and UDP headers
- Modifications to ESP, which protect (via authentication) the outer IP and UDP headers. This mimics the functionality of the AH protocol.
- Anti-replay, which is also part of the standard IPsec software suite, provides a mechanism to number all data packets and to ensure that receiving routers accept only packets with unique numbers.

The first of these components, ESP, is the standard IPsec encryption protocol. ESP protects a data packet's payload and its inner IP header fields both by encryption, which occurs automatically, and authentication. For authentication, ESP performs a checksum calculation on the data packet's payload and inner header fields and places the resultant hash (also called a digest) into a 12-byte HMAC-SHA1 field at the end of the packet. (A hash is a one-way compression.) The receiving device performs the same checksum and compares its calculated hash with that in the packet. If the two checksums match, the packet is accepted. Otherwise, it is dropped. In the figure below, the left stack illustrates the ESP/UDP encapsulation. ESP encrypts and authenticates the inner headers, payload, MPLS label (if present), and ESP trailer fields, placing the HMAC-SHA1 hash in the ICV checksum field at the end of the packet. The outer header fields added by ESP/UDP are neither encrypted nor authenticated.

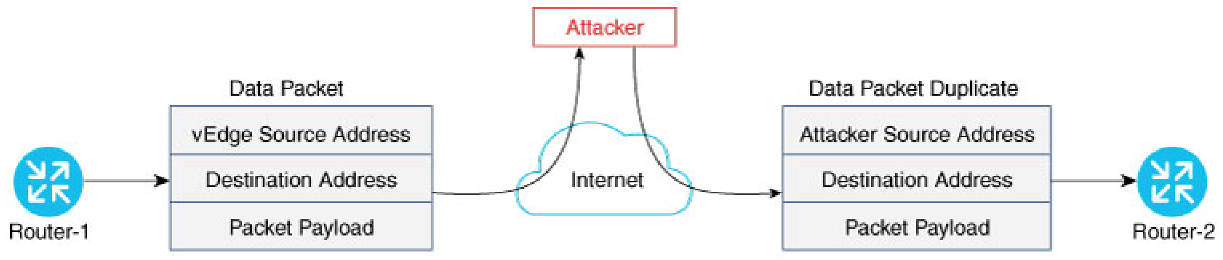
A second component that contributes to data packet integrity is the modifications to ESP to mimic AH. This modification performs a checksum that includes calculating the checksum over all the fields in the packet—the payload, the inner header, and also all the non-mutable fields in the outer IP header. AH places the resultant HMAC-SHA1 hash into the last field of the packet. The receiving device performs the same checksum, and accepts packets whose checksums match. In the figure below, the center stack illustrates the encapsulation performed by the modified version of ESP. ESP again encrypts the inner headers, payload, MPLS label (if present), and ESP trailer fields, and now mimics AH by authenticating the entire packet—the outer IP and UDP headers, the ESP header, the MPLS label (if present), the original packet, and the ESP trailer—and places its calculated HMAC-SHA1 hash into the ICV checksum field at the end of the packet.

For situations in which data packet authentication is not required, you can disable data packet authentication altogether. In this case, data packets are processed just by ESP, which encrypts the original packet, the MPLS label (if present), and the ESP trailer. This scheme is illustrated in the right stack in the figure below.

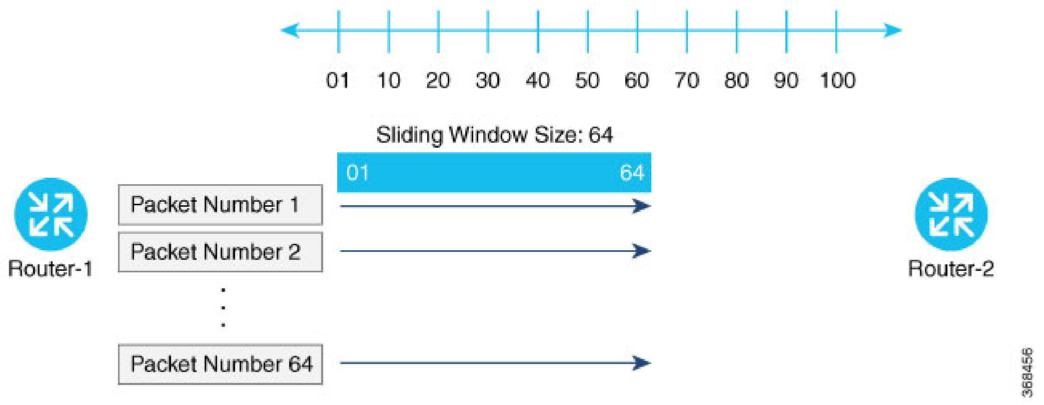


Note that Cisco SD-WAN devices exchange not only the encryption key (which is symmetric), but also the authentication key that is used to generate the HMAC-SHA1 digest. Both are distributed as part of the TLOC properties for a router.

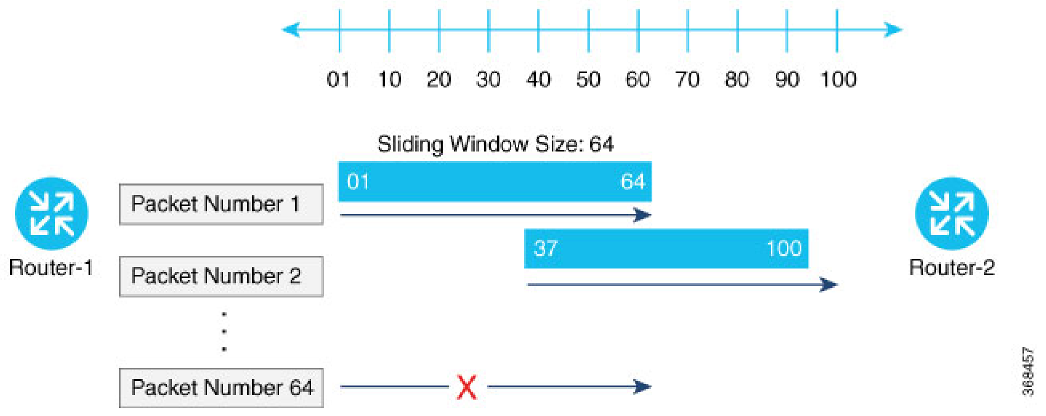
Even though the IPsec connections over which data traffic is exchanged are secure, they often travel across a public network space, such as the Internet, where it is possible for a hacker to launch a replay attack (also called a man-in-the-middle, or MITM, attack) against the IPsec connection. In this type of attack, an adversary tampers with the data traffic by inserting a copy of a message that was previously sent by the source. If the destination cannot distinguish the replayed message from a valid message, it may authenticate the adversary as the source or may incorrectly grant to the adversary unauthorized access to resources or services.



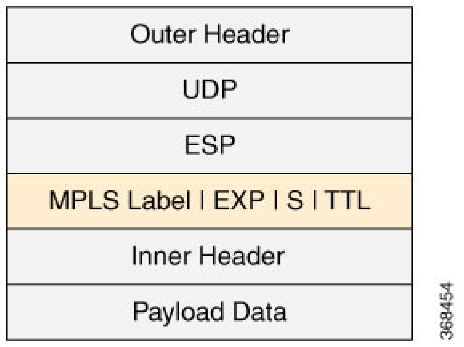
As a counter to such attacks, the Cisco SD-WAN overlay network software implements the IPsec anti-replay protocol. This protocol consists of two components, both of which protect the integrity of a data traffic stream. The first component is to associate sequence numbers with each data packets. The sender inserts a sequence number into each IPsec packet, and the destination checks the sequence number, accepting only packets with unique, non-duplicate sequence numbers. The second component is a sliding window, which defines a range of sequence numbers that are current. The sliding window has a fixed length. The destination accepts only packets whose sequence numbers fall within the current range of values in the sliding window, and it drops all others. A sliding window is used rather than accepting only packets whose sequence number is larger than the last known sequence number, because packets often do not arrive in order.



When the destination receives a packet whose sequence number is larger than the highest number in the sliding window, it slides the window to the right, thus changing the range of valid sequences numbers it will accept. This scheme protects against an MITM type of attack because, by choosing the proper window size, you can ensure that if a duplicate packet is inserted into the traffic stream, its sequence number will either be within the current range but will be a duplicate, or it will be smaller than the lowest current value of the sliding window. Either way, the destination will drop the duplicate packet. So, the sequence numbering combined with a sliding window provide protection against MITM type of attacks and ensure the integrity of the data stream flowing within the IPsec connection.



Carrying VPN Information in Data Packets



For enterprise-wide VPNs, Cisco SD-WAN devices support MPLS extensions to data packets that are transported within IPsec connections. The figure to the right shows the location of the MPLS information in the data packet header. These extensions provide the security for the network segmentation (that is, for the VPNs) that is needed to support multi-tenancy in a branch or segmentation in a campus. The Cisco SD-WAN implementation uses IPsec UDP-based overlay network layer protocol encapsulation as defined in RFC 4023. The security is provided by including the Initialization Vector (IV) at the beginning of the payload data in the ESP header. The IV value is calculated by the AES-256 cipher block chaining (CBC).

CHAPTER 3

Configure Security Parameters

This section describes how to change security parameters for the control plane and the data plane in the Cisco SD-WAN overlay network.

Configure Control Plane Security Parameters

By default, the control plane uses DTLS as the protocol that provides privacy on all its tunnels. DTLS runs over UDP.

You can change the control plane security protocol to TLS, which runs over TCP. The primary reason to use TLS is that, if you consider the vSmart controller to be a server, firewalls protect TCP servers better than UDP servers.

You configure the control plane tunnel protocol on a vSmart controller:

```
vSmart(config)# security control protocol tls
```

With this change, all control plane tunnels between the vSmart controller and the routers and between the controller and vManage use TLS. Control plane tunnels to vBond orchestrators always use DTLS, because these connections must be handled by UDP.

In a domain with multiple vSmart controllers, when you configure TLS on one of the vSmart controllers, all control plane tunnels from that controller to the other controllers use TLS. Said another way, TLS always takes precedence over DTLS. However, from the perspective of the other vSmart controllers, if you have not configured TLS on them, they use TLS on the control plane tunnel only to that one vSmart controller, and they use DTLS tunnels to all the other vSmart controllers and to all their connected routers. To have all vSmart controllers use TLS, configure it on all of them.

By default, the vSmart controller listens on port 23456 for TLS requests. To change this:

```
vSmart(config)# security control tls-port number
```

The port can be a number from 1025 through 65535.

To display control plane security information, use the **show control connections** command on the vSmart controller. For example:

```
vSmart-2# show control connections
```

```
                                                                           PEER                 PEER
PEER    PEER     PEER            SITE   DOMAIN  PEER         PRIVATE  PEER          PUBLIC
TYPE    PROTOCOL SYSTEM IP       ID     ID      PRIVATE IP   PORT     PUBLIC IP     PORT
REMOTE COLOR  STATE  UPTIME
-------------------------------------------------------------------------------------------
vedge   dtls     172.16.255.11  100    1       10.0.5.11    12346    10.0.5.11     12346
lte           up    0:07:48:58
vedge   dtls     172.16.255.21  100    1       10.0.5.21    12346    10.0.5.21     12346
lte           up    0:07:48:51
vedge   dtls     172.16.255.14  400    1       10.1.14.14   12360    10.1.14.14    12360
lte           up    0:07:49:02
vedge   dtls     172.16.255.15  500    1       10.1.15.15   12346    10.1.15.15    12346
default       up    0:07:47:18
vedge   dtls     172.16.255.16  600    1       10.1.16.16   12346    10.1.16.16    12346
default       up    0:07:41:52
vsmart  tls      172.16.255.19  100    1       10.0.5.19    12345    10.0.5.19     12345
default       up    0:00:01:44
vbond   dtls     -              0      0       10.1.14.14   12346    10.1.14.14    12346
default       up    0:07:49:08

vSmart-2# control connections

                                                                            PEER                 PEER
PEER     PEER     PEER            SITE   DOMAIN  PEER         PRIVATE  PEER          PUBLIC
TYPE     PROTOCOL SYSTEM IP       ID     ID      PRIVATE IP   PORT     PUBLIC IP     PORT
 REMOTE COLOR  STATE   UPTIME
-------------------------------------------------------------------------------------------
vedge    tls      172.16.255.11  100    1       10.0.5.11    12345    10.0.5.11     12345
 lte           up     0:00:01:18
vedge    tls      172.16.255.21  100    1       10.0.5.21    12345    10.0.5.21     12345
 lte           up     0:00:01:18
vedge    tls      172.16.255.14  400    1       10.1.14.14   12345    10.1.14.14    12345
 lte           up     0:00:01:18
vedge    tls      172.16.255.15  500    1       10.1.15.15   12345    10.1.15.15    12345
 default       up     0:00:01:18
vedge    tls      172.16.255.16  600    1       10.1.16.16   12345    10.1.16.16    12345
 default       up     0:00:01:18
vsmart   tls      172.16.255.20  200    1       10.0.12.20   23456    10.0.12.20    23456
 default       up     0:00:01:32
vbond    dtls     -              0      0       10.1.14.14   12346    10.1.14.14    12346
 default       up     0:00:01:33
```

Configure DTLS on vManage

If you configure the vManage to use TLS as the control plane security protocol, you must enable port forwarding on your NAT. If you are using DTLS as the control plane security protocol, you do not need to do anything.

The number of ports forwarded depends on the number of vdaemon processes running on the vManage. To display information about these processes and about and the number of ports that are being forwarded, use the **show control summary** command shows that four vdaemon processes are running:

```
vManage# show control summary
          VBOND     VMANAGE    VSMART    VEDGE
INSTANCE  COUNTS    COUNTS     COUNTS    COUNTS
------------------------------------------------
0           2         0           2        7
1           2         0           0        5
2           2         0           0        5
3           2         0           0        4
```

To see the listening ports, use the **show control local-properties** command:

```
vManage# show control local-properties

organization-name            Cisco SD-WAN Inc Test
certificate-status           Installed
root-ca-chain-status         Installed

certificate-validity         Valid
certificate-not-valid-before May 20 00:00:00 2015 GMT
certificate-not-valid-after  May 20 23:59:59 2016 GMT

dns-name                     vbond.cisco.com
site-id                      5000
domain-id                    0
protocol                     dtls
tls-port                     23456
...
...
...
number-active-wan-interfaces 1

                PUBLIC       PUBLIC PRIVATE       PRIVATE
  ADMIN  OPERATION LAST
INDEX INTERFACE IP           PORT   IP            PORT    VSMARTS  VMANAGES COLOR  CARRIER
  STATE  STATE     CONNECTION
------------------------------------------------------------------------------------------
0     eth0      72.28.108.37 12361  172.16.98.150 12361   2        0        silver default
  up     up       0:00:00:08
```

This output shows that the listening TCP port is 23456. If you are running vManage behind a NAT, you should open the following ports on the NAT device:

- 23456 (base - instance 0 port)
- 23456 + 100 (base + 100)
- 23456 + 200 (base + 200)
- 23456 + 300 (base + 300)

Note that the number of instances is the same as the number of cores you have assigned for the vManage, up to a maximum of 8.

Configure Data Plane Security Parameters

In the data plane, IPsec is enabled by default on all routers, and by default IPsec tunnel connections use a modified version of the Encapsulating Security Payload (ESP) protocol for authentication on IPsec tunnels. On the routers, you can change the type of authentication, the IPsec rekeying timer, and the size of the IPsec anti-replay window.

Configure Allowed Authentication Types

By default, IPsec tunnel connections use a modified version of the Encapsulating Security Payload (ESP) protocol for authentication. To modify the negotiated authentication types or to disable authentication, use the following command:

```
Device(config)# security ipsec authentication-type (ah-sha1-hmac | ah-no-id | sha1-hmac |
| none)
```

By default, IPsec tunnel connections use AES-GCM-256, which provides both encryption and authentication.

Configure each authentication type with a separate **security ipsec authentication-type** command. The command options map to the following authentication types, which are listed in order from most strong to least strong:

Note The `sha1` in the configuration options is used for historical reasons. The authentication options indicate over how much of the packet integrity checking is done. They do not specify the algorithm that checks the integrity. Except for the encryption of multicast traffic, the authentication algorithms supported by Cisco SD-WAN do not use SHA1.

- **ah-sha1-hmac** enables encryption and encapsulation using ESP. However, in addition to the integrity checks on the ESP header and payload, the checks also include the outer IP and UDP headers. Hence, this option supports an integrity check of the packet similar to the Authentication Header (AH) protocol. All integrity and encryption is performed using AES-256-GCM.
- **ah-no-id** enables a mode that is similar to **ah-sha1-hmac**, however the ID field of the outer IP header is ignored. This option accommodates some non-Cisco SD-WAN devices, including the Apple AirPort Express NAT, that have a bug that causes the ID field in the IP header, a non-mutable field, to be modified. Configure the **ah-no-id** option in the list of authentication types to have the Cisco SD-WAN AH software ignore the ID field in the IP header so that the Cisco SD-WAN software can work in conjunction with these devices.
- **sha1-hmac** enables ESP encryption and integrity checking.
- **none** maps to no authentication. This option should only be used if it is required for temporary debugging. You can also choose this option in situations where data plane authentication and integrity are not a concern. Cisco does not recommend using this option for production networks.

For information about which data packet fields are affected by these authentication types, see Data Plane Integrity, on page 13.

Cisco IOS XE SD-WAN devices and Cisco vEdge devices advertise their configured authentication types in their TLOC properties. The two routers on either side of an IPsec tunnel connection negotiate the authentication to use on the connection between them, using the strongest authentication type that is configured on both of the routers. For example, if one router advertises the `ah-sha1-hmac` and `ah-no-id` types, and a second router advertises the `ah-no-id` type, the two routers negotiate to use `ah-no-id` on the IPsec tunnel connection between them. If no common authentication types are configured on the two peers, no IPsec tunnel is established between them.

The encryption algorithm on IPsec tunnel connections depends on the type of traffic:

- For unicast traffic, the encryption algorithm is AES-256-GCM.
- For multicast traffic, the encryption algorithm is AES-256-CBC with SHA1-HMAC.

When the IPsec authentication type is changed, the AES key for the data path is changed.

Change the Rekeying Timer

Before Cisco IOS XE SD-WAN devices and Cisco vEdge devices can exchange data traffic, they set up a secure authenticated communications channel between them. The routers use IPSec tunnels between them as the channel, and the AES-256 cipher to perform encryption. Each router generates a new AES key for its data path periodically.

By default, a key is valid for 86400 seconds (24 hours), and the timer range is 10 seconds through 1209600 seconds (14 days). To change the rekey timer value:

```
Device(config)# security ipsec
rekey seconds
```

The configuration looks like this:

```
security
    ipsec
      rekey seconds
    !
```

If you want to generate new IPsec keys immediately, you can do so without modifying the configuration of the router. To do this, issue the **request security ipsec-rekey** command on the compromised router.

For example, the following output shows that the local SA has a Security Parameter Index (SPI) of 256:

```
Device# show ipsec local-sa

                                          SOURCE           SOURCE
TLOC ADDRESS     TLOC COLOR       SPI     IP               PORT    KEY HASH
------------------------------------------------------------------------------
172.16.255.15    lte              256     10.1.15.15       12346   *****b93a
```

A unique key is associated with each SPI. If this key is compromised, use the **request security ipsec-rekey** command to generate a new key immediately. This command increments the SPI. In our example, the SPI changes to 257 and the key associated with it is now used:

```
Device# request security ipsec-rekey
Device# show ipsec local-sa
                                          SOURCE           SOURCE
TLOC ADDRESS     TLOC COLOR       SPI     IP               PORT    KEY HASH
------------------------------------------------------------------------------
172.16.255.15    lte              257     10.1.15.15       12346   *****b93a
```

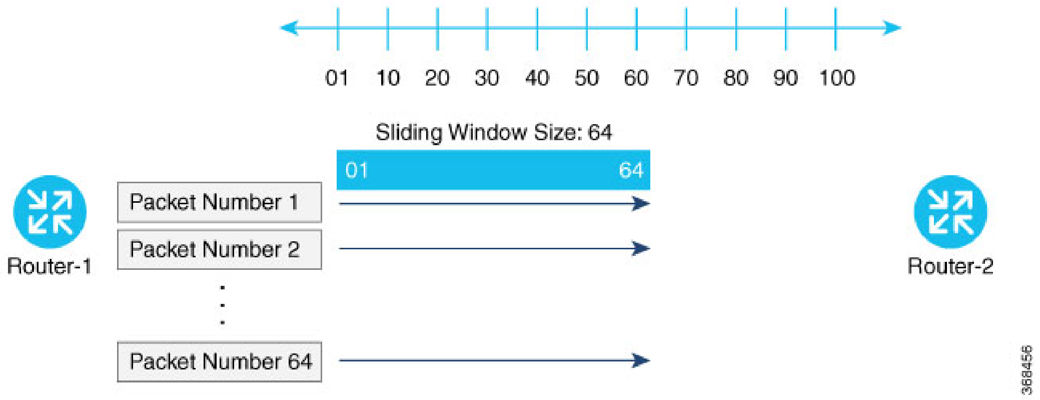
After the new key is generated, the router sends it immediately to the vSmart(s) using DTLS or TLS. The vSmart(s) send the key to the peer routers. The routers begin using it as soon as they receive it. Note that the key associated with the old SPI (256) will continue to be used for a short period of time, until it times out.

To stop using the old key immediately, issue the **request security ipsec-rekey** command twice, in quick succession. This sequence of commands removes both SPI 256 and 257 and sets the SPI to 258. The router then uses the associated key of SPI 258. Note, however, that some packets will be dropped for a short period of time, until all the remote routers learn the new key.

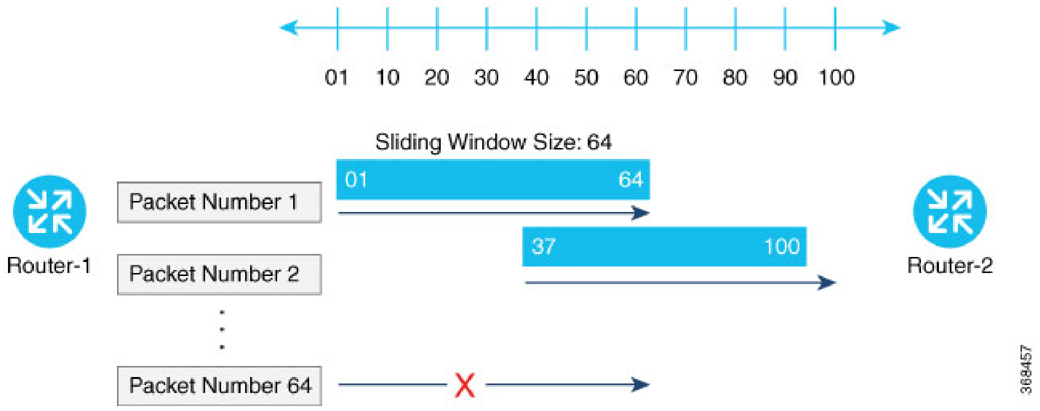
```
Device# request security ipsec-rekey
Device# request security ipsec-rekey
Device# ipsec local-sa
                                          SOURCE           SOURCE
TLOC ADDRESS     TLOC COLOR       SPI     IP               PORT    KEY HASH
------------------------------------------------------------------------------
172.16.255.15    lte              258     10.1.15.15       12346   *****b93a
```

Change the Size of the Anti-Replay Window

IPsec authentication provides anti-replay protection by assigning a unique sequence number to each packet in a data stream. This sequence numbering protects against an attacker duplicating data packets. With anti-replay protection, the sender assigns monotonically increasing sequence numbers, and the destination checks these sequence numbers to detect duplicates. Because packets often do not arrive in order, the destination maintains a sliding window of sequence numbers that it will accept.



Packets with sequence numbers that fall to the left of the sliding window range are considered old or duplicates, and the destination drops them. The destination tracks the highest sequence number it has received, and adjusts the sliding window when it receives a packet with a higher value.



By default, the sliding window is set to 512 packets. It can be set to any value between 64 and 4096 that is a power of 2 (that is, 64, 128, 256, 512, 1024, 2048, or 4096). To modify the anti-replay window size, use the **replay-window** command, specifying the size of the window:

```
Device(config)# security ipsec replay-window
number
```

The configuration looks like this:

```
security
   ipsec
     replay-window number
   !
!
```

To help with QoS, separate replay windows are maintained for each of the first eight traffic channels. The configured replay window size is divided by eight for each channel.

If QoS is configured on a router, that router might experience a larger than expected number of packet drops as a result of the IPsec anti-replay mechanism, and many of the packets that are dropped are legitimate ones. This occurs because QoS reorders packets, giving higher-priority packets preferential treatment and delaying lower-priority packets. To minimize or prevent this situation, you can do the following:

- Increase the size of the anti-replay window.
- Engineer traffic onto the first eight traffic channels to ensure that traffic within a channel is not reordered.

Configure IKE-Enabled IPsec Tunnels

To securely transfer traffic from the overlay network to a service network, you can configure IPsec tunnels that run the Internet Key Exchange (IKE) protocol. IKE-enabled IPsec tunnels provide authentication and encryption to ensure secure packet transport.

You create an IKE-enabled IPsec tunnel by configuring an IPsec interface. IPsec interfaces are logical interfaces, and you configure them just like any other physical interface. You configure IKE protocol parameters on the IPsec interface, and you can configure other interface properties.

Note Cisco recommends using IKE Version 2.

Note From Cisco SD-WAN 19.2.x release onwards, the pre-shared key needs to be at least 16 bytes in length. The IPsec tunnel establishment fails if the key size is less than 16 characters when the router is upgraded to version 19.2.

The Cisco SD-WAN software supports IKE, Version 1, as defined in RFC 2409, *Internet Key Exchange*, and IKE, Version 2, as defined in RFC 7296, *Internet Key Exchange Protocol, Version 2*.

One use for IPsec tunnels is to allow vEdge Cloud router VM instances running on Amazon AWS to connect to the Amazon virtual private cloud (VPC). You must configure IKE Version 1 on these routers.

Note Cisco vEdge devices support only route-based VPNs in an IPSec configuration because these devices cannot define traffic selectors in the encryption domain.

Configure an IPsec Tunnel

To configure an IPsec tunnel interface for secure transport traffic from a service network, you create a logical IPsec interface:

```
vEdge(config)# vpn vpn-id interface ipsec number
vEdge(config-interface-ipsec)# ip address ipv4-prefix/length
vEdge(config-interface-ipsec)# tunnel-source ip-address | tunnel-source-interface
interface-name)
```

```
vEdge(config-interface-ipsec)# tunnel-destination ipv4-address
vEdge(config-interface-ipsec)# no shutdown
```

You can create the IPsec tunnel in the transport VPN (VPN 0) and in any service VPN (VPN 1 through 65530, except for 512).

The IPsec interface has a name in the format **ipsec** number, where *number* can be from 1 through 255.

Each IPsec interface must have an IPv4 address. This address must be a /30 prefix. All traffic in the VPN that is within this IPv4 prefix is directed to a physical interface in VPN 0 to be sent securely over an IPsec tunnel.

To configure the source of the IPsec tunnel on the local device, you can specify either the IP address of the physical interface (in the **tunnel-source** command) or the name of the physical interface (in the **tunnel-source-interface** command). Ensure that the physical interface is configured in VPN 0.

To configure the destination of the IPsec tunnel, specify the IP address of the remote device in the **tunnel-destination** command.

The combination of a source address (or source interface name) and a destination address defines a single IPsec tunnel. Only one IPsec tunnel can exist that uses a specific source address (or interface name) and destination address pair.

Configure an IPsec Static Route

To direct traffic from the service VPN to an IPsec tunnel in the transport VPN (VPN 0), you configure an IPsec-specific static route in a service VPN (a VPN other than VPN 0 or VPN 512) :

```
vEdge(config)# vpn vpn-id
vEdge(config-vpn)# ip ipsec-route prefix/length vpn 0 interface
ipsec number [ipsec number2]
```

The VPN ID is that of any service VPN (VPN 1 through 65530, except for 512).

prefix/length is the IP address or prefix, in decimal four-part-dotted notation, and prefix length of the IPsec-specific static route.

The interface is the IPsec tunnel interface in VPN 0. You can configure one or two IPsec tunnel interfaces. If you configure two, the first is the primary IPsec tunnel, and the second is the backup. With two interfaces, all packets are sent only to the primary tunnel. If that tunnel fails, all packets are then sent to the secondary tunnel. If the primary tunnel comes back up, all traffic is moved back to the primary IPsec tunnel.

Enable IKE Version 1

When you create an IPsec tunnel on a vEdge router, IKE Version 1 is enabled by default on the tunnel interface. The following properties are also enabled by default for IKEv1:

- Authentication and encryption—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity
- Diffie-Hellman group number—16
- Rekeying time interval—4 hours
- SA establishment mode—Main

By default, IKEv1 uses IKE main mode to establish IKE SAs. In this mode, six negotiation packets are exchanged to establish the SA. To exchange only three negotiation packets, enable aggressive mode:

Note IKE aggressive mode with pre-shared keys should be avoided wherever possible. Otherwise a strong pre-shared key should be chosen.

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# mode aggressive
```

By default, IKEv1 uses Diffie-Hellman group 16 in the IKE key exchange. This group uses the 4096-bit more modular exponential (MODP) group during IKE key exchange. You can change the group number to 2 (for 1024-bit MODP), 14 (2048-bit MODP), or 15 (3072-bit MODP):

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# group number
```

By default, IKE key exchange uses AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity. You can change the authentication:

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# cipher-suite suite
```

The authentication *suite* can be one of the following:

- **aes128-cbc-sha1**—AES-128 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity
- **aes128-cbc-sha2**—AES-128 advanced encryption standard CBC encryption with the HMAC-SHA256 keyed-hash message authentication code algorithm for integrity
- **aes256-cbc-sha1**—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity; this is the default.
- **aes256-cbc-sha2**—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA256 keyed-hash message authentication code algorithm for integrity

By default, IKE keys are refreshed every 1 hours (3600 seconds). You can change the rekeying interval to a value from 30 seconds through 14 days (1209600 seconds). It is recommended that the rekeying interval be at least 1 hour.

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# rekey seconds
```

To force the generation of new keys for an IKE session, issue the **request ipsec ike-rekey** command.

```
vEdge(config)# vpn vpn-id interfaceipsec number ike
```

For IKE, you can also configure preshared key (PSK) authentication:

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# authentication-type pre-shared-key pre-shared-secret password
```

password is the password to use with the preshared key. It can be an ASCII or a hexadecimal string from 1 through 127 characters long.

If the remote IKE peer requires a local or remote ID, you can configure this identifier:

```
vEdge(config)# vpn vpn-id interface ipsec number ike authentication-type
vEdge(config-authentication-type)# local-id id
vEdge(config-authentication-type)# remote-id id
```

The identifier can be an IP address or any text string from 1 through 63 characters long. By default, the local ID is the tunnel's source IP address and the remote ID is the tunnel's destination IP address.

Enable IKE Version 2

When you configure an IPsec tunnel to use IKE Version 2, the following properties are also enabled by default for IKEv2:

- Authentication and encryption—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity
- Diffie-Hellman group number—16
- Rekeying time interval—4 hours

By default, IKEv2 uses Diffie-Hellman group 16 in the IKE key exchange. This group uses the 4096-bit more modular exponential (MODP) group during IKE key exchange. You can change the group number to 2 (for 1024-bit MODP), 14 (2048-bit MODP), or 15 (3072-bit MODP):

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# group number
```

By default, IKE key exchange uses AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity. You can change the authentication:

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# cipher-suite suite
```

The authentication *suite* can be one of the following:

- **aes128-cbc-sha1**—AES-128 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity
- **aes128-cbc-sha2**—AES-128 advanced encryption standard CBC encryption with the HMAC-SHA256 keyed-hash message authentication code algorithm for integrity
- **aes256-cbc-sha1**—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA1 keyed-hash message authentication code algorithm for integrity; this is the default.
- **aes256-cbc-sha2**—AES-256 advanced encryption standard CBC encryption with the HMAC-SHA256 keyed-hash message authentication code algorithm for integrity

By default, IKE keys are refreshed every 4 hours (14,400 seconds). You can change the rekeying interval to a value from 30 seconds through 14 days (1209600 seconds):

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)#  rekey seconds
```

To force the generation of new keys for an IKE session, issue the **request ipsec ike-rekey** command.

For IKE, you can also configure preshared key (PSK) authentication:

```
vEdge(config)# vpn vpn-id interface ipsec number ike
vEdge(config-ike)# authentication-type pre-shared-key pre-shared-secret password
```

password is the password to use with the preshared key. It can be an ASCII or a hexadecimal string, or it can be an AES-encrypted key.

If the remote IKE peer requires a local or remote ID, you can configure this identifier:

```
vEdge(config)# vpn vpn-id interface ipsec number ike authentication-type
vEdge(config-authentication-type)# local-id id
vEdge(config-authentication-type)# remote-id id
```

The identifier can be an IP address or any text string from 1 through 64 characters long. By default, the local ID is the tunnel's source IP address and the remote ID is the tunnel's destination IP address.

Configure IPsec Tunnel Parameters

By default, the following parameters are used on the IPsec tunnel that carries IKE traffic:

- Authentication and encryption—AES-256 algorithm in GCM (Galois/counter mode)
- Rekeying interval—4 hours
- Replay window—32 packets

You can change the encryption on the IPsec tunnel to the AES-256 cipher in CBC (cipher block chaining mode, with HMAC-SHA1-96 keyed-hash message authentication or to null, to not encrypt the IPsec tunnel used for IKE key exchange traffic:

```
vEdge(config-interface-ipsecnumber)# ipsec
vEdge(config-ipsec)# cipher-suite (aes256-cbc-sha1 | aes256-gcm | null-sha1)
```

By default, IKE keys are refreshed every 4 hours (14,400 seconds). You can change the rekeying interval to a value from 30 seconds through 14 days (1209600 seconds):

```
vEdge(config-interface-ipsecnumber)# ipsec
vEdge(config-ipsec)#  rekey seconds
```

To force the generation of new keys for an IPsec tunnel, issue the **request ipsec ipsec-rekey** command.

By default, perfect forward secrecy (PFS) is enabled on IPsec tunnels, to ensure that past sessions are not affected if future keys are compromised. PFS forces a new Diffie-Hellman key exchange, by default using the 4096-bit Diffie-Hellman prime module group. You can change the PFS setting:

```
vEdge(config-interface-ipsecnumber)# ipsec
vEdge(config-ipsec)# perfect-forward-secrecy pfs-setting
```

pfs-setting can be one of the following:

- **group-2**—Use the 1024-bit Diffie-Hellman prime modulus group.
- **group-14**—Use the 2048-bit Diffie-Hellman prime modulus group.
- **group-15**—Use the 3072-bit Diffie-Hellman prime modulus group.
- **group-16**—Use the 4096-bit Diffie-Hellman prime modulus group. This is the default.
- **none**—Disable PFS.

By default, the IPsec replay window on the IPsec tunnel is 512 bytes. You can set the replay window size to 64, 128, 256, 512, 1024, 2048, or 4096 packets:

```
vEdge(config-interface-ipsecnumber)# ipsec
vEdge(config-ipsec)# replay-window number
```

Modify IKE Dead-Peer Detection

IKE uses a dead-peer detection mechanism to determine whether the connection to an IKE peer is functional and reachable. To implement this mechanism, IKE sends a Hello packet to its peer, and the peer sends an acknowledgment in response. By default, IKE sends Hello packets every 10 seconds, and after three unacknowledged packets, IKE declares the neighbor to be dead and tears down the tunnel to the peer. Thereafter, IKE periodically sends a Hello packet to the peer, and re-establishes the tunnel when the peer comes back online.

You can change the liveness detection interval to a value from 0 through 65535, and you can change the number of retries to a value from 0 through 255.

Note For transport VPNs, the liveness detection interval is converted to seconds by using the following formula:

`Interval for retransmission attempt number N = interval * 1.8^(N-1)`

For example, if the interval is set to 10 and retries to 5, the detection interval increases as follows:

- Attempt 1: 10 * 1.8^{1-1}= 10 seconds
- Attempt 2: 10 * 1.8^{2-1}= 18 seconds
- Attempt 3: 10 * 1.8^{3-1}= 32.4 seconds
- Attempt 4: 10 * 1.8^{4-1}= 58.32 seconds
- Attempt 5: 10 * 1.8^{5-1}= 104.976 seconds

```
vEdge(config-interface-ipsecnumber)# dead-peer-detection interval retries number
```

Configure Other Interface Properties

For IPsec tunnel interfaces, you can configure only the following additional interface properties:

```
vEdge(config-interface-ipsec)# mtu bytes
vEdge(config-interface-ipsec)# tcp-mss-adjust bytes
```

CHAPTER 4

Enterprise Firewall

Cisco's Enterprise Firewall uses a flexible and easily understood zone-based model for traffic inspection, compared to the older interface-based model.

A firewall policy is a type of localized security policy that allows stateful inspection of TCP, UDP, and ICMP data traffic flows. Traffic flows that originate in a given zone are allowed to proceed to another zone based on the policy between the two zones. A zone is a grouping of one or more VPNs. Grouping VPNs into zones allows you to establish security boundaries in your overlay network so that you can control all data traffic that passes between zones.

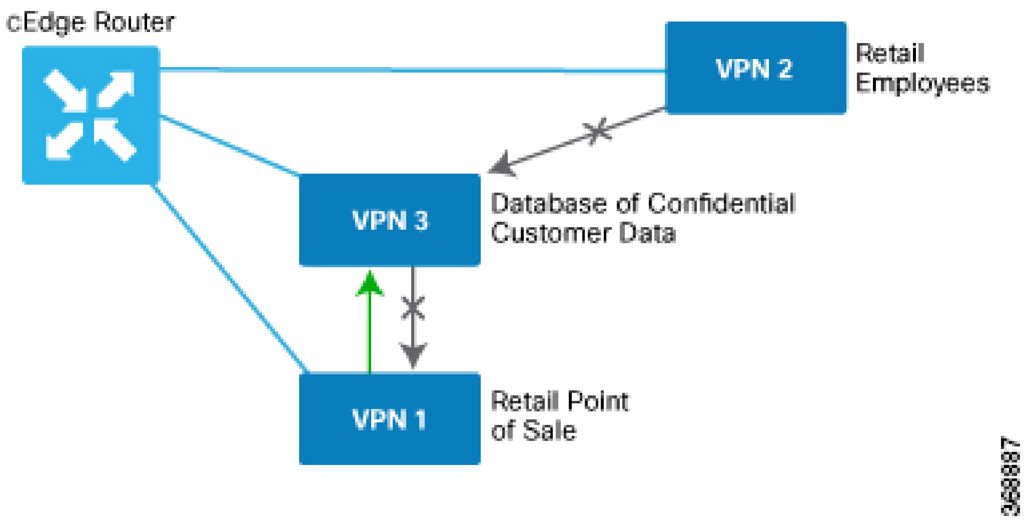
Zone configuration consists of the following components:

- Source zone—A grouping of VPNs where the data traffic flows originate. A VPN can be part of only one zone.
- Destination zone—A grouping of VPNs where the data traffic flows terminate. A VPN can be part of only one zone.
- Firewall policy—A security policy, similar to a localized security policy, that defines the conditions that the data traffic flow from the source zone must match to allow the flow to continue to the destination zone. Firewall policies can match IP prefixes, IP ports, the protocols TCP, UDP, and ICMP. Matching flows for prefixes, ports, and protocols can be accepted or dropped, and the packet headers can be logged. Nonmatching flows are dropped by default.
- Zone pair—A container that associates a source zone with a destination zone and that applies a firewall policy to the traffic that flows between the two zones.

Matching flows that are accepted can be processed in two different ways:

- Inspect—The packet's header can be inspected to determine its source address and port.
- Pass—Allow the packet to pass to the destination zone without inspecting the packet's header at all.

The following figure shows a simple scenario in which three VPNs are configured on a XE SD-WAN Router. One of the VPNs, VPN 3, has shared resources that you want to restrict access to. These resources could be printers or confidential customer data. For the remaining two VPNs in this scenario, only users in one of them, VPN 1, are allowed to access the resources in VPN 3, while users in VPN 2 are denied access to these resources. In this scenario, we want data traffic to flow from VPN 1 to VPN 3, but we do not want traffic to flow in the other direction, from VPN 3 to VPN 1.



Firewall policies perform stateful inspection of TCP, UDP, and ICMP flows between zones. They examine the source and destination IP addresses and ports in the packet headers, as well as the packet's protocol. Then, based on the configured zone-based policy, they allow traffic to pass between the zones or they drop the traffic.

The implementation of firewall policies varies slightly to that of localized security policy. Where you configure and apply localized security policy based only on VPNs, you configure and apply firewall policies to one or more VPNs that have been grouped into a zone. You activate localized security policy by applying it to individual interfaces on the XE SD-WAN Routers. When you activate firewall policies, they apply to the specific VPNs in the zones, without regard to any specific interfaces.

vEdge routers provide Application Layer Gateway (ALG) FTP support with Network Address Translation – Direct Internet Access (NAT-DIA), Service NAT, and Enterprise Firewall. Service NAT support is added for FTP ALG on the client and not on the FTP Server.

- Configure Firewall Policies, on page 30
- Monitor Enterprise Firewall, on page 31
- Zone-Based Firewall Configuration Examples, on page 31

Configure Firewall Policies

In Cisco vManage, you configure firewall policies from the **Configuration** > **Security** screen, using a policy configuration wizard. In the CLI, you configure these firewalls on the router.

General Cisco vManage Configuration Procedure

To configure firewall policies, use the policy configuration wizard. The wizard is a UI policy builder that lets you configure policy components:

- Create Lists—Create lists that group together related items and that you call in the match condition of a firewall policy.
- Firewall Policy—Define the match and action conditions of the firewall policy.
- Apply Configuration—Define zone pairs.

You must configure all these components to create a firewall policy. If you are modifying an existing firewall, you can skip a component by clicking the **Next** button at the bottom of the screen. To return to a component, click the **Back** button at the bottom of the screen.

Monitor Enterprise Firewall

You can monitor Enterprise Firewall by using the statistics created for the firewall.

To monitor Enterprise Firewall and view statistics:

1. Cisco vManage, navigate to **Monitor** > **Network**.
2. Select a device from the list of devices.
3. Under the Security Monitoring pane on the left, click **Real Time**. A pop-up screen appears with **Device Options**.
4. Click on the Search tab, and choose **Policy Zone Based Firewall Statistics** from the list to view the statistics for the firewall policies.

Note Firewall Charts and Policy statistics are not currently supported for Cisco vEdge devices from **Networ** **Firewall** dashboard. However, detailed statistics are available when you navigate to **Network** > **Real**

Zone-Based Firewall Configuration Examples

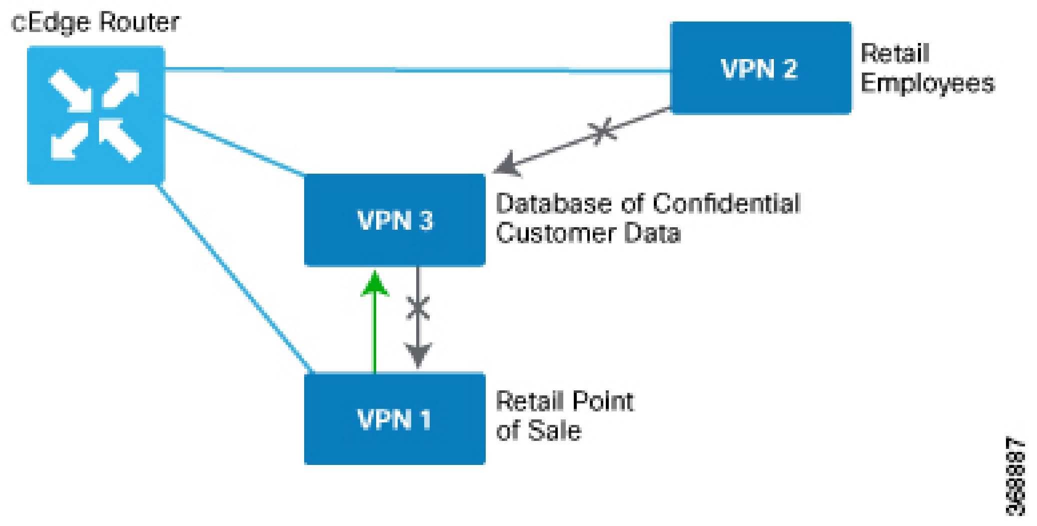
This topic provides an example of configuring a simple zone-based firewall using the CLI or vManage.

Isolating Two VPNs

In this zone-based firewall configuration example, we have a scenario where a router is connected to three service-side networks:

- Guest network that provides point-of-sale (PoS) services
- Employee network
- Network that provides shared services, including shared printers and the customer database

We want users in the employee and guest networks to be able to access the shared services, but we do not want any traffic to be exchanged between the employee and guest networks. Similarly, we do not want any traffic that originates in the shared services network to enter into either the employee network or the guest network. The following figure illustrates this scenario:



In this figure:

- VPN 1 is the guest network used for PoS services.
- VPN 2 is the network used by the enterprise's employees.
- VPN 3 contains the shared services, including printers and customer databases.

The configuration consists of three sections:

- Define the zones.
- Define the zone-based firewall policy.
- Apply the zone-based firewall policy to a source zone and destination zone pair.

CLI Configuration

First, we define the zones for this scenario:

```
vEdge(config)# policy
vEdge(config-policy)# zone pos-zone vpn 1
vEdge(config-policy)# zone employee-zone vpn 2
vEdge(config-policy)# zone services-zone vpn 3
```

In this simple example, each zone corresponds to a single VPN. If you were to later add a second VPN for a discrete group of employees (let's say this is VPN 20) and you wanted this VPN to be subject to the same firewall policy, you could simply add this VPN to the employee zone:

```
vEdge(config-policy)# zone employee-zone vpn 20
vEdge(config-policy)# show full-configuration
policy zone employee-zone
  vpn 2
  vpn 20
 !
!
```

Next, we configure the zone-based firewall policy. The policy matches all traffic that is destined for VPN 3, which is the services zone, and which has an IP prefix of 10.2.2.0/24. Because we want the policy to allow traffic to flow from VPN 1 and VPN 2 to VPN 3, but we do not want traffic to flow in the reverse direction, we set the action to **pass**.

```
vEdge(config-policy)# zone-based-policy vpn-isolation-policy(config-zone-based-policy)#
sequence 10(config-sequence)# match destination-ip 10.2.2.0/24
vEdge(config-sequence)# action pass
```

We want to drop any traffic that does not match the zone-based filrewall policy:

```
vEdge(config-zone-based-policy)# default-action drop
```

In the final step of the configuration process, we apply the zone-based firewall policy to the zones. Here is the zone pairing between the guest and PoS zone and the services zone:

```
vEdge(config-policy)# zone-pair pos-services-pairing
vEdge(config-zone-pair)# source-zone pos-zone
vEdge(config-zone-pair)# destination-zone services-zone
vEdge(config-zone-pair)# zone-policy vpn-isolation-policy
```

And here is the pairing between the employee zone and the services zone:

```
vEdge(config-policy)# zone-pair employee-services-pairing
vEdge(config-zone-pair)# source-zone employee-zone
vEdge(config-zone-pair)# destination-zone services-zone
vEdge(config-zone-pair)# zone-pair employee-services-pairing
```

Here is a view of the entire policy:

```
vEdge(config-policy)# show full-configuration
 policy
 zone employee-zone
  vpn 2
! zone pos-zone
  vpn 1
 ! zone services-zone
  vpn 3
!
zone-pair employee-services-pairing
  source-zone      employee-zone
  destination-zone services-zone
  zone-policy      vpn-isolation-policy
 !
zone-pair services-pairing
  source-zone      pos-zone
  destination-zone services-zone
  zone-policy      vpn-isolation-policy
 !
zone-based-policy vpn-isolation-policy
  sequence 10
   match
   destination-ip 10.2.2.0/24
  !
   action pass
  !
!
 default-action drop
!
!
```

vManage Configuration

To configure this zone-based firewall policy in vManage NMS:

1. Select **Configuration** > **Security**.
2. Click **Add Policy**. The zone-based firewall configuration wizard opens.

Configure data prefix groups and zones in the Create Groups of Interest screen:

1. In the left pane, select **Data Prefix**.
2. In the right pane, click **New Data Prefix List**.
3. Enter a name for the list.
4. Enter the data prefix or prefixes to include in the list.
5. Click **Add**.

Configure zones in the Create Groups of Interest screen:

1. In the left pane, select **Zones**.
2. In the right pane, click **New Zone List**.
3. Enter a name for the list.
4. Enter the number of the zone or zones to include in the list. Separate numbers with a comma.
5. Click **Add**.
6. Click **Next** to move to Zone-Based Firewall in the zone-based firewall configuration wizard.

Configure zone-based firewall policies:

1. Click **Add Configuration**, and select **Create New**.
2. Enter a name and description for the policy.
3. In the left pane, click **Add Sequence**.
4. In the right pane, click **Add Sequence Rule**.
5. Select the desired match and action conditions.
6. Click **Same Match and Actions**.
7. In the left pane, click **Default Action**.
8. Select the desired default action.
9. Click **Save Zone-Based Policy**.

Click **Next** to move to the Apply Configuration in the zone-based firewall configuration wizard.

1. Enter a name and description for the zone-based firewall zone pair.
2. Click **Add Zone Pair**.
3. In the Source Zone drop-down, select the zone from which data traffic originates.
4. In the Destination Zone drop-down, select the zone to which data traffic is sent.
5. Click **Add**.

6. Click **Save Policy**. The **Configuration** > **Security** screen is then displayed, and the zone-based firewalls table includes the newly created policy.

CHAPTER 5

IPSec Pairwise Keys Overview

Table 2: Feature History

Feature Name	Release Number	Feature Description
Secure Communication Using Pairwise IPsec Keys	Cisco SD-WAN Release 19.2.1	This feature allows private pairwise IPSec session keys to be created and installed for secure communication between IPSec devices and its peers.

IPSec Pairwise Keys feature implements controller-based key exchange protocol between device and controller.

Controller-based key exchange protocol is used to create a Gateway-to-Gateway VPN (RFC7018) in either a Full-Mesh Topology or Dynamic Full-Mesh Topology.

The network devices set up a protected control-plane connection to the controller. The controller distributes policies to network devices, which enables the network devices to communicate with each other through a secure data plane.

A pair of IPSec session keys (one encryption key and one decryption key) are configured per pair of local and remote Transport Locations (TLOC).

Supported Platforms

The following platforms are supported for IPSec Pairwise Keys feature:

- Cisco IOS XE SD-WAN devices
- Cisco vEdge devices

Pairwise Keys

Key exchange method combined with authentication policies facilitate pairwise key creation between two network devices. A controller is used to distribute keying material and policies between network devices, resulting in the devices generating private pairwise keys with each other.

IPSec devices share public values from Diffie-Hellman (DH) algorithm with the controllers. The controllers relay the DH public values to authorized peers of the IPsec, device as defined by a centralized policy.

Network devices n create and install private pairwise IPsec session keys to be used to secure communications with their peers.

IPsec Security Association Rekey

Every rekeying IPsec device generates a new DH pair and generates new IPsec security association pairs for each peer with which it is communicating. The new security association pairs are generated as a combination of the new DH private value and the DH public value of each each peer. The IPsec device distributes the new DH public value to the Controller, which forwards it to its authorized peers. Each peer continues to transmit on the existing security association until that peer starts transmitting on the new security associations.

During a simultaneous rekey up to four pairs of IPsec SAs may be temporarily created, and they converge on a single new set of IPsec SAs.

Any IPsec device may initiate a rekey due to reasons such as a local time or volume-based policy, or the counter result of a cipher counter mode Initialization Vector (IV) nearing completion.

When you configure a rekey on a local inbound security association, it triggers peer outbound and inbound security association rekey. The local outbound security association rekey is initiated after the IPSec device recieves the first packet with new Security Parameter Index (SPI) from peer.

Note A pairwise key edge device can form IPSec sessions with both pairwise and non-pairwise edge devices

Note The rekeying process requires higher control plane CPU usage, resulting in lower session scaling

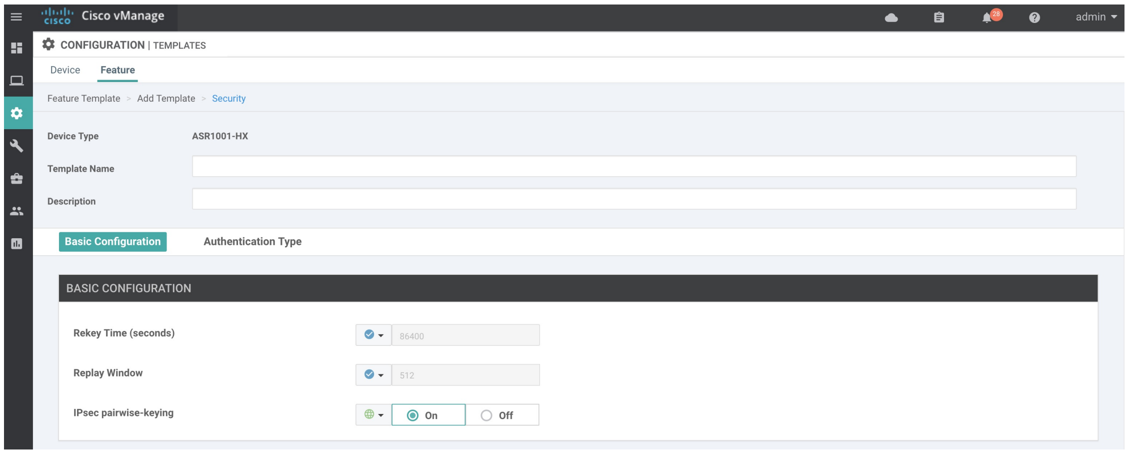
Configure IPSec Pairwise Keys

Configure IPSec Pairwise Keys Using vManage

1. In vManage NMS, select the **Configuration ▶ Templates** screen.
2. In the **Feature** tab, click **Create Template**.
3. From the **Device Model** check box, select the type of device for which you are creating the template.
4. From the **Basic Information** tab, choose **Security** template.

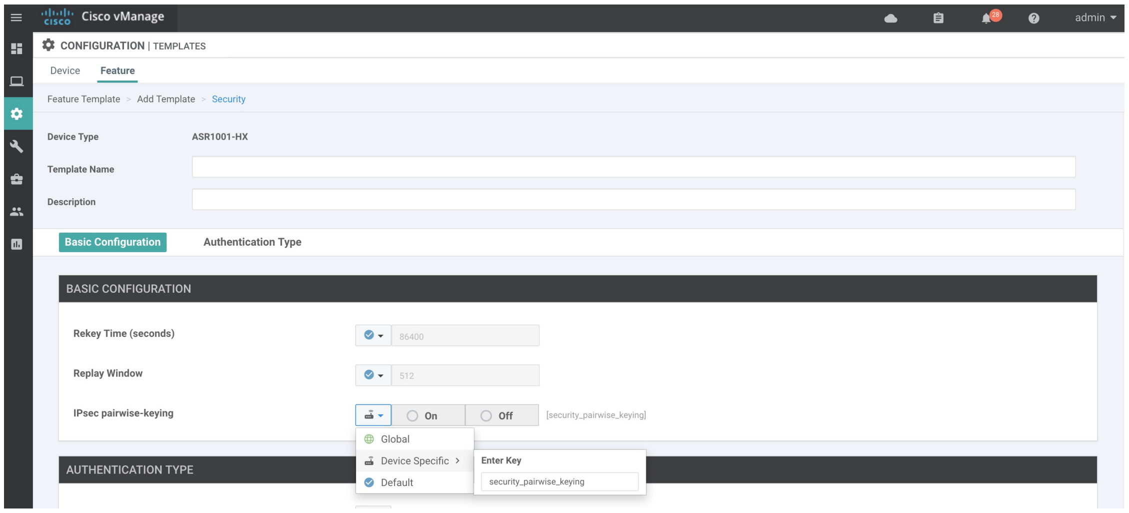
5. From theBasic Configuration tab, select On or Off from the IPsec Pairwise-Keying field..

Figure 1: IPSec Pairwise Keying



Device Type ASR1001-HX
Template Name
Description
Rekey Time (seconds) 86400
Replay Window 512
IPsec pairwise-keying [✓ On] Off

6. Alternatively, enter the pairwise key specific to the device in the **EnterKey** field.



Device Type ASR1001-HX
Template Name
Description
Rekey Time (seconds) 86400
Replay Window 512
IPsec pairwise-keying On Off [security_pairwise_keying]
Enter Key security_pairwise_keying

7. Click **Save**.

Configure Pairwise Keys and Enable Rekeying on the CLI

A pair of IPsec session keys is configured for each pair of local and remote transport locations.

The keys use AES-GCM-256 (AES_256_CBC for multicast) cipher to perform encryption. By default, a key is valid for 3600 seconds.

Configure Pairwise Keys

Use the following command to configure pairwise keys:

```
Device(config)# security ipsec pairwise-keying
```

Note You must reboot the Cisco IOS XE SD-WAN device for the private-key configuration to take effect.

Configure Rekeying for IPsec Pairwise Keys

Use the following command to configure rekeying for pairwise keys:

```
Device(config)# security ipsec pwk-sym-rekey
```

Verify IPSec Pairwise Keys on Cisco vEdge Routers

Use the following command to display IPSec pairwise keys information on Cisco vEdge Routers:

```
Device# show security-info

security-info authentication-type "AH_SHA1_HMAC SHA1_HMAC"
security-info rekey 86400
security-info replay-window 512
security-info encryption-supported "AES_GCM_256 (and AES_256_CBC for multicast)"
security-info fips-mode Enabled
security-info pairwise-keying Enabled
```

Use the following command to verify outbound connection for IPSec pairwise keys:

```
SOURCE SOURCE DEST DEST                                              REMOTE
REMOTE         AUTHENTICATION                      NEGOTIATED
       PEER          PEER
IP            PORT       IP       PORT      SPI          TUNNEL MTU TLOC ADDRESS TLOC COLOR
 USED                      KEY-HASH ENCRYPTION ALGORITHM TC SPIs KEY-HASH SPI
------------------------------------------------------------------------------------------

10.1.16.16 12366 10.1.15.15 12426 260           1441               172.16.255.15      lte
                     AH_SHA1_HMAC *****4aec      AES-GCM-256                         8
       *****d01e 1538
```

Use the following command to verify inboud connection for IPSec pairways keys:

```
Device# show ipsec inbound-connections

SOURCE  SOURCE       DEST    DEST     REMOTE           REMOTE LOCAL LOCAL  NEGOTIATED PEER
  PEER
IP      PORT          IP  PORT       TLOC ADDRESS TLOC   COLOR TLOC  ADDRESS  TLOC
COLOR ENCRYPTION  ALGORITHM  TC SPIs KEY-HASH SPI
------------------------------------------------------------------------------------------

10.1.15.15 12426 10.1.16.16 12366     172.16.255.15      lte 172.16.255.16 lte AES-GCM-256
 8 *****d01e 518
```

CHAPTER 6

Configure Single Sign-On

This chapter describes how to configure single sign-on for Cisco SD-WAN. Cisco SD-WAN supports single sign-on using Okta or Active Directory Federation Services (ADFS).

Configure Single Sign-On using Okta

Okta provides a secure identity management service that lets you connect any person with any application on any device using Single Sign-On (SSO).

Perform the following steps to configure SSO.

Enable an Identity Provider in vManage

To configure Okta SSO, you must use vManage to enable an identity provider and generate a SAML metadata file:

1. In vManage, click **Administration** > **Settings** > **Identify Provider Settings** > **Edit**.
2. Click **Enabled**.
3. Click **Click here to download the SAML metadata** and save the content in a file. This data will be used for configuring Okta.
4. In the metadata, note the following information that you will use to configure Okta with vManage:
    - Entity ID
    - Signing certificate
    - Encryption certificate
    - Logout URL
    - Login URL

Configure SSO on the Okta Website

To configure SSO on the Okta website:

1. Log on to the Okta website.
2. Create a username using your email address.
3. To add vManage as one SSO application, click on the **Admin** button on the upper right corner to go to the next page. Then check the upper left corner to make sure it shows the **Classic UI** view on Okta. If it shows the **Developer Console**, click on the down triangle to select the **Classic UI**.
4. Click on **Add Application** under **Shortcuts** to the right to go to the next page, and then click on **Create New Application** on the pop-up window. Select **Web** for the platform, and select **SAML 2.0** as the **Sign on Method**. Click **Create**.
5. Give a string as **Application name**.
6. Optional: Upload a logo, and then click **Next**.
7. On **SAML Settings** for **Single sign on URL** section, set the value to the **samlLoginResponse** URL from the downloaded metadata from the vManage UI. Check the box **Use this for Recipient URL and Destination URL**.
8. Copy the **entityID** string and paste it in the **Audience URI (SP Entity ID)** field. The value can be an IP address or the name of the vManage site.
9. For **Default RelayState**, leave empty.
10. For **Name ID format**, select **EmailAddress**.
11. For **Application username**, select **Okta username**.
12. For **Show Advanced Settings**, enter the fields as indicated below.

Table 3:

Component	Value	Configuration
Response	Signed	
Assertion Signature	Signed	
Signature Algorithm	RSA-SHA256	
Digest Algorithm	SHA256	
Assertion Encryption	Encrypted	
Encryption Algorithm	AES256-CBC	
Key Transport Algorithm	RSA-OAEP	

Component	Value	Configuration
Encryption Certificate		**a.** Copy the encryption certificate from the metadata you downloaded. **b.** Go to www.samltool.com and click on **X.509 CERTS**, paste there. Click **Format X.509 Certificate**. **c.** Make sure to remove the last empty line and then save the output (**X.509.cert with header**) into a text file **encryption.cer**. **d.** Upload the file. Mozilla Firefox may not allow you to do the upload. Instead, you can use Google Chrome. You should see the certificate information after uploading to Okta.
Enable Single Logout		Make sure this is checked.
Single Logout URL		Get from the metadata.
SP Issuer		Use the entityID from the metadata.
Signature Certificate		**a.** Obtain from the metadata. Format the signature certificate using www.samltool.com as done above. **b.** Save to a file, for example, **signing.cer** and upload.
Authentication context class	X.509 Certificate	
Honor Force Authentication	Yes	
SAML issuer ID string	SAML issuer ID string	
Attribute Statements (optional)	Field: **Name**	Value: *Username*
	Field: **Name format (optional)**	Value: Unspecified
	Field: **Value**	Value: *user.login*
Group Attribute Statements (optional)	Field: **Name**	Value: Groups
	Field: **Name format (optional)**	Value: Unspecified
	Field: **Matches regex**	Value: **.***

Note It is mandatory to use the two strings, Username and Groups, exactly as shown above. Otherwise, you ma be logged in with the default group of Basic.

13. Click **Next**.
14. For **Application Type**, check **This is an internal app that we have created** (optional).
15. Click **Finish**. This brings you to the Okta application page.
16. Click on **View Setup Instructions**.
17. Copy the IDP metadata.
18. In the vManage UI, paste the IDP metadata in vManage using **Identity Provider Settings > Upload Identity Provider Metadata**, and click **Save**.

Assign Users to the Application

To assign users to the application on the Okta website:

1. On the Okta application page, navigate to **Assignments** > **People** > **Assign**.
2. Select **Assign to people** from the drop-down menu.
3. Click on **Assign** next to the user(s) you selected and click **Done**.
4. To add a user, click on **Directory** > **Add Person** > **Save**.

Configure SSO for Active Directory Federation Services (ADFS)

Describes how to use vManage and ADFS to configure Single Sign On (SSO).

The configuration of vManage to use ADFS as IDP involved two steps:

- Step 1 - Import ADFS metadata to vManage
- Step 2- Export vManage metadata to ADFS

Step 2 can be further divided into:

- Edit and then import vManage metadata to ADFS
- Setup ADFS manually using the information from vManage metadata

Import Metadata File into ADFS

Step 1 - Import ADFS metadata to vManage:

1. Download the ADFS Metadata file, typically from the ADFS URL: `https://<your ADFS FQDN or IP>/FederationMetadata/2007-06/FederationMetadata.xml`

2. Save the file as **adfs_metadata.txt**.
3. On the vManage navigate to **Admin > Settings > Identify Provider Settings >Enable**, and then upload **adfs_metadata.txt** to vManage.

   Step 2 - Export vManage metadata to ADFS:
4. With **Identify Provider Settings**enabled, **Click here** to download SAML metadata and save into a file, which is typically `192.168.1.15_saml_metadata.xml`.
5. Edit vManage Metadata file by deleting everything from **<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">** to **</ds:Signature>**.
6. Edit vManage Metadata file by deleting everything from **<md:KeyDescriptor use="encryption">** to **</md:KeyDescriptor>**.
7. Import the new modified vManage Metadata file into ADFS, and enter the **entityID** as **`Display Name`**.
8. Click **Next** until the end.
9. Open **Edit Claim Rule**, and add the following four new custom rules in the exact sequence:

```
@RuleName = "sAMAccountName as Username" c:[Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname",
Issuer == "AD AUTHORITY"]=> issue(store = "Active Directory", types
= ("Username"), query = ";sAMAccountName;{0}", param = c.Value);

@RuleName = "sAMAccountName as NameID" c:[Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname",
Issuer == "AD AUTHORITY"] => issue(store = "Active Directory", types
=
("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"),
query = ";sAMAccountName;{0}", param = c.Value);

@RuleName = "Get User Groups and save in temp/variable" c:[Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname",
Issuer == "AD AUTHORITY"] => add(store = "Active Directory", types =
("http://temp/variable1"), query = ";tokenGroups;{0}", param =
c.Value);

@RuleName = "Parse temp/variable1 and Send Groups Membership" c:[Type
== "http://temp/variable1", Value =~ "(?i)^SSO-"] => issue(Type =
"Groups", Value = RegExReplace(c.Value, "SSO-", ""));
```

10. Verify the final result.
11. In the Active Directory, create the following two security groups: **`SSO-Netadmin`** and **`SSO-Operator`**.

Note If you are using different naming convention for the two security groups, then you have to modify the R expression value **`"(?i)^SSO-"`** in the step above.

Any active directory users who are not members of the two groups will only have **Basic** access to vManage.

Add ADFS Relying Party Trust

Before you begin

To add ADFS relying party trust using vManage:

1. Navigate to **Admin** > **Settings** > **Identify Provider Settings** > **Enable**.
2. Download the ADFS Metadata file, and upload it into **vManage**. An example of a URL, **`https://<your ADFS FQDN or IP>/FederationMetadata/2007-06/FederationMetadata.xml`**.
3. **Click here** to download SAML metadata, and save into a file. An example of a saved file, 192.168.1.15_saml_metadata.xml.
4. Open the file with an XML editor, and check that the following information is available:
   - Entity ID
   - Signing certificate
   - Login URL
   - Logout URL
5. Navigate to **`https://www.samltool.com/format_x509cert.php`**.
6. For **Signing certificate**, copy Signing certificate from “metadata” [everything between <ds:X509Certificate> and </ds:X509Certificate>].
7. Navigate to **`www.samltool.com`** page, click **X.509 CERTS** > **Format X.509 Certificate**, and paste the copied content
8. Save the output (“X.509 cert with header”) into a text file “Signing.cer”. Remember to remove the last empty line.

Add ADFS Relying Party Trust Manually

To add ADFS relying party trust manually:

1. Launch **AD FS 2.0 Management**.
2. Navigate to **Trust Relationships** > **Relying Party Trusts**.
3. Click **Action** > **Add Relying Party Trust**.
4. Click **Start**.
5. Select **Enter data about the relying party manually**, and click **Next**.
6. Enter **Display name** and **Notes**, and then click **Next**.
7. Select **AD FS 2.0 profile**, and click **Next**.
8. Click **Next** to skip **Configure Certificate** page.
9. Click **Enable support for the SAML 2.0 Webs So protocol**.
10. Open a text editor, and open **`10.10.10.15_saml_metadata.xml`** file.

11. Copy the vale of the **Location** attribute for **AssertionConsumerService**, and paste it into the **Relying party SAML 2.0 SSO service URL** text box.
12. Click **Next**.
13. Copy the value of **entityID** attribute, and paste it into the **Relying party trust identifiers** text box.
14. Click **Add**, and click **Next**.
15. Click **Next** to skip **Configure Multi-factor Authentication Now** section.
16. Select **Permit all users to access this relying party**, and click **Next**.
17. Click **Next** to skip **Ready to Add Trust** section.
18. Click **Close**.
19. Open **Edit Claim Rules** window, and add the following four new custom rules in this order:
    - `@RuleName = "sAMAccountName as Username" c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"]=> issue(store = "Active Directory", types = ("Username"), query = ";sAMAccountName;{0}", param = c.Value);`
    - `@RuleName = "sAMAccountName as NameID" c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => issue(store = "Active Directory", types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"), query = ";sAMAccountName;{0}", param = c.Value);`
    - `@RuleName = "Get User Groups and save in temp/variable" c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => add(store = "Active Directory", types = ("http://temp/variable1"), query = ";tokenGroups;{0}", param = c.Value);`
    - `@RuleName = "Parse temp/variable1 and Send Groups Membership" c:[Type == "http://temp/variable1", Value =~ "(?i)^SSO-"]=> issue(Type = "Groups", Value = RegExReplace(c.Value, "SSO-", ""));`
20. Open the **Edit Claim Rules** window, and verify that the rules display in the **Assurance Transform Rules** tab.
21. Click **Finish**.
22. Open **Properties** window of the newly created **Relying Party Trust**, and click the **Signature** tab.
23. Click **Add**, and add the **Signing.cer** created in **Step 6**.
24. In the **Active Directory**, click the **General** tab, and enter the following two security groups in the **Group name** text box:

    `SSO-Netadmin`

    `SSO-Operator`

    Note If you use different naming convention for the two security groups, then you have to modify the **Regu** expression value for `(?i)^SSO-` mentioned in **Step 19**.

Note Any active directory user who is NOT a member of these two groups, will only have **Basic** access to vManag

CHAPTER 7

Security CLI Reference

CLI commands for configuring and monitoring security.

Security Configuration Commands

Use the following commands to configure security parameters:

```
security
 control
   protocol (dtls | tls)
   tls-port number
 ipsec
   authentication-type type
   rekey seconds
   replay-window number
 vpn vpn-id
   interface ipsecnumber
     access-list acl-name
     block-non-source-ip
     clear-dont-fragment
     dead-peer-detection  interval seconds retries number
     description text
   ike
     authentication-type type
       local-id id
       pre-shared-secret password
       remote-id id
    cipher-suite suite
    group number
    mode mode
    rekey seconds
    version number
  ip address ipv4-prefix/length
ipsec
 cipher-suite suite
 perfect-forward-secrecy pfs-setting
 rekey seconds
 replay-window number
mtu bytes
policer policer-name
rewrite-rule rule-name
[no] shutdown
tcp-mss-adjust bytes
tunnel-destination (dns-name | ipv4-address)
(tunnel-source ip-address |  tunnel-source-interface interface-name)
```

Security Monitoring Commands

- **show control connections**
- **show security-info**